# Oracle® Fusion Middleware

# Enterprise Deployment Guide for Oracle Identity and Access Management in a Kubernetes Cluster

12.2.1.4
F35764-18
January 2025

ORACLE

Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity and Access Management in a Kubernetes Cluster, 12.2.1.4

F35764-18



Primary Author: Oracle Corporation

# Contents

## 3 About a Typical Enterprise Deployment



## 4 About the IAM Enterprise Deployment

## 5 Disaster Recovery

## 9 Preparing for an On-Premises Enterprise Deployment

## 10 Preparing the Oracle Cloud Infrastructure for an Enterprise Deployment

## 12 Installing the Monitoring and Visualization Software



## 13 Installing and Configuring Ingress Controller

## 14 Installing and Configuring Oracle Unified Directory

## 15 Installing and Configuring Oracle Unified Directory Services Manager



## 16 Installing and Configuring WebLogic Kubernetes Operator



## 17 Installing and Configuring Oracle HTTP Server

## 18 Configuring Oracle Access Manager Using WDT

## 19 Configuring Oracle Identity Governance Using WDT

# 20 Installing and Configuring Oracle Identity Role Intelligence

## 21 Installing and Configuring Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

## 22 Configuring Disaster Recovery

# 23 Common Configuration and Management Tasks for an Enterprise Deployment

## 24 Using Whole Server Migration and Service Migration in an Enterprise Deployment



## 25 Centralized Monitoring Using Grafana and Prometheus

## 26 Centralized Log File Monitoring Using Elasticsearch and Kibana



## 27 Scaling Procedures for an Enterprise Deployment



## 28 Configuring Single Sign-On for an Enterprise Deployment

## 29 Sanity Checks

## 30 Troubleshooting



## A Sample of the Schema Extension File and the Seeding File

## B Using an OCI Container Registry



## C Automating the Identity and Access Management Enterprise Deployment

## D Cleaning Up After a Failed Installation



## E Automating the OCI Infrastructure Creation for the Identity and Access Management Kubernetes Cluster

## F Automating the Disaster Recovery Setup

# Preface

This guide explains how to install, configure, and manage a highly available Oracle Fusion Middleware enterprise deployment.



## Audience

In general, this document is intended for administrators of Oracle Fusion Middleware, who are assigned the task of installing and configuring Oracle Fusion Middleware software for production deployments.

Specific tasks can also be assigned to specialized administrators, such as database administrators (DBAs) and network administrators, where applicable.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `https://www.oracle.com/corporate/accessibility/`.

**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `https://support.oracle.com/portal/` or visit `Oracle Accessibility Learning and Support` if you are hearing impaired.

## Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

## Conventions

The following text conventions are used in this document:

| Convention | Meaning |
| --- | --- |
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

**Note:**

This guide focuses on the implementation of the enterprise deployment reference topology on Oracle Linux systems.

The topology can be implemented on any certified, supported operating system, but the examples in this guide typically show the commands and configuration steps as they should be performed using the bash shell on Oracle Linux.

# Revision History

This section provides the revision history for the guide (only major refreshes/updates are mentioned in this table).

| Date | Description |
|---|---|
| July 2021 | Initial release. |
| September 2021 | Added the following appendices:<br>• Automating the Identity and Access Management Enterprise deployment.<br>• Using an OCI container registry. |
| December 2021 | This release adds support for:<br>• Using full container image names.<br>• Using a container registry.<br>• Automatic pulling of container images from a container registry.<br>• WebLogic Operator 3.3.<br>• Oracle Identity Role Intelligence.<br>• CRI-O and Docker-based container runtimes. |
| May 2022 | This release adds support for:<br>• Oracle container registry.<br>• Ingress controller.<br>• Oracle Advanced Authentication and Risk Management.<br>• OUD Cron job.<br>• Running the OIG Bulk Load Utility from a container. |
| July 2022 | This release adds support for:<br>• Oracle Unified Directory stateful sets.<br>• Centralized log file management using Elasticsearch and Kibana. |
| November 2022 | This release adds support for:<br>• OUD stateful sets.<br>• Monitoring using Prometheus. |
| December 2022 | This release includes minor bug fixes. |
| April 2023 | This release adds support for:<br>• Oracle WebLogic Server Operator 4.<br>• Automation of OCI setup and other enhancements. |
| June 2023 | This release includes the following updates:<br>• Change in the default OAM security mode to `open`.<br>• Minor bug fixes. |
| October 2023 | This release includes support for:<br>• Disaster Recovery (DR).<br>• WebLogic Kubernetes Operator 4.x.x.<br>• Resource reservation for products. |
| January 2024 | This release includes support for:<br>• Miscellaneous enhancements<br>• Bug fixes |

| Date | Description |
| --- | --- |
| May 2024 | This release includes support for:<br>• Oracle Universal Authenticator<br>• Creation of WebLogic domains using WebLogic Deployment Tools (WDT) |
| January 2025 | This release includes support for:<br>• OAA installation flow changes |

# Part I
# About an Enterprise Deployment

You should understand the concept and general characteristics of a typical enterprise deployment before you configure the Oracle Identity and Access Management enterprise deployment topology.

This part of the guide contains the following chapters:

- About the Enterprise Deployment
  The detailed, validated instructions provided in this guide will help you plan, prepare, install, and configure a multi-host, secure, highly available production topology for selected Oracle Fusion Middleware products.
- About the Kubernetes Deployment
  Containers offer an excellent mechanism to bundle and run applications. In a production environment, you have to manage the containers that run the applications and ensure there is no downtime. For example, if a container goes down, another container has to start immediately. Kubernetes simplifies container management.
- About a Typical Enterprise Deployment
  The illustration of a typical enterprise deployment topology helps you understand the components of the topology. The topology consists of a web tier, application tier, and data tier.
- About the IAM Enterprise Deployment
  Learn about deploying Oracle Identity and Access Management topologies on commodity hardware. These topologies represent specific reference implementations of the concepts described in About a Typical Enterprise Deployment.
- Disaster Recovery
  Each product has a different disaster recovery strategy. This chapter explains the disaster recovery processes for each product.

1

# About the Enterprise Deployment

The detailed, validated instructions provided in this guide will help you plan, prepare, install, and configure a multi-host, secure, highly available production topology for selected Oracle Fusion Middleware products.

> **Note:**
>
> This document cites the use of product options that may be separately licensed. Ensure that you have the appropriate licenses for the options you want to use. Contact your sales representative if in doubt.

This chapter includes the following topics:

- About the Enterprise Deployment Guide
  An Enterprise Deployment Guide provides a comprehensive, scalable example for installing, configuring, and maintaining a secure, highly available, production-quality deployment of selected Oracle Fusion Middleware products. The resulting environment is known as an **enterprise deployment topology**.
- When to Use the Enterprise Deployment Guide
  This guide describes one of the three primary installation and configuration options for Oracle Fusion Middleware. Use this guide to help you plan, prepare, install, and configure a multi-host, secure, highly available, production topology for selected Oracle Fusion Middleware products.

## About the Enterprise Deployment Guide

An Enterprise Deployment Guide provides a comprehensive, scalable example for installing, configuring, and maintaining a secure, highly available, production-quality deployment of selected Oracle Fusion Middleware products. The resulting environment is known as an **enterprise deployment topology**.

For example, the enterprise deployment topology introduces key concepts and best practices that you can use to implement a similar Oracle Fusion Middleware environment for your organization.

Each Enterprise Deployment Guide provides detailed, validated instructions for implementing the reference topology. Along the way, the guide also offers links to supporting documentation that explains concepts, reference material, and additional options for an Oracle Fusion Middleware enterprise deployment.

Note that the enterprise deployment topologies described in the enterprise deployment guides cannot meet the exact requirements of all Oracle customers. In some cases, you can consider alternatives to specific procedures in this guide, depending on whether the variations to the topology are documented and supported by Oracle.

Oracle recommends customers use the Enterprise Deployment Guides as a first option for deployment. If variations are required, then those variations should be verified by reviewing the related Oracle documentation or by working with Oracle Support.

# When to Use the Enterprise Deployment Guide

This guide describes one of the three primary installation and configuration options for Oracle Fusion Middleware. Use this guide to help you plan, prepare, install, and configure a multi-host, secure, highly available, production topology for selected Oracle Fusion Middleware products.

Alternatively, you can use the other primary installation and configuration options.

Use the instructions in one of the product-specific installation guides to install and configure a **standard installation topology** for a selected set of Oracle Fusion Middleware products.

A standard installation topology can be installed on a single host for evaluation purposes, but it can also serve as a starting point for scaling out to a more complex production environment.

For Oracle Identity and Access Management, see Installing and Configuring Oracle Identity and Access Management

Review *Planning an Installation of Oracle Fusion Middleware*, which provides additional information to help you prepare for any Oracle Fusion Middleware installation.

2

# About the Kubernetes Deployment

Containers offer an excellent mechanism to bundle and run applications. In a production environment, you have to manage the containers that run the applications and ensure there is no downtime. For example, if a container goes down, another container has to start immediately. Kubernetes simplifies container management.

This chapter includes the following topics:



## What is Kubernetes?

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation.

Kubernetes sits on top of a container platform such as Crio or Docker. Kubernetes provides a mechanism which enables container images to be deployed to a cluster of hosts. When you deploy a container through Kubernetes, Kubernetes deploys that container on one of its worker nodes. The placement mechanism is transparent to the user.

Kubernetes provides:

- **Service Discovery and Load Balancing**: Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes balances the load and distributes the network traffic so that the deployment remains stable.
- **Storage Orchestration**: Kubernetes enables you to automatically mount a storage system of your choice, such as local storages, NAS storages, public cloud providers, and more.
- **Automated Rollouts and Rollbacks**: You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers, and adopt all their resources to the new container.
- **Automatic Bin Packing**: If you provide Kubernetes with a cluster of nodes that it can use to run containerized tasks, and indicate the CPU and memory (RAM) each container

needs, Kubernetes can fit containers onto the nodes to make the best use of the available resources.

- **Self-healing**: Kubernetes restarts containers that fail, replaces containers, kills containers that do not respond to your user-defined health check, and does not advertise them to clients until they are ready to serve.
- **Secret and Configuration Management**: Kubernetes lets you store and manage sensitive information such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

Kubernetes strength comes in providing a scalable platform for your mid tiers. Consider Kubernetes as a wrapper for your application tier. That is to say, when designing the architecture, the same network security considerations should be applied as in a traditional deployment. You will still keep the web tier in a separate DMZ outside of the network hosting the Kubernetes cluster, controlling traffic using firewall or security lists. The same applies for the database. The database would exist in a separate network outside of the cluster with a firewall or security lists controlling access from the Kubernetes layer.

When deploying Kubernetes, Oracle highly recommends that you use the traditional recommendations of keeping different workloads in separate Kubernetes clusters. For example, it is not a good practice to mix development and production workloads in the same Kubernetes cluster.

# Requirements for Kubernetes

To deploy Oracle Identity and Access Management on Kubernetes, your environment should meet certain criteria.

The criteria are as follows:

- Kubernetes Cluster 1.19.7 or above: The cluster can be either a standalone Kubernetes cluster similar to the one provided by Oracle Cloud Native Environment or a Cloud Managed Kubernetes Environment such as Oracle Container Engine (OKE). There should be sufficient Kubernetes worker nodes with sufficient capacity for the deployment. You must have multiple worker nodes to ensure no single point of failure. If you are using a standalone Kubernetes environment, the Kubernetes control plane must be configured for high availability.
- An administrative host from which to deploy the products: This host could be a Kubernetes Control host, a Kubernetes Worker host, or an independent host. This host must have `kubectl` deployed using the same version as your cluster, and `helm`.

This guide provides instructions that are validated using the following product versions:

- Oracle Containers for Cloud (OKE) 1.28.2
- Oracle Cloud Native Environment 1.8
- Helm 3.13.1

For information about further deployment considerations, see Technical Brief - Deployment and DevOps Considerations of IAM Containerized Services on Cloud Native Infrastructure.

# About the Kubernetes Architecture

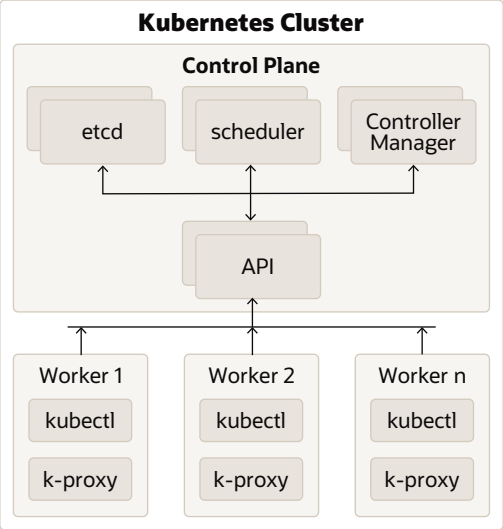
A Kubernetes host consists of a control plane and worker nodes.

**Control Plane**: A control plane is responsible for managing the Kubernetes components and deploying applications. In an enterprise deployment, you need to ensure that the Kubernetes control plane is highly available so that the failure of a control plane host does not fail the Kubernetes cluster.

**Worker Nodes**: Worker nodes which are where the containers are deployed.

> **Note:**
>
> An individual host can be both a control plane host and a worker host.

**Figure 2-1 An Illustration of the Kubernetes Cluster**




**Description of Components**:

- **Control Plane**: The control plane comprises the following:
  - kube-api server: The API server is a component of the control plane that exposes the Kubernetes APIs.
  - etcd: It is used to store the Kubernetes backing store and all the cluster data.
  - Scheduler: The scheduler is responsible for the placement of containers on the worker nodes. It takes into account resource requirements, hardware and software policy constraints, affinity specifications, and data affinity.
  - Control Manager: It is responsible for running the controller processes. Controller processes consist of:
    - Node Controller
    - Route Controller
    - Service Controller

  The control plane consists of three nodes where the Kubernetes API server is deployed, front ended by an LBR.
- **Worker Node Components**: The worker nodes include the following components:

- Kubelet: An Agent that runs on each worker node in the cluster. It ensures that the containers are running in a pod.
- Kube Proxy: Kube proxy is a network proxy that runs on each node of the cluster. It maintains network rules, which enable inter pod communications as well as communications outside of the cluster.
- Add-ons: Add-ons extend the cluster further, providing such services as:
  * DNS
  * Web UI Dashboard
  * Container Resource Monitoring
  * Logging
- A load balancer is often placed in front of the the worker nodes to make it easier to direct traffic to any worker node. A load balancer also simplifies scaling up and scaling out and thereby reducing the need for application configuration changes. The configuration changes can be more difficult if you are using individual NodePort Services because you may have to create a pathway for each NodePort Service.

# Sizing the Kubernetes Cluster

A Kubernetes cluster must have a minimum of two worker nodes and a highly available control plane to ensure that you maintain a high availability solution.

You can choose to have a large number of small capacity worker nodes or a small number of high capacity worker nodes. If you work with two worker nodes, and one becomes unavailable, you lose 50pct of your system capacity. This setup introduce a single point of failure (the surviving worker node), until the surviving node replaces the failed node. Having a higher number of worker nodes means that even if one worker node fails, more than one remains, thus removing that single point of failure. However, the remaining worker nodes should have enough capacity to run pods that initially ran on the failed worker node.

In sizing the cluster, you need to work out the resource requirements of every pod that will run on the cluster, and then add a 20% overhead for the Kubernetes services. Adding worker nodes as your capacity needs expand is a relatively simple process. If your cluster runs several applications, then the capacity must be such that it can run all the pods associated with all those applications in a highly available manner. That is, if your total application requirement is 20CPUs and 100GB of memory, then you would need at least two worker nodes, each with 10CPUs and 50GB of memory plus 20pct, so 12CPUs and 60GB of memory for each worker node.

Storage requirements depend on the type of storage you are planning to use. Each pod will require a small measure of block storage to host the image and any local data. In addition, data stored in persistent volumes will require some shared storage, such as NFS.

The following tables provide a reference for the sizing of small, medium, and large systems. The requirements are for each product and do not take into account the Kubernetes overheads:

**Table 2-1 Sizing for Oracle Unified Directory**

| System Size | Number of Users | CPUs | Memory |
|---|---|---|---|
| Development | - | 0.5 | 2GB |
| Small | 5000 | 4 | 16GB |
| Medium | 50000 | 8 | 32GB |

**Table 2-1 (Cont.) Sizing for Oracle Unified Directory**

| System Size | Number of Users | CPUs | Memory |
|---|---|---|---|
| Large | 2 Million | 16 | 64GB+ |

You require a minimum of two pods to achieve high availability. See Deep Dive into Oracle Unified Directory 12.2.1.4.0 Performance.

**Table 2-2 Sizing for Oracle Access Manager**

| System Size | Number of Users | CPUs | Memory |
|---|---|---|---|
| Development | - | 1 | 2GB |
| Small | 16000 | 4 | 60GB |
| Medium | 24000 | 8 | 60GB |
| Large | 32000+ | 16 | 120GB |

You require a minimum of two pods to achieve high availability. See Deep Dive into Oracle Access Management 12.2.1.4.0 Performance on Oracle Container Engine for Kubernetes.

**Table 2-3 Sizing for Oracle Identity Governance**

| System Size | Number of Users | CPUs | Memory |
|---|---|---|---|
| Development | - | 2 | 4GB |
| Small | 50000 | 4 | 5GB |
| Medium | 150000 | 8 | 8GB |
| Large | 5000 | 20 | 12GB |

You require a minimum of two pods for high availability. Large systems should have a minimum of five pods. See Oracle Identity Governance Sizing Guide.

**Table 2-4 Sizing for Oracle Advanced Authentication**

| System Size | Number of Users | CPUs | Memory |
|---|---|---|---|
| Development | - | 1 | 2GB |
| Small | 16000 | 4 | 60GB |
| Medium | 24000 | 8 | 60GB |
| Large | 32000+ | 16 | 120 |

You require a minimum of two pods to achieve high availability.

**Table 2-5 Sizing for Oracle Identity Role Intelligence**

| Parameter | Description | Small Scale | Medium Scale | Large Scale |
|---|---|---|---|---|
| `executorInstances` | Specify the number of executor pods. | 3 | 5 | 7 |
| `driverRequestCores` | Specify the CPU request for the driver pod. | 2 | 3 | 4 |

**Table 2-5 (Cont.) Sizing for Oracle Identity Role Intelligence**

| Parameter | Description | Small Scale | Medium Scale | Large Scale |
|---|---|---|---|---|
| `driverLimitCores` | Specify the hard CPU limit of the driver pod. | 2 | 3 | 4 |
| `executorRequestCore` | Specify the CPU request for each executor pod. | 2 | 3 | 4 |
| `executorLimitCore` | Specify the hard CPU limit of each executor pod. | 2 | 3 | 4 |
| `driverMemory` | Specify the hard memory limit of the driver pod. | 2GB | 3GB | 4GB |
| `executorMemory` | Specify the hard memory limit of each executor pod. | 2GB | 3GB | 4GB |

You require a minimum of two pods to achieve high availability. See Tuning Performance in *Administering Oracle Identity Role Intelligence*.

# Key Components Used by an Oracle Enterprise Deployment

An Oracle enterprise deployment uses the Kubernetes components such as pods, Kubernetes services, and DNS.

This section describes each of these Kubernetes components used by an Oracle enterprise deployment.

- Container Image
- Pods
- Pod Scheduling
- Persistent Volumes
- Kubernetes Services
- Ingress Controller
- Domain Name System
- Namespaces
- Other Products

## Container Image

A container image is an unchangeable, static file that includes executable code. When deployed into Kubernetes, it is the container image that is used to create a pod. The image contains the system libraries, system tools, and Oracle binaries required to run in Kubernetes. The image shares the OS kernel of its host machine.

A container image is compiled from file system layers built onto a parent or base image. These layers encourage the reuse of various components. So, there is no need to create everything from scratch for every project.

A pod is based on a container image. This container image is read-only. Each pod has its own instance of a container image.

A container image contains all the software and libraries required to run the product. It does not require the entire operating system. Many container images do not include standard operating utilities such as the vi editor or ping.

When you upgrade a pod, you are actually instructing the pod to use a different container image. For example, if the container image for Oracle Unified Directory is based on the July 23 patch, then to upgrade the pod to use the October 23 bundle patch, you have to tell the pod to use the October image and restart the pod.

Oracle containers are built using a specific user and group ID. Oracle supplies its container images using the user ID 1000 and group ID 1000. To enable writing to file systems or persistent volumes, you should grant the write access to this user ID. Oracle supplies all container images using this user and group ID.

If your organization already uses this user or group ID, you should reconfigure the image to use different IDs. This feature is outside the scope of this document.

## Pods

A pod is a group of one or more containers, with shared storage/network resources, and a specification for how to run the containers. A pod's contents are always co-located and co-scheduled, and run in a shared context. A pod models an application-specific logical host that contains one or more application containers which are relatively tightly coupled.

In an Oracle enterprise deployment, each WebLogic Server runs in a different pod. Each pod is able to communicate with other pods inside the Kubernetes cluster.

If a node becomes unavailable, Kubernetes (versions 1.5 and later) does not delete the pods automatically. Pods that run on an unreachable node attain the 'Terminating' or 'Unknown' state after a timeout. Pods may also attain these states when a user attempts to delete a pod on an unreachable node gracefully. You can remove a pod in such a state from the apiserver in one of the following ways:

- You or the Node Controller deletes the node object.
- The kubelet on the unresponsive node starts responding, terminates the pod, and removes the entry from the apiserver.
- You force delete the pod.

Oracle recommends the best practice of using the first or the second approach. If a node is confirmed to be dead (for example: permanently disconnected from the network, powered down, and so on), delete the node object. If the node suffers from a network partition, try to resolve the issue or wait for the partition to heal. When the partition heals, the kubelet completes the deletion of the pod and frees up its name in the apiserver.

Typically, the system completes the deletion if the pod is no longer running on a node or an administrator has deleted it. You may override this by force deleting the pod.

## Pod Scheduling

By default, Kubernetes will schedule a pod to run on any worker node that has sufficient capacity to run that pod. In some situations, it is desirable that scheduling occurs on a subset of the worker nodes available. This type of scheduling can be achieved by using Kubernetes labels.

## Persistent Volumes

When a pod is created, it is based on a container image. A container image is supplied by Oracle for the products you are deploying. When a pod gets created, a runtime environment is created based upon that image. That environment is refreshed with the container image every time the pod is restarted. This means that any changes you make inside a runtime environment are lost whenever the container gets restarted.

A persistent volume is an area of disk, usually provided by NFS that is available to the pod but not part of the image itself. This means that the data you want to keep, for example the WebLogic domain configuration, is still available after you restart a pod, that is to say, that the data is persistent.

There are two ways of mounting a persistent volume (PV) to a pod.

1. Mount the PV to the pod directly, so that wherever the pod starts in the cluster the PV is available to it. The upside to this approach is that a pod can be started anywhere without extra configuration. The downside to this approach is that there is one NFS volume which is mounted to the pod. If the NFS volume becomes corrupted, you will have to either revert to a backup or have to failover to a disaster recovery site.
2. Mount the PV to the worker node and have the pod interact with it as if it was a local file system. The advantages of this approach are that you can have different NFS volumes mounted to different worker nodes, providing built-in redundancy. The disadvantages of this approach are:
   - Increased management overhead.
   - Pods have to be restricted to nodes that use a specific version of the file system. For example, all odd numbered pods use odd numbered worker nodes mounted to file system 1, and all even numbered pods use even numbered worker nodes mounted to file system 2.
   - File systems have to be mounted to every worker node on which a pod may be started. This requirement is not an issue in a small cluster, unlike in a large cluster.
   - Worker nodes become linked to the application. When a worker node undergoes maintenance, you need to ensure that file systems and appropriate labels are restored.

   You will need to set up a process to ensure that the contents of the NFS volumes are kept in sync by using something such as the `rsync` cron job.

   If maximum redundancy and availability is your goal, then you should adopt this solution.

## Kubernetes Services

Kubernetes services expose the processes running in the pods regardless of the number of pods that are running. For example, a cluster of WebLogic Managed Servers, each running in different pods will have a service associated with them. This service will redirect your request to the individual pods in the cluster. If you can interact with a pod in different ways, then you would require multiple services.

For example, if you have a cluster of OAM servers where you can interact with them using either HTTP or OAP, then you would create two services: one for HTTP and another for OAP.

Kubernetes services can be internal or external to the cluster. Internal services are of the type `ClusterIP` and external services are of the type `NodePort`.

Some deployments use a proxy in front of the service. This proxy is typically provided by an 'Ingress' load balancer such as **Ngnix**. Ingress allows a level of abstraction to the underlying Kubernetes services.

When using Kubernetes, NodePort Services have a similar result as using Ingress. In the NodePort mode, Ingress allows for consolidated management of these services.

This guide describes how to use Ingress or the native Kubernetes NodePort Services. Oracle recommends that you select one method instead of opting for a mix and match. For demonstration purposes, this guide explains the usage of the Nginx Ingress Controller.

The Kubernetes services use a small port range. Therefore, when a Kubernetes service is created, there will be a port mapping. For instance, if a pod is using port `7001`, then a Kubernetes/Ingress service may use `30701` as its port, mapping port `30701` to `7001` internally. It is worth noting that if you are using individual NodePort Services, then the corresponding Kubernetes service port will be reserved on every worker node in the cluster.

Kubernetes/Ingress services are known to each worker node, regardless of the worker node on which the containers are running. Therefore, a load balancer is often placed in front of the worker node to simplify routing and worker node scalability.

To interact with a service, you have to refer to it using the format: `worker_node_hostname:Service port`. This format is applicable whether you are using individual NodePort Services or a consolidated Ingress node port service.

If you have multiple worker nodes, then you should include multiple worker nodes in your calls to remove single points of failure. You can do this in a number of ways including:

- Load balancer
- Direct proxy calls such as using `WebLogicCluster` directives for OHS
- DNS CNames

Oracle enterprise deployments use direct proxy calls unless otherwise stated.

# Ingress Controller

There are two ways of interacting with your Kubernetes services. You can create an externally facing service for each Kubernetes object you want to access. This type of service is known as the Kubernetes NodePort Service. Alternatively, you can use an ingress service inside the Kubernetes cluster to redirect requests internally.



## About Ingress

Ingress is a proxy server which sits inside the Kubernetes cluster, unlike the NodePort Services which reserve a port per service on every worker node in the cluster. With an ingress service, you can reserve single ports for all HTTP / HTTPS traffic.

An Ingress service works in the same way as the Oracle HTTP Server. It has the concept of virtual hosts and can terminate SSL, if required. There are various implementations of Ingress. However, this guide describes the installation and configuration of NGNIX. The installation will be similar for other Ingress services but the command syntax may be different. Therefore,

when you use a different Ingress, see the appropriate manufacturer documentation for the equivalent commands.

Ingress can proxy HTTP, HTTPS, LDAP, and LDAPS protocols.

Ingress is not mandatory. You can interact with the Kubernetes services through an Oracle HTTP Server using individual NodePort Services. However, if you are using only the Oracle Identity and Access Management microservices such as Oracle Identity Role Intelligence (OIRI) and Oracle Advanced Authentication (OAA), you may want to use an Ingress service rather than an Oracle HTTP Server.

## Deployment Options

Ingress runs inside the Kubernetes cluster. You can configure it in different ways:

- **Load Balancer**: Load balancer provides an external IP address to which you can connect to interact with the Kubernetes services. This feature is undesirable in an enterprise deployment because you do not want to expose the Kubernetes services directly to the internet to maintain maximum security. The most secure mechanism is to route requests to a load balancer, which in turn forwards the requests to an Oracle HTTP Server that resides in a separate demilitarized zone (DMZ). The HTTP Server then passes on the requests to the application tier.
- **NodePort**: In this mode, Ingress acts as a simple load balancer between the Kubernetes services.
  The difference between using an Ingress NodePort Service as opposed to individual node port services is that the Ingress controller reserves one port for each service type it offers. For example, one for all HTTP communications, another for all LDAP communications, and so on. Individual node port services reserve one port for each service and type used in an application.

  For example, an application such as OAM has four services (two for the Administration Server, one for the OAM Managed Servers, and another for the policy Managed Servers).

  - Ingress reserves one NodePort Service regardless of the number of application services.
  - Using individual NodePort Services, OAM reserves four ports.
    This value will be multiplied by the number of applications deployed in the Kubernetes cluster. However, Ingress will continue to use one port.

## Secure Socket Layer

You can configure Ingress to handle only HTTP requests. However, this guides explains the setting up of an Ingress controller which will handle both HTTP and HTTPS requests. If you are using a traditional deployment and Oracle HTTP Server, it is unlikely that you will require HTTPS connections. However, if you are using microservices directly, then you will most likely use HTTPS connections.

## Scope of Ingress

There is usually one Ingress controller per Kubernetes cluster. This controller manages all namespaces within the cluster. It is possible to use multiple Ingresses in the same cluster with different controllers managing different sets of namespaces. Configuring multiple controllers is outside the scope of this document. However, if you want to use multiple Ingresses, see Running Multiple Ingress Controllers.

## Domain Name System

Every service defined in the cluster (including the DNS server itself) is assigned a DNS name. By default, a client pod's DNS search list includes the pod's own namespace and the cluster's default domain.

The following types of DNS records are created for a Kubernetes cluster:

- **Services**
  Record Type: A or AAAA record

  Name format: `my-svc.namespace.svc.cluster-example.com`
- **Pods**
  Record Type: A or AAAA record

  Name format: `podname.namespace.pod.cluster-example.com`

Kubernetes uses a built-in DNS server called 'CoreDNS' which is used for the internal name resolution.

External name resolution (names used outside of the cluster, for example: `login.example.com`) may not possible inside the Kubernetes cluster. If you encounter this issue, you can use one of the following options:

- **Option 1** - Add a secondary DNS server to CoreDNS for the company domain.
- **Option 2** - Add individual host entries to CoreDNS for the external hosts. For example: `login.example.com`.

## Namespaces

Namespaces enable you to organize clusters into virtual sub-clusters which are helpful when different teams or projects share a Kubernetes cluster. You can add any number of namespaces within a cluster, each logically separated from others but with the ability to communicate with each other.

The enterprise deployment uses different namespaces for each product to keep all the deployed artifacts together. For example, all the Oracle Access Manager components are deployed into a namespace for Oracle Access Manager.

The deployment procedure explained in this guide uses the following namespaces:

**Table 2-6 Namespaces Used in this Guide**

| Name of the Namespace | Description |
|---|---|
| INGRESSNS | Namespace for the Ingress Controller. |
| ELKNS | Namespace for Elasticsearch and Kibana. |
| MONITORING | Namespace for Prometheus and Grafana. |
| OUDNS | Namespace for Oracle Unified Directory (OUD). |
| OAMNS | Namespace for Oracle Access Manager (OAM). |
| OIGNS | Namespace for Oracle Identity Governance (OIG). |
| OIRINS | Namespace for Oracle Identity Role Intelligence (OIRI). |
| DINGNS | Namespace for Oracle Identity Role Intelligence Data Ingester. |

**Table 2-6 (Cont.) Namespaces Used in this Guide**

| Name of the Namespace | Description |
|---|---|
| OPNS | Namespace for Oracle WebLogic Operator. |
| OAANS | Namespace for Oracle Advanced Authentication (OAA). |

# Other Products

In addition to Kubernetes, you can use other products to enhance the Kubernetes experience. These products include:

- **Elasticsearch:** Elasticsearch is a distributed, free, and open search and analytics engine for all types of data. Elasticsearch is the central component of the Elastic Stack, a set of free and open tools for data ingestion, enrichment, storage, analysis, and visualization. Commonly referred to as the ELK Stack (after Elasticsearch, Logstash, and Kibana). Elastic Search is used for:
  - Logging and log analytics
  - Infrastructure metrics and container monitoring
  - Application performance monitoring
  - Infrastructure metrics and container monitoring
- **Kibana:** Kibana is a data visualization and management tool for Elasticsearch. Kibana provides real-time histograms, line graphs, pie charts, and maps. Kibana also includes advanced applications such as Canvas, which allows you to create custom dynamic infographics based on their data; and Elastic Maps for visualizing geospatial data.
- **Logstash:** Logstash is used in conjunction with Filebeat to scrape log files and to place them into a format that Elasticsearch understands, before transmitting them to Elasticsearch.
- **Grafana:** Grafana is a complete observability stack that allows you to monitor and analyze metrics, logs, and traces. It allows you to query, visualize, alert on, and understand your data no matter where it is stored. Grafana is often used in conjunction with Prometheus.
- **Prometheus:** Prometheus is an open-source systems monitoring and alerting toolkit. Prometheus collects and stores its metrics as time series data, that is, metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

3

# About a Typical Enterprise Deployment

The illustration of a typical enterprise deployment topology helps you understand the components of the topology. The topology consists of a web tier, application tier, and data tier.

This chapter includes the following topics:

- Diagram of a Typical Enterprise Deployment
  The illustration of a typical enterprise deployment shows all the components of the deployment, including the web tier, application tier, and data tier. All enterprise deployments are based on these basic principles.
- About the Typical Enterprise Deployment Topology Diagram
  A typical enterprise deployment topology consists of a Hardware Load Balancer (LBR), web tier, an application tier, and data tier. This section provides detailed information on these components.

## Diagram of a Typical Enterprise Deployment

The illustration of a typical enterprise deployment shows all the components of the deployment, including the web tier, application tier, and data tier. All enterprise deployments are based on these basic principles.

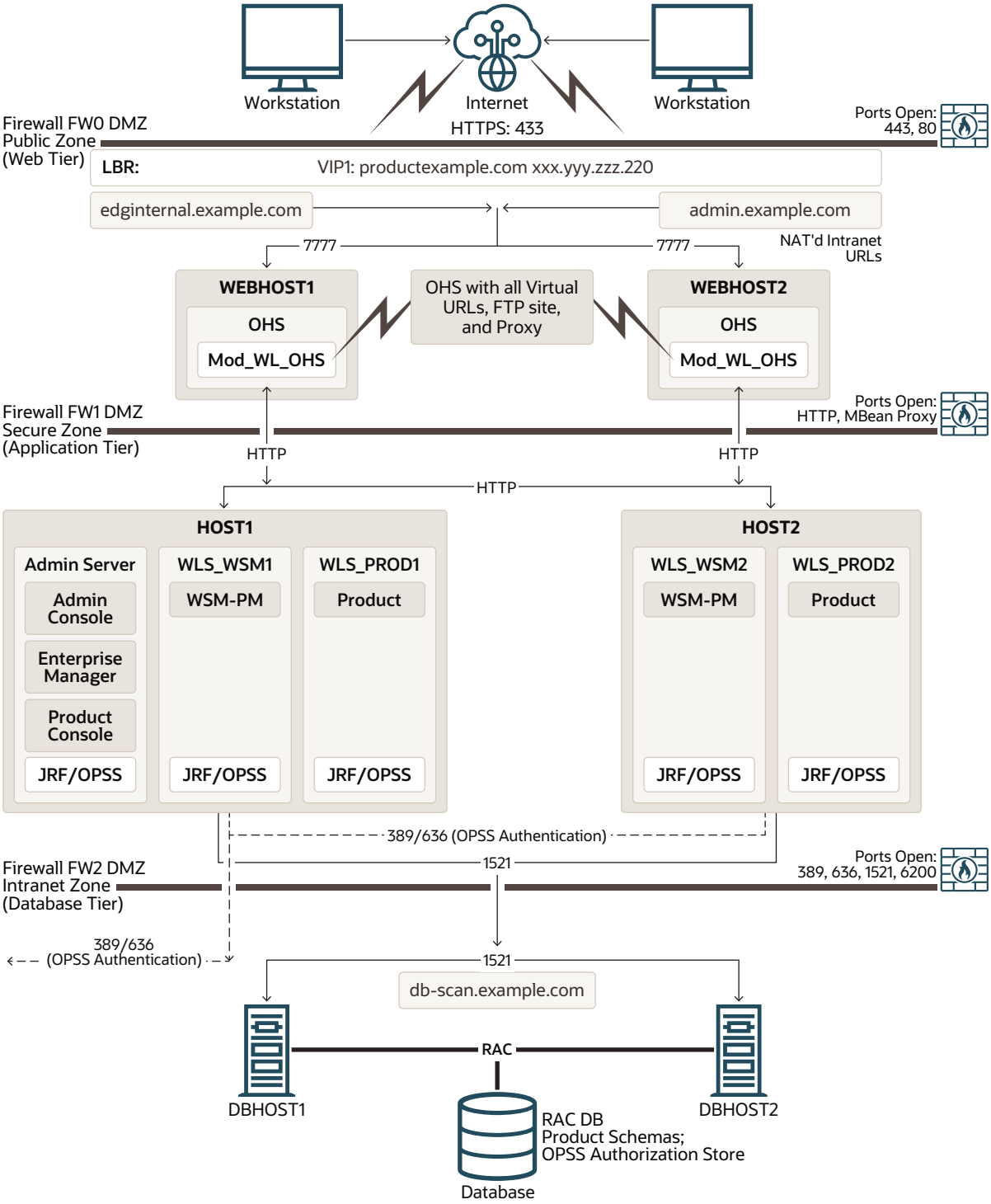
All Oracle Fusion Middleware enterprise deployments are designed to demonstrate the best practices for installing and configuring an Oracle Fusion Middleware production environment.

A best practices approach starts with the basic concept of a multi-tiered deployment and standard communications between the different software tiers.

Figure 3-1 shows a typical enterprise deployment, including the web tier, application tier, and data tier. All enterprise deployments are based on these basic principles.

For a description of each tier and the standard protocols used for communications within a typical Oracle Fusion Middleware enterprise deployment, see About the typical Enterprise Deployment Topology Diagram.

**Figure 3-1 Typical Enterprise Deployment Topology Diagram**




# About the Typical Enterprise Deployment Topology Diagram

A typical enterprise deployment topology consists of a Hardware Load Balancer (LBR), web tier, an application tier, and data tier. This section provides detailed information on these components.

## About Firewalls and Zones of a Typical Enterprise Deployment

The topology is divided into several security zones, which are separated by firewalls:

- The web tier (or DMZ), which is used for the hardware load balancer and Web servers (in this case, Oracle HTTP Server instances) that receive the initial requests from users. This zone is accessible only through a virtual server name that is defined on the load balancer.
- The application tier, which is where the business and application logic resides.
- The data tier, which is not accessible from the Internet and reserved in this topology for the highly available database instances.

The firewalls are configured to allow data to be transferred only through specific communication ports. Those ports (or in some cases, the protocols that need open ports in the firewall) are shown on each firewall line in the diagram.

For example:

- On the firewall protecting the web tier, only the HTTP ports are open: 443 for HTTPS and 80 for HTTP.
- On the firewall protecting an application tier, HTTP ports, and MBean proxy port are open.

  Applications that require external HTTP access can use the Oracle HTTP Server instances as a proxy. Note that this port for outbound communications only and the proxy capabilities on the Oracle HTTP Server must be enabled.
- On the firewall protecting the data tier, the database listener port (typically, 1521) must be open.

  The LDAP ports (typically, 389 and 636) are also required to be open for communication between the authorization provider and the LDAP-based identity store.

  The ONS port (typically, 6200) is also required so that the application tier can receive notifications about workload and events in the Oracle RAC Database. These events are used by the Oracle WebLogic Server connection pools to adjust quickly (creating or destroying connections), depending on the availability and workload on the Oracle RAC database instances.

For a complete list of the ports that you must open for a specific Oracle Fusion Middleware enterprise deployment topology, see the chapter that describes the topology that you want to implement.

## About the Demilitarized Zone

A DMZ or demilitarized zone is a perimeter network that adds an extra layer of security to an organization's internal network from untrusted traffic.

The goal of a DMZ is to allow an organization to access untrusted networks, such as the internet, while ensuring its private network remains secure. A DMZ typically houses services such as Domain Name System (DNS), File Transfer Protocol (FTP), mail, proxy, and web servers in the DMZ.

These servers and resources are isolated and given limited access to the LAN to ensure that they can be accessed using the internet, but the internal LAN remains inaccessible. As a result, a DMZ approach makes it more difficult for a hacker to gain direct access to an organization's data and internal servers using the internet.

## About the Elements of a Typical Enterprise Deployment Topology

The enterprise deployment topology consists of the following high-level elements:

- A hardware load balancer that routes requests from the Internet to the web servers in the web tier. It also routes requests from internal clients or other components that perform internal invocations within the corporate network.
- A web tier, consisting of a hardware load balancer and two or more physical computers that host the web server instances (for high availability).

  The web server instances are configured to authenticate users (through an external identity store and a single sign-on server) and then route the HTTP requests to the Oracle Fusion Middleware products and components that are running in the Application tier.

  The web server instances also host static web content that does not require the application logic to be delivered. Placing such content in the web tier reduces the overhead on the application servers and eliminates unnecessary network activity.
- An application tier, consisting of two or more physical computers that are hosting a cluster of Oracle WebLogic Managed Servers, and the Administration Server for the domain. The Managed Servers are configured to run the various Oracle Fusion Middleware products, such as Oracle SOA Suite, Oracle Service Bus, Oracle WebCenter Content, and Oracle WebCenter Portal, depending on your choice of products in the enterprise deployment.
- A data tier, consisting of two or more physical hosts that are hosting an Oracle RAC Database.

## Receiving Requests Through Hardware Load Balancer

The following topics describe the hardware load balancer and its role in an enterprise deployment.

- Purpose of the Hardware Load Balancer (LBR)
- Summary of the Typical Load Balancer Virtual Server Names
- HTTPS Versus HTTP Requests to the External Virtual Server Name

### Purpose of the Hardware Load Balancer (LBR)

The following topics describe the types of requests handled by the hardware load balancer in an enterprise deployment.

- HTTP Requests From the Internet to the Web Server Instances in the Web Tier
- Specific Internal-Only Communications Between the Components of the Application Tier
- Load Balancer Considerations for Disaster Recovery and Multi-Data Center Topologies

## HTTP Requests From the Internet to the Web Server Instances in the Web Tier

The hardware load balancer balances the load on the web tier by receiving requests to a single virtual host name and then routing each request to one of the web server instances, based on a load balancing algorithm. In this way, the load balancer ensures that no one web server is overloaded with HTTP requests.

For more information about the purpose of specific virtual host names on the hardware load balancer, see Summary of the Typical Load Balancer Virtual Server Names.

Note that in the reference topology, only HTTP requests are routed from the hardware load balancer to the web tier. Secure Socket Layer (SSL) requests are terminated at the load balancer and only HTTP requests are forwarded to the Oracle HTTP Server instances. This guide does not provide instructions for SSL configuration between the load balancer and the Oracle HTTP Server instances or between the web tier and the application tier.

The load balancer provides high availability by ensuring that if one web server goes down, requests are routed to the remaining web servers that are up and running.

Further, in a typical highly available configuration, the hardware load balancers are configured such that a hot standby device is ready to resume service in case a failure occurs in the main load balancing appliance. This is important because for many types of services and systems, the hardware load balancer becomes the unique point of access to make invocations and, as a result, becomes a single point of failure (SPOF) for the whole system if it is not protected.

## Specific Internal-Only Communications Between the Components of the Application Tier

In addition, the hardware load balancer routes specific communications between the Oracle Fusion Middleware components and applications on the application tier. The internal-only requests are also routed through the load balancer by using a unique virtual host name.

## Load Balancer Considerations for Disaster Recovery and Multi-Data Center Topologies

In addition to the load-balancing features for local site traffic as described in the previous topics, many LBR also include features for configuring global load-balancing across multiple sites in DR or active/active MDC topologies.

A global load balancer configuration uses conditional DNS to direct traffic to local load balancers at different sites. A global load balancer for Oracle Fusion Middleware is typically configured for DR or MDC topologies:

- Active/Passive DR: Always send requests to site 1 unless site 1 in unavailable in which case send traffic to site 2.
- Active/Active MDC: Always send requests to both site 1 and site 2, often based on the geographic location of the source request in relation to the physical geographical location of the sites. Active/Active deployments are available only to those applications which support it.

For example:

```
Application entry point:  app.example.com

Site 1 - Local Load Balancer Virtual Host:  site1app.example.com
Site 2 - Local Load Balancer Virtual Host:  site2app.example.com
```

When a request for `app.example.com` is received, the global load balancer would:

- If the topology is active/passive DR:

  Change the IP address of `app.example.com` in DNS to resolve as the IP address of the local load balancer Virtual Host for the active site. For example: `site1app.example.com` (assuming that is the active site).

- If the topology is active/active MDC:

  Change the IP address of `app.example.com` in DNS to resolve as either the IP address of `site1app.example.com` or `site2app.example.com` depending on which site is nearest to the client making the request.

For information on Disaster Recovery, see *Disaster Recovery Guide*.

For more information on Multi-Data Center topologies for various Fusion Middleware products, see the MAA Best Practices for Fusion Middleware page on the Oracle Technology Network website.

## Summary of the Typical Load Balancer Virtual Server Names

In order to balance the load on servers and to provide high availability, the hardware load balancer is configured to recognize a set of virtual server names. By using the naming convention in Figure 3-1, the following virtual server names are recognized by the hardware load balancer in this topology:

- `product.example.com`: This virtual server name is used for all incoming traffic.

  Users enter this URL to access the Oracle Fusion Middleware product that you have deployed and the custom applications that are available on this server. The load balancer then routes these requests (by using a load balancing algorithm) to one of the servers in the web tier. In this way, the single virtual server name can be used to route traffic to multiple servers for load balancing and high availability of the web servers instances.

- `productinternal.example.com`: This virtual server name is for internal communications only.

  The load balancer uses its **Network Address Translation (NAT)** capabilities to route any internal communication from the application tier components that are directed to this URL. This URL is not exposed to external customers or users on the Internet. Each product has specific uses for the internal URL, so in the deployment instructions, the virtual server name is prefixed with the product name.

- `admin.example.com`: This virtual server name is for administrators who need to access the Oracle Enterprise Manager Fusion Middleware Control and Oracle WebLogic Server Administration Console interfaces.

  This URL is known only to internal administrators. It also uses the NAT capabilities of the load balancer to route administrators to the active Administration Server in the domain.

For a complete set of virtual server names that you must define for your topology, see the chapter that describes the product-specific topology.

## HTTPS Versus HTTP Requests to the External Virtual Server Name

Note that when you configure the hardware load balancer, a best practice is to assign the main external URL (for example, `http://myapplication.example.com`) to port 80 and port 443.

Any request on port 80 (non-SSL protocol) should be redirected to port 443 (SSL protocol). Exceptions to this rule include requests from public WSDLs. See Configuring Virtual Hosts on the Hardware Load Balancer.

# About Web Tier

The web tier of the reference topology consists of web servers that receive requests from the load balancer. In a typical enterprise deployment, at least two Oracle HTTP Server instances are configured in the web tier. The following topics provide more detail.

- Benefits of Using a Web Tier to Route Requests
- Alternatives to Using a Web Tier
- Configuration of Oracle HTTP Server in the Web Tier
- About Mod_WL_OHS

## Benefits of Using a Web Tier to Route Requests

A web tier with Oracle HTTP Server is not a requirement for many of the Oracle Fusion Middleware products. You can route traffic directly from the hardware load balancer to the WLS servers in the Application Tier. However, a web tier provides several advantages, which is why it is recommended as part of the reference topology.

- The web tier provides faster fail-over in the event of a WebLogic Server instance failure. The plug-in actively learns about the failed WebLogic Server instance by using the information supplied by its peers. It avoids the failed server until the peers notify the plug-in that it is available. Load balancers are typically more limited and their monitors cause higher overhead.
- The web tier provides DMZ public zone, which is a common requirement in security audits. If a load balancer routes directly to the WebLogic Server, requests move from the load balancer to the application tier in one single HTTP jump, which can cause security concerns.
- The web tier allows the WebLogic Server cluster membership to be reconfigured (new servers added, others removed) without having to change the web server configuration (as long as at least some of the servers in the configured list remain alive).
- Oracle HTTP Server delivers static content more efficiently and faster than WebLogic Server; it also provides the ability to create virtual hosts and proxies via the Oracle HTTP Server configuration files.
- The web tier provides HTTP redirection over and above what the WebLogic Server provides. You can use Oracle HTTP Server as a front end against many different WebLogic Server clusters, and in some cases, control the routing by using content-based routing.
- Oracle HTTP Server provides the ability to integrate single sign-on capabilities into your enterprise deployment. For example, you can later implement single sign-on for the enterprise deployment by using Oracle Access Manager, which is part of the Oracle Identity and Access Management family of products.
- A web tier with Oracle HTTP Server provides support for WebSocket connections deployed within the WebLogic Server.

For more information about Oracle HTTP Server, see Introduction to Oracle HTTP Server in *Administering Oracle HTTP Server*.

## Alternatives to Using a Web Tier

Although a Web tier provides a variety of benefits in an enterprise topology, Oracle also supports routing requests directly from the hardware load balancer to the Managed Servers in the middle tier.

This approach provides the following advantages:

- Lower configuration and processing overhead than using a front-end Oracle HTTP Server Web tier front-end.
- Monitoring at the application level since the LBR can be configured to monitor specific URLs for each Managed Server (something that is not possible with Oracle HTTP Server).

  You can potentially use this load balancer feature to monitor SOA composite application URLs.

> **Note:**
>
> This feature enables routing to the Managed Servers only when all composites are deployed, and you must use the appropriate monitoring software.

This approach provides the following disadvantages:

- The application tier may need to be exposed to the clients, rather than implementing a De-militerized zone (DMZ).
- Applications which require security are generally filtered by using the web-tier. This permits the set up of single-sign on type authentication.

## Configuration of Oracle HTTP Server in the Web Tier

Starting with Oracle Fusion Middleware 12*c*, the Oracle HTTP Server software can be configured in one of two ways: as part of an existing Oracle WebLogic Server domain or in its own standalone domain. Each configuration offers specific benefits.

When you configure Oracle HTTP Server instances as part of an existing WebLogic Server domain, you can manage the Oracle HTTP Server instances, including the wiring of communications between the web servers and the Oracle WebLogic Server Managed Servers by using Oracle Enterprise Manager Fusion Middleware Control. When you configure Oracle HTTP Server in a standalone configuration, you can configure and manage the Oracle HTTP Server instances independently of the application tier domains.

For this enterprise deployment guide, the Oracle HTTP Server instances are configured as separate standalone domains, one on each Web tier host. You can choose to configure the Oracle HTTP Server instances as part of the application tier domain, but this enterprise deployment guide does not provide specific steps to configure the Oracle HTTP Server instances in that manner.

See About Oracle HTTP Server in *Installing and Configuring Oracle HTTP Server*.

## About Mod_WL_OHS

As shown in the diagram, the Oracle HTTP Server instances use the WebLogic Proxy Plug-In (`mod_wl_ohs`) for proxying HTTP requests from Oracle HTTP Server to the Oracle WebLogic Server Managed Servers in the Application tier.

See What are Oracle WebLogic Server Proxy Plug-Ins? in *Using Oracle WebLogic Server Proxy Plug-Ins*.

# About the Application Tier

The application tier consists of two physical host computers, where Oracle WebLogic Server and the Oracle Fusion Middleware products are installed and configured. The application tier computers reside in the secured zone between firewall 1 and firewall 2.

The following topics provide more information:

- Configuration of the Administration Server and Managed Servers Domain Directories
- Using Oracle Web Services Manager in the Application Tier
- Best Practices and Variations on the Configuration of the Clusters and Hosts on the Application Tier
- About the Node Manager Configuration in a Typical Enterprise Deployment
- About Using Unicast for Communications within the Application Tier
- About OPSS and Requests to the Authentication and Authorization Stores

## Configuration of the Administration Server and Managed Servers Domain Directories

Unlike the Managed Servers in the domain, the Administration Server uses an active-passive high availability configuration. This is because only one Administration Server can be running within an Oracle WebLogic Server domain.

In the topology diagrams, the Administration Server on HOST1 is in the active state and the Administration Server on HOST2 is in the passive (inactive) state.

To support the manual fail over of the Administration Server in the event of a system failure, the typical enterprise deployment topology includes:

- A Virtual IP Address (VIP) for the routing of Administration Server requests.
- The configuration of the Administration Server domain directory on a shared storage device.

In the event of a system failure (for example a failure of HOST1), you can manually reassign the Administration Server VIP address to another host in the domain, mount the Administration Server domain directory on the new host, and then start the Administration Server on the new host.

However, unlike the Administration Server, there is no benefit to storing the Managed Servers on shared storage. In fact, there is a potential performance impact when Managed Server configuration data is not stored on the local disk of the host computer.

As a result, in the typical enterprise deployment, after you configure the Administration Server domain on shared storage, a copy of the domain configuration is placed on the local storage device of each host computer, and the Managed Servers are started from this copy of the domain configuration. You create this copy by using the Oracle WebLogic Server pack and unpack utilities.

The resulting configuration consists of separate domain directories on each host: one for the Administration Server (on shared storage) and one for the Managed Servers (on local storage). Depending upon the action required, you must perform configuration tasks from one domain directory or the other.

For more information about structure of the Administration Server domain directory and the Managed Server domain directory, as well as the variables used to reference these directories, see Understanding the Recommended Directory Structure for an Enterprise Deployment.

There is an additional benefit to the multiple domain directory model. It allows you to isolate the Administration Server from the Managed Servers. By default, the primary enterprise deployment topology assumes the Administration Server domain directory is on one of the application tier hosts, but if necessary, you could isolate the Administration Server further by running it from its own host, for example in cases where the Administration Server is consuming high CPU or RAM. Some administrators prefer to configure the Administration Server on a separate, dedicated host, and the multiple domain directory model makes that possible.

## Using Oracle Web Services Manager in the Application Tier

Oracle Web Services Manager (Oracle WSM) provides a policy framework to manage and secure web services in the Enterprise Deployment topology.

In most enterprise deployment topologies, the Oracle Web Services Manager Policy Manager runs on Managed Servers in a separate cluster, where it can be deployed in an active-active highly available configuration.

You can choose to target Oracle Web Services Manager and Fusion Middleware products or applications to the same cluster, as long as you are aware of the implications.

The main reasons for deploying Oracle Web Services Manager on its own Managed Servers is to improve performance and availability isolation. Oracle Web Services Manager often provides policies to custom web services or to other products and components in the domain. In such a case, you do not want the additional Oracle Web Services Manager activity to affect the performance of any applications that are sharing the same Managed Server or cluster as Oracle Web Services Manager.

The eventual process of scaling out or scaling up is also better addressed when the components are isolated. You can scale out or scale up only the Fusion Middleware application Managed Servers where your products are deployed or only the Managed Servers where Oracle Web Services Manager is deployed, without affecting the other product.

## Best Practices and Variations on the Configuration of the Clusters and Hosts on the Application Tier

In a typical enterprise deployment, you configure the Managed Servers in a cluster on two or more hosts in the application tier. For specific Oracle Fusion Middleware products, the enterprise deployment reference topologies demonstrate best practices for the number of Managed Servers, the number of clusters, and the services that are targeted for each cluster.

These best practices consider typical performance, maintenance, and scale-out requirements for each product. The result is the grouping of Managed Servers into an appropriate set of clusters within the domain.

Variations of the enterprise deployment topology allow the targeting of specific products or components to additional clusters or hosts for improved performance and isolation.

For example, you can consider hosting the Administration Server on a separate and smaller host computer, which allows the FMW components and products to be isolated from the Administration Server.

These variations in the topology are supported, but the enterprise deployment reference topology uses the minimum hardware resources while keeping high availability, scalability, and

security in mind. Perform the appropriate resource planning and sizing, based on the system requirements for each type of server and the load that the system must sustain. Based on these decisions, you must adapt the steps to install and configure these variations accordingly from the instructions presented in this guide.

## About the Node Manager Configuration in a Typical Enterprise Deployment

Starting with Oracle Fusion Middleware 12*c*, you can use either a per domain Node Manager or a per host Node Manager. The following sections of this topic provide more information on the impact of the Node Manager configuration on a typical enterprise deployment.

> **Note:**
>
> For general information about these two types of Node Managers, see Overview in *Administering Node Manager for Oracle WebLogic Server*.

### About Using a Per Domain Node Manager Configuration

In a per domain Node Manager configuration—as opposed to a per host Node Manager configuration—you actually start two Node Manager instances on the Administration Server host: one from the Administration Server domain directory and one from the Managed Servers domain directory. In addition, a separate Node Manager instance runs on each of the other hosts in the topology.

The Node Manager that controls the Administration Server uses the listen address of the virtual host name created for the Administration Server. The Node Manager that controls the Managed Servers uses the listen address of the physical host. When the Administration Server fails over to another host, an additional instance of Node Manager is started to control the Administration Server on the failover host.

The key advantages of the per domain configuration are an easier and simpler initial setup of the Node Manager and the ability to set Node Manager properties that are unique to the Administration Server. This last feature was important in previous releases because some features, such as Crash Recovery, applied only to the Administration Server and not to the Managed servers. In the current release, the Oracle SOA Suite products can be configured for Automated Service Migration, rather than Whole Server Migration. This means the Managed Servers, as well as the Administration Server, can take advantage of Crash Recovery, so there is no need to apply different properties to the Administration Server and Managed Server domain directories.

Another advantage is that the per domain Node Manager provides a default SSL configuration for Node Manager-to-Server communication, based on the demo Identity store created for each domain.

### About Using a Per Host Node Manager Configuration

In a per host Node Manager configuration, you start a single Node Manager instance to control the Administration Server and all Managed Servers on a host, even those that reside in different domains. This reduces the footprint and resource utilization on the Administration Server host, especially in those cases where multiple domains coexist on the same computer.

A per host Node Manager configuration allows all Node Managers to use a listen address of ANY, so they listen on all addresses available on the host. This means that when the Administration Server fails over to a new host, no additional configuration is necessary. The per

host configuration allows for simpler maintenance, because you can update and maintain a single Node Manager properties file on each host, rather than multiple Node Manager property files.

The per host Node Manager configuration requires additional configuration steps. If you want SSL for Node Manager-to-Server communication, then you must configure an additional Identity and Trust store, and it also requires using Subject Alternate Names (SAN), because the Node Manager listens on multiple addresses. Note that SSL communications are typically not required for the application tier, because it is protected by two firewalls.

## About Using Unicast for Communications within the Application Tier

Oracle recommends the unicast communication protocol for communication between the Managed Servers and hosts within the Oracle WebLogic Server clusters in an enterprise deployment. Unlike multicast communication, unicast does not require cross-network configuration and it reduces potential network errors that can occur from multicast address conflicts as well.

When you consider using the multicast or unicast protocol for your own deployment, consider the type of network, the number of members in the cluster, and the reliability requirements for cluster membership. Also consider the following features of each protocol.

**Features of unicast in an enterprise deployment:**

- Uses a group leader that every server sends messages directly to. This leader is responsible for retransmitting the message to every other group member and other group leaders, if applicable.
- Works out of the box in most network topologies
- Requires no additional configuration, regardless of the network topology.
- Uses a single missed heartbeat to remove a server from the cluster membership list.

**Features of multicast in an enterprise deployment:**

- Multicast uses a more scalable peer-to-peer model, where a server sends each message directly to the network once and the network makes sure that each cluster member receives the message directly from the network.
- Works out of the box in most modern environments, where the cluster members are in a single subnet.
- Requires additional configuration in the routers and WebLogic Server (that is, Multicast TTL) if the cluster members span more than one subnet.
- Uses three consecutive missed heartbeats to remove a server from the cluster membership list.

Depending on the number of servers in your cluster and on whether the cluster membership is critical for the underlying application (for example, in session-replication intensive applications or clusters with intensive RMI invocations across the cluster), each model may act better.

Consider whether your topology is going to be part of an active-active disaster recovery system or if the cluster is going to traverse multiple subnets. In general, unicast acts better in those cases.

For more information about multicast and unicast communication types, see the following resources:

- Configuring Multicast Messaging for WebLogic Server Clusters in *High Availability Guide*

- One-to-Many Communication Using Unicast in *Administering Clusters for Oracle WebLogic Server*

## About OPSS and Requests to the Authentication and Authorization Stores

Many of the Oracle Fusion Middleware products and components require an Oracle Platform Security Services (OPSS) security store for authentication providers (an identity store), policies, credentials, keystores, and for audit data. As a result, communications must be enabled so the application tier can send requests to and from the security providers.

For authentication, this communication is to an LDAP directory, such as Oracle Unified Directory (OUD), which typically communicates over port 389 or 636. When you configure an Oracle Fusion Middleware domain, the domain is configured by default to use the WebLogic Server Authentication provider. However, for an enterprise deployment, you must use a dedicated, centralized LDAP-compliant authentication provider.

For authorization (and the policy store), the location of the security store varies, depending upon the tier:

- For the application tier, the authorization store is database-based, so frequent connections from the Oracle WebLogic Server Managed Servers to the database are required for the purpose of retrieving the required OPSS data.
- For the web tier, the authorization store is file-based, so connections to the database are not required.

For more information about OPSS security stores, see the following sections of *Securing Applications with Oracle Platform Security Services*:

- Authentication Basics
- The Security Model

## About the Data Tier

In the data tier, an Oracle RAC database runs on the two hosts (DBHOST1 and DBHOST2). The database contains the schemas required by the Oracle Identity and Access Management components and the Oracle Platform Security Services (OPSS) policy store.

You can define multiple services for the different products and components in an enterprise deployment to isolate and prioritize throughput and performance accordingly. In this guide, one database service is used as an example. Furthermore, you can use other high availability database solutions to protect the database:

- Oracle Data Guard: See Introduction to Oracle Data Guard in *Oracle Data Guard Concepts and Administration*.
- Oracle RAC One Node: See Overview of Oracle RAC One Node in *Oracle Real Application Clusters Administration and Deployment Guide*.

These solutions above provide protection for the database beyond the information provided in this guide, which focuses on using an Oracle RAC Database, given the scalability and availability requirements that typically apply to an enterprise deployment.

For more information about using Oracle Databases in a high availability environment, see Database Considerations in *High Availability Guide*.

4

# About the IAM Enterprise Deployment

Learn about deploying Oracle Identity and Access Management topologies on commodity hardware. These topologies represent specific reference implementations of the concepts described in About a Typical Enterprise Deployment.

This chapter includes the following topics:

- About the Primary and Build-Your-Own Enterprise Deployment Topologies
  There are two primary reference topologies for Oracle Identity and Access Management. While the components installed into each topology are the same, the exact Oracle Identity and Access Management topology you install and configure for your organization may vary.
- Using a Demilitarized Zone
- Topology Diagrams for the Deployment of Oracle Identity and Access Management
  There two types of deployment scenarios for Oracle Identity and Access Management. A deployment where you add microservices to an existing enterprise deployment, and the another deployment where you place the entire application tier inside a Kubernetes cluster.
- Topology Diagrams for the Deployment of Oracle Identity and Access Management Components on Kubernetes
- About the Primary Oracle Identity and Access Management Topology Diagrams
  The primary enterprise deployment topology is the main Oracle reference topology for Oracle Identity and Access Management. It provides a solution which is both highly available and scalable.
- Storage Requirements for an Enterprise Deployment
  Preparing the file system for an enterprise deployment involves understanding the requirements for local and shared storage, as well as the terminology that is used to reference important directories and file locations during the installation and configuration of the enterprise topology.
- About Permissions
  When containers are created, the files owned by the container are owned by the user with UID 1000 and group 1000.
- Summary of Microservices and Clusters on the Application Tier
  The application tier not only hosts the WebLogic components of Oracle Identity and Access Management, but it also hosts the Microservice components.
- About the Forgotten Password Functionality
  In Oracle 11*g*, the mechanism to reset passwords was provided by Oracle Identity Governance. In Oracle 12*c*, you have two possibilities - by integrating Oracle Access Management (OAM) and Oracle Identity Governance (OIG) or by using Oracle Access Management.
- Integrating Oracle LDAP, Oracle Access Manager, and Oracle Identity Governance
  Integration of Oracle Identity Manager and Oracle Access Manager with LDAP directories is done by using LDAP Connector.
- Roadmap for Implementing the Primary IAM Suite Topologies
  This roadmap introduces you to the different tasks that need to be performed for implementing the primary IAM suite topologies on Kubernetes. This guide supports

deployments on Oracle Kubernetes Engine (OKE), Oracle Cloud Native Environment, and the on-premises based Kubernetes deployments.

- Building Your Own Oracle Identity and Access Management Topology
  These step-by-step instructions help you configure the two primary enterprise topologies for Oracle Identity and Access Management. However, Oracle recognizes that the requirements of your organization may vary, depending on the specific set of Oracle Fusion Middleware products you purchase and the specific types of applications you deploy.

# About the Primary and Build-Your-Own Enterprise Deployment Topologies

There are two primary reference topologies for Oracle Identity and Access Management. While the components installed into each topology are the same, the exact Oracle Identity and Access Management topology you install and configure for your organization may vary.

This guide provides step-by-step instructions for installing and configuring these topologies.

The components installed into each topology are the same. The difference is that one deployment has all of the Identity Management application tier installed into containers, while the other is a mix of a traditional on-premise deployment with the new microservices deployed into containers.

The topologies depicted in this deployment are the best-practice deployments recommended by Oracle, maximizing security and availability. These topologies have been validated by Oracle.

These are not the only ways of deploying Oracle in a containerized environment; they are Oracle's recommended approach for the deployment of production systems.

To maximize security, Oracle recommends the implementation of a demilitarized zone (DMZ).

# Using a Demilitarized Zone

Oracle strongly recommends the use of a demilitarized zone (DMZ) to ensure maximum security for your organization. See About the Demilitarized Zone .
Kubernetes is at its most powerful when used as an application mid-tier. A DMZ is still required to ensure a strictly controlled access to the Kubernetes network. Outside traffic should always be routed through a DMZ to reduce the likelihood of unscrupulous individuals gaining access to your internal network. You ensure maximum network security by placing the web servers outside the Kubernetes network in a DMZ. Placing the Oracle HTTP servers inside Kubernetes ensures that the Kubernetes network is one step closer than it should be to the outside world.

> **Note:**
>
> For an enterprise deployment, Oracle strongly recommends that the web tier is **NOT** placed inside the Kubernetes cluster containing your business applications, but in a dedicated demilitarized zone.

You can create a separate Kubernetes cluster in the demilitarized zone for your web tier. However, it results in unnecessary overhead because you will require a highly available Kubernetes control plane and multiple Kubernetes worker nodes.

This guide does not cover the scenario of including the Oracle web tier inside Kubernetes because Oracle does not recommend this approach.

# Topology Diagrams for the Deployment of Oracle Identity and Access Management

There two types of deployment scenarios for Oracle Identity and Access Management. A deployment where you add microservices to an existing enterprise deployment, and the another deployment where you place the entire application tier inside a Kubernetes cluster.

- Topology Diagram for a Traditional IAM Deployment with Additional Microservices
- Topology Diagram for the Deployment of Oracle Identity and Access Management on Kubernetes
  This is a deployment of Identity and Access Management where all components of Identity and Access Management, including Microservices, are deployed into a Kubernetes cluster.

## Topology Diagram for a Traditional IAM Deployment with Additional Microservices

This is the traditional Identity and Access Management deployment on a standalone on-premise hardware. In this topology, the software is distributed across eight hosts: two hosts for the web tier and four hosts for the application tier, and two hosts for the directory services. The new Oracle IDM Microservices are deployed inside a Kubernetes cluster and they interact with the core IDM components in a traditional deployment.
This topology is a typical deployment if you are using virtual machines (VMs) that are less powerful, but easy to create and manage. You deploy key components, such as Oracle Access Management, Oracle Identity Governance, and Directory Services on their own dedicated hosts.

The following illustration shows the Oracle Identity and Access Management topology. The diagram is shown with complete separation of components. If you wish to use less hardware, then you can collocate the components as required. For information about the system requirements for each host, see Host Computer Requirements.

Oracle 12c (12.2.1.4) introduces the Oracle Identity Management functionality that is delivered by using microservices. This functionality is delivered in addition to the existing Identity Management functionality that is part of the traditional deployment platform. You must deploy the new microservices inside a Kubernetes cluster. The traditional Identity Management components can be delivered either in a Kubernetes cluster or in a traditional deployment as described in Enterprise Deployment Guide for Oracle Identity and Access Management.

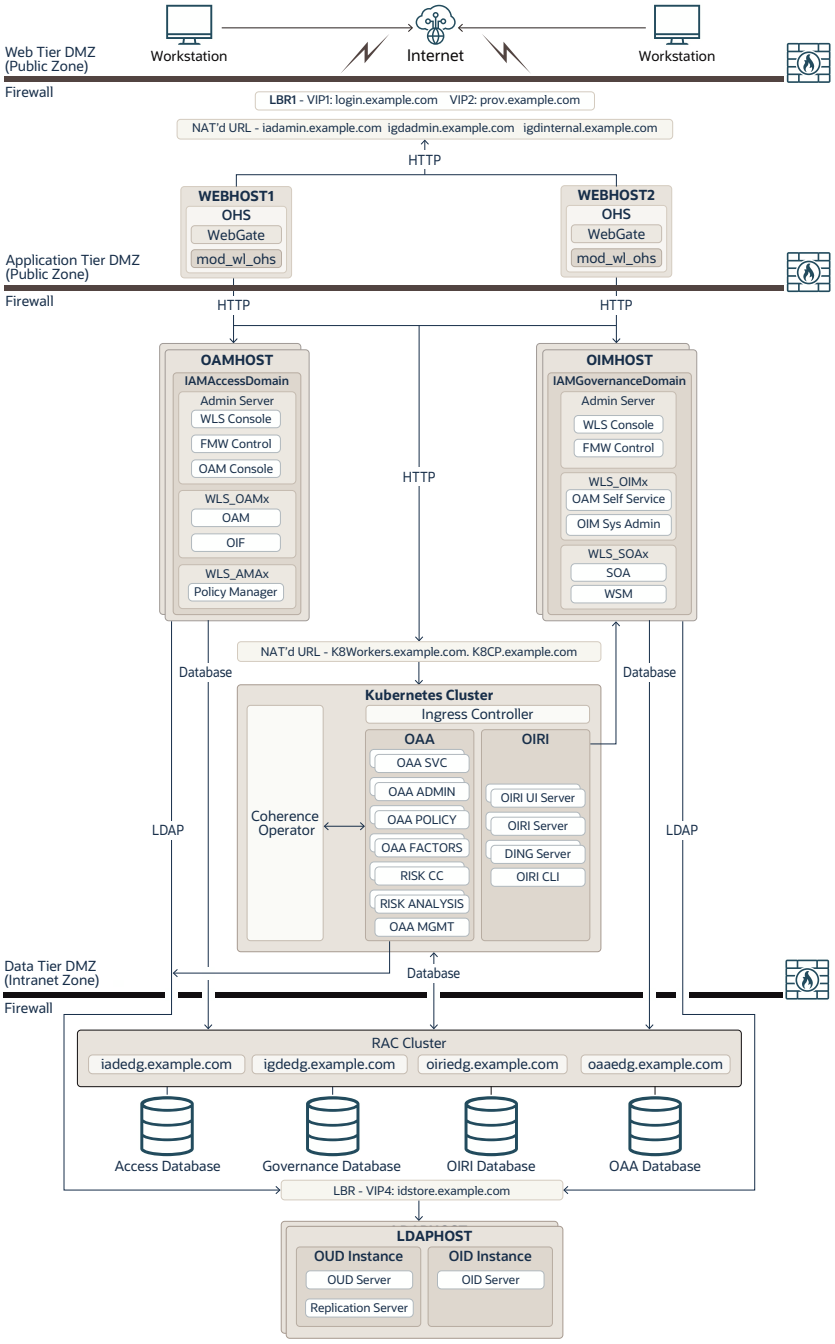
This guide describes how to deploy all the Oracle Identity and Access Management components inside a Kubernetes cluster.

If you are new to Kubernetes and want to keep your existing enterprise deployment, but extend the deployment with the new microservices, then you can follow the instructions in this guide to deploy just those microservices. If you decide to follow this route, then you have two options:

- Deploy the new microservices and integrate them into the existing Oracle HTTP server deployment.
  If you want to follow this deployment method, then use the instructions as described in Installing and Configuring Oracle HTTP Server. This chapter will show the entries you should add to incorporate microservices into the existing virtual hosts.
- Deploy the new microservices completely standalone, using a dedicated Ingress controller.

If you are following this deployment method, then you should define different virtual hosts for each of the microservices, which will resolve to the Ingress controller that is being used.

**Figure 4-1 A Traditional Deployment of Oracle Identity and Access Management with Microservices (in a Kubernetes Cluster)**



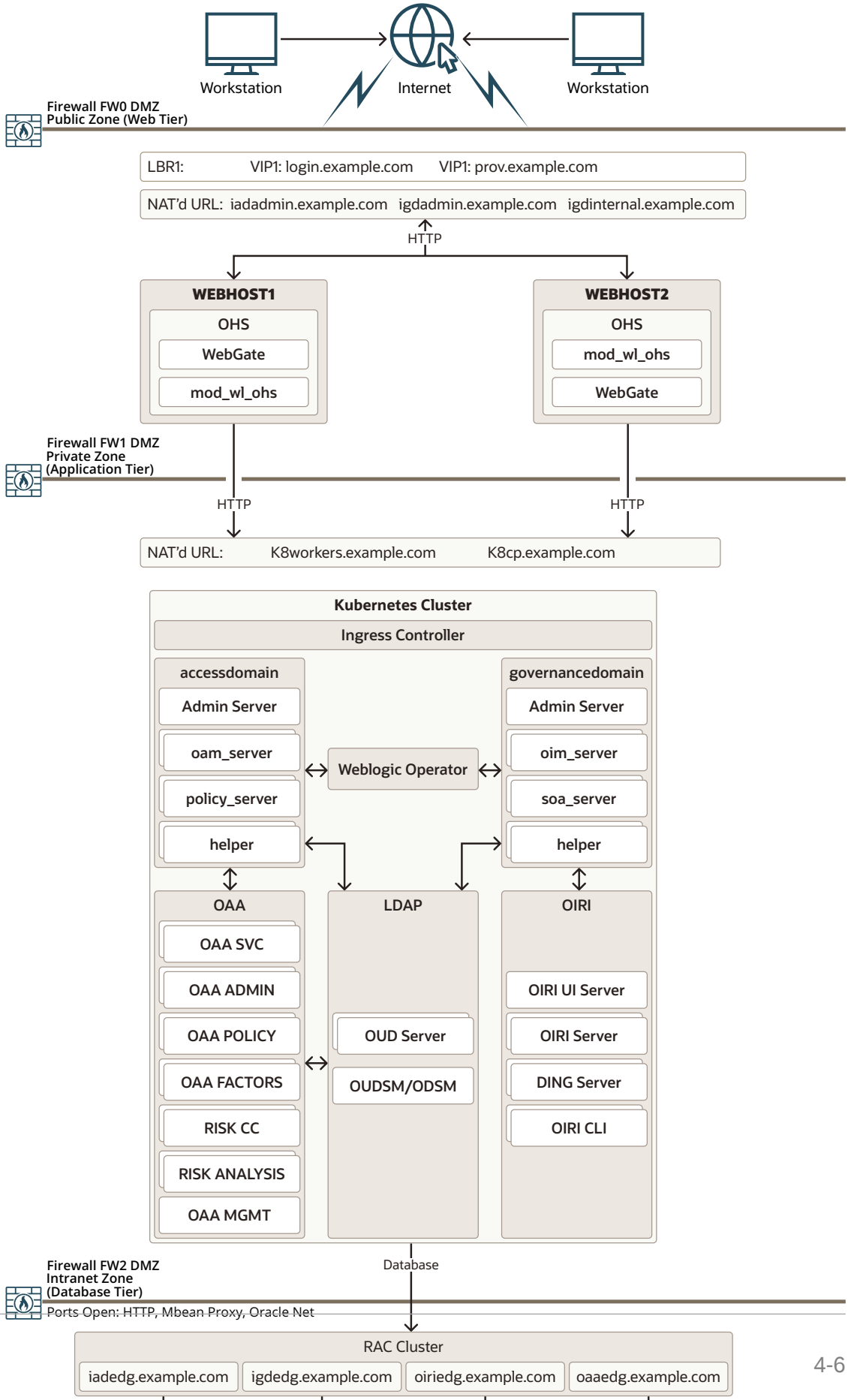

# Topology Diagram for the Deployment of Oracle Identity and Access Management on Kubernetes

This is a deployment of Identity and Access Management where all components of Identity and Access Management, including Microservices, are deployed into a Kubernetes cluster.

The following illustration shows the Oracle Identity and Access Management topology inside Kubernetes. The diagram is shown with complete separation of components. This topology requires that you use a suitably sized Kubernetes cluster.

This guide describes how to deploy all the Oracle Identity and Access Management components inside a Kubernetes cluster.

**Figure 4-2 Deployment of Oracle Identity and Access Management in a Kubernetes Cluster**



# Topology Diagrams for the Deployment of Oracle Identity and Access Management Components on Kubernetes

The sample topologies shown in this section illustrate the deployment of Oracle Identity and Access Management on a Kubernetes cluster by using either an Ingress controller for internal routing or by using individual NodePort Services. For simplicity, each product is separated into distinct diagrams.
The control plane consists of three nodes where the Kubernetes API server is deployed and front ended by a load balancer. The load balancer exposes the required VIP and VS for both the Identity Management suite and the control plane itself. This URL should be site agnostic in preparation for disaster recovery configuration. The control plane and worker nodes should resolve this name properly. The database tier consists of a unique RAC DB that hosts the application schemas, JMS/JTA persistent stores, OPSS, and MDS information.An extra load balancer end point maybe created to include all the Kubernetes worker nodes. You can then use this load balancer to distribute requests seamlessly across the available pool of worker nodes.

This section includes the topology diagrams for the following components:

- Primary Deployment Topologies
- Secondary Deployment Topologies

## Primary Deployment Topologies

These topologies are the recommended approach suggested by Oracle.

- Oracle Unified Directory
- Oracle Access Manager and Oracle Identity Governance
- Oracle Identity Role Intelligence
- Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

### Oracle Unified Directory

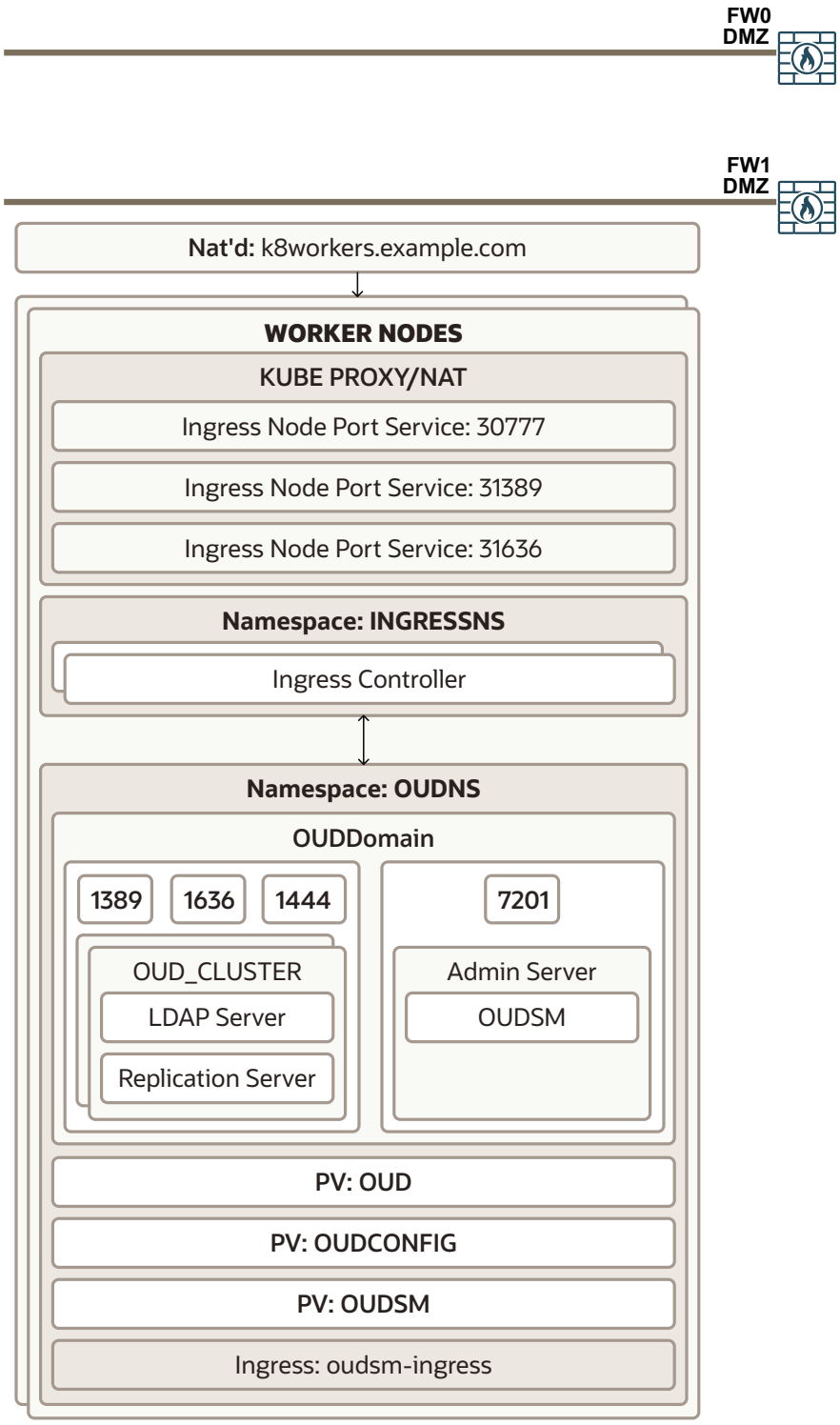
Figure 4-3 shows a deployment of Oracle Unified Directory and Oracle Unified Directory Services Manager. Typically, none of the services in this diagram are exposed to the internet.

In a full Kubernetes deployment, interactions with OUD are normally kept inside of the Kubernetes cluster. In this case, LDAP calls can be confined to the cluster.

This deployment uses an Ingress controller to enable external interactions with OUDSM. Ingress services are used only when access to the OUD LDAP services are required outside of the Kubernetes cluster, such as a third party telephone book application. External access may also be required for OUDSM.

While it is possible to access OUDSM through the Oracle HTTP server, because this is an administration function, you can access OUDM directly through the Ingress controller.

**Figure 4-3 OUD Using the Ingress Controller**



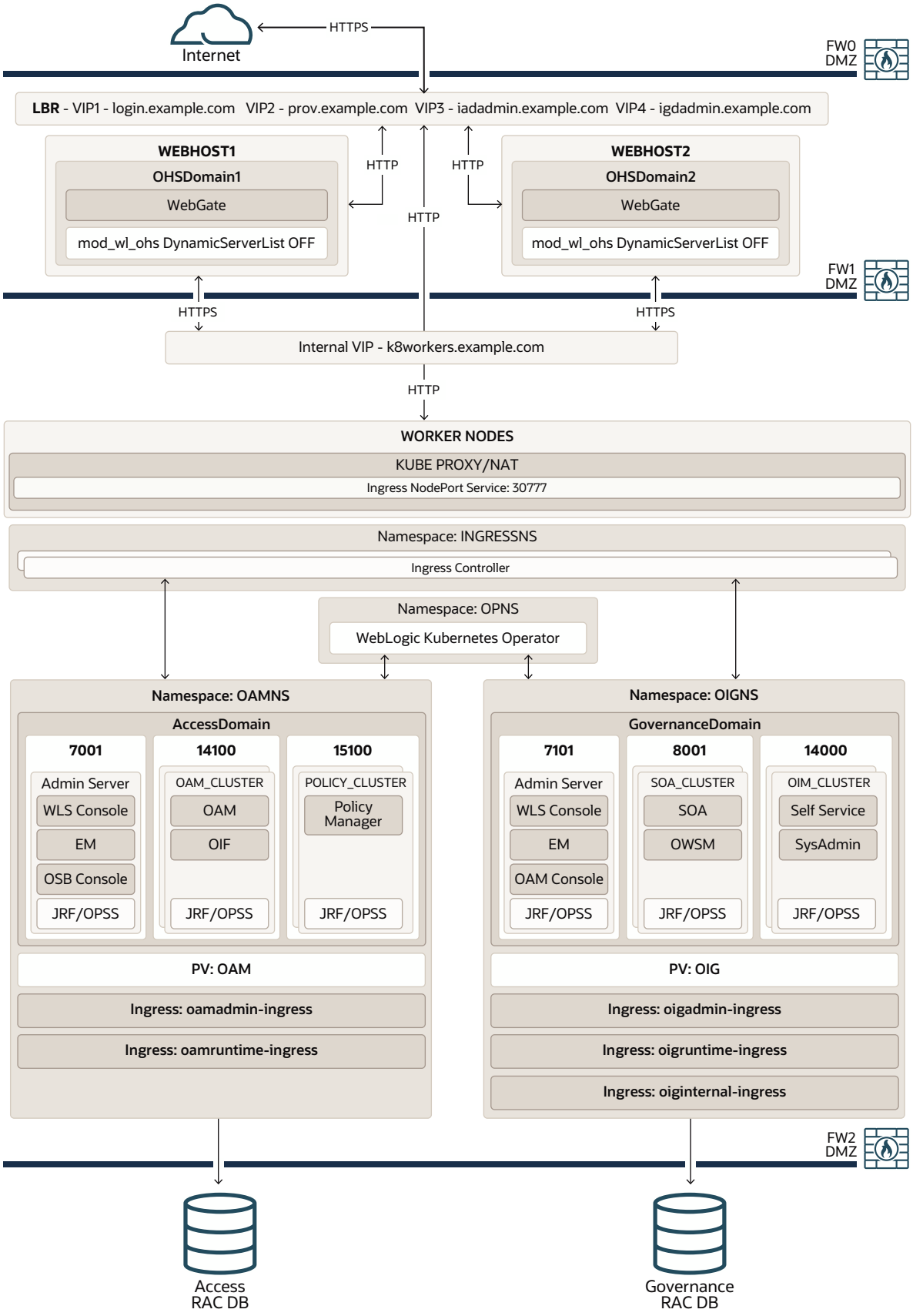

## Oracle Access Manager and Oracle Identity Governance

Figure 4-4 shows a deployment of Oracle Access Manager and Oracle Identity Governance.

A load balancer provides external internet access to the OAM and OIG runtime functionality. The same load balancer provides only internal access to the OAM and OIG administrative services.

External requests are SSL terminated at the load balancer and internal requests are standard HTTP requests. The load balancer uses the Oracle HTTP server to proxy these requests to the OAM and OIG services. The Oracle HTTP server, in turn, passes on the application requests to an Ingress controller which forwards the requests to the OAM and OIG Kubernetes services.

**Figure 4-4 OAM and OIG Using an Ingress Controller**



## Oracle Identity Role Intelligence

Figure 4-5 shows the deployment of Oracle Identity Role Intelligence integrated into a complete Oracle Identity and Access Management deployment where all traffic is routed through an Oracle HTTP Server and then an Ingress controller.

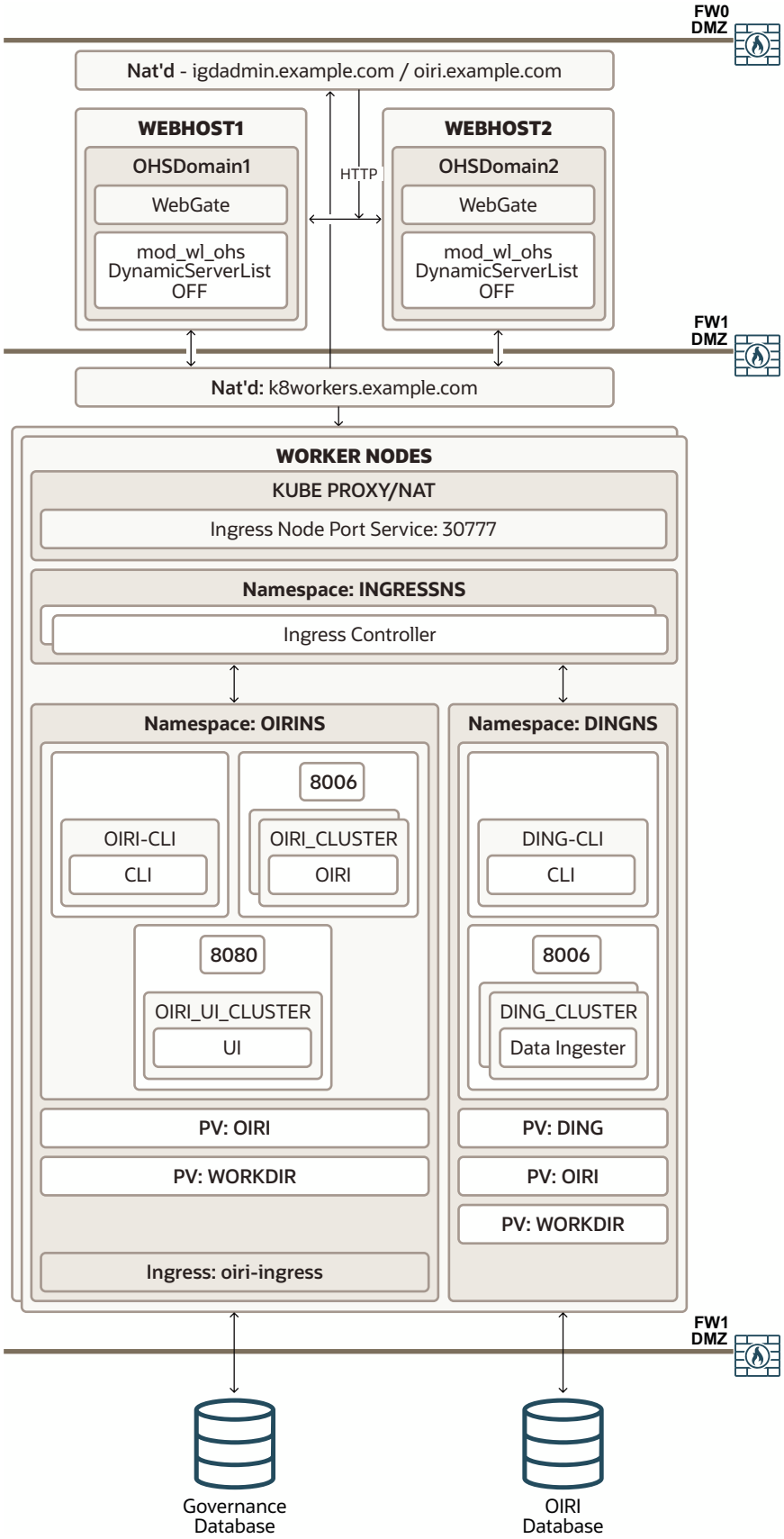
In this type of deployment, you can integrate Oracle Identity Role Intelligence with a complete Oracle Identity and Access deployment whether the other Identity components reside in the Kubernetes cluster or not.

A load balancer provides internal only access to the OIRI services. The load balancer uses the Oracle HTTP Server to proxy these requests to the OIRI. The Oracle HTTP server is used where a fully integrated Identity Management deployment is being used, and everything is proxied through the Oracle HTTP server.

OIRI can either share the `igdadmin.example.com` virtual host or use its own dedicated virtual host.

OIRI is an administration only function and does not have any direct interaction with the internet.

**Figure 4-5 Integrated OIRI Using the Ingress Controller**



## Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

Figure 4-6 shows a deployment of Oracle Advanced Authentication. OAA has both administrative and runtime operations.

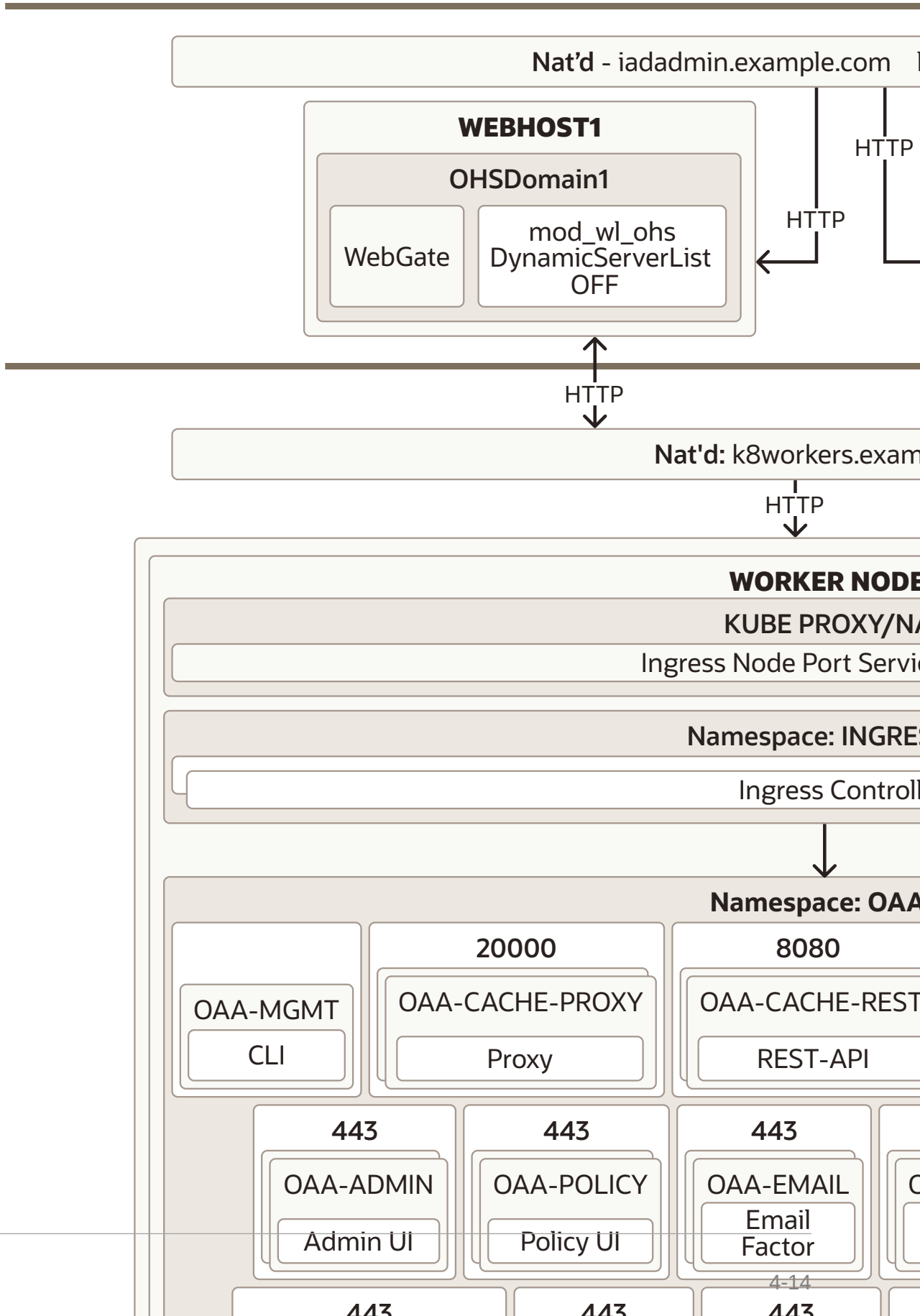
A load balancer provides access to the OAA/OARM services. The load balancer sends these requests to the Ingress controller directly, which in turn directs the requests to the appropriate OAA microservice. The Oracle HTTP Server is used where a fully integrated Identity Management deployment is being used, and everything is proxied through the Oracle HTTP Server.

The Oracle HTTP Server sends requests to an Ingress controller, which sends these requests to the appropriate OAA Kubernetes services.

OAA can either share the `iadadmin`/`login` virtual hosts or use its own dedicated virtual hosts.

External requests are SSL terminated at the load balancer and internal requests are standard HTTP requests.

**Figure 4-6 OAA Using an Ingress Controller**



# Secondary Deployment Topologies

These topologies are fully supported alternative deployments that you can consider using.

- Oracle Unified Directory
- Oracle Access Manager and Oracle Identity Governance
- Oracle Identity Role Intelligence
- Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

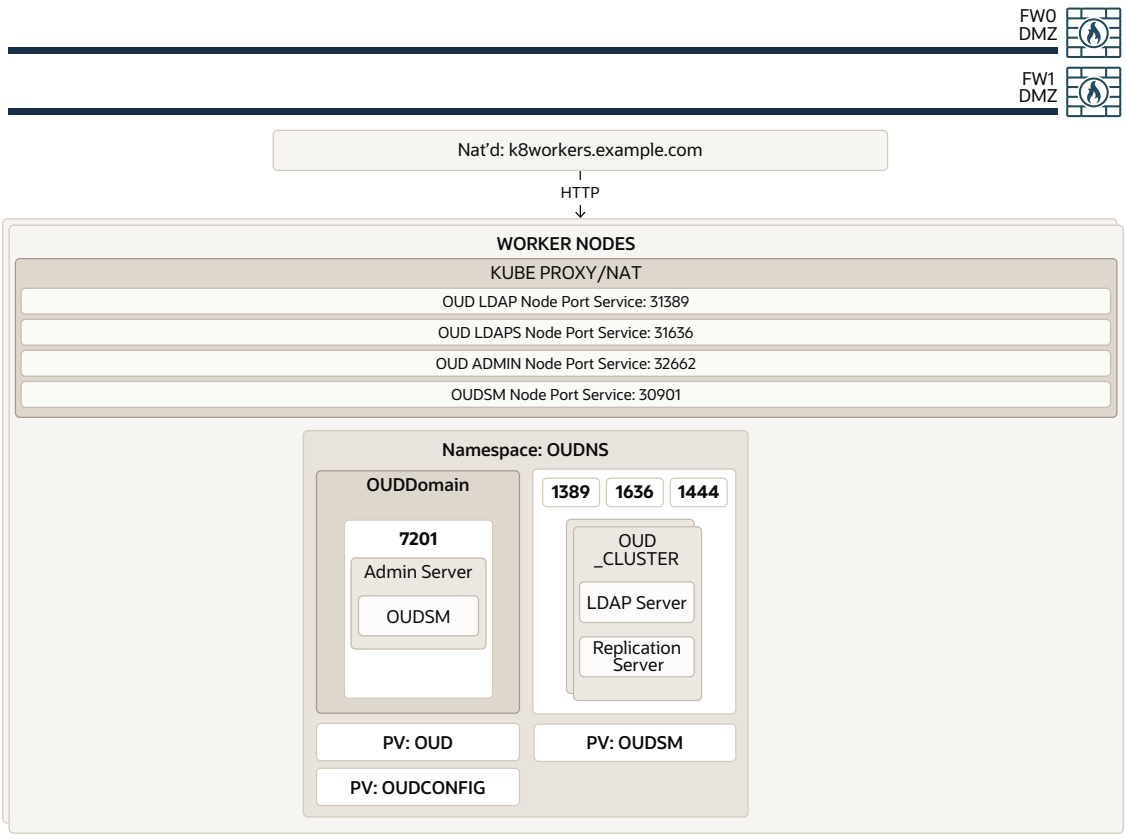
## Oracle Unified Directory

Figure 4-7 shows a deployment of Oracle Unified Directory and Oracle Unified Directory Services Manager. Typically, none of the services in this diagram are exposed to the internet.

In a full Kubernetes deployment, interactions with OUD are normally kept inside of the Kubernetes cluster. In this case, LDAP calls can be confined to the cluster and NodePort Services are not required. NodePort Services are used only when access to the OUD LDAP services are required outside of the Kubernetes cluster, such as a third party telephone book application. External access may also be required for OUDSM.

While it is possible to access OUDSM through the Oracle HTTP server, because this is an administration function, you can access it directly using the NodePort Service exposed for OUDSM.

**Figure 4-7 OUD Using the NodePort Services**

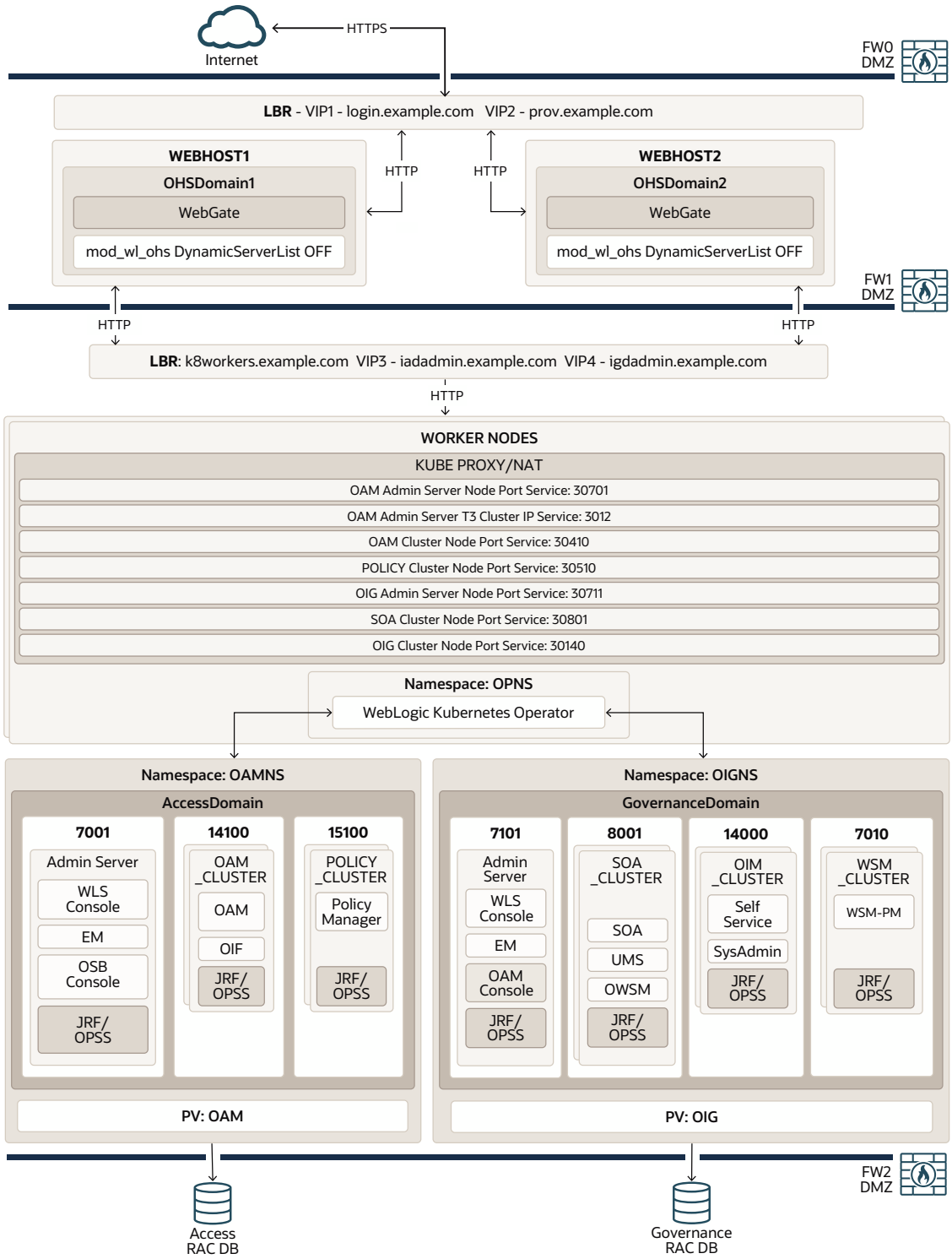


## Oracle Access Manager and Oracle Identity Governance

Figure 4-8 shows a deployment of Oracle Access Manager and Oracle Identity Governance.

A load balancer provides external internet access to the OAM and OIG runtime functionality. The same load balancer provides only internal access to the OAM and OIG administrative services. External requests are SSL terminated at the load balancer and internal requests are standard HTTP requests. The load balancer uses the Oracle HTTP server to proxy these requests to the OAM and OIG services. OAM and OIG services are exposed outside of the Kubernetes cluster using individual NodePort Services.

**Figure 4-8 OAM and OIG Using the NodePort Services**




## Oracle Identity Role Intelligence

Figure 4-9 shows a standalone deployment of Oracle Identity Role Intelligence where traffic is routed through an Ingress controller.

The Ingress controller sends the requests to the appropriate OIRI Kubernetes services. OIRI can either share the `igdadmin.example.com` virtual host or use its own dedicated virtual host.

OIRI is an administration only function and does not have any direct interaction with the internet.

**Figure 4-9 OIRI Using Only an Ingress Controller**

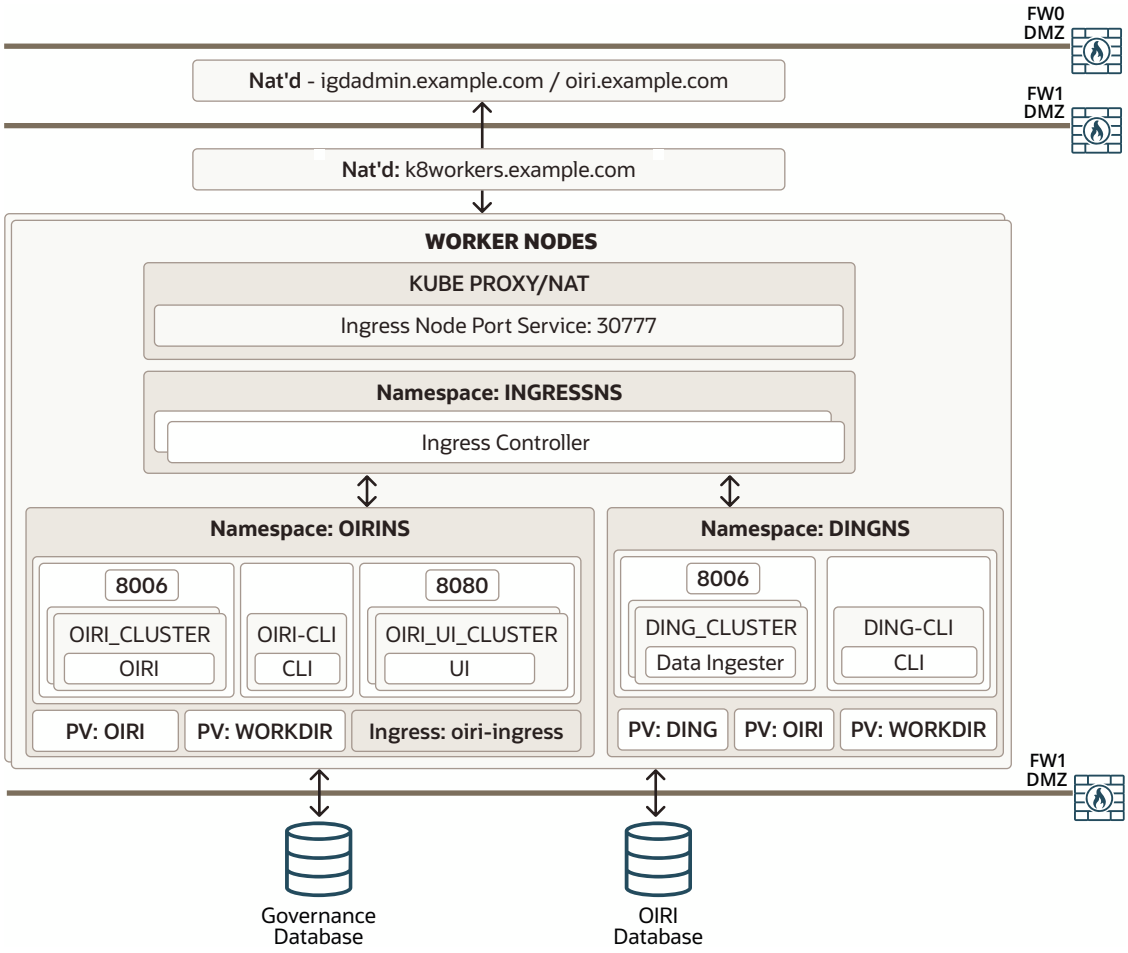



Figure 4-10 shows the deployment of Oracle Identity Role Intelligence integrated into a complete Oracle Identity and Access Management deployment where all traffic is routed through an Oracle HTTP Server and then sent directly to the OIRI microservices by using the Kubernetes NodePort Services. In this type of deployment, you can integrate Oracle Identity Role Intelligence with a complete Oracle Identity and Access Management deployment whether the other Identity components reside in the Kubernetes cluster or not.

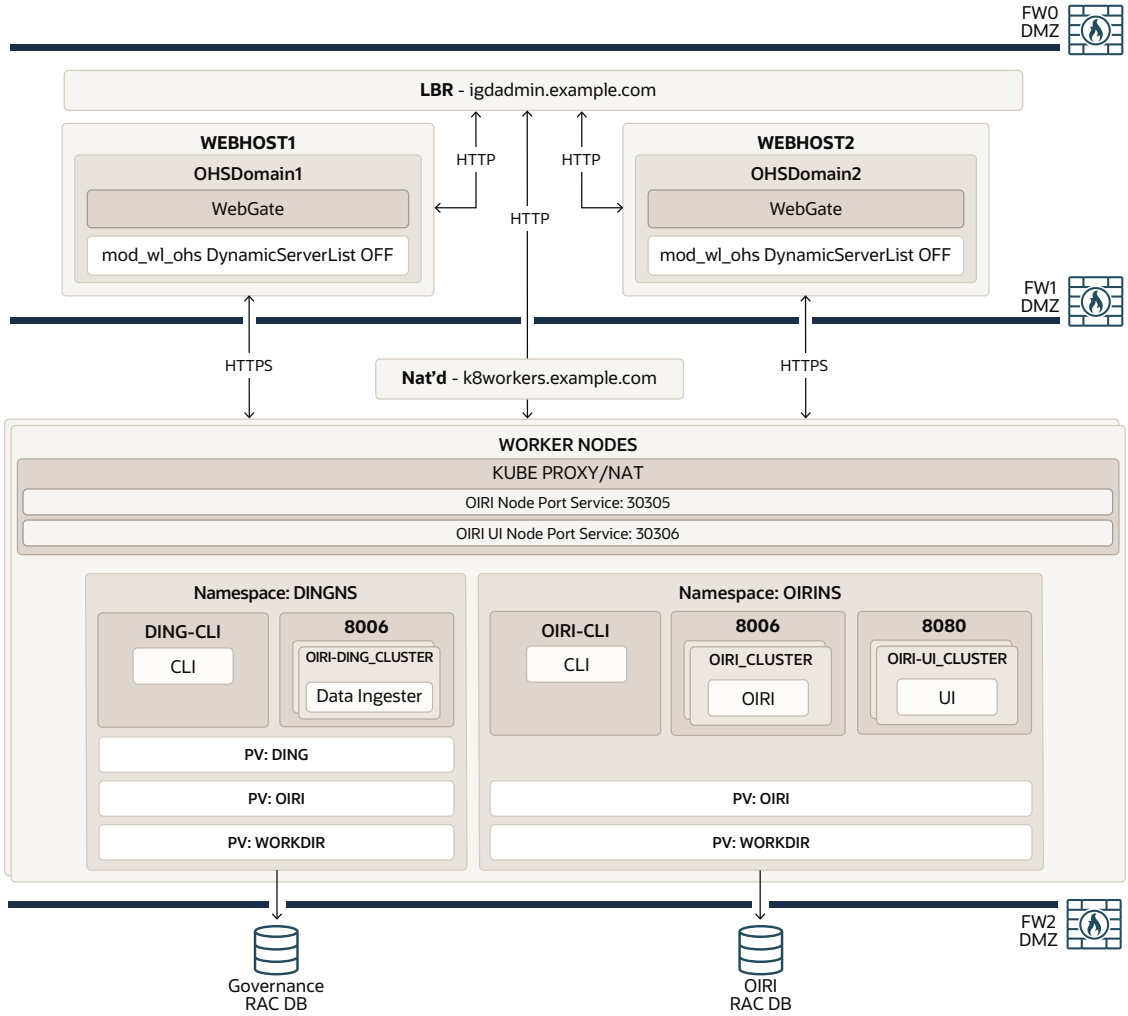
A load balancer provides internal only access to the OIRI services. The load balancer uses the Oracle HTTP Server to proxy these requests to the OIRI. The Oracle HTTP server is used where a fully integrated Identity Management deployment is being used, and everything is proxied through the Oracle HTTP server.

OIRI services are exposed outside of the Kubernetes cluster using individual NodePort services. You can deploy OIRI without the Oracle HTTP server. See Figure 4-10.

OIRI can either share the `igdadmin.example.com` virtual host or use its own dedicated virtual host.

OIRI is an administration only function and does not have any direct interaction with the internet.

**Figure 4-10 Integrated OIRI Using the NodePort Services**




## Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

Figure 4-11 shows a deployment of Oracle Advanced Authentication and Risk Management, in a fully integrated Oracle Identity and Access Management deployment.

A load balancer provides external internet access to the OAA and OARM runtime functionality. The same load balancer provides internal only access to the OAA and ORAM administrative services.

External requests are SSL terminated at the load balancer and internal requests are standard HTTP requests. The load balancer uses the Oracle HTTP server to proxy these requests to the OAA and OARM services. OAA and OARM services are exposed outside of the Kubernetes cluster using individual NodePort Services.

> **Note:**
>
> In an OAA and OARM deployment, the OAA microservices are SSL enabled.

**Figure 4-11 OAA Using the NodePort Services**

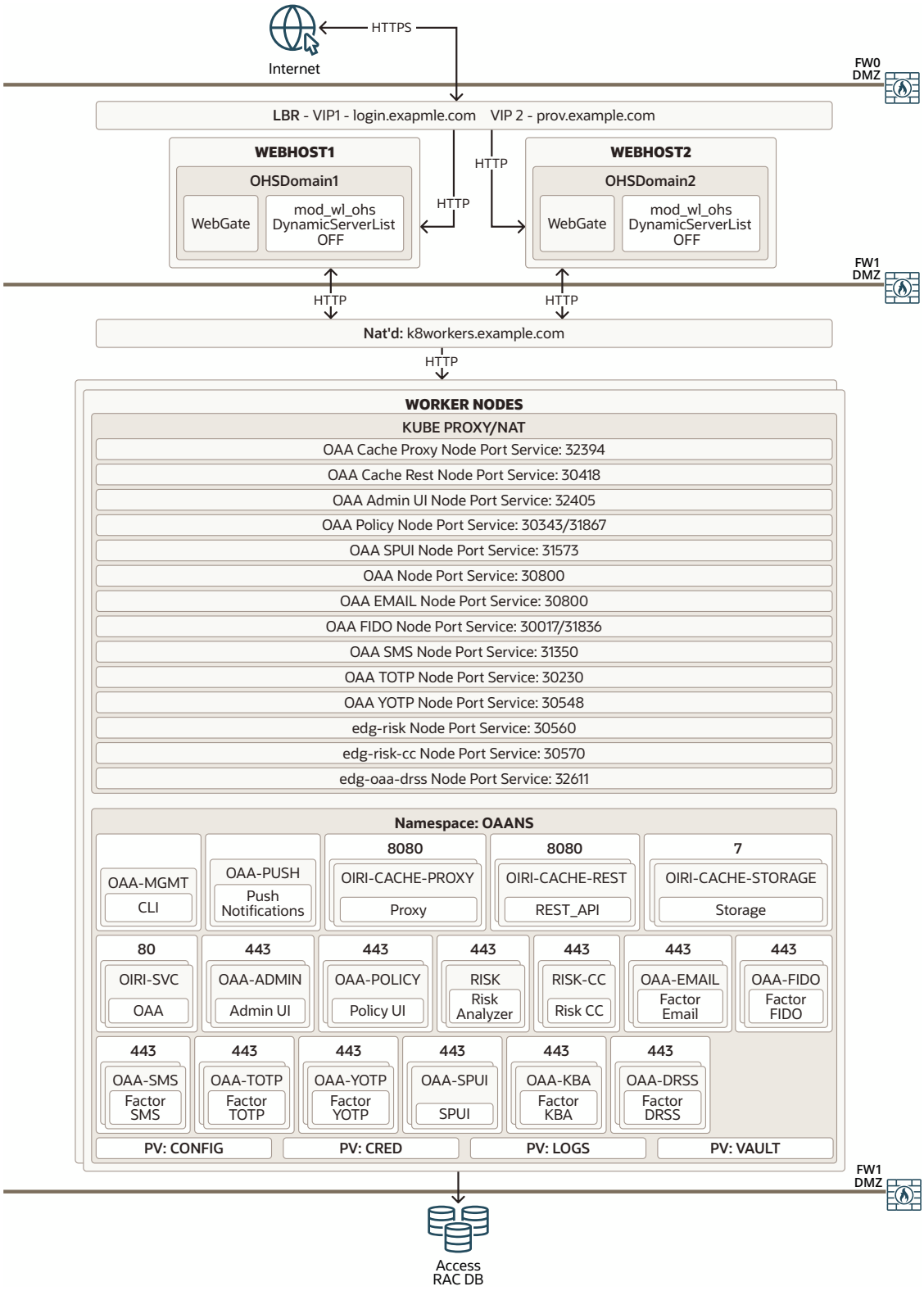
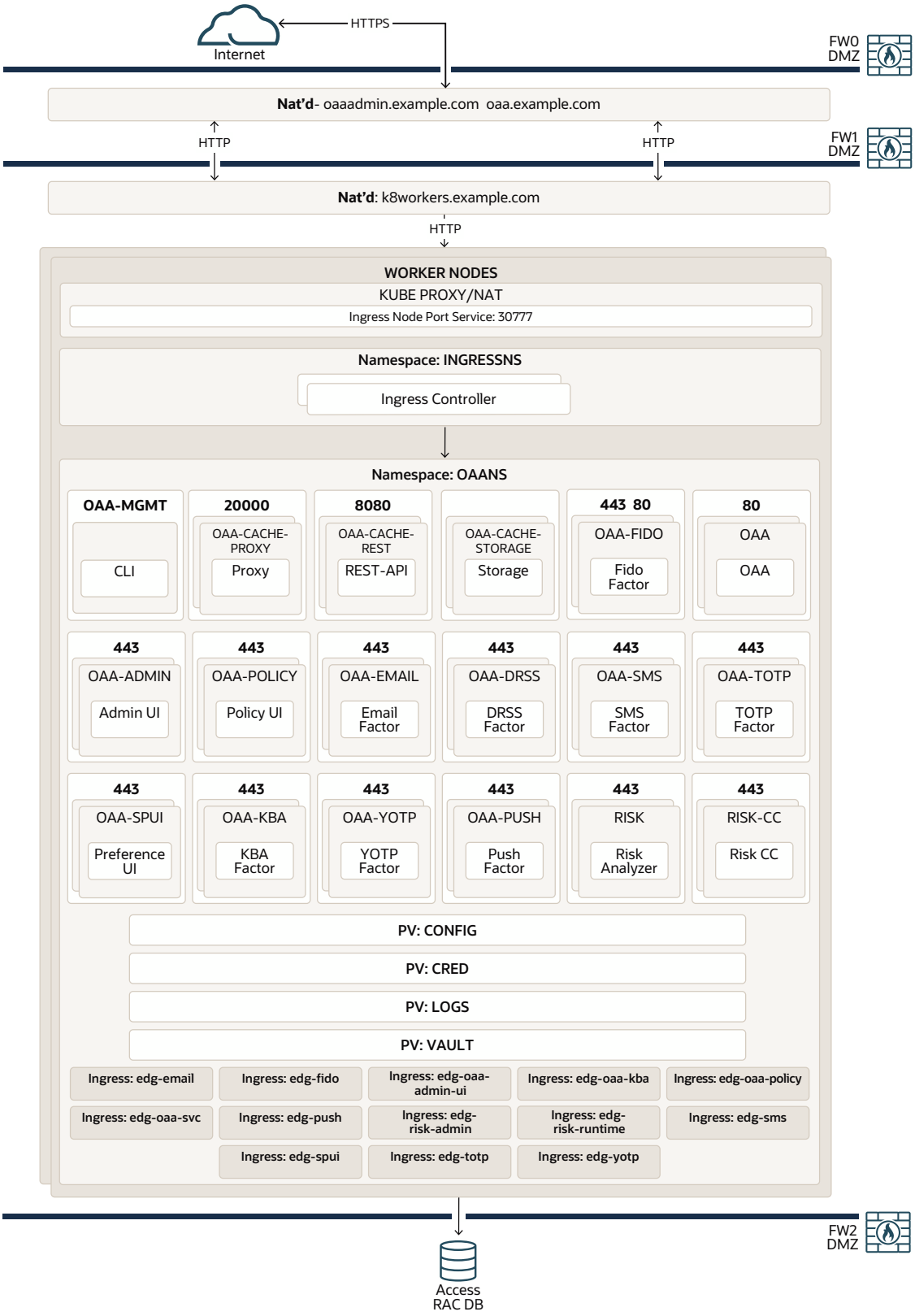



Figure 4-12 shows a standalone deployment of Oracle Advanced Authentication and Risk Management. A load balancer provides external internet access to the OAA and OARM

administrative services. An Ingress controller provides access to the OAA Kubernetes services. In this scenario, OAA uses its own dedicated virtual host.

**Figure 4-12 Standalone OAA Using Ingress on Kubernetes**



# About the Primary Oracle Identity and Access Management Topology Diagrams

The primary enterprise deployment topology is the main Oracle reference topology for Oracle Identity and Access Management. It provides a solution which is both highly available and scalable.



## Product Separation

An IAM deployment is made up of a number of components. These components include:

- Web Servers
- WebLogic Application Servers
- Microservices deployed in containers
- LDAP
- Database

The Oracle Identity and Access Management components are split into a number of different WebLogic domains: IAMAccessDomain and IAMGovernanceDomain. Products are distributed as follows:

- IAMAccessDomain contains Oracle Access Management (OAM).
- IAMGovernanceDomain contains Oracle Identity Governance.
- OHSDomain used to host the Oracle HTTP Servers.
- OUDSMDomain used when OUDSM is deployed.

> **Note:**
>
> In a Kubernetes deployment, the OHS and OUDSM domains are not controlled by the WebLogic Kubernetes Operator.

This split is due to the different operational and availability requirements demanded of the individual components. Typically components in the IAMAccessDomain have a higher availability requirement than those in the IAMGovernanceDomain. By separating these components out, you can manage the availability requirements differently. You can patch

governance components independently of access components, and you can shut down the Governance instance without impacting the access components. From Oracle Identity and Access Management 12c, it is not supported to have Oracle Access Manager and Oracle Identity Governance in the same domain.

The OHSDomain and OUDSMDomain are are used to manage the Oracle HTTP Server and Oracle Unified Directory Services Manager, respectively.

In addition to the domains above, the following components are deployed without a domain:

- Oracle Unified Directory
- Microservices

A further benefit of this separation is that you can build a topology in a modular fashion. You can start with a directory and extend it to Access components, then later extend it to Governance components, without needing to affect the deployed software or configuration of existing components, unless you are wiring them together.

Oracle Identity Role Intelligence and Oracle Advanced Authentication are deployed as dedicated micro services.

## About the Web Tier

The web tier consists of two or more host computers where Oracle HTTP Server is installed. The web tier is located in a demilitarized zone (DMZ) in a different network/subnet from the application tiers. This location ensures that maximum network security is maintained.

The web tier is located between two firewalls, which helps limit traffic from the internet and ensures that only application traffic is allowed to pass through to the application tier.

In an internet-facing application, the web tier is defined so that administration-type functions are constrained to a virtual host, which is only resolvable inside the organization. This feature ensures that someone should be present inside the private network to perform the administrative operations.

The fact that the administration virtual hosts are resolvable only inside the private network means that communications using this virtual host do not require SSL.

Communications with the internet must be to and from the internet client using SSL. However, end-to-end SSL in an application has a performance impact. By placing the web tier in a DMZ, you can ensure that SSL is terminated outside of the web tier by a load balancer, and communication inside the topology uses the much faster non-SSL protocol. This strategy has the benefit of ensuring that all internet traffic is SSL enabled but without the performance overhead of end-to-end SSL.

## About Oracle Unified Directory Assured Replication

Oracle Unified Directory server instances natively use replication to keep their embedded databases in sync. By default, replication employs a loose consistency model in which the updates are replicated to replicas AFTER returning the operation result to the application. In this model it is therefore possible to write some data to a replica, and read outdated information from another replica for a short time after the write. Great efforts have been made in Oracle Unified Directory replication to ensure that the replication process is fast and can achieve replication in the order of one millisecond.

Oracle Unified Directory can be configured to use the Assured Replication model, which has been developed to guarantee that the data in the replicas is consistent. When using the Safe

Read mode of Assured Replication, applications have the guarantee that the replication process is completed before returning the result of a write operation.

Using Assured Replication has a negative impact on the response time of write operations because it requires some communications with remote replicas before returning the operation result. The amount of the delay varies, depending on the network being used and the capacity of the servers hosting Oracle Unified Directory. Using Assured replication has little if any impact on read operations.

If you expect to regularly perform large writes to your directory, consider configuring your load balancer to distribute requests to your Oracle Unified Directory instances in an active/passive mode. This will remove the chance of you reading out of date data from a replica, but could result in overall performance degradation if your Oracle Unified Directory host is not capable of processing all of the requests.

For the purposes of this Guide, it is assumed that the ability to have multiple servers processing requests is more important than the extra overhead incurred with writing requests in Assured mode. To that end, this Guide shows the configuration of Oracle Unified Directory using Assured Replication. Both of the following Oracle Unified Directory configurations, however, are supported:

- Active/Active in an assured configuration
- Active/Passive in a non assured configuration

For more information, see the Assured Replication section of *Oracle Fusion Middleware Administrator's Guide for Oracle Unified Directory*.

## About Oracle Identity Role Intelligence

Role-Based Access Control (RBAC) has the following challenges:

- Building roles as a manual process is time-consuming. Entitlement data is difficult and complex for users to analyze and interpret.
- Entitlements accumulate over time. Users and applications data change constantly.
- Roles are difficult to maintain and change to align with business activities such as reorganization, merge, acquisition, and so on.
- Lack of tooling to provide what if analysis before organizations adopt roles for various business units.

These challenges are addressed by the Oracle Identity Role Intelligence (OIRI) microservice.

Oracle Identity Role Intelligence uses a microservice architecture rather than the traditional WebLogic deployment architecture. This means that Oracle Identity Role intelligence should be deployed using a container based architecture such as Kubernetes. OIRI will continue to integrate with the traditional Oracle Identity and Access Management deployment. However, the OIRI component should be deployed in a container.

Oracle Identity Role Intelligence is an extension to Oracle Identity Governance (OIG). It is used by administrators. Therefore, in an enterprise deployment it is tied to the administrative application URL `igdadmin.example.com`.

> **Note:**
>
> In this release, although OIRI is integrated into the overall Oracle Identity and Access enterprise deployment architecture, it does not support single-sign on through Oracle Access Manager.

**Data Ingress**: Supports data import to OIRI from the OIG database or flat files in full and incremental modes.

**Data Modelling**: Enables you to define role mining tasks based on a combination of user, application, and entitlement attributes.

**Predictive Analytics**: Uses Oracle Database's KMean clustering and unsupervised Machine Learning (ML) algorithms. The regression model groups the user data based on the common entitlement attributes, and predicts the relevant and matching candidate roles.

**Assistant**: Compares candidate roles with the existing roles in the system. You can publish the candidate roles to your system to avoid duplication or explosion of roles.

**Data Egress**: Provides automation to publish the candidate roles to Oracle Identity Governance and triggers the workflow approval.

## About Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

Oracle Advanced Authentication (OAA) supports the establishment and assertion of the identity of users.

OAA provides strong authentication using Multiple Authentication Factors (MFA). A wide range of authentication factors are available out-of-the-box for establishing the identity of users. OAA supports integration with Oracle Access Management (OAM) and Oracle Radius Agent (ORA) to provide Multi Factor Authentication (MFA) capabilities.

OAA constitutes unique features that facilitate deployment, configuration, and integration with other products.

OAA has the following features:

- Runs as a standalone micro-service on a Kubernetes platform and is deployed using the Helm charts
- Supports integration with the following clients to enable Multi-factor Authentication (MFA):
  - Clients providing web-based user login flows, such as Oracle Access Management (OAM): OAA integrates with OAM through the Trusted Authentication Protocol (TAP).
  - Clients providing API-based user login flows, such as Oracle Radius Agent (ORA): OAA integrates with ORA through REST APIs. This type of integration enables clients to manage their own user-flow orchestration.
- Provides `OAAAuthnPlugin` for integrating with OAM. The plug-in also enables migration of user data from the identity store on OAM to OAA.
- Provides a web UI (admin UI console) for administrators to create and manage client registrations, assurance levels, and policies. Administrators can also achieve all the administration tasks using the REST APIs.

- Provides a web UI (user preferences UI) for users to manage and register their challenge factors. User self-registration and management can also be performed using the REST APIs.
- Web UIs are secured by OAM OAuth and OIDC.
- Provides the following challenge factors out-of-the-box:
  - Time-based One Time Password (TOTP): (OMA, Google, and Microsoft)
  - OTP (Email and SMS)
  - Yubikey OTP
  - Fast Identity Online (FIDO2)

You can deploy Oracle Advanced Authentication as a standalone service or as part of an integrated Oracle Identity and Access Management deployment.

# Summary of Oracle Identity and Access Management Load Balancer Virtual Server Names

In order to balance the load on servers and to provide high availability, a hardware load balancer is required. This hardware load balancer should exist on redundant hardware to ensure maximum availability. The hardware load balancer must be configured to recognize a set of virtual server names.

The hardware load balancer in Oracle Identity and Access Management deployments must recognize the following virtual server names.

- `login.example.com` - This virtual server name is used for all incoming Access traffic. It acts as the access point for all HTTP traffic to the runtime Access Management components. The load balancer routes all requests to this virtual server name over SSL. As a result, clients access this service using the following secure address:

  `login.example.com:443`

- `prov.example.com` - This virtual server name is used for all incoming Governance traffic. It acts as the access point for all HTTP traffic to the runtime Governance components. The load balancer routes all requests to this virtual server name over SSL. As a result, clients access this service using the following secure address:

  `prov.example.com:443`

  Note that, in previous releases of the Enterprise Deployment Guide, `login.example.com` and `prov.example.com` were the same entry point. This release allows for them to be separated out. This will enable smarter routing from the load balancer, allow a more modular deployment and will facilitate future Multi-datacenter deployments. If desired these two entry points can still be combined to provide a single point of entry into the IAM deployment.

- `iadadmin.example.com` - This virtual server name is enabled on the load balancer. It acts as the access point for all internal HTTP traffic that gets directed to the administration services in the IAMAccessDomain. The incoming traffic from clients is non-SSL enabled. Therefore, the clients access this service using the following address:

  `iadadmin.example.com:80`

  This in turn is forwarded to port `7777` on WEBHOST1 and WEBHOST2.

  The services accessed on this virtual host include the WebLogic Administration Server Console and Oracle Enterprise Manager Fusion Middleware Control.

Create rules in the firewall to block outside traffic from accessing the /console and /em URLs using this virtual host. Only traffic inside the DMZ should be able to access these URLs on the `iadadmin.example.com` virtual host.

- `igdadmin.example.com` - This virtual server name is enabled on the load balancer. It acts as the access point for all internal HTTP traffic that gets directed to the administration services in the IAMGovernanceDomain. The igdadmin entry point is also used as the entry point for Oracle Identity Role Intelligence. The incoming traffic from clients is non-SSL enabled. Therefore, the clients access this service using the following address:

  igdadmin.example.com:80

  This in turn is forwarded to port `7777` on WEBHOST1 and WEBHOST2.

  The services accessed on this virtual host include the WebLogic Administration Server Console and Oracle Enterprise Manager Fusion Middleware Control.

  Create rules in the firewall to block outside traffic from accessing the /console and /em URLs using this virtual host. Only traffic inside the DMZ should be able to access these URLs on the igdadmin.example.com virtual host.

- `igdinternal.example.com` - This virtual server name is for internal communications between the application tier components in the Governance Domain only and is not exposed to the Internet. This virtual server is used for both Oracle OIM Suite and Oracle SOA Suite internal communications.

  The traffic from clients to this virtual server is not SSL-enabled. Clients access this service using the following address and the requests are forwarded to port 7777 on WEBHOST1 and WEBHOST2 (this is optional in a Kubernetes environment):

  `igdinternal.example.com:7777`

- `idstore.example.com` - This virtual server name is used for all incoming identity store traffic. It acts as the access point to the LDAP directory instances. This virtual server is not exposed to the internet.

  The traffic from clients to this virtual server may or may not be SSL-enabled, depending on the type of LDAP directory in use. Typically, this will be non-SSL enabled for Oracle Unified Directory. Clients access this service using this virtual server name and the requests are forwarded to the LDAP instances.

> **Note:**
>
> If all of the applications that access the LDAP directory are inside the Kubernetes cluster, you can choose to use the Kubernetes service names rather than a load balancer virtual host, for LDAP access.

## About Oracle Identity Management on Kubernetes

Although the topology of Oracle Identity Management on Kubernetes is similar to that of a traditional Oracle Identity Management deployment, there are some notable differences which are described below:

- **Managed Servers**
  Oracle WebLogic Managed Servers, including the Administration Server, are deployed in a container. Each Managed Server resides in its own container.
- **Virtual IP Addresses**

A traditional enterprise deployment makes use of virtual IP addresses to facilitate Administration failover and server migration. This is not required in a Kubernetes deployment because Kubernetes automatically assigns its own IP addresses.

- **Managed Server Interaction**
  In a traditional enterprise deployment, you interact with Managed Servers using the virtual host name assigned to the Managed Server's listen address or the physical host name on which the Managed Server is running, depending on how you have configured the Managed Server.

  In a Kubernetes environment, Kubernetes services are defined at the cluster level. When the Kubernetes containers are added and/or removed, the Kubernetes service gets updated accordingly. All communication occurs through the Kubernetes services.

  Kubernetes services are accessible from each Kubernetes worker host.

- **Microservices**
  Microservices extend the traditional Oracle Identity and Access Management functionality. Each microservice is deployed in a Kubernetes container. Similar to Managed Servers, Microservices are assigned a Kubernetes service defined at the cluster level, to provide High Availability. Interaction with micoservices is through the Oracle HTTP Server.

- **Shared Storage and Persistence**
  Each container is associated with a persistent volume. The persistent volume is where the domain/instance files are located. The persistent volume is mapped to an NFS share enabling you to use traditional backup and recovery techniques.

  Only the information that is stored on a persistent volume survives reboots. The *ORACLE_HOME* is not stored on persistent volumes. Therefore, the changes made will not survive a reboot. This makes patching easier. However, you cannot directly edit the files in *ORACLE_HOME*.

  Unlike the traditional enterprise deployment, the *DOMAIN_HOME* is not split into *ASERVER_HOME* and *MSERVER_HOME*.

- **Customizations**
  Certain Oracle products allow customizations. These are often made by changing/creating `.war` files in *ORACLE_HOME*. In a Kubernetes deployment, any changes made in this way is lost after you restart a container. If you want to make customizations, place the `.war` files (including the dev starter pack) inside the persistent volume mounted at `/u01/oracle/user_projects`, preferably in a dedicated directory. This method persists customizations across restarts and image changes (bundle patches and upgrades).

  After you place the `.war` files in the persistent volume, deploy it to the domain from that location (`/u01/oracle/user_projects`).

- **Containers vs Hosts**
  In a typical enterprise deployment, each WebLogic domain, each LDAP instance, each database and each web server is running on a number of hosts.

  In a Kubernetes environment, the WebLogic/Microservices/LDAP components are moved into the Kubernetes cluster. The web tier and database are not. This enables firewalls to be erected between the internet and the web tier, between the web tier and the Kubernetes cluster, and between the Kubernetes tier and the database. This enables you to utilize the flexibility that Kubernetes offers for the application tier, while maintaining entrerprise security.

## Summary of the Managed Servers and Clusters on the Application Tier Hosts

The Application tier hosts the Administration Server and Managed Servers in the Oracle WebLogic Server domains.

Depending upon the components you select, the Oracle WebLogic Server domain for the Oracle Identity and Access Management consists of the clusters shown in Table 4-1. These clusters function as active-active high availability configurations.

**Table 4-1 Domain Clusters and Managed Servers**

| Domain | Cluster | Managed Servers |
|---|---|---|
| IAMAccessDomain | Oracle Access Manager | oam_server1, oam_server2 |
| | Oracle Policy Manager | oam_policy_mgr1, oam_policy_mgr2 |
| IAMGovernanceDomain | Oracle Identity Governance | oim_server1, oim_server2 |
| | Oracle SOA Suite | soa_server1, soa_server2 |

# Storage Requirements for an Enterprise Deployment

Preparing the file system for an enterprise deployment involves understanding the requirements for local and shared storage, as well as the terminology that is used to reference important directories and file locations during the installation and configuration of the enterprise topology.

- About Preparing Storage for an Enterprise Deployment
- About the Recommended Directory Structure for an Enterprise Deployment
- Summary of the Storage File Systems and Corresponding Mount Points for Containers
- Mapping of File Systems to Hosts
- File System and Directory Variables Used in This Guide

## About Preparing Storage for an Enterprise Deployment

It is important to set up your storage in a way that makes the enterprise deployment easy to understand, configure, and manage.

This section provides an overview of the process of preparing storage for an enterprise deployment. Oracle recommends setting up your storage according to information in this chapter. The terminology defined in this chapter is used in the diagrams and procedures throughout the guide.

Use this chapter as a reference to understand the directory variables that are used in the installation and configuration procedures.

Other directory layouts are possible and supported, but the model adopted in this guide was designed for maximum availability, providing both the best isolation of components and symmetry in the configuration and facilitating backup and disaster recovery. The rest of the document uses this directory structure and directory terminology.

# About the Recommended Directory Structure for an Enterprise Deployment

The diagrams in this section show the directory structure for a typical Oracle Fusion Middleware enterprise deployment.

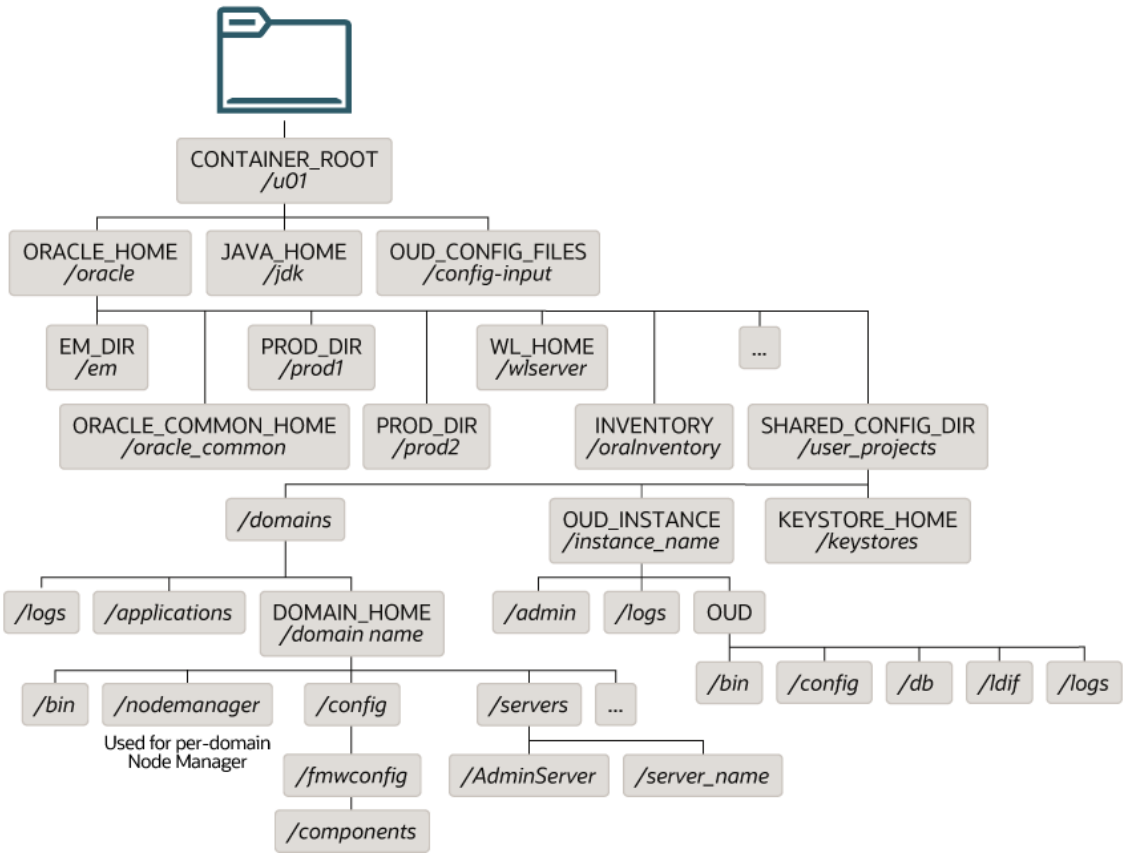
The directories shown in the diagrams contain binary files that are part of the pre-built container images, domain-specific files generated through the domain configuration process, as well as domain configuration files that are propagated to the various host computers through the Oracle WebLogic Server `pack` and `unpack` commands.

The diagrams are used to indicate:

- Figure 4-13 shows the resulting directory structure on the shared storage device after you have installed and configured a typical Oracle Fusion Middleware enterprise deployment. The shared storage directories are accessible by the application tier host computers.
- Figure 4-16 shows the resulting directory structure on the local storage device for a typical web tier host after you have installed and configured an Oracle Fusion Middleware enterprise deployment. Note that the software binaries (in the Oracle home) are installed on the local storage device for each web tier host.

Where applicable, the diagrams also include the standard variables used to reference the directory locations in the installation and configuration procedures in this guide.

**Figure 4-13 Recommended Shared Storage Directory Structure for an Enterprise Deployment**



See About the Node Manager Configuration in a Typical Enterprise Deployment.

**Figure 4-14 Recommended Directory Structure for Oracle Identity Role Intelligence**

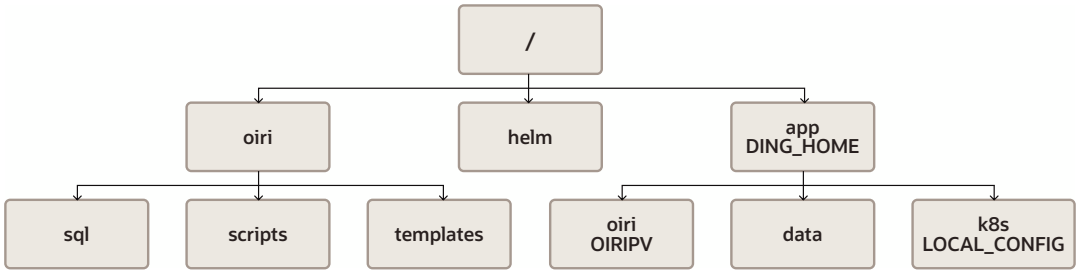



**Figure 4-15 Recommended Directory Structure for Oracle Advanced Authentication**

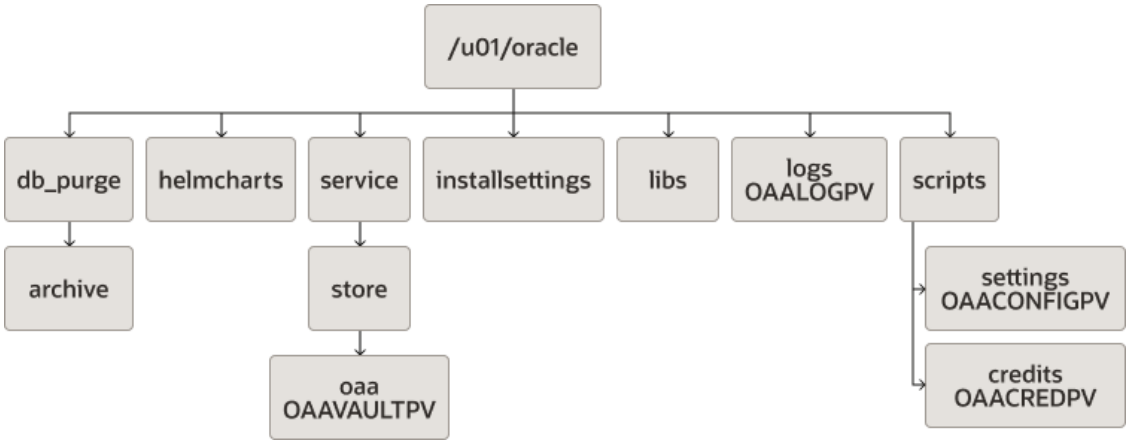
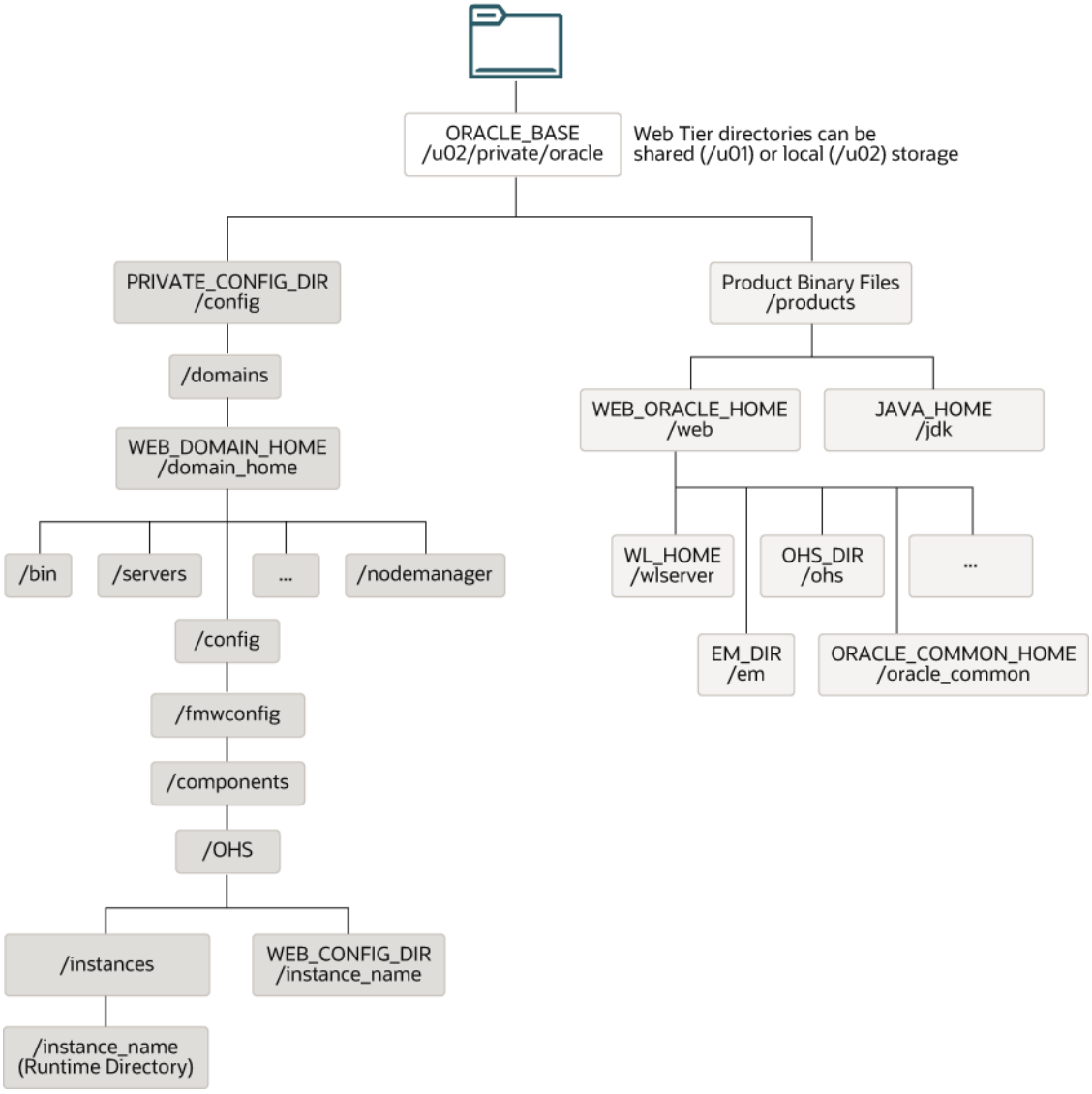


**Figure 4-16 Recommended Local Storage Directory Structure for a Web Tier Host Computer in an Enterprise Deployment**




# Summary of the Storage File Systems and Corresponding Mount Points for Containers

Oracle Identity and Access Management uses shared storage mounted to individual Kubernetes containers. This ensures that wherever a container is started, it always has access to its file system data.

To organize the enterprise deployment software on the appliance, you create a number of projects, file system shares are then created inside these projecst on the appliance. These shares can then be mounted directly to the Kubernetes containers.

To ease management, these file systems are also mounted to an administration host which you use for deployment and management operations. When mounted to the administration host, they are required to be mounted only for the duration of the management task being performed. Oracle recommends that you do not mount the file systems beyond this time.

Binary files for the Oracle HTTP Server can be created on local volumes or shared volumes mounted exclusively to the web tier hosts.

Figure 4-17 shows the recommended physical directory structure on the shared storage device.

Table 4-3 shows how the shares on the storage device map to the mount points inside the Kubernetes containers.

**Figure 4-17 Physical Structure of the Shares on the Shared File System for Virtual Deployments**

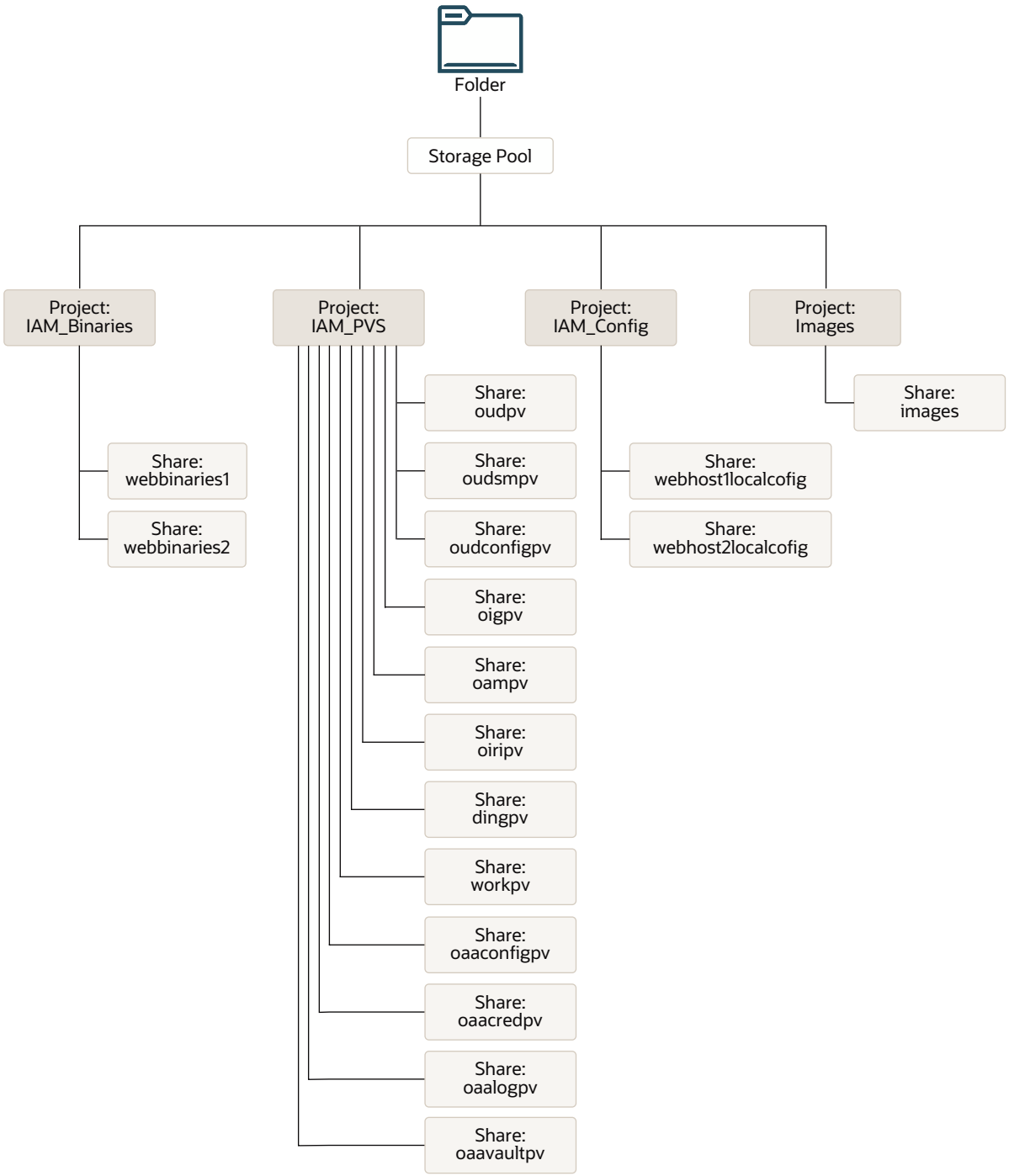


Figure 4-17 illustrates the physical structure of the shares on the shared appliance.

**Table 4-2 Summary of Storage Projects for Virtual Servers**

| Project | Size |
|---|---|
| IAMBINARIES | 10 GB |
| IAMPVS | 300 GB |
| IMAGES | 50 GB<br>**Note**: Size is dependent on how many docker images have to be stored, including versions. You should allow enough space for at least two versions of each image. This enables you to have the current version and the version to which you want to apply the patch.<br>You will not require an 'Images' project if you are using a container registry. |

# Mapping of File Systems to Hosts

This table summarizes how the physical directories on the storage appliance are mapped to the mount points on each container or host.

**Table 4-3 Mapping the Shares on the Appliance to Mount Points on Each Container or Host**

| Project | Share | Mount Point | Container/ Host | Mounted On | Privileges to Assign to User, Group, and Other | Actual Size | Description and Purpose |
|---|---|---|---|---|---|---|---|
| IAMBINARIES1 | webbinaries1 | /exports/IAMBINARIES1/webbinaries1 | WEBHOST1 | /u02/private/oracle/products | Read/Write | 10 GB | Local storage for the Oracle HTTP Server software binaries (Oracle Home). |
| IAMBINARIES2 | webbinaries2 | /exports/IAMBINARIES2/webbinaries2 | WEBHOST2 | /u02/private/oracle/products | Read/Write | 10 GB | Local storage for the Oracle HTTP Server software binaries (Oracle Home). |
| IAMPVS | oampv | /exports/IAMPVS/oampv | OAM | /u01/oracle/user_projects | Read/Write | 5 GB | Shared storage for the OAM domain. |
| IAMPVS | oigpv | /exports/IAMPVS/oigpv | OIG | /u01/oracle/user_projects | Read/Write | 5 GB | Shared storage for the OIG domain. |

**Table 4-3 (Cont.) Mapping the Shares on the Appliance to Mount Points on Each Container or Host**

| Project | Share | Mount Point | Container/ Host | Mounted On | Privileges to Assign to User, Group, and Other | Actual Size | Description and Purpose |
|---|---|---|---|---|---|---|---|
| IAMPVS | oudpv | /exports/IAMPVS/oudpv | LDAP | /u01/oracle/user_projects | Read/Write | 10 GB | Shared storage for OUD instances. This is where the Oracle home directory and product directories are installed. |
| IAMPVS | oudconfigpv | /exports/IAMPVS/oudconfigpv | LDAP | /u01/oracle/config-input | Read/Write | 5 GB | Shared storage for the configuration files required to set up OUD. |
| IAMPVS | oudsmpv | /exports/IAMPVS/oudsmpv | OUDSM | /u01/oracle/user_projects | Read/Write | 10 GB | Shared storage for OUDSM. |
| IAMPVS | oiripv | /exports/IAMPVS/oiripv | OIRI<br>OIRI-CLI | /app/oiri | Read/Write | 10 GB | Shared storage for OIRI. |
| IAMPVS | dingpv | /exports/IAMPVS/dingpv | SPARK-HISTORY (DING)<br>OIRI<br>OIRI-CLI | /app | Read/Write | 10 GB | Shared storage for Data Ingester. |
| IAMPVS | workpv | /exports/IAMPVS/workpv | OIRI-CLI<br>OIRI-DING-CLI | /app/k8s | Read/Write | 200 M | Working directory used for OIRI configuration/ management. |
| IAMPVS | oaaconfigpv | /exports/IAMPVS/oaaconfigpv | oaa-mgmt | /u01/oracle/scripts/settings | Read/Write | | Used to store the customized configuration file for installing OAA and OARM. |
| IAMPVS | oaacredpv | /exports/IAMPVS/oaacredpv | oaa-mgmt | /u01/oracle/scripts/cred | Read/Write | | Used to store credential files such as k8sconfig, helmconfig, trust.p12, and cert.p2. |
| IAMPVS | oaalogpv | /exports/IAMPVS/oaalogpv | oaa-mgmt | /u01/oracle/logs | Read/Write | | Used to store installation logs and status. |

**Table 4-3 (Cont.) Mapping the Shares on the Appliance to Mount Points on Each Container or Host**

| Project | Share | Mount Point | Container/ Host | Mounted On | Privileges to Assign to User, Group, and Other | Actual Size | Description and Purpose |
|---|---|---|---|---|---|---|---|
| IAMPVS | `oaavaultpv` | `/exports/ IAMPVS/ oaavaultpv` | oaa-mgmt | `/u01/oracle/ service/ store/oaa` | Read/Write | | Used to store the vault artifacts for a file-based vault. |
| IAMCONFIG | `webconfig1` | `/exports/ IAMCONFIG/ webconfig1` | WEBHOST1 | `/u02/ private/ oracle/ config` | Read/Write | 5 GB | Local storage for the domain configuration files used by WEBHOST1, if the private Managed Server domain directory resides on a shared storage. |
| IAMCONFIG | `webconfig2` | `/exports/ IAMCONFIG/ webconfig2` | WEBHOST2 | `/u02/ private/ oracle/ config` | Read/Write | 5 GB | Local storage for the domain configuration files used by WEBHOST2, if the private Managed Server domain directory resides on a shared storage. |
| IMAGES | `images` | `/exports/ IMAGES/images` | Kubernetes Workers | `/images` | Read/Write | 10 GB | Used to store the container images locally. This is optional. |

> **Note:**
>
> If required, after the configuration is complete, you can change the `binary` directories to **read only**, .

# File System and Directory Variables Used in This Guide

Understanding the file system directories and the directory variables used to reference these directories is essential for installing and configuring the enterprise deployment topology.

Table 4-4 lists the file system directories and the directory variables that are used to reference the directories on the application tier. Table 4-5 lists the file system directories and variables that are used to reference the directories on the web tier.

For additional information about mounting these directories when you use shared storage, see Creating and Mounting the Directories for an Enterprise Deployment.

Throughout this guide, the instructions for installing and configuring the topology refer to the directory locations that use the variables shown here.

You can also define operating system variables for each of the directories listed in this section. If you define system variables for the particular UNIX shell that you are using, you can then use the variables as they are used in this document, without having to map the variables to the actual values for your environment.

**Table 4-4 Sample Values for Key Directory Variables on the Application Tier**

| Directory Variable | Description | Relative Path | Sample Value on the Application Tier |
|---|---|---|---|
| *CONTAINER_ROOT* | The home directory of the container. The Oracle product binaries and the OUD setup files are present in this directory. | N/A | `/u01` |
| *ORACLE_HOME* | The read-only location for the product binaries. For the application tier host computers, it is stored on shared disk (PV).<br>The Oracle home is created when the container is deployed. In a Kubernetes deployment, *ORACLE_HOME* is the same as *CONTAINER_HOME*. | `CONTAINER_HOME/oracle` | `/u01/oracle` |
| *ORACLE_COMMON_HOME* | The directory within the Oracle Fusion Middleware Oracle home where common utilities, libraries, and other common Oracle Fusion Middleware products are stored. | *`ORACLE_HOME`*`/oracle_common` | `/u01/oracle/oracle_common` |
| *WL_HOME* | The directory within the Oracle home where the Oracle WebLogic Server software binaries are stored. | *`ORACLE_HOME`*`/wlserver` | `/u01/oracle/wlserver` |
| *PROD_DIR* | Individual product directories for each Oracle Fusion Middleware product that you install. | *`ORACLE_HOME`*`/`*`prod_dir`* | `/u01/oracle/`*`prod_dir`*<br>The product can be `soa`, `wcc`, `idm`, `bi`, or another value, depending on your enterprise deployment. |
| *EM_DIR* | The product directory used to store the Oracle Enterprise Manager Fusion Middleware Control software binaries. | *`ORACLE_HOME`*`/em` | `/u01/oracle/em` |
| *JAVA_HOME* | The location where the supported Java Development Kit (JDK) is installed. | *`CONTAINER_ROOT`*`/jdk` | `/u01/jdk` |
| *SHARED_CONFIG_DIR* | The shared parent directory for shared environment configuration files, including domain configuration, keystores, runtime artifacts, and application deployments | *`CONTAINER_HOME`*`/user_projects` | `/u01/oracle/user_projects` |
| *DOMAIN_HOME* | The domain home, which is installed on a shared disk. | *`SHARED_CONFIG_DIR`*`/domains/`*`domain_name`* | `/u01/oracle/user_projects/domains/`*`domain_name`*<br>In this example, replace *`domain_name`* with the name of the WebLogic Server domain. |

**Table 4-4 (Cont.) Sample Values for Key Directory Variables on the Application Tier**

| Directory Variable | Description | Relative Path | Sample Value on the Application Tier |
|---|---|---|---|
| *APPLICATION_HOME* | The Application home directory, which is installed on shared disk, so the directory is accessible by all the application tier host computers. | *SHARED_CONFIG_DIR*/applications/*domain_name* | /u01/oracle/user_projects/applications/*domain_name*<br>In this example, replace *domain_name* with the name of the WebLogic Server domain. |
| *KEYSTORE_HOME* | The shared location for custom certificates and keystores. | *SHARED_CONFIG_DIR*/keystores | /u01/oracle/user_projects/keystores |

**Table 4-5 Sample Values for Key Directory Variables on the Web Tier**

| Directory Variable | Description | Sample Value on the Web Tier |
|---|---|---|
| *WEB_ORACLE_HOME* | The read-only location for the Oracle HTTP Server product binaries. For the web tier host computers, this directory is stored on the local disk.<br>The Oracle home is created when you install the Oracle HTTP Server software . | /u02/private/oracle/products/web |
| *ORACLE_COMMON_HOME* | The directory within the Oracle HTTP Server Oracle home where common utilities, libraries, and other common Oracle Fusion Middleware products are stored. | /u02/private/oracle/products/web/oracle_common |
| *WL_HOME* | The directory within the Oracle home where the Oracle WebLogic Server software binaries are stored. | /u02/private/oracle/products/web/wlserver |
| *PROD_DIR* | Individual product directories for each Oracle Fusion Middleware product that you install. | /u02/private/oracle/products/web/ohs |
| *JAVA_HOME* | The location where you install the supported Java Development Kit (JDK). | /u02/private/oracle/products/jdk |
| *WEB_DOMAIN_HOME* | The Domain home for the standalone Oracle HTTP Server domain, which is created when you install Oracle HTTP Server on the local disk of each web tier host. | /u02/private/oracle/config/domains/*domain_name*<br>In this example, replace *domain_name* with the name of the WebLogic Server domain. |
| *WEB_CONFIG_DIR* | This is the location where you edit the Oracle HTTP Server configuration files (for example, `httpd.conf` and `moduleconf/*.conf`) on each web host.<br>Note that this directory is also referred to as the OHS Staging Directory. Changes made here are later propagated to the OHS Runtime Directory.<br>See Staging and Run-time Configuration Directories in the *Administering Oracle HTTP Server*. | /u02/private/oracle/config/domains/*domain_name*/config/fmwconfig/components/OHS/*instance*/*instance_name* |

# About Permissions

When containers are created, the files owned by the container are owned by the user with UID 1000 and group 1000.

If you have an existing user with the UID 1000 on your system, you will see all container services running as this user.

If you do not have an existing user/group with the UID/GID of 1000, you should create one.

When creating NFS shares for use by containers, ensure that the user/group with UID:GID of 1000:1000 has the write permission.

# Summary of Microservices and Clusters on the Application Tier

The application tier not only hosts the WebLogic components of Oracle Identity and Access Management, but it also hosts the Microservice components.

Depending upon the components you select, Oracle microservices consists of clusters shown in Table 4-6. These clusters function as active-active high availability configurations.

**Table 4-6 Oracle Microservices**

| Component | Function | Microservice |
|---|---|---|
| OIRI | Oracle Role Intelligence | oiri |
| OIRI | Oracle Role Intelligence GUI | oiri-ui |
| OIRI | Oracle Data Ingester | spark-history-server |
| OAA | Install OAA and Risk | oaa-mgmt |
| OAA | Main OAA functionality | oaa-svc |
| OAA + OARM | Policy Management APIs | oaa-policy |
| OAA + OARM | Administrative UI (OAA + OARM) | oaa-admin |
| OAA | • User Preferences User Interface<br>• MFA Authentication UI<br>• Forgot Password UI | oaa-spui |
| OAA | Fido Device Registration and Fido-based Authentication | oaa-factor-fido |
| OAA | Oracle Mobile Authenticator (OMA) OTP Authentication only | oaa-factor-totp |
| OAA | SMS Authentication only | oaa-factor-sms |
| OAA | Email Authentication only | oaa-factor-email |
| OAA | Yubikey-based Authentication only | oaa-factor-yotp |
| OAA | Knowledge-based Challenge Registration and Authentication | oaa-factor-kba |
| OAA | OMA Push Registration and Authentication | oaa-factor-push |
| OARM | Customer Care APIs | risk-cc |

**Table 4-6 (Cont.) Oracle Microservices**

| Component | Function | Microservice |
|---|---|---|
| OARM | Risk Evaluation APIs | risk-engine |

# About the Forgotten Password Functionality

In Oracle 11*g*, the mechanism to reset passwords was provided by Oracle Identity Governance. In Oracle 12*c*, you have two possibilities - by integrating Oracle Access Management (OAM) and Oracle Identity Governance (OIG) or by using Oracle Access Management.

- Oracle Access Management (OAM) and Oracle Identity Governance (OIG) integration:

  In this scenario, when the users first sign in, they can set a number of challenge questions stored in OIG. If users forget their password, they can answer the challenge questions. If they answer their challenge questions successfully, they will be given the option to reset their password by using OIG.

- Oracle Access Management:

  In this scenario, OAM is wired to the Oracle User Messaging Service. Each user is associated with either a telephone number and or an email address. When users request for a forgotten password link, they are sent a one time PIN which they can enter into the application to reset the password.

This guide describes how to set up both the scenarios.

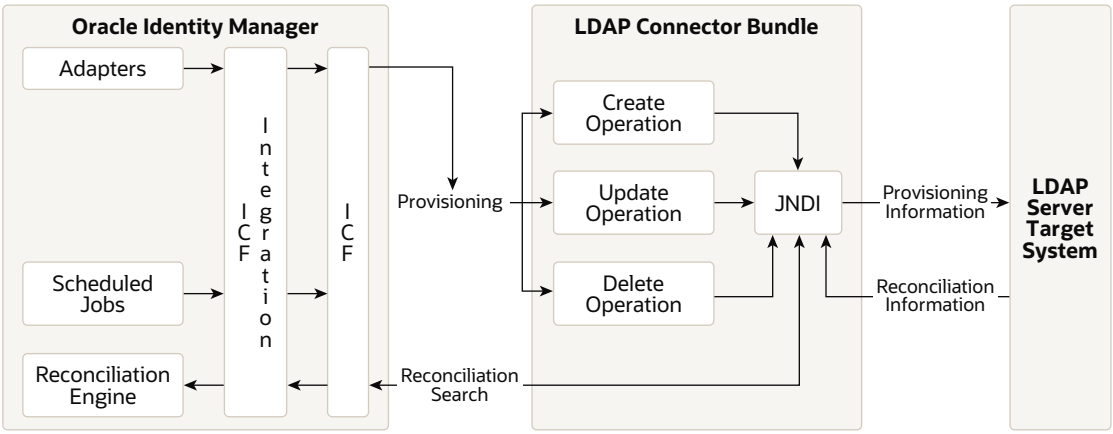
# Integrating Oracle LDAP, Oracle Access Manager, and Oracle Identity Governance

Integration of Oracle Identity Manager and Oracle Access Manager with LDAP directories is done by using LDAP Connector.

To enable termination of user sessions upon disablement or termination of a user, download the 12.2.1.3 version of the LDAP Connector.

This section describes how to obtain, install, and configure the Oracle Connector for LDAP.

### About the Oracle Connector



The LDAP Connector is implemented by using the Identity Connector Framework (ICF). The ICF is a component that provides basic reconciliation and provisioning operations that are common to all Oracle Identity Manager connectors. The ICF is shipped along with Oracle Identity Manager. Therefore, you need not configure or modify the ICF.

The LDAP Connector uses JNDI to access the target system. This connector can be configured to run in one of the following modes:

- **Identity Reconciliation**: Identity reconciliation is also known as authoritative or trusted source reconciliation. In this form of reconciliation, OIM Users are created or updated corresponding to the creation of and updates to users on the target system.

  > **Note:**
  >
  > The identity reconciliation mode supports the reconciliation of user objects only.

- **Account Management**: Account management is also known as target resource management. This mode of the connector enables provisioning and target resource reconciliation.

### Provisioning

Provisioning involves creating, updating, or deleting users, groups, roles, and organizational units (OUs) on the target system through Oracle Identity Manager.

When you allocate (or provision) a target system resource to an OIM user, the operation results in the creation of an account on the target system for that user. In the Oracle Identity Manager context, the term **provisioning** is also used to mean updates (for example enabling or disabling) made to the target system account through Oracle Identity Manager.

Users and organizations are organized in a hierarchical format on the target system. Before you can provision users to (that is, create users in) the required organizational units (OUs) on the target system, you must fetch into Oracle Identity Manager the list of OUs used on the target system. This is achieved by using the LDAP Connector OU lookup Reconciliation scheduled job for lookup synchronization.

Similarly, before you can provision users to the required groups or roles on the target system, you must fetch into Oracle Identity Manager the list of all groups and roles used on the target system. This is achieved by using the LDAP Connector Group Lookup Reconciliation and LDAP Connector Role Lookup Recon scheduled jobs for lookup synchronization.

### Target resource reconciliation

To perform target resource reconciliation, the LDAP Connector User Search Reconciliation or LDAP Connector User Sync Reconciliation scheduled jobs is used. The connector applies filters to locate users to be reconciled from the target system and then fetches the attribute values of these users.

Depending on the data that you want to reconcile, you use different scheduled jobs. For example, you use the LDAP Connector User Search Reconciliation scheduled job to reconcile user data in the target resource mode.

You can deploy the Oracle LDAP Connector either locally in Oracle Identity Manager or remotely in the connector server. A **connector server** enables remote execution of an Identity Connector. This guide explains the steps to install and configure the connector locally in Oracle Identity Manager.

# Roadmap for Implementing the Primary IAM Suite Topologies

This roadmap introduces you to the different tasks that need to be performed for implementing the primary IAM suite topologies on Kubernetes. This guide supports deployments on Oracle Kubernetes Engine (OKE), Oracle Cloud Native Environment, and the on-premises based Kubernetes deployments.

Regardless of the platform chosen, unless otherwise stated, the deployment steps are the same.

**Table 4-7 Roadmap for Implementing Primary IAM Suite Topologies on Kubernetes**

| Scenario | Tasks | For More Information, See |
|---|---|---|
| Creating an IAM Enterprise Deployment manually on commodity hardware | Know about a typical enterprise deployment. | About a Typical Enterprise Deployment<br>About the Kubernetes Deployment<br>About Oracle Identity Management on Kubernetes |
| | Procure resources for the enterprise deployment. | Procuring Resources for an On-Premises Enterprise Deployment<br>Procuring Resources for an Oracle Cloud Infrastructure Deployment |
| | Prepare your environment for enterprise deployment. | Preparing for an On-Premises Enterprise Deployment<br>Preparing the Oracle Cloud Infrastructure for an Enterprise Deployment |
| | Procure the required software. | Procuring Software for an Enterprise Deployment |
| | Prepare the existing database for the deployment. | Preparing an Existing Database for an Enterprise Deployment |
| | Configure Oracle Unified Directory. | Installing and Configuring Oracle Unified Directory |
| | Configure Oracle HTTP Server/Web Tier. | Installing and Configuring Oracle HTTP Server |
| | Install the WebLogic Operator for Kubernetes. | Installing the WebLogic Kubernetes Operator |
| | Create and configure the Oracle Fusion Middleware Infrastructure for Oracle Access Management. | Configuring Oracle Access Manager Using WDT |
| | Create and configure the Oracle Fusion Middleware Infrastructure for Oracle Identity Governance. | Configuring Oracle Identity Governance Using WDT |
| | Install and configure Oracle Identity Role Intelligence | Installing and Configuring Oracle Identity Role Intelligence |
| | Configure server migration settings. | Using Whole Server Migration and Service Migration in an Enterprise Deployment |

# Building Your Own Oracle Identity and Access Management Topology

These step-by-step instructions help you configure the two primary enterprise topologies for Oracle Identity and Access Management. However, Oracle recognizes that the requirements of your organization may vary, depending on the specific set of Oracle Fusion Middleware products you purchase and the specific types of applications you deploy.

For information on the primary enterprise topology for Oracle Identity and Access Management, see #unique_83/unique_83_Connect_42_CHDFBIDE.

In many cases, you can install and configure an alternative topology, one that includes additional components, or one that does not include all the Oracle Identity and Access Management products shown in the primary topology diagrams.

A few alternatives to the reference Oracle Identity and Access Management topologies you can implement by using the steps provided in this guide are:

- OAM only with an existing directory
- OIM only with an existing directory
- OAM/OIM integrated in a modular deployment
- OAM/OIM integrated with an existing directory
- OIRI integrated with OIG
- OAA integrated with OAM

# 5
# Disaster Recovery

Each product has a different disaster recovery strategy. This chapter explains the disaster recovery processes for each product.

This chapter includes the following topics:

- Kubernetes Cluster
  Two independent Kubernetes cluster will exist, one in each site. These Kubernetes clusters need not be symmetrical. However, both clusters should have the spare capacity to run an application in the event of a failover or a switchover.
- Oracle HTTP Server
  The Oracle HTTP Server configuration is static. Therefore, the Oracle HTTP Server will be installed independently on the primary and standby sites.
- Ingress
  The Ingress controller configuration is static. Therefore, the Ingress controller will be installed independently on the primary and standby sites.
- WebLogic Operator
  The Oracle WebLogic Operator configuration is static. Therefore, the Operator will be installed independently on the primary and standby sites.
- Oracle Unified Directory
- Oracle Access Manager
  Oracle Access Manager disaster recovery is achieved using the active/passive model. Currently, the use of Oracle Access Manager Multi-Datacenter is not supported in a Kubernetes environment.
- Oracle Identity Governance
  Disaster recovery for Oracle Identity Governance (OIG) is achieved using the active/passive model.
- Oracle Identity Role Intelligence
  Disaster recovery for Oracle Identity Role Intelligence (OIRI) is achieved using the active/passive model.
- Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator
  Disaster recovery for Oracle Advanced Authentication (OAA) Oracle Adaptive Risk Management (OARM) is achieved using the active/passive model.
- Prometheus, Grafana, Elastic Search and Grafana
  These products are not Oracle products. Therefore, the approach for disaster recovery for them is beyond the scope of this document. For details on how to enable disaster recovery for these products, see the appropriate product documentation.
- Roadmap for Setting Up Disaster Recovery
  This section provides a high-level summary of the steps required to set up disaster recovery for the entire Oracle Identity and Access Management suite.

# Kubernetes Cluster

Two independent Kubernetes cluster will exist, one in each site. These Kubernetes clusters need not be symmetrical. However, both clusters should have the spare capacity to run an application in the event of a failover or a switchover.

Each cluster is created and managed independently. Kubernetes scheduling will be responsible for ensuring that applications are started inside the cluster. There will be no external dependencies. All product disaster recovery solutions will be confined to processes running inside the cluster.

# Oracle HTTP Server

The Oracle HTTP Server configuration is static. Therefore, the Oracle HTTP Server will be installed independently on the primary and standby sites.

The configuration changes will be applied to each site manually and independently.

# Ingress

The Ingress controller configuration is static. Therefore, the Ingress controller will be installed independently on the primary and standby sites.

The configuration changes will be applied to each site manually and independently.

# WebLogic Operator

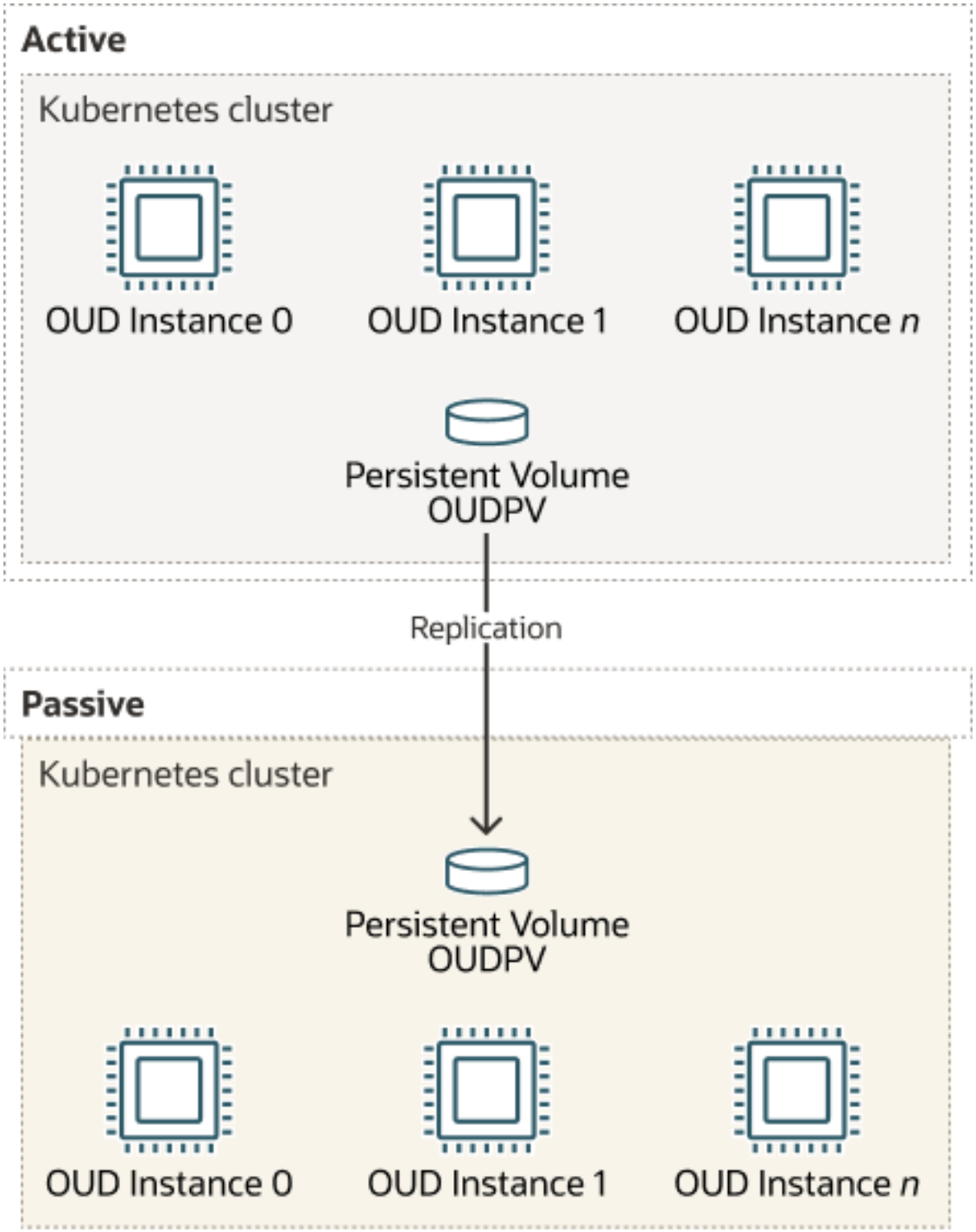
The Oracle WebLogic Operator configuration is static. Therefore, the Operator will be installed independently on the primary and standby sites.

The configuration changes will be applied to each site manually and independently.

# Oracle Unified Directory

In a traditional on-premise deployment, two independent Oracle Unified Directory (OUD) clusters will be set up on the primary and standby sites. Each site will be in a different replication group and the sites will be joined together by integrating both sites into the OUD replication agreement. This type of integration is not possible at this time in Kubernetes. The recommended approach in Kubernetes is to replicate the persistent volume from the primary site to the disaster recovery site. This is an active/passive solution.

**Figure 5-1 OUD Disaster Recovery**




There are several ways of achieving the active/passive solution:

- Disk based replication.
- Process driven replication, where a backup copy of the instance is created locally, shipped to the standby system, and then re-applied on the remote system. This option may introduce a lag where some data may not be available on the standby system.

The overall process of setting up disaster recovery for OUD includes the following steps:

1. Install OUD on the primary site using the standard procedures.
2. Install OUD on the secondary site using the standard procedures.
3. Delete the instance data from the standby site.
4. Enable the replication of the persistent volume between the primary and standby sites.

For more information, see Configuring Disaster Recovery for Oracle Unified Directory.

For a switchover, complete the following steps:

1. Stop the replication of the persistent volume.
2. Shut down OUD on the primary site.
3. Start up OUD on the standby site.
4. Enable the replication of the persistent volume in the reverse direction.

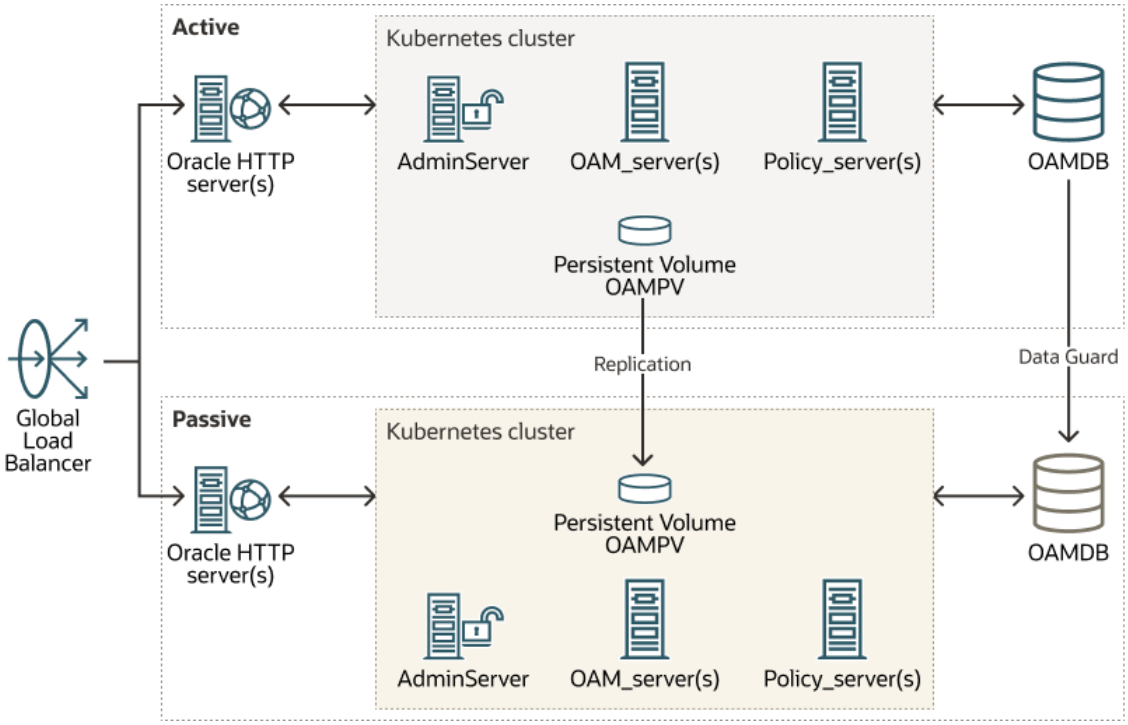
For failover, complete the following steps:

1. Stop the replication/application of the persistent volume.
2. Shut down OUD on the primary site, if available.
3. Start up OUD on the standby site.
4. Enable the replication of the persistent volume in the reverse direction if the site is available, or recreate the primary site using the above steps.

# Oracle Access Manager

Oracle Access Manager disaster recovery is achieved using the active/passive model. Currently, the use of Oracle Access Manager Multi-Datacenter is not supported in a Kubernetes environment.

**Figure 5-2 OAM Disaster Recovery**




Considerations for the disaster recovery solution:

- Every load balancer in the configuration (primary/standby) will contain the same SSL certificates.
- The Oracle Access Manager application data is stored within an Oracle Database, which is then replicated to the standby site using Oracle Data Guard.
- The configuration information (domain definitions) is replicated using the file system replication. This process involves using a replication tool such as 'rsync' on the persistent volumes. Because the configuration information changes rarely, it does not require frequent replication.
- After the configuration information is replicated to the standby site, all database connections will be modified to point to the standby database at the standby site.

For more information, Configuring Disaster Recovery for Oracle Access Manager .

The overall process of setting up disaster recovery for OAM includes the following steps:

1. Install OAM on the primary site using the standard procedures.
2. Install OAM on the standby site using the standard procedures, or backup the Kubernetes objects from the primary Kubernetes cluster and restore it on the standby cluster.
3. Modify the database service to be active only when the database is functioning in the roles of `PRIMARY` or `SNAPSHOT STANDBY` .
4. Delete the configuration data from the standby site (if you created a new environment).
5. Delete the database from the standby site.
6. Enable the replication of the persistent volume between the primary and standby sites.
7. Enable the replication of Data Guard between the primary and standby sites.

8. Replicate the application database service to the standby database.
9. Modify the database connection details on the standby site to use the standby database.

For a switchover, complete the following steps:

1. Stop the replication of the persistent volume.
2. Shut down OAM on the primary site.
3. Switch over to the standby database.
4. Start OAM on the standby site.
5. Enable the persistent volume replication in the reverse direction.
6. Enable the replication of Data Guard in the reverse direction.

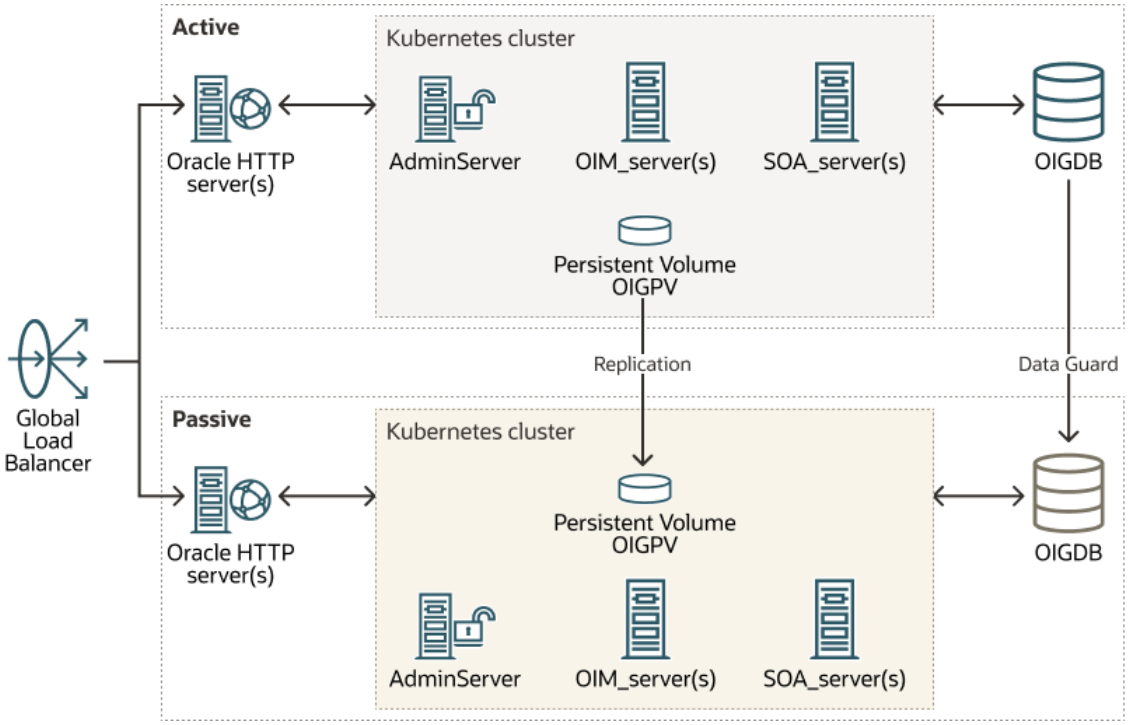
For failover, complete the following steps:

1. Stop the replication/application of the persistent volume.
2. Shut down OAM on the primary site, if available.
3. Activate/failover the standby database on the standby site.
4. Start OAM on the standby site.
5. Enable the replication of the persistent volume in the reverse direction if the site is available, or recreate the primary site using the steps above.
6. Enable the replication of Data Guard in the reverse direction if the site is available, or recreate the database on the old primary site using the above steps.

# Oracle Identity Governance

Disaster recovery for Oracle Identity Governance (OIG) is achieved using the active/passive model.

**Figure 5-3 OIG Disaster Recovery**




Considerations for the disaster recovery solution:

- The OIG application data is stored within an Oracle database, which is then replicated to the standby site using Oracle Data Guard.
- The configuration information (domain definitions) is replicated using the file system replication. This process involves using a replication tool such as 'rsync' on the persistent volumes. Because the configuration information changes rarely, it does not require frequent replication.
- After the configuration information is replicated to the standby site, all database connections will be modified to point to the standby database at the standby site.

For more information, see Configuring Disaster Recovery for Oracle Identity Governance.

The overall process of setting up disaster recovery for OIG includes the following steps:

1. Install OIG on the primary site using the standard procedures.
2. Install OIG on the standby site using the standard procedures, or backup the Kubernetes objects from the primary Kubernetes cluster and restore it on the standby cluster.
3. Modify the database service to be active only when the database is functioning in the roles of `PRIMARY` or `SNAPSHOT STANDBY`.
4. Delete the configuration data from the standby site (if you created a new environment).
5. Delete the database from the standby site.
6. Enable the replication of the persistent volume between the primary and standby sites.
7. Enable the replication of Data Guard between the primary and standby sites.
8. Replicate the application database service to the standby database.
9. Modify the database connection details on the standby site to use the standby database.

For a switchover, complete the following steps:

1. Stop the replication of the persistent volume.
2. Shut down OIG on the primary site.
3. Switch over to the standby database.
4. Start OIG on the standby site.
5. Enable the replication of the persistent volume in the reverse direction.
6. Enable the replication of Data Guard in the reverse direction.

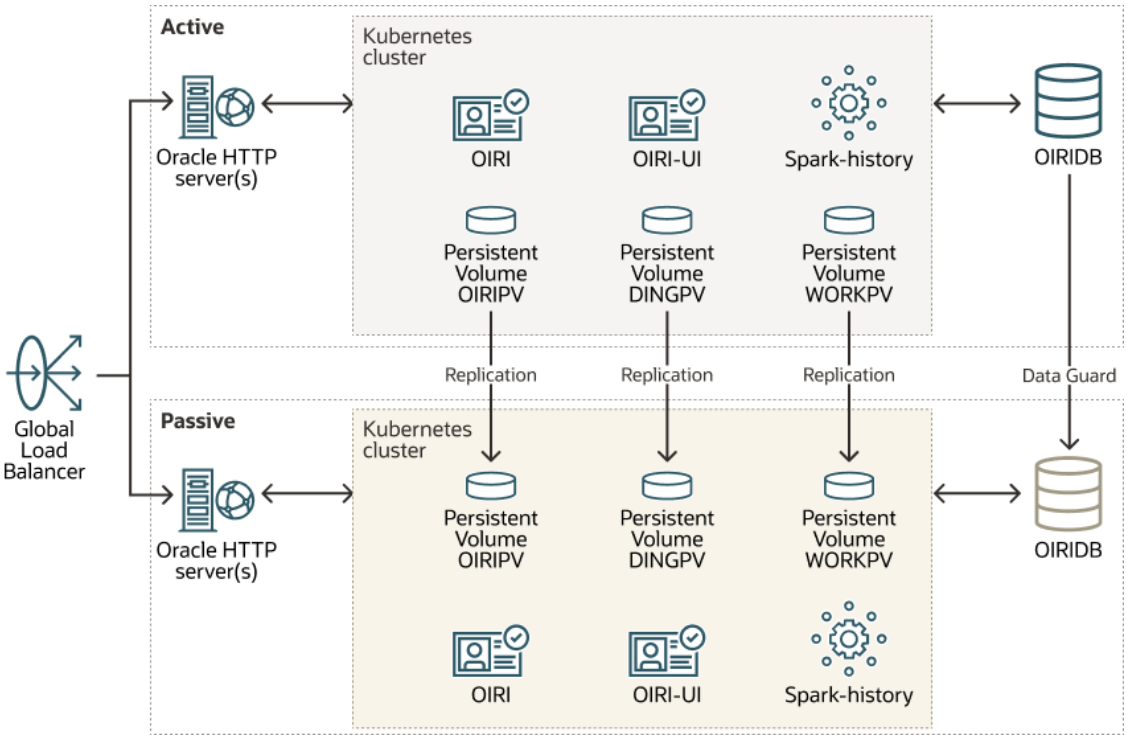
For failover, complete the following steps:

1. Stop the replication/application of the persistent volume.
2. Shut down OIG on the primary site, if available.
3. Activate/failover the standby database on the standby site.
4. Start OIG on the standby site.
5. Enable the replication of the persistent volume in the reverse direction if the site is available, or recreate the primary site using the above steps.
6. Enable the replication of Data Guard in the reverse direction if the site is available, or recreate the database on the old primary site using the above steps.

# Oracle Identity Role Intelligence

Disaster recovery for Oracle Identity Role Intelligence (OIRI) is achieved using the active/passive model.

**Figure 5-4 OIRI Disaster Recovery**



Considerations for the disaster recovery solution:

- The OIRI application data is stored within an Oracle database, which is then replicated to the standby site using Oracle Data Guard.
- The configuration information (domain definitions) is stored in the persistent volume replicated using the file system replication. This process involves using a replication tool such as 'rsync' on the persistent volumes. Because the configuration information changes rarely, it does not require frequent replication.
- After the configuration information is replicated to the standby site, all database connections will be modified to point to the standby database at the standby site.
- If any databases, such as the OIG database, have been switched over, then update the database connections to point to the standby versions.
- OIRI is closely integrated with the Kubernetes framework, allowing it to directly interact with the cluster for data-ingestion tasks. After it is deployed, OIRI has information about the cluster in which it is running. If you run OIRI from a different Kubernetes cluster, you need to update not only the database connection information but also the Kubernetes cluster information.

For more information, see Configuring Disaster Recovery for Oracle Identity Role Intelligence.

The overall process of setting up disaster recovery for OIRI includes the following steps:

1. Install OIRI on the primary site using the standard procedures.
2. Install OIRI on the standby site using the standard procedures, or backup the Kubernetes objects from the primary Kubernetes cluster and restore it on the standby cluster.
3. Modify the database service to be active only when the database is functioning in the roles of `PRIMARY` or `SNAPSHOT STANDBY`.
4. Delete the configuration data from the standby site (if you created a new environment).
5. Delete the database from the standby site.
6. Enable the replication of the persistent volume between the primary and standby sites.
7. Enable the replication of Data Guard between the primary and standby sites.
8. Replicate the application database service to the standby database.
9. Modify the database connection details on the standby site to use the standby database.
10. Modify the Kubernetes cluster connection details on the standby site to use the standby Kubernetes cluster.

For a switchover, complete the following steps:

1. Stop the replication of the persistent volume.
2. Shut down OIRI on the primary site.
3. Switch over to the standby database.
4. Start OIRI on the standby site.
5. Enable the replication of the persistent volume in the reverse direction.
6. Enable the replication of Data Guard in the reverse direction.

For failover, complete the following steps:

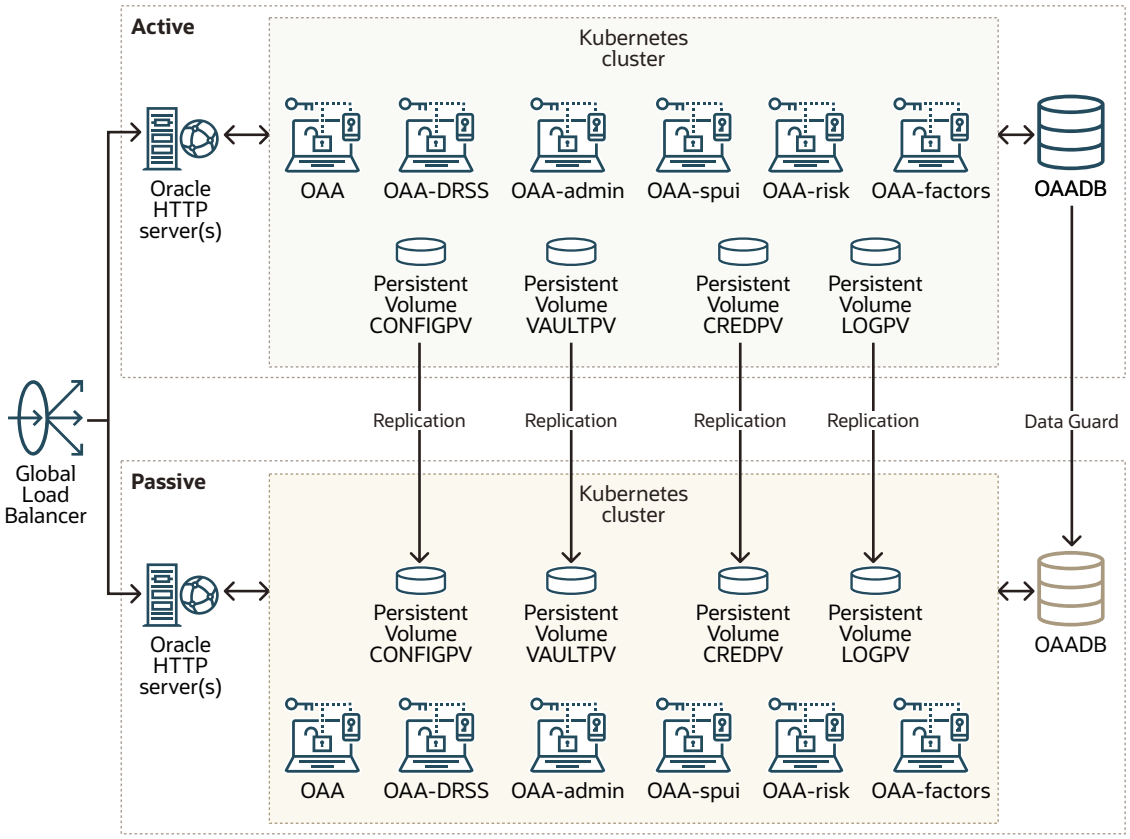
1. Stop the replication/application of the persistent volume.
2. Shut down OIRI on the primary site, if available.

3. Activate the standby database on the standby site.
4. Start OIRI on the standby site.
5. Enable the replication of the persistent volume in the reverse direction if the site is available, or recreate the primary site using the above steps.
6. Enable the replication of Data Guard in the reverse direction if the site is available, or recreate the database on the old primary site using the above steps.

# Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

Disaster recovery for Oracle Advanced Authentication (OAA) Oracle Adaptive Risk Management (OARM) is achieved using the active/passive model.

**Figure 5-5 OAA Disaster Recovery**




Considerations for the disaster recovery solution:

- Every load balancer in the configuration (primary/standby) will contain the same SSL certificates.
- The OAA application data is stored within an Oracle database, which is then replicated to the standby site using Oracle Data Guard.
- The configuration information and vaults are replicated using the file system replication. This process involves using a replication tool such as 'rsync' on the persistent volumes.

Because the configuration information changes rarely, it does not require frequent replication.

- After the configuration information is replicated to the standby site, all database connections will be directed to the standby database at the standby site.
- OAA stores configuration information in multiple Kubernetes objects. Therefore, it is simpler to redeploy the application in the standby site using the updated connection information rather than backup and restore the Kubernetes objects.
- OAA is closely integrated with the Kubernetes framework, allowing it to directly interact with the cluster for data-ingestion tasks. After it is deployed, OAA has information about the cluster in which it is running. If you run OAA from a different site, you need to update not only the database connection information but also the Kubernetes cluster information. This should not be replicated between the sites.
- OAA is closely coupled with Oracle Access Manager (OAM) for OAuth validation. Therefore, OAA should have access to OAM at all times. If you are using an OAM Multi-Datacenter configuration, the OAuth domain must exist in each OAM deployment. If you are using an active/passive OAM disaster recovery solution, the active OAA must have access to an active OAM site.

The overall process of setting up disaster recovery for OAA includes the following steps:

1. Install OAA on the primary site using the standard procedures.
2. Create the OAA management container on the standby system.
3. Enable the replication of the persistent volume between the primary and standby sites (excluding the Kubernetes configuration).
4. Enable the replication of Data Guard between the primary and standby sites.
5. Replicate the application database service to the standby database.
6. Modify the database service to be active only when the database is functioning in the roles of `PRIMARY` or `SNAPSHOT STANDBY`.
7. Convert the Data Guard database to a `SNAPSHOT STANDBY` database.
8. Update the database connection information in the `installOAA.properties` file and set `database.createschema=false`.
9. Install OAA on the standby site by rerunning `OAA.sh` to redeploy the application. It is critical to replicate the file system from primary to standby before you run this command (including the logs persistent volume). This command will then create the Kubernetes objects on the standby site.
10. Validate the configuration.
11. Shut down the OAA deployment and convert the Data Guard database back to a physical standby.

For more information, see Configuring Disaster Recovery for Oracle Advanced Authentication.

For a switchover, complete the following steps:

1. Stop the replication of the persistent volume.
2. Shut down OAA on the primary site.
3. Switch over to the standby database.
4. Start OAA on the standby site.
5. Enable the replication of the persistent volume in the reverse direction.

6. Enable the replication of Data Guard in the reverse direction.

For failover, complete the following steps:

1. Stop the replication/application of the persistent volume.
2. Shut down OAA on the primary site, if available.
3. Activate the standby database on the standby site.
4. Start OAA on the standby site.
5. Enable the replication of the persistent volume in the reverse direction if the site is available, or recreate the primary site using the above steps.
6. Enable the replication of Data Guard in the reverse direction if the site is available, or recreate the database on the old primary site using the above steps.

# Prometheus, Grafana, Elastic Search and Grafana

These products are not Oracle products. Therefore, the approach for disaster recovery for them is beyond the scope of this document. For details on how to enable disaster recovery for these products, see the appropriate product documentation.

# Roadmap for Setting Up Disaster Recovery

This section provides a high-level summary of the steps required to set up disaster recovery for the entire Oracle Identity and Access Management suite.

1. Enable communication between the primary and standby sites.
2. Set up the load balancers on each site to point to local installations.
3. Ensure that the load balancers use the same SSL certificates.
4. Set up Data Guard for the Oracle databases(s) to the standby site.
5. Install the Oracle HTTP Server (OHS) on the standby site.
6. Copy the OHS configuration from the primary to the standby site, changing routing to the standby site.
7. Install Ingress controller on the standby site (if used).
8. Install the Oracle WebLogic Operator on the standby site.
9. Deploy OUD on the standby site using Kubernetes snapshots or a fresh install and data deletion.
10. Enable the OUD persistent volume synchronization between the primary and standby sites.
11. Convert the standby database to a snapshot standby.
12. Deploy OAM on the standby site using Kubernetes snapshots or a fresh install and data deletion.
13. Enable the OAM persistent volume synchronization between the primary and standby sites.
14. Change the OAM database connection settings on the standby site, if needed.
15. Deploy the WebGate on OHS using the WebGate artifacts from the primary site.
16. Deploy OIG on the standby site using Kubernetes snapshots or a fresh install and data deletion.

17. Enable the OIG persistent volume synchronization between the primary and standby sites.
18. Change the OIG database connection settings on the standby site, if needed.
19. Deploy OIG on the standby site using Kubernetes snapshots or a fresh install and data deletion.
20. Enable the OIRI persistent volume synchronization between the primary and standby sites.
21. Change the OIRI database connection settings on the standby site, if needed.
22. Change the OIRI Kubernetes cluster connection settings on the standby site.
23. If not already started, start OAM against the snapshot standby database.
24. Start the OAA Management Container and configure it for the local cluster.
25. Enable the OAA persistent volume synchronization between the primary and standby sites.
26. Change the OAA database connection settings in `installOAA.properties` on the standby site, if needed.
27. Redeploy OAA on the standby site using `OAA.sh`.
28. Validate that the environment is working as required.
29. Enable the replication of the regular file system for each deployed product.
30. Convert the standby database back to a physical standby.

# Part II

# Preparing for an Enterprise Deployment

You have to perform several tasks to prepare for an Oracle Identity and Access Management enterprise deployment. Learn about the preparation required for both on-premises enterprise deployment and an Oracle Cloud Infrastructure (OCI) deployment.

This part of the guide contains the following chapters:

- Procuring Resources for an On-Premises Enterprise Deployment
  It is essential to procure the required hardware resources before you configure the on-premises Oracle Identity and Access Management reference topology. These resources include load balancer, host computers, and operating systems.
- Procuring Resources for an Oracle Cloud Infrastructure Deployment
  Before you deploy Oracle Identity and Access Management on Oracle Cloud Infrastructure (OCI), you need to ensure that you have sufficient OCI resources at your disposal.
- Procuring Software for an Enterprise Deployment
  Procure the required software such as the software distributions, the container images, the WebLogic Kubernetes Operator, and the appropriate connector bundle.
- Preparing for an On-Premises Enterprise Deployment
  An on-premises enterprise deployment involves preparing the hardware load balancer and ports, file system, operating system, and Kubernetes cluster.
- Preparing the Oracle Cloud Infrastructure for an Enterprise Deployment
  If you plan to deploy Identity and Access Management on Oracle Cloud Infrastructure (OCI) using the Oracle Container Engine for Kubernetes, you have to configure OCI to facilitate the deployment. Create the required OCI components to perform the deployment.
- Preparing an Existing Database for an Enterprise Deployment
  As part of preparing an existing database for an enterprise deployment, you should ensure that the database meets specific requirements. Other tasks include creating database services, using SecureFiles for large objects in the database, and creating database backup strategies. In a Kubernetes deployment, the database should reside outside of the cluster.

6

# Procuring Resources for an On-Premises Enterprise Deployment

It is essential to procure the required hardware resources before you configure the on-premises Oracle Identity and Access Management reference topology. These resources include load balancer, host computers, and operating systems.

This chapter includes the following topics:

- Load Balancer Requirements
  The enterprise topology uses an external load balancer.
- Host Computer Requirements
  Ensure that the host computers you procure are configured to support the enterprise deployment topologies.
- Operating System Requirements
  The Oracle Fusion Middleware software products and components that are described in this guide are certified on various operating systems and platforms.
- About Private Networks
  A private network enables you to keep inter-application communications within the private network, providing communication that is both faster and more secure. By keeping inter-application traffic inside the private network, you do not expose traffic to the internet. To use a private network, you have to create a private VLAN.

## Load Balancer Requirements

The enterprise topology uses an external load balancer.

The features of the external load balancer are:

- Ability to load-balance traffic to a pool of real servers through a virtual host name: Clients access services by using the virtual host name (instead of using actual host names). The load balancer can then load balance requests to the servers in the pool.
- Port translation configuration should be possible so that incoming requests on the virtual host name and port are directed to a different port on the backend servers.
- Monitoring of ports on the servers in the pool to determine availability of a service.
- Ability to configure names and ports on your external load balancer. The virtual server names and ports must meet the following requirements:
  - The load balancer should allow configuration of multiple virtual servers. For each virtual server, the load balancer should allow configuration of traffic management on more than one port. For example, for Oracle HTTP Server in the web tier, the load balancer needs to be configured with a virtual server and ports for HTTP and HTTPS traffic.
  - The virtual server names must be associated with IP addresses and be part of your DNS. Clients must be able to access the external load balancer through the virtual server names.
- Ability to detect node failures and immediately stop routing traffic to the failed node.

- It is highly recommended that you configure the load balancer to be in fault-tolerant mode.
- It is highly recommended that you configure the load balancer virtual server to return immediately to the calling client when the backend services to which it forwards traffic are unavailable. This is preferred over the client disconnecting on its own after a timeout based on the TCP/IP settings on the client machine.
- Ability to maintain sticky connections to components. Examples of this include cookie-based persistence, IP-based persistence, and so on.
- The load balancer should be able to terminate SSL requests at the load balancer and forward traffic to the backend real servers by using the equivalent non-SSL protocol (for example, HTTPS to HTTP).
- SSL acceleration (this feature is recommended, but not required for the enterprise topology).

# Host Computer Requirements

Ensure that the host computers you procure are configured to support the enterprise deployment topologies.

- General Considerations for Enterprise Deployment Host Computers
  This section specifies the general considerations that are required for the enterprise deployment host computers.
- Reviewing the Oracle Fusion Middleware System Requirements
  The system requirements information help you ensure that the environment meets the necessary minimum requirements.
- Typical Memory, File Descriptors, and Processes Required for an Enterprise Deployment
  This section specifies the typical memory, number of file descriptors, and operating system processes and tasks details that are required for an enterprise deployment.
- Typical Disk Space Requirements for an Enterprise Deployment
  This section specifies the disk space that is typically required for this enterprise deployment.

## General Considerations for Enterprise Deployment Host Computers

This section specifies the general considerations that are required for the enterprise deployment host computers.

Before you start the process of configuring an Oracle Fusion Middleware enterprise deployment, you must perform the appropriate capacity planning to determine the number of nodes, CPUs, and memory requirements for each node depending on the specific system's load as well as the throughput and response requirements. These requirements vary for each application or custom Oracle Identity and Access Management system being used.

In a Kubernetes deployment, the container that runs the Oracle FMW product is placed onto a Kubernetes worker node which must have sufficient capacity to run the service.

It is easy to assume that you will have the same number of worker nodes as you would have in a traditional enterprise deployment. However, to make the best use of Kubernetes, you should have a cluster of nodes which have enough redundant capacity for scaling the topology and managing unplanned outages.

This chapter provides general guidelines and information that help you determine the host computer requirements. It does not replace the need to perform capacity planning for your specific production environment.

# Reviewing the Oracle Fusion Middleware System Requirements

The system requirements information help you ensure that the environment meets the necessary minimum requirements.

> **Note:**
>
> These requirements are based on the Bare Metal Fusion Middleware deployments. Additional capacity, including memory and CPU capacity, should be added to manage Kubernetes overheads.

Review the *Oracle Fusion Middleware System Requirements and Specifications* to ensure that your environment meets the minimum installation requirements for the products that you are installing.

The Requirements and Specifications document contains information about general Oracle Fusion Middleware hardware and software requirements, minimum disk space and memory requirements, database schema requirements, and the required operating system libraries and packages.

It also provides some general guidelines for estimating the memory requirements for your Oracle Fusion Middleware deployment.

# Typical Memory, File Descriptors, and Processes Required for an Enterprise Deployment

This section specifies the typical memory, number of file descriptors, and operating system processes and tasks details that are required for an enterprise deployment.

The following table summarizes the memory, file descriptors, and processes required for the Administration Server and each of the Managed Servers computers in a typical Oracle Identity and Access Management enterprise deployment. These values are provided as an example only, but they can be used to estimate the minimum amount of memory required for an initial enterprise deployment.

The example in this topic reflects the minimum requirements for configuring the Managed Servers and other services required on OAMHOST1, as depicted in the reference topologies.

When you procure systems, use the information in the **Approximate Top Memory** column as a guide when determining the minimum physical memory that each host computer should have available.

After you procure the host computer hardware and verify the operating system requirements, review the software configuration to be sure that the operating system settings are configured to accommodate the number of open files listed in the **File Descriptors** column and the number processes listed in the **Operating System Processes and Tasks** column.

See Setting the Open File Limit and Number of Processes Settings on UNIX Systems.

| Managed Server, Utility, or Service | Approximate Top Memory | Number of File Descriptors | Operating System Processes and Tasks |
|---|---|---|---|
| Access Administration Server | 3.5 GB | 2300 | 180 |

| Managed Server, Utility, or Service | Approximate Top Memory | Number of File Descriptors | Operating System Processes and Tasks |
|---|---|---|---|
| Governance Administration Server | 3.5 GB | 2100 | 100 |
| soa_server | 2.0 GB | 1400 | 210 |
| oim_server | 8.0 GB | 1400 | 190 |
| oam_server | 4.0 GB | 2000 | 170 |
| oam_policy_mgr | 2.0 GB | 1700 | 160 |
| WLST (connection to the Node Manager) | 1.5 GB | 910 | 20 |
| Node Manager (per domain) | 1.0 GB | 720 | 15 |
| TOTAL | 22.0 GB* | 14430 | 805 |

* Approximate total.

> **Note:**
>
> The above figures are per service, which translates to a pod. The figures are for each occurrence of a service. So, if you had a highly available OAM cluster, then you would need 4GB of memory per instance. That is 2 x 4G for a minimum OAM HA configuration. If you were to have two worker nodes running OAM with one node running the Administration Server, two nodes each running one OAM server and policy server. Then, as a starting point, each worker node would require 9.5GB of memory.
>
> The above worker nodes do not include the 20pct overhead required by Kubernetes. So, using the above example, each worker node should be configured with 12GB of memory (figures rounded up).

## Typical Disk Space Requirements for an Enterprise Deployment

This section specifies the disk space that is typically required for this enterprise deployment.

For the latest disk space requirements for the Oracle Fusion Middleware 12c (12.2.1.4.0) products, including the Oracle Identity and Access Management products, review the *Oracle Fusion Middleware System Requirements and Specifications*.

In addition, the following table summarizes the disk space that is typically required for an Oracle Identity and Access Management enterprise deployment.

Use the this information and the information in Preparing the File System for an Enterprise Deployment to determine the disk space requirements required for your deployment.

| Server | Disk |
|---|---|
| Database | nXm<br>n = number of disks, at least 4 (striped as one disk)<br>m = size of the disk (minimum of 30 GB) |
| WEBHOST*n* | 10 GB |

| Server | Disk |
|---|---|
| OAMHOST*n* | 10 GB* |
| OIMHOST*n* | 10 GB* |
| LDAPHOST*n* | 10 GB* |

# Operating System Requirements

The Oracle Fusion Middleware software products and components that are described in this guide are certified on various operating systems and platforms.

For more information about the operating system requirements, see Oracle Fusion Middleware System Requirements and Specifications.

> **Note:**
>
> This guide focuses on the implementation of the enterprise deployment reference topology on Oracle Linux systems.
>
> The topology can be implemented on any certified, supported operating system, but the examples in this guide typically show the commands and configuration steps as they should be performed by using the bash shell on Oracle Linux.

# About Private Networks

A private network enables you to keep inter-application communications within the private network, providing communication that is both faster and more secure. By keeping inter-application traffic inside the private network, you do not expose traffic to the internet. To use a private network, you have to create a private VLAN.

7

# Procuring Resources for an Oracle Cloud Infrastructure Deployment

Before you deploy Oracle Identity and Access Management on Oracle Cloud Infrastructure (OCI), you need to ensure that you have sufficient OCI resources at your disposal.

For information about the resources, see Preparing the Oracle Cloud Infrastructure for an Enterprise Deployment.

This chapter includes the following topics:

- Procuring Resources for OCI
  It is important to understand the resource requirements for an Oracle Cloud Infrastructure deployment. These resources include load balancer, compute instances, network, gateways, OKE, and databases.
- Sizing
  The sizing guidelines provide the performance recommendations and sizing requirements for Oracle Identity and Access Management, Release 12.2.1.4.0.

## Procuring Resources for OCI

It is important to understand the resource requirements for an Oracle Cloud Infrastructure deployment. These resources include load balancer, compute instances, network, gateways, OKE, and databases.

For an illustration of the OCI layout depicting the use of these resources, see Figure 10-1.

- Load Balancer Requirements
  You will require two load balancers. One for internal traffic and the other for external traffic. The Shape of the load balancer should be sufficient for your expected traffic volume.
- Compute Instances
  You will require a minimum of three compute instances. A bastion node and two Web Tiers.
- Network
  You will require one Virtual Cloud Network (VCN). This network is sub-divided into several subnets to increase security. Each subnet has security lists and route tables.
- Gateway
  You require one public gateway and one service gateway.
- Container Engine for Kubernetes (OKE)
  OKE comprises a Kubernetes control plane which is managed by Oracle and many Kubernetes worker nodes that are used to host the Kubernetes pods.
- Database
  The number of databases you require depends on the disaster recovery strategy you plan to use. If you have a traditional Active/Passive solution, then you can use a single container database (CDB) with two pluggable databases (PDB).

## Load Balancer Requirements

You will require two load balancers. One for internal traffic and the other for external traffic. The Shape of the load balancer should be sufficient for your expected traffic volume.

## Compute Instances

You will require a minimum of three compute instances. A bastion node and two Web Tiers.

- Bastion Node: The bastion node is used to set up the environment and to provide you access to the internal resources in the Kubernetes cluster; you cannot access them directly. The bastion node may be the smallest shape available as it does not perform any day-to-day work. The bastion node can also be used as the administrative node for Oracle Identity Role Intelligence.
- Web Tier Nodes: You will require a minimum of two web tier compute instances. These instances require sufficient resources to handle the expected traffic flow. You can size them the same way as their on-premise equivalents.

## Network

You will require one Virtual Cloud Network (VCN). This network is sub-divided into several subnets to increase security. Each subnet has security lists and route tables.

Oracle virtual cloud networks (VCNs) provide customizable and private cloud networks in Oracle Cloud Infrastructure (OCI). Just like a traditional data center network, the VCN provides you complete control over your cloud networking environment. You can assign private IP address spaces, create subnets and route tables, and configure stateful firewalls.

## Gateway

You require one public gateway and one service gateway.

**Public Gateway**: A public gateway is an optional virtual router you can add to your VCN to enable direct connectivity to the internet. The gateway supports connections initiated from within the VCN (Egress) and connections initiated from the internet (Ingress).

**Service Gateway**: A service gateway enables cloud resources without public IP addresses to privately access Oracle services.

## Container Engine for Kubernetes (OKE)

OKE comprises a Kubernetes control plane which is managed by Oracle and many Kubernetes worker nodes that are used to host the Kubernetes pods.

Each deployment creates a number of pods. You will require sufficient worker nodes of a large enough shape to host your deployment.

### OKE Cluster

You require only one OKE cluster for an enterprise deployment.

### OKE Worker Nodes

You require sufficient OKE nodes to host the Identity and Access Management deployment. The shape of these nodes depend on the capacity requirements and the number of worker

nodes you want to use. As a reference, you can expect to run the following Kubernetes pods in a complete enterprise deployment:

**Table 7-1 Required Kubernetes Pods in an Enterprise Deployment**

| Pod | Quantity |
|---|---|
| OUD Server | 2 |
| WebLogic Operator | 1 |
| OAM Helper | 1 |
| OAM Admin Server | 1 |
| OAM Servers | 2 |
| OAM Policy Servers | 2 |
| OIG Helper | 1 |
| OIG Admin Server | 1 |
| OIG Servers | 2 |
| OIRI Servers | 2 |
| OIRI UI Servers | 2 |
| Spark Servers | 2 |
| OAA Cache | 6 |
| OAA Servers | 2 |
| OAA Administration Servers | 2 |
| OAA Policy Servers | 2 |
| OAA SPUI Servers | 2 |
| OAA Authentication Factors | 2 per factor |
| OAA RISK Analysis | 2 |
| OAA Risk Console | 2 |

> **Note:**
>
> This table shows the recommended minimum values for Oracle components that are deployed in the topology.

## Database

The number of databases you require depends on the disaster recovery strategy you plan to use. If you have a traditional Active/Passive solution, then you can use a single container database (CDB) with two pluggable databases (PDB).

If you are using OIRI, you will need a separate database to hold the analytical information. This should be a separate database because the processing workload is different to the core Identity Management components.

If the disaster recovery strategy is to use Oracle Access Manager Active/Active and Oracle Identity Governance Active/Passive, then you will require two separate databases.

The databases you create should be highly available real application cluster databases. For more information about the database requirements, see Preparing an Existing Database for an Enterprise Deployment.

# Sizing

The sizing guidelines provide the performance recommendations and sizing requirements for Oracle Identity and Access Management, Release 12.2.1.4.0.

For sizing guidelines, see Deep Dive into Oracle Access Management 12.2.1.4.0 Performance.

8

# Procuring Software for an Enterprise Deployment

Procure the required software such as the software distributions, the container images, the WebLogic Kubernetes Operator, and the appropriate connector bundle.

This chapter includes the following topics:

- Identifying and Obtaining Software Distributions for an Enterprise Deployment
  Before you begin to install and configure an enterprise topology, you must download the container images from Oracle Support.
- About Container Image Names
  Container images have names in a specific format depending on whether you have pre-loaded them or pulled them from a container registry.
- Obtaining Software from the Oracle Container Registry
  Before you download container images from the Oracle Container Registry, you must first log in to the Oracle Container Registry using your support credentials and accept the license agreements for each container you want to deploy.
- Downloading Images from a Container Registry
  If you use a container registry, you can pull images from the registry either on-demand or in advance. Pulling images in advance results in a faster deployment but means that you will have the images available on all worker nodes regardless of whether you are running a container on that worker node or not.
- Staging Container Images
- Logging in to GitHub
  The examples below require that you pull images and scripts from `github.com`, a public repository. To successfully pull images from GitHub, should log in to the repository.
- Staging the WebLogic Operator for Kubernetes
  Use the WebLogic Operator for Kubernetes to deploy Oracle Access Manager or Oracle Identity Governance. There are two parts to the WebLogic Operator: the WebLogic Operator Container Image and the WebLogic Operator deployment scripts.
- Staging the Code Repository
  Oracle provides a sample code repository to deploy Oracle Identity and Access Management in Kubernetes. The procedures explained in this guide use the sample code repository extensively. Download the sample code repository to a temporary work directory on your configuration host.
- Downloading the Oracle Connector Bundle for Oracle Identity Governance
  If you are planning to integrate Oracle Identity Governance with outside systems, you have to download the appropriate connector bundle. In an enterprise deployment, the LDAP connector is used to integrate with Oracle Unified Directory (OUD) and Oracle Internet Directory (OID).

# Identifying and Obtaining Software Distributions for an Enterprise Deployment

Before you begin to install and configure an enterprise topology, you must download the container images from Oracle Support.

For more information about downloading the container images, see Container Images for Oracle Identity and Access Management, and Oracle IDM Microservices (Doc ID 2723908.1). Oracle is migrating container delivery to the Oracle Container Registry (https://container-registry.oracle.com). Later images will be delivered by using this mechanism.

After you have downloaded the container images, you can stage them either locally on each of your Kubernetes worker nodes or host them in a container registry. If you want to use a container registry, see Using an OCI Container Registry.

Oracle strongly recommends that you use a container registry to store your images. However, you can manually load the images on to each worker node. See Staging Container Images.

> **Note:**
>
> If you are using Oracle Advanced Authentication, you must use a container registry.

If you use a container registry, you have the option of either pulling the images on demand to the worker node which requires them (see individual product chapters) or pulling the image ahead of time to every Kubernetes worker node in your deployment. If you choose to pull the images ahead of time, see Downloading Images from a Container Registry.

For obtaining the pre-built container images, see the Support document Oracle Identity and Access Management 12.2.1.4 Products with Kubernetes.

Oracle HTTP Server will be installed outside of the Kubernetes cluster. Therefore, a traditional software distribution is required.

**Oracle Software Distributions Used in this Guide**

Table 8-1 lists the distributions used in this guide.

For general information about how to obtain Oracle HTTP Server software, see Obtaining Product Distributions in *Planning an Installation of Oracle Fusion Middleware*.

For more specific information about locating and downloading specific Oracle Fusion Middleware products, see the *Oracle Fusion Middleware Download, Installation, and Configuration Readme Files* on OTN.

> **Note:**
>
> The information in this guide is meant to complement the information contained in the Oracle Fusion Middleware certification matrixes. If there is a conflict of information between this guide and the certification matrixes, then the information in the certification matrixes must be considered the correct version, as they are frequently updated.

**Table 8-1 Oracle Fusion Middleware Distributions to Download for Installing and Configuring the Enterprise Deployment Topology**

| Distribution | Image Name | Tag/Release | Description |
|---|---|---|---|
| Oracle HTTP Server 12c | `fmw_12.2.1.4.0_ohs_linux64.bin` | NA | Download this distribution to install the Oracle HTTP Server software on the web tier. |
| Oracle WebLogic Operator | `weblogic-kubernetes-operator` | 4.1.8 | Used to configure the WebLogic Operator, which is used to configure the domain, and if necessary, run RCU. |
| Oracle Unified Directory 12c | `oracle/oud` | 12.2.1.4-jdk8-ol8-<date> | Download this version or later and install on each Kubernetes worker node or load into a container registry. |
| Oracle Unified Directory Services Manager | `oracle/oudsm` | 12.2.1.4-jdk8-ol8-<date> | Download this version or later and install on each Kubernetes worker node or load into a container registry. |
| Oracle Access Manager 12c | `oracle/oam` | 12.2.1.4-jdk8-ol8-<date> | Download this version or later and install on each Kubernetes worker node or load into a container registry. |
| Oracle Identity Governance 12c | `oracle/oig` | 12.2.1.4-jdk8-ol8-<date> | Download this version or later and install on each Kubernetes worker node or load into a container registry. |
| Oracle Identity Role Intelligence | `oiri`<br>`oiri-ui`<br>`oiri-ding`<br>`oiri-cli` | 12.2.1.4.<date> | Download this version or later and install on each Kubernetes worker node or load into a container registry. |

**Table 8-1 (Cont.) Oracle Fusion Middleware Distributions to Download for Installing and Configuring the Enterprise Deployment Topology**

| Distribution | Image Name | Tag/Release | Description |
|---|---|---|---|
| Oracle Advanced Authentication | oaa-mgmt<br>oaa-svc<br>oaa-policy<br>oaa-admin<br>oaa-spui<br>oaa-factor-fido<br>oaa-factor-totp<br>oaa-factor-sms<br>oaa-factor-email<br>oaa-factor-yotp<br>oaa-factor-push<br>oaa-factor-kba<br>oaa-drss<br>risk-engine<br>risk-cc | 12.2.1.4.1_<date> | Download this version or later and install on each Kubernetes worker node or load it into a container registry. |

**Note:**

- The tags in this table show the base release. You should use the latest image, which contains the most recent Bundle Patch. A tag reflects the version you are using. For example: `12.2.1.4.0-8-ol7-210721.0755`.
- Oracle Advanced Authentication is also dependent on the following images being available in your container registry:
  - `docker.io/library/alpine:latest`
  - `container-registry.oracle.com/database/instantclient:12.2.0.1`
  - `container-registry.oracle.com/os/oraclelinux:8-slim`
- The commands to download and tag these images are as follows:

```
podman pull container-registry.oracle.com/database/
instantclient:12.2.0.1


podman tag container-registry.oracle.com/database/
instantclient:12.2.0.1 <container_image_registry>/shared/oracle/
database-instantclient:12.2.0.1


podman pull container-registry.oracle.com/os/oraclelinux:8-slim


podman tag container-registry.oracle.com/os/oraclelinux:8-slim
<container_image_registry>/shared/oracle/linux:8-slim
```

If you are using Docker, then replace `podman` with `docker` in the above commands.

# About Container Image Names

Container images have names in a specific format depending on whether you have pre-loaded them or pulled them from a container registry.

In the subsequent chapters, some procedures/instructions require you to provide the names of the container images. The image name depends on whether you are using a container registry or staging the images locally. The following description explains how you can determine the container image name to use for your deployments.

Typically, a container image name has the following format:

```
<REPOSITORY_NAME>/<IMAGE_NAME>:<IMAGE_VERSION>
```

If you have pre-loaded your images using the steps described in Staging Container Images, the image name will have this format: `oracle/oam:12.2.1.4.0-8-ol7-210721.0755`.

In Oracle Cloud Native Environment, the format is `localhost/oracle/oam:12.2.1.4.0-8-ol7-210721.0755`

If you are pulling your image from a container registry, the container name will be preceeded with the repository name. Assuming that your resgistry is `iad.ocir.io/mytenancy`, your image name will be `iad.ocir.io/mytenancy/oracle/oam:12.2.1.4.0-8-ol7-210721.0755`.

To determine the exact image name, you can use the following command, where your image has been pre-loaded:

```
docker images
```

OR

```
sudo podman images
```

# Obtaining Software from the Oracle Container Registry

Before you download container images from the Oracle Container Registry, you must first log in to the Oracle Container Registry using your support credentials and accept the license agreements for each container you want to deploy.

After you accept the licence agreement, you can pull the images directly from the Oracle Container Registry. Ensure that you pull the images with the latest bundled patches applied.

You can use these images in the following ways:

- Pull container images from the Oracle Container Registry on demand. If you are using this method follow the instructions in each of the sections for using a container registry.
- Manually pull the container images from the Oracle Container Registry and manually stage them on each worker node. See Downloading Images from a Container Registry.
- Manually pull the container images from the Oracle Container Registry, and then upload them to your own container registry. - See Downloading Images from a Container Registry and Using an OCI Container Registry.

# Downloading Images from a Container Registry

If you use a container registry, you can pull images from the registry either on-demand or in advance. Pulling images in advance results in a faster deployment but means that you will have the images available on all worker nodes regardless of whether you are running a container on that worker node or not.

> **Note:**
>
> The latest images in Oracle container registry is only available after you have logged into the registry and is available with a suffix of `_cpu`. For example, the latest oam images are available in `oam_cpu`.

To download the images from a container registry, you should execute the commands on each worker node.

- Pulling the Images to Docker
- Pulling the Images to CRI-O
- Oracle Advanced Authentication

## Pulling the Images to Docker

To download the images to Docker:

1. Log in to the container registry using the command:

   ```
   docker login registryname
   ```

   For example:

   ```
   docker login container-registry.oracle.com
   ```

2. Pull the image using a command similar to:

   ```
   docker pull container-registry.oracle.com/middleware/oam_cpu:12.2.1.4-jdk8-ol8-<date-stamp>
   ```

   For example:

   ```
   docker pull container-registry.oracle.com/middleware/oam_cpu:12.2.1.4-jdk8-ol8-240415
   ```

## Pulling the Images to CRI-O

You have two options to download the images to CRI-O:

**Option 1: Using `podman`**

If you have podman installed on your worker nodes, you can use podman to load the images.

> **Note:**
>
> You should run podman as the root user.

1. Log in to the container registry using the command:

   ```
   sudo podman login registryname
   ```

   For example:

   ```
   sudo podman login container-registry.oracle.com
   ```

2. Pull the image using a command similar to:

   ```
   sudo podman pull container-registry.oracle.com/middleware/oam_cpu:12.2.1.4-jdk8-ol8-<date-stamp>
   ```

For example:

```
sudo podman pull container-registry.oracle.com/middleware/oam_cpu:12.2.1.4-
jdk8-ol8-240415
```

**Option 2: Using `crictl`**
If you do not have podman installed on your worker nodes, for example in OKE environments, you can load the images using the `crictl` command:

```
crictl pull --creds username:password repository/image:version
```

Where `username` and `password` are the name of the user and the associated password that you use to access the registry.

For example:

```
crictl pull --creds myuser:password container-registry.oracle.com/middleware/
oam_cpu:12.2.1.4-jdk8-ol8-240415
```

## Oracle Advanced Authentication

If you have downloaded the OAA images from Oracle Support rather than via the container registry you will have a downloaded zip file.

To load the images containers in a zip file, first unzip the file using the following command:

```
unzip oaa-install-<REL>.zip
```

You will get an image archive file called `/oaa-install/oaa-<REL>.tar`.

This file can be staged using the commands in Staging Container Images.

# Staging Container Images

If you upload images to a container registry or use manually staged images, you have to stage those images in the local container repository. If you are using your own container registry you must first stage the images in a local repository so that you can upload them to your registry. You can perform this task on any host that has access to the `docker` or `podman` commands. The host does not need to be a part of the Kubernetes cluster.
If you are using locally staged images, each worker node should have access to all the container images in your deployment. Kubernetes decides on which worker node it wants to start a container. Therefore, you should have the image available on all hosts. If you are using locally staged images, you can use these instructions to load the images into each worker node manually. However, if you are using a container registry but want to load the image to the worker nodes ahead of time, you should manually pull the image from the container registry to each worker node.

- Staging Images in Docker
- Staging Images in CRI-O

# Staging Images in Docker

If you are using Docker as your container repository, you must stage the Oracle Identity and Access Manager container images in the Docker repository on each of the Kubernetes worker nodes.

If you want to store your container images in a container registry, see Using an OCI Container Registry.

To stage the container images:

1. Download the prebuilt container image from Oracle Support.
2. Load the images into the local container repository using the downloaded `tar` file by running the following command:

> **Note:**
>
> If you are manually staging the images on each worker host, you should repeat this step on every worker host.

```
docker load < container_image_file
```

For example:

```
docker load < oracle_oig_122140.PSU2020July.tar
```

> **Note:**
>
> If you are loading a multi-image archive file such as the file used by Oracle Advanced Authentication, this command will load multiple images into the local repository.

3. After the image is loaded, ensure that the container images are available by using the following command:

```
docker images
```

The output appears as the sample shown below:

```
REPOSITORY                          TAG                                 IMAGE
ID         CREATED          SIZE
oracle/oud                          12.2.1.4.0
1786d05b19ce     3 months ago     998MB
oracle/oud                          12.2.1.4.0-201209.2159.080
1786d05b19ce     3 months ago     998MB
oracle/oig                          12.2.1.4.0
285184517ac6     3 months ago     5.02GB
oracle/oig                          12.2.1.4.0-201116.0936.300
```

```
285184517ac6     3 months ago     5.02GB
oracle/oudsm                        12.2.1.4.0
8bbcb8e034e5     4 months ago     2.77GB
oracle/oudsm                        12.2.1.4.0-201116.1247.350
8bbcb8e034e5     4 months ago     2.77GB
oracle/oam                          12.2.1.4.0
9a75c5cd0faa     4 months ago     3.39GB
oracle/oam                          12.2.1.4.0-201116.0940.060
9a75c5cd0faa     4 months ago     3.39GB
weblogic-kubernetes-operator        3.3.0
7b26071abce6     7 months ago     388MB
oracle/weblogic-kubernetes-operator 3.3.0
7b26071abce6     7 months ago     388MB
k8s.gcr.io/kube-proxy               v1.18.4
718fa77019f2     9 months ago     117MB
k8s.gcr.io/kube-scheduler           v1.18.4
c663567f869e     9 months ago     95.3MB
k8s.gcr.io/kube-apiserver           v1.18.4
408913fc18eb     9 months ago     173MB
k8s.gcr.io/kube-controller-manager  v1.18.4
e8f1690127c4     9 months ago     162MB
alpine                              latest
a24bb4013296     10 months ago    5.57MB
calico/node                         v3.14.1
04a9b816c753     10 months ago    263MB
calico/pod2daemon-flexvol           v3.14.1
7f93af2e7e11     10 months ago    112MB
calico/cni                          v3.14.1
35a7136bc71a     10 months ago    225MB
calico/kube-controllers             v3.14.1
ac08a3af350b     10 months ago    52.8MB
k8s.gcr.io/pause                    3.2
80d28bedfe5d     13 months ago    683kB
k8s.gcr.io/coredns                  1.6.7
67da37a9a360     14 months ago    43.8MB
k8s.gcr.io/etcd                     3.4.3-0
303ce5db0e90     17 months ago    288MB
```

## Staging Images in CRI-O

If you are using CRI-O as your container repository, the standard repository for Oracle Cloud Native Environment deployments, you must stage the Oracle Identity and Access Manager container images in the CRI-O repository on each Kubernetes worker node.

If you want to store your container images in a container registry, see Using an OCI Container Registry.

> **Note:**
>
> - The commands in this section rely on the `podman` command. By default, this command is available on the Oracle Cloud Native Environment. However, you should manually install the command into the OKE environments which use CRI-O (Release 1.20+). You should run the `podman` commands as the root user.
> - If you are using a multi-image archive files such as the file used by Oracle Advanced Authentication, you should use the latest release of podman. The release provided with OCNE 1.3 does not support multiple-image archives.

To stage the container images:

1. Download the prebuilt container image from Oracle Support.
2. Load the images into the local container repository using the downloaded `tar` file by running the following command:

   > **Note:**
   >
   > If you are manually staging the images on each worker host, you should repeat this step on every worker host.

   ```
   podman load < container_image_file
   ```

   For example:

   ```
   podman load < oracle_oig_122140.PSU2020July.tar
   ```

   > **Note:**
   >
   > If you are loading a multi-image archive file such as the file used by Oracle Advanced Authentication, this command will load multiple images into the local repository.

3. After the image is loaded, ensure that the container images are available by using the following command:

   ```
   podman images
   ```

> **Note:**
>
> Oracle Cloud Native Environment has a different directory container images for each user. The Kubernetes engine for Oracle Cloud Native Environment uses the container store specified in the root user's configuration. Loading images using any user other than the root user will not make those images visible to Kubernetes. To ensure that you satisfy this requirement, `podman` commands must be run by the root user.
>
> As an extra validation, use the following command as a root user to ensure that the system can see your images:
>
> ```
> crictl images
> ```
>
> This command should be run on a worker node.

The output appears as the sample shown below:

```
REPOSITORY                           TAG                                 IMAGE
ID         CREATED          SIZE
oracle/oud                           12.2.1.4.0
1786d05b19ce     3 months ago     998MB
oracle/oud                           12.2.1.4.0-201209.2159.080
1786d05b19ce     3 months ago     998MB
oracle/oig                           12.2.1.4.0
285184517ac6     3 months ago     5.02GB
oracle/oig                           12.2.1.4.0-201116.0936.300
285184517ac6     3 months ago     5.02GB
oracle/oudsm                         12.2.1.4.0
8bbcb8e034e5     4 months ago     2.77GB
oracle/oudsm                         12.2.1.4.0-201116.1247.350
8bbcb8e034e5     4 months ago     2.77GB
oracle/oam                           12.2.1.4.0
9a75c5cd0faa     4 months ago     3.39GB
oracle/oam                           12.2.1.4.0-201116.0940.060
9a75c5cd0faa     4 months ago     3.39GB
weblogic-kubernetes-operator         3.3.0
7b26071abce6     7 months ago     388MB
oracle/weblogic-kubernetes-operator 3.3.0
7b26071abce6     7 months ago     388MB
k8s.gcr.io/kube-proxy                v1.18.4
718fa77019f2     9 months ago     117MB
k8s.gcr.io/kube-scheduler            v1.18.4
c663567f869e     9 months ago     95.3MB
k8s.gcr.io/kube-apiserver            v1.18.4
408913fc18eb     9 months ago     173MB
k8s.gcr.io/kube-controller-manager  v1.18.4
e8f1690127c4     9 months ago     162MB
alpine                               latest
a24bb4013296     10 months ago    5.57MB
calico/node                          v3.14.1
04a9b816c753     10 months ago    263MB
```

```
calico/pod2daemon-flexvol          v3.14.1
7f93af2e7e11     10 months ago    112MB
calico/cni                         v3.14.1
35a7136bc71a     10 months ago    225MB
calico/kube-controllers            v3.14.1
ac08a3af350b     10 months ago    52.8MB
k8s.gcr.io/pause                   3.2
80d28bedfe5d     13 months ago    683kB
k8s.gcr.io/coredns                 1.6.7
67da37a9a360     14 months ago    43.8MB
k8s.gcr.io/etcd                    3.4.3-0
303ce5db0e90     17 months ago    288MB
```

# Logging in to GitHub

The examples below require that you pull images and scripts from `github.com`, a public repository. To successfully pull images from GitHub, should log in to the repository.

You can log in to GitHub in one of two ways:

- By performing a manual login on each host, which requires access to the repository.
- By creating a Kubernetes secret, which grants access to the Kubernetes cluster.
- Creating a Secret to Access GitHub
- Logging in to GitHub Manually

## Creating a Secret to Access GitHub

If you need to deploy containers from GitHub, you should create a token that authenticates you with the container registry.

- Creating a GitHub Token
- Creating a Kubernetes Secret

### Creating a GitHub Token

This section assumes that you have a GitHub account. If you do not have a GitHub account, you must create one before continuing

To create a GitHub token:

1. Access the URL `https://github.com/settings/tokens`. You will be prompted to log in to your account.
2. Ensure that **Personal Access Tokens** is selected and click **Generate New Token**.
3. Specify a name for the token. For example: `Image downloads`.
4. Set an expiry date for the token.
5. In the **Select scopes** section:
   - Select **repo** and under it, select **public_repo**.
   - Select **read:packages**.
6. Click **Generate Token**.

Make a note of the generated token. You will not be able to find it again.

### Creating a Kubernetes Secret

After creating a GitHub token, you can now create a Kubernetes secret that will allow you to pull images from GitHub. The following command will create a Kubernetes secret called 'github' in the default namespace:

```
kubectl create secret docker-registry github --docker-server=ghcr.io --docker-
username=mygituser --docker-password="mytoken"
```

## Logging in to GitHub Manually

To log in to GithHub manually on each worker host, use the following commands.

**For Docker**

```
docker login ghcr.io
```

When prompted, specify your user name and the token you created earlier. See Creating a GitHub Token.

**For Cri-O**

```
sudo  podman login ghcr.io
```

When prompted, specify your user name and the token you created earlier.

# Staging the WebLogic Operator for Kubernetes

Use the WebLogic Operator for Kubernetes to deploy Oracle Access Manager or Oracle Identity Governance. There are two parts to the WebLogic Operator: the WebLogic Operator Container Image and the WebLogic Operator deployment scripts.

The WebLogic Operator for Kubernetes is available from `oracle.github.io`. Sign in to GitHub to download the container image.

- Staging the Oracle WebLogic Kubernetes Image in Docker
- Staging the Oracle WebLogic Kubernetes Image in CRI-O

## Staging the Oracle WebLogic Kubernetes Image in Docker

To download the Docker image, use the following commands:

```
docker pull ghcr.io/oracle/weblogic-kubernetes-operator:<VERSION>
```

```
docker tag ghcr.io/oracle/weblogic-kubernetes-operator:<VERSION> weblogic-
kubernetes-operator:<VERSION>
```

For example: To stage the image on Docker, use the following commands:

```
docker pull ghcr.io/oracle/weblogic-kubernetes-operator:3.3.0
```

```
docker tag ghcr.io/oracle/weblogic-kubernetes-operator:<VERSION> weblogic-kubernetes-operator:3.3.0
```

You should stage the Operator on all the worker nodes or place it into a registry that is accessible by the cluster.

## Staging the Oracle WebLogic Kubernetes Image in CRI-O

To download the container image, use the following command:

```
sudo podman pull ghcr.io/oracle/weblogic-kubernetes-operator:<VERSION>
```

```
sudo podman tag ghcr.io/oracle/weblogic-kubernetes-operator:<VERSION> weblogic-kubernetes-operator:<VERSION>
```

For example: To stage the image on CRI-O, use the following command:

```
sudo podman pull ghcr.io/oracle/weblogic-kubernetes-operator:3.3.0
```

```
sudo podman tag ghcr.io/oracle/weblogic-kubernetes-operator:3.3.0 weblogic-kubernetes-operator:3.3.0
```

> **Note:**
>
> You should use the `podman` commands as the root user.

You should stage the Operator on all the worker nodes or place it into a registry that is accessible by the cluster.

# Staging the Code Repository

Oracle provides a sample code repository to deploy Oracle Identity and Access Management in Kubernetes. The procedures explained in this guide use the sample code repository extensively. Download the sample code repository to a temporary work directory on your configuration host.

A configuration host is any host in your cluster which can run `helm` and `kubectl` commands. For example, in an OCI installation, this host could be a bastion host. For an Oracle Cloud Native Environment installation, it could be the operator node or any of the control plane hosts. Alternatively, you may have defined a specific host for the purpose.

The sample code is available from GitHub. You may need to create an account and log in if you do not have access.

To download the scripts, log in to GitHub and perform the following steps:

1. Create a temporary work directory on the configuration host:

   ```
   mkdir <work_dir>
   ```

   For example:

   ```
   mkdir /workdir
   ```

2. Change directory to this location:

   ```
   cd /workdir
   ```

3. Download the samples using the command:

   ```
   git clone https://github.com/oracle/fmw-kubernetes.git
   ```

These commands create a directory called `fmw-kubernetes`, where all of the sample files are stored. You can use the files directly from this location or copy the files you need to another temporary working directory. To keep everything separate, the procedures explained in this guide recommend copying the files to product working directories. For example: `/workdir/OAM`. However, it is optional.

> **Note:**
>
> This code repository also includes sample automation scripts for deploying Oracle Identity and Access Management as per the instructions provided in this guide.

# Downloading the Oracle Connector Bundle for Oracle Identity Governance

If you are planning to integrate Oracle Identity Governance with outside systems, you have to download the appropriate connector bundle. In an enterprise deployment, the LDAP connector is used to integrate with Oracle Unified Directory (OUD) and Oracle Internet Directory (OID).

Download the latest version of the Oracle Connector bundle from Oracle Identity Manager Connector Downloads.

9

# Preparing for an On-Premises Enterprise Deployment

An on-premises enterprise deployment involves preparing the hardware load balancer and ports, file system, operating system, and Kubernetes cluster.

This chapter includes the following topics:

- Preparing the Load Balancer and Firewalls for an Enterprise Deployment
  Preparing for an on-premises enterprise deployment also includes configuring a hardware or software load balancer and ports that you have to open on the firewalls used in the topology.
- Preparing a Kubernetes Cluster for the Enterprise Deployment
  If you want to use an on-premises based Kubernetes cluster, you have to create the cluster. There are many flavors of Kubernetes available but Oracle recommends the use of Oracle Cloud Native Environment.
- Preparing Storage for an Enterprise Deployment
  Before starting an enterprise deployment, it is important to understand the storage requirements. You should obtain and configure the storage.
- Preparing the Kubernetes Host Computers for an Enterprise Deployment
  Preparing the host computers largely depends on the flavor of Kubernetes you are deploying. See the appropriate Kubernetes installation instructions.
- Preparing the File System for an Enterprise Deployment
  Preparing the file system for an enterprise deployment involves understanding the requirements for local and shared storage, as well as the terminology that is used to reference important directories and file locations during the installation and configuration of the enterprise topology.
- Preparing a Disaster Recovery Environment
  A disaster recovery environment is a replica of your primary environment located in a different region from the primary region. This environment is a standby environment that you switch over to in the event of the failure of your primary environment.

## Preparing the Load Balancer and Firewalls for an Enterprise Deployment

Preparing for an on-premises enterprise deployment also includes configuring a hardware or software load balancer and ports that you have to open on the firewalls used in the topology.

- Configuring Virtual Hosts on the Hardware Load Balancer
- Configuring Firewalls and Ports for an Enterprise Deployment

### Configuring Virtual Hosts on the Hardware Load Balancer

The hardware load balancer configuration facilitates to recognize and route requests to several virtual servers and associated ports for different types of network traffic and monitoring.

The following topics explain how to configure the hardware load balancer, provide a summary of the virtual servers that are required, and provide additional instructions for these virtual servers:



## Overview of the Hardware Load Balancer Configuration

As shown in the topology diagrams, you must configure the hardware load balancer to recognize and route requests to several virtual servers and associated ports for different types of network traffic and monitoring.

In the context of a load-balancing device, a virtual server is a construct that allows multiple physical servers to appear as one for load-balancing purposes. It is typically represented by an IP address and a service, and it is used to distribute incoming client requests to the servers in the server pool.

The virtual servers should be configured to direct traffic to the appropriate host computers and ports for the various services that are available in the enterprise deployment.

In addition, you should configure the load balancer to monitor the host computers and ports for availability so that the traffic to a particular server is stopped as soon as possible when a service is down. This ensures that incoming traffic on a given virtual host is not directed to an unavailable service in the other tiers.

Note that after you configure the load balancer, you can later configure the web server instances in the web tier to recognize a set of virtual hosts that use the same names as the virtual servers that you defined for the load balancer. For each request coming from the hardware load balancer, the web server can then route the request appropriately, based on the server name included in the header of the request. See Configuring Oracle HTTP Server for Administration and Oracle Web Services Manager.

If you want to configure a load balancer to direct traffic to your worker nodes, you should configure the load balancer as a network load balancer, which directs all traffic sent to it to the target nodes regardless of the source port.

## Typical Procedure for Configuring the Hardware Load Balancer

The following procedure outlines the typical steps for configuring a hardware load balancer for an enterprise deployment.

Note that the actual procedures for configuring a specific load balancer will differ, depending on the specific type of load balancer. There may also be some differences depending on the type of protocol that is being load balanced. For example, TCP virtual servers and HTTP virtual servers use different types of monitors for their pools. Refer to the vendor-supplied documentation for actual steps.

1. Create a pool of servers. This pool contains a list of servers and the ports that are included in the load-balancing definition.

   For example, for load balancing between the web hosts, create a pool of servers that would direct requests to hosts WEBHOST1 and WEBHOST2 on port 7777.

2. Create rules to determine whether a given host and service is available and assign it to the pool of servers that are described in Step 1.

3. Create the required virtual servers on the load balancer for the addresses and ports that receive requests for the applications.

   For a complete list of the virtual servers required for the enterprise deployment, see Summary of the Load Balancer Virtual Servers Required for an Enterprise Deployment.

   When you define each virtual server on the load balancer, consider the following:

   a. If your load balancer supports it, specify whether the virtual server is available internally, externally, or both. Ensure that internal addresses are only resolvable from inside the network.

   b. Configure SSL Termination, if applicable, for the virtual server.

   c. Assign the pool of servers created in Step 1 to the virtual server.

## Load Balancer Health Monitoring

The load balancer must be configured to check that the services in the Load Balancer Pool are available. Failure to do so will result in requests being sent to hosts where the service is not running.

The following table shows examples of how to determine whether a service is available:

**Table 9-1 Examples Showing How to Determine Whether a Service is Available**

| Service | Monitor Type | Monitor Mechanism |
|---|---|---|
| OUD | ldap | ldapbind to cn=oudadmin |
| OHS | http | check for GET /\r\n |

## Summary of the Load Balancer Virtual Servers Required for an Enterprise Deployment

This topic provides details of the virtual servers that are required to be configured for an enterprise deployment.

The following table provides a list of the virtual servers that you must define on the hardware load balancer for the Oracle Identity and Access Management enterprise topology:

**Table 9-2 Virtual Servers to be Defined on the Hardware Load Balancer for the Oracle Identity and Access Management Enterprise Topology**

| Virtual Host | Server Pool | Protocol | SSL Termination? | Other Required Configuration/ Comments |
|---|---|---|---|---|
| `login.example.com:443` | `WEBHOST1.example.com:7777`<br>`WEBHOST2.example.com:7777` | HTTPS | Yes | Identity Management requires that the following be added to the HTTP header:<br>`Header Name: IS_SSL`<br>`Header Value: ssl`<br>`Header Name: WL-Proxy-SSL`<br>`Header Value: true` |

**Table 9-2 (Cont.) Virtual Servers to be Defined on the Hardware Load Balancer for the Oracle Identity and Access Management Enterprise Topology**

| Virtual Host | Server Pool | Protocol | SSL Termination? | Other Required Configuration/ Comments |
|---|---|---|---|---|
| prov.example.com:443 | WEBHOST1.example.com:7777<br>WEBHOST2.example.com:7777 | HTTPS | Yes | Identity Management requires that the following be added to the HTTP header:<br>Header Name: IS_SSL<br>Header Value: ssl<br>Header Name: WL-Proxy-SSL<br>Header Value: true |
| iadadmin.example.com:80 | WEBHOST1.example.com:7777<br>WEBHOST2.example.com:7777 | HTTP | | |
| igdadmin.example.com:80 | WEBHOST1.example.com:7777<br>WEBHOST2.example.com:7777 | HTTP | | |
| igdinternal.example.com:7777 | WEBHOST1.example.com:7777<br>WEBHOST2.example.com:7777 | HTTP | | |
| oiri.example.com | Kubernetes Worker Hosts:30777 | HTTP | No | Required only when deploying OIRI in the standalone mode with ingress enabled. |
| oaaadmin.example.com | Kubernetes Worker Hosts:30636 | HTTPS | No | Required only when deploying OAA in the standalone mode with ingress enabled. |
| oaa.example.com | Kubernetes Worker Hosts:307636 | HTTPS | No | Required only when deploying OAA in the standalone mode with ingress enabled. |

**Note:**

- Port 80 is the load balancer port used for HTTP requests.
- Port 443 is the load balancer port used for HTTPS requests.
- Port 7777 is the load balancer port used for internal callback requests.

# Configuring Firewalls and Ports for an Enterprise Deployment

As an administrator, it is important that you become familiar with the port numbers that are used by various Oracle Fusion Middleware products and services. This ensures that the same port number is not used by two services on the same host, and that the proper ports are open on the firewalls in the enterprise topology.

The following tables lists the ports that you must open on the firewalls in the topology:

Firewall notation:

- FW0 refers to the outermost firewall.
- FW1 refers to the firewall between the web tier and the application tier.
- FW2 refers to the firewall between the application tier and the data tier.

**Table 9-3 Firewall Ports Common to All Fusion Middleware Enterprise Deployments**

| Type | Firewall | Port and Port Range | Protocol / Application | Inbound / Outbound | Other Considerations and Timeout Guidelines |
|---|---|---|---|---|---|
| Browser request | FW0 | 80 | HTTP / Load Balancer | Inbound | Timeout depends on the size and type of HTML content. |
| Browser request | FW0 | 443 | HTTPS / Load Balancer | Inbound | Timeout depends on the size and type of HTML content. |
| Browser request | FW1 | 80 | HTTP / Load Balancer | Outbound (for intranet clients) | Timeout depends on the size and type of HTML content. |
| Browser request | FW1 | 443 | HTTPS / Load Balancer | Outbound (for intranet clients) | Timeout depends on the size and type of HTML content. |
| Callbacks and Outbound invocations | FW1 | 80 | HTTP / Load Balancer | Outbound | Timeout depends on the size and type of HTML content. |
| Callbacks and Outbound invocations | FW1 | 443 | HTTPS / Load Balancer | Outbound | Timeout depends on the size and type of HTML content. |
| Load balancer to Oracle HTTP Server | n/a | 7777 | HTTP | n/a | n/a |
| Session replication within a WebLogic Server cluster | n/a | n/a | n/a | n/a | By default, this communication uses the same port as the server's listen address. |
| Database access | FW2 | 1521 | SQL*Net | Both | Timeout depends on database content and on the type of process model used for SOA. |
| Coherence for deployment | n/a | 9991 | n/a | n/a | n/a |

**Table 9-3 (Cont.) Firewall Ports Common to All Fusion Middleware Enterprise Deployments**

| Type | Firewall | Port and Port Range | Protocol / Application | Inbound / Outbound | Other Considerations and Timeout Guidelines |
|---|---|---|---|---|---|
| Oracle Notification Server (ONS) | FW2 | 6200 | ONS | Both | Required for Gridlink. An ONS server runs on each database server. |
| Elasticsearch | FW1 | 31920 | HTTP | Outbound | Used for sending the log files to Elasticsearch (Optional). |
| Elasticsearch | FW12 | 31920 | HTTP | Inbound | Used for sending the log files to Elasticsearch (Optional). |

**Table 9-4 Firewall Ports Specific to the Oracle Identity and Access Management Enterprise Deployment**

| Type | Firewall | Port and Port Range | Protocol / Application | Inbound / Outbound | Other Considerations and Timeout Guidelines |
|---|---|---|---|---|---|
| Webtier Access to Oracle Weblogic Administration Server (IAMAccessDomain) | FW1 | 30701 | HTTP / Oracle HTTP Server and Administration Server | Inbound | N/A |
| Webtier Access to Oracle Weblogic Administration Server (IAMGovernanceDomain) | FW1 | 30711 | HTTP / Oracle HTTP Server and Administration Server | Inbound | N/A |
| Enterprise Manager Agent - web tier to Enterprise Manager | FW1 | 5160 | HTTP / Enterprise Manager Agent and Enterprise Manager | Both | N/A |
| Oracle HTTP Server to Ingress | FW1 | 30777 | HTTP | Inbound | N/A |
| Oracle HTTP Server to oam_server | FW1 | 30410 | HTTP / Oracle HTTP Server to WebLogic Server | Inbound | Timeout depends on the `mod_weblogic` parameters used |
| Oracle HTTP Server oim_server | FW1 | 30140 | HTTP / Oracle HTTP Server to WebLogic Server | Inbound | Timeout depends on the `mod_weblogic` parameters used |

**Table 9-4 (Cont.) Firewall Ports Specific to the Oracle Identity and Access Management Enterprise Deployment**

| Type | Firewall | Port and Port Range | Protocol / Application | Inbound / Outbound | Other Considerations and Timeout Guidelines |
|---|---|---|---|---|---|
| Oracle HTTP Server soa_server | FW1 | 30801 | HTTP / Oracle HTTP Server to WebLogic Server | Both | Timeout depends on the `mod_weblogic` parameters used |
| Oracle HTTP Server oam_policy_mgr | FW1 | 30510 | HTTP / Oracle HTTP Server to WebLogic Server | Both | Timeout depends on the `mod_weblogic` parameters used |
| Oracle HTTP Server to OIRI UI | FW1 | 30305 | HTTP/ Oracle HTTP Server to OIRI UI microservice | Both | NA |
| Oracle HTTP Server to OIRI | FW1 | 30306 | HTTP/ Oracle HTTP Server to OIRI microservice | Both | NA |
| Oracle Coherence Port | FW1 | 8000–8088 | TCMP | Both | N/A |
| OUD Port | FW1 | 31389 | LDAP | Inbound | Ideally, these connections should be configured not to time out<br>**Note**: Required only if you need to access OUD outside of the Kubernetes cluster. |
| OUD SSL Port | FW1 | 31636 | LDAPS | Inbound | Ideally, these connections should be configured not to time out<br>**Note**: Required only if you need to access OUD outside of the Kubernetes cluster. |
| Kubernetes Cluster to Database Listener | FW2 | 1521 | SQL*Net | Both | Timeout depends on all database content and on the type of process model used for Oracle Identity and Access Management |

**Table 9-4 (Cont.) Firewall Ports Specific to the Oracle Identity and Access Management Enterprise Deployment**

| Type | Firewall | Port and Port Range | Protocol / Application | Inbound / Outbound | Other Considerations and Timeout Guidelines |
|---|---|---|---|---|---|
| Oracle Notification Server (ONS) | FW2 | 6200 | ONS | Both | Required for Gridlink. An ONS server runs on each database server |
| Oracle HTTP Server to OAA Admin | FW1 | 32721 | HTTP/ Oracle HTTP Server to OAA Admin UI microservice | Both | NA |

# Preparing a Kubernetes Cluster for the Enterprise Deployment

If you want to use an on-premises based Kubernetes cluster, you have to create the cluster. There are many flavors of Kubernetes available but Oracle recommends the use of Oracle Cloud Native Environment.

While the instructions in this guide should work with any Kubernetes deployment, they have been validated on an Oracle Cloud Native Environment cluster.

- Host Requirements for the Kubernetes Cluster
- Deployment Options
- Hardware Requirements
- Creating a Kubernetes Cluster
- Enabling the Firewall Rule for Oracle Cloud Native Environment

## Host Requirements for the Kubernetes Cluster

A Kubernetes cluster consists of two types of hosts:

- Control plane hosts: The control plane hosts are responsible for managing the Kubernetes cluster. Although you can use a single host for the control plane, Oracle strongly recommends that you create a highly available control plane consisting of a minimum of three control plane hosts, ideally five.
- Worker hosts: The worker hosts are used to run the Kubernetes containers. Each Kubernetes worker node requires to have sufficient capacity to run your deployment. You will need a minimum of two worker nodes to ensure high availability. The number of worker nodes will depend on the application components you plan to deploy and the capacities of the worker nodes. The greater the number of applications, greater the number of worker nodes and/or capacities.

## Deployment Options

You can install Kubernetes on any of the following server types:

- Bare-metal server

- Virtual Machine instance running on on-premises hardware or in the cloud
- Cloud based bare-metal instance
- Cloud based Infrastructure virtual instance
- Oracle Private Cloud Appliance virtual instance
- Oracle Private Cloud at Customer virtual instance

## Hardware Requirements

The following are the minimum hardware requirements for your deployment (based on an Oracle Cloud Native Environment deployment):

- Kubernetes Control Plane Node Hardware - A minimum Kubernetes control plane node configuration is:
  - 4 CPU cores (Intel VT-capable CPU)
  - 16 GB RAM
  - 1 GB Ethernet NIC
  - XFS file system (the default file system for Oracle Linux)
  - 40 GB hard disk space in the `/var` directory
- Kubernetes Worker Node Hardware - A minimum Kubernetes worker node configuration is:
  - 1 CPU cores (Intel VT-capable CPU)
  - 8 GB RAM
  - 1 GB Ethernet NIC
  - XFS file system (the default file system for Oracle Linux)
  - 15 GB hard disk space in the `/var` directory
- Operator Node Hardware - A minimum operator node configuration is:
  - 1 CPU cores (Intel VT-capable CPU)
  - 8 GB RAM
  - 1 GB Ethernet NIC
  - 15 GB hard disk space in the `/var` directory

> **Note:**
>
> These are the minimum requirements to run a Kubernetes cluster. Your application will most likely need far more resources than these minimum values.

## Creating a Kubernetes Cluster

The instructions for deploying the Kubernetes cluster are outside the scope of this guide. Each Kubernetes deployment will have its own deployment mechanisms. To install Oracle Cloud Native Environment, see Installing Oracle Cloud Native Environment.

## Enabling the Firewall Rule for Oracle Cloud Native Environment

If you are deploying a Kubernetes cluster using Oracle Cloud Native Environment, you should ensure that firewall masquerade is enabled on each Kubernetes worker node. To enable this firewall, use the following command on each worker node:

```
sudo firewall-cmd --add-masquerade --permanent
```

# Preparing Storage for an Enterprise Deployment

Before starting an enterprise deployment, it is important to understand the storage requirements. You should obtain and configure the storage.

For instructions to configure storage, see Storage Requirements for an Enterprise Deployment.

- Summary of the Shared Storage Volumes in an Enterprise Deployment
- Summary of Local Storage Used in an Enterprise Deployment
- Local Storage Requirements

## Summary of the Shared Storage Volumes in an Enterprise Deployment

It is important to understand the shared volumes and their purpose in a typical Oracle Fusion Middleware enterprise deployment.

To understand the storage requirements for an Oracle Identity and Access Management deployment, see Storage Requirements for an Enterprise Deployment.

For information about recommendations for storage, see Storage Requirements for an Enterprise Deployment.

## Summary of Local Storage Used in an Enterprise Deployment

To understand the local storage requirements for an enterprise deployment, see Summary of Local Storage Used in an Enterprise Deployment.

## Local Storage Requirements

Each worker node needs to have sufficient storage to hold not only the containers but also the container images.

# Preparing the Kubernetes Host Computers for an Enterprise Deployment

Preparing the host computers largely depends on the flavor of Kubernetes you are deploying. See the appropriate Kubernetes installation instructions.

In addition, you should perform the following actions:

- Verifying the Minimum Hardware Requirements for Each Host
- Verifying Linux Operating System Requirements

- Creating and Mounting the Directories for an Enterprise Deployment
- Enabling Unicode Support
- Setting the DNS Settings
- Configuring a Host to Use an NTP (time) Server

# Verifying the Minimum Hardware Requirements for Each Host

After you procure the required hardware for the enterprise deployment, it is important to ensure that each host computer meets the minimum system requirements of the Kubernetes cluster. See Hardware Requirements.

Ensure that you have sufficient local disk storage and shared storage configured as described in Storage Requirements for an Enterprise Deployment.

Allow sufficient swap and temporary space; specifically:

- **Swap Space** – Most Kubernetes deployments recommend disabling swap. See the Kubernetes documentation for your version specific recommendations.
- **Temporary Space** – There must be a minimum of 500 MB of free space in the `/tmp` directory.

# Verifying Linux Operating System Requirements

You can review the typical Linux operating system settings for an enterprise deployment.

To ensure the host computers meet the minimum operating system requirements of your Kubernetes cluster, ensure that you have installed a certified operating system and that you have applied all the necessary patches for the operating system.

In addition, review the following sections for typical Linux operating system settings for an enterprise deployment.

- Setting Linux Kernel Parameters
- Setting the Open File Limit and Number of Processes Settings on UNIX Systems

## Setting Linux Kernel Parameters

The kernel of the Kubernetes worker hosts must be large enough to support the deployment of all of the containers you want to run on it. The values shown in Table 9-5 are the absolute minimum values to deploy Oracle Identity and Access Management. Oracle recommends that you tune these values to optimize the performance of the system. See your operating system documentation for more information about tuning kernel parameters.

The values in the following table are the current Linux recommendations. For the latest recommendations for Linux and other operating systems, see *Oracle Fusion Middleware System Requirements and Specifications*.

**Table 9-5 UNIX Kernel Parameters**

| Parameter | Value |
|---|---|
| kernel.sem | 256 32000 100 142 |
| kernel.shmmax | 4294967295 |

To set these parameters:

1. Sign in as `root` and add or amend the entries in the `/etc/sysctl.conf` file.
2. Save the file.
3. Activate the changes by entering the following command:

   ```
   /sbin/sysctl -p
   ```

## Setting the Open File Limit and Number of Processes Settings on UNIX Systems

On UNIX operating systems, the `Open File Limit` is an important system setting, which can affect the overall performance of the software running on the host computer.

For guidance on setting the `Open File Limit` for an Oracle Fusion Middleware enterprise deployment, see Host Computer Hardware Requirements.

> **Note:**
>
> The following examples are for Linux operating systems. Consult your operating system documentation to determine the commands to be used on your system.

For more information, see the following sections.

- Viewing the Number of Currently Open Files
- Setting the Operating System Open File and Processes Limits

### Viewing the Number of Currently Open Files

You can see how many files are open with the following command:

```
/usr/sbin/lsof | wc -l
```

To check your open file limits, use the following commands.

**C shell**:

```
limit descriptors
```

**Bash**:

```
ulimit -n
```

### Setting the Operating System Open File and Processes Limits

To change the Open File Limit values on Oracle Enterprise Linux 6 or greater:

1. Sign in as `root` user and edit the following file:

   ```
   /etc/security/limits.d/*-nproc.conf
   ```

   For example:

   ```
   /etc/security/limits.d/20-nproc.conf
   ```

> **Note:**
>
> The number can vary from host to host.

2. Add the following lines to the file. (The values shown here are for example only):

```
* soft  nofile  4096
* hard  nofile  65536
* soft  nproc   2047
* hard  nproc   16384
```

   The `nofiles` values represent the open file limit; the `nproc` values represent the number of processes limit.

3. Save the changes, and close the file.

4. Re-login into the host computer.

# Creating and Mounting the Directories for an Enterprise Deployment

Kubernetes uses storage differently as compared to traditional Unix deployments. In a Kubernetes deployment, container data is stored within a persistent volume. While a persistent volume can be a local file system, Oracle recommends that you create the persistent volume (PV) on the shared storage that is available to each Kubernetes node. This method makes it possible for a Kubernetes container to be started on any of the Kubernetes worker nodes.

- This guide uses NFS persistent volumes. In an NFS persistent volume, you create a "Share" on the shared storage, and then each Kubernetes container mounts that share. This feature ensures that the robustness of enterprise storage is used for crititical data. For details, see Storage Requirements for an Enterprise Deployment.

  An alternative approach is to mount the NFS volumes to each worker node. The downside to this approach is that every Kubernetes worker node where a container starts must have the file system mounted. In a large cluster, this can significantly increase the management overhead. However, it does allow you to use the redundant NFS volumes to protect from possible corruption. In this scenario, you are responsible for creating your own processes to keep your redundant NFS mounts in sync. Using NFS mounted to the pods removes the dependency on the worker nodes. This feature allows the pod to run on any cluster node mounting the NFS as needed, which simplifies management. But you rely on the inbuilt redundancy of your NFS storage rather than creating the redundancy manually.

- This guide assumes that the Oracle Web tier software is installed on a local disk or a privately attached shared storage.

  The Web tier installation is typically performed on local storage to the WEBHOST nodes. When you use shared storage, you can install the Oracle Web tier binaries (and create the Oracle HTTP Server instances) on a shared disk. However, if you do so, then the shared disk *must* be separate from the shared disk used for the application tier, and you must consider the appropriate security restrictions for access to the storage device across tiers.

  As with the application tier servers (OAMHOST1 and OAMHOST2), use the same directory path on both computers.

  For example:

```
/u02/private/oracle/products/web
```

- During the deployment, Oracle recommends mounting the 'Shares', which are mounted to containers, to the deployment host. The mounting makes copying files to the persistent volumes easier even when containers have not started. It also makes it simpler to clean up failed installations and, if needed, debug. Many container images do not contain utilities such as Vim for viewing files.
- Mounting File Systems on Hosts

## Mounting File Systems on Hosts

For OHS hosts, place the entries in `/etc/fstab` with the following mount options:

Sample OHS `/etc/fstab` entry:

```
<IP>:/export/IAMBINARIES/webbinaries1 /u02/private/oracle/products nfs
auto,rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768
<IP>:/export/IAMCONFIG/webconfig1  /u02/private/oracle/config nfs
auto,rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768
```

Before you can use the file system with the containers, ensure that you can write to the file system. Mount the file system to the bastion node and write to it. If you are unable to write, use the `chmod` command to enable writing to the file system.

For example:

```
sudo mkdir -p /u02/private/oracle/products /u02/private/oracle/config
sudo mount -a
sudo chmod -R 777 /u02/private/oracle
```

**Table 9-6 Summary of Hosts and the File Systems to be Mounted**

| Mount Host | File Systems | Comments |
|---|---|---|
| webhost1 | webbinaries1 | Mounted as `/u02/private/oracle/products`. |
| webhost2 | webbinaries2 | Mounted as `/u02/private/oracle/products`. |
| webhost1 | webconfig1 | Mounted as `/u02/private/oracle/config`. |
| webhost2 | webconfig2 | Mounted as `/u02/private/oracle/config`. |
| All Kubernetes nodes | images<br>nfs_volumes* | Used as a staging directory to temporarily store container images.<br>Mounted as `/images`. |
| bastion node | oudconfigpv | Mounted as `/nfs_volumes/oudconfigpv`. |
| | oudpv | Mounted as `/nfs_volumes/oudpv`. |
| | oudsmpv | Mounted as `/nfs_volumes/oudsmpv`. |

**Table 9-6 (Cont.) Summary of Hosts and the File Systems to be Mounted**

| Mount Host | File Systems | Comments |
|---|---|---|
| | oigpv | Mounted as `/nfs_volumes/oigpv`. |
| | oampv | Mounted as `/nfs_volumes/oampv`. |
| | oiripv | Mounted as `/p_volumes/oiripv`. |
| | dingpv | Mounted as `/p_volume/idingpv`. |
| | docker_repo* | Mounted as `/docker_repo`. |

Optionally, mount all PVs. This option lets you delete deployments during the configuration phase, if necessary. Remove these mounts after the system is up and running.

> **Note:**
>
> * Alternatively, for these file systems, you can use block volumes.

## Enabling Unicode Support

Oracle recommends you to enable Unicode support in your operating system to process characters in Unicode.

Your operating system configuration can influence the behavior of characters supported by Oracle Fusion Middleware products.

On UNIX operating systems, Oracle highly recommends that you enable Unicode support by setting the `LANG` and `LC_ALL` environment variables to a locale with the UTF-8 character set. This enables the operating system to process any character in Unicode. Oracle Identity and Access Management technologies, for example, are based on Unicode.

If the operating system is configured to use a non-UTF-8 encoding, Oracle Identity and Access Management components may function in an unexpected way. For example, a non-ASCII file name might make the file inaccessible and cause an error. Oracle does not support problems caused by operating system constraints.

## Setting the DNS Settings

After you have created the Kubernetes cluster, you have to ensure that the pods are capable of resolving the application URLs and hosts used in your deployment. Kubernetes uses 'coreDNS' to resolve host names. Out-of-the-box, this DNS service is used for internal Kubernetes pod name resolution. However, it can be extended to include your custom entries too.

There are two ways of including custom entries:

- By adding the individual host entries to coreDNS
- By adding the corporate DNS server to coreDNS for the application domain

## Adding Individual Host Entries to CoreDNS

To add individual host entries to CoreDNS:

1. Create a config map with your custom hosts in called `coredns_custom.yaml`. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: kube-system
data:
  example.server: |
      example.com {
            hosts {
            1.1.1.1 login.example.com
            1.1.1.2 prov.example.com
            1.1.1.3 iadadmin.example.com
            1.1.1.4 igdadmin.example.com
            1.1.1.5 igdinternal.example.com
            fallthrough
           }
      }
```

2. Save the file.
3. Create the config map using the following command:

```
kubectl create -f coredns_custom.yaml
```

4. Restart coreDNS using the command:

```
kubectl rollout restart -n kube-system deploy coredns
```

   To ensure that the coreDNS pods restart without any issue, use the following command:

```
kubectl get pods -n kube-system
```

   Ensure that the custom config has been loaded by using the following command:

```
kubectl get configmaps --namespace=kube-system coredns-custom -o yaml
```

   If any errors occur, use the following command to view them:

```
kubectl logs -n kube-system coredns--<ID>
```

   Correct the errors by editing the configmap again.

## Adding the Corporate DNS Server to CoreDNS for the Application Domain

To ensure that CoreDNS forwards all entries to the corporate DNS server for the application domain:

1. Create a config map with your custom dns server in called `coredns_custom.yaml`. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: kube-system
data:
  example.server: |
    example.com {
      forward . <CORPORATE_DNS_IP_ADDRESS>
    }
```

2. Save the file.
3. Create the config map using the following command:

```
kubectl create -f coredns_custom.yaml
```

4. Restart coreDNS using the command:

```
kubectl delete pod --namespace kube-system --selector k8s-app=kube-dns
```

Ensure that the coredns pods restart without any issue, using the command:

```
kubectl get pods -n kube-system
```

Ensure that the custom config has been loaded by using the following command:

```
kubectl get configmaps --namespace=kube-system coredns-custom -o yaml
```

If any errors occur, use the following command to view them:

```
kubectl logs -n kube-system coredns--<ID>
```

Correct the errors by editing the configmap again.

## Validating the DNS Resolution

Most containers do not have in-built networking tools that enable you to check that the configuration changes you made are correct. The easiest way to validate the changes is to use a lightweight container with the network tools that you have installed. For example: Alpine.

Alpine is a slimmed down bash environment. You can start an Alpine container using a command as follows:

```
kubectl run -i --tty --rm debug --image=docker.io/library/alpine:latest --
restart=Never -- sh
```

This command provides access to `nslookup` where you can check for host resolution using a command as follows:

```
nslookup login.example.com
```

## Configuring a Host to Use an NTP (time) Server

All hosts in the deployment must have the same time. The best way to achieve this is to use an NTP server. To configure a host to use an NTP server:

1. Determine the name of the NTP server(s) you wish to use. For security reasons, ensure that these are inside your organization.
2. Log in to the host as the root user.
3. Edit the file `/etc/ntp.conf` to include a list of the time servers. After editing, the file appears as follows:

   ```
   server ntphost1.example.com
   server ntphost2.example.com
   ```

4. Run the following command to synchronize the system clock to the NTP server:

   ```
   /usr/sbin/ntpdate ntpserver1.example.com
   /usr/sbin/ntpdate ntpserver2.example.com
   ```

5. Start the NTP client using the following command:

   ```
   service ntpd start
   ```

6. Validate that the time is set correctly using the date command.
7. To make sure that the server always uses the NTP server to synchronize the time. Set the client to start on reboot by using the following command:

   ```
   chkconfig ntpd on
   ```

# Preparing the File System for an Enterprise Deployment

Preparing the file system for an enterprise deployment involves understanding the requirements for local and shared storage, as well as the terminology that is used to reference important directories and file locations during the installation and configuration of the enterprise topology.

- Overview of Preparing the File System

## Overview of Preparing the File System

It is important to set up your storage in a way that makes the enterprise deployment easy to understand, configure, and manage.

To fully understand the storage requirements for your enterprise deployment, see Storage Requirements for an Enterprise Deployment.

In addition to mounting these file systems inside the containers, you have to mount the following volumes to the administration/deployment host. Your administration host is where you will deploy the software.

**Table 9-7 Volume and Mount Point for the Administration Host**

| Shared Volume Name | Mount Point |
|---|---|
| oudconfigpv | `/nfs_volumes/oudconfigpv` |
| oudpv | `/nfs_volumes/oudpv` |
| oudsmpv | `/nfs_volumes/oudsmpv` |
| oigpv | `/nfs_volumes/oigpv` |
| oampv | `/nfs_volumes/oampv` |
| oiripv | `/idmpvs/oiripv` |
| dingpv | `/idmpvs/dingpv` |
| oaacredpv | `/nfs_volume/oaacredpv` |
| oaaconfigpv | `/nfs_volume/oaaconfigpv` |
| oaalogpv | `/nfs_volume/oaalogpv` |
| oaavaultpv | `/nfs_volume/oaavaultpv`<br>**Note**: Required when using a file-based vault |
| docker_repo* | `/docker_repo` |

# Preparing a Disaster Recovery Environment

A disaster recovery environment is a replica of your primary environment located in a different region from the primary region. This environment is a standby environment that you switch over to in the event of the failure of your primary environment.

The standby environment will be a separate cluster, ideally in a different data center. If the cluster is dedicated to the application, the second cluster should be a mirror of the primary cluster with the same number and specifications of worker nodes. If your cluster is a multi-purpose cluster that is used by different applications, ensure sufficient spare capacity in the standby site to run the full application workload of the primary cluster.

Each Kubernetes cluster will run the same operating system version and the Kubernetes major release.

Your network will be such that:

- The primary and standby database networks communicate with each other to facilitate the creation of a Data Guard database.
- The primary and standby file system networks communicate with each other to facilitate the replication of the file system data. If you have run to the `Rsync` process to achieve the

replication inside the cluster, then the primary Kubernetes worker network will be able to communicate with the Kubernetes worker network on the standby site.

- A global load balancer will be used to direct the traffic between the primary and standby sites. This load balancer is often independent of the site-specific load balancers used for on-site communication.
- The SSL certificates used in the load balancers must be the same in each load balancer. The traffic should not be aware when the load balancer switches sites.

10

# Preparing the Oracle Cloud Infrastructure for an Enterprise Deployment

If you plan to deploy Identity and Access Management on Oracle Cloud Infrastructure (OCI) using the Oracle Container Engine for Kubernetes, you have to configure OCI to facilitate the deployment. Create the required OCI components to perform the deployment.

> **Note:**
>
> The instructions provided in this guide are correct at the time of publishing. Due to the evolving nature of the OCI interface, you may find minor changes in the options. See the Oracle Cloud Infrastructure documentation to obtain the latest steps.

This chapter includes the following topics:

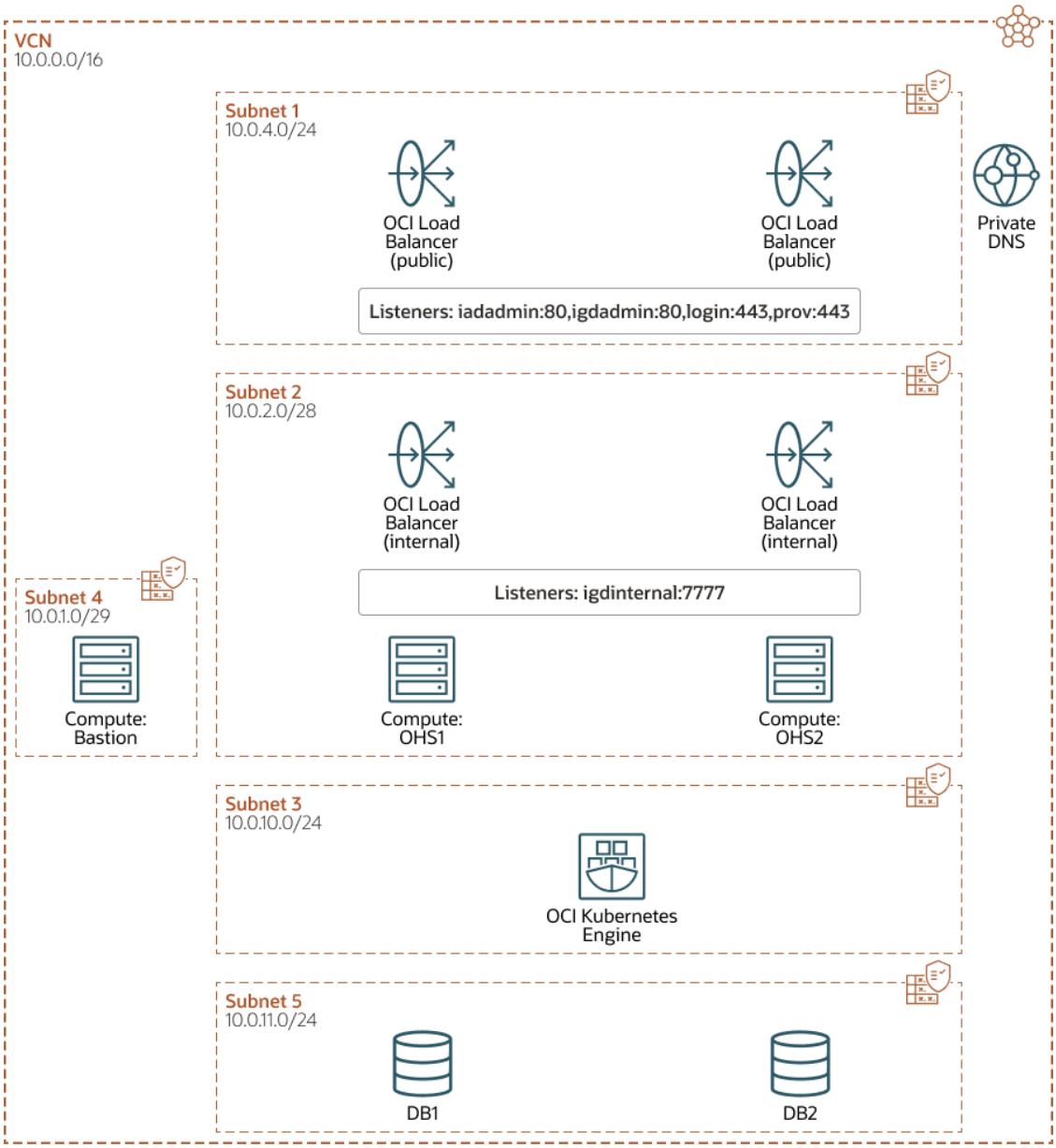
- About the OCI Deployment
  This illustration shows all the OCI components that you require to deploy Oracle Identity and Access Management on OCI. It shows the different network requirements and how the OCI components fit into those networks. Each subnet is protected by security lists.
- Creating an SSH Key Pair
  You can configure OCI by using the Oracle Cloud Console and a bastion node. The SSL certificates provide a secure access to the bastion node, compute instances, OKE worker nodes, and database hosts. You have to create an SSL certificate on the host you use to configure OCI. This host could be a laptop or a desktop.
- Creating an OCI Compartment
  Create a container in your OCI tenancy to hold the deployment.
- Creating an OKE Cluster in OCI
- Creating a Bastion Node
  You cannot access the cluster directly because the cluster is in a dedicated subnet. You can use a bastion node to access the cluster. The bastion node will be publicly available.
- Creating Compute Instances for Oracle HTTP Servers
  The web tier resides in its own subnet separated from both the load balancer and the application tier. This section describes the procedures to create two compute instances for the web tier, place them in different fault domains, and set up security lists and route tables to facilitate access.
- Creating File Systems and Mount Targets
  You need to create NFS file systems for Kubernetes Persistent Volumes and Oracle HTTP Server installations.
- Creating Load Balancers
  You need to create two OCI load balancers. One of these load balances is used to direct public traffic and the other for internal call backs. The load balancer used for internal traffic is not available outside the OCI container.

- Creating a Network Load Balancer
  This step is required only if you want to configure a load balancer to route traffic to the Kubernetes worker nodes.
- Creating a Database
  There are several different databases that you can create in OCI. For this example, a bare metal RAC database will be created. You may need to create one or more databases.
- Creating a Vault
  A vault is used to store the credentials of your deployment. At present, the only Oracle Identity and Access Management product using a vault is Oracle Advanced Authentication (OAA). OAA can use either an OCI-based vault (recommended) or a file-based vault.
- Creating a DNS Server
  This is an optional task. It is important that all host names are resolvable, including the load balancer virtual hosts. You can make them resolvable by adding entries to the local hosts files. However, in OCI, using a private DNS server is the simpler method.
- Updating Kubernetes CoreDNS
- Validating Your Environment
  Perform the checks described in this section to ensure that your environment is ready for a deployment.
- Preparing a Disaster Recovery Environment
  A disaster recovery environment is a replica of your primary environment located in a different region from the primary region. This environment is a standby environment that you switch over to in the event of the failure of your primary environment.

# About the OCI Deployment

This illustration shows all the OCI components that you require to deploy Oracle Identity and Access Management on OCI. It shows the different network requirements and how the OCI components fit into those networks. Each subnet is protected by security lists.

**Figure 10-1 An Illustration of the OCI Layout for OKE**




When deploying Oracle Identity and Access Management in OCI, you have to set up the OCI environment with the following characteristics:

- **VCN**: There will be one public Virtual Cloud Network which provides external access to the environment. For security reasons, the VCN is broken down into a several subnets.
- **Subnets**: The VCN is divided into several subnets to ensure that the network traffic is routed only to the areas requiring it. For instance, traffic to the database subnet will not be available directly from the internet. Traffic is available only to the Application (OKE) tier, which interacts with the database subnet.
- **Security Lists**: Security lists provide an additional layer of security that allows traffic only into and out of a subnet, based on the ports and protocols permitted.
- **Bastion Node**: The Bastion node is a compute instance inside the VCN that you can log in to. The Bastion node can communicate with all the components inside the deployment.

The Bastion node is used for setting up the environment and for ongoing management. Therefore, you must lock down access to the Bastion node to ensure that it is accessed only by clients on your corporate network who are registered with it using an SSL key pair.

- **Load balancer**: The two LBaaS services are created within the OCI framework. The public-facing load balancer is used to access the Oracle Identity and Access Management deployment from the internet. The private load balancer is for internal traffic, routing it is not available outside of the VCN. The public load balancer is the only internet-facing part of your deployment (except for the Bastion node).
- **Oracle Container Engine for Kubernetes (OKE)**: This is where your application is deployed inside the Kubernetes containers. It is not visible to the internet directly. The Oracle HTTP servers are not deployed in the OKE cluster. These servers are placed into a separate demilitarized zone (DMZ).
- **Compute Instances**: You require a minimum of two compute instances to host your Oracle HTTP servers. These are placed into a demilitarized zone (DMZ) below the load balancers. The load balancers send requests to the OHS servers, which pass the traffic onto the application residing within the OKE cluster.
- **Database**: The database(s) are present in a dedicated subnet below the OKE cluster.
- **DNS**: The DNS server is optional. It is used internally to provide name resolution. You can achieve name resolution by maintaining entries in the individual host files.

The following sections describe the procedure to set up the components depicted in this illustration:

# Creating an SSH Key Pair

You can configure OCI by using the Oracle Cloud Console and a bastion node. The SSL certificates provide a secure access to the bastion node, compute instances, OKE worker nodes, and database hosts. You have to create an SSL certificate on the host you use to configure OCI. This host could be a laptop or a desktop.

After you create the certificate on the device, share it with the OCI resources to enable access to the resources and to manage them. If you use more than one device, you have to register the SSL keys for all those devices.

If you do not have an SSL certificate for the device you are using, create the cetificate using the following command:

```
ssh-keygen -t rsa -N "" -b 2048 -f id_rsa
```

This command creates two files `id_rsa` and `id_rsa.pub` in the `.ssh` directory under the home directory. These are the certificate files you will use to access the OCI resources.

# Creating an OCI Compartment

Create a container in your OCI tenancy to hold the deployment.

To create a compartment:

1. Log in to the Oracle Cloud Infrastructure Console, select **Identity**, and then click **Compartments**.
2. Click **Create Compartment**.
3. Specify a **Name** and **Description**.

4. Click **Create Compartment**.

You will create all the OCI objects inside this compartment.

# Creating an OKE Cluster in OCI

You can create a cluster in OKE in one of two ways: by creating a quick cluster with minimum user input or by manually creating a cluster that provides more flexibility. If you create a cluster using the quick cluster, you will have minimum configuration options for cluster networking. Networking is important to specify the network subnets you want to use.

- Creating an OKE Cluster Using Quick Cluster
- Creating an OKE Cluster Manually

## Creating an OKE Cluster Using Quick Cluster

The first step in preparing OCI is to create an OKE cluster. This step creates the virtual cloud network, the OKE cluster, and the worker nodes.

To create a quick cluster with default settings:

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Developer Services** (located in **Containers and Artifacts**) and click **Kubernetes Clusters**.
3. Click **Create Cluster**.
4. Select **Quick Create**.
5. Click **Launch Workflow**.

   The Create Cluster screen is displayed.
6. Enter the following details:
   - **Name**: Specify a name for the cluster. For example: IDMEDG.
   - **Compartment** - Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Node Type** - Select Managed.
   - **Kubernetes Version**: Select the version of Kubernetes you want to use. Ensure that the version you select is supported for IDM Kubernetes deployments.
   - **Kubernetes API End point**: Select **Private**. The Kubernetes cluster will not be exposed directly to the internet.
   - **Kubernetes Workers**: Select **Private**. The Kubernetes cluster will not be exposed directly to the internet.
   - **Shape**: Select the OCI shape you want to use to create the Kubernetes worker nodes. The shape you choose will depend on the number of worker nodes you want to create and the size of those nodes.
   - **Number of nodes**: Select the number of worker nodes you want to create.
7. Click **Show Advanced Options**. In the **Public SSH Key** box, copy the content of the `id_rsa.pub` file which you created earlier. See Creating an SSH Key Pair.
8. Click **Next**.
9. Review the summary and click **Create Cluster**.

The workflow creates:

- Virtual Cloud Network (VCN)
- Route Tables
- Security Lists
- Kubernetes Cluster
- Node Pool

10. Click **Close**.

# Creating an OKE Cluster Manually

To create an OKE cluster manually, you should complete the steps explained in this section. If you want to link two VCNs together, for example, use one VCN for the primary deployment and the other for the DR (Disaster Recovery) deployment, it is essential that the Network CIDRS/IP Addresses do not overlap.

For example, you could use 10.0.0.0/16 for your primary network and 10.1.0.0/16 for your DR network.

- Creating an Oracle Virtual Cloud Network
- Adding Additional Security Rules
- Creating an API Security List
- Creating an API Subnet
- Creating the OKE Cluster

## Creating an Oracle Virtual Cloud Network

To create an Oracle Virtual Cloud Network:

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Networking Virtual Cloud Networks**.
3. Click **Start VCN Wizard**.
4. Select **Create VCN with Internet Connectivity** and click **Start VCN Wizard**.
5. Enter the following information in the wizard:
   - **Name**: Select a name for the network. For example `idm_oke_vcn`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **VCN CIDR Block**: Enter the internal CIDR block you want to use for your network. For example: `10.0.0.0/16`.
   - **Public Subnet CIDR Block**: Enter the CIDR of the subnet you want to export to the internet. For example: `10.0.20.0/24`.
   - **Private Subnet CIDR Block**: Enter the CIDR of the subnet you want to use privately (this is where the Kubernetes worker nodes will reside). For example: `10.0.10.0/24`.
   - **Use DNS Hostnames in this VCN**: Select this option.
6. Click **Next**.
7. Review the summary information of the details specified and click **Create**.

8. When complete, click **View Virtual Cloud Network**.

These steps will create a public and private subnet.

## Adding Additional Security Rules

To add extra security rules to the default security list for OKE:

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Networking Virtual Cloud Networks**.
3. Select the newly created network **idm_oke_vcn**. See Creating an Oracle Virtual Cloud Network.
4. Click **Security Lists**.
5. Click the default security list. For example, security list for the private subnet - **idm_oke_vcn**.
6. Click **Add Ingress Rules** to add an Ingress rule as described in Table 10-1.
7. Click **Egress** in the Resources List
8. Click **Add Egress Rules** to add an Egress rule as described in Table 10-1.

**Table 10-1 Description of Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range | Type | Code |
|---|---|---|---|---|---|---|---|
| Ingress | CIDR | 10.0.10.0/24 | | All protocols | | | |
| Ingress | CIDR | 10.0.0.0/28 | | ICMP | | 3 | 4 |
| Ingress | CIDR | 10.0.0.0/28 | | TCP | | | |
| Ingress | CIDR | 0.0.0.0/0 | | TCP | 22 | | |
| Egress | CIDR | | 10.0.10.0/24 | All protocols | | | |
| Egress | CIDR | | 10.0.0.0/28 | TCP | 6433 | | |
| Egress | CIDR | | 10.0.0.0/28 | TCP | 12250 | | |
| Egress | CIDR | | 10.0.0.0/28 | ICMP | | 3 | 4 |
| Egress | Service | | All Services in Oracle Service Network | TCP | 443 | | |

## Creating an API Security List

Kubernetes requires an additional subnet to communicate with the Kubernetes control plane. To enable this communication, you should first create a security list.

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Networking Virtual Cloud Networks**.
3. Click the newly created network **idm_oke_vcn**. See Creating an Oracle Virtual Cloud Network.

4. Click **Security Lists** and select **Create Security List**.
5. Enter the following information:
   - **Name**: Enter a name for the security list. For example: `api-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
6. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-2 (repeat for each Ingress rule).
7. Click **Add Another Egress Rule** to add an Egress rule as described in Table 10-2 (repeat for each Egress rule).

**Table 10-2 Description of Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range | Type | Code |
|---|---|---|---|---|---|---|---|
| Ingress | CIDR | 0.0.0.0/0 | | TCP | 6443 | | |
| Ingress | CIDR | 10.0.10.0/24 | | TCP | 6443 | | |
| Ingress | CIDR | 10.0.10.0/24 | | TCP | 12250 | | |
| Ingress | CIDR | 10.0.10.0/24 | | ICMP | | 3 | 4 |
| Egress | CIDR | 10.0.10.0/24 | | TCP | | | |
| Egress | CIDR | 10.0.10.0/24 | | ICMP | | 3 | 4 |
| Egress | Service | All services in the Services Network. | | TCP | 443 | | |

8. Click **Create Security List**.

## Creating an API Subnet

Kubernetes requires an additional subnet to communicate with the Kubernetes control plane. To create an API subnet:

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Networking Virtual Cloud Networks**.
3. Click the newly created network **idm_vcn**. See Creating an Oracle Virtual Cloud Network.
4. Click **Subnets** and select **Create Subnet**.
5. Enter the following information:
   - **Name**: Enter a name for the subnet. For example: `api-subnet`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Subnet Type**: Select **Regional**.

- **CIDR Block**: Enter the CIDR of the subnet. For example: `10.0.0.0/28`.
- **Route Table**: Select **Default Route Table** for `idm_vcn`.
- **Subnet Access**: Select **Private**.
- **Use DNS Hostnames in this Subnet** - Select this option.
- **Security List**: Select the security list you created above: **api-seclist**. See Creating an API Security List .

6. Click **Create Subnet**.

## Creating the OKE Cluster

Now that you have created the network, create the OKE cluster:

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Developer Services** (located in **Solutions and Platform**) and click **Kubernetes Containers (OKE)**.
3. Click **Create Cluster** and select **Custom Create**.
4. Click **Submit**.
5. Enter the following information in the wizard:
   - **Name**: Enter a name for the cluster. For example: `idm-oke`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Kubernetes Version**: Select the version of Kubernetes you want to create.
6. Click **Next**.
7. Enter the following information in the Networking Setup page:
   - **Network Type**: Select **VCN-native pod networking**.
   - **VCN**: Select the VCN you created earlier. For example: **idm_oke_vcn**. See Creating an Oracle Virtual Cloud Network.
   - **Kubernetes Service LB Subnets**: Select the public subnet that was automatically created with the VCN.
   - **Kubernetes API Endpoint Subnet**: Select the API subnet you created earlier. For example: **api-subnet**. See Creating an API Subnet.
   - **Assign a Public IP Address to the API Endpoint** - Do not select this option.
   - Click **Next**.
8. Enter the following information in the Node Pools page:
   - **Name**: Specify a name for the pool. For example: `Pool1`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Node Type**: Choose Managed or Virtual depending on your requirements.
   - **Kubernetes Version**: Select the version of Kubernetes you want to create. This should be the same as the cluster version.
   - **Placement Configuration**: You should place the worker nodes in different fault domains. To place the worker nodes, create a placement for each fault zone. For

example, if you have three or four worker nodes, create a placement for three different fault domains. To create a placement, enter the following information:

- **Availability Domain**: Select **one**.
- **Worker Node Subnet**: Select the default private subnet which was created with the VCN.
- **Fault Domain**: Select one.

Click **Add another Row**.

Enter the same information again to distribute your worker nodes across different fault domains.

- **Shape and Image**: Enter details of the shape and capacity of the worker nodes you want to create. For example: **VM.Standard.E3.Flex**.
- **OCPU**: 4.
- **RAM**: 64GB.
- **Image**: Oracle Linux 8.
- **Number of Nodes**: The number of worker nodes you want to create.
- **Boot Volume**: You can use the default value or increase the size of the boot volume if you anticipate using different container images and versions.

9. Click **Next**.
10. Review the cluster summary, and then click **Create Cluster**.

# Creating a Bastion Node

You cannot access the cluster directly because the cluster is in a dedicated subnet. You can use a bastion node to access the cluster. The bastion node will be publicly available.

> **Note:**
>
> The basion node is the window to your environment. Therefore, access to the bastion node should be strictly controlled.

The creation of a bastion node includes the following steps:

- Creating Security Lists
- Creating a Route Table
- Creating a Subnet for the Bastion Node
- Adding the Security List to the Kubernetes Node Subnet
- Creating the Bastion Compute Instance
- Connecting to the Bastion Node
- Configuring the Bastion Node

## Creating Security Lists

You need to create security lists which enable the bastion node to communicate with the subnet that the Kubernetes cluster uses. In addition, you need to allow access to the bastion

node from the internet. This section describes the minimum steps you need to perform to enable this access. You should harden your security lists to ensure that only certain machines/networks have access to this node. Information about restricting access beyond the SSL key generated earlier is outside the scope of this document.

To create security lists:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Select **Security Lists** from the list of resources.

- Creating a Private Security List
- Creating a Public Security List
- Creating a Setup Security List

## Creating a Private Security List

To create a private security list:

1. Click **Create Security List**.
2. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `bastion-private-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
3. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-3 (repeat for each Ingress rule).
4. Click **Add Another Egress Rule** to add an Egress rule as described in Table 10-3 (repeat for each Egress rule).
5. Click **Create Security List**.

**Table 10-3 Description for Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range |
|---|---|---|---|---|---|
| Ingress | CIDR | 10.0.1.0/29 | | TCP | 22 |
| Ingress | CIDR | 10.0.1.0/29 | | ICMP | |
| Egress | CIDR | | 0.0.0.0/0 | All Protocols | |

> **Note:**
>
> 10.0.1.0 is the subnet you will use for the bastion node. You can change this value if required.

## Creating a Public Security List

To create a public security list:

1. Click **Create Security List**.

2. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `bastion-public-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
3. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-4 (repeat for each Ingress rule).
4. Click **Add Another Egress Rule** to add an Egress rule as described in Table 10-4 (repeat for each Egress rule).
5. Click **Create Security List**.

**Table 10-4 Description for Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range | Type |
|---|---|---|---|---|---|---|
| Ingress | CIDR | 0.0.0.0/0 | | TCP | 22 | |
| Ingress | CIDR | 10.0.1.0/29 | | ICMP | | 3 |
| Ingress | CIDR | 0.0.0.0/0 | | ICMP | | |
| Egress | CIDR | | 0.0.0.0/0 | All Protocols | | |

> **Note:**
>
> 10.0.1.0 is the subnet you will use for the bastion node. You can change this value if required. Unless otherwise stated, leave the values blank.

## Creating a Setup Security List

During the set up of Oracle Identity and Access Management, the bastion node requires access to some of the Kubernetes services that get created as part of the build process.

The access is not required after the build process is complete. For manageability reasons, a separate security list is created for this purpose. This way, after the setup is complete, you just have to remove the security list from the subnet. If further setups are required, you can add as needed.

The security list should be added to the following subnets:

- Private subnet for Node Manager
- db-subnet.

To create a setup security list:

1. Click **Create Security List**.
2. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `bastion-setup-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.

3. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-5 (repeat for each Ingress rule).
4. Click **Create Security List**.

**Table 10-5 Description for Ingress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Source Port Range | Destination Port Range | Comment |
|---|---|---|---|---|---|---|---|
| Ingress | CIDR | 10.0.1.0/29 | | TCP | | 30701 | OAM Administration Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.1.0/29 | | TCP | | 31800 | |
| Ingress | CIDR | 10.0.1.0/29 | | TCP | | 31920 | |

> **Note:**
>
> The destination ports listed above are dependent on the values you provide to your installation. Sample values will be used for consistency within this guide.
>
> The destination port range of 31800 and 31920 are required if you are deploying Elasticsearch.

## Creating a Route Table

You should create a route table which enables the bastion node to communicate with the subnet that the Kubernetes cluster uses. In addition, you should also enable access to the bastion node from the internet.

To create a route table:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Select **Route Tables** from the list of resources.
4. Click **Create Route Table**.
5. Enter the following details:
   - **Name**: Enter a name for the route table. For example: `bastion-route-table`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
6. Click **Add Another Route Rule**.
7. Enter the following information:
   - **Target Type** - Select **Internet Gateway**.
   - **Destination CIDR Block** - Enter `0.0.0.0/0`.

- **Compartment** - Select the name of the compartment you created earlier. See Creating an OCI Compartment.
- **Target Internet Gateway** - Select the internet gateway. For example: `oke-igw-quick-clustername-id`.

8. Click **Create Route Table**.

## Creating a Subnet for the Bastion Node

After you create the security rules and route table, you should create a subnet and assign the security rules and route table to it.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Click **Create Subnet**.
4. Enter the following details:
   - **Name**: Enter a name for the subnet. For example: `bastion-subnet`.
   - **Subnet Type**: Select **Regional**.
   - **CIDR Block**: Select the subnet you want to use for the bastion network. For example: `10.0.1.0/29`.
   - **Route Table**: Select the route table you created earlier. For example: `bastion-route-table`. See Creating a Route Table.
   - **Subnet Access**: Select **Public Subnet**.
   - **DNS Resolution**: Select **Use DNS Hostnames in the subnet**.
   - **Security List**: Select the public security list you created earlier. For example: `bastion-public-seclist`. See Creating a Public Security List.
5. Click **Create Subnet**.

## Adding the Security List to the Kubernetes Node Subnet

To enable communication between the bastion subnet and the Kubernetes Cluster Subnet, you need to add the private security list to the Kubernetes Node subnet.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. From your Kubernetes Cluster Summary screen, click the **VCN Name** that looks similar to `oke-vcn-quick-`*`clustername-id`*.
3. Click the Kubernetes node network that looks similar to `oke-nodesubnet-quick-`*`clustername-id-regional`*.
4. Click **Add Security List**.
5. Select your compartment.
6. Select the private bastion security list. For example: `bastion-private-seclist`.
7. Click **Add Security List**.
8. Repeat the above steps to add the and `bastion-setup-seclist` security list.

## Creating the Bastion Compute Instance

After defining the networking details, create the bastion node.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Compute** and click **Instances**.
3. Click **Create Instance**.
4. Enter the following information:
   - **Name**: A name for your bastion node. For example: `idm_bastion`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Placement**: Select an **Availability Domain**.
   - **Image**: Select the operating system image you want to use. For example: **Oracle Enterprise Linux 8.x**.
   - **Shape**: Select an architecture and shape you want to use. For example: **VM.Standard.E4.Flex**.
   - **Network**: Select the VCN that was created when you created the Kubernetes Cluster. See Creating an OKE Cluster in OCI.
   - **Subnet**: Select the bastion subnet you created earlier. For example: `bastion-subnet`. See Creating a Subnet for the Bastion Node.
5. Click **Assign public IP Address** to make this instance available from the internet.
6. In the **Add SSH Keys** box, select **Paste SSH Keys**.
7. Copy the contents of the `id_rsa.pub` file that you created earlier. See Creating an SSH Key Pair.
8. Click **Create**.

The summary screen displays the public IP address assigned to the bastion node. Make a note of this address. You will need it for connecting to the node.

## Connecting to the Bastion Node

You can now connect to the bastion node using the following SSH command:

```
ssh -i id_rsa opc@BastionIPAddress
```

Alternatively, if you are using SSH agent forwarding, which enables you to use your local **SSH** keys instead of leaving the keys (without passphrases) on the server, then you can use the following command:

```
ssh -A opc@BastionIPAddress
```

## Configuring the Bastion Node

After you create the bastion node, you need to configure it. Perform the following steps to configure the bastion node:

> **Note:**
>
> To perform the steps in this section, you will require the following information from the Oracle Cloud Infrastructure Console:
>
> - User OCID: To obtain your User OCID, click your profile in the OCI Console (top right) and select **User Settings** to view your OCID.
> - Tenancy OCID: To obtain your Tenancy OCID, click your profile in the Oracle Cloud Infrastructure Console (top right) and select your tenancy to view the tenancy OCID.
> - Region: The region in which you have deployed the cluster.



## Setting Up the OCI CLI to Download Kubeconfig

To set up the OCI CLI:

1. Ensure that you are using the latest version of Python by using the following command:

   ```
   python -V
   ```

   If you are using Python version 3.6, switch to the latest version by using the command:

   ```
   sudo alternatives --set python3 /usr/bin/python3.9
   ```

   Check the version again.

   Failure to do check the version may result in cryptography errors when you run the `kubectl` commands.

   If Python 3.9 is not available, then install it using the command:

   ```
   sudo yum install -y python39
   ```

2. Install OCI CLI.

   ```
   bash -c "$(curl -L https://raw.githubusercontent.com/oracle/oci-cli/master/
   scripts/install/install.sh)"
   ```

3. Respond to the prompts from the installation script.

4. To download kubeconfig later, after the set up, you need to set up the `oci config` file. Run the following command and enter the details when prompted:

```
oci setup config
```

Sample Setup:

```
$ oci setup config
    This command provides a walk through of creating a valid CLI config
file.

    The following links explain where to find the information required by
this
    script:

    User API Signing Key, OCID and Tenancy OCID:

        https://docs.cloud.oracle.com/Content/API/Concepts/
apisigningkey.htm#Other

    Region:

        https://docs.cloud.oracle.com/Content/General/Concepts/regions.htm

    General config documentation:

        https://docs.cloud.oracle.com/Content/API/Concepts/sdkconfig.htm

Enter a location for your config [/home/opc/.oci/config]:
Enter a user OCID: ocid1.user.oc1..xxxxxxxxxx
Enter a tenancy OCID: ocid1.tenancy.oc1..xxxxxxxxx
Enter a region (e.g. ap-hyderabad-1, ap-melbourne-1, ap-mumbai-1, ap-
osaka-1, ap-seoul-1, ap-sydney-1, ap-tokyo-1, ca-montreal-1, ca-toronto-1,
eu-amsterdam-1, eu-frankfurt-1, eu-zurich-1, me-jeddah-1, sa-saopaulo-1,
uk-gov-london-1, uk-london-1, us-ashburn-1, us-gov-ashburn-1, us-gov-
chicago-1, us-gov-phoenix-1, us-langley-1, us-luke-1, us-phoenix-1): us-
phoenix-1
Do you want to generate a new API Signing RSA key pair? (If you decline
you will be asked to supply the path to an existing key.) [Y/n]: Y
Enter a directory for your keys to be created [/home/opc/.oci]:
Enter a name for your key [oci_api_key]:
Public key written to: /home/opc/.oci/oci_api_key_public.pem
Enter a passphrase for your private key (empty for no passphrase):
Private key written to: /home/opc/.oci/oci_api_key.pem
Fingerprint: 74:d2:f2:db:62:a9:c4:bd:9b:4f:6c:d8:31:1d:a1:d8
Config written to /home/opc/.oci/config


    If you haven't already uploaded your API Signing public key through the
    console, follow the instructions on the page linked below in the
section
    'How to upload the public key':

        https://docs.cloud.oracle.com/Content/API/Concepts/
apisigningkey.htm#How2
```

5. The above command creates a keyfile called `oci_api_key_public.pem` in `$HOME/.oci`. Add this key to the Oracle Cloud Infrastructure Console.
   a. Log in to the Oracle Cloud Infrastructure Console.
   b. Select **Profile** and click **User Settings**.
   c. On the User Settings screen, select **API Keys** .
   d. Click **Add API Key**.
   e. Click **Paste Public Key**.
   f. Copy the contents of the `oci_api_key_public.pem` file to the **Public Key** block and click **Add**.
6. You now need to refer the Oracle Cloud Infrastructure Console to get the remaining steps to set up the bastion node. Each deployment is different:
   a. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
   b. From the Kubernetes Cluster Summary screen, click **Access Cluster**. A screen will be displayed with the remaining steps. These steps will include:
      - Creating a directory for the kube file.
      - Accessing the Kubeconfig file for the cluster.
      - Adding an environment variable to point to the cluster. (You should also add this variable to the `.bashrc` file for persistence.)

      For example:

      **Sample Cluster Access Steps on the bastion node**

```
$ oci -v


$ mkdir -p $HOME/.kube


$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.xxxxx
--file $HOME/.kube/config --region us-phoenix-1 --token-version 2.0.0


$ export KUBECONFIG=$HOME/.kube/config


$ echo "export KUBECONFIG=$HOME/.kube/config" > $HOME/.bashrc
```

7. Install the `kubectl` client to access the cluster from the bastion node.

   Enter the following commands to download the `kubectl` client:

```
$ curl -LO https://storage.googleapis.com/kubernetes-release/release/
v1.20.8/bin/linux/amd64/kubectl
$ sudo mv kubectl  /bin/
$ sudo chmod +x /bin/kubectl
```

> **Note:**
>
> Download the version appropriate to the version of Kubernetes you selected at the time of creating the OKE cluster. See Creating an OKE Cluster in OCI.
>
> If you are unsure of the Kubernetes version:
>
> a. Log in to the Oracle Cloud Infrastructure Console.
>
> b. Select **Developer Services** (located in **Solutions and Platform**), and then click **Kubernetes Clusters**.

8. Validate that `kubectl` works by using the following command:

```
kubectl get nodes
```

## Installing Helm

Helm is required by the WebLogic Operator and Oracle Unified Directory. To install Helm on to the bastion node, run the following commands:

```
$ curl -fsSL -o get_helm.sh
      https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$ chmod 700 get_helm.sh
$ ./get_helm.sh
$ helm version
version.BuildInfo{Version:"v3.13.1",
GitCommit:"3547a4b5bf5edb5478ce352e18858d8a552a4110",
GitTreeState:"clean", GoVersion:"go1.20.8"}
```

## Installing Git

Git contains sample code to deploy Oracle Fusion Middleware on Kubernetes. Install GIT using the following command:

```
sudo yum install git -y
```

## Installing X11 Packages

For security reasons, the Oracle HTTP Server is not installed inside the Kubernetes cluster. To install the Oracle HTTP Server, you need to install the X11 packages to enable X11 forwarding. Use the following command to install the X11 packages:

```
sudo yum install -y libXrender libXtst xauth xterm nc
```

## Installing Other Packages

If you are using the automation scripts provided in this guide, you will also need to install other packages. Use the following command to install other packages:

```
sudo yum install -y openldap java
```

For information about using automation scripts, see Automating the Identity and Access Management Enterprise Deployment.

## Enabling X11 Forwarding

Configure SSHD to not use localhost for X11:

1. Open `/etc/ssh/sshd_config` in your preferred editor.

   ```
   sudo vi /etc/ssh/sshd_config
   ```

2. Search for the line that has "X11UseLocalhost yes" (it is commented out).
3. Remove the comment from the beginning of the line.
4. Change the `yes` to `no`.
5. Save the file.
6. Restart SSHD by using the following command:

   ```
   sudo systemctl restart sshd
   ```

## Setting Up the Hosts File

When setting up, you should make `curl` commands to the load balancer. Because the bastion node uses the private DNS, the IP addresses returned for the load balancer end points is through the internal network that the bastion host does not have access to. To get around this issue, create an entry in the bastion hosts file for each entry point that points to the public IP address of the load balancer.

> **Note:**
>
> You cannot perform this step until you have created the load balancers. See Creating Load Balancers.

For example, if the public IP address of the load balancer is 129.1.1.3, add the following entry to the bastion hosts file:

```
129.1.1.3 login.example.com prov.example.com iadadmin.example.com
igdadmin.example.com
```

# Creating Compute Instances for Oracle HTTP Servers

The web tier resides in its own subnet separated from both the load balancer and the application tier. This section describes the procedures to create two compute instances for the web tier, place them in different fault domains, and set up security lists and route tables to facilitate access.



## Creating a Service Gateway

The web tier is not accessible to the internet directly. But it will require access to internal resources to perform operations such as accessing yum for adding the required packages and performing upgrades. To enable access to these internal systems, you need to create a service gateway if the system did not automatically create one for you.

To create a service gateway:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Virtual Cloud Networks**.
3. Click your Virtual Cloud Network. This is the same network that was created when you created the Kubernetes cluster. See Creating an OKE Cluster in OCI.
4. Select **Service Gateways** from the list of resources.
5. Click **Create Service Gateway**.
6. Enter the following information:
   - **Name**: Specify a name for the service gateway.
   - **Compartment**: Select the compartment you created earlier. See Creating an OCI Compartment.
   - **Services**: Select **All Services** in **Oracle Services Network**.
7. Click **Create Service Gateway**.

## Creating Security Lists

You need to create security lists which enable web tier nodes to communicate with the subnet that the Kubernetes cluster uses. In addition, you need to enable access to the web tier hosts from the load balancer. This section describes the minimum steps you need to perform to

enable this access. You should harden your security lists to ensure that only certain machines/networks have access to this node. This part is outside the scope of this guide.

To create security lists:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Select **Security Lists** from the list of resources.

## Creating an OHS Security List

During the running of Oracle Identity Management, the web tier hosts pass through the requests to the Kubernetes services that get created as part of the provisioning process. You need to create a security list to enable this communication to take place.

1. Click **Create Security List**.
2. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `ohs-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
3. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-6 (repeat for each Ingress rule).
4. Click **Create Security List**.

**Table 10-6 Description for Ingress Rules**

| Rule Type | Type | Source CIDR | Protocol | Destination Port Range | Comment |
|---|---|---|---|---|---|
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30701 | OAM Administration Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30510 | OAM Policy Manager Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30410 | OAM Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30711 | OIG Administration Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30140 | OIM Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30801 | SOA Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30901 | OUDSM Server Kubernetes Service Port |
| Ingress | CIDR | 10.0.2.0/28 | TCP | 30777 | Nginx Ingress Controller |

> **Note:**
>
> The destination ports listed in this table are dependent on the values you provide to your installation.

## Adding the OHS Security List to the Kubernetes Subnet

The security list now needs to be added to the subnet used by Kubernetes.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Select the **Kubernetes** sublist from the list of displayed subnets. The subnet will have a name similar to `oke-nodesubnet-<ClusterName>-<id>`.
4. Click **Add Security List**.
5. Select the compartment and the security list you created earlier. For example: `ohs-seclist`. See Creating an OCI Compartment and Creating an OHS Security List.
6. Click **Add Security List**.

## Creating a Route Table

You need to create a route table which enables the web tier nodes to communicate with the subnet that the Kubernetes cluster uses.

To create a route table:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Select **Route Tables** from the list of resources.
4. Click **Create Route Table**.
5. Enter the following details:
   - **Name**: Enter a name for the route table. For example: `web-route-table`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
6. Click **Add Another Route Rule** .
7. Enter the following information:
   - **Target Type** - Select **Service Gateway**.
   - **Destination Service** - Select **All XXX Services in Oracle Services Network**.
   - **Target Service Gateway** - Select the service gateway. For example: `oke-sgw-quick-clustername-id`.
8. Click **Create Route Table**.

## Creating a Subnet for Web Nodes

Now that you have created security rules and route table, you can create a subnet and assign the security rules and route table to it.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Click **Create Subnet**.
4. Enter the following details:
   - **Name**: Enter a name for the subnet. For example: `web-subnet`.
   - **Subnet Type**: Select **Regional**.
   - **CIDR Block**: Select the subnet you want to use for the web nodes network. For example: `10.0.2.0/28`.
   - **Route Table**: Select the route table you created earlier. For example: `web-route-table`. See Creating a Route Table.
   - **Subnet Access**: Select **Private Subnet**.
   - **DNS Resolution**: Select **Use DNS Hostnames in the subnet**.
   - **Security List**: Select the public security list you created earlier. For example: `web-public-seclist`. See Creating a Public Security List.
5. Click **Create Subnet**.

## Creating the OHS Compute Instances

Now that the networking has been defined, you can create the web tier nodes.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Compute** and click **Instances**.
3. Click **Create Instance**.
4. Enter the following information:
   - **Name**: A name for the OHS node. For example: `webhost1`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Placement**: Select an **Availability Domain**.
   - **Image**: Select **Oracle Enterprise Linux 8.x**.
   - **Shape**: Select an architecture and shape you want to use. For example: `VM.Standard.E4.Flex`.
   - **Network**: Select the VCN that was created when you created the Kubernetes Cluster. See Creating an OKE Cluster in OCI.
   - **Subnet**: Select the web-subnet you created earlier. See Creating a Subnet for Web Nodes.
5. Click **Do not assign public IPv4 Address** to make this instance unavailable directly from the internet.

6. In the **Add SSH Keys** box, select **Paste SSH Keys**.
7. Copy the contents of the `id_rsa.pub` file that you created earlier. See Creating an SSH Key Pair.

> **Note:**
>
> If you use the `id_rsa key` file you created on your laptop to connect to the web tier node, you must either copy that key to the web host or use SSH Agent forwarding. Alternatively, create a new key on the web tier node and use that key here.

8. Click **Create**.

The summary screen displays the private IP address assigned to the web tier node. Make a note of this address. You will need it for connecting to the node.

Repeat the steps for the second node.

## Editing OHS Compute Instance Fault Domain

Each OHS instance should reside in a different fault domain. Perform the following steps after the OHS compute instance is created:

1. Click the newly created compute instance.
2. Click **More Actions** and select **edit**.
3. Ensure each OHS compute instance is in a different fault domain, if not click **Edit Fault domain** and choose a different fault domain.
4. Click **Save changes**.

   Repeat the steps for each OHS compute instance.

## Connecting to the OHS Nodes

You cannot connect to the web tier hosts directly. You must use the bastion node. You can connect to the web tier hosts after you have connected to the bastion node. See Connecting to the Bastion Node.

> **Note:**
>
> When you created the compute instance, you specified the SSH key you would use to connect to the host. If you used the same key as your laptop/desktop, you should use SSH Agent forwarding to connect to the web host. Alternatively, you can also use the SSH key you created on the bastion host.
> You can now connect to the web host from the bastion node using the following SSH command:

```
ssh -i id_rsa opc@webhostIPAddress
```

Alternatively, if you are using SSH agent forwarding, which enables you to use your local **SSH** keys instead of leaving the keys (without passphrases) on the server, you can use the same pass through command:

```
ssh -A opc@webIPAddress
```

# Configuring the OHS Nodes

After you create the web tier nodes, you need to configure them. Perform the following steps to configure the nodes:

- Installing X11 Packages
- Installing Additional Packages
- Enabling X11 Forwarding
- Preparing the Compute Instance for Use by Oracle HTTP Server
- Using the Firewall
- Creating a Software Owner Account
- Preparing the Hosts File
- Connecting to the Compute Instances to Install OHS

## Installing X11 Packages

For security reasons, the Oracle HTTP Server is not installed inside the Kubernetes cluster. To install the Oracle HTTP Server, you have to install the X11 packages to enable X11 forwarding.

Use the following commands to install the X11 packages:

```
sudo yum repolist
```

```
sudo yum install -y libXrender libXtst xauth xterm nc xorg-x11*
```

## Installing Additional Packages

The Oracle HTTP Server requires additional packages to be present as part of the installation. Use the following command to install these additional packages:

```
sudo yum install -y libaio-devel* compat-libstdc++-* compat-libcap* gcc-c++-*
ksh* libnsl*
```

## Enabling X11 Forwarding

Configure SSHD to not use localhost for X11:

1. Open `/etc/ssh/sshd_config` in your preferred editor.
2. Search for the line that has "X11UseLocalhost yes" (it is commented out).
3. Remove the comment from the beginning of the line.
4. Change the `yes` to `no`.

5. Save the file.
6. Restart SSHD by using the following command:

```
sudo systemctl restart sshd
```

## Preparing the Compute Instance for Use by Oracle HTTP Server

You may also need to install additional Linux packages required to install the Oracle HTTP server, as well as setting the kernel parameters. See Preparing the Kubernetes Host Computers for an Enterprise Deployment.

## Using the Firewall

The compute instance is created using an Oracle Linux image. The image comes with a built-in firewall, which is enabled by default. Even though you have security rules defined in your network, the Linux server rejects these requests because of the built-in Linux firewall.

You can decide to use this extra firewall or rely on your OCI security rules.

- Opening the Ports in the Firewall
- Disabling the Firewall

### Opening the Ports in the Firewall

If you decide to use the firewall, you need to add firewall rules that enable every port coming in to the server to be allowed.

1. For every port that needs to be accessed, execute the following command:

```
sudo firewall-cmd --permanent --add-port=YOUR PORT/tcp
```

   For example:

```
sudo firewall-cmd --permanent --add-port=7777/tcp
```

2. Restart the firewall service after you configure all the ports. Use the following command to restart:

```
sudo systemctl restart firewalld
```

3. Validate the firewall configuration by executing the following command:

```
sudo firewall-cmd --list-ports
```

### Disabling the Firewall

To disable the firewall, run the following commands:

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
```

## Creating a Software Owner Account

It is not good practice to install the Oracle software using the OPC user. It is better to create a custom user to own the software. You can create a custom user by running the following commands:

```
sudo adduser -u 1001 oracle
sudo groupadd -g 1002 oinstall
sudo usermod -a -G oinstall oracle
sudo usermod -g oinstall oracle
```

## Preparing the Hosts File

The nature of the networks in an OCI environment means that the Oracle HTTP Server instances will not have access to the public load balancer. This can cause issues when the Oracle HTTP Server tries to access some virtual hosts.

In later sections, you will create a public load balancer for connections from the outside world to your system. See Creating a Public Load Balancer.

You will also create a private load balancer to allow you to route requests from the private subnets to this load balancer. See Creating a Private Load Balancer.

To ensure that the requests from the Oracle HTTP Server are directed to the private load balancer rather than the public, you should create an entry in the `/etc/hosts` file on the web hosts, which looks as follows:

```
IP ADDRESS OF PRIVATE LOAD BALANCER login.example.com
```

For example:

```
10.0.2.7 login.example.com
```

## Connecting to the Compute Instances to Install OHS

To install the Oracle HTTP Server, you will need access to a graphical display. To get this access, you should use X11 Forwarding.

1. From your desktop/laptop, install an 'X' Window server. For example: `XQuartz for MacOS`.
2. SSH to the bastion server by using the following command:

   ```
   ssh -AX opc@bastionserver
   ```

3. SSH to the web server by using the following command:

   ```
   ssh -AX oracle@webserver
   ```

For detailed instructions for installing OHS, see Installing and Configuring Oracle HTTP Server.

# Creating File Systems and Mount Targets

You need to create NFS file systems for Kubernetes Persistent Volumes and Oracle HTTP Server installations.

The filesystems that you have to create are described in Storage Requirements for an Enterprise Deployment.

- Overview of Preparing the File System for an Enterprise Deployment
- Summary of File Systems
- Creating a File System
- Setting the Mount Target Storage Reporting
- Creating a PV Security List
- Adding the Security List to the Subnet
- Mounting File Systems on Hosts

## Overview of Preparing the File System for an Enterprise Deployment

It is important to set up your storage in a way that makes the enterprise deployment easy to understand, configure, and manage.

This chapter provides an overview of the process of preparing the file system for an enterprise deployment. Oracle recommends setting up your storage according to information in this chapter. The terminology defined in this chapter is used in the diagrams and procedures throughout the guide.

## Summary of File Systems

See Table 4-3 for details of the file systems you have to create.

You have to mount the file systems to the bastion node only during the initial set up.

## Creating a File System

To create a file system:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Storage** and click **File Systems**.
3. Click **Create File System**.
4. Select **Filesystem for NFS**.
5. Click **Edit Details** in the **File System Information** section.
6. Enter the following details:
   - **Name**: Provide a name for the file system. For example: `oudpv`.
   - **Compartment**: Select the compartment you created earlier. See Creating an OCI Compartment.
7. Click **Edit Details** in the **Export Information** section.
8. Enter the following:

- **Export Path**: This is the path you want to export. For example: `/exports/IAMPVS/oudpv`.

9. Click **Edit Details** in the **Mount Target Information** section.
10. Enter the following details:
    - **Mount Target Name**: Specify a name for the mount target. For example: `IAMPV`
    - **Virtual Cloud Network**: Select the **VCN**.
    - **Subnet**:
      - For the persistent volumes, select the oke-node subnet.
      - For OHS1, select the subnet you created for the web tier.
11. Click **Create**.

> **Note:**
>
> Create a new mount target only for the first persistent volume (PV). Subsequent PVs should use the same mount target.

## Setting the Mount Target Storage Reporting

When you install Oracle products, the installer checks the available disk storage. This check fails when you use an OCI file system. The system displays a message saying that there is insufficient disk space. To overcome this error, you can configure OCI to report a specified amount of free space.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Storage** and click **Mount Targets**.
3. Click the **NFS** tab.
4. Select the OHS mount target for Availability Domain 1.

   The mount target is displayed.
5. Click **Edit** next to the **Reported Size (GB)** (it looks like a pencil).
6. Set an arbitrary size value. For example: 20.

   This value ensures that the file system, when mounted on the OHS nodes, reports 20GB of free space. This enables the OHS installer to proceed.
7. Click **Save**.

## Creating a PV Security List

Even though you have created the mount point in the same subnet as you want to use it, you still need to create a security list to access it. The web tier entries have already been added. However, you still need to create a security list for the OHS mount target.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. From your Kubernetes Cluster Summary screen, click the **VCN Name** that looks similar to `oke-vcn-quick-`*`clustername-id`*.
3. Select **Security Lists** from the list of resources.

4. Click **Create Security List**.
5. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `pv-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
6. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-7 (repeat for each Ingress rule).
7. Click **Add Another Egress Rule** to add an Egress rule as described in Table 10-7 (repeat for each Egress rule).
8. Click **Create Security List**.

**Table 10-7 Description for Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Source Port Range | Destination Port Range |
|---|---|---|---|---|---|---|
| Ingress | CIDR | 10.0.10.0/24 | | TCP | | 111 |
| Ingress | CIDR | 10.0.10.0/24 | | TCP | | 2048-2050 |
| Ingress | CIDR | 10.0.10.0/24 | | UDP | | 111 |
| Ingress | CIDR | 10.0.10.0/24 | | UDP | | 2048 |
| Ingress | CIDR | 10.0.1.0/29 | | TCP | | 111 |
| Ingress | CIDR | 10.0.1.0/29 | | TCP | | 2048-2050 |
| Ingress | CIDR | 10.0.1.0/29 | | UDP | | 111 |
| Ingress | CIDR | 10.0.1.0/29 | | UDP | | 2048 |
| Ingress | CIDR | 10.0.2.0/28 | | TCP | | 111 |
| Ingress | CIDR | 10.0.2.0/28 | | TCP | | 2048-2050 |
| Ingress | CIDR | 10.0.2.0/28 | | UDP | | 111 |
| Ingress | CIDR | 10.0.2.0/28 | | UDP | | 2048 |
| Egress | CIDR | | 10.0.10.0/24 | TCP | 111 | |
| Egress | CIDR | | 10.0.10.0/24 | TCP | 2048-2050 | |
| Egress | CIDR | | 10.0.10.0/24 | UDP | 111 | |
| Egress | CIDR | | 10.0.1.0/29 | TCP | 111 | |
| Egress | CIDR | | 10.0.1.0/29 | TCP | 2048-2050 | |
| Egress | CIDR | | 10.0.1.0/29 | UDP | 111 | |
| Egress | CIDR | | 10.0.2.0/28 | TCP | 111 | |
| Egress | CIDR | | 10.0.2.0/28 | TCP | 2048-2050 | |
| Egress | CIDR | | 10.0.2.0/28 | UDP | 111 | |

> **Note:**
>
> The rules for the bastion subnet are required only for the initial set up/configuration.

## Adding the Security List to the Subnet

To add the security list to the subnet:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** > **Virtual Cloud Networks** and click the name of your Virtual Cloud Network.
3. Select **oke-nodesubnet**.
4. Click **Add Security List**.
5. Select the security list you created earlier. For example: `pv-seclist`. See Creating a PV Security List.
6. Click **Add Security List**.
7. Repeat Steps 1 to 6 for the subnets `web-subnet` and the `bastion-subnet`.

## Mounting File Systems on Hosts

Each mount target has a different IP address. To determine how to mount a given file system:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Storage** and click **File Systems**.
3. Select a file system.
4. On the File System screen, select an export from the list of exports.
5. Click **Mount Commands** at the top of the screen, to view examples of the mount command.
6. For OHS hosts, place the entries in `/etc/fstab` with the following mount options:

   Sample OHS `/etc/fstab` entry:

   ```
   <IP>:/exports/IAMBINARIES/webbinaries1 /u02/private/oracle/products nfs
   auto,rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768
   <IP>:/exports/IAMCONFIG/webconfig1  /u02/private/oracle/config nfs
   auto,rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768
   ```

   Before you can use the file system with the containers, ensure that you can write to the file system. Mount the file system to the bastion node and write to it. If you are unable to write, use the `chmod` command to enable writing to the file system.

   For example:

   ```
   sudo mkdir -p /u02/private/oracle/products /u02/private/oracle/config
   sudo mount -a
   sudo chmod -R 777 /u02/private/oracle
   ```

**Table 10-8 Summary of Hosts and the File Systems to be Mounted**

| Mount Host | File Systems | Comments |
|---|---|---|
| webhost1 | webbinaries1 | Mounted as `/u02/private/oracle/products`. |
| webhost2 | webbinaries2 | Mounted as `/u02/private/oracle/products`. |
| webhost1 | webconfig1 | Mounted as `/u02/private/oracle/config`. |
| webhost2 | webconfig2 | Mounted as `/u02/private/oracle/config`. |
| All Kubernetes nodes | images<br>nfs_volumes* | Used as a staging directory to temporarily store container images.<br>Mounted as `/images`. |
| bastion node | oudconfigpv | Mounted as `/nfs_volumes/oudconfigpv`. |
| | oudpv | Mounted as `/nfs_volumes/oudpv`. |
| | oudsmpv | Mounted as `/nfs_volumes/oudsmpv`. |
| | oigpv | Mounted as `/nfs_volumes/oigpv`. |
| | oampv | Mounted as `/nfs_volumes/oampv`. |
| | oiripv | Mounted as `/nfs_volumes/oiripv`. |
| | dingpv | Mounted as `/nfs_volumes/dingpv`. |
| | oaacredpv | Mounted as `/nfs_volumes/oaacredpv`. |
| | oaaconfigpv | Mounted as `/nfs_volumes/oaaconfigpv`. |
| | oaalogpv | Mounted as `/nfs_volumes/oaalogpv`. |
| | oaavaultpv | Mounted as `/nfs_volumes/oaavaultpv`.<br>**Note**: Required when using a file-based vault. |

Optionally, mount all PVs. This option lets you delete deployments during the configuration phase, if necessary. Remove these mounts after the system is up and running.

> **Note:**
>
> * Alternatively, for these file systems, you can use block volumes.

# Creating Load Balancers

You need to create two OCI load balancers. One of these load balances is used to direct public traffic and the other for internal call backs. The load balancer used for internal traffic is not available outside the OCI container.

For more information about load balancers, see Getting Started with Load Balancing.

- Creating a Public Load Balancer
- Creating a Private Load Balancer

## Creating a Public Load Balancer

This load balancer directs traffic from the internet to the Oracle HTTP Servers, which in turn pass on the traffic to the Kubernetes pods.

The public load balancer will send traffic to and from the user via SSL but after the traffic moves inside the OCI Virtual Network, it is sent unencrypted. The decryption occurs due to SSL Termination. You will need to provide your own SSL certificate or create a self-signed certificate for testing purposes.

To create a public load balancer, perform the following steps:

- Creating a Self-Signed Certificate
- Creating a Security List
- Creating a Route Table
- Creating Subnets for the Load Balancer
- Creating a Load Balancer
- Uploading Load Balancer Certificates
- Creating Host Names
- Creating Listeners
- Updating the Default Listener

## Creating a Self-Signed Certificate

You can create a self-signed certificate on any host which has access to the `openssl` packages. The following example is from a Linux box (in this case the bastion server was used).

For more information, see Doc ID 2617046.1.

If you prefer, you can also use a certificate provided by a recognized certificate authority.

To create a self-signed certificate:

1. Create the CA (certificate authority) private key by using the following command:

```
openssl genrsa -out ca.key 2048


Generating RSA private key, 2048 bit long modulus
.....................+++
```

```
.....+++
e is 65537 (0x10001)
```

2. Create the Certificate Signing Request (CSR).

```
openssl req -new -key ca.key -out ca.csr


You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CR
State or Province Name (full name) []:SJO
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:mycompany
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:*.example.com
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Create the CA SIGN certificate that will be used to sign the new certificates.

```
openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt


Signature ok
subject=/C=CR/ST=SJO/L=Default City/O=mycompany/CN=*.example.com
Getting Private key
```

4. Create the private key for the load balancer.

```
openssl genrsa -out loadbalancer.key 2048


Generating RSA private key, 2048 bit long modulus
.....................+++
.....+++
e is 65537 (0x10001)
```

5. Create the CSR for the load balancer.

```
openssl req -new -key loadbalancer.key -out loadbalancer.csr


You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CR
State or Province Name (full name) []:SJO
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:mycompany
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:*.example.com
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

6. Sign the certificate with the CA certificate.

```
openssl x509 -req -in loadbalancer.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out loadbalancer.crt -days 50000


Signature ok
subject=/C=CR/ST=SJO/L=Default City/O=mycompany/CN=*.example.com
Getting CA Private Key
```

7. Check that the certificate is signed by the CA.

```
openssl x509 -in loadbalancer.crt -text


Certificate:
Data:
Version: 1 (0x0)
Serial Number:
df:e7:c9:6a:56:e5:e4:c9
Signature Algorithm: sha256WithRSAEncryption
Issuer: Issuer: C=CR, ST=SJO, L=Default City, O=mycompany,
CN=*.example.com <==== here signed by my ca..
Validity
Not Before: Dec 3 16:34:58 2019 GMT
Not After : Oct 25 16:34:58 2156 GMT
Subject: =/C=CR/ST=SJO/L=Default City/O=mycompany/CN=*.example.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:da:62:ce:69:77:ff:45:b0:84:9f:af:53:44:97:
13:28:91:44:cd:0b:1d:e5:a1:f6:a3:ef:f8:98:19:
8d:c2:56:a0:e1:80:1c:e0:0e:ae:34:9a:a8:ae:52:
d4:71:a4:da:10:8b:fd:df:73:0d:8e:98:ef:d4:7b:
36:f1:1c:5a:d7:24:88:63:f5:b2:6b:7a:62:50:3a:
e7:3a:3d:9a:b7:41:db:8e:f5:e8:91:46:48:cf:0c:
54:da:7b:da:20:76:b6:eb:4b:cb:fa:36:09:f7:94:
ea:c9:53:3f:b2:bc:66:4c:6d:7f:3f:09:cc:cd:c2:
10:1f:39:0f:6c:1d:49:7c:db:99:d9:d9:7d:48:dd:
09:52:50:9d:f5:44:fd:2e:48:f2:78:22:20:3c:07:
```

```
b6:a1:4d:f8:17:82:67:a1:45:52:0a:21:78:ed:1b:
ca:45:79:16:21:c9:e3:2f:a4:93:d4:bf:67:68:7a:
b6:d9:8f:e1:53:35:31:a6:17:38:f2:a6:79:b5:12:
6b:36:f2:2d:69:56:c2:d9:c0:89:d9:31:6b:06:0c:
1e:ba:a6:30:88:32:7b:92:e4:af:11:ab:37:1a:cb:
cf:4b:4c:7d:ff:a7:4d:f8:be:cd:98:17:63:83:06:
cf:e7:ae:4a:d5:6e:6b:e4:0d:f3:6f:70:52:2b:8b:
12:83
Exponent: 65537 (0x10001)
Signature Algorithm: sha256WithRSAEncryption
d8:36:2e:2e:42:72:76:15:ec:a8:3a:e9:dd:2d:2e:28:42:97:
48:4e:6f:33:ec:df:3e:a3:11:19:8b:62:d5:89:07:af:b5:ff:
b6:de:d7:5c:8b:7a:46:37:46:da:b7:44:7f:b6:cc:c8:a9:1e:
f9:ca:0f:76:2b:29:d2:4c:6a:af:18:9b:1a:62:42:87:e6:21:
b7:09:15:8d:b3:1d:05:4a:4d:1b:d1:07:00:cd:69:40:92:ed:
f9:3d:24:c9:b7:b9:00:7e:c3:f9:73:42:7f:13:34:a8:d1:e4:
32:91:08:51:07:a5:d0:ab:42:fb:83:c4:a7:b5:94:0f:2a:56:
8b:95:34:1b:63:5b:39:59:88:9b:9f:34:91:98:dc:8c:0a:0e:
01:f9:b2:6e:fd:2e:95:28:4c:76:dd:fe:a0:3f:f1:16:3b:88:
cd:e5:0a:f3:dd:52:0d:39:2a:60:2c:f0:5d:79:3b:7e:99:43:
3b:47:33:85:f9:7c:f1:e8:cb:3d:cd:ab:4c:1f:a2:72:99:70:
f4:8d:92:4a:24:9e:37:96:ad:24:d5:13:33:05:32:ae:d5:58:
ed:3e:32:6f:a7:1e:a8:61:a5:fb:73:ea:54:46:b7:07:77:07:
9a:9d:af:eb:66:5c:55:f1:50:23:fb:da:d9:b7:4b:0b:6d:bb:
c7:39:18:ae
-----BEGIN CERTIFICATE-----
MIIDUDCCAXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXAQsFADBaMQswCQYDVQQGEwJD
UjEMMAoGA1UECAwDU0pPMRUwEwYDVQQHDAxEZWZhdWx0IENpdHkxDzANBgNVBAoM
Bm9yYWNsZTEVMBMGA1UEAwwMKi5vcmFjbGUuY29tMCAXDTE5MTIwMzE2MzQ1OFoY
DzIxNTYxMDI1MTYzNDU4WjB4MQswCQYDVQQGEwJDUjEMMAoGA1UECAwDU0pPMRUw
EwYDVQQHDAxEZWZhdWx0IENpdHkxHDAaBgNVBAoME0RlZmF1bHQgQ29tcGFueSBM
dGQxDzANBgNVBAsMBm9yYWNsZTEVMBMGA1UEAwwMKi5vcmFjbGUuY29tMIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2mLOaXf/RbCEn69TRJcTKJFEzQsd
5aH2o+/4mBmNwlag4YAc4A6uNJqorlLUcaTaEIv933MNjpjv1Hs28Rxa1ySIY/Wy
a3piUDrnOj2at0HbjvXokUZIzwxU2nvaIHa260vL+jYJ95TqyVM/srxmTG1/PwnM
zcIQHzkPbB1JfNuZ2dl9SN0JUlCd9UT9LkjyeCIgPAe2oU34F4JnoUVSCiF47RvK
RXkWIcnjL6ST1L9naHq22Y/hUzUxphc48qZ5tRJrNvItaVbC2cCJ2TFrBgweuqYw
iDJ7kuSvEas3GsvPS0x9/6dN+L7NmBdjgwbP565K1W5r5A3zb3BSK4sSgwIDAQAB
MA0GCSqGSIb3DQEBCwUAA4IBAQDYNi4uQnJ2FeyoOundLS4oQpdITm8z7N8+oxEZ
i2LViQevtf+23tdci3pGN0bat0R/tszIqR75yg92KynSTGqvGJsaYkKH5iG3CRWN
sx0FSk0b0QcAzWlAku35PSTJt7kAfsP5c0J/EzSo0eQykQhRB6XQq0L7g8SntZQP
KlaLlTQbY1s5WYibnzSRmNyMCg4B+bJu/S6VKEx23f6gP/EWO4jN5Qrz3VINOSpg
LPBdeTt+mUM7RzOF+Xzx6Ms9zatMH6JymXD0jZJKJJ43lq0k1RMzBTKu1VjtPjJv
px6oYaX7c+pURrcHdweana/rZlxV8VAj+9rZt0sLbbvHORiu
-----END CERTIFICATE-----
```

This procedure creates the following files to be used later. See Uploading Load Balancer Certificates.

- `ca.crt`
- `loadbalancer.crt`
- `loadbalancer.key`

## Creating a Security List

The security list determines who can access the load balancer and where the load balancer is allowed to send requests.

To create a private security list:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Virtual Cloud Networks**.
3. Click the **VCN Name** that looks similar to `oke-vcn-quick-clustername-id`.
4. Select **Security Lists** from the list of resources.
5. Click **Create Security List**.
6. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `public-lbr-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
7. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-9 (repeat for each Ingress rule).
8. Click **Add Another Egress Rule** to add an Egress rule as described in Table 10-9 (repeat for each Egress rule).
9. Click **Create Security List**.

**Table 10-9 Description for Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range |
|---|---|---|---|---|---|
| Ingress | CIDR | 0.0.0.0/0 | | TCP | 80 |
| Ingress | CIDR | 0.0.0.0/0 | | TCP | 443 |
| Egress | CIDR | | 10.0.2.0/28 | TCP | 7777 |

> **Note:**
>
> 10.0.2.0 is the subnet you will use for the web tier.

## Creating a Route Table

You need to create a route table which enables the load balancer to communicate with the internet.

To create a route table:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Virtual Cloud Networks**.
3. Click the **VCN Name** that looks similar to `oke-vcn-quick-clustername-id`.
4. Select **Route Tables** from the list of resources.

5. Click **Create Route Table**.
6. Enter the following details:
   - **Name**: Enter a name for the route table. For example: `lbr-route-table`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
7. Click **Add Another Route Rule** .
8. Enter the following information:
   - **Target Type**: Select **Internet Gateway**.
   - **Destination CIDR**: Enter `0.0.0.0/0`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Target Internet Gateway**: Select the internet gateway. For example: `oke-igw-quick-`*`clustername-id`*.
9. Click **Create**.

## Creating Subnets for the Load Balancer

The public load balancer is placed into an isolated subnet. The load balancer is created as a pair so that if one fails, the second one takes on the work load. The load balancers reside in availability domains and a different subnet is created for each load balancer. By using different subnets for the load balancers, you create stricter access rules enabling public access only to the load balancer but not the components for which it load balances.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Virtual Cloud Networks**.
3. Click the **VCN Name** that looks similar to `oke-vcn-quick-`*`clustername-id`*.
4. Click **Create Subnet**.
5. Enter the following details:
   - **Name**: Enter a name for the subnet. For example: `lbr-subnet1`.
   - **Subnet Type**: Select **Regional**.
   - **Availability Domain**: Select an availability domain.
   - **CIDR Block**: Select the subnet you want to use for the load balancer network. For example: `10.0.4.0/24`.
   - **Route Table**: Select the route table you created earlier. For example: `lbr-route-table`. See Creating a Route Table.
   - **Subnet Access**: Select **Public Subnet**.
   - **DNS Resolution**: Select **Use DNS Hostnames in the subnet**.
   - **DHCP Options**: Select **Default DHCP Options** for the VCN.
   - **Security List**: Select the public security list you created earlier. For example: `public-lbr-seclist`. See Creating a Security List.
6. Click **Create Subnet**.

## Creating a Load Balancer

To create a load balancer:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click **Create Load Balancer**.
4. Select **Load Balancer** and click **Create Load Balancer**.
5. Enter the following information:
   - **Name**: Enter a name for the load balancer. For example: `public-loadbalancer`.
   - **Visibility Type**: Select **Public**.
   - Select **Assign a Public IP Address** and **Ephemeral IP Address** unless you want to use a specific IP address, in which case select **Reserved IP Address**.
   - **Shapes**: Select **Flexible Shape**.
   - **Bandwidth**: Select the anticipated bandwidth.
   - **Virtual Cloud Network**: Select the Virtual Cloud Network.
   - **Subnet**: Select both the load balancer subnets you created earlier. See Creating Subnets for the Load Balancer.
6. Click **Next**.
7. On the Choose Back Ends screen, select your preferred **Load Balancing Policy**.
8. Click **Add Back Ends**.
   a. Select **Web Server Instances**.
   b. Click **Add Selected Back Ends**.
   c. Change the port to the listen port for Oracle HTTP Server. For example: `7777`.
9. In the Specify Health Check Policy screen, change the port to the HTTP server port. For example: `7777`.
10. Click **Show Advanced Options** and enter a name for the back-end set. For example: `ohs_servers`.
11. Click **Next**.
12. On the Configure Listener screen, enter the following information:

> **Note:**
>
> You will need one listener for each entry point. However, you can add only one listener at this point.

   - **Name**: Select a name for the listener. For example: `iadadmin`.
   - **Traffic Type**: Select the traffic type the listener uses. `iadadmin.example.com` uses **HTTP**.
   - **Port**: Select the load balancer port. `iadadmin.example.com` uses port `80`.
13. Click **Next**.

14. In the Manage Logging screen, ensure that **Create a New Log Group** is selected.
15. Change **Name**. For example: `Public_Lbr`.
16. Change **Log Name**. For example: `Public_lbr_error`.
17. Click **Submit**.

## Uploading Load Balancer Certificates

As the load balancer routes SSL requests, you need to upload the certificates for the load balancer. If you have created a self-signed certificate, add the details of that certificate. If you have your own certificates, upload those.

To upload the certificates:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **public_loadbalancer**.
4. Select **Certificates** from the resource list.
5. Select **Load Balancer Managed Certificate**.
6. Click **Add Certificate**.
7. Enter the following information. You can either upload the files directly or paste the contents of the files.
   - **Certificate Name**: Enter a name for the certificate. For example: `Loadbalancer`.
   - **SSL Certificate**: Include the contents of the `loadbalancer.crt` file.
   - **CA Certificate**: Select the **Specify CA Certificate** check box to include the contents of the `ca.crt` file.
   - **Private Key**: Select the **Specify Private Key** check box to include the contents of the `loadbalancer.key` file.

   See Creating a Self-Signed Certificate.
8. Click **Add Certificate**.

## Creating Host Names

Host names are used to filter the different entry points into the load balancer. You need to create a host name for each load balancer virtual host described in Summary of the Load Balancer Virtual Servers Required for an Enterprise Deployment.

You have to create the following host names:

- `iadadmin.example.com`
- `igdadmin.example.com`
- `login.example.com`
- `prov.example.com`

To create the load balancer host names:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.

3. Click the load balancer. For example: **public_loadbalancer**.
4. Select **Hostnames** from the resource list.
5. Click **Create Hostname**.
6. Enter the following information:
   - **Name**: Enter a name for the host name. For example: `iadadmin`.
   - **Hostname**: Enter the fully qualified host name. For example: `iadadmin.example.com`.
7. Click **Create**.
8. Repeat for each host name to be created.

> **Note:**
>
> If you want to limit the admin access to users inside the network, you should create the hosts `iadadmin.example.com` and `igdadmin.example.com` in the private load balancer.

## Creating Listeners

You need to create a listener for each host name you have created earlier. See Creating Host Names. The `iadadmin` listener has been created at the time of creating the load balancer. See Creating a Load Balancer.

**Table 10-10 Summary of Public Load Balancer Listeners**

| Name | Protocol | Port | SSL | Backend Set | Host Name |
|---|---|---|---|---|---|
| `iadadmin` | http | 80 | | ohs_servers | `iadadmin.example.com` |
| `igdadmin` | http | 80 | | ohs_servers | `igdadmin.example.com` |
| `login` | https | 443 | Yes | ohs_servers | `login.example.com` |
| `prov` | https | 443 | Yes | ohs_servers | `prov.example.com` |

To create the load balancer listeners:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **public_loadbalancer**.
4. Select **Listeners** from the resource list.
5. Click **Create Listener**.
6. Enter the following information:
   - **Name**: Enter a name for the listener. For example: `login`.
   - **Protocol**: Select **https**.

- **Port**: Specify `443`.
- **Certificate Name**: Ensure that the certificate you created for the load balancer is displayed. If not displayed, select the certificate.

> **Note:**
>
> This option will be available only if you use the `HTTPS` protocol.

- **Hostname**: Select **login**.
- **Backend Set**: Select the back end set. For example: **ohs_servers**.

7. Click **Create Listener**.
8. Repeat the steps to create the remaining listeners.

> **Note:**
>
> If you want to limit the admin access to users inside the network, create the listeners `iadadmin.example.com` and `igdadmin.example.com` in the private load balancer.

## Updating the Default Listener

When you created the load balancer, a default listener also gets created. You have to assign the newly created host name to this listener.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **public-loadbalancer**.
4. Select **Listeners** from the resource list.
5. To edit the listener, click the three dots next to the name, and then click **Edit**.
6. Set the host name to the host name you created earlier. For example: `iadadmin`. See Creating Host Names.
7. Click **Update Listener**.

# Creating a Private Load Balancer

The private load balancer, which is used to route internal call backs, resides in the same subnet as the Oracle web servers. This load balancer services requests generated from inside the application.

> **Note:**
>
> Web servers issue `curl` commands to `login.example.com`. Therefore, you also need to define this on the private load balancer because the web servers do not have direct access to the public load balancer. You can use the same certificates that you used when you created the public load balancer.

To create a private load balancer, perform the following steps:

- Creating a Load Balancer
- Creating Host Names
- Updating the Default Listener
- Uploading Load Balancer Certificates
- Creating Listeners

## Creating a Load Balancer

To create a load balancer:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click **Create Load Balancer**.
4. Select **Load Balancer** and click **Create Load Balancer**.
5. Enter the following information:
   - **Name**: Enter a name for the load balancer. For example: `internal-loadbalancer`.
   - **Visibility Type**: Select **Private**.
   - **Shapes**: Select **Flexible Shape**.
   - **Bandwidth**: Select the anticipated bandwidth.
   - **Virtual Cloud Network**: Select the Virtual Cloud Network.
   - **Subnet**: Select the same subnet as the web servers. For example: `web-subnet`. See Creating Subnets for the Load Balancer.
6. Click **Next**.
7. On the Choose Back Ends screen, select your preferred **Load Balancing Policy**.
8. Click **Add Back Ends**.
   a. Select **Web Server Instances**.
   b. Click **Add Selected Back Ends**.
   c. Change the port to the listen port for Oracle HTTP Server. For example: `7777`.
9. In the Specify Health Check Policy screen, change the port to the HTTP server port. For example: `7777`.
10. Click **Show Advanced Options** and enter a name for the back-end set. For example: `ohs_servers`.
11. Click **Next**.
12. On the Configure Listener screen, enter the following information:
    - **Name**: Select a name for the listener. For example: `igdinternal`.
    - **Traffic Type**: Select the traffic type the listener uses. `igdinternal` uses **HTTP**.
    - **Port**: Select the load balancer port. `igdinternal` uses port `7777`.
13. Click **Next**.
14. In the Manage Logging screen, ensure that **Create a New Log Group** is selected.

15. Click **Submit**.

## Creating Host Names

Host names are used to filter the different entry points into the load balancer. You need to create a host name for each load balancer virtual host described in Summary of the Load Balancer Virtual Servers Required for an Enterprise Deployment.

You have to create the following host names:

- `igdinternal.example.com`
- `login.example.com`
- `iadadmin.example.com`
- `igdadmin.example.com`

> **Note:**
>
> `login.example.com` is defined here for internal traffic routing. The EDG uses network segregation. If you do not define it here, calls to `login.example.com` will attempt to communicate using the public network and fail.

To create the load balancer host name:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **internal_loadbalancer**.
4. Select **Hostnames** from the resource list.
5. Click **Create Hostname**.
6. Enter the following information:
   - **Name**: Enter a name for the host name. For example: `igdinternal`.
   - **Hostname**: Enter the fully qualified host name. For example: `igdinternal.example.com`.
7. Click **Create**.
8. Repeat Steps 5 through 7 to create each of the required host names.

## Updating the Default Listener

When you created the load balancer, a default listener also gets created. You have to assign the newly created host name to this listener.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **internal_loadbalancer**.
4. Select **Listeners** from the resource list.
5. To edit the listener, click the three dots next to the name, and then click **Edit**.

6. Set the host name to the host name you created earlier. For example: `igdinternal`. See Creating Host Names.
7. Click **Update Listener**.

## Uploading Load Balancer Certificates

As the load balancer routes SSL requests, you need to upload the certificates for the load balancer. If you have created a self-signed certificate, add the details of that certificate. If you have your own certificates, upload those.

To upload the certificates:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **internal_loadbalancer**.
4. Select **Certificates** from the resource list.
5. Click **Add Certificate**.
6. Enter the following information. You can either upload the files directly or paste the contents of the files.
   - **Name**: Enter a name for the certificate. For example: `Loadbalancer`.
   - **SSL Certificate**: Include the contents of the `loadbalancer.crt` file.
   - **CA Certificate**: Select the **Specify CA Certificate** check box to include the contents of the `ca.crt` file.
   - **Private Key**: Select the **Specify Private Key** check box to include the contents of the `loadbalancer.key` file.

   See Creating a Self-Signed Certificate.
7. Click **Add Certificate**.

## Creating Listeners

You need to create a listener for each host name you have created earlier. See Creating Host Names.

**Table 10-11 Summary of Private Load Balancer Listeners**

| Name | Protocol | Port | SSL | Backend Set | Host Name |
|---|---|---|---|---|---|
| igdinternal | http | 7777 | No | ohs_servers | `igdinternal.example.com` |
| login | https | 443 | Yes | ohs_servers | `login.example.com` |
| iadadmin | http | 80 | No | ohs_servers | `iadadmin.example.com` |
| igdadmin | http | 80 | No | ohs_servers | `igdadmin.example.com` |

To create the load balancer listeners:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.

2. Navigate to **Networking** and click **Load Balancers**.
3. Click the load balancer. For example: **internal_loadbalancer**.
4. Select **Listeners** from the resource list.
5. Click **Create Listener**.
6. Enter the following information:
   - **Name**: Enter a name for the listener. For example: `login.example.com`.
   - **Protocol**: Select **HTTPS**.
   - **Port**: Specify `443`.
   - **Certificate Name**: Ensure that the certificate you created for the load balancer is displayed. If not displayed, select the certificate.
   - **Hostname**: Select **login**.
   - **Backend Set**: Select the back end set. For example: **ohs_servers**.
7. Click **Create Listener**.

# Creating a Network Load Balancer

This step is required only if you want to configure a load balancer to route traffic to the Kubernetes worker nodes.

To create a network load balancer:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Load Balancers**.
3. Click **Create Load Balancer**.
4. Select **Network Load Balancer** and click **Create Load Balancer**.
5. In the **Create Network Loadbalancer** section, specify the following information:
   a. **Load Balancer Name**: Select a name for your load balancer. For example: **k8workers**.
   b. **Visibility Type**: Select **Private**.
   c. **Virtual Cloud Network**: Select **Virtual Cloud Network**.
   d. **Subnet**: Select the same subnet as the Kubernetes worker nodes. For example: **one-nodesubnet-quick-<clustername>-<id>**.
   e. **Compartment**: Select the compartment.
6. Click **Next**.
7. In the Listener screen, specify the following information:
   a. **Listener**: Select **TCP**.
   b. **Type**: Select **TCP**.
   c. Select **Use Any Port**.
8. Click **Next**.
9. In the Backend Set screen, specify the following information:
   a. **Backend Set Name**: Select **K8Workers**.
   b. Click **Add Backends**.

c. In the Backends screen, ensure that all the worker nodes are selected, and then click **Add Backends**.

d. **Health Check Policy** - Select **TCP** and set the port to `22`. Use the default values for all other values.

10. Click **Next**.
11. Review the details and click **Create**.

# Creating a Database

There are several different databases that you can create in OCI. For this example, a bare metal RAC database will be created. You may need to create one or more databases.

See Preparing an Existing Database for an Enterprise Deployment for details on the databases and services you should create. This section shows an example of creating one of these databases in OCI.

- Creating a Security List
- Creating a Route Table
- Creating Subnets for the Database
- Creating the Database
- Creating a Secondary Pluggable Database
- Connecting to the Database Node
- Configuring the Database

## Creating a Security List

To create a private security list:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Virtual Cloud Networks**.
3. Click the **VCN Name** that looks similar to `oke-vcn-quick-`*`clustername-id`*.
4. Click **Create Security List**.
5. Enter the following details:
   - **Name**: Enter a name for the security list. For example: `db-seclist`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
6. Click **Add Another Ingress Rule** to add an Ingress rule as described in Table 10-12 (repeat for each Ingress rule).
7. Click **Add Another Egress Rule** to add an Egress rule as described in Table 10-12 (repeat for each Egress rule).

**Table 10-12 Description for Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range |
|---|---|---|---|---|---|
| Ingress | CIDR | 0.0.0.0/0 | | TCP | 22 |

**Table 10-12 (Cont.) Description for Ingress and Egress Rules**

| Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Destination Port Range |
|---|---|---|---|---|---|
| Ingress | CIDR | 10.0.11.0/24 | | TCP | 1521 |
| Ingress | CIDR | 10.0.11.0/24 | | TCP | 6200 |
| Ingress | CIDR | 10.0.10.0/24 | | TCP | 1521 |
| Ingress | CIDR | 10.0.10.0/24 | | TCP | 6200 |
| Ingress | CIDR | 10.0.1.0/29 | | TCP | 1521<br>**Note**: Used for set up only. |
| Egress | CIDR | | 0.0.0.0/0 | All Protocols | |

> **Note:**
>
> 10.0.11.0 is the subnet to use for the database. You can change this value, if required.

## Creating a Route Table

You need to create a route table which enables the database to communicate with the OKE cluster.

To create a route table:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Virtual Cloud Networks**.
3. Click the **VCN Name** that looks similar to `oke-vcn-quick-clustername-id`.
4. Select **Route Tables** from the list of resources.
5. Click **Create Route Table**.
6. Enter the following details:
   - **Name**: Enter a name for the route table. For example: `db-route-table`.
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
7. Click **Add Another Route Rule** .
8. Enter the following information:
   - **Target Type** - Select **Service Gateway**.
   - **Destination Service** - Select **All XXX Services in Oracle Services Network**.
   - **Compartment** - Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Target Service Gateway** - Select the service gateway. For example: `oke-sgw-quick-clustername-id`.

9. Click **Create Route Table**.

## Creating Subnets for the Database

The database is placed into an isolated subnet.

To create subnets for the database:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking** and click **Virtual Cloud Networks**.
3. Click the **VCN Name** that looks similar to `oke-vcn-quick-clustername-id`.
4. Click **Create Subnet**.
5. Enter the following details:
   - **Name**: Enter a name for the subnet. For example: `db-subnet`.
   - **Subnet Type**: Select **Regional**.
   - **CIDR Block**: Select the subnet you want to use for the database network. For example: `10.0.11.0/24`.
   - **Route Table**: Select the route table you created earlier. For example: `db-route-table`. See Creating a Route Table.
   - **Subnet Access**: Select **Private Subnet**.
   - **DNS Resolution**: Select **Use DNS Hostnames in the subnet**.
   - **DHCP Options**: Select **Default DHCP Options** for the VCN.
   - **Security List**: Select the security list you created earlier. For example: `db-seclist`. See Creating a Security List.
6. Click **Create Subnet**.

## Creating the Database

After establishing the network, you can now create the database.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Oracle Database** and click **Oracle Base Database (VM, BM)**.
3. Click on **Create DB System**.
4. Enter the following information:
   - **Compartment**: Select the name of the compartment you created earlier. See Creating an OCI Compartment.
   - **Name**: Enter a name for the database infrastructure. For example: `Identity_Management_Databases`.
   - **Availability Domain**: Select an availability domain.
   - **Shape Type**: For this example, select **Virtual Machine**.
   - **Choose a Shape**: This value depends on your sizing requirements.
   - **Configure DB System**: Select a node count greater than 1.
   - **Storage Management Software**: Select **Oracle Grid Infrastructure**.
   - **Configure Storage**: Select the sizing requirements for your storage.

- In the **Add SSH Keys** box, select **Paste SSH Keys**.
- Copy the contents of the `id_rsa.pub` file that you created earlier. See Creating an SSH Key Pair.
- **License Type**: Select the type of database license you have.
- **Virtual Cloud Network**: Click the **VCN Name** that looks similar to `oke-vcn-quick-clustername-id`.
- **Client Subnet**: Select the DB subnet. For example: `db-subnet`.
- **Host Name Prefix**: Select a host name prefix. For example: `db`.
- **Database Unique Name Suffix:** Set it to a value unique to your system, which is especially important if you are going to create a disaster recovery site. The best practice is to set the suffix to the abbreviated region. For example: `lon` for London.
- **Database Image:** Select the database release you want to use. For example: `21c`.

5. Click **Next**.
6. On the Database Information screen, enter the following information:
   - Database Name: Select a name for your database. For example: `iamdb1`.
   - **PDB Name** : Enter a name for the Oracle Access Manager PDB you want to create. For example: `iadpdb`.
   - **Sys Password** : Select a password you want to assign to the database SYS account.
   - **Workload Type**: Select **Transaction Processing**.
   - **Configure Database Backups**: Select **Enable automatic backups**.
   - **Backup Retention Period:** Specify the period fo which you want to keep the database backups.
   - **Backup Scheduling:** Specify the preferred time to initiate the backup.
7. Click **Create DB System**.
8. After the database is created, note the following values for use at a later point:
   - **SCAN DNS Name**: This is the host name you use to connect to the database.
   - **db node 1** and **db node 2**: To obtain the names/IP addresses of these nodes, click **Nodes** from the resources list.

> **Note:**
>
> After the database is created, OCI adds a suffix to the database name. Ensure that you use the complete name including this suffix when configuring the database as described in Preparing an Existing Database for an Enterprise Deployment.

## Creating a Secondary Pluggable Database

When you create the database, it creates a single pluggable database (PDB). A single PDB may be sufficient for your needs.

However, if you require more PDBs so that OAM, OIG, OIRI, and OAA use different PDBs in the same database, you have to create additional PDBs. See Creating a PDB Using an Existing PDB as a Template.

You can do this either at the database level or, if you are using Oracle OCI, through the OCI console. For adding a PDB at the database level, see Creating a PDB Using an Existing PDB as a Template.

Alternatively, you can create extra pluggable databases by using the OCI Console.

1. Log in to the Oracle Cloud Infrastructure for your tenancy.
2. Navigate to **Oracle Database** and click **Oracle Base Database (VM, BM)**.
3. Click the DB system hosting the database.
4. Click the **Container Database Name** from the list of displayed databases.
5. Click **Pluggable Databases** in the **Resources** menu.
6. Click **Create Pluggable Database**.
7. Add a name for the new pluggable database and enter the TDE wallet password for the container database. This password may be the same as the database SYS password if you opt not to set an explicit value.
8. Click **Create Pluggable Database**.
9. Repeat **Step 6** through **Step 8** for each additional pluggable database you require.

## Connecting to the Database Node

You can now connect to the database node using the following SSH command:

```
ssh -A opc@databaseNodeIPAdddress
```

Connect to DB node 1 from the bastion node using the command:

```
ssh -A opc@dbnode1
```

After you connect to DB node 1 as `opc`, connect to the oracle user using the following command:

```
sudo su - oracle
```

## Configuring the Database

After you create the skeletal database, you should configure the database as described in Preparing an Existing Database for an Enterprise Deployment.

# Creating a Vault

A vault is used to store the credentials of your deployment. At present, the only Oracle Identity and Access Management product using a vault is Oracle Advanced Authentication (OAA). OAA can use either an OCI-based vault (recommended) or a file-based vault.

If you are planning to use an OCI-based vault, create the vault using the following steps:

1. Log in to the Oracle Cloud Infrastructure for your tenancy.
2. Select **Identity and Security** and click **Vault**.
3. Click **Create Vault** and specify the following details:

a. **Compartment** - Select the compartment you created earlier. See Creating an OCI Compartment.

b. **Name** – Enter a name for the vault. For example: `oaavault`.

c. Select **Make it a private vault**.

d. Click **Create Vault**.

- Creating the Vault Key
- Creating the API Key

## Creating the Vault Key

To create the vault key:

1. Click the name of the newly created vault. For example: **oaavault**.
2. Click **Create Key**.
3. Enter the following information:
   - **Compartment** - Select the compartment you created earlier. Creating an OCI Compartment.
   - **Protections Mode**: **Software**.
   - **Name** - Enter a name for the key. For example: `vaultkey`.
4. Click **Create Key**.

## Creating the API Key

To create the API key:

1. Log in to the Oracle Cloud Infrastructure Console.
2. Select **Profile** and click **User Settings**.
3. On the User Settings screen, select **API Keys**.
4. Click **Add API Key**.
5. Click **Download API Key**. Keep this file safe.
6. Click **Add**.
7. Click **Close**.

# Creating a DNS Server

This is an optional task. It is important that all host names are resolvable, including the load balancer virtual hosts. You can make them resolvable by adding entries to the local hosts files. However, in OCI, using a private DNS server is the simpler method.

By default, the compute hosts are configured to use a private DNS server. You have to add the entries only for the local hosts.

- Creating a DNS Zone
- Creating DNS Records

# Creating a DNS Zone

To create a DNS zone:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Navigate to **Networking**, select **DNS Management**, and then click **Zones**.
3. Click **Private Zones**.
4. Click **Create Zone**.
5. Enter the following information:
   - **Name**: Enter a name for the zone. For example: `example.com`.
   - Select **Existing DNS Private View**.
   - **DNS Private View**: Select **Virtual Cloud Network**.
6. Click **Create**.

# Creating DNS Records

After you create the zone, you can create records in the zone for each host. There are two types of DNS records that have to be created:

- **A Record**: This is an IP address association with a host name.
- **CNAME**: This is an alias for the A Record.

If you have multiple hosts using the same IP address, Oracle recommends you to create one 'A Record' and multiple 'CNAME' records.

To create a record:

1. Click **Add Record**.
2. Select the **Record Type**: **A** or **CNAME**.
3. Specify the name of the host in the domain. For example: `loadbalancer.example.com`.
4. Specify the **Address** which is the IP Address of the host. For example: The IP address of the public load balancer.

   OR

   Specify the **Target** which is the name of the **A** record with which you want to associate the alias.
5. Set the **TTL** value to `86400`. If the **TTL** field is disabled, select the lock icon at the end of the row to specify a value.
6. Click **Submit**.

> **Note:**
>
> To continue adding another record, select the **ADD ANOTHER RECORD** check box. After you click **Submit**, the Add Record screen remains open to add another record.

You have to create the following entries:

**Table 10-13 DNS Record Type and the Associated Host Name**

| Host Name | Type | Target | Address |
|---|---|---|---|
| `loadbalancer.example.com` | A | | IP address of the Internal load balancer. |
| `iadadmin.example.com` | CNAME | `loadbalancer.example.com` | |
| `igdadmin.example.com` | CNAME | `loadbalancer.example.com` | |
| `login.example.com` | CNAME | `loadbalancer.example.com` | |
| `prov.example.com` | CNAME | `loadbalancer.example.com` | |
| `igdinternal.example.com` | A | | IP address of the Internal load balancer. |
| `webhost1.example.com` | A | | IP address of *WEBHOST1*. |
| `webhost2.example.com` | A | | IP address of *WEBHOST2*. |

7. After entering all your entries, click **Publish** to ensure that they are made available.

# Updating Kubernetes CoreDNS

The Kubernetes cluster resolves hostnames using the built-in CoreDNS server. By default, this server will not interact with the corporate DNS server. You must configure this server to either perform local hostname resolution for the application end points or to resolve those end points using the corporate DNS server.

To configure the CoreDNS server:

1. Create a config map with your custom hosts in called `coredns_custom.yaml`. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: kube-system
data:
  example.server: |
      example.com {
            hosts {
            1.1.1.1 login.example.com
            1.1.1.2 prov.example.com
            1.1.1.3 iadadmin.example.com
            1.1.1.4 igdadmin.example.com
            1.1.1.5 igdinternal.example.com
            fallthrough
```

```
            }
        }
```

2. Save the file.
3. Create the config map using the following command:

   ```
   kubectl create -f coredns_custom.yaml
   ```

4. Restart CoreDNS using the command:

   ```
   kubectl rollout restart -n kube-system deploy coredns
   ```

   Ensure that the CoreDNS pods restart without any issue, using the command:

   ```
   kubectl get pods -n kube-system
   ```

   Ensure that the custom config has been loaded by using the following command:

   ```
   kubectl get configmaps --namespace=kube-system coredns-custom -o yaml
   ```

   If any errors occur, use the following command to view them:

   ```
   kubectl logs -n kube-system coredns--<ID>
   ```

   Correct the errors by editing the configmap again.

# Validating Your Environment

Perform the checks described in this section to ensure that your environment is ready for a deployment.

**For the bastion node**

- Check the network connectivity

  ```
  ping webhost1.example.com
  ```

  ```
  ping webhost2.example.com
  ```

- Resolve the public address of the load balancer

  ```
  ping login.example.com
  ```

  ```
  ping prov.example.com
  ```

- Check that Kubernetes is working

  ```
  kubectl get nodes
  ```

  Ping each of the worker nodes that are listed as the output of the above command.

**From the Web Tier**

- Ensure that the kubernetes worker node names are resolvable

  ```
  nslookup k8workers
  ```

- Verify if the web tier can communicate with the Kubernetes worker nodes which includes the port that you communicate on. If you are using an ingress controller, then this will be the node port assigned to that ingress controller. If you are using individual nodePort Services, then you should check each of those ports.

  Use the following command to check network connectivity:

  ```
  nc -zv <WORKER_NODE> <PORT>
  ```

  For example, if your ingress server is using NodePort server 30777 then the command is as follows:

  ```
  nc -zv 1.1.1.1 30777
  ```

  Ncat: Version 7.70 ( https://nmap.org/ncat ).

  Ncat: Connected to 1.1.1.1:30777.

  Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.

- Resolve the public address of the load balancer

  ```
  ping login.example.com
  ```

# Preparing a Disaster Recovery Environment

A disaster recovery environment is a replica of your primary environment located in a different region from the primary region. This environment is a standby environment that you switch over to in the event of the failure of your primary environment.

The standby environment will be a separate cluster, ideally in a different data center. If the cluster is dedicated to the application, the second cluster should be a mirror of the primary cluster with the same number and specifications of worker nodes. If your cluster is a multi-purpose cluster that is used by different applications, ensure sufficient spare capacity in the standby site to run the full application workload of the primary cluster.

Each Kubernetes cluster will run the same operating system version and the Kubernetes major release.

Your network will be such that:

- The primary and standby database networks communicate with each other to facilitate the creation of a Data Guard database.
- The primary and standby file system networks communicate with each other to facilitate the replication of the file system data. If you have to run the `Rsync` process to achieve the replication inside the cluster, then the primary Kubernetes worker network will be able to communicate with the Kubernetes worker network on the standby site.
- A global load balancer will be used to direct the traffic between the primary and standby sites. This load balancer is often independent of the site-specific load balancers used for on-site communication.

- The SSL certificates used in the load balancers must be the same in each load balancer. The traffic should not be aware when the load balancer switches sites.

There are several ways to create a DR environment. This document makes the following assumptions:

- You will create an exact replica of your primary environment.
- The DR environment will reside in the same tenancy as your primary environment.
- The DR environment will reside in the same compartment as your primary environment.
- The DR environment will reside in a different region from your primary environment.

You will create the standby environment using the same steps as the primary environment, followed by the steps to create an OKE cluster, with the following characteristics:

- The VCN must use different CIDRs. For example,`10.0.0.0/16` for the primary cluster and `10.1.0.0/16` for the standby cluster.
- Subnets will use different IP address ranges. For example, a primary subnet may be `10.0.4.0/24` and the equivalent standby subnet would be `10.1.4.0/24`.
- The load balancer virtual host names will be the same.
- The load balancer SSL certificates will be the same.
- The ports used will be the same.
- The webhost names will be the same.
- Dedicated bastion nodes will exist.

After you create both the environments, complete the following additional steps:

- Creating a Dynamic Routing Gateway
- Creating a Dynamic Routing Gateway Attachment
- Creating a Remote Peering Connection
- Connecting the Site 1 and Site 2 VCNs
- Creating Routing Tables for Dynamic Routing Gateways
- Creating the Security Lists for Subnets
- Checking the Site Connectivity

## Creating a Dynamic Routing Gateway

A dynamic routing gateway (DRG) is required to ensure that two different virtual cloud networks (VCNs) can communicate with each other. Each site requires a DRG. Therefore, perform the following steps on each site.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Dynamic Routing Gateway**.
3. Click **Create Dynamic Routing Gateway** and enter the following information:
    - **Name**: Provide a meaningful name for the gateway. For example, `site1-drg`.
    - **Compartment**: Select the compartment you created earlier. See Creating an OCI Compartment.
4. Click **Create Dynamic Routing Gateway**.

## Creating a Dynamic Routing Gateway Attachment

After you create the DRG, you should attach it to the VCN of the site.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Virtual Cloud Networks**.
3. Select your Virtual Cloud Network. This is the same network that was created when you created the Kubernetes cluster. See Creating an OKE Cluster in OCI.
4. From the list of resources, select **Dynamic Routing Gateways Attachments**.
5. Click **Create Virtual Cloud Network Attachment** and enter the following information:
   - **Name**: Provide a name for the attachment. For example, `DRG-Attachment-Site1`.
   - **DRG**: Select the Dynamic Routing Gateway you created earlier. For example: `site1-drg`. See Creating a Dynamic Routing Gateway.

   Leave the remaining options at the default values.
6. Click **Create Virtual Cloud Network Attachment**.

## Creating a Remote Peering Connection

To create a Remote Peering Connection (RPC):

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Dynamic Routing Gateway**.
3. Select the Dynamic Routing Gateway you created earlier. For example, `site1-DRG`. See Creating a Dynamic Routing Gateway.
4. From the list of resources, select **Remote Peering Connections Attachments**.
5. Click **Create Remote Peering Connection** and enter the following information:
   - **Name**: Provide a meaningful name for the connection. For example, `site1-RPC`.
   - **Compartment**: Select the compartment you created earlier. See Creating an OCI Compartment.
6. Click **Create Remote Peering Connection**.

## Connecting the Site 1 and Site 2 VCNs

Before you perform this step, ensure that you have created the Dynamic Routing Gateway (RPG) (see Creating a Dynamic Routing Gateway and the Remote Peering Connection (RPC) (see Creating a Remote Peering Connection) on both the sites. The following steps will help link the two sites.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Dynamic Routing Gateway**.
3. Select the Dynamic Routing Gateway you created earlier. For example, `site1-DRG`. See Creating a Dynamic Routing Gateway.
4. From the list of resources, select **Remote Peering Connections Attachments**.
5. From the Remote Peering Connections Attachments section, select the Remote Peering Connection you created earlier. For example, `site1-RPC`. See Creating a Remote Peering

Connection. Make a note of the **Remote Peering Connection OCID** displayed at the top of the screen.

Perform the following steps on Site 2:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Dynamic Routing Gateway**.
3. Select the Dynamic Routing Gateway you created earlier. For example, `site1-DRG`. See Creating a Dynamic Routing Gateway.
4. From the list of resources, select **Remote Peering Connections Attachments**.
5. From the Remote Peering Connections Attachments section, select the Remote Peering Connection you created For Site 2. For example, `site2-RPC`. For instructions, see Creating a Remote Peering Connection.
6. Click **Establish Connection** and enter the following information:
   - **Region**: Select the region that hosts Site 1.
   - **Remote Peering Connection OCID**: Enter the RPC OCID from Site 1, obtained above.
7. Click **Establish Connection**.
   The **Peering Status** changes to **Peered**. This change may take a few minutes.

## Creating Routing Tables for Dynamic Routing Gateways

For subnets in Site 1 to communicate with subnets in Site 2, you should create routing entries in both directions.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Virtual Cloud Networks**.
3. Select your Virtual Cloud Network. This is the same network that was created when you created the Kubernetes cluster. See Creating an OKE Cluster in OCI.
4. From the list of resources, select **Route Table**.
5. Select the route table you want to update. For example, `db-route-table`.
6. From the list of resources, select **DRG Route Tables**.
7. Click **Add Route Rules** and enter the following information:
   - **Target Type**: Dynamic Routing Gateway.
   - **Destination Type**: CIDR Block
   - **Destination CIDR Block**: 10.1.11.0/24
   - **Description**: Traffic to DRG
8. Click **Add Route Rules**.

   Repeat for each rule in the tables below:

**Table 10-14 Routing Rules for Site 1**

| Route Name | Destination CIDR | Next Hop Attachment | Attachment Name |
|---|---|---|---|
| db-route | 10.1.11.0/24 | Virtual Cloud Network | DRG-Attachment-Site1 |
| oke-node-subnet | 10.1.10.0/24 | Virtual Cloud Network | DRG-Attachment-Site1 |

**Table 10-15 Routing Rules for Site 2**

| Route Name | Destination CIDR | Next Hop Attachment | Attachment Name |
|---|---|---|---|
| db-route | 10.1.11.0/24 | Virtual Cloud Network | DRG-Attachment-Site2 |
| oke-node-subnet | 10.1.10.0/24 | Virtual Cloud Network | DRG-Attachment-Site2 |

# Creating the Security Lists for Subnets

After creating routes between the subnets, you need to create security lists for each network and add them to the corresponding subnet.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Virtual Cloud Networks**.
3. From the list of resources, select **Security Lists**.
4. Select a security list. For example, `db-seclist`.
5. From the list of resources, select **Ingress Rules**.
6. Click **Add Ingress Rules**.
7. Enter the information, as described in Table 10-14 and Table 10-15.
8. Click **Add Ingress Rule**.
9. From the list of resources, select **Egress Rules**.
10. Click **Add Ingress Rules**.
11. Enter the information, as described in Table 10-14 and Table 10-15.
12. Click **Add Ingress Rule**.

**Table 10-16 Security List for Site 1**

| List | Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Source Port Range | Destination Port Range | Type |
|---|---|---|---|---|---|---|---|---|
| db-seclist | Ingress | CIDR | 10.1.11.0/24 | | TCP | | 1521 | |
| db-seclist | Ingress | CIDR | 10.1.11.0/24 | | TCP | | 6200 | |
| Security List for Private Subnet for VCN | Ingress | CIDR | 10.1.11.0/24 | | | | 31444 | |
| Pv-seclist | Ingress | CIDR | 10.1.11.0/24 | | TCP | | 111 | |
| Pv-seclist | Ingress | CIDR | 10.1.11.0/24 | | TCP | | 2048-2050 | |
| Pv-seclist | Ingress | CIDR | 10.1.11.0/24 | | UDP | | 111 | |
| Pv-seclist | Ingress | CIDR | 10.1.11.0/24 | | UDP | | 2048 | |

**Table 10-16 (Cont.) Security List for Site 1**

| List | Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Source Port Range | Destination Port Range | Type |
|---|---|---|---|---|---|---|---|---|
| Pv-seclist | Egress | CIDR | 10.1.11.0/24 | | TCP | 111 | | |
| Pv-seclist | Egress | CIDR | 10.1.11.0/24 | | TCP | 2048-2050 | | |
| Pv-seclist | Egress | CIDR | 10.1.11.0/24 | | UDP | 111 | | |
| Pv-seclist | Egress | CIDR | 10.1.11.0/24 | | UDP | 2048 | | |

**Table 10-17 Security List for Site 2**

| List | Rule Type | Type | Source CIDR | Destination CIDR | Protocol | Source Port Range | Destination Port Range | Type |
|---|---|---|---|---|---|---|---|---|
| db-seclist | Ingress | CIDR | 10.0.11.0/24 | | TCP | | 1521 | |
| db-seclist | Ingress | CIDR | 10.0.11.0/24 | | TCP | | 6200 | |
| Security List for Private Subnet for VCN | Ingress | CIDR | 10.0.11.0/24 | | | | 31444 | |
| Pv-seclist | Ingress | CIDR | 10.0.11.0/24 | | TCP | | 111 | |
| Pv-seclist | Ingress | CIDR | 10.0.11.0/24 | | TCP | | 2048-2050 | |
| Pv-seclist | Ingress | CIDR | 10.0.11.0/24 | | UDP | | 111 | |
| Pv-seclist | Ingress | CIDR | 10.0.11.0/24 | | UDP | | 2048 | |
| Pv-seclist | Egress | CIDR | 10.0.11.0/24 | | TCP | 111 | | |
| Pv-seclist | Egress | CIDR | 10.0.11.0/24 | | TCP | 2048-2050 | | |
| Pv-seclist | Egress | CIDR | 10.0.11.0/24 | | UDP | 111 | | |
| Pv-seclist | Egress | CIDR | 10.0.11.0/24 | | UDP | 2048 | | |

# Checking the Site Connectivity

After configuring the Dynamic Gateway and Security Lists, use the Network Path Analyzer to validate inter-region connectivity.

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Networking** and click **Network Path Analyzer**.
3. Click **Create Path Analysis**.
4. On the Configure Analysis screen, enter the following information:
   - **Name**: Enter a name for the test. For example, `DB Check`.
   - **Source IP Address**: Enter the IP address of one of the database hosts in Site 1.
   - **Destination Address**: Enter the IP address of one the database hosts in Site 2.
   - **Destination Port**: Enter the database port `1521`.
5. Click **Run Analysis**.
6. Ensure that the test is successful before you continue.
7. Repeat the test for each destination port in tables Table 10-16 and Table 10-17.

11

# Preparing an Existing Database for an Enterprise Deployment

As part of preparing an existing database for an enterprise deployment, you should ensure that the database meets specific requirements. Other tasks include creating database services, using SecureFiles for large objects in the database, and creating database backup strategies. In a Kubernetes deployment, the database should reside outside of the cluster.

This chapter includes the following topics:

- About Preparing the Database for an Enterprise Deployment
  You have to configure a supported database as part of an Oracle Fusion Middleware enterprise deployment. Most Oracle Fusion Middleware products require a specific set of schemas that must be installed in a supported database. The schemas are installed by using the Oracle Fusion Middleware Repository Creation Utility (RCU).
- About Database Requirements
  Before you configure the enterprise deployment topology, you have to verify that the database meets the requirements described in the following sections.
- Adding Database Options
  Oracle Identity Management requires that Oracle JVM and Oracle Text is available in the database. By default, these options are not available in the database created.
- Adding XA Views
  Oracle Identity Governance requires the use of XA views.
- Applying the Database Patches
  Ensure that you apply all the necessary database patches. The patches are required only if you are using Oracle Identity Governance.
- Creating a PDB Using an Existing PDB as a Template
  When you create a database, an empty PDB is created at the same time. This example used the empty PDB to create a secondary PDB with the same configuration.
- Creating Database Services
  When multiple Oracle Fusion Middleware products are sharing the same database, each product should be configured to connect to a separate, dedicated database service. This service should be different from the default database service.
- Preventing Password Timeouts for System Accounts
- Using SecureFiles for Large Objects (LOBs) in an Oracle Database
  SecureFiles is a new LOB storage architecture introduced in Oracle Database 11g Release 1. It is recommended to use SecureFiles for the Oracle Fusion Middleware schemas, in particular for the Oracle SOA Suite schemas.
- About Database Backup Strategies
  Performing a database backup at key points in the installation and configuration of an enterprise deployment enables you to recover quickly from any issue that might occur in the later configuration steps.

# About Preparing the Database for an Enterprise Deployment

You have to configure a supported database as part of an Oracle Fusion Middleware enterprise deployment. Most Oracle Fusion Middleware products require a specific set of schemas that must be installed in a supported database. The schemas are installed by using the Oracle Fusion Middleware Repository Creation Utility (RCU).

In an enterprise deployment, Oracle recommends a highly available Real Application Clusters (Oracle RAC) database for the Oracle Fusion Middleware product schemas.

# About Database Requirements

Before you configure the enterprise deployment topology, you have to verify that the database meets the requirements described in the following sections.

- Supported Database Versions
- Additional Database Software Requirements
- Databases Required
- Minimum Initialization Parameters

## Supported Database Versions

Use the following information to verify what databases are supported by each Oracle Fusion Middleware release and which version of the Oracle database you are currently running:

- For a list of all certified databases, refer to *Oracle Fusion Middleware Supported System Configurations*.
- To check the release of your database, query the `PRODUCT_COMPONENT_VERSION` view:

```
SQL> SELECT VERSION FROM SYS.PRODUCT_COMPONENT_VERSION WHERE
         PRODUCT LIKE 'Oracle%';
```

Oracle Fusion Middleware requires that the database supports the AL32UTF8 character set. Check the database documentation for information on choosing a character set for the database.

For enterprise deployments, Oracle recommends that you use GridLink data sources to connect to Oracle RAC databases.

> **Note:**
>
> For more information about using GridLink data sources and SCAN, see Using Active GridLink Data Sources in *Administering JDBC Data Sources for Oracle WebLogic Server*.
>
> Use of Active GridLink has specific licensing requirements, including a valid WebLogic Suite license. See Oracle WebLogic Server data sheet.

## Additional Database Software Requirements

In the enterprise topology, there are two database host computers in the data tier that host the two instances of the RAC database. These hosts are referred to as DBHOST1 and DBHOST2.

Before you install or configure the enterprise topology, you must ensure that the following software is installed and available on DBHOST1 and DBHOST2:

- **Oracle Clusterware**

  See Installing Oracle Grid Infrastructure for a Cluster in *Oracle Grid Infrastructure Installation Guide for Linux*.

- **Oracle Real Application Clusters**

  See Installing Oracle RAC and Oracle RAC One Node in *Oracle Real Application Clusters Installation Guide for Linux and UNIX*.

- **Time synchronization between Oracle RAC database instances**

  The clocks of the database instances must be in sync if they are used by servers in a Fusion Middleware cluster configured with server migration.

- **Automatic Storage Management** (optional)

  See Introducing Oracle Automatic Storage Management in *Oracle Automatic Storage Management Administrator's Guide*.

**General Database Characteristics**

- Character Set – The character set must be unicode compliant. For example: AL32UTF8.
- Database Options – The following database options must be installed into the database:
  - Oracle JVM
  - Oracle Text
- Database Views – The following database view must be created on the database:
  - XAVIEWS
- Database Packages – The following database package must exist in the database:
  - DBMS_SHARED_POOL

## Databases Required

For Oracle Identity and Access Management, a number of separate databases are recommended. Table 11-1 provides a summary of these databases. Which database or databases you use depends on the topology that you are implementing.

For this release of Oracle Identity and Access Management, you must use a separate RCU schema prefix for each domain. This allows different products to use a the same or different databases if required.

If you are planning on creating a Multi-Datacenter, you should use separate databases for Access and Governance. This allows different replication mechanisms to be used for each.

Oracle Identity and Access Management fully supports the use of Container (PDB) databases. If you utilize the same disaster recovery strategy for all components, for example, Active/Passive then, utilizing a single container database provides many benefits.

However, if you are using a hybrid disaster recovery strategy and Oracle Access Manager is using Active/Active multi-data center while everything else is using the Active/Passive strategy, then the Access Manager database must be a separate database from the others because you can use Oracle Data Guard only at the container database level.

**Table 11-1 Mapping between Databases and Schemas for an Active/Passive Disaster Recovery Strategy**

| Container Database | PDB Name | Database Hosts | Scan Address | Service Name | RCU Prefix | Schemas in Database |
|---|---|---|---|---|---|---|
| IAMDB | IADPDB | DBHOST1<br>DBHOST2 | DBSCAN | iadedg.example.com | EDGIAD | OAM, IAU, MDS, OPSS |
| IAMDB | IGDPDB | DBHOST1<br>DBHOST2 | DBSCAN | igdedg.example.com | EDGIGD | OIM, SOAINFRA, MDS, OPSS, ORASDPM, BI, ODS |
| IAMDB | OAAPDB | DBHOST1<br>DBHOST2 | DBSCAN | oaaedg.example.com | EDGOAA | OAA |
| OIRIDB | | DBHOST1<br>DBHOST2 | DBSCAN | oiriedg.example.com | EDGOIRI | OIRI DING |

**Table 11-2 Mapping between Databases and Schemas for a Hybrid Disaster Recovery Solution**

| Container Database | PDB Name | Database Hosts | Scan Address | Service Name | RCU Prefix | Schemas in Database |
|---|---|---|---|---|---|---|
| IAMDB1 | IADPDB | DBHOST1<br>DBHOST2 | DBSCAN | iadedg.example.com | EDGIAD | OAM, IAU, MDS, OPSS |
| IAMDB2 | IGDPDB | DBHOST1<br>DBHOST2 | DBSCAN | igdedg.example.com | EDGIGD | OIM, SOAINFRA, MDS, OPSS, ORASDPM, BI, ODS |
| IAMDB2 | OAAPDB | DBHOST1<br>DBHOST2 | DBSCAN | oaaedg.example.com | EDGOAA | OAA |
| OIRIDB | | DBHOST1<br>DBHOST2 | DBSCAN | oiriedg.example.com | EDGOIRI | OIRI DING |

> **Note:**
>
> In this scenario, OAAPDB is shown as a separate PDB but it could be placed into IGDPDB, if desired.

## Minimum Initialization Parameters

The databases must have the following minimum initialization parameters defined:

**Table 11-3 Minimum Initialization Parameters for Oracle Databases**

| Parameter | Development | Small System | Medium System | Large System |
|---|---|---|---|---|
| aq_tm_processes | 1 | 1 | 1 | 1 |
| dml_locks | 200 | 200 | 200 | 200 |
| job_queue_processes | 12 | 12 | 12 | 12 |
| open_cursors | 1600 | 1600 | 1600 | 1600 |
| session_max_open_files | 50 | 50 | 50 | 50 |
| sessions | 4000 | 4000 | 4000 | 4000 |
| processes | 5000 | 5000 | 5000 | 5000 |
| sga_target | 5G | 28G | 58G | 118G |
| pga_aggregate_target | 2G | 7G | 14G | 29G |
| pga_aggregate_limit | 0 | 0 | 0 | 0 |
| sga_max_size | 5 G | 28 G | 58 G | 118 G |
| session_cached_cursors | 1000 | 1000 | 1000 | 1000 |
| db_keep_cache_size | 0 | 800M | 800M | 800M |
| cursor_sharing | FORCE | FORCE | FORCE | FORCE |
| query_rewrite_integrity | TRUSTED | TRUSTED | TRUSTED | TRUSTED |
| query_rewrite_enabled | TRUE | TRUE | TRUE | TRUE |
| max_dispatchers | 0 | 0 | 0 | 0 |
| max_shared_servers | 0 | 0 | 0 | 0 |
| disk_asynch_io | NATIVE | FALSE | FALSE | FALSE |
| db_securefile | TRUE | ALWAYS | ALWAYS | ALWAYS |
| plsql_code_type | NATIVE | NATIVE | NATIVE | NATIVE |
| _active_session_legacy_behavior | TRUE | TRUE | TRUE | TRUE |

**Table 11-4 Additional Requirements for OIG Deployments**

| Parameter | Development | Small System | Medium System | Large System |
|---|---|---|---|---|
| sessions | 5000 | 5000 | 5000 | 5000 |
| open_cursors | 3000 | 3000 | 3000 | 3000 |
| processors | 5000 | 5000 | 5000 | 5000 |
| aq_tm_processes | 10 | 10 | 10 | 10 |
| open_links | 20 | 20 | 20 | 20 |
| nls_sort | BINARY | BINARY | BINARY | BINARY |
| shared_servers | 0 | 0 | 0 | 0 |

Oracle recommends you to set these parameters in the database configuration assistant when creating the database. If not, you can adjust them after creating the database, by using the `alter system` database command. For example:

```
sqlplus / as sysdba
alter system set aq_tm_processes=1 scope=spfile;
```

For `_parameters`, use the syntax:

```
alter system set "_active_session_legacy_behavior"=true scope=spfile;
```

After making changes in the `spfile`, restart the database. For example

```
srvctl stop database -d iamdb
srvctl start database -d iamdb
```

> **Note:**
>
> For guidelines on setting up optimum parameters for the database, see Tuning Database Parameters in *Tuning Performance*.

# Adding Database Options

Oracle Identity Management requires that Oracle JVM and Oracle Text is available in the database. By default, these options are not available in the database created.

To add these options later, run the following command from db node 1:

```
dbca -silent -configureDatabase -sourceDB iamdb_iad18h -addDBOption
JSERVER,ORACLE_TEXT
```

Where the database name is the name of the database you created earlier. See Creating the Database.

# Adding XA Views

Oracle Identity Governance requires the use of XA views.

To install these into the database, run the following command:

```
sqlplus / as sysdba @$ORACLE_HOME/rdbms/admin/xaview.sql
```

# Applying the Database Patches

Ensure that you apply all the necessary database patches. The patches are required only if you are using Oracle Identity Governance.

Include the database patches from Oracle Text Mandatory Patches.

Also, ensure that you include the latest patch set bundles for the release of database that you are using.

# Creating a PDB Using an Existing PDB as a Template

When you create a database, an empty PDB is created at the same time. This example used the empty PDB to create a secondary PDB with the same configuration.

This example shows the commands used to create a secondary PDB by using the empty PDB. The new database is called OIG and the existing one is called OAM:

```
sqlplus / as sysdba
```

> **Note:**
>
> You may need to set *ORACLE_HOME* and *ORACLE_SID* to the database you just created to ensure that you are connecting to the correct database, especially if you have more than one database in the compartment. Creating a Database.

```
create pluggable database IGDPDB from IADPDB keystore identified by
syspassword;
```

> **Note:**
>
> '`syspassword`' is the database sys password.

```
Pluggable database created.
```

```
show pdbs;
```

```
CON_ID     CON_NAME      OPEN MODE      RESTRICTED
-------    ----------    ----------     -----------
 2          PDB$SEED      READ ONLY      NO
 3          IADPDB        READ WRITE     NO
 5          IGDPDB        MOUNTED
```

```
alter pluggable database IGDPDB open read write;
```

```
Pluggable database altered.
```

```
show pdbs;
```

```
CON_ID     CON_NAME        OPEN MODE      RESTRICTED
```

```
-------   ----------      ----------     ----------
  2        PDB$SEED        READ ONLY      NO
  3        IADPDB          READ WRITE     NO
  5        IGDPDB          READ WRITE     NO
```

# Creating Database Services

When multiple Oracle Fusion Middleware products are sharing the same database, each product should be configured to connect to a separate, dedicated database service. This service should be different from the default database service.

A different service name enables you to create role-based database services for Disaster Recovery and Multi-Datacenter topologies.

> **Note:**
>
> The instructions in this section are for the Oracle Database 12*c* (12.1) release. If you are using another supported database, refer to the appropriate documentation library for more up-to-date and release-specific information.

Beginning with Data Guard 11*g* Release 2, you can automatically control the startup of database services on primary and standby database by assigning a database role to each service. This service is in addition to the default service created when the database was commissioned. A role based database service will automatically start upon database startup if the management policy of the service is AUTOMATIC and if one of the roles assigned to that service matches the current role of the database; for example, if the database is running as a primary.

Creating a database service in this way means that, the service is started whenever the database with the role `primary` is started. The service will move between sites as the underlying databases roles are moved through switchover or failover.

If you are planning to use a standard disaster recovery solution as described in Disaster Recovery Guide, then each database service should be defined as a Role Based Database service.

If you are planning on using a multi-datacenter deployment, then the database service created for the Oracle Identity Governance (IGDDB) database should be a role based service.

For more information about connecting to Oracle databases using services, see Overview of Using Dynamic Database Services to Connect to Oracle Databases in *Real Application Clusters Administration and Deployment Guide*.

In addition, the database service should be different from the default database service. For complete instructions on creating and managing database services for an Oracle Database 12*c* database, see Overview of Automatic Workload Management with Dynamic Database Services in *Real Application Clusters Administration and Deployment Guide*.

Runtime connection load balancing requires configuring Oracle RAC Load Balancing Advisory with service-level goals for each service for which load balancing is enabled.

You can configure the Oracle RAC Load Balancing Advisory for `SERVICE_TIME` or `THROUGHPUT`. Set the connection load-balancing goal to **SHORT**.

You create and modify Oracle Database services by using the `srvctl` utility.

To create and modify a database service:

1. Add the service to the database and assign it to the instances by using `srvctl`:

   ```
   srvctl add service -db iamdb1 -service iadedg.example.com -preferred iamdb11, iamdb12
   ```

   If you use PDB, change the command to the following:

   ```
   srvctl add service -db iamdb1 -pdb iadpdb -service iadedg.example.com -
   preferred iamdb11, iamdb12
   ```

   > **Note:**
   >
   > For the Service Name of the Oracle RAC database, use lowercase letters, followed by the domain name. For example: `iadedg.example.com`.

2. If you want to use Oracle Data Guard for disaster protection, then you need to ensure that the database service is started only when the database has a primary role. To do this step, modify the service you created in **Step 1** using the following commands:

   ```
   srvctl add service -db iamdb1 -service iadedg.example.com -preferred
   iabdb11,iamdb12 -pdb iadpdb -role "PRIMARY,SNAPSHOT_STANDBY"
   ```

   ```
   srvctl modify service -db iamdb1 -service iadedg.example.com -rlbgoal
   SERVICE_TIME -clbgoal SHORT
   ```

   Verify that the service is setup correctly using the command:

   ```
   srvctl config service -db iamdb1 -service iadedg.example.com
   ```

3. Start the service:

   ```
   srvctl start service -db iamdb1 -service iadedg.example.com
   ```

   > **Note:**
   >
   > For complete instructions on creating and managing database services with SRVCTL, see Creating Services with SRVCTL in the *Real Application Clusters Administration and Deployment Guide*.

4. Modify the service so that it uses the Load Balancing Advisory and the appropriate service-level goals for runtime connection load balancing.

   Use the following resources in the Oracle Database 12*c Real Application Clusters Administration and Deployment Guide* to set the SERVICE_TIME and THROUGHPUT service-level goals:

   - Overview of the Load Balancing Advisory
   - Configuring Your Environment to Use the Load Balancing Advisory

   For example:

Check the default configuration of the service by using this command:

```
srvctl config service -db iamdb1 -service iadedg.example.com
```

Several parameters are shown. Check the following parameters:

- Connection Load Balancing Goal: Long
- Runtime Load Balancing Goal: NONE

You can modify these parameters by using the following command:

```
srvctl modify service -db iamdb1 -service iadedg.example.com -rlbgoal
SERVICE_TIME -clbgoal SHORT
```

5. Restart the service:

```
srvctl stop service -db iamdb1 -service iadedg.example.com
srvctl start service -db iamdb1 -service iadedg.example.com
```

6. Verify the change in the configuration:

```
srvctl config service -db iamdb1 -service iadedg.example.com


Runtime Load Balancing Goal: SERVICE_TIME
  Service name: iadedg.example.com
  Service is enabled
  Server pool: iamdb1_iadedg.example.com
  ...
  Connection Load Balancing Goal: SHORT
  Runtime Load Balancing Goal: SERVICE_TIME
  ...
```

# Preventing Password Timeouts for System Accounts

When Oracle Fusion Middleware products are installed several database schemas are created. These database schemas are associated with the Default password profile, which forces passwords to be changed at periodic intervals, otherwise they are locked out pending the password being changed and the account unlocked. This behaviour may be undesirable for system accounts, as the locking of a system account may result in interrupted service.

It is possible to create a password policy which ages out system passwords on a different schedule to regular database users. Creating a password policy which does not age system accounts should be used with care. It is still a good practice to ensure that system passwords are changed on a periodic basis.

### Creating a Password Policy for System Accounts

To create a password policy, run the following command in sqlplus:

```
CREATE PROFILE SYSTEM_PASSWORD_PROFILE LIMIT PASSWORD_LIFE_TIME <DAYS BEFORE
EXPIRY>
```

> **Note:**
>
> If creating the profile in a container database the profile name must be preceded by c## (this is not the case if created at the PDB level). For example:
>
> ```
> CREATE PROFILE C##SYSTEM_PASSWORD_PROFILE LIMIT PASSWORD_LIFE_TIME
> 360;
> ```

To prevent password from never expiring the following profile can be used.

```
CREATE PROFILE SYSTEM_PASSWORD_PROFILE LIMIT PASSWORD_LIFE_TIME UNLIMITED;
```

### Assigning Users to the System Password Policy

To assign a user to the System Password Profile issue the following command in sqlplus:

```
ALTER USER <username> SYSTEM_PASSWORD_PROFILE;
```

# Using SecureFiles for Large Objects (LOBs) in an Oracle Database

SecureFiles is a new LOB storage architecture introduced in Oracle Database 11g Release 1. It is recommended to use SecureFiles for the Oracle Fusion Middleware schemas, in particular for the Oracle SOA Suite schemas.

Beginning with Oracle Database 11g Release 1, Oracle introduced SecureFiles, a new LOB storage architecture. Oracle recommends that you use SecureFiles for the Oracle Fusion Middleware schemas, in particular for the Oracle SOA Suite schemas. See Using Oracle SecureFiles LOBs in the *Oracle Database SecureFiles and Large Objects Developer's Guide*.

In Oracle 12*c* Databases, the default setting for using SecureFiles is `PREFERRED` . This means that the database attempts to create a SecureFiles LOB unless a BasicFiles LOB is explicitly specified for the LOB or the parent LOB (if the LOB is in a partition or sub-partition). The Oracle Fusion Middleware schemas do not explicitly specify BasicFiles, which means that Oracle Fusion Middleware LOBs defaults to SecureFiles when installed in an Oracle 12*c* database.

For Oracle 11*g* databases, the `db_securefile` system parameter controls the SecureFiles usage policy. This parameter can be modified dynamically. The following options can be used for using SecureFiles:

- `PERMITTED`: Allows SecureFiles to be created (This is the default setting for `db_securefile`. The default storage method uses BasicFiles).
- `FORCE`: Creates all (new) LOBs as SecureFiles.
- `ALWAYS`: Tries to create LOBs as SecureFiles, but falls back to BasicFiles if not possible (if ASSM is disabled).

Other values for the `db_securefile` parameter are:

- `IGNORE`: Ignore attempts to create SecureFiles.

- `NEVER`: Disallow new SecureFiles creations.

For Oracle 11*g* Databases, Oracle recommends that you set the `db_securefile` parameter to `FORCE` before you create the Oracle Fusion Middleware schemas with the Repository Creation Utility (RCU).

Note that the SecureFiles segments require tablespaces managed with automatic segment space management (ASSM). This means that LOB creation on SecureFiles will fail if ASSM is not enabled. However, the Oracle Fusion Middleware tablespaces are created by default with ASSM enabled. As a result, with the default configuration, nothing needs to be changed to enable SecureFiles for the Oracle Fusion Middleware schemas.

# About Database Backup Strategies

Performing a database backup at key points in the installation and configuration of an enterprise deployment enables you to recover quickly from any issue that might occur in the later configuration steps.

At key points in the installation and configuration of an enterprise deployment, this guide recommends that you back up your current environment. For example, after you install the product software and create the schemas for a particular Oracle Fusion Middleware product, you should perform a database backup. Performing a backup allows you to perform a quick recovery from any issue that might occur in the later configuration steps.

You can choose to use your own backup strategy for the database, or you can simply make a backup by using operating system tools or RMAN for this purpose.

Oracle recommends that you use Oracle Recovery Manager for the database, particularly if the database was created using Oracle Automatic Storage Management. If possible, you can also perform a cold backup by using operating system tools such as tar.

# Part III
# Configuring the Enterprise Deployment

Configuring the enterprise deployment includes a series of procedures. Configuring LDAP, installing the WebLogic Kubernetes Operator, creating infrastructure for Oracle Access Management (OAM) and Oracle Identity Governance (OIG), and configuring Oracle HTTP Server, OAM, and OIG are the required procedures.

This part of the guide contains the following chapters:

- Installing the Monitoring and Visualization Software
- Installing and Configuring Ingress Controller
  An Ingress controller is a load balancer that enables simple host or URL-based HTTO routing.
- Installing and Configuring Oracle Unified Directory
  Create a new, highly available Oracle Unified Directory (OUD) deployment inside a Kubernetes cluster.
- Installing and Configuring Oracle Unified Directory Services Manager
  Oracle Unified Directory Service Manager (OUDSM) is a Graphical User Interface (GUI) tool that is used to manage Oracle Unified Directory. It is not mandatory to install OUDSM in the production environments. However, OUDSM makes managing Oracle Unified Directory easier.
- Installing and Configuring WebLogic Kubernetes Operator
  The WebLogic Operator for Kubernetes facilitates the creation and management of WebLogic domains in a Kubernetes cluster.
- Installing and Configuring Oracle HTTP Server
  For an enterprise deployment, Oracle HTTP Server must be installed on each of the web tier hosts and configured as Oracle HTTP standalone domains on each host.
- Configuring Oracle Access Manager Using WDT
  Install and configure an initial domain, which can be used as the starting point for an enterprise deployment. Later, configure the domain.
- Configuring Oracle Identity Governance Using WDT
  Install and configure an initial domain to use as the starting point for an enterprise deployment. Later, configure this domain.
- Installing and Configuring Oracle Identity Role Intelligence
  Oracle Identity Role Intelligence (OIRI) authenticates using users and groups defined in Oracle Identity Governance. Therefore, you have to install and configure Oracle Identity Governance first.
- Installing and Configuring Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator
  Oracle Advanced Authentication and Risk Management (OAA and OARM) authenticates users by using multi-factor authentication. It integrates with Oracle Access Manager for OAuth authentication.
- Configuring Disaster Recovery

12

# Installing the Monitoring and Visualization Software

If you want to deploy your own Monitoring and Visualization software on the Kubernetes cluster, you can use the steps described in this chapter.

Production best practices for Elasticsearch and Prometheus deployments are outside the scope of this document. For detailed installation instructions, see:

- Installing Elasticsearch (ELK) Stack and Kibana
- Installing Prometheus and Grafana

This chapter includes the following topics:

- Installing the Logging and Visualization Software
- Installing the Monitoring Software

## Installing the Logging and Visualization Software

Elasticsearch enables you to aggregate logs from various products on your system. You can analize the logs and use the Kibana console to visualize the data in the form of charts and graphs. Elasticsearch recommends the best practice of using the API keys for connecting to a centralized Elasticsearch deployment. For simplicity, this document illustrates the process using user names and passwords. For complete information on these products, see the manufacturer documentation at https://www.elastic.co/.
This section includes the following topics:

- Kubernetes Services
  The Kubernetes services are created as part of the Monitoring and Visualization deployment process.
- Variables Used in this Section
  This section provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.
- Prerequisites
  The latest releases of Elasticsearch uses the Elasticsearch Operator. The Elasticsearch Operator deploys an elastic cluster using the Kubernetes stateful sets. Stateful sets create dynamic persistent volumes.
- Installing Elasticsearch (ELK) Stack and Kibana

# Kubernetes Services

The Kubernetes services are created as part of the Monitoring and Visualization deployment process.

**Table 12-1 Kubernetes Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| `elasticsearch` | ClusterIP | | 9600 |
| `kibana-nodeport` | NodePort | 31800 | 6501 |
| `elk-nodeport` | NodePort | 31920 | 9200 |

> **Note:**
>
> The mapped port is randomly assigned at install time. The values provided in this table are examples only.

# Variables Used in this Section

This section provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 12-2 List of Variables**

| Variable | Sample Value | Description |
|---|---|---|
| <*ELKNS*> | `elkns` | The name of the Elasticsearch namespace. |
| <*ELK_OPER_VER*> | `2.10.0` | The version of the Elasticsearch Operator. |
| <*ELK_VER*> | `8.11.0` | The version of Elasticsearch/ Kibana you want to install. |
| <*ELK_USER*> | `logstash_internal` | The name of the user for logstash to access Elasticsearch. |
| <*ELK_PASSWORD*> | <*password*> | The password for ELK_USER. |
| <*ELK_K8*> | `31920` | The Kubernetes port used to access Elasticsearch externally. |
| <*ELK_KIBANA_K8*> | `31800` | The Kubernetes port used to access Kibana externally. |
| <*DH_USER*> | *username* | The user name for Docker Hub. |
| <*DH_PWD*> | *mypassword* | The password for Docker Hub. |

**Table 12-2 (Cont.) List of Variables**

| Variable | Sample Value | Description |
|---|---|---|
| *<PV_SERVER>* | `mynfsserver.example.com` | The name of the NFS server.<br>**Note**: This name should be resolvable inside the Kubernetes cluster. |
| *<ELK_SHARE>* | `/export/IAMPVS/elkpv` | The NFS mount point for the ELK persistent volumes. |

# Prerequisites

The latest releases of Elasticsearch uses the Elasticsearch Operator. The Elasticsearch Operator deploys an elastic cluster using the Kubernetes stateful sets. Stateful sets create dynamic persistent volumes.

Before installing Elasticsearch, you should ensure that you have a default Kubernetes storage class defined for your environment that allows dynamic storage. Each vendor has its own storage provider but it may not be configured to provide dynamic storage allocation.

To determine what your default storage class is, use the following command:

```
kubectl get storageclass


NAME            PROVISIONER                           RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
oci (default)   kubernetes.io/is-default-class        Delete
Immediate           false                  6d21h
```

If you do not have a storage provider that allows you to dynamically create storage, you can use an external NFS storage provider such as NFS subdir external provisioner.

For more information on storage classes, see Storage Classes.

For more information on NFS subdir external provisioner, see Kubernetes NFS Subdir External Provisioner.

For completeness, the following steps show how to install Elasticsearch and Kibana by using NFS subdir:

- Creating a Filesystem for the ELK Data
- Installing NFS Subdir External Provisioner

## Creating a Filesystem for the ELK Data

Before you can deploy the NFS client, you need to create a mount point/export on your NFS storage for storing your ELK data. This mount point is used by the NFS subdir external provider.

## Installing NFS Subdir External Provisioner

1. To install the NFS subdir external provisioner, use the following commands:

```
helm repo add nfs-subdir-external-provisioner https://kubernetes-
sigs.github.io/nfs-subdir-external-provisioner/


helm install nfs-subdir-external-provisioner nfs-subdir-external-
provisioner/nfs-subdir-external-provisioner \
    --set nfs.server=<PVSERVER>\
    --set nfs.path=<ELK_SHARE>
```

2. Issue the following command:

```
kubectl get storageclass
```

   You should now see a new storage class called `nfs-client`.

# Installing Elasticsearch (ELK) Stack and Kibana

Each of the product chapters show how to send log files to a centralized Elasticsearch stack which includes the visualization console Kibana. The instructions help you deploy a simple ELK cluster. This is sufficient for testing. In production environments, you should obtain appropriate licenses from the vendor.
For setting up the production security, see Configure Security for the Elastic Stack.

This section includes the following topics:

- Setting Up a Product Specific Work Directory
  Before you begin the installation, you should have already downloaded and staged the ELK container image or should be using the Oracle Container Registry and the code repository.
- Creating a Kubernetes Namespace
- Creating a Kubernetes Secret for Docker Hub Images
- Installing the Elasticsearch Operator
- Creating an Elasticsearch Cluster
- Creating a Kibana Cluster
- Creating the Kubernetes Services
- Granting Access to Logstash
- Accessing the Kibana Console
- Creating a Kibana Index

## Setting Up a Product Specific Work Directory

Before you begin the installation, you should have already downloaded and staged the ELK container image or should be using the Oracle Container Registry and the code repository.

See Identifying and Obtaining Software Distributions for an Enterprise Deployment. This section describes the procedure to copy the downloaded sample deployment scripts to a temporary working directory for ELK.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubernetes cluster.

   ```
   mkdir <WORKDIR>
   ```

   For example:

   ```
   mkdir /workdir/ELK
   ```

2. Change the directory to this location:

   ```
   cd /workdir/ELK
   ```

> **Note:**
>
> The same set of sample files are used by several products in this guide. To avoid having to download them each time, the files are staged in a non-product specific working directory.

## Creating a Kubernetes Namespace

The Kubernetes namespace is used to store the Elasticsearch stack.

Use the following command to create the namespace for ELK:

```
kubectl create namespace <ELKNS>
```

For example:

```
kubectl create namespace elkns
```

## Creating a Kubernetes Secret for Docker Hub Images

This secret allows Kubernetes to pull an image from `hub.docker.com` which contains the Elasticsearch images.

You should have an account on `hub.docker.com`.

Use the following command to create a Kubernetes secret for `hub.docker.com`:

```
kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="<DH_USER>" --docker-
password="<DH_PWD>" --namespace=<ELKNS>
```

For example:

```
kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="username" --docker-
password="mypassword" --namespace=elkns
```

If you are pulling the images from your own registry, then create a secret for your registry using the following command:

```
kubectl create secret -n <ELKNS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n elkns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

## Installing the Elasticsearch Operator

To install the Elasticsearch Operator, perform the following steps:

1. Use the command:

```
helm repo add elastic https://helm.elastic.co


helm repo update


helm install elastic-operator elastic/eck-operator -n <ELKNS> --set
image.tag=<ELK_OPER_VER>
```

> **Note:**
>
> If you are pulling the images from your own repository, the command will be as follows:
>
> ```
> helm install elastic-operator elastic/eck-operator -n <ELKNS> --
> set image.repository=container-registry.oracle.com/eck/eck-
> operator --set imagePullSecrets[0].name=regcred
> ```

For example:

```
helm install elastic-operator elastic/eck-operator -n elkns --set
image.tag=2.11.0
```

2. Verify the installation by using the command:

```
kubectl get all -n elkns
```

## Creating an Elasticsearch Cluster

To create an Elasticsearch cluster using the Elasticsearch Operator, perform the following steps:

- Creating a Configuration File
- Creating the Elasticsearch Cluster
- Copying the Elasticsearch Certificate
- Elasticsearch Access Details

### Creating a Configuration File

To create a configuration file, complete the following steps:

1. Create a configuration file called `<WORKDIR>/ELK/elk_cluster.yaml` with the following contents:

```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: elasticsearch
  namespace: <ELKNS>
spec:
  version: <ELK_VER>
  nodeSets:
  - name: default
    count: 1
    config:
      node.store.allow_mmap: false
    volumeClaimTemplates:
    - metadata:
        name: elasticsearch-data # Do not change this name unless you set
up a volume mount for the data path.
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 5Gi
        storageClassName: nfs-client
```

> **Note:**
>
> If you are using your own container registry, you should add the following lines above:
>
> ```
> spec:
>   image: iad.ocir.io/mytenancy/idm/elasticsearch/
> elasticsearch:<ELK_VER>
>
>     podTemplate:
>       spec:
>         imagePullSecrets:
>         - name: regcred
> ```

2. Ensure that the `storageClassName` is the name of the storage class you want to use. For example, `oci` or `nfs-client`.

## Creating the Elasticsearch Cluster

To create the Elasticsearch cluster, complete the following steps

1. Run the following command:

   ```
   kubectl  create -f <WORKDIR>/ELK/elk_cluster.yaml
   ```

2. Monitor the creation of the pods by using the following commands:

   ```
   kubectl get all -n elkns -o wide
   ```

   ```
   kubectl get elasticsearch -n elkns
   ```

## Copying the Elasticsearch Certificate

Logstash requires access to the Elasticsearch CA (Certificate Authority) certificate to connect to the Elasticsearch server. Logstash places a copy of the certificate into config maps loaded into each namespace from where logstash runs.

In a production environment, it is recommended that you use production certificates. However, if you have allowed Elasticsearch to create its own self-signed certificates, you should copy this certificate to your work directory for easy access later.

Copy the self-signed certificates to your work directory by using the following command:

```
kubectl cp <ELKNS>/elasticsearch-es-default-0:/usr/share/elasticsearch/config/
http-certs/..data/ca.crt <WORKDIR>/ELK/elk.crt
```

For example:

```
kubectl cp elkns/elasticsearch-es-default-0:/usr/share/elasticsearch/config/
http-certs/..data/ca.crt /workdir/ELK/elk.crt
```

### Elasticsearch Access Details

After the cluster starts, you will need the following information to interact with it:

- Credentials
- URL

### Credentials

When the cluster gets created, it creates a user called `elastic`. You can obtain the password for the user `elastic` by using the command:

```
kubectl get secret elasticsearch-es-elastic-user -n <ELKNS> -o go-
template='{{.data.elastic | base64decode}}'
```

For example:

```
kubectl get secret elasticsearch-es-elastic-user -n elkns -o go-
template='{{.data.elastic | base64decode}}'
```

### URL

The URL for sending logs can be determined using the command:

```
https://elasticsearch-es-http.<ELKNS>.svc.cluster.local:9200/
```

For example:

```
https://elasticsearch-es-http.elkns.svc.cluster.local:9200/
```

## Creating a Kibana Cluster

To create an Kibana cluster, perform the following steps:

- Creating a Configuration File
- Deploying Kibana

### Creating a Configuration File

Create a configuration file called `<WORKDIR>/ELK/kibana.yaml` with the following contents:

```
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: kibana
  namespace: <ELKNS>
spec:
  version: <ELK_VER>
  count: 1
  elasticsearchRef:
    name: elasticsearch
```

> **Note:**
>
> If you are using your own container registry, you should add the following lines above:
>
> ```
> spec:
>   image: iad.ocir.io/mytenancy/idm/kibana/kibana:<ELK_VER>
>
>   podTemplate:
>     spec:
>       imagePullSecrets:
>       - name: regcred
> ```

### Deploying Kibana

1. Use the following command to deploy Kibana:

   ```
   kubectl  create -f <WORKDIR>/ELK/kibana.yaml
   ```

2. Monitor the creation of the pods by using the commands:

   ```
   kubectl get all -n elkns -o wide
   ```

   ```
   kubectl get kibana -n elkns
   ```

## Creating the Kubernetes Services

If you are using an Ingress controller, it is possible to expose these services through Ingress. However, if you want to send the Ingress log files to Elasticsearch, it makes sense not to make them dependent on each other.

You should create two NodePort Services to access Elasticsearch and Kibana.

- A NodePort Service for the external Elasticsearch interactions. For example, to send logs from sources outside the cluster and to make API calls to the ELK cluster.
- Another, to access the Kibana console.
- Creating a NodePort Service for Kibana
- Creating a NodePort Service for Elasticsearch

### Creating a NodePort Service for Kibana

To enable access to the Kibana console from a browser, you must expose it outside of the cluster. You can do this by creating a NodePort Service:

1. Create a file called `<WORKDIR>/ELK/kibana_nodeport.yaml` with the following contents:

   ```
   kind: Service
   apiVersion: v1
   metadata:
   ```

```
  name: kibana-nodeport
  namespace: <ELKNS>
spec:
  type: NodePort
  selector:
    common.k8s.elastic.co/type: kibana
    kibana.k8s.elastic.co/name: kibana
  ports:
    - targetPort: 5601
      port: 5601
      nodePort: <ELK_KIBANA_K8>
      protocol: TCP
```

2. Create the NodePort Service by using the following command:

```
kubectl create -f <WORKDIR>/ELK/kibana_nodeport.yaml
```

For example:

```
kubectl create -f </workdir/ELK/kibana_nodeport.yaml
```

## Creating a NodePort Service for Elasticsearch

To enable access to the Elasticsearch stack from outside the cluster, you must expose it outside of the cluster. You can do this by creating a NodePort Service.

1. Create a file called `<WORKDIR>/ELK/elk_nodeport.yaml` with the following contents:

```
kind: Service
apiVersion: v1
metadata:
  name: elk-nodeport
  namespace: <ELKNS>
spec:
  type: NodePort
  selector:
    common.k8s.elastic.co/type: elasticsearch
    elasticsearch.k8s.elastic.co/cluster-name: elasticsearch
  ports:
    - targetPort: 9200
      port: 9200
      nodePort: <ELK_K8>
      protocol: TCP
```

2. Create the NodePort Service using the following command:

```
kubectl create -f <WORKDIR>/ELK/elk_nodeport.yaml
```

For example:

```
kubectl create -f </workdir/ELK/elk_nodeport.yaml
```

# Granting Access to Logstash

In a production deployment, you would have enabled Elasticsearch security. Security has two parts - SSL communication between Logstash and Elasticsearch, and an API key or a user name and password combination to gain access.

- Creating a Role and a User for Logstash
- Creating an API Key for Logstash

## Creating a Role and a User for Logstash

In a production deployment, you would have enabled Elasticsearch security. Security has two parts - SSL communication between Logstash and Elasticsearch and a user/password combination to gain access. You can create user names and roles through the command-line API or the Kibana console. The instructions provided below are for the command line.

> **Note:**
>
> Elasticsearch recommends the use of API keys instead of user names and passwords. For more information, see https://www.elastic.co.

Oracle recommends that you create a dedicated role and user for this purpose.

The following commands require that the Elasticsearch user '`elastic`' be encoded. To encode user `elastic`, use the following command:

```
echo -n elastic:<ELASTIC PASSWORD> | base64
```

To obtain the password for the `elastic` user, see Credentials.

1. To create a role with restricted privileges, use the following command:

```
curl -k  -X  POST "https://k8worker1.example.com:31920/_security/role/
logstash_writer" -H "Authorization: Basic <ENCODED USER>" -H 'Content-
Type: application/json' -d'
{
  "cluster": ["manage_index_templates", "monitor", "manage_ilm"],
  "indices": [
    {
      "names":
["oiglogs*","oamlogs*","oudlogs*","oudsmlogs*","oirilogs*","oaalogs*],
      "privileges": ["write","create","create_index","manage","manage_ilm"]
    }
  ]
}'
```

2. To create a user with the above role, use the following command:

```
curl -k  -X  POST "https://k8worker1.example.com:31920/_security/user/
<ELK_USER>" -H "Authorization: Basic <ENCODED USER>" -H 'Content-Type:
application/json' -d'
{
```

```
  "password" : "<ELK_PASSWORD>",
  "roles" : [ "logstash_writer"],
  "full_name" : "Internal Logstash User"
}'
```

## Creating an API Key for Logstash

Elastic search recommends that you use an API key to access Elasticsearch. To create an API Key using the command line option:

1. Run the following command:

```
curl -XPOST -u 'elastic:<ELK_PWD>' -k "https://
K8WORKER1.example.com:<ELK_K8>/_security/api_key" -H "kbn-xsrf: reporting"
-H "Content-Type: application/json" -d'
{
  "name": "logstash_host001",
  "role_descriptors": {
    "logstash_writer": {
      "cluster": ["monitor", "manage_ilm",
"manage_index_templates","read_ilm"],
      "index": [
        {
          "names": ["logs*","oiglogs*","oamlogs*","oudlogs*","oudsmlogs*"],
          "privileges":
["write","create","create_index","manage","manage_ilm"]
        }
      ]
    }
  }
}'
```

For example:

```
curl -XPOST -u 'elastic:3k2KuO7eNxI2ZeB350H71VW2' -k "https://
k8worker1.example.com:31920/_security/api_key" -H "kbn-xsrf: reporting" -H
"Content-Type: application/json" -d'
{
  "name": "logstash_host001",
  "role_descriptors": {
    "logstash_writer": {
      "cluster": ["monitor", "manage_ilm",
"manage_index_templates","read_ilm"],
      "index": [
        {
          "names": ["logs*","oiglogs*","oamlogs*","oudlogs*","oudsmlogs*"],
          "privileges":
["write","create","create_index","manage","manage_ilm"]
        }
      ]
    }
  }
}'
```

The output will appear as follows:

```
{"id":"PyGUf4MBwDOoSXCOlS5W","name":"logstash_host001","api_key":"Cs1YDZCvTG6SviN0ejL
4-Q","encoded":"UHlHVWY0TUJ3RE9vU1hDT2xTNVc6Q3MxWURaQ3ZURzZTdmlOMGVqTDQtUQ=="}
```

2. Make a note of the API key. You should used this value when you configure Logstash.

## Accessing the Kibana Console

To access the Kibana console, use the following URL:

```
http://k8workers.example.com:30094/app/kibana
```

The default user is `elastic`. You can obtain the password using the following command:

```
kubectl get secret elasticsearch-es-elastic-user -n $ELKNS -o go-
template='{{.data.elastic | base64decode}}'
```

## Creating a Kibana Index

When you add log files to Elasticsearch, you will not be able to see them until you create an index pattern. You can perform this step only after Elasticsearch has received some logs.

1. Log in to the Kibana console.
2. Click **Stack Management**.
3. Click **Data Views** in the Kibana section.
4. Click **Create Data View**.
5. Enter the following information:
   - **Name**: `Logs*`
   - **Timestamp**: `@timestamp`
6. Click **Create Data View**.
7. Click **Discover** to view the log file entries.

# Installing the Monitoring Software

Prometheus and Grafana help monitor your environment. The instructions given in this chapter are for a simple deployment by using the `kube-prometheus` installer. See kube-prometheus. For more information and documentation on the Prometheus product, see Prometheus. Before starting the installation process, ensure that you use the version of Prometheus that is supported with your Kubernetes release. See the Prometheus/Kubernetes compatibility matrix.

This section includes the following topics:

- Kubernetes Services
- Variables Used in this Section
  This section provides instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.
- Installing Prometheus and Grafana
  Each of the product chapters show how to send monitoring data to Prometheus and Grafana. This section explains how to install the Prometheus and Grafana software.
- About Grafana Dashboards

## Kubernetes Services

The Kubernetes services created as part of the installation are:

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| `prometheus-nodeport` | NodePort | 32101 | 9090 |
| `grafana-nodeport` | NodePort | 32100 | 3000 |
| `alertmanager-nodeport` | NodePort | 32102 | 9093 |

## Variables Used in this Section

This section provides instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <VARIABLE_NAME>. The following table provides the values you should set for each of these variables.

**Table 12-3 List of Variables**

| Variable | Sample Value | Description |
|---|---|---|
| *<PROMNS>* | `monitoring` | The name of the Kubernetes namespace to use for the deployment. |
| *<IMAGE_VER>* | `8.3.0` | The version of Prometheus to install. |
| *<PROM_GRAF_K8>* | `30900` | The Kubernetes port used to access Grafana externally. |
| *<PROM _K8>* | `30901` | The Kubernetes port used to access Prometheus externally. |
| *<PROM_ALERT_K8>* | `30902` | The Kubernetes port used to access Alert Manager externally. |

## Installing Prometheus and Grafana

Each of the product chapters show how to send monitoring data to Prometheus and Grafana. This section explains how to install the Prometheus and Grafana software.

The installation process consists of the following steps:

- Setting Up a Product Specific Work Directory
- Downloading the Prometheus Installer
- Creating a Kubernetes Namespace
- Creating a Helm Override File
- Deploying Prometheus and Grafana
- Validating the Installation

## Setting Up a Product Specific Work Directory

This section describes the procedure to copy the downloaded sample deployment scripts to a temporary working directory for Prometheus.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubernetes cluster.

   ```
   mkdir <WORKDIR>
   ```

   For example:

   ```
   mkdir /workdir/PROM
   ```

2. Change the directory to this location:

   ```
   cd /workdir/PROM
   ```

> **Note:**
>
> The same set of sample files are used by several products in this guide. To avoid having to download them each time, the files are staged in a non-product specific working directory.

## Downloading the Prometheus Installer

The Prometheus installer is available on GitHub as a helm repository. To download the installer:

1. Change directory to the working directory:

   ```
   cd /workdir/PROM
   ```

2. Run the following command:

   ```
   helm repo add prometheus-community https://prometheus-community.github.io/
   helm-charts
   helm repo update
   ```

## Creating a Kubernetes Namespace

You have to create a namespace to contain all the objects for Prometheus and Grafana. To create a namespace, run the following command:

```
kubectl create namespace monitoring
```

The output appears as follows:

```
namespace/monitoring created
```

## Creating a Helm Override File

Create a helm override file called `<WORKDIR>/override_prom.yaml` to determine how the deployment is created. This file will have the following contents:

```
alertmanager:
  service:
    nodePort: <PROM_ALERT_K8>
    type: NodePort

prometheus:
  image:
    tag: <IMAGE_VER>
  service:
    nodePort: <PROM_K8>
    type: NodePort

grafana:
  image:
    tag: <IMAGE_VER>
  service:
    nodePort: <PROM_GRAF_K8>
    type: NodePort

  adminPassword: <PROM_ADMIN_PWD>
```

This example uses the NodePort services because Prometheus is capable of monitoring Ingress and you do not want issues associated with Ingress from preventing access to Prometheus. Therefore, NodePort is used to keep it standalone.

> **Note:**
>
> If you want to pull the images from a repository other than `docker.io`, add the following entries to the top of the file:
>
> ```
> global:
>   imageRegistry: <REPOSITORY>
> ```
>
> For example:
>
> ```
> global:
>   ImageRegistry: iad.ocir.io/mytenancy/idm
> ```

## Deploying Prometheus and Grafana

Use the override file created earlier to deploy the Prometheus application.

1. Change the directory to the working directory.

   ```
   cd <WORKDIR>
   ```

2. Run the following command:

   ```
   helm install -n <PROMNS> kube-prometheus prometheus-community/kube-
   prometheus-stack -f <WORKDIR>/override_prom.yaml
   ```

   This command creates the containers associated with the Prometheus and Grafana application.

> **Note:**
>
> If you are using your own repository, then in addition to the changes in the override file, you may need to add the following to your `helm` command:
>
> ```
> helm install -n <PROMNS> --set
> grafana.image.repository=<REPOSITORY>/grafana/grafana kube-
> prometheus prometheus-community/kube-prometheus-stack -f <WORKDIR>/
> override_prom.yaml
> ```

## Validating the Installation

To ensure that the application is deployed, and the installation is complete, run the following command:

```
kubectl get all -n monitoring
```

The output apears as follows:

```
NAME                                                     READY
STATUS    RESTARTS   AGE
pod/alertmanager-kube-prometheus-kube-prome-alertmanager-0   2/2
Running   0          15h
pod/kube-prometheus-grafana-95944596-kcd9k               3/3
Running   0          15h
pod/kube-prometheus-kube-prome-operator-84c5bc5876-klvrs     1/1
Running   0          15h
pod/kube-prometheus-kube-state-metrics-5f9b85478f-qtwnz      1/1
Running   0          15h
pod/kube-prometheus-prometheus-node-exporter-9h86g           1/1
Running   0          15h
pod/kube-prometheus-prometheus-node-exporter-gbkgb           1/1
Running   0          15h
pod/kube-prometheus-prometheus-node-exporter-l99sb           1/1
Running   0          15h
pod/kube-prometheus-prometheus-node-exporter-r7d77           1/1
Running   0          15h
pod/kube-prometheus-prometheus-node-exporter-rnq42           1/1
Running   0          15h
```

```
pod/prometheus-kube-prometheus-kube-prome-prometheus-0       2/2
Running   0          15h

NAME                                                  TYPE        CLUSTER-
IP       EXTERNAL-IP   PORT(S)                       AGE
service/alertmanager-operated                         ClusterIP
None             <none>        9093/TCP,9094/TCP,9094/UDP   15h
service/kube-prometheus-grafana                       NodePort
10.97.137.130    <none>        80:30900/TCP                 15h
service/kube-prometheus-kube-prome-alertmanager       NodePort
10.97.153.100    <none>        9093:30903/TCP               15h
service/kube-prometheus-kube-prome-operator           ClusterIP
10.108.174.205   <none>        443/TCP                      15h
service/kube-prometheus-kube-prome-prometheus         NodePort
10.110.156.35    <none>        9090:30901/TCP               15h
service/kube-prometheus-kube-state-metrics            ClusterIP
10.96.233.108    <none>        8080/TCP                     15h
service/kube-prometheus-prometheus-node-exporter      ClusterIP
10.107.188.115   <none>        9100/TCP                     15h
service/prometheus-operated                           ClusterIP
None             <none>        9090/TCP                     15h

NAME                                                    DESIRED   CURRENT
READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/kube-prometheus-prometheus-node-exporter   5         5
5       5            5           <none>          15h

NAME                                                  READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/kube-prometheus-grafana               1/1     1
1           15h
deployment.apps/kube-prometheus-kube-prome-operator   1/1     1
1           15h
deployment.apps/kube-prometheus-kube-state-metrics    1/1     1
1           15h

NAME                                                        DESIRED
CURRENT   READY   AGE
replicaset.apps/kube-prometheus-grafana-95944596            1
1         1       15h
replicaset.apps/kube-prometheus-kube-prome-operator-84c5bc5876   1
1         1       15h
replicaset.apps/kube-prometheus-kube-state-metrics-5f9b85478f    1
1         1       15h

NAME
READY   AGE
statefulset.apps/alertmanager-kube-prometheus-kube-prome-alertmanager
1/1     15h
statefulset.apps/prometheus-kube-prometheus-kube-prome-prometheus
1/1     15h
```

# About Grafana Dashboards

Grafana dashboards are used to visualize information from your targets. There are different types of dashboards for different products. You should install a dashboard to monitor your Kubernetes environment.

The following dashboards are relevant to an Oracle Identity Management deployment:

**Table 12-4 Dashboards Relevant to an Oracle Identity Management Deployment**

| Dashboard | Location | Description |
|---|---|---|
| Kubernetes | https://grafana.com/grafana/dashboards/10856 | Used to monitor the Kubernetes cluster. |
| Nginx | https://grafana.com/grafana/dashboards/9614-nginx-ingress-controller/ | Used to monitor the Ingress controller. |
| WebLogic | `<WORKDIR>/samples/monitoring-service/config/weblogic-server-dashboard-import.json` | Included in the Oracle download from GitHub.<br>Used to Monitor the WebLogic domain. |
| Apache | https://grafana.com/grafana/dashboards/3894-apache/ | Several Apache dashboards are available. This is an example. |
| Oracle Database | https://grafana.com/grafana/dashboards/3333-oracledb/ | A sample database dashboard. |

- Installing a Grafana Dashboard

## Installing a Grafana Dashboard

To install a dashboard:

1. Download the Kubernetes Dashboard JSON file from the Grafana website. For example: https://grafana.com/grafana/dashboards/10856.
2. Access the Grafana dashboard with the `http://<K8_WORKER1>:30900` URL and log in with `admin`/`<PROM_ADMIN_PWD>`. Change your password if prompted.
3. Click the search box at the top of the screen and select **Import New Dashboard**.
4. Either drag the JSON file you downloaded in Step 1 to the **Upload JSON File** box or click the box and browse to the file. Click **Import**.
5. When prompted, select the Prometheus data source. For example: **Prometheus**.
6. Click **Import**. The dashboard is displayed in the **Dashboards** panel.

13

# Installing and Configuring Ingress Controller

An Ingress controller is a load balancer that enables simple host or URL-based HTTO routing.

This chapter includes the following topics:

- Kubernetes Services
  The Kubernetes services are created as part of the Ingress deployment.
- Variables Used in this Chapter
  The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.
- Creating a Kubernetes Namespace
  The Kubernetes namespace is used to store the Ingress controller.
- Creating a Registry Secret
- Adding the Ingress Image to the Helm Repository
  Use the `helm` commands to add the nginx container image to the Helm repository.
- Installing an Ingress Controller to Support HTTPS/HTTP and LDAPS/LDAP
  When you create an Ingress controller, it will support HTTP and/or HTTPS traffic. If you terminate the SSL traffic behind a load balancer, there is no need to enable the HTTPS traffic.

## Kubernetes Services

The Kubernetes services are created as part of the Ingress deployment.

**Table 13-1 Kubernetes Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| `nginx-ingress-ingress-nginx-idmedg` | NodePort | 80<br>443 | 30777<br>30443 |

These are the ports you will use to interact with the controller.

## Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 13-2 The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<INGRESSNS>* | `ingressns` | The name of the Ingress namespace. |
| *<GIT_USER>* | `mygituser` | The name of the user to log in to GitHub. |
| *<GIT_TOKEN>* | `mytoken` | The git login token. |
| *<WORKDIR>* | `/workdir/INGRESS/` | The working directory for Ingress. |
| *<INGRESS_SERVICE_TYPE>* | `NodePort` | The type of Ingress service type to create. Options are `NodePort` and `LoadBalancer`. |
| *<INGRESS_NAME>* | `idmedg` | An arbitrary name for the controller. |
| *<INGRESS_REPLICAS>* | `2` | The number of copies of the Ingress controller you want to start. For highly available implementations, this value should be `2` or greater. |
| *<INGRESS_HTTP_K8>* | `30777` | The Kubernetes service port you want to use for HTTP communications. |
| *<INGRESS_HTTPS_K8>* | `30443` | The Kubernetes service port you want to use for HTTPS communications. |
| *<LDAPNS>* | `oudns` | The namespace where the LDAP directory is running. |
| *<LDAP_K8>* | `30389` | The Kubernetes port you want to use to expose the Ingress LDAP traffic. |
| *<LDAPS_K8>* | `30636` | The Kubernetes port you want to use to expose the Ingress LDAPS traffic. |
| *<OUD_POD_PREFIX>* | `edg` | The prefix used in the name of the service in the LDAP namespace to which you want to send traffic. |
| *<USE_PROM>* | `false` | Set to `true` if you want Ingress metrics to be sent to Prometheus. |

# Creating a Kubernetes Namespace

The Kubernetes namespace is used to store the Ingress controller.

Use the following command to create a namespace for Ingress:

```
kubectl create namespace <INGRESSNS>
```

For example:

```
kubectl create namespace ingressns
```

# Creating a Registry Secret

The nginx Ingress controller is obtained from the container registry on GitHub. To download the image on demand, you should create a registry secret with your GitHub credentials. See Logging in to GitHub.

To create a secret called `gitcred` in the Ingress namespace, use the following command:

```
kubectl create secret docker-registry gitcred -n <INGRESSNS> --docker-
server=ghcr.io --docker-username=<GIT_USER> --docker-password="<GIT_TOKEN>"
```

For example:

```
kubectl create secret docker-registry gitcred -n ingressns --docker-
server=ghcr.io --docker-username=mygituser --docker-password="mytoken"
```

# Adding the Ingress Image to the Helm Repository

Use the `helm` commands to add the nginx container image to the Helm repository.

The commands are:

```
helm repo add stable https://kubernetes.github.io/ingress-nginx
```

```
helm repo update
```

# Installing an Ingress Controller to Support HTTPS/HTTP and LDAPS/LDAP

When you create an Ingress controller, it will support HTTP and/or HTTPS traffic. If you terminate the SSL traffic behind a load balancer, there is no need to enable the HTTPS traffic.

If you are deploying only Oracle Identity Management Microservices and do not want to place them behind an existing Oracle HTTP server deployment, you would either configure Ingress for end-to-end SSL or terminate SSL at the Ingress.

If you are deploying an Oracle LDAP directory inside a Kubernetes cluster, you can optionally expose the LDAP and LDAPS traffic through Ingress. If you have no direct requirement to access the LDAP directory outside of the Kubernetes cluster, Oracle recommends that you do not enable this option.

Perform the following steps to create an Ingress controller:

- Creating a Self-Signed Certificate for SSL Requests
- Creating a Kubernetes Secret with the Certificate
- Creating the Helm Override File

- Creating the Ingress Controller
- Validating the Ingress Controller

## Creating a Self-Signed Certificate for SSL Requests

You may skip this step if you have purchased your own CA certificate, or if you are placing Ingress behind an Oracle HTTP Server.

If you want to use a self-signed certificate, run the following commands:

1. Create a text file called `ssl_cert_config.txt` with the following content:

```
# Copyright (c) 2022, Oracle Corporation and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at
https://oss.oracle.com/licenses/upl.
#
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no
[req_distinguished_name]
CN = *.example.com
[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[alt_names]
DNS.1 = example.com
```

   Where `example.com` is the domain from where your requests will originate. For example, if you have an URL similar to `http://iadadmin.example.com`, your domain will be `example.com`.

2. Create a certificate by using the Linux command `openssl`:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /workdir/
INGRESS/ingress.key -out /workdir/INGRESS/ingress.crt -config /workdir/
INGRESS/ssl_cert_config.txt -extensions v3_req
```

3. Validate the generated certificate using the command:

```
openssl x509 -noout -text /workdir/INGRESS/ingress.crt
```

## Creating a Kubernetes Secret with the Certificate

Load the certificate into a Kubernetes secret. If you have purchased your own certificate, you can use that instead.

To create the secret in Kubernetes, use the following command:

```
kubectl -n <INGRESSNS> create secret tls common-tls-cert --key /workdir/
ingress.key --cert /workdir/ingress.crt
```

# Creating the Helm Override File

Depending on what you want to expose on your Ingress controller, you will use a different override file. The following examples show:

- Creating an Ingress controller for HTTP and HTTPS terminated traffic.
- Creating an Ingress controller for HTTP, HTTPS, and LDAP traffic.

Create a file called `ingress_override.yaml` in your working directory. Depending on your deployment, the content of the file will be different.

This section includes the following topics:



## Types of Ingress Service

You can create two types of Ingress services.

If you create a `NodePort` service, then interactions with the Ingress controller are through the Kubernetes worker nodes and the associated Kubernetes service port assigned to the controller.

If you create a `LoadBalancer` service, then the Ingress controller is assigned to an IP address. Interactions with the Ingress controller are directly through this IP address.

Creating a `LoadBalancer` type service is dependent on the flavor of Kubernetes you are using. You should refer to your platform's documentation to create this service.

For example, you can use the following steps to create a `LoadBalancer` service on Oracle Cloud Infrastructure:

1. Using the example files (see Override File for HTTP and HTTPS and Override File for HTTP, HTTPS, LDAP, and LDAPS), make the following changes:
   `Set type: LoadBalancer`
2. Under the `LoadBalancer` type, add the following lines:

```
annotations:
      service.beta.kubernetes.io/oci-load-balancer-internal: "true"
      service.beta.kubernetes.io/oci-load-balancer-subnet1:
"ocid1.subnet.oc1.iad....kdi3ds.....vqri332zdrr3rm"
      service.beta.kubernetes.io/oci-load-balancer-shape-flex-min: "10"
      service.beta.kubernetes.io/oci-load-balancer-shape-flex-max: "10"
      service.beta.kubernetes.io/oci-load-balancer-shape: "flexible"
```

3. Set `oci-load-balancer-subnet1` to the OCID of the subnet in which the Kubernetes nodes reside.

## Override File for HTTP and HTTPS

The following is an example of the Helm file for HTTP and HTTPS terminated traffic:

```
imagePullSecrets:
 - name: gitcred
controller:
  name: <INGRESS_NAME>
  ingressClassResource:
    name: nginx
  config:
    use-forwarded-headers: true
    enable-underscores-in-headers: true
  wildcardTLS:
    secret: tls-cert
  replicaCount: <INGRESS_REPLICAS>
  service:
    type: <INGRESS_SERVICE_TYPE>
    nodePorts:
      http: <INGRESS_HTTP_K8>
      https: <INGRESS_HTTPS_K8>
  admissionWebhooks:
    enabled: false
  metrics:
    enabled: <USE_PROM>
    serviceMonitor:
      enabled: <USE_PROM>
```

## Override File for HTTP, HTTPS, LDAP, and LDAPS

The following is an example of the `Helm` file for HTTP and HTTPS terminated traffic with the addition of exposing the LDAP/LDAPS directory services:

```
imagePullSecrets:
 - name: gitcred
tcp:
  1389: <LDAPNS>/<OUD_POD_PREFIX>-oud-ds-rs-lbr-ldap:ldap
  1636: <LDAPNS>/<OUD_POD_PREFIX>-oud-ds-rs-lbr-ldap:ldaps
controller:
  name: <INGRESS_NAME>
  ingressClassResource:
    name: nginx
  config:
    use-forwarded-headers: true
    enable-underscores-in-headers: true
  wildcardTLS:
    secret: tls-cert
  replicaCount: <INGRESS_REPLICAS>
  service:
    type: <INGRESS_SERVICE_TYPE>
    nodePorts:
      http: <INGRESS_HTTP_K8>
      https: <INGRESS_HTTPS_K8>
      tcp:
```

```
        1389: <LDAP_K8>
        1636: <LDAPS_K8>
  admissionWebhooks:
    enabled: false
  metrics:
    enabled: <USE_PROM>
    serviceMonitor:
      enabled: <USE_PROM>
```

Where:

- `tls-cert` is the name of the SSL certificate you created earlier. See Creating a Self-Signed Certificate for SSL Requests.
- `edg-<OUD_POD_PREFIX>-ds-rs-lbr-ldap` is the name of the service in the LDAP namespace to which you want to send the traffic.

## Creating the Ingress Controller

To create the Ingress controller, you need to run the following `helm` command specifying the override file you created earlier:

```
helm install nginx-ingress -n ingressns \
      --values /workdir/INGRESS/ingress_override.yaml \
      stable/ingress-nginx
```

Where `nginx-ingress` is the Ingress namespace. For example: `ingressns`.

1. Enter the text of the first step here.
2. Enter the text of the second step here.

## Validating the Ingress Controller

To validate that the Ingress controller has been successfully created, use the following command:

```
kubectl get all,ingress -n <INGRESSNS>
```

For example:

```
kubectl get all,ingress -n ingressns
```

The output appears as follows:

```
NAME                                                              READY
STATUS      RESTARTS   AGE
pod/nginx-ingress-ingress-nginx-idmedg-bd4fdc996-79415       1/1
Running     0          28h
pod/nginx-ingress-ingress-nginx-idmedg-bd4fdc996-qqvqf       1/1
Running     0          28h

NAME                                          TYPE       CLUSTER-IP
EXTERNAL-IP  PORT(S)                                                AGE
```

```
service/nginx-ingress-ingress-nginx-idmedg   NodePort   10.107.148.40
<none>       80:30777/TCP,443:30443/TCP,1389:31389/TCP,1636:31636/TCP 28h

NAME                                                    READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/nginx-ingress-ingress-nginx-idmedg   2/2     2
2           28h

NAME                                                              DESIRED
CURRENT   READY   AGE
replicaset.apps/nginx-ingress-ingress-nginx-idmedg-bd4fdc996  2
2         2       28h
```

14

# Installing and Configuring Oracle Unified Directory

Create a new, highly available Oracle Unified Directory (OUD) deployment inside a Kubernetes cluster.

This chapter includes the following topics:

- Configuring Oracle Unified Directory
  Oracle Unified Directory is an LDAP compliant directory which you can use as a standalone entity or with other Oracle Identity and Access Management components.
- Setting Up a Product Specific Work Directory
  Before you begin the installation, you should have already downloaded and staged the Oracle Unified Directory Service Manager container image or should be using the Oracle Container Registry and the code repository.
- About Deploying Oracle Unified Directory
  Oracle recommends you to use Helm to create and configure Oracle Unified Directory (OUD).
- Creating a Kubernetes Namespace
  You have to create a namespace to contain all the objects for Oracle Unified Directory.
- Creating a Container Registry Secret
  Oracle recommends that you use a container registry. If you use a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.
- Creating a Kubernetes Secret for Docker Hub Images
  This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash`. These commands are used by the OUD cron job to test for pods that are stuck in the `Terminating` state, and restart them if necessary.
- Creating Configuration Files
  Before beginning the deployment of OUD, you need to create a series of configuration files. These files are used to configure and seed the data required by OAM and OIG.
- Creating OUD Containers
  Create the server overrides file first and then use the `Helm` command to create the OUD containers.
- Creating External Access to OUD
  By default, the OUD deployment gets created with all the components configured as ClusterIP services. This means that the Oracle Unified Directory components are visible only within the Kubernetes cluster.
- Centralized Monitoring Using Grafana and Prometheus
  There is no specific metric collection for OUD. However, you can monitor the OUD pods using the standard Kubernetes Dashboard in Kibana.
- Centralized Log File Monitoring Using Elasticsearch and Kibana

# Configuring Oracle Unified Directory

Oracle Unified Directory is an LDAP compliant directory which you can use as a standalone entity or with other Oracle Identity and Access Management components.

This chapter describes how you can install Oracle Unified Directory inside a Kubernetes cluster. The chapter also explains how you can seed the directory with data required by other Oracle Identity and Access Manager components.

After deploying OUD, if the Kubernetes node, where the OUD pod(s) is/are running, goes down after the pod eviction time-out, the pod(s) do not get evicted but move to a `Terminating` state. The pod(s) then remain in that state forever. To avoid this problem, a cron job is created during the OUD deployment process, which checks for any pods in the `Terminating` state, deletes them, and then starts the pod again. The cron job requires access to images on `hub.docker.com`. Therefore, you should create a Kubernetes secret to enable access to these images.

This job ensures that you always have the number of OUD instances running as you have specified in the `helm` configuration file.

> **Note:**
>
> - If you upgrade your container, you should recopy the `helm` charts to the persistent volume.
> - If you change your Helm/Kubernetes version, you should update the `helm` file with the revised versions.



## Sizing Guidelines

When deploying OUD, you can use the following information as reference for the initial system sizing. For more information about sizing, see Deep Dive into Oracle Unified Directory 12.2.1.4.0 Performance.

| System Size | Number of Users | Memory | JVM Heap Size (Min) | JVM Heap Size (Max) |
|---|---|---|---|---|
| Development | - | 2GB | 1GB | 2GB |
| Small | 5000 | 8GB | 4GB | 8GB |
| Medium | 50000 | 16GB | 8GB | 16GB |
| Large | 2 Million | 64GB | 16GB | 64GB |

## Kubernetes/Ingress Services

After you configure OUD, the following OUD services will be available on each worker node:

**Table 14-1 OUD Services on Each Worker Node**

| Service | Type | Service Port | Mapped Port |
|---|---|---|---|
| OUD LDAP | NodePort | 31389 | 1389 |
| OUD LDAPS | Node Port | 31636 | 636 |
| OUD ADMIN | NodePort | 30777 | 80 |

> **Note:**
>
> OUD randomly picks its own Kubernetes service port. The numbers given in this table are only examples.

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 14-2 The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| <*WORKDIR*> | `/workdir/OUD` | The location where you want to create the working directory for OUD. |
| <*REGISTRY_ADDRESS*> | `iad.ocir.io/<mytenancy>` | The location of the registry. If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/oud_cpu`. |
| <*REG_USER*> | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The user id you use to log in to the registry. If you are use the Oracle container registry, this value will be your Oracle single sign-on user name. |
| <*REG_PWD*> | <*password*> | The registry user password. |

**Table 14-2 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OUD_REPOSITORY>* | `oracle/oud`<br>`local/oracle/oud`<br>`container-registry.oracle.com/middleware/oud_cpu`<br>`<REGISTRY_ADDRESS>/oracle/oud` | The name of the OUD software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/oud`. If you are using OLCNE, the value will be `local/oracle/oud`.<br>If you are using the Oracle container registry, the value will be:`container-registry.oracle.com/middleware/oud_cpu`.<br>If you are using a container registry, the value will be the name of the registry with the product name: *<REGISTRY_ADDRESS>*`/oracle/oud`. |
| *<OUD_VER>* | `12.2.1.4-jdk8-ol7-220411.1613` or latest | The version of the image you want to use. This will be the version you have downloaded and staged either locally or in the container registry. |
| *<DH_USER>* | *username* | The Docker user name for `hub.docker.com`. Used for CronJob images. |
| *<DH_PWD>* | *mypassword* | The Docker password for `hub.docker.com`. Used for CronJob images. |
| *<OUDNS>* | `oudns` | The name of the OUD namespace. |
| *<PVSERVER>* | `mynfsserver.example.com` | The name of the NFS server.<br>**Note**: This name should be resolvable inside the Kubernetes cluster. |
| *<OUD_SHARE>* | `/exports/IAMPVS/oudpv` | The NFS mount point for the OUD persistent volume. |
| *<OUD_CONFIG_SHARE>* | `/exports/IAMPVS/oudconfigpv` | The NFS mount point for the OUD config persistent volume. |
| *<OUD_LOCAL_SHARE>* | `/nfs_volumes/oudconfigpv` | The local directory where *<OUD_CONFIG_SHARE>* is mounted. Used to hold seed files. |
| *<LDAP_SEARCHBASE>* | `dc=example,dc=com` | The directory tree for your organization. This is where all the data is stored. |
| *<LDAP_GROUP_SEARCHBASE>* | `cn=Groups,dc=example,dc=com` | The location in the directory where groups/roles are stored. |

**Table 14-2 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<LDAP_USER_SEARCHBASE>* | `cn=Users,dc=example,dc=com` | The location in the directory where names of users are stored. |
| *<LDAP_RESERVE_SEARCHBASE>* | `cn=Reserve,dc=example,dc=com` | Set this to the name of the reservation container in OIG. Kept for legacy reasons. |
| *<LDAP_ADMIN_USER>* | `cn=oudadmin` | The user name of the directory administrator. |
| *<LDAP_ADMIN_PWD>* | `password` | The password of the directory administrator. |
| *<LDAP_OAMLDAP_USER>* | `oamLDAP` | The name of a user that OAM will use to connect to the directory for validating logins. |
| *<LDAP_OIGLDAP_USER>* | `oimLDAP` | The name of a user that OIG will use to connect to the directory to manage users. |
| *<LDAP_OAMADMIN_USER>* | `oamadmin` | The name of the user you want to administer OAM. |
| *<LDAP_WLSADMIN_USER>* | `weblogic_iam` | The name of the user you want to use to administer the domain. |
| *<LDAP_XELSYSADM_USER>* | `xelsysadm` | The name of the user you want to create for administering OIG. |
| *<LDAP_USER_PWD>* | *<password>* | The password you want to assign to the user names you are creating. IDM on Kubernetes expects this to be the same for each account. You can change this to different values post deployment, if required. |
| *<LDAP_OIGADMIN_GRP>* | `OIMAdministrators` | The name of the group consisting of the names of the administrators of OIG. |
| *<LDAP_WLSADMIN_GRP>* | `WLSAdministrators` | Users assigned to this role will be able to log in to the WebLogic Administration Console and FMW Control. |
| *<LDAP_OAMADMIN_GRP>* | `OAMAdministrators` | Users assigned to this role will be able to log in to the OAM Administration Console and configure OAM. |
| *<LDAP_SYSTEMIDS>* | `systemids` | The name of a container where you want to store system ids. User names placed in this container are not subject to OIM reconciliation or password aging. This container is reserved for users such as *<LDAP_OAMLDAP_USER>* and *<LDAP_OIGLDAP_USER>*. |

**Table 14-2 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OUD_PWD_EXPIRY>* | `2024-01-02` | The date on which the password for the user accounts will expire. The date should be in the YYYY-MM-DD format. |
| *<OUD_LDAP_K8>* | `31389` | Port to use for OUD LDAP requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| *<OUD_LDAPS_K8>* | `31636` | Port to use for OUD LDAPS requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| *<OUD_PREFIX>* | `edg` | |
| *<OUD_REPLICAS>* | `1` | The number of OUD replica instances you want to create. |
| *<REGION>* | `example` | This is the top level region and is usually the first part of the search base. |
| *<HELM_VER>* | - | The version of Helm you are running. You can obtain it by using the command: `helm version --short`<br>Only the first three indices are required. For example: `3.5.4`. |
| *<KUBERNETES_VER>* | - | The version of Kubernetes you are running. You can obtain it by using the command: `kubectl version --short=true \| grep Server`.<br>Only the first three indices are required. For example: `1.20.6`. |
| *<ELK_HOST>* | `https://elasticsearch-es-http.elkns.svc:9200` | The host and port of the centralized Elasticsearch deployment. This host can be inside the Kubernetes cluster or external to it. This host is used only when Elasticsearch is used. |
| *<ELK_VER>* | `8.11.0` | The version of Elasticsearch you want to use. |
| *<ELK_USER_PWD>* | *<password>* | The password assigned to the ELK user. See Creating a Role and a User for Logstash. |

# Setting Up a Product Specific Work Directory

Before you begin the installation, you should have already downloaded and staged the Oracle Unified Directory Service Manager container image or should be using the Oracle Container Registry and the code repository.

See Identifying and Obtaining Software Distributions for an Enterprise Deployment. This section describes the procedure to copy the downloaded sample deployment scripts to a temporary working directory for OUD.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubernetes cluster.

   ```
   mkdir <WORKDIR>
   ```

   For example:

   ```
   mkdir /workdir/OUD
   ```

2. Change the directory to this location:

   ```
   cd /workdir/OUD
   ```

   > **Note:**
   >
   > The same set of sample files are used by several products in this guide. To avoid having to download them each time, the files are staged in a non-product specific working directory.

3. Copy the sample scripts to your work directory.

   ```
   cp -R <work_dir>/fmw-kubernetes/OracleUnifiedDirectory /<WORKDIR>/samples
   ```

   For example:

   ```
   cp -R /workdir/fmw-kubernetes/OracleUnifiedDirectory /workdir/OUD/samples
   ```

# About Deploying Oracle Unified Directory

Oracle recommends you to use Helm to create and configure Oracle Unified Directory (OUD).

To deploy OUD:

- Deploy several OUD servers and set up replication between those containers (there should be more than one container for high availability).
- Create NodePort or Ingress services if you require access to the OUD directory outside of the Kubernetes cluster.
- Configure OUD to support Oracle Identity and Access Management.

Traditionally, the process of configuring OUD to support Oracle Identity and Access Management has been through the use of the Oracle Identity and Access Management tool '`idmConfigTool`'. This tool is used after you have installed and configured OUD and deployed OAM or OIG. If you want to use the traditional method of preparing and seeding the directory, the option is available but is not discussed in this guide.

For information on using `idmConfigTool` to configure OUD, see Preparing an Existing LDAP Directory.

The traditional method of configuring OUD includes the following steps:

- Deploy OUD.
- Deploy OAM or OIG.
- Log in to the OAM/OIG container.
- Run the `idmConfigTool` commands. See Preparing an Existing LDAP Directory.

The method discussed in this guide is to create seed datafiles and to provide these as input into the OUD creation process.

# Creating a Kubernetes Namespace

You have to create a namespace to contain all the objects for Oracle Unified Directory.

To create a namespace, run the following command:

```
kubectl create namespace oudns
```

The output appears as follows:

```
namespace/oudns created
```

# Creating a Container Registry Secret

Oracle recommends that you use a container registry. If you use a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

If you have staged your container images locally, there is no need to perform this step.

To create a container registry secret, use the following command:

```
kubectl create secret -n <OUDNS> docker-registry regcred --docker-
server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n oudns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

# Creating a Kubernetes Secret for Docker Hub Images

This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash`. These commands are used by the OUD cron job to test for pods that are stuck in the `Terminating` state, and restart them if necessary.

> **Note:**
>
> If you are pulling the images from your own container registry, then this step is not required.

You should have an account on `hub.docker.com`. If you want to stage the images in your own repository, you can do so and modify the `helm` override file as appropriate.

To create a Kubernetes secret for `hub.docker.com`, use the following command:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="<DH_USER>" --docker-
password="<DH_PWD>" --namespace=<OUDNS>
```

For example:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="username" --docker-
password="<mypassword>" --namespace=oudns
```

# Creating Configuration Files

Before beginning the deployment of OUD, you need to create a series of configuration files. These files are used to configure and seed the data required by OAM and OIG.

The entries in the files are based on the standard Enterprise Deployment Guide naming conventions. If you want to use alternative names for these entries, edit the files per your organizational requirements.

- Creating the Schema Extensions File
- Creating the Seeding File
- Setting Passwords

## Creating the Schema Extensions File

This file is used to extend the OUD schema with Oracle Access Manager Object Classes. You can skip this section if you are not deploying Oracle Access Manager.

Create the `99-user.ldif` file with the contents as shown in the Sample of the Schema Extension File.

All information should remain the same. If you have no plans to use Oracle Access Manager, you do not require this file.

## Creating the Seeding File

This file is used to seed OUD with the names of Users and Groups required by Oracle Access Manager and Oracle Identity Governance.

Create a file called `base.ldif` with the contents as shown in the Sample of the Seeding File.

You should perform a global search and replace on this file to make it specific to your organization. To make things easier, the sample file has a number of variables inserted to help you identify the entries that need to be changed. Each variable is enclosed in '<>'. For the list of variables used, see Variables Used in this Chapter.

> **Note:**
>
> This file contains all the entries for Oracle Access Manager and Oracle Identity Governance. If you are not deploying any of these products, you can amend this file as per your requirements. It is provided here as an example and to make deployment of a full suite simpler.

Perform a global search and replace to change these entries.

> **Note:**
>
> Do not change the values for the following variables:
>
> - *<DenySSORead ACI>*
> - *<AllowSSORead ACI>*
> - *<AllowSSOAll ACI>*

## Setting Passwords

You can set user passwords in the file by providing a value for the LDAP attribute: `userPassword`.

Passwords entered as plain text will be encrypted upon loading. For ease of use, you can search for the term <PASSWORD> in the file, for the entries you have to provide.

# Creating OUD Containers

Create the server overrides file first and then use the `Helm` command to create the OUD containers.

- Creating a Server Overrides File
- Changing the OUD Heap Size
- Enabling Assured Replication
- Creating Containers
- Troubleshooting the OUD Instances

# Creating a Server Overrides File

OUD containers are deployed using Helm. You must create a `helm` override file to customize the deployment based on your deployment needs. This file is used to determine how the OUD pods will be created. You can specify the following details in this file:

- The container images to use.
- The number of replicas to create.
- The number of resources to allocate to each pod.
- The base DN to create.
- An `ldif` file to load to seed the OUD data.
- An `ldif` file to create the schema extensions.
- The access control lists (ACLs) to create.
- The indexes that need to be created on each pod (base and replicas).
- Any specific `ds_config` commands you want to run at pod instantiation.

Create this file by substituting values from Table 14-2.

The following is a sample override file for OUD.

`/workdir/OUD/override_oud.yaml`

> **Note:**
>
> You can find a sample of this file along with the sample files you downloaded from GitHub. It will be located in `/workdir/fmw-kubernetes/FMWKubernetesMAA/OracleEnterpriseDeploymentAutomation/OracleIdentityManagement/templates/oud`.

A resource limit is defined as the maximum resources allowed for each pod:

- CPU measured in CPU cores. Value of 1 = 1 CPU core or 1 virtual core.
- Memory is measured in standard units 1G = 1GB.

Resources are the initial startup values added to the pod:

- CPU measured in CPU cycles. Value of 1000m = 1 CPU core or 1 virtual core.
- Memory is measured in standard units 1G = 1GB.

The server tuning values should not conflict with these values.

```
image:
  repository: <OUD_REPOSITORY>
  tag: <OUD_VER>
  pullPolicy: IfNotPresent

busybox:
  image: docker.io/busybox

imagePullSecrets:
```

```
  - name: regcred
oudConfig:
  baseDN: <LDAP_SEARCHBASE>
  rootUserDN: <LDAP_ADMIN_USER>
  rootUserPassword: <LDAP_ADMIN_PWD>
  sleepBeforeConfig: 1300
  resources:
    limits:
      cpu: 1
      memory: 2Gi
    requests:
      cpu: 500m
      memory: 1Gi

persistence:
  type: networkstorage
  networkstorage:
    nfs:
      server: <PVSERVER>
      path: <OUD_SHARE>
  size: 30Gi

configVolume:
  enabled: true
  type: networkstorage
  networkstorage:
    nfs:
      server: <PVSERVER>
      path: <OUD_CONFIG_SHARE>
  mountPath: /u01/oracle/config-input

replicaCount: <OUD_REPLICAS>

ingress:
  enabled: false
  type: nginx
  tlsEnabled: true

cronJob:
  kubectlImage:
    repository: bitnami/kubectl
    tag: <KUBERNETES_VER>
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: dockercred

baseOUD:
  envVars:
    - name: schemaConfigFile_1
      value: /u01/oracle/config-input/99-user.ldif
    - name: restartAfterSchemaConfig
      value: "true"
    - name: importLdif_1
      value: --append --replaceExisting --includeBranch ${baseDN} --backendID
userRoot --ldifFile /u01/oracle/config-input/base.ldif --rejectFile /u01/
oracle/config-input/rejects.ldif --skipFile /u01/oracle/config-input/skip.ldif
```

```
    - name: serverTuning
      value: -Xms1024m -Xmx2048m -d64 -XX:+UseCompressedOops -server -Xmn1g -
XX:MaxTenuringThreshold=1 -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=60
    - name: dsconfig_1
      value: set-global-configuration-prop --set lookthrough-limit:75000
    - name: dsconfig_2
      value: set-access-control-handler-prop --remove global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; deny (all) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_3
      value: set-access-control-handler-prop --add global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; allow
(read,search,compare,add,write,delete,export) groupdn=\"ldap:///
cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_4
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"1.3.6.1.4.1.26027.1.5.4 || 1.3.6.1.4.1.26027.2.3.4\")
(version 3.0; acl \"<LDAP_OIGADMIN_GRP> control access\"; allow(read)
groupdn=\"ldap:///cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_5
      value: set-access-control-handler-prop --add global-
aci:"(target=\"ldap:///\")(targetscope=\"base\")
(targetattr=\"lastExternalChangelogCookie\")(version 3.0; acl \"User-Visible
lastExternalChangelog\"; allow (read,search,compare) groupdn=\"ldap:///
cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_6
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.840.113556.1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18
|| 2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_7
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 ||
2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9 || 1.3.6.1.4.1.26027.1.5.4 ||
1.3.6.1.4.1.26027.2.3.4\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_8
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31\") (version 3.0; acl \"Anonymous control access\";
allow(read) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_9
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31 || 1.2.840.113556.1.4.319\") (version 3.0; acl
\"Anonymous control access\"; allow(read) userdn=\"ldap:///anyone\";)"
```

```
    - name: dsconfig_10
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGranter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: dsconfig_11
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGrantee --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: dsconfig_12
      value: create-local-db-index --element-name userRoot --index-name obid
--set index-type:equality --set index-type:presence --set index-type:substring
    - name: dsconfig_13
      value: create-local-db-index --element-name userRoot --index-name
oblocationdn --set index-type:equality
    - name: dsconfig_14
      value: create-local-db-index --element-name userRoot --index-name
oblocationname --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_15
      value: create-local-db-index --element-name userRoot --index-name
oblocationtitle --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_16
      value: create-local-db-index --element-name userRoot --index-name
obrectangle --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: dsconfig_17
      value: create-local-db-index --element-name userRoot --index-name
obdirectreports --set index-type:equality
    - name: dsconfig_18
      value: create-local-db-index --element-name userRoot --index-name
obindirectmanager --set index-type:equality
    - name: dsconfig_19
      value: create-local-db-index --element-name userRoot --index-name
obuseraccountcontrol --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_20
      value: create-local-db-index --element-name userRoot --index-name
obobjectclass --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: dsconfig_21
      value: create-local-db-index --element-name userRoot --index-name
obparentlocationdn --set index-type:equality
    - name: dsconfig_22
      value: create-local-db-index --element-name userRoot --index-name
obgroupcreator --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_23
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptiontype --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_24
      value: create-local-db-index --element-name userRoot --index-name
obgroupdynamicfilter --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_25
```

```
      value: create-local-db-index --element-name userRoot --index-name
obgroupexpandeddynamic --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_26
      value: create-local-db-index --element-name userRoot --index-name
obgroupadministrator --set index-type:equality
    - name: dsconfig_27
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptionfilter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: dsconfig_28
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribemessage --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_29
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribenotification --set index-type:equality --set index-
type:presence --set index-type:substring
    - name: dsconfig_30
      value: create-local-db-index --element-name userRoot --index-name
obgrouppuredynamic --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_31
      value: list-local-db-indexes --element-name userRoot
    - name: rebuildIndex_1
      value: --rebuildAll
    - name: restartAfterRebuildIndex
      value: "true"

replOUD:
  envVars:
    - name: serverTuning
      value: -Xms1024m -Xmx2048m -d64 -XX:+UseCompressedOops -server -Xmn1g -
XX:MaxTenuringThreshold=1 -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=60
    - name: dsconfig_1
      value: set-global-configuration-prop --set lookthrough-limit:75000
    - name: dsconfig_2
      value: set-access-control-handler-prop --remove global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; deny (all) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_3
      value: set-access-control-handler-prop --add global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; allow
(read,search,compare,add,write,delete,export) groupdn=\"ldap:///
cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_4
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.840.113556.1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18
|| 2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_5
      value: set-access-control-handler-prop --add global-
```

```
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 ||
2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9 || 1.3.6.1.4.1.26027.1.5.4 ||
1.3.6.1.4.1.26027.2.3.4\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_6
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31\") (version 3.0; acl \"Anonymous control access\";
allow(read) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_7
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31 || 1.2.840.113556.1.4.319\") (version 3.0; acl
\"Anonymous control access\"; allow(read) userdn=\"ldap:///anyone\";)"
    - name: post_dsreplication_dsconfig_2
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGranter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: post_dsreplication_dsconfig_3
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGrantee --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: post_dsreplication_dsconfig_4
      value: create-local-db-index --element-name userRoot --index-name obid
--set index-type:equality --set index-type:presence --set index-type:substring
    - name: post_dsreplication_dsconfig_5
      value: create-local-db-index --element-name userRoot --index-name
oblocationdn --set index-type:equality
    - name: post_dsreplication_dsconfig_6
      value: create-local-db-index --element-name userRoot --index-name
oblocationname --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_7
      value: create-local-db-index --element-name userRoot --index-name
oblocationtitle --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_8
      value: create-local-db-index --element-name userRoot --index-name
obrectangle --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: post_dsreplication_dsconfig_9
      value: create-local-db-index --element-name userRoot --index-name
obdirectreports --set index-type:equality
    - name: post_dsreplication_dsconfig_10
      value: create-local-db-index --element-name userRoot --index-name
obindirectmanager --set index-type:equality
    - name: post_dsreplication_dsconfig_11
      value: create-local-db-index --element-name userRoot --index-name
obuseraccountcontrol --set index-type:equality --set index-type:presence --
set index-type:substring
```

```
    - name: post_dsreplication_dsconfig_12
      value: create-local-db-index --element-name userRoot --index-name
obobjectclass --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: post_dsreplication_dsconfig_13
      value: create-local-db-index --element-name userRoot --index-name
obparentlocationdn --set index-type:equality
    - name: post_dsreplication_dsconfig_14
      value: create-local-db-index --element-name userRoot --index-name
obgroupcreator --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_15
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptiontype --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_16
      value: create-local-db-index --element-name userRoot --index-name
obgroupdynamicfilter --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_17
      value: create-local-db-index --element-name userRoot --index-name
obgroupexpandeddynamic --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_18
      value: create-local-db-index --element-name userRoot --index-name
obgroupadministrator --set index-type:equality
    - name: post_dsreplication_dsconfig_19
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptionfilter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: post_dsreplication_dsconfig_20
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribemessage --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_21
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribenotification --set index-type:equality --set index-
type:presence --set index-type:substring
    - name: post_dsreplication_dsconfig_22
      value: create-local-db-index --element-name userRoot --index-name
obgrouppuredynamic --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_23
      value: list-local-db-indexes --element-name userRoot
    - name: rebuildIndex_1
      value: --rebuildAll
    - name: restartAfterRebuildIndex
      value: "true"
```

For example:

```
image:
  repository: oracle/oud
  tag: 12.2.1.4-jdk8-ol7-220411.1613
  pullPolicy: IfNotPresent
```

```
busybox:
  image: docker.io/busybox

imagePullSecrets:
  - name: regcred
oudConfig:
  baseDN: dc=example,dc=com
  rootUserDN: cn=oudadmin
  rootUserPassword: password
  sleepBeforeConfig: 1300

persistence:
  type: networkstorage
  networkstorage:
    nfs:
      server: mynfsserver.example.com
      path: /exports/IAMPVS/oudpv
  size: 30Gi

configVolume:
  enabled: true
  type: networkstorage
  networkstorage:
    nfs:
      server: mynfsserver.example.com
      path: /exports/IAMPVS/oudconfigpv
  mountPath: /u01/oracle/config-input

replicaCount: 1

ingress:
  enabled: false
  type: nginx
  tlsEnabled: true

cronJob:
  kubectlImage:
    repository: bitnami/kubectl
    tag: <KUBERNETES_VER>
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: dockercred

baseOUD:
  envVars:
    - name: schemaConfigFile_1
      value: /u01/oracle/config-input/99-user.ldif
    - name: restartAfterSchemaConfig
      value: "true"
    - name: importLdif_1
      value: --append --replaceExisting --includeBranch ${baseDN} --backendID
userRoot --ldifFile /u01/oracle/config-input/base.ldif --rejectFile /u01/
oracle/config-input/rejects.ldif --skipFile /u01/oracle/config-input/skip.ldif
    - name: serverTuning
      value: -Xms1024m -Xmx2048m -d64 -XX:+UseCompressedOops -server -Xmn1g -
```

```
XX:MaxTenuringThreshold=1 -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=60
    - name: dsconfig_1
      value: set-global-configuration-prop --set lookthrough-limit:75000
    - name: dsconfig_2
      value: set-access-control-handler-prop --remove global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; deny (all) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_3
      value: set-access-control-handler-prop --add global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; allow
(read,search,compare,add,write,delete,export) groupdn=\"ldap:///
cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_4
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"1.3.6.1.4.1.26027.1.5.4 || 1.3.6.1.4.1.26027.2.3.4\")
(version 3.0; acl \"<LDAP_OIGADMIN_GRP> control access\"; allow(read)
groupdn=\"ldap:///cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_5
      value: set-access-control-handler-prop --add global-
aci:"(target=\"ldap:///\")(targetscope=\"base\")
(targetattr=\"lastExternalChangelogCookie\")(version 3.0; acl \"User-Visible
lastExternalChangelog\"; allow (read,search,compare) groupdn=\"ldap:///
cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_6
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.840.113556.1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18
|| 2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_7
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 ||
2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9 || 1.3.6.1.4.1.26027.1.5.4 ||
1.3.6.1.4.1.26027.2.3.4\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_8
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31\") (version 3.0; acl \"Anonymous control access\";
allow(read) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_9
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31 || 1.2.840.113556.1.4.319\") (version 3.0; acl
\"Anonymous control access\"; allow(read) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_10
      value: create-local-db-index --element-name userRoot --index-name
```

```
orclImpersonationGranter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: dsconfig_11
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGrantee --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: dsconfig_12
      value: create-local-db-index --element-name userRoot --index-name obid
--set index-type:equality --set index-type:presence --set index-type:substring
    - name: dsconfig_13
      value: create-local-db-index --element-name userRoot --index-name
oblocationdn --set index-type:equality
    - name: dsconfig_14
      value: create-local-db-index --element-name userRoot --index-name
oblocationname --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_15
      value: create-local-db-index --element-name userRoot --index-name
oblocationtitle --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_16
      value: create-local-db-index --element-name userRoot --index-name
obrectangle --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: dsconfig_17
      value: create-local-db-index --element-name userRoot --index-name
obdirectreports --set index-type:equality
    - name: dsconfig_18
      value: create-local-db-index --element-name userRoot --index-name
obindirectmanager --set index-type:equality
    - name: dsconfig_19
      value: create-local-db-index --element-name userRoot --index-name
obuseraccountcontrol --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_20
      value: create-local-db-index --element-name userRoot --index-name
obobjectclass --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: dsconfig_21
      value: create-local-db-index --element-name userRoot --index-name
obparentlocationdn --set index-type:equality
    - name: dsconfig_22
      value: create-local-db-index --element-name userRoot --index-name
obgroupcreator --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_23
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptiontype --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_24
      value: create-local-db-index --element-name userRoot --index-name
obgroupdynamicfilter --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_25
      value: create-local-db-index --element-name userRoot --index-name
obgroupexpandeddynamic --set index-type:equality --set index-type:presence --
```

```
set index-type:substring
    - name: dsconfig_26
      value: create-local-db-index --element-name userRoot --index-name
obgroupadministrator --set index-type:equality
    - name: dsconfig_27
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptionfilter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: dsconfig_28
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribemessage --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: dsconfig_29
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribenotification --set index-type:equality --set index-
type:presence --set index-type:substring
    - name: dsconfig_30
      value: create-local-db-index --element-name userRoot --index-name
obgrouppuredynamic --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: dsconfig_31
      value: list-local-db-indexes --element-name userRoot
    - name: rebuildIndex_1
      value: --rebuildAll
    - name: restartAfterRebuildIndex
      value: "true"

replOUD:
  envVars:
    - name: serverTuning
      value: -Xms1024m -Xmx2048m -d64 -XX:+UseCompressedOops -server -Xmn1g -
XX:MaxTenuringThreshold=1 -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=60
    - name: dsconfig_1
      value: set-global-configuration-prop --set lookthrough-limit:75000
    - name: dsconfig_2
      value: set-access-control-handler-prop --remove global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; deny (all) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_3
      value: set-access-control-handler-prop --add global-
aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*\")(version 3.0; acl
\"External changelog access\"; allow
(read,search,compare,add,write,delete,export) groupdn=\"ldap:///
cn=<LDAP_OIGADMIN_GRP>,cn=groups,${baseDN}\";)"
    - name: dsconfig_4
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.840.113556.1.4.319 || 1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18
|| 2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_5
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"1.3.6.1.1.12 || 1.3.6.1.1.13.1 || 1.3.6.1.1.13.2 ||
1.2.826.0.1.3344810.2.3 || 2.16.840.1.113730.3.4.18 ||
```

```
2.16.840.1.113730.3.4.9 || 1.2.840.113556.1.4.473 ||
1.3.6.1.4.1.42.2.27.9.5.9 || 1.3.6.1.4.1.26027.1.5.4 ||
1.3.6.1.4.1.26027.2.3.4\") (version 3.0; acl \"Authenticated users control
access\"; allow(read) userdn=\"ldap:///all\";)"
    - name: dsconfig_6
      value: set-access-control-handler-prop --remove global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31\") (version 3.0; acl \"Anonymous control access\";
allow(read) userdn=\"ldap:///anyone\";)"
    - name: dsconfig_7
      value: set-access-control-handler-prop --add global-
aci:"(targetcontrol=\"2.16.840.1.113730.3.4.2 || 2.16.840.1.113730.3.4.17 ||
2.16.840.1.113730.3.4.19 || 1.3.6.1.4.1.4203.1.10.2 ||
1.3.6.1.4.1.42.2.27.8.5.1 || 2.16.840.1.113730.3.4.16 ||
2.16.840.1.113894.1.8.31 || 1.2.840.113556.1.4.319\") (version 3.0; acl
\"Anonymous control access\"; allow(read) userdn=\"ldap:///anyone\";)"
    - name: post_dsreplication_dsconfig_2
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGranter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: post_dsreplication_dsconfig_3
      value: create-local-db-index --element-name userRoot --index-name
orclImpersonationGrantee --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: post_dsreplication_dsconfig_4
      value: create-local-db-index --element-name userRoot --index-name obid
--set index-type:equality --set index-type:presence --set index-type:substring
    - name: post_dsreplication_dsconfig_5
      value: create-local-db-index --element-name userRoot --index-name
oblocationdn --set index-type:equality
    - name: post_dsreplication_dsconfig_6
      value: create-local-db-index --element-name userRoot --index-name
oblocationname --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_7
      value: create-local-db-index --element-name userRoot --index-name
oblocationtitle --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_8
      value: create-local-db-index --element-name userRoot --index-name
obrectangle --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: post_dsreplication_dsconfig_9
      value: create-local-db-index --element-name userRoot --index-name
obdirectreports --set index-type:equality
    - name: post_dsreplication_dsconfig_10
      value: create-local-db-index --element-name userRoot --index-name
obindirectmanager --set index-type:equality
    - name: post_dsreplication_dsconfig_11
      value: create-local-db-index --element-name userRoot --index-name
obuseraccountcontrol --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_12
      value: create-local-db-index --element-name userRoot --index-name
```

```
obobjectclass --set index-type:equality --set index-type:presence --set index-
type:substring
    - name: post_dsreplication_dsconfig_13
      value: create-local-db-index --element-name userRoot --index-name
obparentlocationdn --set index-type:equality
    - name: post_dsreplication_dsconfig_14
      value: create-local-db-index --element-name userRoot --index-name
obgroupcreator --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_15
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptiontype --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_16
      value: create-local-db-index --element-name userRoot --index-name
obgroupdynamicfilter --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_17
      value: create-local-db-index --element-name userRoot --index-name
obgroupexpandeddynamic --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_18
      value: create-local-db-index --element-name userRoot --index-name
obgroupadministrator --set index-type:equality
    - name: post_dsreplication_dsconfig_19
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscriptionfilter --set index-type:equality --set index-type:presence
--set index-type:substring
    - name: post_dsreplication_dsconfig_20
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribemessage --set index-type:equality --set index-type:presence --
set index-type:substring
    - name: post_dsreplication_dsconfig_21
      value: create-local-db-index --element-name userRoot --index-name
obgroupsubscribenotification --set index-type:equality --set index-
type:presence --set index-type:substring
    - name: post_dsreplication_dsconfig_22
      value: create-local-db-index --element-name userRoot --index-name
obgrouppuredynamic --set index-type:equality --set index-type:presence --set
index-type:substring
    - name: post_dsreplication_dsconfig_23
      value: list-local-db-indexes --element-name userRoot
    - name: rebuildIndex_1
      value: --rebuildAll
    - name: restartAfterRebuildIndex
      value: "true"
```

Only the first three indices are required for `<KUBERNETES_VER>`. For example: `1.20.6`.

If the organization prevents access to the internet for public images, you can host the `kubectl` image in your own registry and update the repository value in the file above to match this value.

> **Note:**
>
> If you want to enable Enterprise User Security (EUS) integration in OUD, add the following line in the `oudConfig` section:
>
> ```
> oudConfig:
>   integration: eus
> ```

If you are not using OAM, you can remove the following:

- Lines `dsconfig_1` to `dsconfig_30` from the `baseOUD` section of the file.
- Lines `post_dsreplication_dsconfig_2` to `post_dsreplication_dsconfig_22` from the `replOUD` section of the file.
- <`schemaConfigFile_1`> from the Server Overrides file if you do not want to extend the schema definition for Oracle Access Manager.

## Changing the OUD Heap Size

Instructions for tuning OUD is beyond the scope of this guide. For maximum and minimum heap size recommendations for OUD, see Sizing Guidelines.

To modify the heap size of an OUD instance, set `Xms` to the minimum heap size and `Xmx` to the maximum heap size in the `serverTuning` section of the Server Overrides file.

For example, to set the values for a small system, the entry would be as follows:

```
- name: serverTuning
value: -Xms4096m -Xmx8192m -d64 -XX:+UseCompressedOops -server -Xmn1g -
XX:MaxTenuringThreshold=1 -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=60
```

For more information about the performance tuning recommendations, see Deep Dive into Oracle Unified Directory 12.2.1.4.0 Performance.

## Enabling Assured Replication

If you want to enable assured replication between the OUD instances, add the following content to the `replOUD` section of the `override_oud.yaml` file:

```
replOUD:
  envVars:
    - name: post_dsreplication_dsconfig_1
      value: set-replication-domain-prop --domain-name ${baseDN} --advanced --
set assured-type:safe-read
    - name: execCmd_1
      value: /u01/oracle/user_projects/${OUD_INSTANCE_NAME}/OUD/bin/dsconfig
--no-prompt --hostname ${sourceHost} --port ${adminConnectorPort} --bindDN "$
{rootUserDN}" --bindPasswordFile /u01/oracle/user_projects/$
{OUD_INSTANCE_NAME}/admin/rootPwdFile.txt --trustAll set-replication-domain-
prop --domain-name ${baseDN} --advanced --set assured-type:safe-read --set
```

```
assured-sd-level:2 --set assured-timeout:5s --provider-name "Multimaster
Synchronization"
```

## Creating Containers

After you create the Helm Override file, you now need to create the OUD containers using the `helm` command.

The `helm` command will:

- Create OUD instances
- Add OAM Schema Extensions
- Seed OAM/OIG users and groups
- Update the OUD change log permissions
- Create additional OUD indexes
- Rebuild OUD indexes
- Create Kubernetes services for OUD

To create containers, use the following command:

```
cd /workdir/OUD/samples/kubernetes/helm
```

```
helm install --namespace <OUDNS> --values /workdir/OUD/override_oud.yaml
<OUD_PREFIX> oud-ds-rs
```

For example:

```
helm install --namespace oudns --values /workdir/OUD/override_oud.yaml edg
oud-ds-rs
```

> **Note:**
>
> `edg` is used to prefix each of the OUD instances. It can be any value.

## Troubleshooting the OUD Instances

You can monitor the creation of each OUD instance using the following commands:

**Objects created in the namespace**:

```
kubectl -n oudns get all -o wide
```

Only when you see each container with the status `READY 1/1 and Status = Running` will the installation and configuration be complete.

If you do not see objects being created, use the following command to check the issue:

```
kubectl get pod -n oudns
```

For a detailed description, use:

```
kubectl describe pod -n oudns
```

**Container Logs**

To view the progress of each container as it is being created, use a command similar to:

```
kubectl logs edg-oud-ds-rs-0 -n oudns
```

Review the `skips.ldif` and `rejects.ldif` files that are created after the OUD servers are initialized. These files are created when the `base.ldif` and `99-user.ldif` files are loaded. The OUD servers start even if there are errors but all of the data is not loaded causing problems down the road for other product integrations. You may not see the errors by reviewing only the OUD logs.

# Creating External Access to OUD

By default, the OUD deployment gets created with all the components configured as ClusterIP services. This means that the Oracle Unified Directory components are visible only within the Kubernetes cluster.

If you are going to access the cluster only from within Kubernetes, then this is sufficient. However, if you want to interact with Oracle Unified Directory from outside of Kubernetes, you should create either an Ingress Controller service or individual NodePort services.

- Creating the Kubernetes NodePort Services

## Creating the Kubernetes NodePort Services

To create the native Kubernetes NodePort Services, you have to perform the steps provided in this section. If you want to expose the OUD services using an Ingress controller, see Installing and Configuring Ingress Controller.

- Creating an LDAP NodePort Service

### Creating an LDAP NodePort Service

To create an LDAP NodePort Service:

1. Create a text file called `/workdir/OUD/oud_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OUD_PREFIX>-oud-ds-rs-lbr-ldap-nodeport
  namespace: <OUDNS>
spec:
  type: NodePort
  selector:
    app.kubernetes.io/instance: <OUD_PREFIX>
    app.kubernetes.io/name: oud-ds-rs
  ports:
    - name: ldap
      targetPort: 1389
```

```
      port: 1389
      nodePort: <OUD_LDAP_K8>
      protocol: TCP
    - name: ldaps
      targetPort: 1636
      port: 1636
      nodePort: <OUD_LDAPS_K8>
      protocol: TCP
```

For example:

```
kind: Service
apiVersion: v1
metadata:
  name: oud-edg-oud-ds-rs-lbr-ldap-nodeport
  namespace: oudns
spec:
  type: NodePort
  selector:
    app.kubernetes.io/instance: oud-edg
    app.kubernetes.io/name: oud-ds-rs
  ports:
    - name: ldap
      targetPort: 1389
      port: 1389
      nodePort: 31389
      protocol: TCP
    - name: ldaps
      targetPort: 1636
      port: 1636
      nodePort: 31636
      protocol: TCP
```

2. Create the service using the following command:

```
kubectl create -f /workdir/OUD/oud_nodeport.yaml
```

The output appears as follows:

```
service/edg--oud-ds-rs-lbr-ldap-nodeport created
```

# Centralized Monitoring Using Grafana and Prometheus

There is no specific metric collection for OUD. However, you can monitor the OUD pods using the standard Kubernetes Dashboard in Kibana.

# Centralized Log File Monitoring Using Elasticsearch and Kibana

If you are using Elasticsearch and Kibana, you can configure a Logstash pod to send the log files to the centralized Elasticsearch/Kibana console. Before you configure the Logstash pod, ensure that you have access to a centralized Elasticsearch deployment.

- OUD persistent volume, so it can be loaded by the Logstash pod to hunt for log files.

- The location of the log files in the persistent volumes.
- The location of the centralized Elasticsearch.

To configure the Logstash pod, perform the following steps. The assumption is that you have an Elasticsearch running inside the Kubernetes cluster, in a namespace called `elkns`.

- Creating a Secret for Elasticsearch
- Creating a Configuration Map for ELK Certificate
- Configuring Log File Monitoring for OUD

## Creating a Secret for Elasticsearch

Logstash requires credentials to connect to the elasticsearch deployment. These credentials are stored in Kubernetes as a secret.

If your Elasticsearch uses an API key for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OUDNS> --from-
literal password=<ELK_APIKEY>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oudns --from-
literal password=afshfashfkahf5f
```

If Elasticsearch uses a user name and password for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OUDNS> --from-
literal password=<ELK_PWD>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oudns --from-
literal password=mypassword
```

You can find the Elasticsearch password using the following command:

```
kubectl get secret elasticsearch-es-elastic-user -n <ELKNS> -o go-
template='{{.data.elastic | base64decode}}'
```

## Creating a Configuration Map for ELK Certificate

If you have configured a production ready Elasticsearch deployment, you would have configured SSL. Logstash needs to trust the Elasticsearch certificate to be able to communicate with it. To enable this trust, you should create a configuration map with the contents of the Elasticsearch certificate.

You would have already saved the Elasticsearch self-signed certificate. See Copying the Elasticsearch Certificate. If you have a production certificate you can use that instead.

Create the configuration map using the certificate, run the following command:

```
kubectl create configmap elk-cert --from-file=<WORKDIR>/ELK/elk.crt -n <OUDNS>
```

For example:

```
kubectl create configmap elk-cert --from-file=/workdir/ELK/elk.crt -n oudns
```

# Configuring Log File Monitoring for OUD

Complete the following steps to configure log file monitoring:

- Creating a Configuration Map for Logstash
- Creating a Logstash Deployment

## Creating a Configuration Map for Logstash

Logstash looks for log files in the OUD installations and sends them to the centralized Elasticsearch. The configuration map is used to instruct Logstash where the log files reside and where to send them.

1. Create a file called `<WORKDIR>/OUD/logstash_cm.yaml` with the following contents:

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: oud-logstash-configmap
 namespace: <OUDNS>
data:
 logstash.yaml: |
 #http.host: "0.0.0.0"
 logstash-config.conf: |
   input {
      file {
       path => "/u01/oracle/user_projects/oud-ds-rs-*/logs/*.log"
       type => "setup-logs"
       start_position => beginning
       sincedb_path => "/dev/null"
      }
      file {
       path => "/u01/oracle/user_projects/oud-ds-rs-*/OUD/logs/*.log"
       type => "access-logs"
       start_position => beginning
       sincedb_path => "/dev/null"
      }
     }
     filter {
      if [type] == "setup-logs" {
       grok {
        match => [ "message", "<%{DATA:log_timestamp}> <%{WORD:log_level}>
<%{WORD:thread}> <%{HOSTNAME:hostname}> <%{HOSTNAME:servername}> <%
{DATA:timer}> <<%{DATA:kernel}>> <> <%{DATA:uuid}> <%{NUMBER:timestamp}> <%
{DATA:misc}> <%{DATA:log_number}> <%{DATA:log_message}>" ]
        }
```

```
        }
       if [type] == "access-logs" {
        grok {
         match => [ "message", "\[%{TIMESTAMP_ISO8601:timestamp}\] \[%
{DATA:component}\] \[%{LOGLEVEL:loglevel}\] \[%{DATA:misc}\] \[%
{DATA:logtype}\] \[%{DATA:host}\] \[%{DATA:nwaddr}\] %
{GREEDYDATA:message}" ]
        }
       }
       if "_grokparsefailure" in [tags] {
        mutate {
         remove_tag => [ "_grokparsefailure" ]
        }
       }
      }
   output {
    elasticsearch {
     hosts => ["<ELK_HOST>"]
     cacert => '/usr/share/logstash/config/certs/elk.crt'
     user => "<ELK_USER>"
     password => "<ELK_USER_PWD>"
     index => "oudlogs-000001"
     ssl => true
     ssl_certificate_verification => false
    }
   }
```

2. Save the file.
3. Create the configuration map using the following command:

   ```
   kubectl create -f <WORKDIR>/OUD/logstash_cm.yaml
   ```

   For example:

   ```
   kubectl create -f /workdir/OUD/logstash_cm.yaml
   ```

4. Validate that the configuration map has been created by using the following command:

   ```
   kubectl get cm -n <OUDNS>
   ```

   You should see `oud-logstash-configmap` in the list of configuration maps.

## Creating a Logstash Deployment

After you create the configuration map, you can create the Logstash deployment. This deployment resides in the OUD namespace.

1. Create a file called `<WORKDIR>/OUD/logstash.yaml` with the following contents:

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: oud-logstash
     namespace: <OUDNS>
   ```

```
spec:
  selector:
    matchLabels:
      k8s-app: logstash
  template: # create pods using pod definition in this template
    metadata:
      labels:
        k8s-app: logstash
    spec:
      imagePullSecrets:
      - name: dockercred
      containers:
      - command:
        - logstash
        image: logstash:<ELK_VER>
        imagePullPolicy: IfNotPresent
        name: oud-logstash
        env:
        - name: ELASTICSEARCH_PASSWORD
          valueFrom:
            secretKeyRef:
              name: elasticsearch-pw-elastic
              key: password
        ports:
        - containerPort: 5044
          name: logstash
        volumeMounts:
        - mountPath: /u01/oracle/user_projects
          name: oud-storage-volume
        - name: shared-logs
          mountPath: /shared-logs
        - mountPath: /usr/share/logstash/pipeline/
          name: oud-logstash-pipeline
        - mountPath: /usr/share/logstash/config/certs
          name: elk-cert
      volumes:
      - configMap:
          defaultMode: 420
          items:
          - key: logstash-config.conf
            path: logstash-config.conf
          name: oud-logstash-configmap
        name: oud-logstash-pipeline
      - configMap:
          defaultMode: 420
          items:
          - key: ca.crt
            path: elk.crt
          name: elk-cert
      - name: oud-storage-volume
        persistentVolumeClaim:
          claimName: <OUD_POD_PREFIX>-oud-ds-rs-pvc
      - name: shared-logs
        emptyDir: {}
```

> **Note:**
>
> If you are using your own registry, include the registry name in the image tag. If you have created a `regcred` secret for your registry, replace the `imagePullSecrets` name with the secret name you created. For example: `regcred`.

2. Save the file.
3. Create the Logstash deployment by using the following command:

   ```
   kubectl create -f <WORKDIR>/OUD/logstash.yaml
   ```

   For example:

   ```
   kubectl create -f /workdir/OUD/logstash.yaml
   ```

4. You can now create a pod called `logstash` by using the following command:

   ```
   kubectl get pod -n oudns
   ```

   Your logs will now be available in the Kibana console.

15

# Installing and Configuring Oracle Unified Directory Services Manager

Oracle Unified Directory Service Manager (OUDSM) is a Graphical User Interface (GUI) tool that is used to manage Oracle Unified Directory. It is not mandatory to install OUDSM in the production environments. However, OUDSM makes managing Oracle Unified Directory easier.

Oracle recommends that if you are installing OUDSM, you should install it into a different Kubernetes namespace as OUD. OUDSM contains no data and separating it from OUD simplifies disaster recovery for OUD. You can access OUDSM directly through:

- NodePort Services
- Ingress Controller
- Oracle HTTP Server if added to a virtual host such as `iadadmin.example.com`

This chapter includes the following topics:

- Configuring Oracle Unified Directory Services Manager
  Oracle Unified Directory Services Manager (OUDSM) is a tool that is used to manage the Oracle Unified Directory. It is optional.
- Setting Up a Product Specific Work Directory
- Creating a Kubernetes Namespace
  You have to create a namespace to contain all the objects for Oracle Unified Directory Services Manager.
- Creating a Container Registry Secret
  Oracle recommends that you use a container registry. If you use a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.
- Creating OUDSM Containers
- Creating External Access to OUDSM
- Configuring Oracle HTTP Server for Oracle Unified Directory Services Manager
- Centralized Log File Monitoring Using Elasticsearch and Kibana

## Configuring Oracle Unified Directory Services Manager

Oracle Unified Directory Services Manager (OUDSM) is a tool that is used to manage the Oracle Unified Directory. It is optional.

This chapter describes how you can install OUDSM inside a Kubernetes cluster.

- Kubernetes/Ingress Services
- Variables Used in this Chapter

# Kubernetes/Ingress Services

After you configure OUDSM, the following OUDSM service will be available on each worker node:

**Table 15-1 OUDSM Service on Each Worker Node**

| Service | Type | Service Port | Mapped Port |
|---|---|---|---|
| OUDSM | NodePort | 30901 | 7001 |

> **Note:**
>
> OUDSM randomly picks its own Kubernetes service port. The number given in this table is only an example.

If you use an Ingress-based deployment, the following Ingress service is created as part of the deployment:

**Table 15-2 Ingress Services**

| Service Name | Host Name |
|---|---|
| `oudsm-ingress` | `oudsm.example.com` |

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 15-3 The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| <*WORKDIR*> | `/workdir/OUD` | The location where you want to create the working directory for OUD. |
| <*REGISTRY_ADDRESS*> | `iad.ocir.io/<mytenancy>` | The location of the registry. If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/oud_cpu`. |
| <*REG_USER*> | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The user id you use to log in to the registry. If you are use the Oracle container registry, this value will be your Oracle single sign-on user name. |
| <*REG_PWD*> | <*password*> | The registry user password. |

**Table 15-3 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OUDSM_REPOSITORY>* | `oracle/oudsm`<br>`local/oracle/oudsm`<br>`container-registry.oracle.com/middleware/oudsm_cpu`<br>`<REGISTRY_ADDRESS>/oracle/oudsm` | The name of the OUDSM software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/oudsm`. If you are using OLCNE, the value will be `local/oracle/oudsm`.<br>If you are using the Oracle container registry, the value will be:`container-registry.oracle.com/middleware/oudsm_cpu`.<br>If you are using a container registry, the value will be the name of the registry with the product name: *<REGISTRY_ADDRESS>*`/oracle/oudsm`. |
| *<OUDSM_VER>* | `12.2.1.4-jdk8-ol7-220411.1608` or latest | The version of the image you want to use. This will be the version you have downloaded and staged either locally or in your container registry |
| *<PVSERVER>* | `mynfsserver.example.com` | The name of the NFS server.<br>**Note**: This name should be resolvable inside the Kubernetes cluster. |
| *<OUDSM_WEBLOGIC_USER>* | `weblogic` | The name of the user who administers the OUDSM domain. |
| *<OUDSM_WEBLOGIC_PWD>* | - | The password assigned to the admin user (`<AdminUser>` account. |
| *<OUDSM_SERVICE_PORT>* | `30901` | Port to use for OUDSM requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| *<OUDSM_SHARE>* | `/exports/IAMPVS/oudsmpv` | The NFS mount point for the share. |
| *<OUSDM_INGRESS_HOST>* | `oudsm.example.com` | If you are using a dedicated hostname for OUDSM, then set this value to the name of that host. For example: `oudsm.example.com`. If you are making OUDSM accessible through an existing virtual host, then set this value to that host name. For example: `iadadmin.example.com`. |

**Table 15-3 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<ELK_HOST>* | `https://elasticsearch-es-http.elkns.svc:9200` | The host and port of the centralized Elasticsearch deployment. This host can be inside the Kubernetes cluster or external to it. This host is used only when Elasticsearch is used. |
| *<ELK_VER>* | `8.11.0` | The version of Elasticsearch you want to use. |
| *<ELK_USER_PWD>* | *<password>* | The password assigned to the ELK user. See Creating a Role and a User for Logstash. |

# Setting Up a Product Specific Work Directory

Before you begin the installation, you must have already downloaded and staged the Oracle Unified Directory Service Manager container image and the code repostory. See Identifying and Obtaining Software Distributions for an Enterprise Deployment.

This section describes the procedure to copy the downloaded sample deployment scripts to a temporary working directory for OUDSM.

1. Create the working directory.

   ```
   mkdir <WORKDIR>
   ```

   For example:

   ```
   mkdir /workdir/OUDSM
   ```

2. Change the directory to this location:

   ```
   cd /workdir/OUDSM
   ```

   > **Note:**
   >
   > The same set of sample files are used by several products in this guide. To avoid having to download them each time, the files are staged in a non-product specific working directory.

3. Copy the sample scripts to your work directory.

   ```
   cp -R <WORKDIR>/fmw-kubernetes/OracleUnifiedDirectorySM /<WORKDIR>/samples
   ```

   For example:

   ```
   cp -R /workdir/OUDSM/fmw-kubernetes/OracleUnifiedDirectorySM /workdir/
   OUDSM/samples
   ```

# Creating a Kubernetes Namespace

You have to create a namespace to contain all the objects for Oracle Unified Directory Services Manager.

To create a namespace, run the following command:

```
kubectl create namespace oudsmns
```

The output appears as follows:

```
namespace/oudsmns created
```

# Creating a Container Registry Secret

Oracle recommends that you use a container registry. If you use a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

If you have staged your container images locally, there is no need to perform this step.

To create a container registry secret, use the following command:

```
kubectl create secret -n <OUDSMNS> docker-registry regcred --docker-
server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n oudsmns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

# Creating OUDSM Containers

Before you create the OUDSM containers, you should create the Helm override file. This section describes the commands you need to use to perform these tasks.

- Creating the Helm Overrides File
- Creating the Containers
- Troubleshooting the OUDSM Instances

## Creating the Helm Overrides File

The OUDSM containers are deployed using Helm. Create a Helm override file to inform how you want to deploy OUDSM.

The following is a sample override file called `/workdir/OUDSM/override_oudsm.yaml` for OUDSM:

```
image:
  repository: <OUDSM_REPOSITORY>
  tag: <OUDSM_VER>
  pullPolicy: IfNotPresent

imagePullSecrets:
  - name: regcred
oudsm:
  adminUser: <OUDSM_WEBLOGIC_USER>
  adminPass: <OUDSM_WEBLOGIC_PWD>
  startupTime: 200
persistence:
  type: networkstorage
  networkstorage:
    nfs:
      server: <PVSERVER>
      path: <OUDSM_SHARE>
  size: 5Gi
replicaCount: 1

elk:
  enabled: false
ingress:
  enabled: false
  type: nginx
  tlsEnabled: true
```

For example:

```
image:
  repository: iad.ocir.io/mytenancy/oudsm
  tag: 12.2.1.4-jdk8-ol7-220119.2054
  pullPolicy: IfNotPresent

imagePullSecrets:
  - name: regcred
oudsm:
  adminUser: weblogic
  adminPass: password
  startupTime: 200
persistence:
  type: networkstorage
  networkstorage:
    nfs:
      server: mynfsserver.example.com
      path:/export/IAMPVS/oudsmpv
  size: 5Gi
replicaCount: 1

elk:
  enabled: false
ingress:
```

```
  enabled: false
  type: nginx
  tlsEnabled: true
```

## Creating the Containers

After creating the overrides file, you can now create the container using the command:

```
cd /workdir/OUDSM/samples/kubernetes/helm
```

```
helm install --namespace <OUDSMNS> --values /workdir/OUDSM/
override_oudsm.yaml oudsm oudsm
```

The output appears as follows:

```
NAME: oudsm
LAST DEPLOYED: Thu Jan 28 08:08:40 2021
NAMESPACE: oudsmns
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

## Troubleshooting the OUDSM Instances

You can monitor the creation of each OUDSM instance using the following commands:

**Objects created in the namespace:**

```
kubectl -n oudsmns get all -o wide
```

The installation and configuration is complete only when you see each container with the status `READY 1/1` and `Status = Running`.

If you do not see objects being created, use the following command to check the issue:

```
kubectl get pod -n oudsmns
```

For a detailed description, use:

```
kubectl describe pod -n oudsmns
```

**Container Logs**

To view the progress of each container as it is being created, use the following command:

```
kubectl logs oudsm-1 -n oudsmns
```

# Creating External Access to OUDSM

By default, the Oracle Unified Directory Service Manager deployment gets created with all the components configured as ClusterIP services. This means that OUDSM is visible only within the Kubernetes cluster.

To gain access to OUDSM, you should expose it outside of the Kuberenetes cluster. You can do this in one of two ways:

- By using an Ingress controller
- By using a Kubernetes NodePort Service
- Creating the Kubernetes OUDSM NodePort Service
- Creating an Ingress Service for Oracle Unified Directory Services Manager

## Creating the Kubernetes OUDSM NodePort Service

You have to create an OUDSM NodePort Service to connect to OUDSM from outside the Kubernetes cluster.

1. Create a text file called `/workdir/OUDSM/oudsm_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: oudsm-nodeport
  namespace: <OUDSMNS>
spec:
  type: NodePort
  selector:
    app.kubernetes.io/instance: oudsm
    app.kubernetes.io/name: oudsm
  ports:
    - targetPort: 7001
      port: 7001
      nodePort: <OUDSM_SERVICE_PORT>
      protocol: TCP
```

For example:

```
kind: Service
apiVersion: v1
metadata:
  name: oudsm-nodeport
  namespace: oudsmns
spec:
  type: NodePort
  selector:
    app.kubernetes.io/instance: oudsm
    app.kubernetes.io/name: oudsm
  ports:
    - targetPort: 7001
```

```
    port: 7001
    nodePort: 30901
    protocol: TCP
```

> **Note:**
>
> Ensure that the namespace is set to the namespace you want to use.

2. Create the service using the following command:

```
kubectl create -f /workdir/OUDSM/oudsm_nodeport.yaml
```

The output appears as follows:

```
service/oudsm-nodeport created
```

3. Validate that you can access OUDSM by using the `http://k8worker1.example.com:30901/oudsm` URL.

# Creating an Ingress Service for Oracle Unified Directory Services Manager

To access OUDSM through Ingress, you should create an Ingress service. Create the Ingress service inside the product namespace. The Ingress service tells the Ingress controller how to direct requests inside the namespace.

To create an Ingress service:

1. Create a file called `oudsm_ingress.yaml` in the working directory, with the following contents:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: oudsm-ingress
  namespace: <OUDSMNS>
  annotations:
    nginx.ingress.kubernetes.io/affinity: "cookie"
    nginx.ingress.kubernetes.io/proxy-buffer-size: "2000k"
    nginx.ingress.kubernetes.io/enable-access-log: "false"
spec:
  ingressClassName: nginx
  rules:
  - host: <OUSDM_INGRESS_HOST>
    http:
      paths:
      - backend:
          service:
            name: oudsm-1
            port:
              number: 7001
        path: /oudsm
        pathType: Prefix
```

2. Create the Ingress service using the command:

```
kubectl create -f /workdir/OUDSM/oudsm_ingress.yaml
```

3. Validate that the Ingress service is been created correctly by using the following command:

```
kubectl get ingress -n oudsmns
```

# Configuring Oracle HTTP Server for Oracle Unified Directory Services Manager

It is not necessary to incorporate OUDSM into the Oracle HTTP server configuration. You can access OUDSM directly by using the Ingress or NodePort Services. However, if you want to incorporate OUDSM into the Oracle HTTP servers, you should perform the steps described in this section.

After you have configured your Oracle HTTP server as described in Installing and Configuring Oracle HTTP Server, you can configure the Oracle HTTP Server to route requests to the Oracle Unified Directory Services Manager.

To configure the Oracle HTTP Server:

1. Add the following entries to the `iadadmin_vh.conf` or `igdadmin_vh.conf` files located at *`WEB_DOMAIN_HOME`*`/config/fmwconfig/components/OHS/ohs1/moduleconf/`:

```
<Location /oudsm>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster k8worker1.example.com:30901,
k8worker2.example.com:30901
</Location>
```

> **Note:**
>
> - There are separate directories for configuration and runtime instance files. The runtime files under the `.../OHS/instances/ohsn/*` folder should not be edited directly. Edit only the `.../OHS/ohsn/*` configuration files.
> - `iadadmin_vh.conf` and `igdadmin_vh.conf` will only be available after you have configured Oracle Access Manager or Oracle Identity Governance.
> - If you are using an Ingress controller, the port should be the HTTP port assigned to the Ingress Controller. For example: `30777`.
> - If you are using an Ingress controller, ensure that you add the directive into the OHS virtual host file that corresponds to the ingress host name.

2. Copy the `igdadmin_vh.conf` or `iadadmin_vh.conf` file to the following configuration directory of the second Oracle HTTP Server instance (ohs2):

   *`OHS_DOMAIN_HOME`*`/config/fmwconfig/components/ohs2/moduleconf/`

3. Restart the Oracle HTTP server instances on WEBHOST1 and WEBHOST2.

# Centralized Log File Monitoring Using Elasticsearch and Kibana

If you are using Elasticsearch and Kibana, you can configure a Logstash pod to send the log files to the centralized Elasticsearch/Kibana console. Before you configure the Logstash pod, ensure that you have access to a centralized Elasticsearch deployment.

- OUDSM persistent volume, so it can be loaded by the Logstash pod to hunt for log files.
- The location of the log files in the persistent volumes.
- The location of the centralized Elasticsearch.

To configure the Logstash pod, perform the following steps. The assumption is that you have an Elasticsearch running inside the Kubernetes cluster, in a namespace called `elkns`.

- Creating a Secret for Elasticsearch
- Creating a Configuration Map for ELK Certificate
- Configuring Log File Monitoring for OUDSM

## Creating a Secret for Elasticsearch

Logstash requires credentials to connect to the elasticsearch deployment. These credentials are stored in Kubernetes as a secret.

If your Elasticsearch uses an API key for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OUDSMNS> --from-
literal password=<ELK_APIKEY>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oudsmns --from-
literal password=afshfashfkahf5f
```

If Elasticsearch uses a user name and password for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OUDSMNS> --from-
literal password=<ELK_PWD>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oudsmns --from-
literal password=mypassword
```

You can find the Elasticsearch password using the following command:

```
kubectl get secret elasticsearch-es-elastic-user -n <ELKNS> -o go-
template='{{.data.elastic | base64decode}}'
```

# Creating a Configuration Map for ELK Certificate

If you have configured a production ready Elasticsearch deployment, you would have configured SSL. Logstash needs to trust the Elasticsearch certificate to be able to communicate with it. To enable this trust, you should create a configuration map with the contents of the Elasticsearch certificate.

You would have already saved the Elasticsearch self-signed certificate. See Copying the Elasticsearch Certificate. If you have a production certificate you can use that instead.

Create the configuration map using the certificate, run the following command:

```
kubectl create configmap elk-cert --from-file=<WORKDIR>/ELK/elk.crt -n
<OUDSMNS>
```

For example:

```
kubectl create configmap elk-cert --from-file=/workdir/ELK/elk.crt -n oudsmns
```

# Configuring Log File Monitoring for OUDSM

Complete the following steps to configure log file monitoring:

- Creating a Configuration Map for Logstash
- Creating a Logstash Deployment

## Creating a Configuration Map for Logstash

Logstash looks for log files in the OUDSM installations and sends them to the centralized Elasticsearch. The configuration map is used to instruct Logstash where the log files reside and where to send them.

1. Create a file called `<WORKDIR>/OUDSM/logstash_cm.yaml` with the following contents:

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: oudsm-logstash-configmap
 namespace: <OUDSMNS>
data:
 logstash.yaml: |
 #http.host: "0.0.0.0"
 logstash-config.conf: |
   input {
      file {
        path => "/u01/oracle/user_projects/domains/oudsmdomain-1/servers/
AdminServer/logs/*.log"
        type => "setup-logs"
        start_position => beginning
        sincedb_path => "/dev/null"
      }
    }
    filter {
```

```
      if [type] == "setup-logs" {
        grok {
          match => [ "message", "<%{DATA:log_timestamp}> <%
{WORD:log_level}> <%{WORD:thread}> <%{HOSTNAME:hostname}> <%
{HOSTNAME:hostserver}> %{GREEDYDATA:message}" ]
          }
      }
      if "_grokparsefailure" in [tags] {
        mutate {
          remove_tag => [ "_grokparsefailure" ]
          }
        }
      }
   output {
    elasticsearch {
     hosts => ["<ELK_HOST>"]
     cacert => '/usr/share/logstash/config/certs/elk.crt'
     user => "<ELK_USER>"
     password => "<ELK_USER_PWD>"
     index => "oudsmlogs-000001"
     ssl => true
     ssl_certificate_verification => false
    }
   }
```

2. Save the file.
3. Create the configuration map using the following command:

   ```
   kubectl create -f <WORKDIR>/OUDSM/logstash_cm.yaml
   ```

   For example:

   ```
   kubectl create -f /workdir/OUDSM/logstash_cm.yaml
   ```

4. Validate that the configuration map has been created by using the following command:

   ```
   kubectl get cm -n <OUDSMNS>
   ```

   You should see `oudsm-logstash-configmap` in the list of configuration maps.

## Creating a Logstash Deployment

After you create the configuration map, you can create the Logstash deployment. This deployment resides in the OUD namespace.

1. Create a file called `<WORKDIR>/OUDSM/logstash.yaml` with the following contents:

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: oudsm-logstash
     namespace: <OUDSMNS>
   spec:
     selector:
   ```

```
    matchLabels:
      k8s-app: logstash
  template: # create pods using pod definition in this template
    metadata:
      labels:
        k8s-app: logstash
    spec:
      imagePullSecrets:
      - name: dockercred
      containers:
      - command:
        - logstash
        image: logstash:<ELK_VER>
        imagePullPolicy: IfNotPresent
        name: oudsm-logstash
        env:
        - name: ELASTICSEARCH_PASSWORD
          valueFrom:
            secretKeyRef:
              name: elasticsearch-pw-elastic
              key: password
        ports:
        - containerPort: 5044
          name: logstash
        volumeMounts:
        - mountPath: /u01/oracle/user_projects
          name: oudsm-storage-volume
        - name: shared-logs
          mountPath: /shared-logs
        - mountPath: /usr/share/logstash/pipeline/
          name: oudsm-logstash-pipeline
        - mountPath: /usr/share/logstash/config/certs
          name: elk-cert
      volumes:
      - configMap:
          defaultMode: 420
          items:
          - key: logstash-config.conf
            path: logstash-config.conf
          name: oudsm-logstash-configmap
        name: oudsm-logstash-pipeline
      - configMap:
          defaultMode: 420
          items:
          - key: ca.crt
            path: elk.crt
          name: elk-cert
      - name: oudsm-storage-volume
        persistentVolumeClaim:
          claimName: oudsm-pvc
      - name: shared-logs
        emptyDir: {}
```

> **Note:**
>
> If you are using your own registry, include the registry name in the image tag. If you have created a `regcred` secret for your registry, replace the `imagePullSecrets` name with the secret name you created. For example: `regcred`.

2. Save the file.
3. Create the Logstash deployment by using the following command:

   ```
   kubectl create -f <WORKDIR>/OUDSM/logstash.yaml
   ```

   For example:

   ```
   kubectl create -f /workdir/OUDSM/logstash.yaml
   ```

4. You can now create a pod called `logstash` by using the following command:

   ```
   kubectl get pod -n oudsmns
   ```

   Your logs will now be available in the Kibana console.

16

# Installing and Configuring WebLogic Kubernetes Operator

The WebLogic Operator for Kubernetes facilitates the creation and management of WebLogic domains in a Kubernetes cluster.

The WebLogic Operator can manage several different domains in different namespaces. The WebLogic Operator for Kubernetes is installed in its own dedicated namespace.

This chapter includes the following topics:

- Setting Up a Product Specific Work Directory
  Before you begin the installation, you must have downloaded and staged the Oracle Access Manager container image and the sample code repository.
- Variables Used in this Chapter
  The later sections of this chapter provide instructions to create a files. These sample files contain variables which you need to substitute with values applicable to your deployment.
- Removing Existing Custom Resource Definitions
  Remove any custom resource definitions for WebLogic, if they exist.
- Installing the WebLogic Kubernetes Operator
  The procedure to install the WebLogic Kubernetes Operator consists of creating a namespace and a Kubernetes service account. Start the operator after the installation.

## Setting Up a Product Specific Work Directory

Before you begin the installation, you must have downloaded and staged the Oracle Access Manager container image and the sample code repository.

See Identifying and Obtaining Software Distributions for an Enterprise Deployment.

You must also have deployed the Oracle WebLogic Operator as described in Installing the WebLogic Kubernetes Operator.

This section describes the procedure to copy the downloaded sample deployment scripts to a temporary working directory on the configuration host for OAM.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubenetes cluster.

   ```
   mkdir -p <WORKDIR>
   ```

   For example:

   ```
   mkdir -p /workdir/OPER
   ```

2. Change directory to this location:

   ```
   cd /workdir/OPER
   ```

> **Note:**
>
> The same set of sample files are used by several products in this guide. To avoid having to download them each time, the files are staged in a non-product specific working directory.
>
> The WebLogic Operator files are identical for Oracle Access Manager and Oracle Identity Governance. Therefore, you can use any version from the samples.

3. Copy the sample scripts to the work directory.

```
cp -R <WORKDIR>/fmw-kubernetes/OracleAccessManagement/kubernetes <WORKDIR>/
samples
```

For example:

```
cp -R /workdir/OPER/fmw-kubernetes/OracleAccessManagement/kubernetes /
workdir/OPER/samples
```

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 16-1 The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| <*WORKDIR*> | `/workdir/OPER` | The location where you want to create the working directory for the Kubernetes Operator. |
| <*OPERNS*> | `opns` | The Kubernetes namespace to hold the Operator objects. |
| <*OPER_VER*> | `4.1.8` | The version of Kubernetes Operator. |
| <*OPER_ACT*> | `operadmin` | The service account for Kubernetes Operator. |
| <*USE_ELK*> | `false` | Set to `true` if you are using Elasticsearch/Kibana monitoring. |
| <*ELK_PROTO*> | `https` | The `http` or `https` protocol used to access the Elasticsearch cluster. |
| <*ELK_HOST*> | `elasticsearch-es-http.elkns.svc` | The host to which you want to send the Elasticsearch logs. This can be inside the Kubernetes cluster or external to it. |
| <*ELK_PORT*> | `9200` | The Elasticsearch port used to receive the log information. |

**Table 16-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<ELK_VER>* | `8.11.0` | The version of Elasticsearch you want to use. |
| *<REGISTRY_SECRET_NAME>* | `regcred` | The name of the Kubernetes secret containing the container registry credentials. Required only if you are pulling images directly from a container registry. See Creating a Container Registry Secret. |
| *<REGISTRY_ADDRESS>* | `iad.ocir.io/<mytenancy>` | The location of the registry. If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/oud_cpu`. |
| *<REG_USER>* | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The user id you use to log in to the registry. If you are use the Oracle container registry, this value will be your Oracle single sign-on user name. |
| *<REG_PWD>* | *<password>* | The registry user password. |

# Removing Existing Custom Resource Definitions

Remove any custom resource definitions for WebLogic, if they exist.

Use the following commands to determine if a custom resource definition exists, and if so to delete it:

```
kubectl get crd
```

```
NAME                                  AGE
domains.weblogic.oracle               5d
```

```
kubectl delete crd domains.weblogic.oracle
```

```
customresourcedefinition.apiextensions.k8s.io "domains.weblogic.oracle"
deleted
```

# Installing the WebLogic Kubernetes Operator

The procedure to install the WebLogic Kubernetes Operator consists of creating a namespace and a Kubernetes service account. Start the operator after the installation.

- Creating a Namespace
- Creating a Container Registry Secret

## Creating a Namespace

Create a namespace to contain all the WebLogic Operator Kubernetes objects.

```
kubectl create namespace <OPERNS>
```

For example:

```
kubectl create namespace opns
```

The output will look similar to the following:

```
namespace/opns created
```

## Creating a Container Registry Secret

If you are using your own container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

If you have staged your container images locally or if you are pulling them from a public repository there is no need to run the following command.

Use the following command to create a container registry secret:

```
kubectl create secret -n <OPERNS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For Example:

```
kubectl create secret -n opns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

## Creating a Kubernetes Service Account

Create a Kubernetes Service Account for the WebLogic Operator. This account is created inside the namespace.

```
kubectl create serviceaccount -n <OPERNS> <OPER_ACT>
```

For example:

```
kubectl create serviceaccount -n opns operadmin
```

The output will look similar to the following:

```
serviceaccount/op-sa created
```

## Creating a Secret for Elasticsearch

If you are using Elasticsearch and your Elasticsearch requires connections to use SSL, then you must place the elasticsearch certificate in a Kubernetes secret.

If you are using Elasticsearch in Kubernetes, then you can obtain the certificate by using the following command:

```
kubectl cp <ELKNS>/elasticsearch-es-default-0:/usr/share/elasticsearch/config/
http-certs/..data/ca.crt <WORKDIR>/ca.crt
```

If you are not using Elasticsearch inside Kubernetes, then you must copy the `ca.crt` file from your Elasticsearch installation to your working directory.

To create a secret from the Elasticsearch certificate, use the following command:

```
kubectl create secret generic logstash-certs-secret --from-file=<WORKDIR>/
ca.crt -n $<OPERNS>
```

## Installing and Starting the WebLogic Operator

To install and start the WebLogic Operator:

1. Use the following command:

```
cd <WORKDIR>/samples
```

```
helm install kubernetes/charts/weblogic-operator \
 --namespace <OPERNS> \
 --set image=weblogic-kubernetes-operator:<OPER_VER> \
 --set serviceAccount=<OPER_ACT> \
 --set "enableClusterRoleBinding=true" \
 --set "domainNamespaceSelectionStrategy=LabelSelector" \
 --set "domainNamespaceLabelSelector=weblogic-operator\=enabled" \
 --set "elasticSearchProtocol=<ELK_PROTO>" \
 --set "elkIntegrationEnabled=<USE_ELK>" \
 --set "elasticSearchHost=<ELK_HOST>" \
 --set "elasticSearchPort=<ELK_PORT>" \
 --set "logStashImage=docker.elastic.co/logstash/logstash:<ELK_VER>" \
 --set "createLogStashConfigMap=true" \
 --wait
```

**Note:**

- The image name includes the name of the repository. For example, if you use a container registry, it will have appear as `registry/weblogic-kubernetes-operator`. If you use Oracle Cloud Native Environment, the name will appear as `local/weblogic-kubernetes-operator`.
- If you use locally staged images, you can use the `podman images` or `docker images` commands to determine the full name.
- The ELK/Elasticsearch parameters are required only if you are using Elasticsearch/Kibana Monitoring.
- The `createLogStashConfigMap` parameter will work only with WebLogic Operator release 4.0+. The WebLogic Operator log files can sent to a secured Elasticsearch only when using WebLogic Operator release 4.0+. With earlier releases of WebLogic Operator, log files can be sent only to a fully open Elasticsearch.
- If you are using your own container registry, then you should add the additional argument for your registry secret:

  ```
  --set "imagePullSecrets[0].name=regcred"
  ```

For example:

```
cd /workdir/OPER/samples
```

```
helm install weblogic-kubernetes-operator charts/weblogic-operator \
 --namespace opns \
 --set image=weblogic-kubernetes-operator:4.1.8 \
 --set serviceAccount=operadmin \
 --set "enableClusterRoleBinding=true" \
 --set "domainNamespaceSelectionStrategy=LabelSelector" \
 --set "domainNamespaceLabelSelector=weblogic-operator\=enabled" \
 --set "elkIntegrationEnabled=true" \
 --set "elasticSearchHost=https://elasticsearch-es-http.elkns.svc" \
 --set "elasticSearchPort=9200" \
 --set "logStashImage=docker.elastic.co/logstash/logstash:8.11.0" \
 --set "createLogStashConfigMap=true" \
 --wait
```

The output appears as follows:

```
NAME: weblogic-kubernetes-operator
 LAST DEPLOYED: Wed Sep 23 08:04:20 2020
 NAMESPACE: opns
 STATUS: deployed
 REVISION: 1
 TEST SUITE: None
```

2. Verify that the operator's pod and services are running by using the following command:

```
kubectl get all -n opns
```

The output appears as follows:

```
NAME                                                READY      STATUS
RESTARTS    AGE
pod/weblogic-operator-759b7c657-8gd7g               2/2        Running
0           107s
pod/weblogic-operator-webhook-859b5755b6-2zlq8      2/2        Running
0           24m

NAME                                          TYPE        CLUSTER-IP
EXTERNAL-IP    PORT(S)    AGE
service/internal-weblogic-operator-svc        ClusterIP   10.102.11.143
<none>         8082/TCP   107s

NAME                                          READY   UP-TO-DATE
AVAILABLE     AGE
deployment.apps/weblogic-operator             1/1     1
1             107s

NAME                                           DESIRED   CURRENT   READY
AGE
replicaset.apps/weblogic-operator-759b7c657    1         1         1
107s
```

3. Verify the operator's pod log:

```
kubectl logs -n opns weblogic-operator-759b7c657-8gd7g
```

The output appears as follows:

```
{"timestamp":"09-23-2020T15:04:30.485+0000","thread":28,"fiber":"fiber-1","
namespace":"opns","domainUID":"","level":"INFO","class":"oracle.kubernetes.
operator.rest.RestServer",ethod":"start","timeInMillis":1600873470485,"mess
age":"Started the internal ssl REST server on https://0.0.0.0:8082/
operator","exception":"","code":"","headers":{},"body":""}
{"timestamp":"09-23-2020T15:04:30.487+0000","thread":28,"fiber":"fiber-1","
namespace":"opns","domainUID":"","level":"INFO","class":"oracle.kubernetes.
operator.Main","method":"mkReadyAndStartLivenessThread","timeInMillis":1600
873470487,"message":"Starting Operator Liveness
Thread","exception":"","code":"","headers":{},"body":""}
{"timestamp":"09-23-2020T15:06:27.528+0000","thread":22,"fiber":"engine-
operator-thread-5-
fiber-2","namespace":"opns","domainUID":"","level":"FINE","class":"oracle.k
ubernetes.orator.helpers.ConfigMapHelper$ScriptConfigMapContext","method":"
loadScriptsFromClasspath","timeInMillis":1600873587528,"message":"Loading
scripts into domain control config mapor namespace:
opns","exception":"","code":"","headers":{},"body":""}
{"timestamp":"09-23-2020T15:06:27.529+0000","thread":22,"fiber":"engine-
operator-thread-5-
```

```
fiber-2","namespace":"opns","domainUID":"","level":"FINE","class":"oracle.k
ubernetes.orator.Main","method":"readExistingDomains","timeInMillis":160087
3587529,"message":"Listing WebLogic
Domains","exception":"","code":"","headers":{},"body":""}
{"timestamp":"09-23-2020T15:06:27.576+0000","thread":20,"fiber":"fiber-2-
child-1","namespace":"opns","domainUID":"","level":"FINE","class":"oracle.k
ubernetes.operator.helpers.CfigMapHelper$ConfigMapContext$ReadResponseStep"
,"method":"logConfigMapExists","timeInMillis":1600873587576,"message":"Exis
ting config map, ConfigMapHelper$ConfigMapContext$Readsponse, is correct
for namespace: opns.","exception":"","code":"","headers":{},"body":""}
```


# Updating the Elasticsearch Configuration

When the WebLogic Kubernetes Operator is deployed, it creates a configuration map called `weblogic-operator-logstash-cm`. This configuration map contains the details of how Logstash sends logs to the Elasticsearch cluster. This configuration map requires modifications for sending the logs successfully.

1. Edit the configuration map using the following command:

   ```
   kubectl edit cm -n <OPERNS> weblogic-operator-logstash-cm
   ```

2. Look for the output section and uncomment the lines as appropriate, and add values. Typically, the output section should appear as follows:

   ```
       output {
         elasticsearch {
           hosts => ["${ELASTICSEARCH_PROTOCOL}://${ELASTICSEARCH_HOST}:$
   {ELASTICSEARCH_PORT}"]
           # Example configuration if Elasticsearch instance requires
   authentication and SSL:
           ssl => true
           user => "logstash_internal"
           password => "<ELK_USER_PWD>"
           cacert => '/usr/share/logstash/config/certs/ca.crt'
           index => "wkologs-000001"
         }
         stdout { codec => rubydebug }
       }
   ```

3. Save the configuration map.
4. Restart the WebLogic Kubernetes Operator by deleting the pods. For example:

   ```
   kubectl delete pod -n opns weblogic-operator-759b7c657-8gd7g


   kubectl delete pod -n opns weblogic-operator-webhook-859b5755b6-2zlq8
   ```

5. Verify the Logstash's pod log by using the following command:

   ```
   kubectl logs -n opns weblogic-operator-759b7c657-8gd7g -c logstash
   ```

17

# Installing and Configuring Oracle HTTP Server

For an enterprise deployment, Oracle HTTP Server must be installed on each of the web tier hosts and configured as Oracle HTTP standalone domains on each host.

The Oracle HTTP Server instances on the web tier direct HTTP requests from the hardware load balancer to specific Managed Servers in the application tier.

Before you configure Oracle HTTP Server, be sure to review About Web Tier.

This chapter includes the following topics:

- Variables Used When Configuring the Oracle HTTP Server
  You reference these directory variables as you perform the different tasks explained in this chapter.
- About Storage
  When you deploy Oracle HTTP Servers, the configuration information is stored locally or on a dedicated NFS volume.
- About the Oracle HTTP Server Domains
  In an enterprise deployment, each Oracle HTTP Server instance is configured on a separate host and in its own standalone domain. This allows for a simple configuration that requires a minimum amount of configuration and a minimum amount of resources to run and maintain.
- Installing a Supported JDK
  Oracle Fusion Middleware requires you to install a certified Java Development Kit (JDK) on your system.
- Installing Oracle HTTP Server on WEBHOST1
  Install the Oracle HTTP Server software on the web tier by using the Oracle Universal Installer. Verify the installation after you complete the procedure.
- Creating an Oracle HTTP Server Domain on WEBHOST1
  You can create a new Oracle HTTP Server standalone domain on the first web tier host by using the Configuration Wizard.
- Installing and Configuring an Oracle HTTP Server Domain on WEBHOST2
  After you install Oracle HTTP Server and configure a domain on WEBHOST1, then you must also perform the same tasks on WEBHOST2.
- Starting the Node Manager and Oracle HTTP Server Instances on WEBHOST1 and WEBHOST2
  Start the Node Manager on both the hosts before starting the Oracle HTTP Server instances.
- Creating a Health Check
  Create a health check on each Oracle HTTP Server instance. Oracle recommends using a specific page for health checks to avoid failures.
- Backing Up the Configuration
  As a best practice, Oracle recommends you to back up the configuration after you have successfully extended a domain or at another logical point. Back up only after you have verified that the installation is successful so far. This is a quick backup to enable immediate restoration in case of problems in later steps.

- Configuring Oracle HTTP Server to Route Requests to the Application Tier
  Update the Oracle HTTP Server configuration files so that the web server instances route requests to the servers in the domain.
- Configuring Oracle HTTP Server for Oracle Access Manager
  You have to configure Oracle HTTP Server for the Oracle Access Manager Managed Servers to ensure they route requests correctly to the Oracle Access Management cluster.
- Configuring Oracle HTTP Server for Oracle Identity Governance
  To configure the Oracle HTTP Server instances in the web tier so they route requests correctly to the Oracle SOA Suite cluster, use the following procedure to create an additional Oracle HTTP Server configuration file that creates and defines the parameters of the `https://igdinternal.example.com:7777` virtual server.
- Configuring Oracle HTTP Server for Oracle Identity Role Intelligence
  You should configure Oracle HTTP Server for the Oracle Identity Role Intelligence (OIRI) Servers to ensure that they route requests correctly to the Oracle Role Intelligence cluster.
- Configuring Oracle HTTP Server for Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator
  You should configure Oracle HTTP Server for Oracle Advanced Authentication servers to ensure that they route requests correctly to the OAA microservices.
- Restarting the OHS Instances
  Ensure that you have copied the configuration files to each WEBHOST, and then restart the Oracle HTTP Service instance on each host.
- Validating the Oracle HTTP Server Configuration
  To ensure that the Oracle HTTP server is working correctly, you should perform a few validations after configuring the Oracle Identity Management products.
- Sample Virtual Host Files
  The sample list includes the complete examples of all the virtual host files used in an Oracle Identity and Access Management deployment.

# Variables Used When Configuring the Oracle HTTP Server

You reference these directory variables as you perform the different tasks explained in this chapter.

The values for several directory variables are defined in File System and Directory Variables Used in This Guide.

- *WEB_ORACLE_HOME*
- *WEB_DOMAIN_HOME*
- *JAVA _HOME*

# About Storage

When you deploy Oracle HTTP Servers, the configuration information is stored locally or on a dedicated NFS volume.

In the sections below, local refers to either local storage or an NFS volume. If you want to deploy Oracle HTTP Server on OCI, you should create a dedicated NFS volume. Whenever you see the reference `/private`, it refers to this private storage area – NFS or local.

# About the Oracle HTTP Server Domains

In an enterprise deployment, each Oracle HTTP Server instance is configured on a separate host and in its own standalone domain. This allows for a simple configuration that requires a minimum amount of configuration and a minimum amount of resources to run and maintain.

> **Note:**
>
> Oracle Fusion Middleware requires that a certified Java Development Kit (JDK) is installed on your system and JAVA_HOME is set on the web tier hosts.

For more information about the role and configuration of the Oracle HTTP Server instances in the web tier, see Understanding the Web Tier.

# Installing a Supported JDK

Oracle Fusion Middleware requires you to install a certified Java Development Kit (JDK) on your system.

The installation should be performed by `oracle user` who has the required permissions to install and configure the software. See Creating a Software Owner Account.

- Locating and Downloading the JDK Software
- Installing the JDK Software

## Locating and Downloading the JDK Software

To find a certified JDK, see the certification document for your release on the Oracle Fusion Middleware Supported System Configurations page.

After you identify the Oracle JDK for the current Oracle Fusion Middleware release, you can download an Oracle JDK from the following location on Oracle Technology Network:

```
http://www.oracle.com/technetwork/java/index.html
```

Be sure to navigate to the download for the Java SE JDK.

## Installing the JDK Software

You must install the JDK in the following locations:

On the local storage device for each of the Web tier host computers. The Web tier host computers, which reside in the DMZ, do not necessarily have access to the shared storage on the application tier.

See the Understanding the Recommended Directory Structure for an Enterprise Deployment.

To install JDK 1.8.0_211:

1. Change directory to the location where you downloaded the JDK archive file.

   ```
   cd download_dir
   ```

2. Unpack the archive into the JDK home directory, and then run the following commands:

```
tar -xzvf jdk-8u201-linux-x64.tar.gz
```

Note that the JDK version listed here was accurate at the time this document was published. For the latest supported JDK, see the *Oracle Fusion Middleware System Requirements and Specifications* for the current Oracle Fusion Middleware release.

3. Move the JDK directory to the recommended location in the directory structure.

   For example:

   ```
   mv ./jdk1.8.0_211 /u02/oracle/products/jdk
   ```

   See File System and Directory Variables Used in This Guide.

4. Define the *JAVA_HOME* and *PATH* environment variables for running Java on the host computer.

   For example:

   ```
   export JAVA_HOME=/u02/oracle/products/jdk
   export PATH=$JAVA_HOME/bin:$PATH
   ```

5. Run the following command to verify that the appropriate `java` executable is in the path and your environment variables are set correctly:

   ```
   java -version
   ```

   The Java version in the output should be displayed as `1.8.0_211`.

# Installing Oracle HTTP Server on WEBHOST1

Install the Oracle HTTP Server software on the web tier by using the Oracle Universal Installer. Verify the installation after you complete the procedure.

The installation should be performed by `oracle user` who has the required permissions to install and configure the software. See Creating a Software Owner Account.

The installation should be performed by `oracle user` who has the required permissions to install and configure the software. See Creating a Software Owner Account.

- Starting the Installer on WEBHOST1
- Navigating the Oracle HTTP Server Installation Screens
- Verifying the Oracle HTTP Server Installation

## Starting the Installer on WEBHOST1

To start the installation program, perform the following steps.

1. Log in to WEBHOST1.
2. Go to the directory in which you downloaded the installation program.
3. Enter the following command to launch the installation program:

   ```
   ./fmw_12.2.1.4.0_ohs_linux64.bin
   ```

   When the installation program appears, you are ready to begin the installation.

> **Note:**
>
> If you are installing on Oracle Linux 8 and the prerequisite checks fail because the following packages do not exist:
>
> - `compat-libcap1-1.10`
> - `compat-libstdc++-33-3.2.3-x86_64`
>
> Then, you can ignore them and carry on. These packages do not exist in Oracle Linux 8.
>
> To continue, you can start the installation using the following command:
>
> ```
> ./fmw_12.2.1.4.0_ohs_linux64.bin -ignoreSysPrereqs
> ```

# Navigating the Oracle HTTP Server Installation Screens

The following table lists the screens in the order that the installation program displays them.

If you need additional help with any of the installation screens, click the Help button on the screen.

**Table 17-1 Oracle HTTP Server Installation Screens**

| Screen | Description |
|---|---|
| Installation Inventory Setup | On UNIX operating systems, this screen appears if you install any Oracle product on this host for the first time. Specify the location where you want to create your central inventory. Ensure that the operating system group name selected on this screen has write permissions to the central inventory location.<br>See Understanding the Oracle Central Inventory in *Installing Software with the Oracle Universal Installer*.<br>**Note:**<br>Oracle recommends that you configure the central inventory directory within the products directory. Example: `/u02/oracle/products/oraInventory`<br>You may also need to execute the `createCentralinventory.sh` script as root from the `oraInventory` folder after the installer completes. |

**Table 17-1 (Cont.) Oracle HTTP Server Installation Screens**

| Screen | Description |
|---|---|
| Welcome | This screen introduces you to the product installer. |
| Auto Updates | Use this screen to automatically search My Oracle Support for available patches or automatically search the local directory for patches that you have already downloaded for your organization. |
| Installation Location | Use this screen to specify the location of your Oracle home directory.<br>For the purposes of an enterprise deployment, enter the value of the *WEB_ORACLE_HOME* variable listed in Table 4-5. |
| Installation Type | Select **Standalone HTTP Server (Managed independently of WebLogic server)**.<br>This installation type allows you to configure the Oracle HTTP Server instances independently from any other existing Oracle WebLogic Server domains. |
| JDK Selection | For the value of JDK Home, enter the value of *JAVA_HOME* that you set when installing the JDK software. |
| Prerequisite Checks | This screen verifies that your system meets the minimum necessary requirements.<br>If there are any warning or error messages, verify that your host computers and the required software meet the system requirements and certification information described in Host Computer Hardware Requirements and Operating System Requirements for the Enterprise Deployment Topology. |
| Installation Summary | Use this screen to verify the installation options that you selected. If you want to save these options to a response file, click **Save Response File** and provide the location and name of the response file. Response files can be used later in a silent installation situation.<br>See Using the Oracle Universal Installer in Silent Mode in *Installing Software with the Oracle Universal Installer*. |
| Installation Progress | This screen allows you to see the progress of the installation. |
| Installation Complete | This screen appears when the installation is complete. Review the information on this screen, then click **Finish** to close the installer. |

# Verifying the Oracle HTTP Server Installation

Verify that the Oracle HTTP Server installation completed successfully by validating the `WEB_ORACLE_HOME` folder contents.

Run the following command to compare the installed folder structure with the following list:

```
ls --format=single-column WEB_ORACLE_HOME
```

The following files and directories are listed in theOracle HTTP Server Oracle Home:

```
bin
cdata
cfgtoollogs
```

```
crs
css
cv
has
install
inventory
jlib
ldap
lib
network
nls
ohs
OPatch
oracle_common
oracore
oraInst.loc
oui
perl
plsql
plugins
precomp
QOpatch
racg
rdbms
slax
sqlplus
srvm
webgate
wlserver
xdk
```

# Creating an Oracle HTTP Server Domain on WEBHOST1

You can create a new Oracle HTTP Server standalone domain on the first web tier host by using the Configuration Wizard.

- Starting the Configuration Wizard on WEBHOST1
- Navigating the Configuration Wizard Screens for an Oracle HTTP Server Domain

## Starting the Configuration Wizard on WEBHOST1

To start the Configuration Wizard, navigate to the following directory and start the WebLogic Server Configuration Wizard, as follows:

```
cd WEB_ORACLE_HOME/oracle_common/common/bin
./config.sh
```

## Navigating the Configuration Wizard Screens for an Oracle HTTP Server Domain

Oracle recommends that you create a standalone domain for the Oracle HTTP Server instances on each web tier host.

The following topics describe how to create a new standalone Oracle HTTP Server domain:

- Task 1, Selecting the Domain Type and Domain Home Location
- Task 2, Selecting the Configuration Templates
- Task 3, Selecting the JDK for the Web Tier Domain.
- Task 4, Configuring System Components
- Task 5, Configuring OHS Server
- Task 7, Reviewing Your Configuration Specifications and Configuring the Domain
- Task 8, Writing Down Your Domain Home

**Task 1 Selecting the Domain Type and Domain Home Location**
On the Configuration Type screen, select **Create a new domain**.
In the **Domain Location** field, enter the value assigned to the *WEB_DOMAIN_HOME* variable.
Note the following:

- The Configuration Wizard creates the new directory that you specify here.
- Create the directory on local storage, so the web servers do not have any dependencies on storage devices outside the DMZ.

**Task 2 Selecting the Configuration Templates**
On the Templates screen, select **Oracle HTTP Server (Standalone) - 12.2.1.4.0 [ohs]**.

> **Tip:**
>
> More information about the options on this screen can be found in Templates in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard*.

**Task 3 Selecting the JDK for the Web Tier Domain.**
Select the Oracle HotSpot JDK installed in the `/u02/oracle/products/jdk` directory prior to the Oracle HTTP Server installation.

**Task 4 Configuring System Components**
On the System Components screen, configure one Oracle HTTP Server instance. The screen should, by default, have a single instance defined. This is the only instance that you need to create.

1. The default instance name in the **System Component** field is `ohs1`. Use this default name when you configure `WEBHOST1`.
2. Make sure that `OHS` is selected in the **Component Type** field.
3. If an application is not responding, use the **Restart Interval Seconds** field to specify the number of seconds to wait before you attempt a restart if an application is not responding.
4. Use the **Restart Delay Seconds** field to specify the number of seconds to wait between restart attempts.

**Task 5 Configuring OHS Server**
Use the OHS Server screen to configure the OHS servers in your domain:

1. Select **ohs1** from the **System Component** drop-down menu.

2. In the **Listen Address** field, enter *WEBHOST1*.

   All the remaining fields are prepopulated, but you can change the values as required for your organization. See OHS Server in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard*.

3. In the **Server Name** field, verify the value of the listen address and listen port.

   It should appear as follows:

   ```
   http://WEBHOST1:7777
   ```

**Task 6 Configuring Node Manager**
Select **Per Domain Default Location** as the Node Manager type, and specify the user name and password for the Node Manager.

> **Note:**
>
> For more information about the options on this screen, see Node Manager in *Creating WebLogic Domains Using the Configuration Wizard*.
> For information about Node Manager configuration, see Configuring Node Manager on Multiple Machines in *Administering Node Manager for Oracle WebLogic Server*.

**Task 7 Reviewing Your Configuration Specifications and Configuring the Domain**
The Configuration Summary screen contains detailed configuration information for the domain that you are about to create. Review the details of each item on the screen and verify that the information is correct.
If you need to make any changes, you can go back to any previous screen either by using the **Back** button or by selecting the screen in the navigation pane.
Domain creation does not begin until you click **Create**.
In the Configuration Progress screen, click **Next** when it finishes.

> **Tip:**
>
> More information about the options on this screen can be found in Configuration Summary in *Creating WebLogic Domains Using the Configuration Wizard*.

**Task 8 Writing Down Your Domain Home**
The Configuration Success screen shows the domain home location.
Make a note of the information provided here, as you need it to start the servers and access the Administration Server.
Click **Finish** to close the Configuration Wizard.

# Installing and Configuring an Oracle HTTP Server Domain on WEBHOST2

After you install Oracle HTTP Server and configure a domain on WEBHOST1, then you must also perform the same tasks on WEBHOST2.

1. Log in to WEBHOST2 and install Oracle HTTP Server by using the instructions in Installing Oracle HTTP Server on WEBHOST1.

2. Configure a new standalone domain on WEBHOST2 by using the instructions in Creating a Web Tier Domain on WEBHOST1.

   Use the name `ohs2` for the instance on WEBHOST2, and be sure to replace all occurrences of WEBHOST1 with WEBHOST2 and all occurrences of `ohs1` with `ohs2` in each of the examples.

# Starting the Node Manager and Oracle HTTP Server Instances on WEBHOST1 and WEBHOST2

Start the Node Manager on both the hosts before starting the Oracle HTTP Server instances.

- Starting the Node Manager on WEBHOST1 and WEBHOST2
- Starting the Oracle HTTP Server Instances

## Starting the Node Manager on WEBHOST1 and WEBHOST2

Before you can start the Oracle HTTP Server instances, you must start the Node Manager on WEBHOST1 and WEBHOST2:

1. Log in to WEBHOST1 and navigate to the following directory:

   *WEB_DOMAIN_HOME*/bin

2. Start the Node Manager as shown in the following sections by using `nohup` and `nodemanager.out` as an example output file:

   ```
   nohup WEB_DOMAIN_HOME/bin/startNodeManager.sh > WEB_DOMAIN_HOME/nodemanager/
   nodemanager.out 2>&1 &
   ```

3. Log in to WEBHOST2 and perform steps 1 and 2.

See Advanced Node Manager Configuration in *Administering Node Manager for Oracle WebLogic Server*.

## Starting the Oracle HTTP Server Instances

To start the Oracle HTTP Server instances:

1. Navigate to the following directory on WEBHOST1:

   *WEB_DOMAIN_HOME*/bin

   For more information about the location of the WEB_DOMAIN_HOME directory, see File System and Directory Variables Used in This Guide.

2. Enter the following command:

   ```
   ./startComponent.sh ohs1
   ```

> **Note:**
>
> Every time you start the Oracle HTTP server, you will be asked for the Node Manager password. If you do not wish this behaviour, then use the following command the first time you start the Oracle HTTP server:
>
> ```
> ./startComponent.sh ohs1 storeUserConfig
> ```
>
> This time when you enter the Node Manager password, it will be encrypted and stored. Future start and stop of the Oracle HTTP server will not require you to enter the Node Manager password.

> **Note:**
>
> For more information, see Storing Your Node Manager Password.

3. When prompted, enter the Node Manager password.
4. Repeat steps 1 through 3 to start the `ohs2` instance on WEBHOST2. See Starting Oracle HTTP Server Instances in *Administering Oracle HTTP Server*.

# Creating a Health Check

Create a health check on each Oracle HTTP Server instance. Oracle recommends using a specific page for health checks to avoid failures.

When an Oracle HTTP Server is accessed via a load balancer, the load balancer periodically checks if the Oracle HTTP Server is alive by requesting a page from the Oracle HTTP server. The default health check tries to access the root page from the server. If a WebGate is used, you need to ensure that this page is not intercepted as it can cause the health check to fail.

It is not recommended to open the root page for the purpose of a health check because this can cause a security risk. It is recommended to use a specific page that is used only for the health check.

You can also enable the server status page and check the page. However, the server status page contains lot of information when exposed can pose a security risk that can be avoided by creating a dedicated page (excluding any sensitive information) in the HTTP server for the purpose of a health check.

When the WebGate bypass is created it should be locked down so that health check requests can only come from certain sources (not from the internet).

Perform the following steps on each Oracle HTTP Server instance to create a simple health check page:

1. Create a file named `health-check.html` in the directory *`WEB_DOMAIN_HOME`*`/config/fmwconfig/components/OHS/<OHS_NAME>/htpdocs` with the following:

```
<!DOCTYPE html>
<html>
<body>
```

```
<h1>OK</h1>

</body>
</html>
```

2. Verify whether you can view the page by accessing *http://WEBHOST1.example.com:7777/health-check.html*.

   A page is displayed with the message OK.

> **Note:**
>
> Ensure that you are checking `/health-check.html` when configuring your load balancer.

# Backing Up the Configuration

As a best practice, Oracle recommends you to back up the configuration after you have successfully extended a domain or at another logical point. Back up only after you have verified that the installation is successful so far. This is a quick backup to enable immediate restoration in case of problems in later steps.

In a Kubernetes environment, it is sufficient to back up the persistent volume and the database.

The backup destination is the local disk. You can discard this backup when the enterprise deployment setup is complete. After the enterprise deployment setup is complete, you can initiate the regular deployment-specific Backup and Recovery process.

For information about backing up your configuration, see Performing Backups and Recoveries for an Enterprise Deployment.

# Configuring Oracle HTTP Server to Route Requests to the Application Tier

Update the Oracle HTTP Server configuration files so that the web server instances route requests to the servers in the domain.

- About the Oracle HTTP Server Configuration for an Enterprise Deployment
- Modifying the httpd.conf File to Include Virtual Host Configuration Files
- Modifying the httpd.conf File to Set Server Runtime Parameters
- Creating an Oracle HTTP Server Wallet
- Obtaining the Port for the Kubernetes Node Port Service
- Routing Requests
- Creating the Virtual Host Configuration Files

## About the Oracle HTTP Server Configuration for an Enterprise Deployment

The following topics provide an overview about the changes that are required to the Oracle HTTP Server configuration files on each WEBHOST, in an enterprise deployment.

## Purpose of the Oracle HTTP Server Virtual Hosts

The reference topologies in this guide require that you define a set of virtual servers on the hardware load balancer. You can then configure Oracle HTTP Server to recognize requests to specific virtual hosts (that map to the load balancer virtual servers) by adding `<VirtualHost>` directives to the Oracle HTTP Server instance configuration files.

For each Oracle HTTP Server virtual host, you define a set of specific URLs (or context strings) that route requests from the load balancer through the Oracle HTTP Server instances to the appropriate Administration Server or Managed Server in the Oracle WebLogic Server domain.

## About the WebLogicCluster Parameter of the <VirtualHost> Directive

A key parameter of the Oracle HTTP Server `<VirtualHost>` directive is the `WebLogicCluster` parameter, which is part of the WebLogic Proxy Plug-in for Oracle HTTP Server. When you configure Oracle HTTP Server for an enterprise deployment, consider the following information when you add this parameter to the Oracle HTTP Server configuration files.

In a Kubernetes environment, the WebLogic servers are deployed in pods and these pods use internal Kubernetes host names. These host names are not resolvable outside of the Kubernetes cluster. Kubernetes interacts with the WebLogic server pods using a Kubernetes service. This service expands and contracts dynamically as WebLogic Managed Server pods are added and taken away.

The servers specified in the `WebLogicCluster` parameter in a Kubernetes environment cannot reference the WebLogic Managed Server pods directly. They must interact by using a Kubernetes service. Kubernetes services are exposed on Kubernetes worker hosts through a mapped Kubernetes port. If you are using NodePort Services, there will be a unique port for each service. If you are using an ingress controller, you will use a single port for all services.

In a traditional on-premise deployment, the `WebLogicCluster` directive will reference the WebLogic server hosts and corresponding ports. In a Kubernetes environment, the `WebLogicCluster` directive must reference the Kubernetes worker nodes and the exposed Kubernetes service mapped port. If you have created a network load balancer to route requests to the worker nodes, you can specify this as the host name.

Because a Kubernetes service expands and contracts dynamically as WebLogic pods are added/removed, pointing the `WeblogicCluster` parameter at a Kubernetes worker node and the exposed port is sufficient to ensure that you are load balancing across all the WebLogic Managed Servers in the cluster.

However, including only one worker node in the `WeblogicCluster` directive means that if that worker node fails, but the cluster survives, the system will cease to work. To mitigate the impact of this failure, be sure to include several worker nodes (not necessarily all) or the network load balancer in the `WebLogicCluster` directive.

Associated with the `WebLogicCluster` directive it the `DynamicServerList` directive. If enabled (the default option), when new Managed Servers are added to a cluster, the server it is running on is published to the Oracle `WebLogicCluster` directive so that you do not need to change the Oracle HTTP Server configuration when the cluster changes. This option works

well in a traditional deployment. However, in a Kubernetes deployment, where the internal host names are unresolvable outside the cluster, it will cause issues. It is also unnecessary because the Kubernetes service provides the same functionality. Therefore, in an Oracle HTTP server, which directs a request to a Kubernetes cluster, the WebLogic directive `DynamicServerList` should be set to `false`.

## Recommended Structure of the Oracle HTTP Server Configuration Files

Rather than adding multiple virtual host definitions to the `httpd.conf` file, Oracle recommends that you create separate, smaller, and more specific configuration files for each of the virtual servers required for the products that you are deploying. This avoids populating an already large `httpd.conf` file with additional content, and it can make troubleshooting configuration problems easier.

For example, in a typical Oracle Fusion Middleware Infrastructure domain, you can add a specific configuration file called `admin_vh.conf` that contains the virtual host definition for the Administration Server virtual host (ADMINVHN).

# Modifying the httpd.conf File to Include Virtual Host Configuration Files

Perform the following tasks to prepare the `httpd.conf` file for the additional virtual hosts required for an enterprise topology:

1. Log in to WEBHOST1.
2. Locate the `httpd.conf` file for the first Oracle HTTP Server instance (`ohs1`) in the domain directory:

   ```
   cd WEB_DOMAIN_HOME/config/fmwconfig/components/OHS/ohs1/
   ```

3. Verify if the `httpd.conf` file has the appropriate configuration as follows:
   a. Run the following command to verify the `ServerName` parameter, be sure that it is set correctly, substituting the correct value for the current WEBHOST*n*:

      ```
      grep "ServerName http" httpd.conf
      ServerName http://WEBHOST1:7777
      ```

   b. Run the following command to verify there is an include statement that includes all `*.conf` files from the moduleconf subdirectory:

      ```
      grep moduleconf httpd.conf
      IncludeOptional "moduleconf/*.conf"
      ```

   c. If either validation fails to return results, or returns results that are commented out, open the `httpd.conf` file in a text editor and make the required changes in the appropriate locations.

      ```
      #
      # ServerName gives the name and port that the server uses to identify
      itself.
      # This can often be determined automatically, but we recommend you
      specify
      # it explicitly to prevent problems during startup.
      #
      # If your host doesn't have a registered DNS name, enter its IP address
      here.
      ```

```
#
ServerName http://WEBHOST1:7777
#  and at the end of the file:
# Include the admin virtual host (Proxy Virtual Host) related
configuration
include "admin.conf"
IncludeOptional "moduleconf/*.conf"
```

d. Save the `httpd.conf` file.

4. Log in to `WEBHOST2` and perform steps 2 and 3 for the `httpd.conf` file, replacing any occurrences of `WEBHOST1` or `ohs1` with `WEBHOST2` or `ohs2` in the instructions as necessary.

## Modifying the httpd.conf File to Set Server Runtime Parameters

Out of the box, the Oracle HTTP Server comes configured with a number of values which effect how the server behaves when it is running. For most of the deployments, these values are sufficient. However, in an Oracle Identity and Access Management deployment, it is recommended that you update these values by doing the following:

1. Log in to WEBHOST1.
2. Locate the `httpd.conf` file for the first Oracle HTTP Server instance (`ohs1`) in the domain directory:

   `cd` *`WEB_DOMAIN_HOME`*`/config/fmwconfig/components/OHS/`**`ohs1`**`/`

3. Locate the section of the file with the following line:

   ```
   <IfModule mpm_worker_module>
   ```

4. Update the entries in this section to reflect the following:

   ```
   <IfModule mpm_worker_module>
     ServerLimit             20
     StartServers            10
     MaxClients              1500
     MinSpareThreads         200
     MaxSpareThreads         800
     ThreadsPerChild         250
     ThreadLimit             250
     MaxRequestsPerChild     1000
     MaxRequestWorkers       400
     MaxConnectionsPerChild  0
    </IfModule>
   ```

5. Update the following values:
   - `MaxKeepAliveRequests 0`
   - `Timeout 300`
   - `KeepAliveTimeout 10`
6. Save the `httpd.conf` file.
7. Update the file `mod_wl_ohs.conf` to reflect the following:

   ```
   <IfModule weblogic_module>
   ```

```
   WLDNSRefreshInterval 10
</IfModule>
```

8. Save the file.
9. Log in to `WEBHOST2` and perform steps 2 and 3 for the `httpd.conf` and `mod_wl_ohs.conf` files, replacing any occurrences of `WEBHOST1` or `ohs1` with `WEBHOST2` or `ohs2` in the instructions as necessary.

## Creating an Oracle HTTP Server Wallet

If the back-end application is SSL enabled, such as Oracle Advanced Authentication, you should enable Oracle HTTP Server to trust the back end's SSL certificate. For establishing this trust, you should create a wallet in Oracle HTTP Server and store the trusted certificates.

> **Note:**
>
> A wallet is not required if you are using Ingress.

To create the OHS wallet, perform the following steps on each web server - WEBHOST1 and WEBHOST2. The wallet is created in the `OHS Domain` folder and is called `ohswallet`. Further sections of the guide assumes this location. However, you can place the wallet in any location.

1. Set the environment variables.

   *ORACLE_HOME*, *OHS_DOMAIN_HOME* and add *`ORACLE_HOME`*`/bin` and *`ORACLE_HOME`*`/oracle_common/bin` to the PATH:

   For example:

   ```
   export ORACLE_HOME=/u02/private/oracle/products/ohs/
   export OHS_DOMAIN_HOME=/u02/private/oracle/config/domains/ohsDomain
   export PATH=$ORACLE_HOME/bin:$ORACLE_HOME/oracle_common/bin:$PATH
   ```

2. Create the wallet using the following command:

   ```
   orapki wallet create -wallet $OHS_DOMAIN_HOME/ohswallet -auto_login_only
   ```

3. Repeat on each webhost.

- Adding Certificates to the Wallet

## Adding Certificates to the Wallet

1. Add certificates to the wallet by using the following command:

   ```
   orapki wallet add -wallet $OHS_DOMAIN_HOME /ohswallet -trusted_cert -cert
   <CERTIFICATE_FILE> -auto_login_only
   ```

2. Repeat on each webhost.

## Obtaining the Port for the Kubernetes Node Port Service

Each of the configuration procedures explained in this chapter directs Oracle HTTP to send requests to the Kubernetes Node Port service for the cluster of Managed Servers/instances or micro services. These procedures use sample ports for illustration.

To obtain the port that is actually being used, run the following command:

```
kubectl get service -n <NAMESPACE> | grep NodePort | grep <SERVICE_NAME> |
awk '{ print $5 }'
```

If you are using an Ingress controller instead of individual node port services, you should use the Ingress NodePort Service for each entry. To obtain the Ingress NodePort, use the following command:

```
kubectl get service -n <INGRESSNS> | grep NodePort | awk '{ print $5 }'
```

For example:

```
kubectl get service -n ingressns | grep NodePort | awk '{ print $5 }'
```

## Routing Requests

In the examples below, you will see routing rules which are of the form:

```
WeblogicCluster K8worker1.example.com:Port, K8Worker2.example.com:port
```

If you have defined a network load balancer, use the following:

```
WeblogicCluster K8workers.example.com:Port, K8Workers.example.com:port
```

Here, `K8workers.example.com` is the name of your network load balancer.

If you are using an Ingress controller, the port will always be the port that is assigned to the Ingress controller.

If you are using the NodePort Services, the port will depend on the NodePort Services you create.

## Creating the Virtual Host Configuration Files

To create the virtual host configuration files:

> **Note:**
>
> Before you create the virtual host configuration files, be sure that you have configured the virtual servers on the load balancer, as described in Purpose of the Oracle HTTP Server Virtual Hosts.

1. Log in to WEBHOST1 and change directory to the configuration directory for the first Oracle HTTP Server instance (`ohs1`):

   ```
   cd WEB_DOMAIN_HOME/config/fmwconfig/components/OHS/ohs1/moduleconf
   ```

2. If you are configuring Oracle Access Management, create the `iadadmin_vh.conf` file and add the following directive:

   ```
   <VirtualHost WEBHOST1.example.com:7777>
       ServerName http://iadadmin.example.com:80
       ServerAdmin you@your.address
       RewriteEngine On
       RewriteOptions inherit
       UseCanonicalName On
       RequestHeader set "X-Forwarded-Host" "iadadmin.example.com"
   </VirtualHost>
   ```

   > **Note:**
   >
   > `"X-Forwarded-Host"` is required only if you use an Ingress controller.

3. If you are configuring Oracle Access Management, create the `login_vh.conf` file and add the following directive:

   ```
   <VirtualHost WEBHOST1.example.com:7777>
       ServerName https://login.example.com:443
       ServerAdmin you@your.address
       RewriteEngine On
       RewriteOptions inherit
       UseCanonicalName On
       RequestHeader set "X-Forwarded-Host" "login.example.com"
   </VirtualHost>
   ```

   > **Note:**
   >
   > `"X-Forwarded-Host"` is required only if you use an Ingress controller.

   If you are using Oracle Advanced Authentication as well as OAM, add the following entries to the `login_vh.conf` file:

   ```
   <VirtualHost WEBHOST1.example.com:7777>
       ServerName https://login.example.com:443
       ServerAdmin you@your.address
       RewriteEngine On
       RewriteOptions inherit
       UseCanonicalName On
       RequestHeader set "X-Forwarded-Host" "iadadmin.example.com"
       RequestHeader set X-OAUTH-IDENTITY-DOMAIN-NAME "OAADomain"
       RewriteRule ^/oauth2/rest/authorize? /oauth2/rest/authorize?
   domain=OAADomain [PT,QSA,L]
       RewriteRule ^/oauth2/rest/token? /oauth2/rest/token?domain=OAADomain
   [PT,QSA,L]
       RewriteRule ^/oauth2/rest/token/info? /oauth2/rest/token/info?
   domain=OAADomain [PT,QSA,L]
   ```

```
    RewriteRule ^/oauth2/rest/authz? /oauth2/rest/authz?domain=OAADomain
[PT,QSA,L]
    RewriteRule ^/oauth2/rest/userinfo? /oauth2/rest/userinfo?
domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/security? /oauth2/rest/security?
domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/userlogout? /oauth2/rest/userlogout?
domain=OAADomain [PT,QSA,L]
</VirtualHost>
```

Where `OAADomain` can be any value you prefer, as long as it is consistent with the value you use when deploying OAA.

4. If you are configuring Oracle Identity Governance, create the `igdadmin_vh.conf` file, and add the following directive:

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName http://igdadmin.example.com:80
    ServerAdmin you@your.address
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
    RequestHeader set "X-Forwarded-Host" "igdadmin.example.com"
</VirtualHost>
```

> **Note:**
>
> `"X-Forwarded-Host"` is required only if you use an Ingress controller.

5. If you are configuring Oracle Identity Governance, create the `prov_vh.conf` file, and add the following directive:

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName https://prov.example.com:443
    ServerAdmin you@your.address
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
    RequestHeader set "X-Forwarded-Host" "prov.example.com"
</VirtualHost>
```

> **Note:**
>
> `"X-Forwarded-Host"` is required only if you use an Ingress controller.

6. If you are configuring Oracle Identity Governance, create the `igdinternal_vh.conf` file, and add the following directive:

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName http://igdinternal.example.com:7777
    ServerAdmin you@your.address
    RewriteEngine On
```

```
    RewriteOptions inherit
    RequestHeader set "X-Forwarded-Host" "igdinternal.example.com"
</VirtualHost>
```

> **Note:**
>
> "X-Forwarded-Host" is required only if you use an Ingress controller.

# Configuring Oracle HTTP Server for Oracle Access Manager

You have to configure Oracle HTTP Server for the Oracle Access Manager Managed Servers to ensure they route requests correctly to the Oracle Access Management cluster.

The following variables are used in this section:

**Table 17-2 List of Variables and Their Values**

| Variable | Value |
|---|---|
| *<OAM_OAM_K8>* | The Kubernetes service port of OAM. For example: 30410. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |
| *<OAM_ADMIN_K8>* | The Kubernetes service port of the OAM Administration Server. For example: 30701. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |
| *<OAM_POLICY_K8>* | The Kubernetes service port of the OAM Policy Service. For example: 30510. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |

To configure the Oracle HTTP Server instances in the web tier so they route requests correctly to the Oracle Access Management cluster, use the following procedure to create an additional Oracle HTTP Server configuration file that creates and defines the parameters of the login.example.com virtual server. To configure Oracle HTTP Server for the oam_server Managed Servers:

1. Log in to WEBHOST1 and change directory to the configuration directory for the first Oracle HTTP Server instance (ohs1).

   ```
   cd WEB_DOMAIN_HOME/config/fmwconfig/components/OHS/ohs1/moduleconf/
   ```

> **Note:**
>
> There are separate directories for configuration and runtime instance files. The runtime files under the .../OHS/instances/ohsn/* folder should not be edited directly. Edit only the .../OHS/ohsn/* configuration files.

2. In the `login_vh.conf` file, add the following lines between the `<VirtualHost>` and `</VirtualHost>` tags:

```
#OAM Entries
<Location /oam>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
</Location>

<Location /oam/services/rest/auth>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
</Location>

<Location /oam/services/rest/access>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
</Location>

<Location /oamfed>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

# OAM Forgotten Password Page
<Location /otpfp/>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_OA
M_K8>
    WLCookieName OAMJSESSIONID
```

```
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /ms_oauth>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /oauth2>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /.well-known/openid-configuration>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    PathTrim /.well-known
    PathPrepend /oauth2/rest
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /.well-known/oidc-configuration>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    PathTrim /.well-known
    PathPrepend /oauth2/rest
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /CustomConsent>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
```

```
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /iam/access>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_OAM_K8>,K8_WORKER_HOST2.example.com:<OAM_O
AM_K8>
    WLCookieName OAMJSESSIONID
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>
```

3. In the `iadadmin_vh.conf` file, add the following lines between the `<VirtualHost>` and `</VirtualHost>` tags:

```
<Location /console>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
</Location>

# WebLogic Remote Console Access
#
<Location /management>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
</Location>

<Location /consolehelp>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
</Location>

<Location /em>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
</Location>

<Location /oamconsole>
```

```
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
</Location>

<Location /access>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OAM_POLICY_K8>,K8_WORKER_HOST2.example.com:<OA
M_POLICY_K8>
    WLCookieName OAMJSESSIONID
</Location>

<Location /iam/admin>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
</Location>

<Location /oam/services/rest/11.1.2.0.0>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
    WLCookieName OAMJSESSIONID
</Location>

<Location /oam/services/rest/ssa>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
    WLCookieName OAMJSESSIONID
</Location>

<Location /oam/services>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
_ADMIN_K8>
    WLCookieName OAMJSESSIONID
</Location>

<Location /dms>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OAM_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OAM
```

```
_ADMIN_K8>
</Location>
```

4. Copy the `iadadmin_vh.conf` file and `login_vh.conf` to the configuration directory for the second Oracle HTTP Server instance (`ohs2`):

   *WEB_DOMAIN_HOME*/config/fmwconfig/components/**ohs2**/moduleconf/

5. Edit the `login_vh.conf` and `iadadmin_vh.conf` files to change references of WEBHOST1 to WEBHOST2 in the `<VirtualHost>` directives.

# Configuring Oracle HTTP Server for Oracle Identity Governance

To configure the Oracle HTTP Server instances in the web tier so they route requests correctly to the Oracle SOA Suite cluster, use the following procedure to create an additional Oracle HTTP Server configuration file that creates and defines the parameters of the `https://igdinternal.example.com:7777` virtual server.

This procedure assumes that you have performed the Oracle HTTP Server configuration tasks described in Configuring Oracle HTTP Server to Route Requests to the Application Tier.

The following variables are used in this section:

**Table 17-3 List of Variables and Their Values**

| Variable | Value |
|---|---|
| *<OIG_OIM_PORT_K8>* | The Kubernetes service port of the OIG OIM Service. For example: `30140`. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |
| *<OIG_ADMIN_K8>* | The Kubernetes service port of the OIG Administration Server service. For example: `30711`. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |
| *<OIG_SOA_PORT_K8>* | The Kubernetes service port of the OIG SOA service. For example: `30801`. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |

To create the virtual host configuration file so requests are routed properly to the Oracle Identity Governance clusters:

1. Log in to WEBHOST1 and change directory to the configuration directory for the first Oracle HTTP Server instance (OHS1):

   ```
   cd WEB_DOMAIN_HOME/config/fmwconfig/components/OHS/ohs1/moduleconf/
   ```

2. Edit the file `prov_vh.conf` and add the following directives inside the `<VirtualHost>` tags:

**Note:**

- The URL entry for `/workflow` is optional. It is for workflow tasks associated with Oracle ADF task forms. The `/workflow` URL itself can be a different value, depending on the form.
- Configure the port numbers appropriately, as assigned for your static or dynamic cluster. Dynamic clusters with the Calculate Listen Port option selected will have incremental port numbers for each dynamic managed server that you create.

  The WebLogicCluster directive needs only a sufficient number of redundant *server:port* combinations to guarantee an initial contact in case of a partial outage. The actual total list of cluster members is retrieved automatically on the first contact with any given node.

```
<Location /identity>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /HTTPClnt>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

# Requests webservice URL
<Location /reqsvc>
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /FacadeWebApp>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
```

```
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /iam>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>

<Location /OIGUI>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
    WLProxySSL ON
    WLProxySSLPassThrough ON
</Location>
```

The `prov_vh.conf` file will appear as it does in Step 2.

3. In the `igdadmin_vh.conf` file, add the following lines between `<VirtualHost>` and `</VirtualHost>` tags:

```
## Entries Required by Oracle Identity Governance
<Location /console>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OIG_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OIG_ADMIN_K8>
</Location>

# WebLogic Remote Console Access
#
<Location /management>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OIG_ADMIN_K8>
</Location>

<Location /consolehelp>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OIG_ADMIN_K8>
</Location>

<Location /em>
    WLSRequest ON
    DynamicServerList OFF
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OIG_ADMIN_K8>
</Location>

<Location /oim>
```

```
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /iam>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /sysadmin>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /admin>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# OIM self service console
<Location /identity>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /OIGUI>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /FacadeWebApp>
    SetHandler weblogic-handler
```

```
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# Scheduler webservice URL
<Location /SchedulerService-web>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /dms>
    WLSRequest ON
    DynamicServerList OFF
    WeblogicCluster
K8_WORKER_HOST1.example.com:<OIG_ADMIN_K8>,K8_WORKER_HOST2.example.com:<OIG_ADMIN_K8>
</Location>
```

4. In the `igdinternal_vh.conf` file, add the following lines between the `<VirtualHost>` and `</VirtualHost>` tags:

```
## Entries Required by Oracle Identity Governance
#SOA Callback webservice for SOD - Provide the SOA Managed Server Ports

<Location /sodcheck>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_SOA_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>

# OIM, role-sod profile
<Location /role-sod>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# Callback webservice for SOA. SOA calls this when a request is approved/rejected
# Provide the SOA Managed Server Port
<Location /workflowservice>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>
```

```
# used for FA Callback service.
<Location /callbackResponseService>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName    oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# spml xsd profile
<Location /spml-xsd>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# OIM, spml dsml profile
<Location /spmlws>
    WLSRequest ON
    DynamicServerList OFF
    PathTrim /weblogic
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /reqsvc>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>

# SOA Infra
<Location /soa-infra>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_SOA_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/OHS/component/oim_component.log"
</Location>

# UMS Email Support
<Location /ucs>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    WebLogicCluster
```

```
K8_WORKER_HOST1.example.com:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_SOA_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/OHS/component/oim_component.log"
</Location>

<Location /provisioning-callback>
    WLSRequest ON
    DynamicServerList
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /CertificationCallbackService>
   WLSRequest ON
   DynamicServerList OFF
   WLCookieName oimjsessionid
   WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
   WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /IdentityAuditCallbackService>
   WLSRequest ON
   DynamicServerList OFF
   WLCookieName oimjsessionid
   WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
   WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# SOA Callback webservice for SOD - Provide the SOA Managed Server Ports
<Location /soa/composer>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_SOA_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>

<Location /integration>
    SetHandler weblogic-handler
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_SOA_PO
RT_K8>
    WLCookieName oimjsessionid
</Location>

<Location /sdpmessaging/userprefs-ui>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_SOA_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>
```

```
<Location /iam>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1.example.com:<OIG_OIM_PORT_K8>,K8_WORKER_HOST2.example.com:<OIG_OIM_PO
RT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /ws_utc>
    SetHandler weblogic-handler
    WLCookieName oimjsessionid
    WebLogicCluster
K8_WORKER_HOST1:<OIG_SOA_PORT_K8>,K8_WORKER_HOST2:<OIG_SOA_PORT_K8>
    WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>
```

5. Copy the `igdadmin_vh.conf`, `igdinternal_vh.conf`, and `prov_vh.conf` files to the configuration directory for the second Oracle HTTP Server instance (ohs2):

   *WEB_DOMAIN_HOME*`/config/fmwconfig/components/OHS/`**`ohs2`**`/moduleconf/`

6. Edit the `igdadmin_vh.conf`, `prov_vh.conf`, and `igdinternal_vh.conf` files and change any references to WEBHOST1 to WEBHOST2 in the `<VirtualHost>` directives.

> **Note:**
>
> If internal invocations are going to be used in the system, add the appropriate locations to the soainternal virtual host.

# Configuring Oracle HTTP Server for Oracle Identity Role Intelligence

You should configure Oracle HTTP Server for the Oracle Identity Role Intelligence (OIRI) Servers to ensure that they route requests correctly to the Oracle Role Intelligence cluster.

The following variables are used in this section:

**Table 17-4 List of Variables and Their Values**

| Variable | Value |
|---|---|
| *<OIRI_UI_K8>* | The Kubernetes service port of the OIRI UI service. For example: `30306`. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |
| *<OIRI_K8>* | The Kubernetes service port of the OIRI service. For example: `30305`. If you are using an Ingress controller, this value will be the Kubernetes service port of the Kubernetes controller. |

To configure Oracle HTTP Server:

1. Log in to WEBHOST1 and change directory to the configuration directory for the first Oracle HTTP Server instance (OHS1):

   ```
   cd WEB_DOMAIN_HOME/config/fmwconfig/components/OHS/ohs1/moduleconf/
   ```

2. Edit the `igdadmin_vh.conf` file and add the following directives inside the <`VirtualHost`> tags:

   ```
   # OIRI UI
   # <Location /oiri/ui>
        WLSRequest ON
        DynamicServerList OFF
        WLCookieName oimjsessionid
        DynamicServerList OFF
        WebLogicCluster
   K8_WORKER_HOST1.example.com:<OIRI_UI_K8>,K8_WORKER_HOST2.example.com:<OIRI_
   UI_K8>
   </Location>
   ```

3. In the `igdinternal_vh.conf` file, add the following lines between the <`VirtualHost`> and <`/VirtualHost`> tags:

   ```
   # OIRI API
   # <Location /oiri/api>
        WLSRequest ON
        DynamicServerList OFF
        WLCookieName oimjsessionid
        DynamicServerList OFF
        WebLogicCluster
   K8_WORKER_HOST1.example.com:<OIRI_K8>,K8_WORKER_HOST2.example.com:<OIRI_K8>
   </Location>
   ```

4. Copy the `igdadmin_vh.conf` and `igdinternal_vh.conf` files to the configuration directory for the second Oracle HTTP Server instance (ohs2): *`WEB_DOMAIN_HOME`*`/config/fmwconfig/components/OHS/ohs2/moduleconf/`

# Configuring Oracle HTTP Server for Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

You should configure Oracle HTTP Server for Oracle Advanced Authentication servers to ensure that they route requests correctly to the OAA microservices.

The following variables are used in this section:

**Table 17-5 List of Variables and Their Values**

| Variable | Value |
|---|---|
| *<K8_WORKER_HOST1>* | The name of one of the Kubernetes worker hosts. |
| *<K8_WORKER_HOST2>* | The name of a different Kubernetes worker host. |

**Table 17-5 (Cont.) List of Variables and Their Values**

| Variable | Value |
|---|---|
| *<OAA_ADMIN_K8>* | The node port for the `oaa-admin` Kubernetes service. For example: `31338`. |
| *<OAA_K8>* | The node port for the `oaa-svc` Kubernetes service. For example: `31047`. |
| *<OAA_POLICY_K8>* | The node port for the `oaa-policy` Kubernetes service. For example: `31957`. |
| *<OAA_SPUI_K8>* | The node port for the `oaa-spui` Kubernetes service. For example: `30532`. |
| *<OAA_FIDO_K8>* | The node port for the `oaa-factor-fido` Kubernetes service. For example: `32438`. |
| *<OAA_EMAIL_K8>* | The node port for the `oaa-factor-email` Kubernetes service. For example: `30614`. |
| *<OAA_SMS_K8>* | The node port for the `oaa-factor-sms` Kubernetes service. For example: `31930`. |
| *<OAA_TOTP_K8>* | The node port for the `oaa-factor-totp` Kubernetes service. For example: `31950`. |
| *<OAA_YOTP_K8>* | The node port for the `oaa-factor-yotp` Kubernetes service. For example: `31946`. |
| *<OAA_PUSH_K8>* | The node port for the `oaa-factor-push` Kubernetes service. For example: `31166`. |
| *<OAA_KBA_K8>* | The node port for the `oaa-factor-kba` Kubernetes service. For example: `31147`. |
| *<OAA_RISK_ANAL_K8>* | The node port for the `risk-analysis` Kubernetes service. For example `30507`. |
| *<OAA_RISKCC_K8>* | The node port for the `risk-cc` Kubernetes service. For example: `30981`. |
| *<OAA_OUA_K8>* | The node port for the `oua` Kubernetes service. For example: `30520`. |
| *<OAA_OUAUI_K8>* | The node port for the `oua-ui` Kubernetes service. For example: `30525`. |
| *<OAA_DRSS>* | The node port for the `oaa-drss` service. For example, `30580`. |

> **Note:**
>
> The actual node port values in this table will be determined after you have deployed OAA.

To configure Oracle HTTP Server:

1. Log in to WEBHOST1 and change the directory to the configuration directory of the first Oracle HTTP Server instance (OHS1):

   ```
   cd WEB_DOMAIN_HOME/config/fmwconfig/components/OHS/ohs1/moduleconf/
   ```

2. Edit the `iadadmin_vh.conf` file and add the following directives inside the `<VirtualHost>` tags:

   ```
   # OAA
   #
   <Location /oaa-admin>
       WLSRequest ON
       WLCookieName OAMJSESSIONID
       DynamicServerList OFF
       SecureProxy ON
       WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
       WebLogicCluster
   <K8_WORKER_HOST1>:<OAA_ADMIN_K8>,<K8_WORKER_HOST2>:<OAA_ADMIN_K8>
   </Location>

   <Location /admin-ui>
       WLSRequest ON
       WLCookieName OAMJSESSIONID
       DynamicServerList OFF
       SecureProxy ON
       WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
       WebLogicCluster
   <K8_WORKER_HOST1>:<OAA_ADMIN_K8>,<K8_WORKER_HOST2>:<OAA_ADMIN_K8>
   </Location>

   <Location /oaa-policy>
       WLSRequest ON
       WLCookieName OAMJSESSIONID
       WLProxySSL ON
       WLProxySSLPassThrough ON
       DynamicServerList OFF
       SecureProxy ON
       WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
       WebLogicCluster
   <K8_WORKER_HOST1>:<OAA_POLICY_K8>,<K8_WORKER_HOST2>:<OAA_POLICY_K8>
   </Location>

   <Location /policy>
       WLSRequest ON
       WLCookieName OAMJSESSIONID
       WLProxySSL ON
       WLProxySSLPassThrough ON
       DynamicServerList OFF
       SecureProxy ON
       WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
       WebLogicCluster
   <K8_WORKER_HOST1>:<OAA_POLICY_K8>,<K8_WORKER_HOST2>:<OAA_POLICY_K8>
   </Location>

   <Location /risk-cc>
       WLSRequest ON
   ```

```
        WLCookieName OAMJSESSIONID
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_RISKCC_K8>,<K8_WORKER_HOST2>:<OAA_RISKCC_K8>
    </Location>

    <Location /oua-admin-ui>
        WLSRequest ON
        WLCookieName OAMJSESSIONID
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_OUAUI_K8>,<K8_WORKER_HOST2>:<OAA_OUAUI_K8>
    </Location>
```

3. Edit the `login_vh.conf` file and add the following directives inside the `<VirtualHost>` tags:

```
    # OAA
    #
    <Location /oaa/runtime>
        WLSRequest ON
        WLProxySSL ON
        WLCookieName OAMJSESSIONID
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_K8>,<K8_WORKER_HOST2>:<OAA_K8>
    </Location>

    <Location /oaa-policyi>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_POLICY_K8>,<K8_WORKER_HOST2>:<OAA_POLICYI_K8>
    </Location>

    <Location /policyi>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
```

```
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_POLICY_K8>,<K8_WORKER_HOST2>:<OAA_POLICYI_K8>
    </Location>

    <Location /oaa/rui>
        WLSRequest ON
        WLProxySSL ON
        WLCookieName OAMJSESSIONID
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_SPUI_K8>,<K8_WORKER_HOST2>:<OAA_SPUI_K8>
    </Location>

    <Location /oaa/authnui>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_SPUI_K8>,<K8_WORKER_HOST2>:<OAA_SPUI_K8>
    </Location>

    <Location /fido>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_FIDO_K8>,<K8_WORKER_HOST2>:<OAA_FIDO_K8>
    </Location>

    <Location /oaa-email-factor>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_EMAIL_K8>,<K8_WORKER_HOST2>:<OAA_EMAIL_K8>
    </Location>

    <Location /oaa-sms-factor>
        WLSRequest ON
        WLProxySSL ON
```

```
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_SMS_K8>,<K8_WORKER_HOST2>:<OAA_SMS_K8>
    </Location>

    <Location /oaa-totp-factor>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_TOTP_K8>,<K8_WORKER_HOST2>:<OAA_TOTP_K8>
    </Location>

    <Location /oaa-push-factor>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_PUSH_K8>,<K8_WORKER_HOST2>:<OAA_PUSH_K8>
    </Location>

    <Location /oaa-yotp-factor>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_YOTP_K8>,<K8_WORKER_HOST2>:<OAA_YOTP_K8>
    </Location>

    <Location /oaa-kba>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_KBA_K8>,<K8_WORKER_HOST2>:<OAA_KBA_K8>
    </Location>
```

```
    <Location /risk-analyzer>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_RISK_ANAL_K8>,<K8_WORKER_HOST2>:<OAA_RISK_ANAL_K8>
    </Location>

    <Location /risk-cc>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_RISKL_K8>,<K8_WORKER_HOST2>:<OAA_RISKL_K8>
    </Location>

    <Location /oua>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_OUA_K8>,<K8_WORKER_HOST2>:<OAA_OUA_K8>
    </Location>

    <Location /oaa-drss>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster
<K8_WORKER_HOST1>:<OAA_DRSS_K8>,<K8_WORKER_HOST2>:<OAA_DRSS_K8>
    </Location>
```

4. Edit the `login_vh.conf` file and add the following after the `RewriteEngine On` tag:

```
    RequestHeader set X-OAUTH-IDENTITY-DOMAIN-NAME "OAADomain"
    RewriteRule ^/oauth2/rest/authorize? /oauth2/rest/authorize?
domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/token? /oauth2/rest/token?domain=OAADomain
[PT,QSA,L]
```

```
    RewriteRule ^/oauth2/rest/token/info? /oauth2/rest/token/info?
domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/authz? /oauth2/rest/authz?domain=OAADomain
[PT,QSA,L]
    RewriteRule ^/oauth2/rest/userinfo? /oauth2/rest/userinfo?
domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/security? /oauth2/rest/security?
domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/userlogout? /oauth2/rest/userlogout?
domain=OAADomain [PT,QSA,L]
```

Where `OAADomain` is the name of the OAA domain. Ensure that this value is consistent with the value you use when you install OAA. See Installing and Configuring Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator.

5. Copy the `iadadmin_vh.conf` and `login_vh.conf` files to the configuration directory of the second Oracle HTTP Server instance (OHS2):

   *WEB_DOMAIN_HOME*`/config/fmwconfig/components/OHS/ohs2/moduleconf/`

# Restarting the OHS Instances

Ensure that you have copied the configuration files to each WEBHOST, and then restart the Oracle HTTP Service instance on each host.

To do this:

1. Restart the ohs1 instance by doing the following:
   a. Change directory to the following location:

      `cd` *WEB_DOMAIN_HOME*`/bin`

   b. Enter the following commands to stop and start the instance:

      ```
      ./stopComponent.sh ohs1
      ```

      ```
      ./startComponent.sh ohs1
      ```

2. Restart the ohs2 instance by doing the following:
   a. Change directory to the following location:

      `cd` *WEB_DOMAIN_HOME*`/bin`

   b. Enter the following commands to stop and start the instance:

      ```
      ./stopComponent.sh ohs2
      ```

      ```
      ./startComponent.sh ohs2
      ```

# Validating the Oracle HTTP Server Configuration

To ensure that the Oracle HTTP server is working correctly, you should perform a few validations after configuring the Oracle Identity Management products.



## Validating Access Through the Load Balancer

You should verify URLs to ensure that appropriate routing and failover is working from Oracle HTTP Server to OAM_Cluster.



### Verifying the URLs

To verify the URLs:

1. While `oam_server2` is running, stop `oam_server1` using the WebLogic Server Administration Console.
2. Access `https://login.example.com/oam/server/logout`.
3. Start `oam_server1` from the WebLogic Server Administration Console.
4. Stop `oam_server2` from the WebLogic Server Administration Console.
5. Access `http://login.example.com/oam/server/logout`.

You can verify the cluster node to which you were directed after the traffic balancing provided through your load balancer and then again through the web tier.

## Validating the Virtual Server Configuration and Access to the Consoles

Validate the virtual server configuration on the load balancer, and the access to the management console and the Administration Server.

From the load balancer, access the following URLs to ensure that the load balancer and Oracle HTTP Server are configured properly. These URLs should show the initial Oracle HTTP Server 12*c* web page.

- `https://login.example.com/index.html`
- `https://prov.example.com/index.html`
- `http://iadadmin.example.com/index.html`
- `http://igdadmin.example.com/index.html`

Use the following URLs to the hardware load balancer to display the Oracle WebLogic Server Administration Console, and log in using the Oracle WebLogic Server `iadadmin` credentials:

- `http://iadadmin.example.com/console`
- `http://iadadmin.example.com/em`

This validates that the `iadadmin.example.com` virtual host on the load balancer is able to route requests to the Oracle HTTP Server instances on the web tier, which in turn can route requests

for the Oracle WebLogic Server Administration Console to the Administration Server in the application tier.

Similarly, you should be able to access the WebLogic Server Administration Console and Fusion Middleware Control for the `igdadmin` virtual host using the following URLs:

- `http://igdadmin.example.com/console`
- `http://igdadmin.example.com/em`

# Sample Virtual Host Files

The sample list includes the complete examples of all the virtual host files used in an Oracle Identity and Access Management deployment.



**Example 1 `login_vh.conf`**

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName https://login.example.com:443
    ServerAdmin you@your.address
    RewriteEngine On
    RequestHeader set X-OAUTH-IDENTITY-DOMAIN-NAME "OAADomain"
    RewriteOptions inherit
    UseCanonicalName On
    RequestHeader set "X-Forwarded-Host" "login.example.com"
    RequestHeader set X-OAUTH-IDENTITY-DOMAIN-NAME "OAADomain"
    RewriteRule ^/oauth2/rest/authorize? /oauth2/rest/authorize?domain=OAADomain
[PT,QSA,L]
    RewriteRule ^/oauth2/rest/token? /oauth2/rest/token?domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/token/info? /oauth2/rest/token/info?domain=OAADomain
[PT,QSA,L]
    RewriteRule ^/oauth2/rest/authz? /oauth2/rest/authz?domain=OAADomain [PT,QSA,L]
    RewriteRule ^/oauth2/rest/userinfo? /oauth2/rest/userinfo?domain=OAADomain
[PT,QSA,L]
    RewriteRule ^/oauth2/rest/security? /oauth2/rest/security?domain=OAADomain
[PT,QSA,L]
    RewriteRule ^/oauth2/rest/userlogout? /oauth2/rest/userlogout?domain=OAADomain
[PT,QSA,L]

    #OAM Entries
    <Location /oam>
      WLSRequest ON
      DynamicServerList OFF
      WLProxySSL ON
      WLProxySSLPassThrough ON
      WLCookieName OAMJSESSIONID
      WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
    </Location>

    <Location /oam/services/rest/auth>
```

```
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
</Location>

<Location /oam/services/rest/access>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
</Location>

<Location /oamfed>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

# OAM Forgotten Password Page
<Location /otpfp/>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

<Location /ms_oauth>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

<Location /oauth2>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

<Location /.well-known/openid-configuration>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  PathTrim /.well-known
  PathPrepend /oauth2/rest
```

```
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

<Location /.well-known/oidc-configuration>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  PathTrim /.well-known
  PathPrepend /oauth2/rest
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

<Location /CustomConsent>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

<Location /iam/access>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30410,k8workerhost2.example.com:30410
  WLCookieName OAMJSESSIONID
  WLProxySSL ON
  WLProxySSLPassThrough ON
</Location>

# OAA
#
<Location /oaa/runtime>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31047,k8workerhost2.example.com:31047
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa-policy>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31957,k8workerhost2.example.com:31957
</Location>

<Location /policy>
    WLSRequest ON
```

```
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31957,k8workerhost2.example.com:31957
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa/rui>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:30532,k8workerhost2.example.com:30532
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa/authnui>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:30532,k8workerhost2.example.com:30532
    WLCookieName OAMJSESSIONID
</Location>

<Location /fido>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:32438,k8workerhost2.example.com:32438
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa-email-factor>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:30614,k8workerhost2.example.com:30614
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa-sms-factor>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
```

```
    WebLogicCluster k8workerhost1.example.com:31930,k8workerhost1.example.com:31930
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa-totp-factor>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31950,k8workerhost1.example.com:31950
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa-push-factor>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31166,k8workerhost2.example.com:31166
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa-yotp-factor>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31946,k8workerhost2.example.com:31946
    WLCookieName OAMJSESSIONID
</Location>

<Location /oaa/kba>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:31147,k8workerhost2.example.com:31147
    WLCookieName OAMJSESSIONID
</Location>

<Location /risk-analyzer>
    WLSRequest ON
    WLProxySSL ON
    WLProxySSLPassThrough ON
    DynamicServerList OFF
    SecureProxy ON
    WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
    WebLogicCluster k8workerhost1.example.com:30507,k8workerhost2.example.com:30507
    WLCookieName OAMJSESSIONID
</Location>

<Location /risk-cc>
```

```
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:30981,k8workerhost2.example.com:30981
        WLCookieName OAMJSESSIONID
    </Location>

    <Location /oua>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:30520,k8workerhost2.example.com:30520
        WLCookieName OAMJSESSIONID
    </Location>

    <Location /oaa-drss>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:30580,k8workerhost2.example.com:30580
        WLCookieName OAMJSESSIONID
    </Location>

    <Location /oua/rui>
        WLSRequest ON
        WLProxySSL ON
        WLProxySSLPassThrough ON
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:30580,k8workerhost2.example.com:30580
        WLCookieName OAMJSESSIONID
    </Location>
</VirtualHost>
```

### Example 2 `prov_vh.conf`

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName https://prov.example.com:443
    ServerAdmin you@your.address
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
    RequestHeader set "X-Forwarded-Host" "prov.example.com"

    <Location /identity>
      WLSRequest ON
      DynamicServerList OFF
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
```

```
      WLProxySSL ON
      WLProxySSLPassThrough ON
    </Location>

    <Location /HTTPClnt>
      WLSRequest ON
      DynamicServerList OFF
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
      WLProxySSL ON
      WLProxySSLPassThrough ON
    </Location>

    # Requests webservice URL
    <Location /reqsvc>
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLProxySSL ON
      WLProxySSLPassThrough ON
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
    </Location>

    <Location /FacadeWebApp>
      SetHandler weblogic-handler
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
      WLProxySSL ON
      WLProxySSLPassThrough ON
    </Location>

    <Location /iam>
      SetHandler weblogic-handler
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
      WLProxySSL ON
      WLProxySSLPassThrough ON
    </Location>

    <Location /OIGUI>
      SetHandler weblogic-handler
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
      WLProxySSL ON
      WLProxySSLPassThrough ON
    </Location>
</VirtualHost>
```

### Example 3 `iadadmin_vh.conf`

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName iadadmin.example.com:80
    ServerAdmin you@your.address
```

```
RewriteEngine On
RewriteOptions inherit
UseCanonicalName On
RequestHeader set "X-Forwarded-Host" "iadadmin.example.com"

# Admin Server and EM
<Location /console>
   WLSRequest ON
   DynamicServerList OFF
   WebLogicCluster k8workerhost1.example.com:30701,k8workerhost2.example.com:30701
</Location>

# WebLogic Remote Console Access
#
<Location /management>
   WLSRequest ON
   DynamicServerList OFF
   WebLogicCluster k8workerhost1.example.com:30711,k8workerhost2.example.com:30711
</Location>

<Location /consolehelp>
   WLSRequest ON
   DynamicServerList OFF
   WebLogicCluster k8workerhost1.example.com:30701,k8workerhost2.example.com:30701
</Location>

<Location /em>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30701,k8workerhost2.example.com:30701
</Location>

<Location /oamconsole>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30701,k8workerhost2.example.com:30701
</Location>

<Location /access>
  WLSRequest ON
  DynamicServerList OFF
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30510,k8workerhost2.example.com:30510
  WLCookieName OAMJSESSIONID
</Location>

<Location /iam/admin>
  WLSRequest ON
  DynamicServerList OFF
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com: 30701,k8workerhost2.example.com:30701
  WLCookieName OAMJSESSIONID
</Location>

<Location /oam/services/rest/11.1.2.0.0>
  WLSRequest ON
  DynamicServerList OFF
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com: 30701,k8workerhost2.example.com:30701
  WLCookieName OAMJSESSIONID
```

```
    </Location>

    <Location /oam/services/rest/ssa>
      WLSRequest ON
      DynamicServerList OFF
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com: 30701,k8workerhost2.example.com: 30701
      WLCookieName OAMJSESSIONID
    </Location>

    # Required for Multi-Datacenter
    <Location /oam/services>
      WLSRequest ON
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30701,k8workerhost2.example.com:30701
    </Location>
    # OAA
    #
    <Location /oaa-admin>
        WLSRequest ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:31338,k8workerhost2.example.com:31338
    </Location>

    <Location /admin-ui>
        WLSRequest ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet   "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:31338,k8workerhost2.example.com:31338
    </Location>

    <Location /oua-admin-ui>
        WLSRequest ON
        WLCookieName OAMJSESSIONID
        DynamicServerList OFF
        SecureProxy ON
        WLSSLWallet "${ORACLE_INSTANCE}/ohswallet"
        WebLogicCluster k8workerhost1.example.com:30525,k8workerhost2.example.com:30525
    </Location>

    <Location /dms>
       WLSRequest ON
       DynamicServerList OFF
       WebLogicCluster k8workerhost1.example.com:30701,k8workerhost2.example.com:30701
    </Location>

</VirtualHost>
```

### Example 4 `igdadmin_vh.conf`

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName igdadmin.example.com:80
    ServerAdmin you@your.address
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
```

```
RequestHeader set "X-Forwarded-Host" "igdadmin.example.com"

# Admin Server and EM
<Location /console>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30711,k8workerhost2.example.com:30711
</Location>

# WebLogic Remote Console Access
#
<Location /management>
   WLSRequest ON
   DynamicServerList OFF
   WebLogicCluster k8workerhost1.example.com:30711,k8workerhost2.example.com:30711
</Location>

<Location /consolehelp>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30711,k8workerhost2.example.com:30711
</Location>

<Location /em>
  WLSRequest ON
  DynamicServerList OFF
  WebLogicCluster k8workerhost.example.com:30711,k8workerhost.example.com:30711
</Location>

<Location /oim>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /iam>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /sysadmin>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /admin>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
```

```
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# OIM self service console
<Location /identity>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /OIGUI>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /FacadeWebApp>
  SetHandler weblogic-handler
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# Scheduler webservice URL
<Location /SchedulerService-web>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# OIRI UI
# <Location /oiri/ui>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    DynamicServerList OFF
    WebLogicCluster k8workerhost1.example.com:30306,k8workerhost2.example.com:30306
 </Location>

# OIRI API
# <Location /oiri/api>
    WLSRequest ON
    DynamicServerList OFF
    WLCookieName oimjsessionid
    DynamicServerList OFF
    WebLogicCluster k8workerhost1.example.com:30305,k8workerhost2.example.com:30305
 </Location>
```

```
    <Location /dms>
      WLSRequest ON
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30711,k8workerhost2.example.com:30711
    </Location>

</VirtualHost>
```

### Example 5 `igdinternal_vh.conf`

```
<VirtualHost WEBHOST1.example.com:7777>
    ServerName igdinternal.example.com:7777
    ServerAdmin you@your.address
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
    RequestHeader set "X-Forwarded-Host" "igdinternal.example.com"

    # WSM-PM
    <Location /wsm-pm>
      WLSRequest ON
      DynamicServerList OFF
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:7010,k8workerhost2.example.com:7010
      WLProxySSL OFF
      WLProxySSLPassThrough OFF
    </Location>

    <Location /sodcheck>
      WLSRequest ON
      DynamicServerList OFF
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30801,k8workerhost2.example.com:30801
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
    </Location>

    # OIM, role-sod profile
    <Location /role-sod>
      WLSRequest ON
      DynamicServerList OFF
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
    </Location>

    # Callback webservice for SOA. SOA calls this when a request is approved/rejected
    # Provide the SOA Managed Server Port
    <Location /workflowservice>
      WLSRequest ON
      DynamicServerList OFF
      WLCookieName oimjsessionid
      DynamicServerList OFF
      WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
      WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
    </Location>

    # used for FA Callback service.
    <Location /callbackResponseService>
      WLSRequest ON
```

```
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# spml xsd profile
<Location /spml-xsd>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# OIM, spml dsml profile
<Location /spmlws>
  WLSRequest ON
  DynamicServerList OFF
  PathTrim /weblogic
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /reqsvc>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>

# SOA Infra
<Location /soa-infra>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30801,k8workerhost2.example.com:30801
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/OHS/component/oim_component.log"
</Location>

# UMS Email Support
<Location /ucs>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30801,k8workerhost2.example.com:30801
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/OHS/component/oim_component.log"
</Location>

<Location /provisioning-callback>
  WLSRequest ON
  DynamicServerList OFF
```

```
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /CertificationCallbackService>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

<Location /IdentityAuditCallbackService>
  WLSRequest ON
  DynamicServerList OFF
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# SOA Callback webservice for SOD - Provide the SOA Managed Server Ports
<Location /soa/composer>
  SetHandler weblogic-handler
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30801,k8workerhost2.example.com:30801
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>

<Location /integration>
  SetHandler weblogic-handler
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30801,k8workerhost2.example.com:30801
  WLCookieName oimjsessionid
</Location>

<Location /sdpmessaging/userprefs-ui>
  SetHandler weblogic-handler
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30801,k8workerhost2.example.com:30801
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/soa_component.log"
</Location>

<Location /iam>
  SetHandler weblogic-handler
  WLCookieName oimjsessionid
  DynamicServerList OFF
  WebLogicCluster k8workerhost1.example.com:30140,k8workerhost2.example.com:30140
  WLLogFile "${ORACLE_INSTANCE}/diagnostics/logs/mod_wl/oim_component.log"
</Location>

# OIRI API
# <Location /oiri/api>
    WLSRequest ON
    DynamicServerList OFF
```

```
        WLCookieName oimjsessionid
        DynamicServerList OFF
        WebLogicCluster k8workerhost1.example.com:30305,k8workerhost2.example.com:30305
    </Location>
</VirtualHost>
```

18

# Configuring Oracle Access Manager Using WDT

Install and configure an initial domain, which can be used as the starting point for an enterprise deployment. Later, configure the domain.

A complete Oracle Identity and Access Management uses a split domain deployment, where there is a single domain for Oracle Access Management and a different domain for Oracle Identity Governance.

In version 4.1.2 of the WebLogic Kubernetes Operator, two different methods to create Oracle WebLogic domains are available. The traditional WLST method uses WLST scripts to create the domain which is the method employed in the Enterprise Deployment Guide for several releases.

Starting with this release, the Enterprise Deployment Guide will use the Weblogic Deployment Tools (WDT) to create the domains. The WDT uses templates to create domains which simplifies the installation procedure. For more information about WebLogic deployment tools, see WebLogic Deploy Tooling

This chapter includes the following topics:

- About the Initial Infrastructure Domain
  Before you create the initial Infrastructure domain, ensure that you review the key concepts.
- Installing Oracle Access Manager (OAM) on the Kubernetes Infrastructure
  Before creating the OAM Kubernetes infrastructure, you should have downloaded the Oracle Access Manager Image and installed the Oracle WebLogic Operator.
- Creating a Namespace for Oracle Access Manager
  Create a namespace to contain all the Oracle Access Manager Kubernetes objects.
- Creating a Container Registry Secret
  If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.
- Creating a Kubernetes Secret for Docker Hub Images
  This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands.
- Creating the Database Schemas for Access Manager
  Oracle Fusion Middleware components require schemas in a database, these schemas are created by the WebLogic Deployment Tools at the time of deployment.
- Creating the Oracle Access Manager Domain
  To configure the Oracle Access Manager domain, you should configure the WebLogic Operator for the domain namespace, create the Kubernetes secrets, and then create the Access domain.
- Updating the Domain
  When you first create the domain, some information may be missing. You can add this information using a simple `curl` script.

- Performing the Post-Configuration Tasks for Oracle Access Management Domain
  The post-configuration tasks for the OAM domain include creating the server overrides file and updating the data sources.
- Restarting the Domain
  Restart the domain for the changes to take effect.
- Validating the Administration Server
  Before you perform the configuration steps, validate that the Administration Server has started successfully by ensuring that you have access to the Oracle WebLogic Server Administration Console and Oracle Enterprise Manager Fusion Middleware Control, both installed and configured on the Administration Server.
- Removing OAM Server from WebLogic Server 12c Default Coherence Cluster
  Exclude all Oracle Access Management (OAM) clusters (including Policy Manager and OAM runtime server) from the default WebLogic Server 12*c* coherence cluster by using the WebLogic Server Administration Console.
- Tuning the WebLogic Server
  Tune the WebLogic Server for optimum performance by adding the Minimum Thread Constraint and removing the Max Thread and Capacity constraints.
- Enabling Virtualization
  You can use the Fusion Middleware Control to enable virtualization.
- Restarting the Domain
  Restart the domain for the changes to take effect.
- Configuring and Integrating with LDAP
  Configuration and integration of OAM with LDAP requires you to first set a global passphrase. You then configure OAM to use the LDAP directory. In addition, create the Webgate_IDM agent if not present and finally assign administration rights to the LDAP users to access the MBeans stored in the Administration Server.
- Updating WebGate Agents
  When you run `idmConfigTool`, it changes the default OAM security model and creates a new WebGate SSO agent. However, it does not change the existing WebGate SSO agents to the new security model. After you run `idmConfigTool`, you must update any WebGate agents that previously existed.
- Updating Host Identifiers
  When you access the domain, you enter using different load balancer entry points. You must add each of these entry points (virtual hosts) to the policy list. Adding these entry points ensures that if you request access to a resource using `login.example.com` or `prov.example.com`, you have access to the same set of policy rules.
- Adding the Missing Policies to OAM
  If any policies are missing, you have to add to ensure that Oracle Access Manager functions correctly.
- Validating the Authentication Providers
  Set the order of identity assertion and authentication providers in the WebLogic Server Administration Console.
- Configuring Oracle ADF and OPSS Security with Oracle Access Manager
  Some Oracle Fusion Middleware management consoles use Oracle Application Development Framework (Oracle ADF) security, which can integrate with Oracle Access Manager Single Sign-on (SSO). These applications can take advantage of Oracle Platform Security Services (OPSS) SSO for user authentication, but you must first configure the domain-level `jps-config.xml` file to enable these capabilities.

- Enabling Forgotten Password
  This section describes how to set up the One Time Pin forgotten password functionality which is provided with Oracle Access Manager
- Restarting the Access Domain
  Restart the Access domain in Kubernetes for the changes to take effect.
- Setting the Initial Server Count
  When you first created the domain, you specified that only one Managed Server should be started. After you complete the configuration, you can increase the initial server count to the actual number you require.
- Centralized Monitoring Using Grafana and Prometheus
- Centralized Log File Monitoring Using Elasticsearch and Kibana
- Backing Up the Configuration
  As a best practice, Oracle recommends you to back up the configuration after you have successfully extended a domain or at another logical point. Back up only after you have verified that the installation is successful so far. This is a quick backup to enable immediate restoration in case of problems in later steps.

# About the Initial Infrastructure Domain

Before you create the initial Infrastructure domain, ensure that you review the key concepts.

- About the Software Distribution
- Characteristics of the Domain
- Variables Used in this Chapter
- Kubernetes Services

## About the Software Distribution

You create the initial infrastructure domain for an enterprise deployment by using the Oracle Weblogic Operator. The Oracle Access Manager software is distrubted as a pre-built container image. See Identifying and Obtaining Software Distributions for an Enterprise Deployment. This distribution contains all of the necessary components to install and configure Oracle Access Manager.

See Understanding Oracle Fusion Middleware Infrastructure in *Understanding Oracle Fusion Middleware*.

## Characteristics of the Domain

The following table lists some of the key characteristics of the domain that you are about to create. Reviewing these characteristics helps you to understand the purpose and context of the procedures that are used to configure the domain.

Many of these characteristics are described in more detail in Understanding a Typical Enterprise Deployment.

| Characteristic of the Domain | More Information |
|---|---|
| Each WebLogic Server is placed into a pod in the Kubernetes cluster. | See About the Kubernetes Deployment. |

| Characteristic of the Domain | More Information |
|---|---|
| Places each Kubernetes domain object in a dedicated Kubernetes namespace. | See About the Kubernetes Deployment. |
| Uses Kubernetes services to interact with the WebLogic Managed Servers. | See Creating the Kubernetes Services. |
| Uses Kubernetes persistent volumes to hold the domain configuration. | See #unique_520. |
| Each Kubernetes pod is built from a pre-built Oracle container image. | See Identifying and Obtaining Software Distributions for an Enterprise Deployment. |
| Uses a per domain Node Manager configuration. | See About the Node Manager Configuration in a Typical Enterprise Deployment. |
| Requires a separately installed LDAP-based authentication provider. | See Installing and Configuring Oracle Unified Directory. |
| Certificates are stored in Oracle Keystore Service. | See Configuring Oracle OPSS Keystore Service. |

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 18-1 The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| <*REGISTRY_ADDRESS*> | `iad.ocir.io/<mytenancy>` | The location of the container registry. |
| <*REGISTRY_SECRET_NAME*> | `regcred` | The name of the Kubernetes secret containing the container registry credentials. Required only if you are pulling images directly from a container registry. See Creating a Container Registry Secret. |
| <*REG_USER*> | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The name of the user you use to log in to the container registry. |
| <*REG_PWD*> | <*password*> | The container registry user password. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OAM_REPOSITORY>* | `oracle/oam`<br>`local/oracle/oam`<br>`container-registry.oracle.com/middleware/oam_cpu`<br>*`<REGISTRY_ADDRESS>`*`/oracle/oam` | The name of the OAM software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/oam`. If you use OLCNE, the value will be `local/oracle/oam`.<br>If you are using the Oracle container registry, the value will be: `container-registry.oracle.com/middleware/oam_cpu`.<br>If you are using a container registry, the value will be the name of the registry with the product name: *`<REGISTRY_ADDRESS>`*`/oracle/oam`. |
| *<OAM_VER>* | `12.2.1.4-jdk8-ol7-220418.0839`<br>`latest` | The version of the image you want to use. The value will be the version you have downloaded and staged either locally or in the container registry. |
| *<PVSERVER>* | `nfsserver.example.com` | The name or IP address of the NFS server.<br>**Note**: This name should be resolvable inside the Kubernetes cluster. |
| *<OAMNS>* | `oamns` | The domain namespace to be used to store OAM objects. |
| *<WORKDIR>* | `workdir/OAM` | The location where you want to create the working directory for OAM. |
| *<K8_WORKDIR>* | `/u01/oracle/user_projects/workdir` | Working directory inside the Kubernetes container. |
| *<OAM_SHARE>* | `/exports/IAMPVS/oampv` | The NFS export for the persistence store. |
| *<OAM_DB_SCAN>* | `dbscan.example.com` | The SCAN address of the database cluster. |
| *<OAM_DB_LISTENER>* | `1521` | The database listener port number. |
| *<OAM_DB_SERVICE>* | `iadedg.example.com` | The name of the database service to use. |
| *<OAM_DB_SYS_PWD>* | `MySysPWD__001` | The SYS password for the database. |
| *<OAM_RCU_PREFIX>* | `IADEDG` | The prefix used when the database schemas are created. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OAM_COOKIE_DOMAIN>* | `.example.com` | The domain you want to associate the OAM cookie with this is normally the same as the *<LDAP_SEARCHBASE>* in the domain format. |
| *<OAM_OIG_INTEG>* | `false` | If you are intending Oracle Identity Governance to handle forgotten password functionality, set this parameter to `true`. If you are using the new OAM forgotten password functionality, set the value to `false`. |
| *<OAM_SCHEMA_PWD>* | `MySchemaPWD__001` | The password you want to set for the product schemas being created. |
| *<OAM_DOMAIN_NAME>* | `accessdomain` | The name of the Kubernetes domain to be created. |
| *<OAM_DOMAIN_SECRET>* | `accessdomain-weblogic-credentials` | The name of the secret you want to create, for the namespace that is used. The name of the secret must be `<OAM_DOMAIN_NAME>-weblogic-credentials`. |
| *<OAM_ADMIN_LBR_HOST>* | `iadadmin.example.com` | The virtual host of the OAM administration functions. |
| *<OAM_RCU_SECRET>* | `accessdomain-rcu-credentials` | The name of the RCU secret. The name of the secret must be `<OAM_DOMAIN_NAME>-rcu-credentials`. See Creating the RCU Secret. |
| *<OAM_LOGIN_LBR_HOST>* | `login.example.com` | The name of your login load balancer virtual name. |
| *<OAM_LOGIN_LBR_PROTOCOL>* | `https` | The type of access protocol used for the login load balancer. In an SSL terminated environment, this value will be `https`. |
| *<OAM_LOGIN_LBR_PORT>* | `443` | The port of the login load balancer virtual name. |
| *<OAM_OAP_HOST>* | `oap.example.com`<br>`accessdomain-oap.oamns.svc.cluster.local`. | An externally resolvable host name for legacy WebGate communications. This host can be either a Kubernetes worker node or a load balancer entry point that points to multiple Kubernetes worker nodes.<br>If you are only interacting inside the Kubernetes cluster, the host name will be `<OAM_DOMAIN_NAME>-oap.<OAMNS>.svc.cluster.local`. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OAP_MODE>* | `open` or `cert` | The OAP security mode you want to use if you are using the native OAP calls from WebGate. Oracle recommends that you conduct interactions with OAM using OAP over REST, in which case the transport mode is not used. In these scenarios, set the *<OAP_MODE>* to `Open` so that you can remove the need to manage certificates. |
| *<WG_CONNECTIONS>* | `20` | The maximum number of connections supported by the WebGate agent. |
| *<OAM_SERVER_COUNT>* | `5` | The number of Managed Servers required. Oracle highly recommends you to set this value to a number greater than the anticipated need in the system's lifetime. It creates a number of server definitions in the WebLogic domain and ensures that you have a simple mechanism to scale up the system when the demand increases. This value does not reflect the number of server instances you actually start with; it just enables you to start additional servers if your needs change. Adding additional server definitions post domain creation is a complex task and should be avoided, if possible. |
| *<OAM_INITIAL_SERVERS>* | `1` | The number of Managed Servers to start. Oracle recommends you to set this value to `1` for the the duration of the configuration. |
| *<OAM_MAX_CPU>* | `1` | The maximum number of CPUs each `OAM_SERVER` pod is allowed to consume. CPU is measured in CPU cores. The value of 1 is equal to 1 CPU core or 1 virtual core. |
| *<OAM_CPU>* | `500m` | The initial number of CPUs each `OAM_SERVER` pod is allowed to consume. It is measured in CPU cycles and the value of 1000m is equal to 1 CPU core or 1 virtual core. CPU is measured in CPU cores and the value of 1 is equal to 1 CPU core or 1 virtual core. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OAM_MAX_MEMORY>* | `8Gi` | The maximum amount of memory that the `OAM_SERVER` pods are allowed to consume. Memory is measured in standard units where 1G is equal to 1Gi. |
| *<OAM_MEMORY>* | `2Gi` | The initial amount of memory that the `OAM_SERVER` pods are allowed to consume. Memory is measured in standard units where 1G is equal to 1Gi. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OAMSERVER_JAVA_PARAMS>* | `-Xms2048m -Xmx8192m` | The maximum (Xmx) and minimum heap size allocated to each `OAM_SERVER`. The size is represented as a number of Mb.<br><br>**Note:** The maximum amount of heap size must be less than the maximum amount allowed to be used by the Pod *<OAM_MAX_MEMORY>* |
| *<LDAP_HOST>* | `edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local`<br><br>For OUD, this value will be `<OUD_PREFIX>-oud-ds-rs-lbr-ldap.<OUDNS>.svc.cluster.local`. | The name of the host where the LDAP directory is running. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<LDAP_PORT>* | `1389` (OUD) | The port used to connect to LDAP. If your directory is stored in a Kubernetes environment, this will be the internal Kubernetes service port. If your directory is outside of Kubernetes, it will be the regular LDAP port. |
| *<LDAP_ADMIN_USER>* | `cn=oudadmin` | The user name of the directory administrator. |
| *<LDAP_SEARCHBASE>* | `dc=example,dc=com` | The directory tree for your organization. This is where all the data is stored. |
| *<LDAP_GROUP_SEARCHBASE>* | `cn=Groups,dc=example,dc=com` | The location in the directory where groups/roles are stored. |
| *<LDAP_USER_SEARCHBASE>* | `cn=Users,dc=example,dc=com` | The location in the directory where names of users are stored. |
| *<LDAP_SYSTEMIDS>* | `cn=systemids,dc=example,dc=com` | The name of a container where you want to store system ids. The user names placed in this container are not subject to OIM reconciliation or password aging. This container is reserved for users such as *<LDAP_OAMLDAP_USER>* and *<LDAP_OIGLDAP_USER>*. |
| *<LDAP_TYPE>* | `OUD` | It is the type of directory: OUD or OID. |
| *<LDAP_WLSADMIN_USER>* | `weblogic_iam` | The name of the user who administers the WebLogic domain.<br>This name is an LDAP user name used for single sign-on authentication. The *<OAM_WEBLOGIC_USER>* is the internal WebLogic user name assigned during the domain creation process. |
| *<LDAP_OAMADMIN_USER>* | `oamadmin` | The name of the user who administers OAM. |
| *<LDAP_OAMLDAP_USER>* | `oamLDAP` | The name of a user that OAM will use to connect to the directory for validating logins. |
| *<LDAP_WLSADMIN_GRP>* | `WLSAdministrators` | Users assigned to this role will be able to log in to the WebLogic Administration Console and FMW Control. |
| *<LDAP_OAMADMIN_GRP>* | `OAMAdministrators` | Users assigned to this role will be able to log in to the OAM Administration Console and configure OAM. |

**Table 18-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OAM_OAP_SERVICE_PORT>* | 30540 | The Kubernetes service port for the OAM OAP requests. |
| *<OAM_ADMIN_PORT>* | 7001 | The internal port assigned to the OAM Administration Server. |
| *<OAM_OAP_PORT>* | 5575 | The internal OAP port number.<br>If you are using the Kubernetes service, this value can be the internal port number. |
| *<OAP_SERVICE_PORT>* | 30540 | The Kubernetes service port which fronts the OAM OAP cluster nodes. If you are using the Kubernetes service, this value can be the internal port number. |
| *<OAM_EXT_T3_PORT>* | 30012 | The external T3 port. |
| *<OAM_OAM_K8>* | 30410 | The Kubernetes service port for the OAM Managed Servers. |
| *<OAM_POLICY_K8>* | 30510 | The Kubernetes service port for the OAM Policy Manager Servers. |
| *<OAM_ADMIN_K8>* | 30701 | The external OAM WebLogic Kubernetes service port for the external WebLogic Administration Server. |
| *<ELK_HOST>* | https://elasticsearch-es-http.elkns.svc:9200 | The host and port of the centralized Elasticsearch deployment. This host can be inside the Kubernetes cluster or external to it. This host is used only when Elasticsearch is used. |
| *<ELK_VER>* | 8.11.0 | The version of Elasticsearch you want to use. |

# Kubernetes Services

If you are using NodePort Services, the following Kubernetes services are created as part of the deployment:

**Table 18-2 Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| accessdomain-oam-policy-NodePort | NodePort | 30510 | 15000 |
| accessdomain-oam-oam-NodePort | NodePort | 30410 | 14100 |
| accessdomain-adminserver-external | NodePort | 30701 | 7001 |

**Table 18-2 (Cont.) Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| accessdomain-adminserver | ClusterIP | 7001<br>3012<br>**Note**: 3012 is the T3 Service port. | 7001/3012 |

If you use an Ingress-based deployment, the following Ingress services will be created as part of this deployment:

**Table 18-3 Ingress Services**

| Service Name | Host Name |
|---|---|
| oamadmin-ingress | iadadmin.example.com |
| oamruntime-ingress | login.example.com |

# Installing Oracle Access Manager (OAM) on the Kubernetes Infrastructure

Before creating the OAM Kubernetes infrastructure, you should have downloaded the Oracle Access Manager Image and installed the Oracle WebLogic Operator.

- To download the Oracle Access Manager Image and load it into the container image repository (this repository must be visible to each Kubernetes node), see Identifying and Obtaining Software Distributions for an Enterprise Deployment.
- To install the Oracle WebLogic Operator, see Installing and Configuring WebLogic Kubernetes Operator.
- Prerequisites
  Before creating the Oracle Access Manager (OAM) on the kubernetes infrastructure, you must download the Oracle Access Manager container image and installed the Oracle WebLogic Operator.

## Prerequisites

Before creating the Oracle Access Manager (OAM) on the kubernetes infrastructure, you must download the Oracle Access Manager container image and installed the Oracle WebLogic Operator.

- To download the Oracle Access Manager Image and load it into the Docker image repository (this repository must be visible to each Kubernetes node), see Identifying and Obtaining Software Distributions for an Enterprise Deployment.
- To install the Oracle WebLogic Operator, see Installing and Configuring WebLogic Kubernetes Operator.
- Setting Up a Product Specific Work Directory

## Setting Up a Product Specific Work Directory

Before you begin the installation, you must have downloaded and staged the Oracle Access Manager container image and the sample code repository. See Identifying and Obtaining Software Distributions for an Enterprise Deployment.

You must also have deployed the Oracle WebLogic Operator as described in Installing the WebLogic Kubernetes Operator.

This section describes the procedure to copy the downloaded sample deployment scripts to a temporary working directory on the configuration host for OAM.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubernetes cluster.

   ```
   mkdir -p /<WORKDIR>
   ```

   For example:

   ```
   mkdir -p /workdir/OAM
   ```

2. Change directory to this location:

   ```
   cd /workdir/OAM
   ```

3. Copy the sample scripts to the work directory.

   ```
   cp -R <WORKDIR>/fmw-kubernetes/OracleAccessManagement/kubernetes <WORKDIR>/
   samples
   ```

   For example:

   ```
   cp -R /workdir/OAM/fmw-kubernetes/OracleAccessManagement/kubernetes /
   workdir/OAM/samples
   ```

# Creating a Namespace for Oracle Access Manager

Create a namespace to contain all the Oracle Access Manager Kubernetes objects.

1. To create a namespace, use the following command:

   ```
   kubectl create namespace oamns
   ```

   The output appears as follows:

   ```
   namespace/oamns created
   ```

2. Tag the namespace so that the WebLogic Kubernetes Operator can manage it.

   ```
   kubectl label namespaces oamns weblogic-operator=enabled
   ```

# Creating a Container Registry Secret

If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

If you have staged your container images locally, there is no need to perform this step.

To create a container registry secret, use the following command:

```
kubectl create secret -n <OAMNS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n oamns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

# Creating a Kubernetes Secret for Docker Hub Images

This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands.

> **Note:**
>
> If you are pulling the images from your own container registry, then this step is not required.

You should have an account on `hub.docker.com`. If you want to stage the images in your own repository, you can do so and modify the `helm` override file as appropriate.

To create a Kubernetes secret for `hub.docker.com`, use the following command:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="<DH_USER>" --docker-
password="<DH_PWD>" --namespace=<OAMNS>
```

For example:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="username" --docker-
password="<mypassword>" --namespace=oamns
```

# Creating the Database Schemas for Access Manager

Oracle Fusion Middleware components require schemas in a database, these schemas are created by the WebLogic Deployment Tools at the time of deployment.

The tool creates the following schemas:

- Metadata Services (MDS)
- Audit Services (IAU)
- Audit Services Append (IAU_APPEND)
- Audit Services Viewer (IAU_VIEWER)
- Oracle Platform Security Services (OPSS)
- User Messaging Service (UMS)
- WebLogic Services (WLS)
- Common Infrastructure Services (STB)
- Oracle Access Manager (OAM)

For more information about RCU and how the schemas are created and stored in the database, see Preparing for Schema Creation in *Creating Schemas with the Repository Creation Utility*.

You must install and configure a certified database and ensure that the database is up and running.

See Preparing an Existing Database for an Enterprise Deployment

# Creating the Oracle Access Manager Domain

To configure the Oracle Access Manager domain, you should configure the WebLogic Operator for the domain namespace, create the Kubernetes secrets, and then create the Access domain.

- Creating the Kubernetes Secrets
- Creating the Access Domain
- Creating the Kubernetes Services

## Creating the Kubernetes Secrets

Rather than passing the credentials directly into the domain creation process, you can use the Kubernetes secrets to store the credentials in the encrypted format. The WebLogic Operator reads these secrets instead of asking for credentials.

- Creating the Domain Secret
- Creating the RCU Secret

### Creating the Domain Secret

The domain secret contains information about the WebLogic Administration user who creates the domain.

1. Use the following command to create the domain secret:

```
cd <WORKDIR>/samples/create-access-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "username=<OAM_WEBLOGIC_USER>" -l
"password=<OAM_WEBLOGIC_PWD>" -n <OAMNS> -d <OAM_DOMAIN_NAME> -s
<OAM_DOMAIN_SECRET>
```

For example:

```
cd /workdir/OAM/samples/create-access-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "username=weblogic" -l "password=mypassword" -n
oamns -d accessdomain -s accessdomain-weblogic-credentials
```

The output appears as follows:

```
@@ Info: Setting up secret 'accessdomain-weblogic-credentials'.
secret/accessdomain-credentials created
secret/accessdomain-credentials labeled
```

2. Verify that the secret has been created, using the following command:

```
kubectl get secret accessdomain-weblogic-credentials -o yaml -n oamns
```

## Creating the RCU Secret

The RCU secret is used by the WebLogic Operator to determine how to connect to the database schemas that you have already created. See Creating the Database Schemas for Access Manager.

To create the RCU secret, perform the following steps:

1. Use the following command:

```
cd <WORKDIR>/samples/create-access-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "rcu_prefix=<OAM_RCU_PREFIX>" -l
"rcu_schema_password=<OAM_SCHEMA_PWD>" -l "db_host=<DB_HOST>" -l
"db_port=<DB_PORT>" -l "db_service=<OAM_DB_SERVICE>" -l "dba_user=sys" -l
"dba_password=<OAM_SYS_PWD>" -n <OAMNS> -d <OAM_DOMAIN_NAME> -s
<OAM_RCU_SECRET>
```

For example:

```
cd /workdir/OAM/samples/create-access-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "rcu_prefix=IADEDG" -l
"rcu_schema_password=MySchemaPWD__001" -l "db_host=DB-SCAN.example.com" -l
```

```
"db_port=1521" -l "db_service=oam_s.example.com" -l "dba_user=sys" -l
"dba_password= MySysPWD__001" -n oamns -d accessdomain -s accessdomain-rcu-
credentials
```

The output appears as follows:

```
@@ Info: Setting up secret 'accessdomain-rcu-credentials'.
secret/accessdomain-rcu-credentials created
secret/accessdomain-rcu-credentials labeled
```

2. Verify that the secret has been created, using the command:

```
kubectl get secret accessdomain-rcu-credentials -o yaml -n oamns
```

# Creating the Access Domain

The procedure to create the Access domain includes creating the domain configuration file, creating the domain using the WebLogic Kubernetes Operator, setting the memory parameters, initializing the domain, and verifying the domain.

- Creating the Domain Configuration File
- Generating WDT Auxiliary Image
- Updating domain.yaml
- Creating the Domain Using the WebLogic Operator
- Verifying the Domain

## Creating the Domain Configuration File

A configuration file is used to tell the WebLogic Operator how to create the domain. This configuration file is named `create-domain-wdt.yaml` and is located in `<WORKDIR>/samples/create-access-domain/domain-home-on-pv/wdt-utils/generate_models_utils/create-domain-wdt.yaml`.

1. Create a copy of the `/workdir/OAM/create-domain-wdt.yaml` file. For example:

```
cp /workdir/OAM/samples/create-access-domain/domain-home-on-pv/wdt-utils/
generate_models_utils/create-domain-wdt.yaml /workdir/OAM
```

2. Update the following values in the `create-domain-wdt.yaml` file:

```
domainUID: <OAM_DOMAIN_NAME>
domainPVMountPath: /u01/oracle/user_projects/
domainHome: /u01/oracle/user_projects/domains/<OAM_DOMAIN_NAME>
image: <OAM_REPOSITORY>:<OAM_VER>
imagePullSecretName: <REGISTRY_SECRET_NAME>
namespace: <OAMNS>
logHome: /u01/oracle/user_projects/domains/logs/<OAM_DOMAIN_NAME>
weblogicDomainStorageNFSServer: <PVSERVER>
weblogicDomainStorageType: NFS
weblogicDomainStoragePath: <OAM_SHARE>
edgInstall: true
oamServerJavaParams: <OAMSERVER_JAVA_PARAMS>
```

```
oamMaxCPU: <OAM_MAX_CPU>
oamCPU: <OAM_CPU>
oamMaxMemory: <OAM_MAX_MEMORY>
oamMemory: <OAM_MEMORY>
exposeAdminNodePort: true
configuredManagedServerCount: <OAM_SERVER_COUNT>
initialManagedServerReplicas: <OAM_INITIAL_SERVERS>
productionModeEnabled: true
exposeAdminT3Channel: true
adminNodePort: <OAM_ADMIN_K8>
datasourceType: agl
```

Where:

- `domainUID` is the name you wish to assign to the domain.
- `domainPVMountPath` is the location inside the container where the persistent volume is mounted.
- `logHome` is the location of the log files.
- `exposeAdminNodePort` is the parameter that should be set to '`True`' to enable direct access to the Administration Server outside of the Kubernetes cluster. This is required for the post domain configuration regardless of whether or not an Ingress controller is used.
- `productionModeEnabled` is the parameter that should be set to '`True`'.
- `exposeAdminT3Channel` is the parameter that should be set to '`True`' to enable T3 to communicate outside of the domain. This communication is used to configure the domain after it is created.
- `datasourceType` is the type of datasource to create. AGL (Active Grid Link) is the recommended option for maximum availability.
- `edgInstall` identifies the installation as an EDG type deployment. Setting this value allows the installation to implement best practice rather than this being an after deployment exercise.

For example:

```
# Copyright (c) 2024, Oracle and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at
https://oss.oracle.com/licenses/upl.

# The version of this inputs file. Do not modify.
version: create-weblogic-sample-domain-inputs-v1

# Port number for admin server
adminPort: 7001

# Unique ID identifying a domain.
# This ID must not contain an underscope ("_"), and must be lowercase and
unique across all domains in a Kubernetes cluster
domainUID: accessdomain

# Number of managed servers to generate for the domain
configuredManagedServerCount: 5
```

```
# Number of managed servers to initially start for the domain
initialManagedServerReplicas: 1

# Boolean indicating if production mode is enabled for the domain
productionModeEnabled: true

# Port for the T3Channel of the NetworkAccessPoint
t3ChannelPort: 30012

# dataSource Type
# supported values are agl or generic. Use agl for active gridlink type of
datasource
# generic datasource is not applicable for RAC DB.
datasourceType: agl

# If set to true, generated model will have EDG recommended datasource
type i.e. AGL and necessary connection pool parameters.
edgInstall: true

# Home of the WebLogic domain
domainHome: /u01/oracle/user_projects/domains/accessdomain
# OAM Docker image.
image: iad.ocir.io/<mytenancy>/idm/oam:12.2.1.4-jdk8-ol8-240112

# Name of the Kubernetes secret to pull the images from container registry.
# The presence of the secret will be validated when this parameter is
enabled.
imagePullSecretName: regcred

# The in-pod location for domain log, server logs, server out, and node
manager log files
logHome: /u01/oracle/user_projects/domains/logs/accessdomain

# Public address for T3Channel of the NetworkAccessPoint. This value
should be set to the
# kubernetes server address, which you can get by running "$
{KUBERNETES_CLI:-kubectl} cluster-info". If this
# value is not set to that address, WLST will not be able to connect from
outside the
# kubernetes cluster.
# t3PublicAddress:
# Boolean to indicate if the channel should be exposed as a service
exposeAdminT3Channel: true

# NodePort to expose for the admin server
adminNodePort: 30701

# Boolean to indicate if the adminNodePort will be exposed
exposeAdminNodePort: true

# Name of the domain namespace
namespace: oamns

#Java Option for WebLogic Server #NOT IN EDG
javaOptions: -Dweblogic.StdoutDebugEnabled=false
```

```
# Mount path of the domain persistent volume.
domainPVMountPath: /u01/oracle/user_projects
## Persistent volume type for the persistent storage.
## The value must be 'HOST_PATH' or 'NFS'.
## If using 'NFS', weblogicDomainStorageNFSServer must be specified.
weblogicDomainStorageType: NFS
#
## The server name or ip address of the NFS server to use for the
persistent storage.
## The following line must be uncomment and customized if
weblogicDomainStorateType is NFS:
weblogicDomainStorageNFSServer: nfsserver.example.com
#
## Physical path of the persistent storage.
## When weblogicDomainStorageType is set to HOST_PATH, this value should
be set the to path to the
## domain storage on the Kubernetes host.
## When weblogicDomainStorageType is set to NFS, then
weblogicDomainStorageNFSServer should be set
## to the IP address or name of the DNS server, and this value should be
set to the exported path
## on that server.
## Note that the path where the domain is mounted in the WebLogic
containers is not affected by this
## setting, that is determined when you create your domain.
## The following line must be uncommented and customized:
weblogicDomainStoragePath: /exports/IAMPVS/oampv
#
## Reclaim policy of the persistent storage
## The valid values are: 'Retain', 'Delete', and 'Recycle'
weblogicDomainStorageReclaimPolicy: Retain
#
## Total storage allocated to the persistent storage.
weblogicDomainStorageSize: 10Gi
#
# Pod Resource Allocation
#
oamServerJavaParams: -Xms2048m -Xmx8192m
# Max CPU Cores pod is allowed to consume.
oamMaxCPU: 1
# Initial CPU Units 1000m = 1 CPU core
oamCPU: 500m
 # Max Memory pod is allowed to consume.
oamMaxMemory: 8Gi
# Initial Memory allocated to pod.
oamMemory: 2Gi
```

3. Save the configuration file.

## Generating WDT Auxiliary Image

While creating a domain using the WebLogic deployment tool, a dedicated image is created that describes the deployment. This is based on the domain creation file described in Creating the Domain Configuration File. This image is then stored in a local container registry.

The benefit of using an auxiliary image with the configuration is that it can be used repeatedly to create multiple environments with slightly different properties. For example, the same image file can be used to create a development, testing, and production environment where only the database connection details vary. You need not create a new image each time you create a similar environment. This image must be stored in a registry where images are loaded, and you need have access to this registry.

The following sections describe how to generate the WDT model files, create an auxiliary image, and upload it to your repository.

### Generating WDT Model Files

Perform the following steps to generate the WDT model files from the Domain Configuration file:

1. Change the directory to WDT utils directory of the samples downloaded.

   ```
   cd <WORKDIR>/samples/create-access-domain/domain-home-on-pv/wdt-utils/
   generate_models_utils
   ```

   For Example:

   ```
   cd /workdir/OAM//samples/create-access-domain/domain-home-on-pv/wdt-utils/
   generate_models_utils
   ```

2. Generate the model files using the `generate_wdt_models.sh` utility.

   ```
   ./generate_wdt_models.sh -i <WORKDIR>/create-domain-wdt.yaml -o <WORKDIR>
   ```

   Use `-i` to specify the location of the Domain configuration file you created in Creating the Domain Configuration File.

   Use `-o` to specify where the WDT Model files and templates should be created.

   For Example:

   ```
   ./generate_wdt_models.sh -i /workdir/OAM/create-domain-wdt.yaml -o /
   workdir/OAM
   ```

   After running the utility, a directory is created called `weblogic-domains` which contain the generated files.

Sample Output with Input Parameters

```
export version="create-weblogic-sample-domain-inputs-v1"
export adminPort="7001"
export domainUID="accessdomain"
export configuredManagedServerCount="5"
export initialManagedServerReplicas="1"
export productionModeEnabled="true"
export t3ChannelPort="30012"
export datasourceType="agl"
export edgInstall="true"
export domainHome="/u01/oracle/user_projects/domains/accessdomain"
export image="iad.ocir.io/<mytenancy>/oam:12.2.1.4-jdk8-ol8-240112"
```

```
export imagePullSecretName="regcred"
export logHome="/u01/oracle/user_projects/domains/logs/accessdomain"
export exposeAdminT3Channel="true"
export adminNodePort="30701"
export exposeAdminNodePort="true"
export namespace="oamns"
javaOptions=-Dweblogic.StdoutDebugEnabled=false
export domainPVMountPath="/u01/oracle/user_projects"
export weblogicDomainStorageType="NFS"
export weblogicDomainStorageNFSServer="nfsserver.example.com"
export weblogicDomainStoragePath="/exports/IAMPVS/oampv"
export weblogicDomainStorageReclaimPolicy="Retain"
export weblogicDomainStorageSize="10Gi"
export oamServerJavaParams="-Xms2048m -Xmx8192m"
export oamMaxCPU="1"
export oamCPU="500m"
export oamMaxMemory="8Gi"
export oamMemory="2Gi"
validateWlsDomainName called with accessdomain
WDT model file, property file and sample domain.yaml are genereted
successfully at /workdir/OAM/weblogic-domains/accessdomain
```

### Creating Image Property File

After the model files are created, they need to be added to an image, and uploaded to your registry which begins with describing the target registry in a property file.

Perform the following steps to create an image property file:

1. Run the following command to ensure java is installed on your machine:

   ```
   which java
   ```

2. Copy the property file to your work directory.

   ```
   cp <WORKDIR>/samples/create-access-domain/domain-home-on-pv/wdt-utils/
   build-domain-creation-image/properties/build-domain-creation-
   image.properties <WORKDIR>
   ```

   For Example:

   ```
   cp/workdir/OAM/samples/create-access-domain/domain-home-on-pv/wdt-utils/
   build-domain-creation-image/properties/build-domain-creation-
   image.properties /workdir/OAM
   ```

3. Edit the file `build-domain-creation-image.properties` and add the following values:
   `JAVA_HOME` set this to the location of your JAVA installation found in *Step 1*.

   For Example,

   ```
   /usr
   ```

   .

   - `REPOSITORY` set this to the location in your registry where the image file is to reside.

For Example,

```
iad.ocir.io/<mytenancy>/idm/oam_wdt
```

.

Where `oam_wdt` is the name of the image you wish to create.

- `IMAGE_TAG` used to assign a tag to the uploaded image, you can use anything here. In case of this example, we can use `<OAM_DOMAIN_NAME>`.
- `IMAGE_PUSH_REQUIRES_AUTH` must be set to true if you do not allow non-authenticated uploads to your registry.
- `REG_USER` must be set to the user in your registry where you wish to upload the image. This user must have upload privileges.
- `WDT_MODEL_FILE` must be set to the file `oam.yaml` which was generated in the step above. For example, `<WORKDIR>/weblogic-domains/<OAM_DOMAIN_NAME>/oam.yaml`.
- `WDT_VARIABLE_FILE` must be set to the file `oam.properties` which was generated in the step above. For example, `<WORKDIR>/weblogic-domains/<OAM_DOMAIN_NAME>/oam.properties`.
- `REG_PWD` must be set to the password of the above user and placed in a separate file in `buiidpwd` in the `<WORKDIR>`as shown below:

  ```
  REG_PASSWORD="<mypwd>"
  ```

  Sample `build-domain-creation-image.properties`

```
# Copyright (c) 2024, Oracle and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at
https://oss.oracle.com/licenses/upl.
# Input Property file for build-domain-creation-image.sh script

#
# set the JAVA_HOME environment variable to match the location of your
Java installation. Java 8 or newer is required
#
JAVA_HOME=/usr

#
# Image Details
#
#Set the IMAGE_TAG, default oam-aux-v1 if not set.
IMAGE_TAG=accessdomain
# Set the BASE_IMAGE, default ghcr.io/oracle/oraclelinux:8-slim if not
set.
BASE_IMAGE=ghcr.io/oracle/oraclelinux:8-slim

#
# Container Registry
#
#Image will be created with REPOSITORY:IMAGE_TAG
REPOSITORY=iad.ocir.io<mytenancy>idm/oam_wdt
# Container registry username
```

```
REG_USER=<mytenancy>/oracleidentitycloudservice/my.user@example.com
#Set it to false if authentication is not required for pushing the
image to registry, for example docker login already done in the host
before invoking the script.
IMAGE_PUSH_REQUIRES_AUTH=true

#
# WDT and WIT Variables
#
#Full path to wdt model files
WDT_MODEL_FILE=/workdir/OAM/weblogic-domains/accessdomain/oam.yaml
#Full path to wdt variable files
WDT_VARIABLE_FILE=/workdir/OAM/weblogic-domains/accessdomain/
oam.properties
#Full path to wdt archive files
WDT_ARCHIVE_FILE=""
#If not set, Latest version will be used.
WDT_VERSION="3.5.3"
#If not set, latest will be used during every fresh run
WIT_VERSION="1.12.1"

#In Most cases, no need to use these parameters. Please refer https://
oracle.github.io/weblogic-image-tool/userguide/tools/create-aux-image/
for
details about them.
TARGET=""
CHOWN=""
```

### Uploading WDT Auxiliary Image

Use the utility `build-domain-creation-image.sh` to create and upload the Auxiliary image:

For Example:

```
cd <WORKDIR>/samples/create-access-domain/domain-home-on-pv/wdt-utils/build-
domain-creation-image
```

```
./build-domain-creation-image.sh -i <WORKDIR>/build-domain-creation-
image.properties -p <WORKDIR>/.buildpwd
```

For Example:

```
cd /workdir/OAM/samples/create-access-domain/domain-home-on-pv/wdt-utils/
build-domain-creation-image
```

```
./build-domain-creation-image.sh -i /workdir/OAM/build-domain-creation-
image.properties -p /workdir/OAM/.buildpwd
```

Extract from Sample Output

```
[INFO ] Build successful. Build time=74s. Image tag=iad.ocir.io/
<mytenancy>/idm/oam_wdt:accessdomain
```

```
Getting image source signatures
Copying blob
sha256:432308aaf1ccdd6c69ff6e6f6d6c762e55e183284ca57d31228bd3578275f9a9
Copying blob
sha256:8b4d3bacf0d79476c744efb9d80fc05c5e1298b2ce8c5ed88edc9a4a01198ba9
Copying blob
sha256:c5a8db0bbcb50dce5017361d8a2c11f42b221f9e6842d439b562657a6669cc2a
Copying blob
sha256:812776fce264cf4a8e82c7d839ba60603e449b47f52582b0a5d68e730fc0b01e
Copying blob
sha256:0bcef3ba673ac9fd91ac95ae8c19fa3cb29a9ad107bec305e8772e2cc968ce2a
Copying blob
sha256:b782e2701e7e72e55c4cd2e849f9dc9baf602d7eec7eb89ffda3329f9b784f36
Copying blob
sha256:1d8523ddf53e8404bc9d4020673d36854e721cd878e209e2649acac359b2555a
Copying blob
sha256:cd00e719df55595b05a92e0741a09ad88a07c0c67caa65c78902f3c00214c72f
Copying blob
sha256:520d935025c94d503a6d9f31b029f5ee4c2f4b8a8326d94dbf2caa80f8c71151
Copying blob
sha256:9a9ca1edb11ff224553c91c7cf5f032f68db7a5f45080b567db3d6e9dee25e4e
Copying blob
sha256:1a47311837bde5a83fcb02ba004ed5f015c2c3d73172a7082126417db874bd1b
Copying config
sha256:1ae5d3f21cd522491aff21083c6618f954f6a5684b31b958c28d23b7b8c096af
Writing manifest to image destination
Storing signatures
Pushed image iad.ocir.io/<mytenancy>/idm/oam_wdt:accessdomain to image
repository
```

## Updating domain.yaml

While generating the WDT model files, a file called `domain.yaml` was created in the directory `<WORKDIR>/weblogic-domains/<OAM_DOMAIN_NAME>`. This file is used to create the WebLogic domain. Before using this file the auxiliary image you create must be added to the file by editing

`domain.yaml`

Locate the variable in the file `%DOMAIN_CREATION_IMAGE%` and replace it with the name of your image as `<REPOSITORY>:<IMAGE_TAG>` obtained from the file `build-domain-creation-image.properties`.

For example,

```
iad.ocir.io/mytenancy/idm/oam_wdt:accessdomain
```

> **Note:**
>
> If the registry where your image is located is different to the registry where your OAM image is stored then create a new secret with the credentials for the Auxiliary image registry using a different name to the main registry.
>
> For example:
>
> ```
> kubectl create secret -n oamns docker-registry regcred2 --docker-
> server=iad.ocir.io/mytenancy2 --docker-username=mytenancy/
> oracleidentitycloudservice/myemail@email.com --docker-
> password=<password>
> ```

Update the file `domain.yaml` and replace the lines with the name of your new secret.

Add additional secret name if you are using a different registry for domain creation image.

Identify which secret contains the credentials for pulling an image.

```
imagePullSecrets:

- name: regcred2
```

## Creating the Domain Using the WebLogic Operator

Create the domain using the following command:

```
cd <WORKDIR>/weblogic-domains/<OAM_DOMAIN_NAME>

kubectl create -f <WORKDIR>/weblogic-domains/<OAM_DOMAIN_NAME>/domain.yaml
```

For example:

```
cd /workdir/OAM/weblogic-domains/accessdomain

kubectl create -f /workdir/OAM/weblogic-domains/accessdomain/domain.yaml
```

Use the following to monitor the domain creation:

```
kubectl logs -n <OAMNS> <OAM_DOMAIN_DOMAIN>-introspector

kubectl describe domain -n <OAMNS> <OAM_DOMAIN_NAME>
```

For Example:

```
kubectl logs -n oamns accessdomain-introspector


kubectl describe domain -n oamns accessdomain
```

For more information, see the WebLogic operator logs.

For Example:

```
kubectl logs -n opns weblogic-operator-688f5dcdc4-qxnnz | grep
<OAM_DOMAIN_NAME>
```

After the domain is created, the OAM Kubernetes pods is started automatically and can be viewed using the command:

```
kubectl get pods -n <OAMNS>
```

## Verifying the Domain

To verify the creation of the domain, perform the following steps:

1. To confirm that the domain is created, use the following command:

   ```
   kubectl describe domain <OAM_DOMAIN_NAME> -n <OAMNS>
   ```

   For example:

   ```
   kubectl describe domain accessdomain -n oamns
   ```

2. Verify that the domain pods and services have been created, using the following command:

   ```
   kubectl get all,domains -n oamns
   ```

   The output will look similar to the following:

   ```
   NAME                                          TYPE        CLUSTER-IP
   EXTERNAL-IP  PORT(S)          AGE     SELECTOR
   service/accessdomain-adminserver              ClusterIP   None
   <none>       7001/TCP         22m
   weblogic.createdByOperator=true,weblogic.domainUID=accessdomain,weblogic.se
   rverName=AdminServer
   service/accessdomain-adminserver-external     NodePort    10.97.64.5
   <none>       7001:30701/TCP   3h46m
   weblogic.createdByOperator=true,weblogic.domainUID=accessdomain,weblogic.se
   rverName=AdminServer
   service/accessdomain-cluster-oam-cluster      ClusterIP   10.108.180.115
   <none>       14100/TCP        3h21m
   weblogic.clusterName=oam_cluster,weblogic.createdByOperator=true,weblogic.d
   omainUID=accessdomain
   service/accessdomain-cluster-policy-cluster ClusterIP   10.99.138.102
   ```

```
<none>       15100/TCP        3h21m
weblogic.clusterName=policy_cluster,weblogic.createdByOperator=true,weblogi
c.domainUID=accessdomain
service/accessdomain-oam-policy-mgr1        ClusterIP   None
<none>       15100/TCP        9m48s
weblogic.createdByOperator=true,weblogic.domainUID=accessdomain,weblogic.se
rverName=oam_policy_mgr1
service/accessdomain-oam-policy-mgr2        ClusterIP   10.96.151.188
<none>       15100/TCP        9m48s
weblogic.createdByOperator=true,weblogic.domainUID=accessdomain,weblogic.se
rverName=oam_policy_mgr2
service/accessdomain-oam-server1            ClusterIP   None
<none>       14100/TCP        9m48s
weblogic.createdByOperator=true,weblogic.domainUID=accessdomain,weblogic.se
rverName=oam_server1
service/accessdomain-oam-server2            ClusterIP   None
<none>       14100/TCP        9m48s
weblogic.createdByOperator=true,weblogic.domainUID=accessdomain,weblogic.se
rverName=oam_server2
```

**Note:**

It will take several minutes before all the services listed above appear. A pod with a STATUS of 0/1 indicates that the pod has already started but the OAM server associated with the pod is just starting. While the pods are starting, you can check the startup status in the pod logs by using the following commands:

```
kubectl logs accessdomain-adminserver -n oamns
kubectl logs accessdomain-oam-policy-mgr1 -n oamns
kubectl logs accessdomain-oam-server1 -n oamns
```

# Creating the Kubernetes Services

By default, the OAM domain gets created with all the components (except the Administration Server) configured as `ClusterIP` services. This means that the Oracle Access Manager components are visible only within the Kubernetes cluster.

In an enterprise deployment, all interactions with the WebLogic components take place through the Oracle HTTP Server which sits outside of the Kubernetes cluster. You should expose the WebLogic components to the outside world by creating additional services. You can use either NodePort Services or an Ingress controller.

- Creating the NodePort Services
- Creating the Ingress Services
- Creating an OAM ClusterIP Service
- Validating Services

## Creating the NodePort Services

You should create NodePort Services for each of the following OAM components:

- Creating an OAM NodePort Service
- Creating a Policy Manager NodePort Service
- Creating an OAP Legacy NodePort Service

## Creating an OAM NodePort Service

To create an OAM NodePort Service:

1. Create a text file called `/workdir/OAM/oam_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OAM_DOMAIN_NAME>-oam-nodeport
  namespace: <OAMNS>
spec:
  type: NodePort
  selector:
    weblogic.clusterName: oam_cluster
  ports:
    - targetPort: 14100
      port: 14100
      nodePort: <OAM_OAM_K8>
      protocol: TCP
```

> **Note:**
>
> Ensure that the namespace is set to the namespace you want to use.

2. Create the service using the following command:

```
kubectl create -f /workdir/OAM/oam_nodeport.yaml
```

The output appears as follows:

```
service/accessdomain-oam-nodeport created
```

## Creating a Policy Manager NodePort Service

To create a policy manager NodePort Service:

1. Create a text file called `/workdir/OAM/policy_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OAM_DOMAIN_NAME>-oam-policy-nodeport
  namespace: <OAMNS>
spec:
  type: NodePort
  selector:
```

```
    weblogic.clusterName: policy_cluster
  ports:
    - targetPort: 15100
      port: 15100
      nodePort: <OAM_POLICY_K8>
      protocol: TCP
```

> **Note:**
>
> Ensure that the namespace is set to the namespace you want to use.

2. Create the service using the following command:

```
kubectl create -f /workdir/OAM/policy_nodeport.yaml
```

The output appears as follows:

```
service/accessdomain-oam-policy-nodeport created
```

## Creating an OAP Legacy NodePort Service

If you are using legacy WebGates which communicate with OAM using the OAP protocol, then you will need to create an additional service.

To create a OAM legacy NodePort Service:

1. Create a text file called `/workdir/OAM/oap_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OAM_DOMAIN_NAME>-oap-nodeport
  namespace: <OAMNS>
spec:
  type: NodePort
  selector:
    weblogic.clusterName: oam_cluster
  ports:
    - targetPort: 5575
      port: 5575
      nodePort: <OAM_OAP_SERVICE_PORT>
      protocol: TCP
```

2. Create the service using the following command:

```
kubectl create -f /workdir/OAM/oap_nodeport.yaml
```

The output appears as follows:

```
service/accessdomain-oap-nodeport created
```

## Creating the Ingress Services

To create Ingress services, you must first create an Ingress controller. For more information about the installation procedure, see Installing and Configuring Ingress Controller.

The Ingress service is created inside the product namespace. It tells the Ingress controller how to direct requests inside the namespace.

> **Note:**
>
> The example below creates two Ingress services, one for each of the OAM virtual hosts.
>
> - `iadadmin.example.com`
> - `login.example.com`

To create an Ingress service:

1. Copy the `values.yaml` file from the `<WORKDIR>/samples/charts/ingress-per-domain` directory to the working directory and rename the file to `override_ingress.yaml`.
2. Edit the `<WORKDIR>/override_ingress.yaml` file and set the values as follows:

```
set domainUID to <OAM_DOMAIN_NAME>
set adminServerPort to <OAM_ADMIN_PORT>
set hostName.enabled to true
set admin to <OAM_ADMIN_LBR_HOST>
set runtime to <OAM_LOGIN_LBR_HOST>
```

For example:

```
# Load balancer type. Supported values are: NGINX
type: NGINX

# SSL configuration Type. Supported Values are : NONSSL,SSL
sslType: NONSSL

# domainType. Supported values are: oam
domainType: oam

#WLS domain as backend to the load balancer
wlsDomain:
  domainUID: accessdomain
  adminServerName: AdminServer
  adminServerPort: 7001
  adminServerSSLPort:
  oamClusterName: oam_cluster
  oamManagedServerPort: 14100
  oamManagedServerSSLPort:
  policyClusterName: policy_cluster
  policyManagedServerPort: 15100
```

```
    policyManagedServerSSLPort:

  # Host specific values
  hostName:
    enabled: true
    admin: iadadmin.example.com
    runtime: login.example.com
```

3. Create the Ingress services by running the following commands:

```
cd <WORKDIR>/samples


helm install oam-nginx charts/ingress-per-domain --namespace <OAMNS> --
values <WORKDIR>/override_ingress.yaml
```

4. Validate that the Ingress service has been created correctly by using the following command:

```
kubectl get ingress -n oamns
```

## Creating an OAM ClusterIP Service

Create a ClusterIP service for OAP connections.

For this service, create the `oap_clusterip.yaml` file with the following contents:

```
kind: Service
apiVersion: v1
metadata:
  name: <OAM_DOMAIN_NAME>-oap
  namespace: <OAMNS>
spec:
  type: ClusterIP
  selector:
    weblogic.clusterName: oam_cluster
  ports:
    - port: 5575
      protocol: TCP
```

Create the service using the following command:

```
kubectl create -f /workdir/OAM/oap_clusterip.yaml
```

The output appears as follows:

```
service/accessdomain-oap created
```

## Validating Services

To validate that the services have been created correctly, use the following command:

```
kubectl get service -n oamns
```

The output of this commands appears as follows:

```
NAME                                        TYPE             CLUSTER-
IP          EXTERNAL-IP PORT(S)             AGE
accessdomain-adminserver              ClusterIP      None <none>
30012/TCP,7001/TCP             4d15h
accessdomain-adminserver-ext                NodePort       10.96.127.191
<none>    30012:30012/TCP,7001:30701/TCP  4d15h
accessdomain-cluster-oam-cluster          ClusterIP      10.96.43.35
<none>      14100/TCP              4d15h
accessdomain-cluster-policy-cluster       ClusterIP      10.96.8.16
<none>       15100/TCP              4d15h
accessdomain-oam-nodeport           NodePort      10.96.104.168 <none>
14100:30410/TCP          4d15h
accessdomain-oam-policy-mgr1                ClusterIP      None
<none>               15100/TCP              4d15h
accessdomain-oam-policy-mgr2                ClusterIP      None
<none>               15100/TCP              4d15h
accessdomain-oam-policy-mgr3                ClusterIP      10.96.36.96
<none>      15100/TCP              4d15h
accessdomain-oam-policy-mgr4                ClusterIP      10.96.171.61
<none>     15100/TCP              4d15h
accessdomain-oam-policy-mgr5                ClusterIP      10.96.200.171
<none>    15100/TCP              4d15h
accessdomain-oam-server1           ClusterIP      None <none>
14100/TCP              4d15h
accessdomain-oam-server2           ClusterIP      None <none>
14100/TCP              4d15h
accessdomain-oam-server3           ClusterIP      10.96.93.51 <none>
14100/TCP              4d15h
accessdomain-oam-server4               ClusterIP      10.96.223.123 <none>
14100/TCP              4d15h
accessdomain-oam-server5           ClusterIP      10.96.143.94 <none>
14100/TCP              4d15h
accessdomain-oap               ClusterIP      10.96.171.107 <none>
5575/TCP             4d15h
accessdomain-policy-nodeport              NodePort       10.96.169.250
<none>    15100:30510/TCP          4d15h
```

# Updating the Domain

When you first create the domain, some information may be missing. You can add this information using a simple `curl` script.

This procedure performs the following actions:

- Adds an internally resolvable hostname to each OAM Server.

- Sets the OAP port number for each OAM Server.
- Changes the OAP settings replacing the local host name with the load balancer host name.
- Updates the REST points for Oracle 12c WebGate HTTP OAM APIs in the existing WebGate agents.
- Updates the federation entry points to use the load balancer.
- Changes the OAP security mode for legacy WebGates in the ready-to-use WebGate agents.
- Updates the timeout settings.
- Updates the maximum number of connections for ready-to-use WebGates.

1. Create a file called `/workdir/OAM/modify_oam.xml` with the following information in it:

```
<Configuration>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server1/host"><OAM_DOMAIN_NAME>-oam-server1</
Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server2/host"><OAM_DOMAIN_NAME>-oam-server2</
Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server3/host"><OAM_DOMAIN_NAME>-oam-server3</
Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server4/host"><OAM_DOMAIN_NAME>-oam-server4</
Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server5/host"><OAM_DOMAIN_NAME>-oam-server5</
Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server1/oamproxy/Port"><OAP_PORT></Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server2/oamproxy/Port"><OAP_PORT></Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server3/oamproxy/Port"><OAP_PORT></Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server4/oamproxy/Port"><OAP_PORT></Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server5/oamproxy/Port"><OAP_PORT></Setting>
<Setting Name="serverhost" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/
serverhost"><OAM_LOGIN_LBR_HOST></Setting>
<Setting Name="serverport" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/
serverport"><OAM_LOGIN_LBR_PORT></Setting>
<Setting Name="serverprotocol" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/
serverprotocol"><OAM_LOGIN_LBR_PROTOCOL></Setting>
<Setting Name="serverhost" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMServerBackChannel/
serverhost"><OAM_LOGIN_LBR_HOST></Setting>
<Setting Name="serverport" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMServerBackChannel/
```

```
serverport"><OAM_LOGIN_LBR_HOST></Setting>
<Setting Name="serverprotocol" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMServerBackChannel/
serverprotocol"><OAM_LOGIN_LBR_PROTOCOL></Setting>
<Setting Name="OAMRestEndPointHostName" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
UserDefinedParameters/OAMRestEndPointHostName"><OAM_LOGIN_LBR_HOST></
Setting>
<Setting Name="OAMRestEndPointPort" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
UserDefinedParameters/OAMRestEndPointPort"><OAM_LOGIN_LBR_PORT></Setting>
<Setting Name="providerid" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/fedserverconfig/
providerid"><OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oam/fed</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server1/CoherenceConfiguration/LocalHost/
Value"><OAM_DOMAIN_NAME>-oam-server1</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server2/CoherenceConfiguration/LocalHost/
Value"><OAM_DOMAIN_NAME>-oam-server2</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server3/CoherenceConfiguration/LocalHost/
Value"><OAM_DOMAIN_NAME>-oam-server3</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server4/CoherenceConfiguration/LocalHost/
Value"><OAM_DOMAIN_NAME>-oam-server4</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server5/CoherenceConfiguration/LocalHost/
Value"><OAM_DOMAIN_NAME>-oam-server5</Setting>
<Setting Name="assertionissuer" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/issuancetemplates/saml11-issuance-template/
assertionissuer"><OAM_LOGIN_LBR_HOST></Setting>
<Setting Name="assertionissuer" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/issuancetemplates/saml20-issuance-template/
assertionissuer"><OAM_LOGIN_LBR_HOST></Setting>
<Setting Name="openid20realm" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/spglobal/
openid20realm"><OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT></Setting>
<Setting Name="logoutRedirectUrl" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
logoutRedirectUrl"><OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oam/server/logout</Setting>
<Setting Name="security" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/security"><OAP_MODE></Setting>
<Setting Name="security" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/IAMSuiteAgent/security"><OAP_MODE></Setting>
<Setting Name="logoutRedirectUrl" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/IAMSuiteAgent/
UserDefinedParameters/logoutRedirectUrl"><OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oam/server/logout</Setting>
<Setting Name="Timeout" Type="htf:timeInterval" Path="/DeployedComponent/
Server/NGAMServer/Profile/Sme/SessionConfigurations/Timeout">15 M</Setting>

<Setting Name="PrimaryServerList" Type="htf:list" Path="/DeployedComponent/
```

```
Agent/WebGate/Instance/IAMSuiteAgent/PrimaryServerList">
<Setting Name="0" Type="htf:map" Path="/DeployedComponent/Agent/WebGate/
Instance/IAMSuiteAgent/PrimaryServerList/0">
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/host"><OAM_OAP_HOST></
Setting>
<Setting Name="port" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/
port"><OAP_SERVICE_PORT></Setting>
<Setting Name="numOfConnections" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
PrimaryServerList/0/numOfConnections"><WG_CONNECTIONS></Setting>
</Setting>
</Setting>

<Setting Name="PrimaryServerList" Type="htf:list" Path="/DeployedComponent/
Agent/WebGate/Instance/accessgate-oic/PrimaryServerList">
<Setting Name="0" Type="htf:map" Path="/DeployedComponent/Agent/WebGate/
Instance/accessgate-oic/PrimaryServerList/0">
<Setting Name="port" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/
port"><OAP_SERVICE_PORT></Setting>
<Setting Name="numOfConnections" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
PrimaryServerList/0/numOfConnections"><WG_CONNECTIONS></Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/host"><OAM_OAP_HOST></
Setting>
</Setting>
</Setting>
</Configuration>
```

**Note:**

- You need one entry of `NGAMServer/Instance/<servername>/host` for each `oam_server` in the domain.
- You need one entry of `NGAMServer/Instance/<servername>/port` for each `oam_server` in the domain.
- You need one entry of `NGAMServer/Instance/<servername>/CoherenceConfiguration` for each `oam_server` in the domain.

For example:

```
<Configuration>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server1/host">accessdomain-oam-server1</Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server2/host">accessdomain-oam-server2</Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server3/host">accessdomain-oam-server3</Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
```

```
NGAMServer/Instance/oam_server4/host">accessdomain-oam-server4</Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server5/host">accessdomain-oam-server5</Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server1/oamproxy/Port">5575</Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server2/oamproxy/Port">5575</Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server3/oamproxy/Port">5575</Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server4/oamproxy/Port">5575</Setting>
<Setting Name="Port" Type="xsd:integer" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server5/oamproxy/Port">5575</Setting>
<Setting Name="serverhost" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/
serverhost">login.example.com</Setting>
<Setting Name="serverport" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/serverport"></Setting>
<Setting Name="serverprotocol" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/
serverprotocol">https</Setting>
<Setting Name="serverhost" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMServerBackChannel/
serverhost">login.example.com</Setting>
<Setting Name="serverport" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMServerBackChannel/
serverport"></Setting>
<Setting Name="serverprotocol" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/OAMServerProfile/OAMServerBackChannel/
serverprotocol">https</Setting>
<Setting Name="OAMRestEndPointHostName" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
UserDefinedParameters/OAMRestEndPointHostName">login.example.com</Setting>
<Setting Name="OAMRestEndPointPort" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
UserDefinedParameters/OAMRestEndPointPort"></Setting>
<Setting Name="providerid" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/fedserverconfig/providerid">https://
login.example.com:443/oam/fed</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server1/CoherenceConfiguration/LocalHost/
Value">accessdomain-oam-server1</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server2/CoherenceConfiguration/LocalHost/
Value">accessdomain-oam-server2</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server3/CoherenceConfiguration/LocalHost/
Value">accessdomain-oam-server3</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server4/CoherenceConfiguration/LocalHost/
Value">accessdomain-oam-server4</Setting>
<Setting Name="Value" Type="xsd:string" Path="/DeployedComponent/Server/
NGAMServer/Instance/oam_server5/CoherenceConfiguration/LocalHost/
Value">accessdomain-oam-server5</Setting>
<Setting Name="assertionissuer" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/issuancetemplates/saml11-issuance-template/
```

```
assertionissuer">login.example.com</Setting>
<Setting Name="assertionissuer" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/issuancetemplates/saml20-issuance-template/
assertionissuer">login.example.com</Setting>
<Setting Name="openid20realm" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/STS/spglobal/openid20realm">https://
login.example.com:443</Setting>
<Setting Name="logoutRedirectUrl" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
logoutRedirectUrl">https://login.example.com:443/oam/server/logout</
Setting>
<Setting Name="security" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/security">open</Setting>
<Setting Name="security" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/IAMSuiteAgent/security">open</Setting>
<Setting Name="logoutRedirectUrl" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/IAMSuiteAgent/
UserDefinedParameters/logoutRedirectUrl">https://login.example.com:/oam/
server/logout</Setting>
<Setting Name="Timeout" Type="htf:timeInterval" Path="/DeployedComponent/
Server/NGAMServer/Profile/Sme/SessionConfigurations/Timeout">15 M</Setting>

<Setting Name="PrimaryServerList" Type="htf:list" Path="/DeployedComponent/
Agent/WebGate/Instance/IAMSuiteAgent/PrimaryServerList">
<Setting Name="0" Type="htf:map" Path="/DeployedComponent/Agent/WebGate/
Instance/IAMSuiteAgent/PrimaryServerList/0">
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/host">oam.example.com</
Setting>
<Setting Name="port" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/port">30540</Setting>
<Setting Name="numOfConnections" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
PrimaryServerList/0/numOfConnections">20</Setting>
</Setting>
</Setting>

<Setting Name="PrimaryServerList" Type="htf:list" Path="/DeployedComponent/
Agent/WebGate/Instance/accessgate-oic/PrimaryServerList">
<Setting Name="0" Type="htf:map" Path="/DeployedComponent/Agent/WebGate/
Instance/accessgate-oic/PrimaryServerList/0">
<Setting Name="port" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/port">30540</Setting>
<Setting Name="numOfConnections" Type="xsd:string" Path="/
DeployedComponent/Agent/WebGate/Instance/accessgate-oic/
PrimaryServerList/0/numOfConnections">20</Setting>
<Setting Name="host" Type="xsd:string" Path="/DeployedComponent/Agent/
WebGate/Instance/accessgate-oic/PrimaryServerList/0/host">oam.example.com</
Setting>
</Setting>
</Setting>
</Configuration>
```

2. Save the file.

3. Apply the configuration changes using `curl`. For example:

```
curl -x '' -X PUT http://<K8_WORKER_NODE1>:<OAM_ADMIN_K8>/iam/admin/
config/api/v1/config -ikL -H 'Content-Type: application/xml' --user
<OAM_WEBLOGIC_USER>:<OAM_WEBLOGIC_PWD> -H 'cache-control: no-cache' -d @/
workdir/OAM/modify_oam.xml
```

For example:

```
curl -x '' -X PUT http://k8worker1.example.com:30701/iam/admin/
config/api/v1/config -ikL -H 'Content-Type: application/xml' --user
weblogic:<password> -H 'cache-control: no-cache' -d @/workdir/OAM/
modify_oam.xml
```

# Performing the Post-Configuration Tasks for Oracle Access Management Domain

The post-configuration tasks for the OAM domain include creating the server overrides file and updating the data sources.

- Limiting Pods to Specific Worker Nodes
- Creating the Server Overrides File

## Limiting Pods to Specific Worker Nodes

If you want to ensure that the OAM servers start only on a specific set of worker servers, complete the following steps:

- Labeling the Kubernetes Worker Nodes
- Restricting Processes to Labels

## Labeling the Kubernetes Worker Nodes

Label the worker nodes you want to include in scheduling. This can be as granular as you need. For example, if you want to schedule the OAM processes to run on a set of nodes, then label that set with a label such as `oamservers`. If you want to dictate that the Administration Server runs on a specific set of worker nodes and `oam_server` on a different set, then create two labels, `oamadmin` and `oamservers`.

Add a label to a Kubernetes node using the following command:

```
kubectl label node worker1 name=oamservers
```

## Restricting Processes to Labels

To ensure that the OAM pods run on only worker nodes with the appropriate label, edit the `domain.yaml` file located in the following path:

```
<WORKDIR>/samples/create-access-domain/domain-home-on-pv/output/weblogic-
domains/<OAM_DOMAIN_NAME>/
```

For example:

```
/workdir/OAM/samples/create-access-domain/domain-home-on-pv/output/weblogic-domains/accessdomain/
```

Alter the Managed Servers section for all the Managed Servers configured in the cluster and ensure that only the labeled worked nodes are used for scheduling.

For `oam_server1` and `oam_server2`, the entries will look similar to:

```
managedServers:
- serverName: oam_server1
  serverPod:
    nodeSelector:
      name: oamservers
- serverName: oam_server2
  serverPod:
    nodeSelector:
      names: oamservers
```

# Creating the Server Overrides File

The `serverOverrides` file is used to set specific Java values when the containers start. The parameters are appended to the configuration in the `setDomainEnv.sh` file but unlike the `setDomainEnv.sh`file, the `serverOverrides` file is not overwritten during the upgrade.

- Disabling the Derby Database
- Enabling the Managed Servers to Use IPv4 Networking
- Setting the Memory Parameters in IAMAccessDomain
- Copying Server Overrides to the Kubernetes Containers

## Disabling the Derby Database

Disable the embedded Derby database, which is a file-based database, packaged with Oracle WebLogic Server. The Derby database is used primarily for development environments. Therefore, you must disable it when you are configuring a production-ready enterprise deployment environment. Otherwise, the Derby database process starts automatically when you start the Managed servers.

To disable the Derby database:

1. Create a file called `/workdir/OAM/setUserOverrides.sh` with the following content:

   ```
   DERBY_FLAG=false
   ```

2. Save and close the file.

## Enabling the Managed Servers to Use IPv4 Networking

If the Managed Server is configured to use IPv6 networking, you may encounter issues when you start the Managed Server. Therefore, you must enable the Managed Servers to use IPv4 networking.

1. Edit the `setUserOverrides.sh` file and add the following line:

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Djava.net.preferIPv4Stack=true"
```

2. Save and close the file.

## Setting the Memory Parameters in IAMAccessDomain

The initial startup parameter in the IAMAccessDomain, which defines the memory usage, is insufficient. You must increase the value of this parameter.

To change the memory allocation setting:

1. Change the following memory allocation in the `setUserOverrides.sh` file by updating the Java maximum memory allocation pool (Xmx) to 8192m and initial memory allocation pool (Xms) to 8192m. For example, add the following line:

```
MEM_ARGS="-Xms8192m -Xmx8192m"
```

> **Note:**
>
> For larger systems, these values should be:
>
> ```
> MEM_ARGS="-Xms8192m -Xmx8192m"
> ```

2. Save and close the file.

## Copying Server Overrides to the Kubernetes Containers

In a Kubernetes environment, there is no editior inside the container. To work around this issue, create the file on the master node and copy it to the Kubernetes container using the following commands:

```
chmod 755 /workdir/OAM/setUserOverrides.sh
```

```
kubectl cp <WORKDIR>/setUserOverrides.sh <OAMNS>/<OAM_DOMAIN_NAME>-
adminserver:/u01/oracle/user_projects/domains/<OAM_DOMAIN_NAME>/bin/
setUserOverrides.sh
```

For example:

```
kubectl cp /workdir/OAM/setUserOverrides.sh oamns/accessdomain-
adminserver:/u01/oracle/user_projects/domains/accessdomain/bin/
setUserOverrides.sh
```

# Restarting the Domain

Restart the domain for the changes to take effect.

To restart the domain, use the following commands:

```
kubectl -n <OAMNS> patch domains <OAM_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
```

After all the Kubernetes pods (with the exception of the helper pod) in the namespace have stopped, you can restart the domain by using the following command:

```
kubectl -n <OAMNS> patch domains <OAM_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IfNeeded" }]'
```

> **Note:**
>
> Check that all the Kubernetes pods (with the exception of the helper pod) in the namespace have stopped by using the following command:
>
> ```
> kubectl -n <OAMNS> get all
> ```
>
> All the Kubernetes pods (with the exception of the helper pod) in the namespace will be stopped when there are no entries for the Administration Server or the Managed Servers.

For example:

```
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IfNeeded" }]'
```

# Validating the Administration Server

Before you perform the configuration steps, validate that the Administration Server has started successfully by ensuring that you have access to the Oracle WebLogic Server Administration Console and Oracle Enterprise Manager Fusion Middleware Control, both installed and configured on the Administration Server.

> **Note:**
>
> For the validation to work, you need to have a browser capable of communicating with the Kubernetes worker nodes.

To navigate to Fusion Middleware Control, enter the following URL, and log in with the Oracle WebLogic Server administrator credentials:

```
http://k8worker1.example.com:30701/em
```

To navigate to the Oracle WebLogic Server Administration Console, enter the following URL, and log in with the same administration credentials:

```
http://k8worker1.example.com:30701/console
```

# Removing OAM Server from WebLogic Server 12*c* Default Coherence Cluster

Exclude all Oracle Access Management (OAM) clusters (including Policy Manager and OAM runtime server) from the default WebLogic Server 12*c* coherence cluster by using the WebLogic Server Administration Console.

From release 12.2.1.3.0 onwards, OAM server-side session management uses database and does not require coherence cluster to be established. In some environments, warnings and errors are observed due to default coherence cluster initialized by WebLogic. To avoid or fix these errors, exclude all of the OAM clusters from default WebLogic Server coherence cluster using the following steps:

1. Log in to the WebLogic Server Administration Console, using the URL:

   `http://k8worker1.example.com:30701/console`
2. In the left pane of the console, expand **Environment** and select **Coherence Clusters**.

   The Summary of Coherence Clusters page displays the Coherence cluster configurations that have been created in this domain.
3. Click **defaultCoherenceCluster** and select the **Members** tab.
4. Click **Lock and Edit**.
5. From **Servers and Clusters**, deselect all OAM clusters (including policy manager and OAM runtime server).
6. Click **Save**.
7. Click **Activate changes**.

# Tuning the WebLogic Server

Tune the WebLogic Server for optimum performance by adding the Minimum Thread Constraint and removing the Max Thread and Capacity constraints.

**Adding the Minimum Thread Constraint to Worker Manager `OAPOverRestWM`**

1. Log in to the WebLogic Server Console at `http://k8worker1.example.com:30701/console`.
2. Click **Lock & Edit**.
3. In Domain Structure, click **Deployments**.
4. On the Deployments page, click **Next** until you see `oam_server`.
5. Expand `oam_server` by clicking the + icon, and then click **/iam/access/binding**.
6. Click the **Configuration** tab, followed by the **Workload** tab.
7. Click **wm/OAPOverRestWM**.
8. Under **Application Scoped Work Managed Components**, click **New**.

9. In **Create a New Work Manager Component**, select **Minumum Threads Constraint** and click **Next**.
10. In **Minimum Threads Constraint Properties**, enter the **Count** as `400` and click **Finish**.
11. In **Save Deployment Plan**, change the **Path** to `/u01/oracle/user_projects/domains/accessdomain/Plan.xml`.
12. Click **OK**, and then click **Activate Changes**.

**Removing the Maximum Thread Constraint and Capacity Constraint**

1. Log in to the WebLogic Server Console at `http://k8worker1.example.com:30701/console`.
2. Click **Lock & Edit**.
3. In Domain Structure, click **Deployments**.
4. On the Deployments page, click **Next** until you see `oam_server`.
5. Expand `oam_server` by clicking the + icon, and then click **/iam/access/binding**.
6. Click the **Configuration** tab, followed by the **Workload** tab.
7. Click **wm/OAPOverRestWM**.
8. Under **Application Scoped Work Managed Components**, select **Capacity and MaxThreadsCount**, and then click **Delete**.
9. In the Delete Work Manage Components screen, click **OK** to delete.
10. Click **Release Configuration**.

# Enabling Virtualization

You can use the Fusion Middleware Control to enable virtualization.

To enable virtualization:

1. Log in to Oracle Fusion Middleware Console using the URL:

   `http://k8worker1.example.com:30701/em`
2. Click **WebLogic Domain > Security > Security Provider Configuration**.
3. Expand **Security Store Provider**.
4. Expand **Identity Store Provider**.
5. Click **Configure**.
6. Add a custom property.
7. Select `virtualize` property with value `true` and click **OK**.
8. Click **OK** again to persist the change.

For more information about the virtualize property, see OPSS System and Configuration Properties in *Securing Applications with Oracle Platform Security Services*.

# Restarting the Domain

Restart the domain for the changes to take effect.

To restart the domain, use the following commands:

```
kubectl -n <OAMNS> patch domains <OAM_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
```

After all the Kubernetes pods (with the exception of the helper pod) in the namespace have stopped, you can restart the domain by using the following command:

```
kubectl -n <OAMNS> patch domains <OAM_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IfNeeded" }]'
```

> **Note:**
>
> Check that all the Kubernetes pods (with the exception of the helper pod) in the namespace have stopped by using the following command:
>
> ```
> kubectl -n <OAMNS> get all
> ```
>
> All the Kubernetes pods (with the exception of the helper pod) in the namespace will be stopped when there are no entries for the Administration Server or the Managed Servers.

For example:

```
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IfNeeded" }]'
```

# Configuring and Integrating with LDAP

Configuration and integration of OAM with LDAP requires you to first set a global passphrase. You then configure OAM to use the LDAP directory. In addition, create the Webgate_IDM agent if not present and finally assign administration rights to the LDAP users to access the MBeans stored in the Administration Server.

- Obtaining a Global Passphrase
- Configuring Access Manager to Use the LDAP Directory
- Creating the Webgate_IDM Agent
- Adding LDAP Groups to WebLogic Administrators

## Obtaining a Global Passphrase

By default, Oracle Access Manager is configured to use the open security model. If you plan to change this mode using `idmConfigTool`, you must know the global passphrase. By default, Oracle creates a global passphrase for you. You can override this value, if required.

> **Note:**
>
> If you are using the latest 12c WebGate functionality using OAP over REST calls, it is not important to change the security mode because REST calls do not use the OAP transport mode.

- Obtaining the Default Global Passphrase

## Obtaining the Default Global Passphrase

You will need the global passphrase while creating the WebGate agents. To obtain the global passphrase:

1. Start a bash shell in the Administration Server container using the command:

   ```
   kubectl exec -n <OAMNS> -ti <OAM_DOMAIN_NAME>-adminserver -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
   ```

2. Start WLST by using the following command:

   ```
   /u01/oracle/oracle_common/common/bin/wlst.sh
   ```

3. Connect to the domain using the command:

   ```
   connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://<OAM_DOMAIN_NAME>-
   domain-adminserver.<OAMNS>.svc.cluster.local:<OAM_EXT_T3_PORT>')
   ```

   For example:

   ```
   connect('weblogic','<password>','t3://accessdomain-
   adminserver.oamns.svc.cluster.local:30012')
   ```

4. Issue the WLST command:

   ```
   displaySimpleModeGlobalPassphrase()
   ```

The system generated passphrase is displayed.

# Configuring Access Manager to Use the LDAP Directory

After completing the initial installation and setting the security model, you have to associate Oracle Access Manager with the LDAP directory. You can use Oracle Unified Directory (OUD) as the LDAP directory.

To associate Access Manager and the LDAP directory, perform the following tasks:

- Creating a Configuration File
- Integrating Oracle Access Manager and LDAP Using the idmConfigTool
- Validating the OAM LDAP Integration

## Creating a Configuration File

Configuring Oracle Access Management to use LDAP requires you to run the `idmConfigTool` utility. Therefore, you must create a configuration file called `oam.props` to use during the configuration. The contents of this file are:

```
#IDSTORE PROPERTIES

IDSTORE_HOST: <LDAP_HOSTNAME>
IDSTORE_PORT: <LDAP_PORT>
IDSTORE_BINDDN:<LDAP_ADMIN_USER>
IDSTORE_SEARCHBASE: <LDAP_SEARCHBASE>
IDSTORE_GROUPSEARCHBASE: <LDAP_GROUP_SEARCHBASE>
IDSTORE_USERNAMEATTRIBUTE: cn
IDSTORE_LOGINATTRIBUTE: uid
IDSTORE_USERSEARCHBASE: <LDAP_USER_SEARCHBASE>
IDSTORE_SYSTEMIDBASE: <LDAP_SYSTEMIDS>
IDSTORE_NEW_SETUP: true
IDSTORE_DIRECTORYTYPE: <LDAP_TYPE>
IDSTORE_WLSADMINUSER: <LDAP_WLSADMIN_USER>
IDSTORE_WLSADMINGROUP: <LDAP_WLSADMIN_GRP>
IDSTORE_OAMADMINUSER: <LDAP_OAMADMIN_USER>
IDSTORE_OAMSOFTWAREUSER: <LDAP_OAMLDAP_USER>
# OAM Properties
OAM11G_SERVER_LOGIN_ATTRIBUTE: uid
OAM11G_IDSTORE_NAME: OAMIDSTORE
OAM11G_IDSTORE_ROLE_SECURITY_ADMIN: <LDAP_OAMADMIN_GRP>
PRIMARY_OAM_SERVERS: <OAM_DOMAIN_NAME>-
oap.<OAMNS>.svc.cluster.local:<OAM_OAP_PORT>
WEBGATE_TYPE: ohsWebgate12c
ACCESS_GATE_ID: Webgate_IDM
OAM11G_OIM_WEBGATE_PASSWD: <LDAP_USER_PWD>
COOKIE_DOMAIN: <OAM_COOKIE_DOMAIN>
COOKIE_EXPIRY_INTERVAL: 120
OAM11G_WG_DENY_ON_NOT_PROTECTED: true
OAM11G_IDM_DOMAIN_OHS_HOST: <OAM_LOGIN_LBR_HOST>
OAM11G_IDM_DOMAIN_OHS_PORT: <OAM_LOGIN_LBR_PORT>
OAM11G_IDM_DOMAIN_OHS_PROTOCOL: <OAM_LOGIN_LBR_PROTOCOL>
OAM11G_SERVER_LBR_HOST: <OAM_LOGIN_LBR_HOST>
OAM11G_SERVER_LBR_PORT: <OAM_LOGIN_LBR_PORT>
OAM11G_SERVER_LBR_PROTOCOL: <OAM_LOGIN_LBR_PROTOCOL>
OAM11G_OAM_SERVER_TRANSFER_MODE: open
OAM_TRANSFER_MODE: open
OAM11G_SSO_ONLY_FLAG: false
OAM11G_IMPERSONATION_FLAG: false
OAM11G_IDM_DOMAIN_LOGOUT_URLS: /console/jsp/common/logout.jsp,/em/targetauth/
emaslogout.jsp
OAM11G_OIM_INTEGRATION_REQ: <OAM_OIG_INTEG>
OAM11G_OIM_OHS_URL: <OIG_LBR_PROTOCOL>://<OIG_LBR_HOST>:<OIG_LBR_PORT>/
# WebLogic Properties
WLSHOST:<OAM_DOMAIN_NAME>-adminserver.<OAMNS>.svc.cluster.local
WLSPORT: 7001
WLSADMIN: <OAM_WEBLOGIC_USER>
```

For example:

```
#IDSTORE PROPERTIES
IDSTORE_HOST: edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local
IDSTORE_PORT: 1389
IDSTORE_BINDDN: cn=oudadmin
IDSTORE_SEARCHBASE: dc=example,dc=com
IDSTORE_GROUPSEARCHBASE: cn=Groups,dc=example,dc=com
IDSTORE_USERNAMEATTRIBUTE: cn
IDSTORE_LOGINATTRIBUTE: uid
IDSTORE_USERSEARCHBASE: cn=Users,dc=example,dc=com
IDSTORE_SYSTEMIDBASE: cn=systemids,dc=example,dc=com
IDSTORE_NEW_SETUP: true
IDSTORE_DIRECTORYTYPE: OUD
IDSTORE_WLSADMINUSER: weblogic_iam
IDSTORE_WLSADMINGROUP: WLSAdministrators
IDSTORE_OAMADMINUSER: oamadmin
IDSTORE_OAMSOFTWAREUSER: oamLDAP
# OAM Properties
OAM11G_SERVER_LOGIN_ATTRIBUTE: uid
OAM11G_IDSTORE_NAME: OAMIDSTORE
OAM11G_IDSTORE_ROLE_SECURITY_ADMIN: OAMAdministrators
PRIMARY_OAM_SERVERS: accessdomain-oap.oamns.svc.cluster.local:5575
WEBGATE_TYPE: ohsWebgate12c
ACCESS_GATE_ID: Webgate_IDM
OAM11G_OIM_WEBGATE_PASSWD: Password
COOKIE_DOMAIN: .example.com
COOKIE_EXPIRY_INTERVAL: 120
OAM11G_WG_DENY_ON_NOT_PROTECTED: true
OAM11G_IDM_DOMAIN_OHS_HOST: login.example.com
OAM11G_IDM_DOMAIN_OHS_PORT: 443
OAM11G_IDM_DOMAIN_OHS_PROTOCOL: https
OAM11G_SERVER_LBR_HOST: login.example.com
OAM11G_SERVER_LBR_PORT: 443
OAM11G_SERVER_LBR_PROTOCOL: https
OAM11G_OAM_SERVER_TRANSFER_MODE: open
OAM_TRANSFER_MODE: open
OAM11G_SSO_ONLY_FLAG: false
OAM11G_IMPERSONATION_FLAG: false
OAM11G_IDM_DOMAIN_LOGOUT_URLS: /console/jsp/common/logout.jsp,/em/targetauth/
emaslogout.jsp
OAM11G_OIM_INTEGRATION_REQ: false
OAM11G_OIM_OHS_URL: https://prov.example.com:443/
# WebLogic Properties
WLSHOST:accessdomain-adminserver.oamns.svc.cluster.local
WLSPORT: 7001
WLSADMIN: weblogic
```

> **Note:**
>
> It is not possible to use `vi` or similar commands in a WebLogic Kubernetes container. Therefore, this file should be created outside of the container, and then copied into the container.
>
> In this example, the file is copied to the `/u01/oracle/user_projects/workdir` directory. If this directory does not exist inside the container, you will first need to create it.
>
> ```
> kubectl exec -n <OAMNS>-ti accessdomain-adminserver mkdir <K8_WORKDIR>
> ```
>
> For example:
>
> ```
> kubectl exec -n oamns -ti accessdomain-adminserver mkdir /u01/oracle/
> user_projects/workdir
> ```
>
> After you create the directory, copy the file to this directory.
>
> ```
> kubectl cp /workdir/OAM/oam.props oamns/accessdomain-adminserver:/u01/
> oracle/user_projects/workdir
> ```

## Integrating Oracle Access Manager and LDAP Using the idmConfigTool

> **Note:**
>
> Before running the `idmconfigTool`, ensure that the `oam_server1` and `oam_server2` Managed Servers are shut down.
>
> To stop the OAM servers, use the following command:
>
> ```
> cd /workdir/OPER/samples/domain-lifecycle
> ```
>
> ```
> ./stopCluster.sh -c oam_cluster -d accessdomain -n oamns -v
> ```

To integrate Oracle Access Manager with the LDAP directory:

1. Connect to the AdminServer container and start a shell:

   ```
   kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
   ```

2. Set the environment variables *MW_HOME*, *JAVA_HOME*, and *ORACLE_HOME*.

   ```
   export CLASSPATH=$CLASSPATH:/u01/oracle/wlserver/server/lib/weblogic.jar
   export ORACLE_HOME=/u01/oracle/idm
   export MW_HOME=/u01/oracle
   ```

3. Run the idmConfigTool utility to perform the integration.

   The syntax of the command on Linux is:

   ```
   cd $ORACLE_HOME/idmtools/bin
   ./idmConfigTool.sh -configOAM input_file=configfile
   ```

   For example:

   ```
   ./idmConfigTool.sh -configOAM input_file=/u01/oracle/user_projects/workdir/oam.props
   ```

   When the command runs you are prompted to enter the password of the account you are connecting to the Identity Store with. You are also asked to specify the passwords you want to assign to these accounts:

   - IDSTORE_PWD_OAMSOFTWAREUSER
   - IDSTORE_PWD_OAMADMINUSER
   - OAM11G_WLS_ADMIN_PASSWD

4. Check the log file for any errors or warnings and correct them. A file named `automation.log` is created in the directory where you run the tool.

5. Restart the domain using the following commands:

   ```
   kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
   "replace", "path": "/spec/serverStartPolicy", "value": "NEVER" }]'
   ```

   Wait for all processes to shut down, and then use the following command:

   ```
   kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
   "replace", "path": "/spec/serverStartPolicy", "value": "IF_NEEDED" }]'
   ```

6. Check whether oam_server1 has restarted. If not, use the following commands:

   ```
   cd /workdir/OPER/weblogic-kubernetes-operator/kubernetes/samples/scripts/
   domain-lifecycle
   ```

   ```
   ./startCluster.sh -c oam_cluster -d accessdomain -n oamns
   ```

> **Note:**
>
> After you run `idmConfigTool`, several files are created that you need for subsequent tasks. Keep these in a safe location.
>
> The following files exist in this directory: `/u01/oracle/user_projects/domains/accessdomain/output/Webgate_IDM .`
>
> You will require these files when you install the WebGate software.
>
> - `cwallet.sso`
> - `ObAccessClient.xml`
> - `password.xml`
> - `aaa_cert.pem`
> - `aaa_key.pem`

## Validating the OAM LDAP Integration

To validate that the OAM LDAP integration has completed correctly:

1. Access the OAM Console using the following URL:

   ```
   http://iadadmin.example.com/oamconsole
   ```

   or, if you have not yet configured the Oracle HTTP Server, you can use:

   ```
   http://k8worker.example.com:30701/oamconsole
   ```

2. Log in as the Access Manager administration user you created when you prepared the ID Store. For example `oamadmin`.
3. Click **Agents** from the Application Security screen.
4. When the Search SSO Agents screen appears, click **Search**.
5. You should see the WebGate agent `Webgate_IDM`.

   > **Note:**
   >
   > If you discover that the Webgate_IDM Agent does not exist, you can create it manually. See Creating the Webgate_IDM Agent.

6. Log in to the WebLogic Administration Server Console as the default administrative user. For example, `weblogic`.
7. Click **Security Realms** on the left navigation pane.
8. On the Summary of Security Realms page, click **myrealm** under the **Realms** table.
9. On the Settings page for **myrealm**, go to the **Users and Groups** tab.

> **Note:**
>
> The list of users and groups will be visible only after you restart the domain.

10. Go to the Users tab and check to see that LDAP users are displayed from the directory connector. For example: `OUDAuthenticator`.
11. Go to the Groups tab and check to see that LDAP groups are displayed from the directory connector. For example: `OUDAuthenticator`.

# Creating the Webgate_IDM Agent

If you discover that the idmConfigTool has not created the `Webgate_IDM` agent, you can create it.

To create a `Webgate_IDM` agent:

1. Create a file called `Webgate_IDM.xml` with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
 Copyright (c) 2009, 2015, Oracle and/or its affiliates. All rights
reserved.

   NAME: OAM11GRequest_short.xml - Template for OAM 11G Agent Registration
Request file
         (Shorter version - Only mandatory values - Default values will be
used for all other fields)
   DESCRIPTION: Modify with specific values and pass file as input to the
tool.

-->
<OAM11GRegRequest>

    <serverAddress>http://<OAM_DOMAIN_NAME>-
adminserver.<OAMNS>.svc.cluster.local:7001</serverAddress>
    <hostIdentifier>IAMSuiteAgent</hostIdentifier>
    <agentName>Webgate_IDM</agentName>
    <autoCreatePolicy>false</autoCreatePolicy>
    <protectedResourcesList>
        <resource>/**</resource>
    </protectedResourcesList>
    <publicResourcesList>
        <resource>/public/**</resource>
    </publicResourcesList>
    <excludedResourcesList>
        <resource>/excluded/**</resource>
    </excludedResourcesList>

</OAM11GRegRequest>
```

For example:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
 Copyright (c) 2009, 2015, Oracle and/or its affiliates. All rights
reserved.

   NAME: OAM11GRequest_short.xml - Template for OAM 11G Agent Registration
Request file
         (Shorter version - Only mandatory values - Default values will be
used for all other fields)
   DESCRIPTION: Modify with specific values and pass file as input to the
tool.

-->
<OAM11GRegRequest>

    <serverAddress>http://accessdomain-
adminserver.oamns.svc.cluster.local:7001</serverAddress>
    <hostIdentifier>IAMSuiteAgent</hostIdentifier>
    <agentName>Webgate_IDM</agentName>
    <autoCreatePolicy>false</autoCreatePolicy>
    <protectedResourcesList>
        <resource>/**</resource>
    </protectedResourcesList>
    <publicResourcesList>
        <resource>/public/**</resource>
    </publicResourcesList>
    <excludedResourcesList>
        <resource>/excluded/**</resource>
    </excludedResourcesList>

</OAM11GRegRequest>
```

Save the file.

2. Copy the file to the Kubernetes container using the following command:

```
kubectl cp /workdir/OAM/Webgate_IDM.xml <OAMNS>/<OAM_DOMAIN_NAME>-
adminserver:/u01/oracle/idm/oam/server/rreg/Webgate_IDM.xml
```

For example:

```
kubectl cp /workdir/OAM/Webgate_IDM.xml oamns/accessdomain-
adminserver:/u01/oracle/idm/oam/server/rreg/Webgate_IDM.xml
```

3. Create the WebGate Agent by using the following commands:

```
kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
cd /u01/oracle/idm/oam/server/rreg/bin
./oamreg.sh inband /u01/oracle/idm/oam/server/rreg/Webgate_IDM.xml
```

You are prompted to enter your administrative credentials. Provide the name of the OAM Administration User and the Password. In addition, you are asked whether you want to create a WebGate password. This password is optional.

For example:

```
kubectl exec -n oamns -ti accessdomain-adminserver /bin/bash

cd /u01/oracle/idm/oam/server/rreg/bin
./oamreg.sh inband /u01/oracle/idm/oam/server/rreg/Webgate_IDM.xml


----------------------------------------
Request summary:
OAM11G Agent Name:Webgate_IDM
URL String:IAMSuiteAgent
Registering in Mode:inband
Your registration request is being sent to the Admin server at: http://
accessdomain-adminserver.oamns.svc.cluster.local:7001
----------------------------------------
Mar 30, 2021 12:33:32 PM oracle.security.jps.util.JpsUtil disableAudit
INFO: JpsUtil: isAuditDisabled set to true
Inband registration process completed successfully! Output artifacts are
created in the output folder.
```

When prompted, enter your administration user name which is `oamadmin` unless you have changed it. Specify the `oamadmin` password. When asked whether you want to enter a password for WebGate, select **Yes** and specify a suitable password.

4. The WebGate artifacts are created in the `/u01/oracle/idm/oam/server/rreg/output/Webgate_IDM` directory. In a Kubernetes environment, this directory is not persistent. Therefore, you should copy the files to the `domain_home/output` directory.

   For example:

   ```
   cp -r /u01/oracle/idm/oam/server/rreg/output/Webgate_IDM /u01/oracle/
   user_projects/domains/accessdomain/output
   ```

5. Verify that the WebGate agent `Webgate_IDM` exists in the oamconsole. See Validating the OAM LDAP Integration.

# Adding LDAP Groups to WebLogic Administrators

Oracle Access Manager requires access to the MBeans stored within the Administration Server. To enable the LDAP users to log in to the WebLogic Console and Fusion Middleware Control, you must assign them the WebLogic administration rights. For Oracle Access Manager to invoke these Mbeans, users in the OAMAdministrators group must have the WebLogic administration rights.

- Using the WebLogic Console
- Using WLST

## Using the WebLogic Console

To add the LDAP Groups `OAMAdministrators` and `WLSAdministrators` to the WebLogic Administrators:

1. Log in to the WebLogic Administration Server Console as the default administrative user. For example, `weblogic`.
2. In the left pane of the console, click **Security Realms**.
3. On the Summary of Security Realms page, click **myrealm** under the Realms table.
4. On the Settings page for **myrealm**, click the **Roles & Policies** tab.
5. On the Realm Roles page, expand the **Global Roles** entry under the Roles table.
6. Click the **Roles** link to go to the Global Roles page.
7. On the Global Roles page, click the **Admin** role to go to the Edit Global Roles page.
8. On the Edit Global Roles page, under the **Role Conditions** table, click the **Add Conditions** button.
9. On the Choose a Predicate page, select **Group** from the drop down list for predicates and click **Next**.
10. On the Edit Arguments Page, Specify **OAMAdministrators** in the **Group Argument** field and click **Add**.
11. Repeat for the Group **WLSAdministrators**.
12. Click **Finish** to return to the Edit Global Roles page.
13. The **Role Conditions** table now shows the groups **OAMAdministrators** or **WLSAdministrators** as role conditions.
14. Click **Save** to finish adding the Admin role to the OAMAdministrators and IDM Administrators Groups.

## Using WLST

You can also add the LDAP Groups by using WLST:

1. Start WLST using the following command:

   ```
   ORACLE_HOME/oracle_common/common/bin/wlst.sh
   ```

2. Connect to the Administration Server using:

   ```
   connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://<OAM_DOMAIN_NAME>-
   adminserver.<OAMNS>.svc.cluster.local:<OAM_ADMIN_PORT>')
   ```

3. Use the following WLST commands after a successful connection:

   ```
   cd('/SecurityConfiguration/accessdomain/Realms/myrealm/RoleMappers/
   XACMLRoleMapper')


   cmo.setRoleExpression('', 'Admin', 'Grp(<LDAP_OAMADMIN_GRP>)|
   Grp(<LDAP_WLSADMIN_GRP>)|Grp(Administrators)')
   exit()
   ```

For example:

```
connect('weblogic','password','t3://accessdomain-
adminserver.oamns.svc.cluster.local:7001')


cd('/SecurityConfiguration/accessdomain/Realms/myrealm/RoleMappers/
XACMLRoleMapper')


cmo.setRoleExpression('', 'Admin', 'Grp(OAMAdministrators)|
Grp(WLSAdministrators)|Grp(Administrators)')
exit()
```

# Updating WebGate Agents

When you run `idmConfigTool`, it changes the default OAM security model and creates a new WebGate SSO agent. However, it does not change the existing WebGate SSO agents to the new security model. After you run `idmConfigTool`, you must update any WebGate agents that previously existed.

To update the WebGate agents:

- Change the security mode to match that of the OAM servers. Failure to do so will result in a security mismatch error.
- When WebGates are created at first install, they are unaware that a highly available (HA) installation is performed. After enabling HA, you must ensure that all of the OAM servers are included in the agent configuration, to ensure system continuity.
- When WebGates are created at first install, they are unaware that a highly available (HA) install is performed. You must check that any logout URLs are redirected to the hardware load balancer than one of the local OAM servers.
- A WebGate agent called **IAMSuiteAgent** is created out of the box. This is created without any password protection and needs to have one added.

To perform these actions, complete the following steps:

1. Log in to the OAM Console at `http://iadadmin.example.com/oamconsole` using the OAM administration user (`oamadmin`).
2. Click **Agents** pad on the Application Security screen.
3. Ensure that the **WebGates** tab is selected.
4. Click **Search**.
5. Click an agent, for example: **IAMSuiteAgent**.
6. Set the Security value to the same value you defined for **OAM Transfer Mode** on the Access Manager Configuration screen during response file creation.

   If you have changed the OAM security model using the `idmConfigTool`, change the security model used by any existing WebGates to reflect this change.

   Click **Apply**.
7. In the **Primary Server** list, click + and add any missing Access Manager Servers.
8. If a password has not already been assigned, enter a password into the **Access Client Password** field and click **Apply**.

Assign an Access Client Password, such as the **Common IAM Password** (*COMMON_IDM_PASSWORD*) you used during the response file creation or an Access Manager-specific password, if you have set one.

9. Set **Maximum Connections** to 50. This is the maximum number of connections for the primary servers, which is 10 x the number of OAM servers. In this case, the number of OAM servers is 5. Therefore, the number of connections is 10 x 5 = 50.
10. If you see the following in the **User Defined Parameters** or the **Logout redirect URL**:

    ```
    logoutRedirectUrl=http://OAMHOST1.example.com:14100/oam/server/logout
    ```

    Change it to:

    ```
    logoutRedirectUrl=https://login.example.com/oam/server/logout
    ```

11. Click **Apply**.
12. Repeat Steps through for each WebGate.
13. Check that the security setting matches that of your Access Manager servers.

# Updating Host Identifiers

When you access the domain, you enter using different load balancer entry points. You must add each of these entry points (virtual hosts) to the policy list. Adding these entry points ensures that if you request access to a resource using `login.example.com` or `prov.example.com`, you have access to the same set of policy rules.

To update the host identifiers:

1. Access the OAM Console at `http://iadadmin.example.com/oamconsole`.
2. Log in as the Access Manager administration user you created when you prepared the ID Store. For example: `oamadmin`.
3. Select **Launch Pad** if not already displayed.
4. Click on **Host Identifiers** under **Access Manager**.
5. Click **Search**.
6. Click on **IAMSuiteAgent**.
7. Click **+** in the operations box.
8. Enter the following information.

**Table 18-4 Host Name Port Values**

| Host Name | Port |
|---|---|
| `iadadmin.example.com` | 80 |
| `igdadmin.example.com` | 80 |
| `igdinternal.example.com` | 7777 |
| `prov.example.com` | 443 |
| `login.example.com` | 443 |
| `ohs1.example.com` | 7777 |
| `ohs2.example.com` | 7777 |

9. Click **Apply**.

# Adding the Missing Policies to OAM

If any policies are missing, you have to add to ensure that Oracle Access Manager functions correctly.

You need to add the following additional policies:

**Table 18-5 OAM Policy Information**

| Product | Resource Type | Host Identifier | Resource URL | Protection Level | Authentication Policy | Authorization Policy |
|---|---|---|---|---|---|---|
| ALL | HTTP | IAMSuiteAgent | `/consolehelp/**` | Excluded | | |
| ALL | HTTP | IAMSuiteAgent | `/management/**` | Excluded | | |
| ALL | HTTP | IAMSuiteAgent | `/otpfp/**` | Excluded | | |
| ALL | HTTP | IAMSuiteAgent | `/dms/**` | Excluded | | |
| OIG | HTTP | IAMSuiteAgent | `/OIGUI/**` | Protected | Protected Higher Level Policy | Protected Resource Policy |
| OAM | HTTP | IAMSuiteAgent | `/iam/access/binding/api/v10/oap/**` | Excluded | | |
| OAM | HTTP | IAMSuiteAgent | `/oam/services/rest/**` | Excluded | | |
| OAM | HTTP | IAMSuiteAgent | `/iam/admin/config/api/v1/config/**` | Excluded | | |
| OAM | HTTP | IAMSuiteAgent | `/oauth2/rest/**` | Excluded | | |
| OAM | HTTP | IAMSuiteAgent | `/.well-known/openid-configuration` | Excluded | | |
| OIG | HTTP | IAMSuiteAgent | `/iam/**` | Protected | Protected Higher Level Policy | Protected Resource Policy |
| OIG | HTTP | IAMSuiteAgent | `/iam/governance/**` | Excluded | | |
| OIG | HTTP | IAMSuiteAgent | `/FacadeWebApp/**` | Protected | Protected Higher Level Policy | Protected Resource Policy |
| OIG | HTTP | IAMSuiteAgent | `/IdentityAuditCallbackService/**` | Excluded | | |

**Table 18-5 (Cont.) OAM Policy Information**

| Product | Resource Type | Host Identifier | Resource URL | Protection Level | Authentication Policy | Authorization Policy |
|---|---|---|---|---|---|---|
| OIG | HTTP | IAMSuiteAgent | /soa/composer | Protected | Protected Higher Level Policy | Protected Resource Policy |
| OIG | HTTP | IAMSuiteAgent | /soa-infra | Protected | Protected Higher Level Policy | Protected Resource Policy |
| OIG | HTTP | IAMSuiteAgent | /integration/** | Protected | Protected Higher Level Policy | Protected Resource Policy |
| OUDSM | HTTP | IAMSuiteAgent | /oudsm | Excluded | | |
| OIRI | HTTP | IAMSuiteAgent | /oiri/api/** | Excluded | | |
| OIRI | HTTP | IAMSuiteAgent | /oiri/ui/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-admin/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /admin-ui/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /policy/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-policy/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-email-factor/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-sms-factor/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-totp-factor/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-yotp-factor/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /fido/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-kba/** | Excluded | | |
| OAA | HTTP | IAMSuiteAgent | /oaa-push-factor/** | Excluded | | |
| OARM | HTTP | IAMSuiteAgent | /risk-analyzer/** | Excluded | | |
| OARM | HTTP | IAMSuiteAgent | /risk-cc/** | Excluded | | |

**Table 18-5 (Cont.) OAM Policy Information**

| Product | Resource Type | Host Identifier | Resource URL | Protection Level | Authentication Policy | Authorization Policy |
|---|---|---|---|---|---|---|
| OUA | HTTP | IAMSuiteAgent | `/oua/**` | Excluded | | |
| OUA | HTTP | IAMSuiteAgent | `/oua-admin-ui/**` | Excluded | | |
| OUA | HTTP | IAMSuiteAgent | `/oaa-drss/**` | Excluded | | |

> **Note:**
>
> `/otpfp` is only required if you have implemented the OAM forgotten password functionality.
>
> `/management` is only required if you are using the WebLogic Remote Console.

To add these policies:

1. Log in to the OAM Console at `http://iadadmin.example.com/oamconsole` using the `oamadmin` user.
2. From the Launch pad click **Application Domains** in the **Access Manager** section.
3. Click **Search** on the Search page.

   A list of application domains appears.
4. Click the domain **IAM Suite**.
5. Click the **Resources** Tab.
6. Click **Create**.
7. Enter the information specified in the table above.
8. Click **Apply**.

# Validating the Authentication Providers

Set the order of identity assertion and authentication providers in the WebLogic Server Administration Console.

To set the order:

1. Log in to the WebLogic Server Administration Console, if not already logged in.
2. Click **Lock & Edit**.
3. From the left navigation, select **Security Realms**.
4. Click the **myrealm** default realm entry.
5. Click the **Providers** tab.
6. From the table of providers, click the **DefaultAuthenticator**.

7. Set the Control Flag to `SUFFICIENT`.
8. Click **Save** to save the settings.
9. From the navigation breadcrumbs, click **Providers** to return to the list of providers.
10. Click **Reorder**.
11. Sort the providers to ensure that the OAM Identity Assertion provider is first and the DefaultAuthenticator provider is last.

**Table 18-6 Sort order**

| Sort Order | Provider | Control Flag |
|---|---|---|
| 1 | OAMIDAsserter | `REQUIRED` |
| 2 | LDAP Authentication Provider | `SUFFICIENT` |
| 3 | DefaultIdentityAsserter | `N/A` |
| 4 | Trust Service Identity Asserter | `N/A` |
| 5 | DefaultAuthenticator | `SUFFICIENT` |

12. Click **OK**.
13. Click **Activate Changes** to propagate the changes.

# Configuring Oracle ADF and OPSS Security with Oracle Access Manager

Some Oracle Fusion Middleware management consoles use Oracle Application Development Framework (Oracle ADF) security, which can integrate with Oracle Access Manager Single Sign-on (SSO). These applications can take advantage of Oracle Platform Security Services (OPSS) SSO for user authentication, but you must first configure the domain-level `jps-config.xml` file to enable these capabilities.

The domain-level `jps-config.xml` file is located in the following location after you create an Oracle Fusion Middleware domain:

```
/u01/oracle/user_projects/domain/<OAM_DOMAIN_NAME>/config/fmwconfig/
jps-config.xml
```

> **Note:**
>
> The domain-level `jps-config.xml` should not be confused with the `jps-config.xml` that is deployed with custom applications.

To update the OPSS configuration to delegate SSO actions in Oracle Access Manager, complete the following steps:

1. Connect to the AdminServer container using the following command:

   ```
   kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
   ```

2. Start the WebLogic Server Scripting Tool (WLST):

*ORACLE_COMMON_HOME*/common/bin/wlst.sh

For example:

```
/u01/oracle/oracle_common/common/bin/wlst.sh
```

3. Connect to the Administration Server, by using the following WLST command:

```
connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://<OAM_DOMAIN_NAME>-
adminserver.<OAMNS>.svc.cluster.local:<OAM_ADMIN_PORT>')
```

For example:

```
connect('weblogic','<password>','t3://accessdomain-
adminserver.oamns.svc.cluster.local:7001')
```

4. Run the `addOAMSSOProvider` command, as shown:

```
addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication",
logouturi="/oam/logout.html")
```

5. Disconnect from the Administration Server by entering the following command:

```
exit()
```

6. Exit from the container using the `exit` command.

# Enabling Forgotten Password

This section describes how to set up the One Time Pin forgotten password functionality which is provided with Oracle Access Manager

If you want to configure the Challenge Question forgotten password functionality, as provided by Oracle Identity Governance, see Configuring and Integrating with LDAP and Integrating Oracle Identity Governance and Oracle Access Manager.

- Prerequisites for Enabling Forgotten Password
- Adding Permissions to the oamLDAP User
- Enabling Adaptive Authentication Service
- Configuring Adaptive Authentication Plug-in
- Enabling Password Management in the Directory
- Storing the User Messaging Credentials in CSF
- Setting Up the Forgot Password Link on the Login Page
- Adding the Oracle Access Manager Load Balancer Certificate to the Oracle Keystore Service
- Configuring a Custom Host Name Verifier
- Validating the Forgotten Password Functionality

## Prerequisites for Enabling Forgotten Password

Forgotten Password Management in Oracle Access Manager takes the form of sending an Email or SMS message with a link to reset the password.

Email or SMS is sent using the Oracle User Messaging Service. Before enabling the Oracle Forgotten Password functionality, you first need to have an Oracle User Messaging deployment. This is often located inside the Oracle Governance Domain but can be located inside the Access Domain if that is all you are installing. Alternatively, it could be a completely independent domain.

Forgotten Password functionality works only if you have successfully configured Single Sign-On as described in Configuring Single Sign-On for an Enterprise Deployment.

Adding the User Messaging Service to the Access domain or creating a User Messaging Service domain is outside of the scope of the this EDG. For more information about installing and configuring the Oracle User Messaging Service, see Installing User Messaging Service and Configuring Oracle User Messaging Service in *Administering Oracle User Messaging Service*.

## Adding Permissions to the oamLDAP User

When created out of the box, the oamLDAP user (the user who links OAM to LDAP) is granted privileges to read user data in the LDAP directory. However, the oamLDAP user is not granted permission to update users information. You need to add these privileges for the OAM forgotten password functionality to work.

To add the privileges:

1. Create the `ldif` file outside of the OUD host, using your preferred text editor, and then copy the file to the LDAPHOST1 machine.
   This file has the following content:

```
dn: cn=Users,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*")(targetfilter=
"(objectclass=inetorgperson)")(targetscope = "subtree") (version 3.0; acl
"iam admin changepwd"; allow
(compare,search,read,selfwrite,add,write,delete)
userdn = "ldap:///cn=oamLDAP,cn=systemids,dc=example,dc=com";)
```

2. Save the file.

> **Note:**
>
> You should create this file outside of the 'oud' container, and then copy it into the container.
>
> If you have mounted the OUD config persistent volume to your install host, you can directly create the file in that location. To mount the config file, see Installing and Configuring Oracle Unified Directory.

To copy the file to the OUD node, use the following command:

```
kubectl cp add_aci.ldif oudns/edg-oud-ds-rs-0:add_aci.ldif
```

Connect to the OUD node to run the `ldapmodify` command, using the following command:

```
kubectl exec -it  edg-oud-ds-rs-0 -n oudns -- /bin/bash
```

Add the ACI to OUD by using the following command:

```
/u01/oracle/user_projects/edg-oud-ds-rs-0/OUD/bin/ldapmodify -c -D
cn=oudadmin -h edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local -p 1389 -f /u01/
oracle/config-input/add_aci.ldif
```

# Enabling Adaptive Authentication Service

Forgotten password requires you to enable the Adaptive Authentication Service.

To enable this service:

1. Log in to the Oracle Access Management Administration Console as the `oamadmin` user, using the following URL:

   `http://iadadmin.example.com/oamconsole`

2. Click **Configuration**.
3. Click **Available Services**.
4. Click **Enable Service** next to Adaptive Authentication Service.
5. When prompted, confirm that you want to enable the service.

# Configuring Adaptive Authentication Plug-in

After you enable the Adaptive Authentication Service, it needs to be informed about the User Messaging Service.

To configure the Adaptive Authentication Plug-in:

1. Log in to the Oracle Access Management Administration Console as the `oamadmin` user, using the following URL:

   `http://iadadmin.example.com/oamconsole`

2. From the Application Security Launch Pad, click **Authentication Plug-ins** in the Plug-ins panel. From the Authentication Plug-in tab, type **Adaptive** in the quick search box above the Plug-in Name column and hit **Enter**.

   The **AdaptiveAuthenticationPlugin** is displayed.

3. Enter the following plug in properties:

**Table 18-7 AdaptiveAuthentication Plug-In Properties**

| Attribute | Value |
|---|---|
| UmsAvailable | True |

**Table 18-7 (Cont.) AdaptiveAuthentication Plug-In Properties**

| Attribute | Value |
|---|---|
| UmsClientURL | Specify the entry point of your User Messaging service. If you have configured Oracle Identity Manager, then this will be:`http://igdinternal.example.com:7777/ucs/messaging/webservice`<br>If your UMS server is inside the Kubernetes cluster, you can access it using the Kubernetes service name. For example: `http://<OIG_DOMAIN_NAME>-cluster-soa-cluster.<OIGNS>.svc.cluster.local:8001/ucs/messaging/webservice`. |

4. Click **Save**.

# Enabling Password Management in the Directory

By default, OAM is not set to enable password management. You have to enable it through the OAM Console.

To enable Password Management in the directory:

1. Log in to the Oracle Access Management Administration Console as the `oamadmin` user, using the following URL:

   `http://iadadmin.example.com/oamconsole`

2. Click **Configuration**.
3. Click **User Identity Stores**.
4. Click the LDAP identity store in the OAM Identity Store section. For example: `OAMIDSTORE`.
5. Click **Edit**.
6. Select **Enable Password Management**.
7. Enter the details in the **User Information** field.

**Table 18-8 User Information Details**

| Attribute | Description |
|---|---|
| Global Common ID Attribute | The unique identifier in LDAP for the user. For example: **uid**. |
| First Name | The LDAP attribute which holds the users name. For example: **cn**. |
| Last Name | The LDAP attribute which holds the users last name. For example: **sn**. |
| Email Address | The LDAP attribute which holds the user's email address. For example: **mail**. |

8. Click **Apply**.

# Storing the User Messaging Credentials in CSF

Before you can access the User Messaging Service, you need to store the credentials in the WebLogic credential store.

To store the credentials:

1. Connect to the Administration Server by using the following command:

```
kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
```

2. Execute the following set of WLST commands:

```
/u01/oracle/oracle_common/common/bin/wlst.sh
connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://<OAM_DOMAIN_NAME>-
adminserver.<OAMNS>.svc.cluster.local:<OAM_ADMIN_PORT>').
createCred(map="OAM_CONFIG", key="umsKey", user="weblogic",
password="password")
createCred(map="OAM_CONFIG", key="oam_rest_cred", user="oamadmin",
password="password")
exit ()
```

The **umsKey** is used to provide the credentials to the unified messaging server that sends out your email or sms notifications.

The **oam_rest_cred** is the user who is allowed to invoke the REST Services in the OAM Server.

In the above commands, `weblogic` is the domain administrative user and `password` is its associated password.

# Setting Up the Forgot Password Link on the Login Page

The following REST API command enables the OTP forgot password link on the default login page in OAM.

```
 curl -X -k PUT \
  https://login.example.com/oam/services/rest/access/api/v1/config/
otpforgotpassword/ \
  -u oamadmin:Password \
  -H 'content-type: application/json' \
  -d
'{"displayOTPForgotPassworLink":"true","defaultOTPForgotPasswordLink":"false",
"localToOAMServer":"true","forgotPasswordURL":"https://login.example.com/
otpfp/pages/fp.jsp", "mode":"userselectchallenge"}'
```

Enter the required attributes and values:

**Table 18-9 Forgot Password Link on Login Page**

| Attributes | Value |
|---|---|
| `ForgotPasswordURL` | The OAM Forgotten Password URL. For example, https://login.example.com/otpfp/pages/fp.jsp |

**Table 18-9 (Cont.) Forgot Password Link on Login Page**

| Attributes | Value |
|---|---|
| `mode` | distribution_mode<br>The distribution mode determines how the password reset URL is sent to the end user. Valid values are: **email, sms, userchoose, userselectchallenge**. The last entry enables the user to choose from masked values.<br>• **Email** - An OTP is sent to the email configured in the mail field.<br>• **SMS** - An OTP is sent to the mobile number configured in the mobile field.<br>• **Userchoose** - An OTP is sent by letting the user choose either the email or the mobile option, without the exact values.<br>• **Userselectchallenge** - User can see the masked values either as email or mobile and select one of the options. |

**Note:**

If you are using self signed certificates in the load balancer the curl command may object with a message similar to:
curl performs SSL certificate verification by default, using a **bundle** of Certificate Authority (CA) public keys (CA certs). If the default bundle file isn't adequate, you can specify an alternate file using the --cacert option. If this HTTPS server uses a certificate signed by a CA represented in the bundle, the certificate verification probably failed due to a problem with the certificate (it might be expired, or the name might not match the domain name in the URL). If you like to turn off curl's verification of the certificate, use the **-k (or --insecure)** option.

If you see this message and are sure, add **-k after -u oamadmin:Password**.

Verify that this has succeeded by accessing the followig URL in a browser:

`https://login.example.com/oam/services/rest/access/api/v1/config/otpforgotpassword`

When prompted, enter your `oamadmin` account and password.

**Note:**

One of the OAM Managed Servers must be running for this command to succeed.

# Adding the Oracle Access Manager Load Balancer Certificate to the Oracle Keystore Service

The Oracle Access Manager forgot password functionality requires that the SSL certificate used by the load balancer be added to the Oracle Keystore Service Trusted Certificates.

To add the certificate, do the following:

1. Create a directory to hold user created keystores and certificates.

   ```
   kubectl exec -n <OAMNS> -ti <OAM_DOMAIN_NAME>-adminserver -- mkdir -p
   SHARED_CONFIG_DIR/keystores
   ```

   For example:

   ```
   kubectl exec -n oamns -ti accessdomain-adminserver -- mkdir -p /u01/oracle/
   user_projects/keystores
   ```

2. Obtain the certificate from the load balancer. You can obtain the load balancer certificate from using a browser, such as Firefox. However, the easiest way to obtain the certificate is to use the `openssl` command. The syntax of the command is as follows:

   ```
   openssl s_client -connect <LOADBALANCER>:<PORT> -showcerts </dev/null
   2>/dev/null| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /
   workdir/OAM/LOADBALANCER.pem
   ```

   For example:

   ```
   openssl s_client -connect login.example.com:443 -showcerts </dev/null
   2>/dev/null| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > >/
   workdir/OAM/login.example.com.pem
   ```

   The `openssl` command saves the certificate to a file called `login.example.com.pem` in `/workdir/OAM`.

3. Copy the certificate to the Kubernetes container by using the following command:

   ```
   kubectl cp <FILENAME> <OAMNS>/<OAM_DOMAIN_NAME>-
   adminserver:<SHARED_CONFIG_DIR>/keystores
   ```

   For example:

   ```
   kubectl cp login.example.com.pem oamns/accessdomain-adminserver:/u01/
   oracle/user_projects/keystores/login.example.com.pem
   ```

4. Load the certificate into the Oracle Keystore Service using WLST.

   a. Connect to the container using the command:

      ```
      kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
      ```

b. Connect to WLST by using the following command:

```
ORACLE_HOME/oracle_common/common/bin/wlst.sh
```

c. Connect to the Administration Server using the following command:

```
connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://
<OAM_DOMAIN_NAME>-
adminserver.<OAMNS>.svc.cluster.local:<OAM_ADMIN_PORT>')
```

For example:

```
connect('weblogic','<password>','t3://accessdomain-
adminserver.oamns.svc.cluster.local:7001')
```

d. Download the access artifacts by using the following command:

```
downloadAccessArtifacts(domain_home="/u01/oracle/user_projects/domains/
accessdomain",
      propsFile="/u01/oracle/user_projects/workdir/db.props"
```

> **Note:**
>
> For information about the contents of the properties file, see Doc ID 2318818.1.

e. Load the certificate using the following commands:

```
svc = getOpssService(name='KeyStoreService')
svc.importKeyStoreCertificate(appStripe='system',name='trust',password='
', keypassword='',alias='<CertificateName>',type='TrustedCertificate',
filepath='/<SHARED_CONFIG_DIR>/keystores/<LOADBALANCER>.pem')
```

f. Synchronize the keystore service with the file system by using the following command:

```
syncKeyStores(appStripe='system', keystoreFormat='KSS')
```

For example:

```
connect('weblogic','password','t3://accessdomain-
adminserver.oamns.svc.cluster.local:7101')
svc = getOpssService(name='KeyStoreService')
svc.importKeyStoreCertificate(appStripe='system',name='trust',password='
', keypassword='',alias='login.example.com',type='TrustedCertificate',
filepath='/u01/oracle/user_projects/keystores/login.example.com.pem')
syncKeyStores(appStripe='system',keystoreFormat='KSS')
exit()
```

g. Save the access artifacts by using the following command:

```
saveAccessArtifacts(domain_home="/u01/oracle/user_projects/domains/
accessdomain",
      propsFile="/u01/oracle/user_projects/work/db.props"
```

You will need to restart the domain for the changes to take effect. The default password for the Node Manager keystores is *COMMON_IAM_PASSWORD*. You will be prompted to confirm that the certificate is valid.

## Configuring a Custom Host Name Verifier

If using a wildcard certificate, you have to change the default value for the host name verifier from `BEA Hostname Verifier` to the custom value `weblogic.security.utils.SSLWLSWildcardHostnameVerifier`.

To change the default value of the host name verifier:

1. Log in to the WebLogic Server Administration Console.
2. Click **Lock & Edit**.
3. Navigate to **Summary of Servers** and click **oam_server1**.
4. Click the **SSL** tab, expand the **Advanced** section.
5. Set the **Hostname Verification** field to **Custom Hostname Verifier**.
6. In the **Custom Hostname Verifier** field, enter `weblogic.security.utils.SSLWLSWildcardHostnameVerifier`.
7. Click **Save**.

> **Note:**
>
> Follow these steps to change the host name verifier for all the OAM Managed Servers.

## Validating the Forgotten Password Functionality

If you have set up the OAM Forgotten Password functionality, rather than off-loading to OIM, you can validate the forgotten password using the curl command, which shows you the password policies in force.

To validate the Forgotten Password functionality, run the following `curl` command:

```
curl -X GET https://login.example.com/oam/services/rest/access/api/v1/pswdmanagement/
UserPasswordPolicyRetriever/oamadmin?description=true  -u oamadmin:<password> -k
```

This command displays the password policies.

If this command works, access the protected URL listed below. After you enable single sign-on, you see a link for the forgotten password on the login page. Click this link and enter the user name for which you want to reset the password. Click **Generate Pin** to receive an email, which enables you to change the password.

```
http://iadadmin.example.com/console
```

> **Note:**
>
> Before validating, ensure that you enable SSO as described in Configuring Single Sign-On for an Enterprise Deployment. Else, validation fails.

# Restarting the Access Domain

Restart the Access domain in Kubernetes for the changes to take effect.

To restart the domain:

1. Stop the domain.

```
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "NEVER" }]'
```

2. Start the domain.

```
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IF_NEEDED" }]'
```

# Setting the Initial Server Count

When you first created the domain, you specified that only one Managed Server should be started. After you complete the configuration, you can increase the initial server count to the actual number you require.

To increase the server count, use the following command:

```
kubectl patch cluster -n <OAMNS> <OAM_DOMAIN_NAME>-${CLUSTER_NAME} --
type=merge -p '{"spec":{"replicas":<INITIAL_SERVER_COUNT>}}'
```

If you want two OAM and two Policy Manager Managed Servers, use the following commands:

```
kubectl patch cluster -n oamns accessedomain-oam-cluster --type=merge -p
'{"spec":{"replicas":2}}'
```

```
kubectl patch cluster -n oamns accessdomain-policy-cluster --type=merge -p
'{"spec":{"replicas":2}}'
```

# Centralized Monitoring Using Grafana and Prometheus

If you are using a centralized Prometheus and Grafana deployment to monitor your infrastructure, you can send Oracle Access Management data to this application. If you have not yet deployed Prometheus and Grafana, see Installing Prometheus and Grafana.
To use the centralized Prometheus and Grafana for monitoring your infrastructure, perform the following steps:

- Downloading and Compiling the Monitoring Application
- Deploying the Monitoring Application into the WebLogic Domain

- Configuring the Prometheus Operator
- Discovering the Prometheus Service

# Downloading and Compiling the Monitoring Application

To download and configure the monitoring application for the Oracle Access Manager WebLogic cluster:

1. Change the directory to the sample scripts which set up monitoring:

   ```
   cd <WORKDIR>/samples/monitoring-service/scripts
   ```

   For example:

   ```
   cd /workdir/OAM/samples/monitoring-service/scripts
   ```

2. Run the `get-wls-exporter.sh` script.

   Before you run the script, set the environment variables that determine your environment setup:

   ```
   export adminServerPort=<OAM_ADMIN_PORT>
   ```

   ```
   export wlsMonitoringExporterTopolicyCluster=true
   ```

   ```
   export policyManagedServerPort=15100
   ```

   ```
   export wlsMonitoringExporterTooamCluster=true
   ```

   ```
   export oamManagedServerPort=14100
   ```

   ```
   export domainNamespace=<OAMNS>
   ```

   ```
   export domainUID=<OAM_DOMAIN_NAME>
   ```

   ```
   export weblogicCredentialsSecretName=<OAM_DOMAIN_NAME>-credentials
   ```

For example:

```
export adminServerPort=7001


export wlsMonitoringExporterTopolicyCluster=true


export policyManagedServerPort=15100


export wlsMonitoringExporterTooamCluster=true


export oamManagedServerPort=14100


export domainNamespace=oamns


export domainUID=accessdomain


export weblogicCredentialsSecretName=accessdomain-credentials
```

Execute the script using the following command:

```
./get-wls-exporter.sh
```

The output appears as follows:

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed

  0     0    0     0    0     0      0     0 --:--:-- --:--:--
--:--:--     0
  0     0    0     0    0     0      0     0 --:--:-- --:--:--
--:--:--     0

  0     0    0     0    0     0      0     0 --:--:-- --:--:--
--:--:--     0
  5 2196k    5  127k    0     0  74408     0  0:00:30  0:00:01  0:00:29
129k
100 2196k  100 2196k    0     0  1017k     0  0:00:02  0:00:02 --:--:--
1582k
created /home/opc/workdir/OAM/samples/monitoring-service/scripts/wls-
exporter-deploy dir
adminServerName is empty, setting to default "AdminServer"
oamClusterName is empty, setting to default "oam_cluster"
policyClusterName is empty, setting to default "policy_cluster"
created /tmp/ci-x9jR7O0jjm
/tmp/ci-x9jR7O0jjm ~/workdir/OAM/samples/monitoring-service/scripts
in temp dir
```

```
  adding: WEB-INF/weblogic.xml (deflated 61%)
  adding: config.yml (deflated 60%)
~/workdir/OAM/samples/monitoring-service/scripts
created /tmp/ci-mRTU0hjM1q
/tmp/ci-mRTU0hjM1q ~/workdir/OAM/samples/monitoring-service/scripts
in temp dir
  adding: WEB-INF/weblogic.xml (deflated 61%)
  adding: config.yml (deflated 60%)
~/workdir/OAM/samples/monitoring-service/scripts
created /tmp/ci-OD0EAjb7M5
/tmp/ci-OD0EAjb7M5 ~/workdir/OAM/samples/monitoring-service/scripts
in temp dir
  adding: WEB-INF/weblogic.xml (deflated 61%)
  adding: config.yml (deflated 60%)
~/workdir/OAM/samples/monitoring-service/scripts
```

# Deploying the Monitoring Application into the WebLogic Domain

The earlier section created a number of WAR files containing the monitoring application. See Downloading and Compiling the Monitoring Application. These files need to be deployed inside the WebLogic domain. Oracle provides a script to deploy the files. Before you run the script, copy the files to the container containing the WebLogic Administration Server.

To deploy the application:

1. Change directory to the sample file location:

   ```
   cd <WORKDIR>/samples/monitoring-service/scripts
   ```

   For example:

   ```
   cd /workdir/OAM/samples/monitoring-service/scripts
   ```

2. Copy files to the Administration Server container by using the following commands:

   ```
   kubectl cp <WORKDIR>/samples/monitoring-service/scripts/wls-exporter-
   deploy  <OAMNS>/<OAM_DOMAIN_NAME>-adminserver:/u01/oracle
   ```

   ```
   kubectl cp <WORKDIR>/samples/monitoring-service/scripts/deploy-weblogic-
   monitoring-exporter.py  <OAMNS>/<OAM_DOMAIN_NAME>-adminserver:/u01/oracle/
   wls-exporter-deploy
   ```

   For example:

   ```
   kubectl cp /workdir/OAM/samples/monitoring-service/scripts/wls-exporter-
   deploy  oamns/accessdomain -adminserver:/u01/oracle
   ```

   ```
   kubectl cp /workdir/OAM/samples/monitoring-service/scripts/deploy-weblogic-
   monitoring-exporter.py  oamns/accessdomain-adminserver:/u01/oracle/wls-
   exporter-deploy
   ```

3. Deploy the application using the following command:

```
kubectl exec -it -n <OAMNS> <OAM_DOMAIN_NAME>-adminserver -- /u01/oracle/
oracle_common/common/bin/wlst.sh \
-domainName <OAM_DOMAIN_NAME> \
-adminServerName AdminServer \
-adminURL <OAM_DOMAIN_NAME>-adminserver:<OAM_ADMIN_PORT> \
-username <OAM_WEBLOGIC_USER> \
-password <OAM_WEBLOGIC_PWD> \
-oamClusterName oam_cluster \
-wlsMonitoringExporterTooamCluster true \
-policyClusterName policy_cluster \
-wlsMonitoringExporterTopolicyCluster true
```

For example:

```
kubectl exec -it -n oamns acccessdomain-adminserver -- /u01/oracle/
oracle_common/common/bin/wlst.sh \
-domainName accessdomain \
-adminServerName AdminServer \
-adminURL accessdomain-adminserver:7001 \
-username weblogic \
-password MyPassword \
-oamClusterName oam_cluster \
-wlsMonitoringExporterTooamCluster true \
-policyClusterName policy_cluster \
-wlsMonitoringExporterTopolicyCluster true
```

The output appears as follows:

```
Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

Connecting to t3://accessdomain-adminserver:7001 with userid weblogic ...
Successfully connected to Admin Server "AdminServer" that belongs to
domain "accessdomain".

Warning: An insecure protocol was used to connect to the server.
To ensure on-the-wire security, the SSL port or Admin port should be used
instead.

Deploying .........
Deploying application from /u01/oracle/wls-exporter-deploy/wls-exporter-
adminserver.war to targets AdminServer (upload=true) ...
<Aug 18, 2022 10:12:01 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating deploy operation for application, wls-exporter-adminserver
[archive: /u01/oracle/wls-exporter-deploy/wls-exporter-adminserver.war],
to AdminServer .>
.Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
```

```
Deployment Message : no message
Starting application wls-exporter-adminserver.
<Aug 18, 2022 10:12:07 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, wls-exporter-adminserver
[archive: null], to AdminServer .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State : completed
Deployment Message : no message
Deploying .........
Deploying application from /u01/oracle/wls-exporter-deploy/wls-exporter-
oam.war to targets oam_cluster (upload=true) ...
<Aug 18, 2022 10:12:10 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating deploy operation for application, wls-exporter-oam
[archive: /u01/oracle/wls-exporter-deploy/wls-exporter-oam.war], to
oam_cluster .>
..Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
Starting application wls-exporter-oam.
<Aug 18, 2022 10:12:18 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, wls-exporter-oam [archive:
null], to oam_cluster .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State : completed
Deployment Message : no message
Deploying .........
Deploying application from /u01/oracle/wls-exporter-deploy/wls-exporter-
policy.war to targets policy_cluster (upload=true) ...
<Aug 18, 2022 10:12:22 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating deploy operation for application, wls-exporter-policy
[archive: /u01/oracle/wls-exporter-deploy/wls-exporter-policy.war], to
policy_cluster .>
.Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
Starting application wls-exporter-policy.
<Aug 18, 2022 10:12:26 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, wls-exporter-policy [archive:
null], to policy_cluster .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State : completed
Deployment Message : no message
Disconnected from weblogic server: AdminServer


Exiting WebLogic Scripting Tool.
```

```
<Aug 18, 2022 10:12:30 AM GMT> <Warning> <JNDI> <BEA-050001>
<WLContext.close() was called in a different thread than the one in which
it was created.>
```

# Configuring the Prometheus Operator

Prometheus enables you to collect metrics from the WebLogic Monitoring Exporter. The Prometheus Operator identifies the targets by using service discovery. To get the WebLogic Monitoring Exporter end point discovered as a target, you must create a service monitor that points to the service.

The exporting of metrics from `wls-exporter` requires `basicAuth`. Therefore, a Kubernetes secret is created with the user name and password that are `base64` encoded. This secret is used in the `ServiceMonitor` deployment. The `wls-exporter-ServiceMonitor.yaml` file has `basicAuth` with credentials as username: <*OAM_WEBLOGIC_USER*> and password: <*OAM_WEBLOGIC_PWD*> in `base64` encoded.

1. Run the following command to get the `base64` encoded version of the `weblogic` username:

   ```
   echo -n "weblogic” | base64
   ```

   The output appears as follows:

   ```
   d2VibG9naWM=
   ```

2. Run the following command to get the `base64` encoded version of the `weblogic` password:

   ```
   echo -n "<OAM_WEBLOGIC_PWD>" | base64
   ```

   The output appears as follows:

   ```
   V2VsY29tZTE=
   ```

3. Copy the template file `<WORKDIR>/samples/monitoring-service/manifests/wls-exporter-ServiceMonitor.yaml.template` to `<WORKDIR>/samples/monitoring-service/manifests/wls-exporter-ServiceMonitor.yaml`.

4. Update the `<WORKDIR>/samples/monitoring-service/manifests/wls-exporter-ServiceMonitor.yaml` location and change the user and password values to the values returned in Step 2.

   Also, change the values of the following to match the OAM namespace, domain name, and Prometheus release name. For example:

   - `namespace: oamns`
   - `weblogic.domainName: accessdomain`
   - `release: kube-prometheus`
     You can get the release name by using the command:

     ```
     kubectl get prometheuses.monitoring.coreos.com --all-namespaces -o
     jsonpath="{.items[*].spec.serviceMonitorSelector}"
     ```

For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: basic-auth
  namespace: oamns
data:
  password: V2VsY29tZTE=
  user: d2VibG9naWM=
type: Opaque
---
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: wls-exporter
  namespace: oamns
  labels:
    k8s-app: wls-exporter
    release: monitoring
spec:
  namespaceSelector:
    matchNames:
    - oamns
  selector:
    matchLabels:
      weblogic.domainName: accessdomain
  endpoints:
  - basicAuth:
      password:
        name: basic-auth
        key: password
      username:
        name: basic-auth
        key: user
    port: default
    relabelings:
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
    interval: 10s
    honorLabels: true
    path: /wls-exporter/metrics
```

5. Update the `<WORKDIR>/samples/monitoring-service/manifests/prometheus-roleSpecific-domain-namespace.yaml` and `<WORKDIR>/samples/monitoring-service/manifests/prometheus-roleBinding-domain-namespace.yaml` files and change the namespace to match the OAM namespace. For example:

   **prometheus-roleSpecific-domain-namespace.yaml**

```
apiVersion: rbac.authorization.k8s.io/v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: Role
  metadata:
    name: prometheus-k8s
```

```
    namespace: oamns
  rules:
  - apiGroups:
    - ""
    resources:
    - services
    - endpoints
    - pods
    verbs:
    - get
    - list
    - watch
kind: RoleList
```

**prometheus-roleBinding-domain-namespace.yaml:**

```
apiVersion: rbac.authorization.k8s.io/v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    name: prometheus-k8s
    namespace: oamns
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: prometheus-k8s
  subjects:
  - kind: ServiceAccount
    name: prometheus-k8s
    namespace: monitoring
kind: RoleBindingList
```

6. Run the following command to enable Prometheus:

```
kubectl apply -f
```

The output appears as follows:

```
rolebinding.rbac.authorization.k8s.io/prometheus-k8s created
role.rbac.authorization.k8s.io/prometheus-k8s created
secret/basic-auth created
servicemonitor.monitoring.coreos.com/wls-exporter created
```

# Discovering the Prometheus Service

After you deploy `ServiceMonitor`, `wls-exporter` is discovered by Prometheus and is able to collect the metrics.

1. Access the following URL to view the Prometheus service discovery:

   `http://<K8_WORKER1>:32101/service-discovery`

2. Click `<OAMNS>/wls-exporter/0`, and then click **Show More**. Verify that all the targets are listed.

# Centralized Log File Monitoring Using Elasticsearch and Kibana

If you are using Elasticsearch and Kibana, you can configure a Logstash pod to send the log files to the centralized Elasticsearch/Kibana console. Before you configure the Logstash pod, ensure that you have access to a centralized Elasticsearch deployment.

- OAM persistent volume, so it can be loaded by the Logstash pod to hunt for log files.
- The location of the log files in the persistent volumes.
- The location of the centralized Elasticsearch.

To configure the Logstash pod, perform the following steps. The assumption is that you have an Elasticsearch running inside the Kubernetes cluster, in a namespace called `elkns`.

- Creating a Secret for Elasticsearch
- Creating a Configuration Map for ELK Certificate
- Creating a Configuration Map for Logstash
- Creating a Logstash Deployment

## Creating a Secret for Elasticsearch

Logstash requires credentials to connect to the elasticsearch deployment. These credentials are stored in Kubernetes as a secret.

If your Elasticsearch uses an API key for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OAMNS> --from-
literal password=<ELK_APIKEY>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oamns --from-
literal password=afshfashfkahf5f
```

If Elasticsearch uses a user name and password for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OAMNS> --from-
literal password=<ELK_PWD>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oamns --from-
literal password=mypassword
```

You can find the Elasticsearch password using the following command:

```
kubectl get secret elasticsearch-es-elastic-user -n <ELKNS> -o go-
template='{{.data.elastic | base64decode}}'
```

# Creating a Configuration Map for ELK Certificate

If you have configured a production ready Elasticsearch deployment, you would have configured SSL. Logstash needs to trust the Elasticsearch certificate to be able to communicate with it. To enable this trust, you should create a configuration map with the contents of the Elasticsearch certificate.

You would have already saved the Elasticsearch self-signed certificate. See Copying the Elasticsearch Certificate. If you have a production certificate you can use that instead.

Create the configuration map using the certificate by running the following command:

```
kubectl create configmap elk-cert --from-file=<WORKDIR>/ELK/elk.crt -n <OAMNS>
```

For example:

```
kubectl create configmap elk-cert --from-file=/workdir/ELK/elk.crt -n oamns
```

# Creating a Configuration Map for Logstash

Logstash looks for log files in the OAM installations and sends them to the centralized Elasticsearch. The configuration map is used to instruct Logstash where the log files reside and where to send them.

1. Create a file called `<WORKDIR>/OAM/logstash_cm.yaml` with the following contents:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: oam-logstash-configmap
  namespace: <OAMNS>
data:
  logstash.yaml: |
  #http.host: "0.0.0.0"
  logstash-config.conf: |
    input {
      file {
        path => "/u01/oracle/user_projects/domains/logs/accessdomain/
AdminServer*.log"
        tags => "Adminserver_log"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/logs/accessdomain/
oam_policy_mgr*.log"
        tags => "Policymanager_log"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/logs/accessdomain/
oam_server*.log"
        tags => "Oamserver_log"
        start_position => beginning
      }
```

```
      file {
        path => "/u01/oracle/user_projects/domains/accessdomain/servers/
AdminServer/logs/AdminServer-diagnostic.log"
        tags => "Adminserver_diagnostic"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/accessdomain/servers/**/
logs/oam_policy_mgr*-diagnostic.log"
        tags => "Policy_diagnostic"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/accessdomain/servers/**/
logs/oam_server*-diagnostic.log"
        tags => "Oamserver_diagnostic"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/accessdomain/servers/**/
logs/access*.log"
        tags => "Access_logs"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/accessdomain/servers/
AdminServer/logs/auditlogs/OAM/audit.log"
        tags => "Audit_logs"
        start_position => beginning
      }
    }
    filter {
      grok {
        match => [ "message", "<%{DATA:log_timestamp}> <%{WORD:log_level}>
<%{WORD:thread}> <%{HOSTNAME:hostname}> <%{HOSTNAME:servername}> <%
{DATA:timer}> <<%{DATA:kernel}>> <> <%{DATA:uuid}> <%{NUMBER:timestamp}> <%
{DATA:misc}> <%{DATA:log_number}> <%{DATA:log_message}>" ]
      }
    if "_grokparsefailure" in [tags] {
        mutate {
            remove_tag => [ "_grokparsefailure" ]
        }
    }
    }
    output {
      elasticsearch {
        hosts => ["<ELK_HOST>"]
        cacert => '/usr/share/logstash/config/certs/elk.crt'
        index => "oamlogs-000001"
        ssl => true
        ssl_certificate_verification => false
        user => "<ELK_USER>"
        password => "${ELASTICSEARCH_PASSWORD}"
      }
    }
```

2. Save the file.
3. Create the configuration map using the following command:

```
kubectl create -f <WORKDIR>/OAM/logstash_cm.yaml
```

For example:

```
kubectl create -f /workdir/OAM/logstash_cm.yaml
```

4. Validate that the configuration map has been created by using the following command:

```
kubectl get cm -n <OAMNS>
```

You should see `oam-logstash-configmap` in the list of configuration maps.

## Creating a Logstash Deployment

After you create the configuration map, you can create the Logstash deployment. This deployment resides in the OAM namespace.

1. Determine the mount point of the OAM persistent volume by using the following command:

```
kubectl describe domains <OAM_DOMAIN_NAME> -n <OAMNS> | grep "Mount Path"
```

For example:

```
kubectl describe domains accessdomain -n oamns | grep "Mount Path"
```

Make a note of this value. You will require it for the file that is created in the next step.

2. Create a file called `<WORKDIR>/OAM/logstash.yaml` with the following contents:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oam-logstash
  namespace: <OAMNS>
spec:
  selector:
    matchLabels:
      k8s-app: logstash
  template: # create pods using pod definition in this template
    metadata:
      labels:
        k8s-app: logstash
    spec:
      imagePullSecrets:
      - name: dockercred
      containers:
      - command:
        - logstash
        image: logstash:<ELK_VER>
        imagePullPolicy: IfNotPresent
```

```
    name: oam-logstash
    env:
    - name: ELASTICSEARCH_PASSWORD
      valueFrom:
        secretKeyRef:
          name: elasticsearch-pw-elastic
          key: password
    ports:
    - containerPort: 5044
      name: logstash
    volumeMounts:
    - mountPath: <MOUNT_PATH>
      name: weblogic-domain-storage-volume
    - name: shared-logs
      mountPath: /shared-logs
    - mountPath: /usr/share/logstash/pipeline/
      name: oam-logstash-pipeline
    - mountPath: /usr/share/logstash/config/certs
      name: elk-cert
  volumes:
  - configMap:
      defaultMode: 420
      items:
      - key: ca.crt
        path: elk.crt
      name: elk-cert
    name: elk-cert
  - configMap:
      defaultMode: 420
      items:
      - key: logstash-config.conf
        path: logstash-config.conf
      name: oam-logstash-configmap
    name: oam-logstash-pipeline
  - configMap:
      defaultMode: 420
      items:
      - key: logstash.yaml
        path: logstash.yaml
      name: oam-logstash-configmap
    name: config-volume
  - name: weblogic-domain-storage-volume
    persistentVolumeClaim:
      claimName: <OAM_DOMAIN_NAME>-domain-pvc
  - name: shared-logs
    emptyDir: {}
```

**Note:**

If you are using your own registry, include the registry name in the image tag. If you have created a `regcred` secret for your registry, replace the `imagePullSecrets` name with the secret name you created. For example: `regcred`.

3. Save the file.
4. Create the Logstash deployment by using the following command:

   ```
   kubectl create -f <WORKDIR>/OAM/logstash.yaml
   ```

   For example:

   ```
   kubectl create -f /workdir/OAM/logstash.yaml
   ```

5. You can now create a pod called `logstash` by using the following command:

   ```
   kubectl get pod -n oamns
   ```

   Your logs will now be available in the Kibana console.

# Backing Up the Configuration

As a best practice, Oracle recommends you to back up the configuration after you have successfully extended a domain or at another logical point. Back up only after you have verified that the installation is successful so far. This is a quick backup to enable immediate restoration in case of problems in later steps.

In a Kubernetes environment, it is sufficient to back up the persistent volume and the database.

The backup destination is the local disk. You can discard this backup when the enterprise deployment setup is complete. After the enterprise deployment setup is complete, you can initiate the regular deployment-specific Backup and Recovery process.

For information about backing up your configuration, see Performing Backups and Recoveries for an Enterprise Deployment.

19

# Configuring Oracle Identity Governance Using WDT

Install and configure an initial domain to use as the starting point for an enterprise deployment. Later, configure this domain.

A complete Oracle Identity and Access Management uses a split domain deployment, where there is a single domain for Oracle Access Management and a different domain for Oracle Identity Governance.

In version 4.1.2 of the WebLogic Kubernetes Operator, two different methods to create Oracle WebLogic domains are available. The traditional WLST method uses WLST scripts to create the domain which is the method employed in the Enterprise Deployment Guide for several releases.

Starting with this release, the Enterprise Deployment Guide will use the Weblogic Deployment Tools (WDT) to create the domains. The WDT uses templates to create domains which simplifies the installation procedure. For more information about WebLogic deployment tools, see WebLogic Deploy Tooling

This chapter includes the following topics:

- Synchronizing the System Clocks
  Before you deploy Oracle Identity Governance, verify that the system clocks on each host computer are synchronized. You can do this by running the `date` command simultaneously on all the hosts in each cluster.
- About the Initial Infrastructure Domain
  Before you create the initial Infrastructure domain, ensure that you review the key concepts.
- Prerequisites
  Before creating the Oracle Identity Governance (OIG) on the kubernetes infrastructure, you should have downloaded the Oracle Identity Governance container image and installed the Oracle WebLogic Operator.
- Creating a Namespace for Oracle Identity Governance
  Create a namespace to contain all the Oracle Identity Governance Kubernetes objects.
- Creating a Container Registry Secret
  If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.
- Creating a Kubernetes Secret for Docker Hub Images
  This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands.
- Creating the Database Schemas for Oracle Identity Governance
  Oracle Fusion Middleware components require schemas in a database, these schemas are handled by the WebLogic Deployment Tools at the time of deployment.

- Creating the Oracle Identity Governance Domain
  To create the Oracle Identity Governance domain, you should configure the WebLogic Kubernetes Operator for the domain namespace, create the Kubernetes secrets, and then create the Governance domain.
- Creating the Kubernetes Services
  By default, the OIG domain gets created with all the components (except the Administration Server) configured as ClusterIP services. This means that the Oracle Identity Governance components are visible only within the Kubernetes cluster.
- Tuning JMS Queues
  To ensure maximum throughput, tune the JMS queues.
- Installing the Connector Bundle
  After you create the domain, you need to copy any connectors you require, to the Kubernetes container. These connectors must be stored on the persistent volume.
- Performing the Post-Configuration Tasks for Oracle Identity Management Domain
  The post-configuration tasks for the OIG domain include creating the server overrides file and updating the data sources.
- Validating Identity Governance
  Perform a few tests to validate your installation.
- Analyzing the Bootstrap Report
  When you start the Oracle Identity Governance server, the bootstrap report is generated at `$DOMAIN_HOME/servers/oim_server1/logs/BootStrapReportPreStart_XXXX.html`.
- Configuring the Web Tier for the Domain
  If you have not already done so, configure the web server instances on the web tier so that the instances route requests for both public and internal URLs to the proper clusters in the extended domain.
- Integrating Oracle Identity Governance with Oracle SOA Suite
  You can integrate Oracle Identity Governance with Oracle SOA suite using the load balancer entry points to maintain high availability.
- Managing the Notification Service
  An event is an operation that occurs in Oracle Identity Manager, such as user creation, request initiation, or any custom event created by the user. These events are generated as part of the business operations or through the generation of errors. Event definition is the metadata that describes the event.
- Configuring the Messaging Drivers
  Each messaging driver needs to be configured. You have to configure this service if you want to enable OAM's forgotten password functionality.
- Increasing Database Connection Pool Size
  The default database connection pool size needs to be increased when Oracle Identity Governance is used in conjunction with a connector that allows interactions with an LDAP directory.
- Integrating Oracle Identity Governance with LDAP
  Before you integrate OIG with LDAP, you should configure the connector for LDAP and add the required object classes if any are missing.
- Integrating Oracle Identity Governance and Oracle Access Manager
  You have to complete several tasks to integrate Oracle Identity Governance and Oracle Access Manager. These tasks include creating the WLS authentication providers, deleting OIMSignatureAuthenticator and recreating OUDAuthenticator, adding the administration role to the new administration group, and so on.

- Running the Reconciliation Jobs
  Run the Oracle Identity Governance domain to import the LDAP user names into the Oracle Identity Governance database.
- Configuring OIM Workflow Notifications to be Sent by Email
  OIM uses the human workflow, which is integrated with the SOA workflow. The SOA server configures email to receive the notifications that are delivered to the user mailbox. The user can accept or reject the notifications.
- Adding the wsm-pm Role to the Administrators Group
  After you configure a new LDAP-based Authorization Provider and restart the Administration Server, add the enterprise deployment administration LDAP group (`WLSAdministrators`) as a member to the `policy.Updater` role in the `wsm-pm` application stripe.
- Adding the WebLogic Administration Group to SOA Administrators
  To manage SOA using the users in the LDAP administration group 'WLSAdministrators', you should add the name of the group to the SOA Administrators group.
- Adding the Oracle Access Manager Load Balancer Certificate to the Oracle Keystore Service
  The Oracle Identity Governance to Business Intelligence Reports link inside of the Self Service application requires that the SSL certificate used by the load balancer be added to the Oracle Keystore Service Trusted Certificates.
- Setting the Initial Server Count
  When you first created the domain, you specified that only one Managed Server has to be started. This value ensured that the OIG bootstrap process was completed successfully. After you complete the configuration, you can increase the initial server count to the actual number you require.
- Setting Challenge Questions
  If you have integrated OAM and OIM, then after the environment is ready, you need to set up the challenge questions for your system users.
- Integrating Oracle Identity Manager with Oracle Business Intelligence Publisher
  Oracle Identity Manager comes with a number of prebuilt reports that can be used to provide information about Oracle Identity and Access Management.
- Enabling Design Console Access
  You cannot access the Design Console that is installed as part of the installation because it is inside a container and requires access to an external X Window environment.
- Centralized Monitoring Using Grafana and Prometheus
- Centralized Log File Monitoring Using Elasticsearch and Kibana
- Backing Up the Configuration
  As a best practice, Oracle recommends you to back up the configuration after you have successfully extended a domain or at another logical point. Back up only after you have verified that the installation is successful so far. This is a quick backup to enable immediate restoration in case of problems in later steps.
- Running the OIM Bulkload Utility from a Container
  If you want to run the `oimbulkload` utility from a container, create a new container image based on the Oracle Database Instant Client which also has a JDK and the `oimbulkload` utility installed.

# Synchronizing the System Clocks

Before you deploy Oracle Identity Governance, verify that the system clocks on each host computer are synchronized. You can do this by running the `date` command simultaneously on all the hosts in each cluster.

Alternatively, there are third-party and open-source utilities you can use for this purpose.

# About the Initial Infrastructure Domain

Before you create the initial Infrastructure domain, ensure that you review the key concepts.

- About the Software Distribution
- Characteristics of the Domain
- Variables Used in this Chapter
- Kubernetes Services

## About the Software Distribution

You create the initial Infrastructure domain for an enterprise deployment by using the Oracle WebLogic Operator. The Oracle Identity Governance software is distrubted as a pre-built container image. See Identifying and Obtaining Software Distributions for an Enterprise Deployment. This distribution contains all of the necessary components to install and configure Oracle Identity Governance.

See Understanding Oracle Fusion Middleware Infrastructure in *Understanding Oracle Fusion Middleware*.

## Characteristics of the Domain

The following table lists some of the key characteristics of the domain that you are about to create. Reviewing these characteristics helps you to understand the purpose and context of the procedures that are used to configure the domain.

Many of these characteristics are described in more detail in Understanding a Typical Enterprise Deployment.

| Characteristic of the Domain | More Information |
|---|---|
| Places each WebLogic domain in a Kubernetes cluster. | See About the Kubernetes Deployment. |
| Each WebLogic Server is placed into a pod in the Kubernetes cluster. | See About the Kubernetes Deployment. |
| Places each Kubernetes domain object in a dedicated Kubernetes namespace. | See About the Kubernetes Deployment. |
| Uses Kubernetes NodePort Services to interact with the WebLogic Managed servers. | See Creating the Kubernetes Services. |
| Uses Kubernetes persistent volumes to hold the domain configuration. | See #unique_614. |
| Each Kubernetes pod is built from a pre-built Oracle container image. | See Identifying and Obtaining Software Distributions for an Enterprise Deployment. |

| Characteristic of the Domain | More Information |
|---|---|
| Uses a per domain Node Manager configuration. | See About the Node Manager Configuration in a Typical Enterprise Deployment. |
| Requires a separately installed LDAP-based authentication provider. | See Installing and Configuring Oracle Unified Directory. |
| Certificates are stored in the Oracle Keystore Service. | See Configuring Oracle OPSS Keystore Service. |
| JMS and TLOGS are stored in the database. | See Using a JDBC Store. |

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 19-1 The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| <*REGISTRY_ADDRESS*> | `iad.ocir.io/<mytenancy>` | The location of the registry. |
| <*REGISTRY_SECRET_NAME*> | `regcred` | The name of the Kubernetes secret containing the container registry credentials. Required only if you are pulling images directly from a container registry. See Creating a Container Registry Secret. |
| <*REG_USER*> | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The name of the user you use to log in to the registry. |
| <*REG_PASSWORD*> | <*password*> | The registry user password. |
| <*OIG_REPOSITORY*> | `oracle/oig`<br>`local/oracle/oig`<br>`container-registry.oracle.com/middleware/oig_cpu`<br>`<REGISTRY_ADDRESS>/oracle/oig` | The name of the OIG software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/oig`. If you use OLCNE, the value will be `local/oracle/oig`.<br>If you use the oracle container registry, the value will be `container-registry.oracle.com/middleware/oig_cpu`.<br>If you use a container registry, the value will be the name of the registry with the product name: <*REGISTRY_ADDRESS*>`/oracle/oig`. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OIG_VER>* | `12.2.1.4-jdk8-ol7-220420.0828` or latest. | The version of the image you want to use. This will be the version you have downloaded and staged either locally or in the container registry. |
| *<PVSERVER>* | `nfsserver.example.com` | The name or IP address of the NFS server<br>**Note**: This name should be resolvable inside the Kubernetes cluster. |
| *<OIGNS>* | `oigns` | The name of the OIG domain namespace. |
| *<WORKDIR>* | `workdir/OIG` | The location where you want to create the working directory for OAM. |
| *<K8WORKER>* | `k8worker1.example.com` | One of the Kubernetes hosts where the external WebLogic Administration Server Kubernetes service is resolvable. |
| *<OIG_SHARE>* | `/exports/IAMPVS/oigpv` | The NFS export for the persistence store. |
| *<OID_BULK_SHARE>* | `/exports/IAMPVS/oigbulkpv` | The persistent volume used for storing data to be loaded through the bulk load utility. |
| *<OIG_DB_SCAN>* | `dbscan.example.com` | The SCAN address of the RAC database. |
| *<OIG_DB_LISTENER>* | `1521` | The listener port number of the RAC database . |
| *<OIG_DB_SERVICE>* | `igdedg.example.com` | The name of the database service. If you are using a PDB, specify the name of the PDB service. |
| *<OIG_DB_SYS_PWD>* | `MySysPWD__001` | The SYS password for the database. |
| *<OIG_RCU_PREFIX>* | `IADEDG` | The prefix used when the database schemas are created. |
| *<OIG_SCHEMA_PWD>* | `MySchemaPWD__001` | The password you want to set for the product schemas being created. |
| *<OIG_WEBLOGIC_USER>* | `weblogic` | The name of the administration user for IAMGovernance domain. |
| *<OIG_WEBLOGIC_PWD>* | `mypassword` | The password for the WebLogic user. |
| *<OIG_DOMAIN_NAME>* | `governancedomain` | The name of the domain to be created. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OIG_DOMAIN_SECRET>* | `governancedomain-weblogic-credentials` | The name of the secret you want to create, for the namespace that is used. The name of the secret must be *<OIG_DOMAIN_NAME>*`-weblogic-credentials`. |
| *<OIG_RCU_SECRET>* | `governancedomain-rcu-credentials` | The name of the RCU secret. The name of the secret must be *<OIG_DOMAIN_NAME>*`-rcu-credentials`. See Creating the RCU Secret. |
| *<OIG_LBR_HOST>* | `prov.example.com` | The load balancer entry point for OIM. |
| *<OIG_LBR_PORT>* | `443` | The load balancer port for OIM. |
| *<OIM_SERVER_NAME>* | `oim_server1` | The name of the OIM server. |
| *<OIG_EMAIL_DOMAIN>* | `example.com` | The email domain. |
| *<OIG_ADMIN_PORT>* | `7101` | The internal port assigned to the OIG Administration Server. |
| *<WG_CONNECTIONS>* | `20` | The maximum number of connections supported by the WebGate agent. |
| *<LDAP_TYPE>* | `OUD` | It is the type of directory you are using: OUD or OID. |
| *<LDAP_OAMADMIN_USER>* | `oamadmin` | The name of the user you want to administer OAM. See Creating a Configuration File. |
| *<LDAP_OAMLDAP_USER>* | `oamLDAP` | The name of a user that OAM will use to connect to the directory for validating logins. |
| *<LDAP_XELSYSADM_USER>* | `xelsysadm` | The OIM administrator account. |
| *<LDAP_HOST>* | `idstore.example.com`<br>`edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local` | The load balancer name for the LDAP directory.<br>If your LDAP directory is inside the Kubernetes cluster, then you can use the Kubernetes service name, which has this format: *<K8_SERVICE_NAME>*`.<NAMESPACE>.svc.cluster.local`.<br>If you are wiring to an LDAP directory external to the Kubernetes cluster, then you should specify the name of the external load balancer. |
| *<LDAP_PORT>* | `1389` | It is the LDAP port of the load balancer. |
| *<LDAP_ADMIN_USER>* | `cn=oudadmin` for OUD. | The credential used to connect to the directory to perform administrative actions. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<LDAP_OIGLDAP_USER>* | `oimLDAP` | The name of the user that OIM uses to connect to LDAP for validating logins. |
| *<LDAP_SYSTEMIDS>* | `cn=systemids` | The name of a container where you want to store system ids. The user names placed in this container are not subject to OIM reconciliation or password aging. This container is reserved for users such as *<LDAP_OAMLDAP_USER>* and *<LDAP_OIGLDAP_USER>*. |
| *<LDAP_SEARCHBASE>* | `dc=example,dc=com` | The directory tree for your organization. This is where all the data is stored. |
| *<LDAP_USER_SEARCHBASE>* | `cn=Users,dc=example,dc=com` | The location in the directory where names of users are stored. |
| *<LDAP_GROUP_SEARCHBASE>* | `cn=Groups,dc=example,dc=com` | The location in the directory where user groups are stored. |
| *<LDAP_USER_PWD>* | *<password>* | Contains the password of the *<LDAP_OAMADMIN_USER>* account. |
| *<OAM_LOGIN_LBR_HOST>* | `login.example.com` | The listen address of the front end load balancer for the OAM cluster. |
| *<OAM_LOGIN_LBR_PORT>* | `443` | The port of the front end load balancer for the OAM cluster. |
| *<OAM_WEBLOGIC_USER>* | `weblogic` | The administration user of the OAM Administration Server. |
| *<OAM_WEBLOGIC_PWD>* | *<password>* | The optional password for *<OAM_WEBLOGIC_USER>*. |
| *<OAM_OAP_PORT>* | `5575` | The internal OAP port number.<br>If you are using the Kubernetes service, this value can be the internal port number. |
| *<OAP_SERVICE_PORT>* | `30540` | The Kubernetes service port which fronts the OAM OAP cluster nodes. If you are using the Kubernetes service, this value can be the internal port number. |
| *<GLOBAL_PASSPHRASE>* | - | Set this to the global passphrase. For obtaining the value, see Obtaining a Global Passphrase. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OIG_SERVER_COUNT>* | `5` | The number of Managed Servers required. Oracle highly recommends you to set this value to a number greater than the anticipated need in the system's lifetime. It creates a number of server definitions in the WebLogic domain and ensures that you have a simple mechanism to scale up the system when the demand increases. This value does not reflect the number of server instances you actually start with; it just enables you to start additional servers if your needs change. Adding additional server definitions post domain creation is a complex task and should be avoided, if possible. |
| *<OIG_INITIAL_SERVERS>* | `1` | The number of Managed Servers to start. Oracle recommends you to set this value to `1` for the duration of the configuration. |
| *<OIM_MAX_CPU>* | `1` | Maximum number of CPUs each `OIG_SERVER` pod is allowed to consume. The CPU is measured in CPU cores and the value of 1 is equal to 1 CPU core or 1 virtual core. |
| *<OIM_CPU>* | `500m` | The initial number of CPUs each `OIM_SERVER` pod is allowed to consume. It is measured in CPU cycles and the value of 1000m is equal to 1 CPU core or 1 virtual core. The CPU is measured in CPU cores and the value of 1 is equal to 1 CPU core or 1 virtual core. |
| *<SOA_MAX_CPU>* | `1` | Maximum number of CPUs each `SOA_SERVER` pod is allowed to consume. The CPU is measured in CPU cores and the value of 1 is equal to 1 CPU core or 1 virtual core. |
| *<SOA_CPU>* | `500m` | The initial number of CPUs each `SOA_SERVER` pod is allowed to consume. It is measured in CPU cycles and the value of 1000m is equal to 1 CPU core or 1 virtual core. The CPU is measured in CPU cores and the value of 1 is equal to 1 CPU core or 1 virtual core. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OIM_MAX_MEMORY>* | `8Gi` | The maximum amount of memory that the `OIM_SERVER` pods are allowed to consume. The memory is measured in standard units, where 1G is equal to 1Gi. |
| *<OIM_MEMORY>* | `4Gi` | The initial amount of memory that the `OIM_SERVER` pods are allowed to consume. The memory is measured in standard units, where 1G is equal to 1Gi. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<OIMSERVER_JAVA_PARAMS>* | `-Xms4096m -Xmx8192m` | The maximum (Xmx) and minimum heap size to allocate to each `OIM_SERVER`. Sizes are shown as a number of Mb. |

**Note:**

The maximum amount of heap size must be less than the maximum amount allowed to be used by the pod *<OIM_MAX_MEMORY>*.

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<SOASERVER_JAVA_PARAMS>* | `-Xms4096m -Xmx8192m` | The maximum (Xmx) and minimum heap size to allocate to each `SOA_SERVER`. Sizes are shown as a number of Mb.<br><br>**Note:**<br>The maximum amount of heap size must be less than the maximum amount allowed to be used by the Pod *<SOA_MAX_MEMORY>*. |
| *<OIG_OIM_T3_PORT_K8>* | `30142` | The Kubernetes service port you want to use. |
| *<OIG_ADMIN_K8>* | `30711` | The external Kubernetes service port for the external WebLogic Administration Server. |

**Table 19-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value(s) | Description |
|---|---|---|
| *<ELK_HOST>* | `https://elasticsearch-es-http.elkns.svc:9200` | The host and port of the centralized Elasticsearch deployment. This host can be inside the Kubernetes cluster or external to it. This host is used only when Elasticsearch is used. |
| *<ELK_VER>* | `8.11.0` | The version of Elasticsearch you want to use. |

## Kubernetes Services

If you are using NodePort Services, the following Kubernetes services are created as part of this deployment:

**Table 19-2 Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| `governancedomain-oim-http-NodePort` | `NodePort` | `30140` | `14000` |
| `governancedomain-soa-http-NodePort` | `NodePort` | `30801` | `8001` |
| `governancedomain-adminserver-external` | `NodePort` | `30711` | `7101` |

If you are using an Ingress-based deployment, the following Ingress services are created as part of your deployment:

**Table 19-3 Ingress Services**

| Service Name | Host Name |
|---|---|
| `oigadmin-ingress` | `igdadmin.example.com` |
| `oigruntime-ingress` | `prov.example.com` |
| `oiginternal-ingress` | `igdinternal.example.com` |

# Prerequisites

Before creating the Oracle Identity Governance (OIG) on the kubernetes infrastructure, you should have downloaded the Oracle Identity Governance container image and installed the Oracle WebLogic Operator.

- To download the Oracle Governance Manager Image and load it into the Docker image repository (this repository must be visible to each Kubernetes node), see Identifying and Obtaining Software Distributions for an Enterprise Deployment.

- To install the Oracle WebLogic Operator, see Installing and Configuring WebLogic Kubernetes Operator.
- Setting Up a Product Specific Work Directory

## Setting Up a Product Specific Work Directory

Before you begin the installation, you should have downloaded and staged the Oracle Identity Governance container image and the code repository. See Downloading Images from a Container Registry and Staging the Code Repository. You must also have deployed the Oracle WebLogic Operator as described in Installing the WebLogic Kubernetes Operator.

This section describes copying the downloaded sample deployment scripts to a temporary working directory on the configuration host for OIG.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubernetes cluster.

   ```
   mkdir -p /<WORKDIR>
   ```

   For example:

   ```
   mkdir -p /workdir/OIG
   ```

2. Change directory to this location:

   ```
   cd /workdir/OIG
   ```

3. Copy the sample scripts to the work directory.

   ```
   cp -R <WORKDIR>/fmw-kubernetes/OracleIdentityGovernance/kubernetes
   <WORKDIR>/samples
   ```

   For example:

   ```
   cp -R /workdir/OIG/fmw-kubernetes/OracleIdentityGovernance/kubernetes /
   workdir/OIG/samples
   ```

# Creating a Namespace for Oracle Identity Governance

Create a namespace to contain all the Oracle Identity Governance Kubernetes objects.

1. To create a namespace, use the following command:

   ```
   kubectl create namespace <OIGNS>
   ```

   For example:

   ```
   kubectl create namespace oigns
   ```

The output appears as follows:

```
namespace/oigns created
```

2. Tag the namespace so that the WebLogic Kubernetes Operator can manage it.

```
kubectl label namespaces oamns weblogic-operator=enabled
```

# Creating a Container Registry Secret

If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

> **Note:**
>
> If you are not using a container registry, you still need to create the registry secret. However, the user name and password need not contain meaningful data.

To create a container registry secret, use the following command:

```
kubectl create secret -n <OIGNS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n oigns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

# Creating a Kubernetes Secret for Docker Hub Images

This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands.

> **Note:**
>
> If you are pulling the images from your own container registry, then this step is not required.

You should have an account on `hub.docker.com`. If you want to stage the images in your own repository, you can do so and modify the `helm` override file as appropriate.

To create a Kubernetes secret for `hub.docker.com`, use the following command:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="<DH_USER>" --docker-
password="<DH_PWD>" --namespace=<OIGNS>
```

For example:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="username" --docker-
password="<mypassword>" --namespace=oigns
```

# Creating the Database Schemas for Oracle Identity Governance

Oracle Fusion Middleware components require schemas in a database, these schemas are handled by the WebLogic Deployment Tools at the time of deployment.

The tool creates the following schemas:

- Metadata Services (MDS)
- Audit Services (IAU)
- Audit Services Append (IAU_APPEND)
- Audit Services Viewer (IAU_VIEWER)
- Oracle Platform Security Services (OPSS)
- User Messaging Service (UMS)
- WebLogic Services (WLS)
- Common Infrastructure Services (STB)

For more information about RCU and how the schemas are created and stored in the database, see Preparing for Schema Creation in *Creating Schemas with the Repository Creation Utility*.

Complete the following steps to install the required schemas:

- Installing and Configuring a Certified Database
- Configuring OIM Schemas for Transactional Recovery

## Installing and Configuring a Certified Database

Make sure that you have installed and configured a certified database, and that the database is up and running.

See Preparing an Existing Database for an Enterprise Deployment.

## Configuring OIM Schemas for Transactional Recovery

After you have installed the Oracle Identity Governance successfully, use the procedure in this section to configure the schemas for transactional recovery.

This procedure sets the appropriate database privileges so that the Oracle WebLogic Server transaction manager can query the schemas for transaction state information and issue the appropriate commands, such as commit and rollback, during recovery of in-flight transactions after a WebLogic Server is unexpectedly unavailable.

These privileges should be granted to the owner of the OIM schema, which you defined when you created the schemas with the Repository Creation Utility.

To configure the OIM schemas for transactional recovery privileges:

1. Log on to SQL*Plus as a user with `sysdba` privileges. For example:

   ```
   sqlplus "/ as sysdba"
   ```

2. If you are using a PDB, change your session to the PDB hosting IDM. For example:

   ```
   alter session set container=igdpdb;
   ```

3. Enter the following commands:

   ```
   grant select on sys.dba_pending_transactions to <OIG_RCU_PREFIX>_oim;


   Grant succeeded.


   SQL> grant force any transaction to <OIG_RCU_PREFIX>_oim;


   Grant succeeded.
   ```

# Creating the Oracle Identity Governance Domain

To create the Oracle Identity Governance domain, you should configure the WebLogic Kubernetes Operator for the domain namespace, create the Kubernetes secrets, and then create the Governance domain.

- Creating the Kubernetes Secrets
- Creating the Governance Domain

## Creating the Kubernetes Secrets

Rather than passing the credentials directly into the domain creation process, you can use the Kubernetes secrets to store the credentials in the encrypted format. The WebLogic Operator reads these secrets instead of asking for credentials.

- Creating the Domain Secret
- Creating the RCU Secret

### Creating the Domain Secret

The domain secret contains information about the WebLogic Administration user who creates the domain.

1. Use the following command to create the domain secret:

   ```
   cd <WORKDIR>/samples/create-oim-domain/domain-home-on-pv/wdt-utils


   ./create-secret.sh -l "username=<OIG_WEBLOGIC_USER>" -l
   "password=<OIG_WEBLOGIC_PWD>" -n <OIGNS> -d <OIG_DOMAIN_NAME> -s
   <OIG_DOMAIN_SECRET>
   ```

For example:

```
cd /workdir/OIG/samples/create-oim-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "username=weblogic" -l "password=mypassword" -n
oigns -d governancedomain -s governancedomain-weblogic-credentials
```

The output appears as follows:

```
secret/governancedomain-weblogic-credentials created
secret/governancedomain-weblogic-credentials labeled
The secret governancedomain-weblogic-credentials has been successfully
created in the oigns namespace
```

2. Verify that the secret has been created, using the following command:

```
kubectl get secret governancedomain-weblogic-credentials -o yaml -n oigns
```

## Creating the RCU Secret

The RCU secret is used by the WebLogic Operator to determine how to connect to the database schemas that you have already created. See Creating the Database Schemas for Access Manager.

To create the RCU secret, perform the following steps:

1. Use the following command:

```
cd <WORKDIR>/samples/create-oim-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "rcu_prefix=<OIG_RCU_PREFIX>" -l
"rcu_schema_password=<OIG_SCHEMA_PWD>" -l "db_host=<DB_HOST>" -l
"db_port=<DB_PORT>" -l "db_service=<OIG_DB_SERVICE>" -l "dba_user=sys" -l
"dba_password=<OIG_SYS_PWD>" -n <OIGNS> -d <OIG_DOMAIN_NAME> -s
<OIG_RCU_SECRET>
```

For example:

```
cd /workdir/OIG/samples/create-oim-domain/domain-home-on-pv/wdt-utils
```

```
./create-secret.sh -l "rcu_prefix=IGDEDG" -l
"rcu_schema_password=MySchemaPWD__001" -l "db_host=DB-SCAN.example.com" -l
"db_port=1521" -l "db_service=oig_s.example.com" -l "dba_user=sys" -l
"dba_password=MySysPassword" -n oigns -d governancedomain -s
governancedomain-rcu-credentials
```

The output appears as follows:

```
secret/governancedomain-rcu-credentials created
secret/governancedomain-rcu-credentials labeled
```

```
The secret governancedomain-rcu-credentials has been successfully created
in the oigns namespace
```

2. Verify that the secret has been created, using the command:

```
kubectl get secret governancedomain-rcu-credentials -o yaml -n oigns
```

# Creating the Governance Domain

The procedure to create the Oracle Identity Governance domain includes creating the domain configuration file, creating the domain using the WebLogic Kubernetes Operator, setting the memory parameters, initializing the domain, and verifying the domain.

- Creating the Domain Configuration File
- Generate WDT Auxiliary Image
- Updating domain.yaml
- Creating the Domain Using the WebLogic Operator
- Verifying the Domain

## Creating the Domain Configuration File

A configuration file is used to tell the WebLogic Operator how to create the domain. This configuration file is named `create-domain-wdt.yaml` and is located in `<WORKDIR>/samples/create-oim-domain/domain-home-on-pv`.

1. Create a copy of the `create-domain-wdt.yaml` file.

   For example:

```
cp /workdir/OIG/samples/create-oim-domain/domain-home-on-pv/create-domain-
wdt.yaml /workdir/OIG/wdt-utils/generate_models_utils/create-domain-
wdt.yaml
```

2. Make the following changes to the `/workdir/OIG/create-domain-wdt.yaml` file:

```
domainUID: <OIG_DOMAIN_NAME>
domainPVMountPath: /u01/oracle/user_projects/
domainHome: /u01/oracle/user_projects/domains/<OIG_DOMAIN_NAME>
image: <OIG_REPOSITORY>:<OIG_VER>
imagePullSecretName: <REGISTRY_SECRET_NAME>
namespace: <OIGNS>
weblogicCredentialsSecretName: <OIG_DOMAIN_SECRET>
logHome: /u01/oracle/user_projects/domains/logs/<OIG_DOMAIN_NAME>
exposeAdminNodePort: true
configuredManagedServerCount: <OIG_SERVER_COUNT>
initialManagedServerReplicas: <OIG_INITIAL_SERVERS>
productionModeEnabled: true
exposeAdminT3Channel: false
adminPort: 7101
adminNodePort: <OIG_ADMIN_K8>
# Front End Host that will front end the oim and soa servers
frontEndHost: <OIG_LBR_HOST>
frontEndPort: <OIG_LBR_PORT>
```

```
datasourceType: agl
weblogicDomainStorageNFSServer: <PVSERVER>
weblogicDomainStorageType: NFS
weblogicDomainStoragePath: <OIG_SHARE>
edgInstall: true
oimServerJavaParams: <OIMSERVER_JAVA_PARAMS>

oimMaxCPU: <OIM_MAX_CPU>
oimCPU: <OIM_CPU>
oimMaxMemory: <OIM_MAX_MEMORY>
oimMemory: <OIM_MEMORY>

soaServerJavaParams: <SOASERVER_JAVA_PARAMS>
soaMaxCPU: <SOA_MAX_CPU>
soaCPU: <OAM_CPU>
soaMaxMemory: <SOA_MAX_MEMORY>
soaMemory: <SOA_MEMORY>
```

For example:

```
domainUID: governancedomain
domainPVMountPath: /u01/oracle/user_projects/
domainHome: /u01/oracle/user_projects/domains/governancedomain
image: latest
imagePullSecretName: regcred
namespace: oigns
weblogicCredentialsSecretName: oig-domain-credentials
persistentVolumeClaimName: governancedomain-domain-pvc
logHome: /u01/oracle/user_projects/domains/logs/governancedomain
rcuSchemaPrefix: IGDEDG
rcuDatabaseURL: dbscan.example.com:1521/igdedg.example.com
rcuCredentialsSecret: oig-rcu-credentials
exposeAdminNodePort: true
configuredManagedServerCount: 5
initialManagedServerReplicas: 1
productionModeEnabled: true
exposeAdminT3Channel: false
adminPort: 7101
adminNodePort: 30711
# Front End Host that will front end the oim and soa servers
frontEndHost: prov.example.com
frontEndPort: 443
datasourceType: agl
```

3. Save the configuration file.

## Generate WDT Auxiliary Image

When creating a domain using the WebLogic deployment tool, a dedicated image is created which describes the deployment. This is based on the domain creation file described in Creating the Domain Configuration File. This image is then stored in a local container registry.

The benefit of using an auxiliary image with the configuration is that it can be used repeatedly to create multiple environments with slightly different properties. For example, the same image file can be used to create a development, testing, and production environment where only the

database connection details vary. You need not create a new image each time you create a similar environment. This image must be stored in a registry where images are loaded, and you need have access to this registry.

The following sections describe how to generate the WDT model files, create an auxiliary image, and upload it to your repository.

### Generating WDT Model Files

Perform the following steps to generate the WDT model files from the Domain Configuration file:

1. Change the directory to WDT utils directory of the samples download.

   ```
   cd <WORKDIR>/samples/create-oim-domain/domain-home-on-pv/wdt-utils/
   generate_models_utils
   ```

   For Example:

   ```
   cd /workdir/OIG//samples/create-oim-domain/domain-home-on-pv/wdt-utils/
   generate_models_utils
   ```

2. Generate the model files using the `generate_wdt_models.sh` utility.

   ```
   ./generate_wdt_models.sh -i <WORKDIR>/create-domain-wdt.yaml -o <WORKDIR>
   ```

   Use `-i` to specify the location of the Domain configuration file you created in Creating the Domain Configuration File.

   Use `-o` to specify where the WDT Model files and templates should be created.

   For example:

   ```
   ./generate_wdt_models.sh -i /workdir/OIG/create-domain-wdt.yaml -o /
   workdir/OIG
   ```

   After running the utility, a directory is created called `weblogic-domains` which contain the generated files.

Sample Output with Input Parameters

```
export version="create-weblogic-sample-domain-inputs-v1"
export adminPort="7101"
export domainUID="governancedomain"
export configuredManagedServerCount="5"
export initialManagedServerReplicas="1"
export productionModeEnabled="true"
export t3ChannelPort="30012"
export datasourceType="agl"
export edgInstall="true"
export domainHome="/u01/oracle/user_projects/domains/governancedomain"
export image="iad.ocir.io/mytenancyoig:12.2.1.4-jdk8-ol8-apr24"
export imagePullSecretName="regcred"
export logHome="/u01/oracle/user_projects/domains/logs/governancedomain"
export exposeAdminT3Channel="false"
```

```
export adminNodePort="30711"
export exposeAdminNodePort="false"
export namespace="oigns"
javaOptions=-Dweblogic.StdoutDebugEnabled=false
export domainPVMountPath="/u01/oracle/user_projects"
export weblogicDomainStorageType="NFS"
export weblogicDomainStorageNFSServer="mynfsserver.example.com"
export weblogicDomainStoragePath="/exports/IAMPVS/oigpv"
export weblogicDomainStorageReclaimPolicy="Retain"
export weblogicDomainStorageSize="10Gi"
export frontEndHost="prov.example.com"
export frontEndPort="443"
export oimServerJavaParams="-Xms8192m -Xmx8192m "
export soaServerJavaParams="-Xms4096m -Xmx8192m"
export oimMaxCPU="1"
export oimCPU="500m"
export oimMaxMemory="8Gi"
export oimMemory="4Gi"
export soaMaxCPU="1"
export soaCPU="1000m"
export soaMaxMemory="10Gi"
export soaMemory="4Gi"

validateWlsDomainName called with governancedomain
WDT model file, property file and sample domain.yaml are genereted
successfully at /workdir/OIG/weblogic-domains/governancedomain
```

### Create Image Property File

After the model files are created, they need to be added to an image, and uploaded to your registry which begins with describing the target registry in a property file.

Perform the following steps to create an image property file:

1. Run the following command to ensure java is installed on your machine:

   ```
   which java
   ```

2. Copy the property file to your work directory.

   ```
   cp <WORKDIR>/samples/create-oim-domain/domain-home-on-pv/wdt-utils/build-
   domain-creation-image/properties/build-domain-creation-image.properties
   <WORKDIR>
   ```

   For Example:

   ```
   cp /workdir/OIG/samples/create-oim-domain/domain-home-on-pv/wdt-utils/
   build-domain-creation-image/properties/build-domain-creation-
   image.properties /workdir/OIG
   ```

3. Edit the file `build-domain-creation-image.properties` and add the following values:
   `JAVA_HOME` set this to the location of your JAVA installation found in *Step 1*.

For Example:

```
/usr
```

- `REPOSITORY` set this to the location in your registry where the image file is to reside.

  For example,

  ```
  iad.ocir.io/<mytenancy>/idm/oig_wdt
  ```

  .

  Where `oig_wdt` is the name of the image you wish to create.
- `IMAGE_TAG` used to assign a tag to the uploaded image, you can use anything here. In case of this example, we can use <*OIG_DOMAIN_NAME*>.
- `IMAGE_PUSH_REQUIRES_AUTH` must be set to true if you do not allow non-authenticated uploads to your registry.
- `REG_USER` must be set to the user in your registry where you wish to upload the image. This user must have upload privileges.
- `WDT_MODEL_FILE` must be set to the file `oam.yaml` which was generated in the step above. For Example, `<WORKDIR>/weblogic-domains/<OIG_DOMAIN_NAME>/oim.yaml`.
- `WDT_VARIABLE_FILE` must be set to the file `oam.properties` which was generated in the step above. For example, `<WORKDIR>/weblogic-domains/<OIG_DOMAIN_NAME/>oim.properties`.
- `REG_PWD` must be set to the password of the above user and placed in a separate file in `buiidpwd` in the `<WORKDIR>` as shown below:

  ```
  REG_PASSWORD="<mypwd>"
  ```

  Sample `build-domain-creation-image.properties`

  ```
  # Copyright (c) 2024, Oracle and/or its affiliates.
  # Licensed under the Universal Permissive License v 1.0 as shown at
  https://oss.oracle.com/licenses/upl.
  # Input Property file for build-domain-creation-image.sh script

  #
  # set the JAVA_HOME environment variable to match the location of your
  Java installation. Java 8 or newer is required
  #
  JAVA_HOME=/usr

  #
  # Image Details
  #
  #Set the IMAGE_TAG, default oam-aux-v1 if not set.
  IMAGE_TAG=governancedomain
  # Set the BASE_IMAGE, default ghcr.io/oracle/oraclelinux:8-slim if not
  set.
  BASE_IMAGE=ghcr.io/oracle/oraclelinux:8-slim
  #
  ```

```
# Container Registry
#
#Image will be created with REPOSITORY:IMAGE_TAG
REPOSITORY=iad.ocir.io/<mytenancy>idm/oig_wdt
# Container registry username
REG_USER=<mytenancy>/oracleidentitycloudservice/my.user@example.com
#Set it to false if authentication is not required for pushing the
image to registry, for example docker login already done in the host
before invoking the script.
IMAGE_PUSH_REQUIRES_AUTH=true
#
# WDT and WIT Variables
#
#Full path to wdt model files
WDT_MODEL_FILE=/workdir/OIG/weblogic-domains/governancedomain/oim.yaml
#Full path to wdt variable files
WDT_VARIABLE_FILE=/workdir/OIG/weblogic-domains/governancedomain/
oim.properties
#Full path to wdt archive files
WDT_ARCHIVE_FILE=""
#If not set, Latest version will be used.
WDT_VERSION="3.5.3"
#If not set, latest will be used during every fresh run
WIT_VERSION="1.12.1"

#In Most cases, no need to use these parameters. Please refer https://
oracle.github.io/weblogic-image-tool/userguide/tools/create-aux-image/
for details about them.
TARGET=""
CHOWN=""
```

### Uploading WDT Auxiliary Image

Use the utility `build-domain-creation-image.sh` to create and upload the Auxiliary image:

For Example:

```
cd <WORKDIR>/samples/create-oim-domain/domain-home-on-pv/wdt-utils/build-
domain-creation-image
```

```
./build-domain-creation-image.sh -i <WORKDIR>/build-domain-creation-
image.properties -p <WORKDIR>/.buildpwd
```

For Example:

```
cd /workdir/OIG/samples/create-oim-domain/domain-home-on-pv/wdt-utils/build-
domain-creation-image
```

```
./build-domain-creation-image.sh -i /workdir/OIG/build-domain-creation-
image.properties -p /workdir/OIG/.buildpwd
```

Extract from Sample Output

```
Getting image source signatures
Copying blob
sha256:d56869e2b34f592d78b05cce249e0130a52fb73209bbb394bb329b1fed54a652
Copying blob
sha256:ba40a64765a65573fb1b9cfc9e175bd53c420c7ce8ec1424fda55835efbb7055
Copying blob
sha256:0b458bb8ab4506598a0a925a3110c079ffbf77f85e3b97713e4592a2cb47a97f
Copying blob
sha256:9aa2b64b3e6fefe00a04b52511ffda5b5ab3018538a1c7b11c4af4300e9220e0
Copying blob
sha256:8b4d3bacf0d79476c744efb9d80fc05c5e1298b2ce8c5ed88edc9a4a01198ba9
Copying blob
sha256:3ae779ed2d0c15ccbf8b31ae75afcbb857bb731618e9bde89108e8079ed4e9fe
Copying blob
sha256:306ac5e1f9589c0be83dfa010d3bc53097c7acb7ef0fd51d054d1e6545c35c84
Copying blob
sha256:5002f0067a2f325a4e67415b2e6889568719d0020b1df7b07af6be945c332210
Copying blob
sha256:97acec59a7dd3180feaa8c2257fa8ed8027e5763d6aedb1c8a4df1a740e1ecb7
Copying blob
sha256:5778e746ec78114f3569e4729dc11cc746dc6af9b5ebcf577ae9fe94867b495d
Copying blob
sha256:6e841a878721c1c37fc0885152697674be097dc5d81004188a2e1dd647850e3e
Copying config
sha256:ff14e6503d9efa8858512e8e7401e9c1b7d532acf11ffc44fb866ed5f4de00f1
Writing manifest to image destination
Storing signatures
Pushed image iad.ocir.io/mytenancy/oig_wdt:governancedomain to image
repository
```

## Updating domain.yaml

While generating the WDT model files, a file called `domain.yaml` was created in the directory `<WORKDIR>/weblogic-domains/<OIG_DOMAIN_NAME>`. This file is used to create the WebLogic domain. Before using this file the auxiliary image you create must be added to the file by editing

```
domain.yaml
```

Locate the variable in the file `%DOMAIN_CREATION_IMAGE%` and replace it with the name of your image as `<REPOSITORY>:<IMAGE_TAG>` obtained from the file `build-domain-creation-image.properties`.

For example,

```
iad.ocir.io/mytenancy/idm/oig_wdt:accessdomain
```

> **Note:**
>
> If the registry where your image is located is different to the registry where your OIG image is stored then create a new secret with the credentials for the Auxiliary image registry using a different name to the main registry.
>
> For example:
>
> ```
> kubectl create secret -n oigns docker-registry regcred2 --docker-
> server=iad.ocir.io/mytenancy2 --docker-username=mytenancy/
> oracleidentitycloudservice/myemail@email.com --docker-
> password=<password>
> ```

Update the file `domain.yaml` and replace the lines with the name of your new secret.

Add additional secret name if you are using a different registry for domain creation image.

Identify which secret contains the credentials for pulling an image.

```
imagePullSecrets:

- name: regcred2
```

## Creating the Domain Using the WebLogic Operator

Create the domain using the following command:

```
cd <WORKDIR>/weblogic-domains/<OIG_DOMAIN_NAME>

kubectl create -f <WORKDIR>/weblogic-domains/<OIG_DOMAIN_NAME>/domain.yaml
```

For example:

```
cd /workdir/OIG/weblogic-domains/governancedomain

kubectl create -f /workdir/OIG/weblogic-domains/governancedomain/domain.yaml
```

Use the following to monitor the domain creation:

```
kubectl logs -n <OIGNS> <OIG_DOMAIN_DOMAIN>-introspector

kubectl describe domain -n <OIGNS> <OIG_DOMAIN_NAME>
```

For Example:

```
kubectl logs -n oigns governancedomain-introspector
```

```
kubectl describe domain -n oigns governancedomain
```

For more information, see the WebLogic operator logs.

For Example:

```
kubectl logs -n opns weblogic-operator-688f5dcdc4-qxnnz | grep
<OIG_DOMAIN_NAME>
```

After the domain is created, the OAM Kubernetes pods is started automatically and can be viewed using the command:

```
kubectl get pods -n <OIGNS>
```

## Verifying the Domain

To verify the creation of the domain, perform the following steps:

1. To confirm that the domain is created, use the following command:

   ```
   kubectl describe domain <domain_uid> -n <namespace>
   ```

   For example:

   ```
   kubectl describe domain governancedomain -n oigns
   ```

2. Verify that the domain pods and services have been created and started, using the following command:

   ```
   kubectl get all,domains -n oigns
   ```

   The output appears as follows:

   ```
   NAME                                                              READY
   STATUS      RESTARTS   AGE   IP              NODE       NOMINATED NODE
   READINESS GATES
   pod/governancedomain-adminserver                                  1/1
   Running     0          17h   192.168.14.205   slc09byd   <none>
   <none>
   pod/governancedomain-create-fmw-infra-sample-domain-job-45mwk   0/1
   Completed   0          23h   192.168.14.203   slc09byd   <none>
   <none>
   pod/governancedomain-soa-server1                                  1/1
   Running     0          16h   192.168.14.206   slc09byd   <none>
   <none>
   pod/helper                                                        1/1
   Running     0          45h   192.168.14.202   slc09byd   <none>
   <none>
   ```

```
NAME                                                  TYPE        CLUSTER-
IP       EXTERNAL-IP   PORT(S)               AGE    SELECTOR
service/governancedomain-adminserver            ClusterIP
None             <none>        7101/TCP              17h
weblogic.createdByOperator=true,weblogic.domainUID=governancedomain,weblogi
c.serverName=AdminServer
service/governancedomain-adminserver-external   NodePort
10.96.33.206     <none>        7101:30711/TCP        17h
weblogic.createdByOperator=true,weblogic.domainUID=governancedomain,weblogi
c.serverName=AdminServer
service/governancedomain-cluster-oim-cluster    ClusterIP
10.103.195.154   <none>        14002/TCP,14000/TCP   16h
weblogic.clusterName=oim_cluster,weblogic.createdByOperator=true,weblogic.d
omainUID=governancedomain
service/governancedomain-cluster-soa-cluster    ClusterIP
10.106.89.77     <none>        8001/TCP              16h
weblogic.clusterName=soa_cluster,weblogic.createdByOperator=true,weblogic.d
omainUID=governancedomain
service/governancedomain-soa-server1            ClusterIP
None             <none>        8001/TCP              16h
weblogic.createdByOperator=true,weblogic.domainUID=governancedomain,weblogi
c.serverName=soa_server1

NAME
COMPLETIONS   DURATION   AGE   CONTAINERS
IMAGES                  SELECTOR
job.batch/governancedomain-create-fmw-infra-sample-domain-job
1/1           9m9s       23h   create-fmw-infra-sample-domain-job   oracle/
oig:12.2.1.4.0   controller-uid=a724d3ea-cbf0-43e1-9743-61a4f753c8b7
```

**Note:**

It will take several minutes before all the services listed above appear. A pod with a STATUS of 0/1 indicates that the pod has already started but the SOA server associated with the pod is just starting. While the pods are starting, you can check the startup status in the pod logs by using the following commands:

```
kubectl logs governancedomain-adminserver -n oigns
```

```
kubectl logs governancedomain-soa-server1 -n oigns
```

Only after the Administration Server and SOA servers have started successfully, you start the remaining servers using the command:

```
kubectl patch cluster -n <OIGNS> <OIG_DOMAIN_NAME>-oim-cluster --type=merge -
p '{"spec":{"replicas":1}}"'
```

For example:

```
kubectl patch cluster -n signs governancedomain-oim-cluster --type=merge -p
'{"spec":{"replicas":1}}"'
```

Verify that the domain has been initialized, using the following command:

```
kubectl get all,domains -n oigns
```

The output appears as follows:

```
NAME                                                              READY
STATUS      RESTARTS   AGE
pod/governancedomain-adminserver                                  1/1
Running     0          13h
pod/governancedomain-create-fmw-infra-sample-domain-job-ks2rj     0/1
Completed   0          14h
pod/governancedomain-oim-server1                                  1/1
Running     0          12h
pod/governancedomain-oim-server2                                  1/1
Running     0          12h
pod/governancedomain-soa-server1                                  1/1
Running     0          12h
pod/governancedomain-soa-server2                                  1/1
Running     0          12h
pod/helper                                                        1/1
Running     0          14h

NAME                                             TYPE        CLUSTER-IP
EXTERNAL-IP   PORT(S)               AGE
service/governancedomain-adminserver             ClusterIP   None
<none>        7101/TCP              13h
service/governancedomain-cluster-oim-cluster     ClusterIP   10.98.52.6
<none>        14002/TCP,14000/TCP   14h
service/governancedomain-cluster-soa-cluster     ClusterIP   10.104.237.225
<none>        8001/TCP              14h
service/governancedomain-oim-server1             ClusterIP   None
<none>        14002/TCP,14000/TCP   12h
service/governancedomain-oim-server2             ClusterIP   None
<none>        14002/TCP,14000/TCP   12h
service/governancedomain-oim-server3             ClusterIP   10.107.107.116
<none>        14002/TCP,14000/TCP   12h
service/governancedomain-oim-server4             ClusterIP   10.98.134.71
<none>        14002/TCP,14000/TCP   12h
service/governancedomain-oim-server5             ClusterIP   10.103.64.15
<none>        14002/TCP,14000/TCP   12h
service/governancedomain-soa-server1             ClusterIP   None
<none>        8001/TCP              12h
service/governancedomain-soa-server2             ClusterIP   None
<none>        8001/TCP              12h
service/governancedomain-soa-server3             ClusterIP   10.111.204.234
<none>        8001/TCP              12h
service/governancedomain-soa-server4             ClusterIP   10.107.90.229
<none>        8001/TCP              12h
service/governancedomain-soa-server5             ClusterIP   10.97.72.84
```

```
<none>        8001/TCP             12h

NAME
COMPLETIONS   DURATION   AGE
job.batch/governancedomain-create-fmw-infra-sample-domain-job
1/1           5m41s      14h

NAME                                          AGE
domain.weblogic.oracle/governancedomain       14h

NAME                                                    AGE
cluster.weblogic.oracle/governancedomain-oim-cluster    14h
cluster.weblogic.oracle/governancedomain-soa-cluster    14h
```

> **Note:**
>
> It will take several minutes before all the services listed above show up. When a pod has a STATUS of 0/1, the pod is started but the OIM server associated with it is just starting. While the pods are starting, you can check the startup status in the pod logs, by running the following commands:
>
> ```
> kubectl logs governancedomain-oim-server1 -n oigns
> ```

# Creating the Kubernetes Services

By default, the OIG domain gets created with all the components (except the Administration Server) configured as ClusterIP services. This means that the Oracle Identity Governance components are visible only within the Kubernetes cluster.

In an enterprise deployment, all interactions with the WebLogic components take place through the Oracle HTTP Server which sits outside of the Kubernetes cluster. You expose the WebLogic components to the outside world by creating Kubernetes additional services. You can use either NodePort Services or an Ingress controller.

- Creating NodePort Services
- Creating a SOA NodePort Service
- Validating the Services
- Creating the Ingress Services

## Creating NodePort Services

To create an OIM NodePort Service:

1. Create a text file called `<WORKDIR>/oim_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OIG_DOMAIN_NAME>-oim-nodeport
  namespace: <OIGNS>
```

```
spec:
  type: NodePort
  selector:
    weblogic.clusterName: oim_cluster
  ports:
    - targetPort: 14000
      port: 14000
      nodePort: <OIG_OIM_PORT_K8>
      protocol: TCP
  sessionAffinity: ClientIP
```

2. Create the service using the following command:

```
kubectl create -f /workdir/OIG/oim_nodeport.yaml
```

The output appears as follows:

```
service/governancedomain-oim-nodeport created
```

## Creating a SOA NodePort Service

To create a SOA NodePort Service:

1. Create a text file called `<WORKDIR>/soa_nodeport.yaml` with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OIG_DOMAIN_NAME>-soa-nodeport
  namespace: <OIG>
spec:
  type: NodePort
  selector:
    weblogic.clusterName: soa_cluster
  ports:
    - targetPort: 8001
      port: 8001
      nodePort: <OIG_SOA_PORT_K8>
      protocol: TCP
```

2. Create the service using the following command:

```
kubectl create -f /workdir/OIG/soa_nodeport.yaml
```

The output appears as follows:

```
service/governancedomain-soa-nodeport created
```

## Validating the Services

To validate that the services have been created correctly, use the following command:

```
kubectl -n oigns get all -o wide
```

# Creating the Ingress Services

To create Ingress services, you must first create an Ingress controller. For more information about the installation procedure, see Installing and Configuring Ingress Controller.

The Ingress service is created inside the product namespace. It tells the Ingress controller how to direct requests inside the namespace.

> **Note:**
>
> The example below creates three Ingress services, one for each of the OIG virtual hosts.
>
> - `igdadmin.example.com`
> - `prov.example.com`
> - `igdinternal.example.com`

To create an Ingress service:

1. Copy the `values.yaml` file from the `<WORKDIR>/samples/charts/ingress-per-domain` to your working directory and rename the file to `override_ingress.yaml`.
2. Edit the file `<WORKDIR>/override_ingress.yaml` and set the values as follows:

```
set domainUID to <OIG_DOMAIN_NAME>
set adminServerPort to <OIG_ADMIN_PORT>
set hostName.enabled to true
set admin to <OIG_ADMIN_LBR_HOST>
set runtime to <OIG_LBR_HOST>
set internal to <OIG_LBR_INT_HOST>
```

For example:

```
# Load balancer type. Supported values are: NGINX
type: NGINX

# SSL configuration Type. Supported Values are : NONSSL,SSL
sslType: NONSSL

# domainType. Supported values are: oim
domainType: oim

#WLS domain as backend to the load balancer
wlsDomain:
  domainUID: governancedomain
  adminServerName: AdminServer
  adminServerPort: 7101
  adminServerSSLPort:
  soaClusterName: soa_cluster
  soaManagedServerPort: 8001
  soaManagedServerSSLPort:
  oimClusterName: oim_cluster
```

```
  oimManagedServerPort: 14000
  oimManagedServerSSLPort:

# Host specific values
hostName:
  enabled: true
  admin: igdadmin.example.com
  runtime: prov.example.com
  internal: internal.example.com

# Ngnix specific values
nginx:
  nginxTimeOut: 180
```

3. Create the Ingress services by running the following commands:

```
cd <WORKDIR>/samples
```

```
helm install oig-nginx charts/ingress-per-domain --namespace <OIGNS> --
values <WORKDIR>/override_ingress.yaml
```

4. Validate that the Ingress service has been created correctly by using the following command:

```
kubectl get ingress -n oigns
```

# Tuning JMS Queues

To ensure maximum throughput, tune the JMS queues.

To tune the OIM JMS queue:

1. Log in to the WebLogic Server Console at: `http://k8worker1.example.com:30711/console`.
2. Click **Lock & Edit**.
3. In **Domain Structure**, expand **Services and Messaging**, and then click **JMS Servers**.
4. Click on **OIMJMSServer**. Change the values for the following:
   - Message Buffer Size to `1073741824 (1GB)`.
   - Under **Thresholds and Quota**, change **Messages Maximum** to `1000000`.
5. Click **Save**.
6. Click **Activate Changes**.

# Installing the Connector Bundle

After you create the domain, you need to copy any connectors you require, to the Kubernetes container. These connectors must be stored on the persistent volume.

To install the connector bundle, this example uses the Oracle Internet Directory connector bundle, which is used to integrate Oracle Identity Governance with Oracle Unified Directory.

1. If not already done, download the connector bundle to the `<WORKDIR>`.

   https://www.oracle.com/middleware/technologies/identity-management/oim-connectors-downloads.html

2. Create a sub-directory and unzip the connector into it. For example:

```
mkdir /workdir/OIG/connectors
cd /workdir/OIG/connectors
unzip /workdir/OIG/oid-12.2.1.3.zip
```

3. Copy the connector to the Kubernetes container locating it in persistent storage.

```
kubectl exec -ti oimserver-1 -n <OIGNS> -- mkdir -p /u01/oracle/
user_projects/domains/ConnectorDefaultDirectory
```

```
kubectl cp <CONNECTOR_DIR>/OID-12.2.1.3.0 <OIGNS>/<domain-uid>-
adminserver:/u01/oracle/user_projects/domains/ConnectorDefaultDirectory
```

   For example:

```
kubectl exec -ti governancedomain-oim-server1 -n oigns -- mkdir -p /u01/
oracle/user_projects/domains/ConnectorDefaultDirectory
```

```
kubectl cp /workdir/OIG/connectors/OID-12.2.1.3.0 oigns/governancedomain-
adminserver:/u01/oracle/user_projects/domains/ConnectorDefaultDirectory
```

# Performing the Post-Configuration Tasks for Oracle Identity Management Domain

The post-configuration tasks for the OIG domain include creating the server overrides file and updating the data sources.

- Limiting Pods to Specific Worker Nodes
- Creating the Server Overrides File

## Limiting Pods to Specific Worker Nodes

If you want to ensure that the OIG servers start only on a specific set of worker servers, complete the following steps:

- Labeling the Kubernetes Worker Nodes
- Restricting Processes to Labels

### Labeling the Kubernetes Worker Nodes

Label the worker nodes you want to include in scheduling. This can be as granular as you need. For example, if you want to schedule the OIG processes to run on a set of nodes, then label that set with a label such as `oigservers`. If you want to dictate that the Administration

Server runs on a specific set of worker nodes and the `oim_server` on a different set, then create two labels, `oigadmin` and `oimservers`.

Add a label to a Kubernetes node using the following command:

```
kubectl label node worker1 name=oimservers
```

## Restricting Processes to Labels

To ensure that the OIM pods run on only worker nodes with the appropriate label, edit the `domain.yaml` file located in the following path:

```
<WORKDIR>/samples/create-oim-domain/domain-home-on-pv/output/weblogic-domains/
<OIG_DOMAIN_NAME>/
```

For example:

```
/workdir/OIG/samples/create-oim-domain/domain-home-on-pv/output/weblogic-
domains/accessdomain/
```

Alter the Managed Servers section for all the Managed Servers configured in the cluster and ensure that only the labeled worked nodes are used for scheduling.

For `oim_server1` and `oim_server2`, the entries will look similar to:

```
managedServers:
- serverName: oim_server1
  serverPod:
    nodeSelector:
      name: oimservers
- serverName: oim_server2
  serverPod:
    nodeSelector:
      names: oimservers
```

# Creating the Server Overrides File

The `serverOverrides` file is used to set specific Java values when the containers start. The parameters are appended to the configuration in the `setDomainEnv.sh` file but unlike the `setDomainEnv.sh`file, the `serverOverrides` file is not overwritten during the upgrade.

- Disabling the Derby Database
- Enabling the Managed Servers to Use IPv4 Networking
- Setting the Memory Parameters in IAMGovernanceDomain
- Copying Server Overrides to the Kubernetes Containers

## Disabling the Derby Database

Disable the embedded Derby database, which is a file-based database, packaged with Oracle WebLogic Server. The Derby database is used primarily for development environments. Therefore, you must disable it when you are configuring a production-ready enterprise

deployment environment. Otherwise, the Derby database process starts automatically when you start the Managed servers.

To disable the Derby database:

1. Create a file called `/workdir/OIG/setUserOverrides.sh` with the following content:

```
DERBY_FLAG=false
```

2. Save and close the file.

## Enabling the Managed Servers to Use IPv4 Networking

If the Managed Server is configured to use IPv6 networking, then you may encounter issues when you start the Managed Server. Therefore, you must enable the Managed Servers to use IPv4 networking.

1. Edit the `setUserOverrides.sh` file and add the following line:

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Djava.net.preferIPv4Stack=true"
```

> **Note:**
>
> If the file does not exist, create it.

2. Save and close the file.

## Setting the Memory Parameters in IAMGovernanceDomain

The initial startup parameter in the IAMGovernanceDomain, which defines the memory usage, is insufficient. You must increase the value of this parameter.

1. Change the following memory allocation in the `setUserOverrides.sh` file, by updating the Java maximum memory allocation pool (Xmx) to 8192m and initial memory allocation pool (Xms) to 4096m. For example, add the following line:

```
MEM_ARGS="-Xms4096m -Xmx8192m"
```

2. Save and close the file.

## Copying Server Overrides to the Kubernetes Containers

In a Kubernetes environment, there is no editior inside the container. To work around this issue, create the file on the master node and copy it to the Kubernetes container using the following commands:

```
chmod 755 /workdir/OIG/setUserOverrides.sh
```

```
kubectl cp /workdir/OIG/setUserOverrides.sh oigns/governancedomain-
adminserver:/u01/oracle/user_projects/domains/governancedomain/bin/
setUserOverrides.sh
```

Where `oigns` is the OIG namespace and `governancedomain` is the *`DOMAIN_NAME`*`/UID`.

# Validating Identity Governance

Perform a few tests to validate your installation.

- Validating OIM by Logging in to the Identity Console
- Validating the SOA Application
- Validating the Fusion Middleware Control Application

## Validating OIM by Logging in to the Identity Console

You can validate the Oracle Identity Manager Server instance by bringing up the Oracle Identity Manager Console in a web browser.

1. Launch the Oracle Identity Manager Console in a web browser at:

   ```
   http://k8worker1.example.com:30140/identity/

   http://k8worker1.example.com:30140/sysadmin/
   ```

2. Log in using the **xelsysadm** user name and password.

## Validating the SOA Application

Validate SOA by logging into soa-infra:

```
http://k8worker1.example.com:30801/soa-infra
```

Log in using the `weblogic` user name.

## Validating the Fusion Middleware Control Application

You can access the Fusion Middleware Control application after you execute the bootstrap process and validate it.

> **Note:**
>
> Provide the challenge questions if you are prompted to enter them.

To navigate to the Fusion Middleware Control application, enter the following URL, and log in with the Oracle WebLogic Server administrator credentials:

```
http://k8worker1.example.com:30711/em
```

# Analyzing the Bootstrap Report

When you start the Oracle Identity Governance server, the bootstrap report is generated at *$DOMAIN_HOME*`/servers/`*oim_server1*`/logs/BootStrapReportPreStart_XXXX.html`.

The bootstrap report `BootStrapReportPreStart_XXXX.html` is an HTML file that contains information about the topology that you have deployed, the system level details, the connection

details like the URLs to be used, the connectivity check, and the task execution details. You can use this report to check if the system is up, and also to troubleshoot the issues, post-configuration.
Every time you start the Oracle Identity Governance server, the bootstrap report is updated.

### Sections in the Bootstrap Report

- **Topology Details**

  This section contains information about your deployment. It shows whether you have configured a cluster setup, SSL enabled, or upgraded an Oracle Identity Manager environment from 11*g* to 12*c*.

- **System Level Details**

  This section contains information about the JDK version, Database version, JAVA_HOME, DOMAIN_HOME, OIM_HOME, and MIDDLEWARE_HOME.

- **Connection Details**

  This section contains information about the connect details like the Administration URL, OIM Front End URL, SOA URL, and RMI URL.

  This also shows whether the Administration Server, Database, and SOA server is up or not.

- **Execution Details**

  This section lists the various tasks and their statuses.

To obtain the bootstrap report, you can do one of the following:

- Connect to the Oracle Identity Governance Administration Server by using the command:

  ```
  kubectl exec -n <OIGNS> -ti <OIG_DOMAIN_NAME>-adminserver -- /bin/bash


  cat /u01/oracle/user_projects/domains/<OIG_DOMAIN_NAME>/servers/
  oim_server1/logs/BootStrapReportPreStart_XXXX.html
  ```

  For example:

  ```
  kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash


  cat /u01/oracle/user_projects/domains/governancedomain/servers/oim_server1/
  logs/BootStrapReportPreStart_XXXX.html
  ```

- If you have mounted the IAMPVS on your configuration host, you can simply point a browser at:

  ```
  /nfs_volumes/oigpv/domains/governancedomain/servers/oim_server1/logs/
  BootStrapReportPreStart_XXXX.html
  ```

# Configuring the Web Tier for the Domain

If you have not already done so, configure the web server instances on the web tier so that the instances route requests for both public and internal URLs to the proper clusters in the extended domain.

For more information about configuring Oracle HTTP Server, see Installing and Configuring Oracle HTTP Server.

For additional steps in preparation for possible scale-out scenarios, see Updating Cross Component Wiring Information.

# Integrating Oracle Identity Governance with Oracle SOA Suite

You can integrate Oracle Identity Governance with Oracle SOA suite using the load balancer entry points to maintain high availability.

- Updating the OIM Integration URLs

## Updating the OIM Integration URLs

This section describes how to update the SOA integration URLs to use the load balanced URLs. If you want to integrate Oracle Identity Governance with Oracle SOA suite, use the Enterprise Manager Console.

You need to perform certain tasks in order to configure the newly created domain with the Oracle Identity Governance. These tasks are post-domain creation tasks.

To integrate Oracle Identity Governance with Oracle SOA Suite, do the following:

1. Log in to Oracle Fusion Middleware Control using the following URL:

   ```
   http://igdadmin.example.com/em
   ```

   or

   ```
   http://k8worker1.example.com:30711/em
   ```

   The Administration Server host and port number were in the URL on the End of Configuration screen (Writing Down Your Domain Home and Administration Server URL). The default Administration Server port number is `7101`.

   The login credentials were provided on the Administrator Account screen (Configuring the Administrator Account).
2. Click **weblogic_domain**, and then click **System Mbean Browser**.
3. In the search box, enter `OIMSOAIntegrationMBean`, and click **Search**. The mbean is displayed.

> **Note:**
>
> If Oracle Identity Governance still starting (coming up) or is just started (RUNNING MODE), the Enterprise Manager does not show any Mbeans defined by OIM. Wait for two minutes for the server to start, and then try searching for the Mbean in **System Mbean Browser** of the Enterprise Manager.

4. Go to the **Operations** tab of mbean, and select **integrateWithSOAServer**.
5. Enter the following information:
   - **Weblogic Administrator User Name**: Enter the name of the WebLogic administrator. For example: `weblogic`.
   - **Weblogic Administrator Password**: Enter the password for the above account.
   - **OIM Front end URL**: Set this URL to the load balancer virtual host used for internal call backs. For example:
     `http://igdinternal.example.com:7777/`
   - **OIM External Front End URL**: Set this URL to the main load balancer virtual host used for Oracle Identity Governance. For example:
     `https://prov.example.com:443/`
   - **SOA SOAP URL**: Set this URL to the SOA Kubernetes Service used for internal call backs. For example:
     `http://governancedomain-cluster-soa-cluster.oigns.svc.cluster.local:8001`
   - **SOA RMI URL**: Set this URL to the SOA Kubernetes Service used for internal call backs. For example:
     `t3://<OIG_DOMAIN_NAME>-cluster-soa-cluster.<OIG NAMESPACE>.svc.cluster.local:8001`

     Example of the above URL:

     `t3://governancedomain-cluster-soa-cluster.oigns.svc.cluster.local:8001`
   - **UMS Webservice URL**: Set this URL to the SOA Kubernetes Service used for internal call backs. For example:
     `http://governancedomain-cluster-soa-cluster.oigns.svc.cluster.local:8001/ucs/messaging/webservice`
6. Click **Invoke**.

# Managing the Notification Service

An event is an operation that occurs in Oracle Identity Manager, such as user creation, request initiation, or any custom event created by the user. These events are generated as part of the business operations or through the generation of errors. Event definition is the metadata that describes the event.

To define the metadata for events, you must identify all event types supported by a functional component. For example, as a part of the scheduler component, metadata is defined for a scheduled job execution failure and shutting down of the scheduler. Every time a job fails or the scheduler shuts down, the associated events get triggered, and the notifications associated with the event get sent.

The data available in the event is used to create the content of the notification. The different parameters defined for an event help the system to select the appropriate notification template.

The various parameters defined for an event help the system decide which event variables should be made available at template design time.

A notification template is used to send notifications. These templates contain variables that refer to available data to provide more context to the notifications. The notification is sent through a notification provider. Examples of such channels are e-mail, Instant Messaging (IM), Short Message Service (SMS), and voice. To use these notification providers, Oracle Identity Manager uses Oracle User Messaging Service (UMS).

At the back end, the notification engine is responsible for generating the notification and utilizing the notification provider to send the notification.



# Using Oracle Unified Messaging for Notification

Using Oracle Unified Messaging (UMS) for notification involves configuring the UMS email notification provider properties and adding the CSF key.

To configure SMTP Email Notification Provider properties by using the **UMSEmailNotificationProviderMBean**:

1. Log in to the Oracle Fusion Middleware Control using the following URL:

   ```
   http://igdadmin.example.com/em
   ```

   or

   ```
   http://k8worker1.example.com:30711/em
   ```

   The Administration Server host and port number were in the URL on the End of Configuration screen (Writing Down Your Domain Home and Administration Server URL). The default Administration Server port number is `7001`.

   The login credentials were provided on the Administrator Account screen (Configuring the Administrator Account).
2. Click **weblogic_domain**, and then click **System Mbean Browser**.
3. In the search box, enter `UMSEmailNotificationProviderMBean`, and click **Search**. The mbean is displayed.

   > **Note:**
   >
   > If Oracle Identity Governance still starting (coming up) or is just started (RUNNING MODE), the Enterprise Manager does not show any Mbeans defined by OIM. Wait for two minutes for the server to start, and then try searching for the Mbean in **System Mbean Browser** of the Enterprise Manager.
4. Ensure that the correct information is entered for your email server in particular:

**Table 19-4 SMTP Email Notification Provider Properties**

| Attribute | Value |
|---|---|
| Enabled | Set to true. |
| MailServerName | Set to the host name of your email server. |
| WSUrl | `http://<OIG_DOMAIN_NAME>-cluster-soa-cluster.<OIGNS>.svc.cluster.local:8001/ucs/messaging/webservice` |

5. Click **Apply** to save the changes.

## Updating the CSF Key

To update the CSF key:

1. Log in to Oracle Enterprise Manager.
2. Click WebLogic Domain, click **Security**, and then **Credentials**.
3. Expand **oracle.wsm.security** and click **Create Key**.
4. Enter the following information.

**Table 19-5 CSF Key Properties**

| Attribute | Value |
|---|---|
| Key name | Enter the value of the credential Key, this must be the same value as defined in Using Oracle Unified Messaging for Notification for example; mailUser. |
| Username | Enter the name of the user you use to authenticate with your email server. |
| Password/Confirm Password | Enter the password of the user you use to authenticate with your email server. |
| Description | Provide a description of the key being created. For example, Mail Server Credentials |

5. Click **OK**.

# Configuring the Messaging Drivers

Each messaging driver needs to be configured. You have to configure this service if you want to enable OAM's forgotten password functionality.

- Configuring the Email Driver

## Configuring the Email Driver

To configure the driver to send and emails then you need to perform the following steps:

1. Log in to the Oracle Fusion Middleware Control.
2. Click the **Target Navigation** icon next to the Domain name.
3. Click **usermessagingserver (soa_server)** under User Messaging Service. A list of all the drivers will be shown.

4. Click **Configure Driver** next to the User Messaging Email Driver.
5. If a configuration does not exist then click **Create**. If the configuration exists, click **Edit**.
6. Update the attributes with the required details.

**Table 19-6 Configuring the Email Driver Attributes**

| Attributes | Values |
|---|---|
| Name | MyemailServer |
| Sender Address | Enter the From email address for the emails you wish to send in the format: EMAIL:myuser@example.com |
| Capability | Choose whether you are going to send or receive emails.<br>Complete the following Email Properties using the values specific to your organisation. Contact your email administrator for details, the details below are for Sending only. Refer to the documentation for receiving email details.<br>• Outdoing Mail server.<br>• Outgoing Mail server port<br>• Outgoing email Server Security<br>• Outgoing User name and password, if your email server requires it. |

7. Click **Test** to validate the information.
8. Click **OK** to save the information.

# Increasing Database Connection Pool Size

The default database connection pool size needs to be increased when Oracle Identity Governance is used in conjunction with a connector that allows interactions with an LDAP directory.

To do this, complete the following steps:

1. Log in to the WebLogic Server Administration Console in `IAMGovernanceDomain`.
2. Click **Lock & Edit**.
3. Click **Services** and then click **Data Sources**.
4. Click the data source **mds-oim**.
5. Go to the **Connection Pool** tab.
6. Modify the following properties with the values specified:
   - Initial Capacity: 50
   - Maximum Capacity: 150
   - Minimum Capacity: 50
   - Inactive Connection Timeout value to 30 from any other value

> **Note:**
>
> Inactive Connection Timeout is in the Advanced section.

7. Click **Save**.
8. Click **Activate Changes**.
9. You will receive a message **All changes have been activated. No restarts are necessary**.

# Integrating Oracle Identity Governance with LDAP

Before you integrate OIG with LDAP, you should configure the connector for LDAP and add the required object classes if any are missing.

> **Note:**
>
> The following sections require that you edit the property files and use those property files with the `./OIGOAMIntegration.sh` script.
>
> In a Kubernetes container there is no editor. So, copy the file to the local machine, edit it, and then copy it back. The syntax of the command is:
>
> ```
> kubectl cp <OIGNS>/<OIG_DOMAIN_NAME>-adminserver:<SOURCE FILENAME>
> <DESTINATION_FILENAME>
> ```
>
> Edit the file, and then copy it back. The syntax to copy a file back to Kubernetes is:
>
> ```
> kubectl cp /workdir/OIG/configureLDAPConnector.config oigns/
> governancedomain-adminserver:/u01/oracle/idm/server/ssointg/config/
> configureLDAPConnector.config
> ```



## Configuring the Oracle Connector for LDAP

The Oracle Connector for LDAP enables you to store users and passwords in a certified LDAP directory. Configure the connector before using it. Perform the following steps to configure the connector:

1. Change directory to `/u01/oracle/idm/server/ssointg/config`.
2. Edit the `configureLDAPConnector.config` file.

   Following is the template file:

   ```
   ## [configureLDAPConnector]
   IDSTORE_DIRECTORYTYPE=<LDAP_TYPE>
   OIM_HOST=<OIG_DOMAIN_NAME>-cluster-oim-cluster.<OIGNS>.svc.cluster.local
   ```

```
OIM_PORT=14000
OIM_SERVER_SYSADMIN_USER=<LDAP_XELSYSADM_USER>
OIM_WLSHOST=<OIG_DOMAIN_NAME>-adminserver-
external.<OIGNS>.svc.cluster.local
OIM_WLSPORT=<OIG_ADMIN_PORT>
OIM_WLSADMIN=<OIG_WEBLOGIC_USER>
OIM_WLSADMIN_PWD=<OIG_WEBLOGIC_PWD>
IDSTORE_HOST=<LDAP_HOST>
IDSTORE_PORT=<LDAP_PORT>
IDSTORE_BINDDN=<LDAP_ADMIN_USER>
IDSTORE_OIMADMINUSERDN=cn=<LDAP_OIGLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEA
RCHBASE>
IDSTORE_SEARCHBASE=<LDAP_SEARCHBASE>
IDSTORE_USERSEARCHBASE=<LDAP_USER_SEARCHBASE>
IDSTORE_GROUPSEARCHBASE=<LDAP_GROUP_SEARCHBASE>
IDSTORE_USERSEARCHBASE_DESCRIPTION=Default user container
IDSTORE_GROUPSEARCHBASE_DESCRIPTION=Default group container
IDSTORE_EMAIL_DOMAIN=<OIG_EMAIL_DOMAIN>
## For ActiveDirectory use the values of "yes" or "no". i.e.
IS_LDAP_SECURE=yes/no
IS_LDAP_SECURE=false
SSO_TARGET_APPINSTANCE_NAME=SSOTarget
## Path to expanded connector bundle: e.g. for OID and OUD
CONNECTOR_MEDIA_PATH=/u01/oracle/user_projects/domains/
ConnectorDefaultDirectory/OID-12.2.1.3.0
WLS_OIM_SYSADMIN_USER_PWD=<LDAP_USER_PWD>
IDSTORE_BINDDN_PWD=<LDAP_ADMIN_PWD>
IDSTORE_OIMADMINUSER_PWD=<LDAP_USER_PWD>
```

This is a sample of the file, as an example:

```
##------------------------------------------------------------##
## [configureLDAPConnector]
IDSTORE_DIRECTORYTYPE=OUD
OIM_HOST=governancedomain-cluster-oim-cluster.oigns.svc.cluster.local
OIM_PORT=14000
OIM_SERVER_SYSADMIN_USER=xelsysadm
OIM_WLSHOST=governancedomain-adminserver-external.oigns.svc.cluster.local
OIM_WLSPORT=7101
OIM_WLSADMIN=weblogic
OIM_WLSADMIN_PWD=<Password1>
IDSTORE_HOST=edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local
IDSTORE_PORT=1389
IDSTORE_BINDDN=cn=oudadmin
IDSTORE_OIMADMINUSERDN=cn=oimLDAP,cn=systemids,dc=example,dc=com
IDSTORE_SEARCHBASE=dc=example,dc=com
IDSTORE_USERSEARCHBASE=cn=Users,dc=example,dc=com
IDSTORE_GROUPSEARCHBASE=cn=Groups,dc=example,dc=com
IDSTORE_USERSEARCHBASE_DESCRIPTION=Default user container
IDSTORE_GROUPSEARCHBASE_DESCRIPTION=Default group container
IDSTORE_EMAIL_DOMAIN=example.com
## For ActiveDirectory use the values of "yes" or "no". i.e.
IS_LDAP_SECURE=yes/no
IS_LDAP_SECURE=false
SSO_TARGET_APPINSTANCE_NAME=SSOTarget
```

```
## Path to expanded connector bundle: e.g. for OID and OUD
CONNECTOR_MEDIA_PATH=/u01/oracle/user_projects/ConnectorDefaultDirectory/
OID-12.2.1.3.0
WLS_OIM_SYSADMIN_USER_PWD=<PASSWORD>
IDSTORE_BINDDN_PWD=<PASSWORD>
IDSTORE_OIMADMINUSER_PWD=<PASSWORD>
```

> **Note:**
>
> You can also specify the passwords directly in the file, if required. If you do not specify the passwords, you will be prompted for them at runtime.
>
> Parameters are:
>
> - *OIM_WLSADMIN_PWD*
> - *IDSTORE_BINDDN_PWD*
> - *WLS_OIM_SYSADMIN_USER_PWD*
> - *ADMIN_USER_PWD*
> - *IDSTORE_OIMADMINUSER_PWD*

Save the file.

> **Note:**
>
> You should use the same values as you specified for these parameters in Creating Configuration Files.

3. Execute the script `OIGOAMIntegration` for configuring the connector.

For example:

```
kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash

cd /u01/oracle/idm/server/ssointg/bin

export JAVA_HOME=/u01/jdk

export APPSERVER_TYPE=wls

export ORACLE_HOME=/u01/oracle

export OIM_ORACLE_HOME=/u01/oracle/idm

export WL_HOME=$ORACLE_HOME/wlserver

chmod 750 _OIGOAMIntegration.sh OIGOAMIntegration.sh

./OIGOAMIntegration.sh -configureLDAPConnector
```

## Adding Missing Object Classes

If any users existed in LDAP prior to enabling the Oracle Identity Manager, then these new users may be missing the object classes used to control OIM/OAM integration. To add these missing object classes to these users, run the following commands:

> **Note:**
>
> To successfully execute this process, the `ldapsearch` binary is required to be in your user's PATH and the `screen` package is required to be installed on your host.

1. Change directory to `/u01/oracle/idm/server/ssointg/config`.
2. Edit the file `addMissingObjectClasses.config` updating the properties as shown below:

   Following is the template file:

```
IDSTORE_DIRECTORYTYPE=<LDAP_TYPE>
IDSTORE_HOST=<LDAP_HOST>
IDSTORE_PORT=<LDAP_PORT>
IDSTORE_BINDDN=<LDAP_ADMIN_USER>
IDSTORE_USERSEARCHBASE=<LDAP_USER_SEARCHBASE>
```

This is a sample of the file, as an example:

```
IDSTORE_DIRECTORYTYPE=OUD
IDSTORE_HOST=edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local
IDSTORE_PORT=1389
IDSTORE_BINDDN=oudadmin
IDSTORE_USERSEARCHBASE=cn=Users,dc=example,dc=com
```

Save the file.

3. Execute the script `OIGOAMIntegration`.

   For example:

```
kubectl exec -n oigns -ti governancedomain-adminserver --/bin/bash

cd /u01/oracle/idm/server/ssointg/bin

export JAVA_HOME=/u01/jdk

export APPSERVER_TYPE=wls

export ORACLE_HOME=/u01/oracle

export OIM_ORACLE_HOME=/u01/oracle/idm

export WL_HOME=$ORACLE_HOME/wlserver

./OIGOAMIntegration.sh -addMissingObjectClasses
```

   You will be prompted to enter the password of the LDAP directory administrator account, if you have not provided them as inputs to the parameter file.

# Integrating Oracle Identity Governance and Oracle Access Manager

You have to complete several tasks to integrate Oracle Identity Governance and Oracle Access Manager. These tasks include creating the WLS authentication providers, deleting OIMSignatureAuthenticator and recreating OUDAuthenticator, adding the administration role to the new administration group, and so on.

- Creating WLS Authentication Providers
- Deleting OIMSignatureAuthenticator
- Recreating OUDAuthenticator
- Adding the Administration Role to the New Administration Group

- Configuring SSO Integration in the Governance Domain
- Enabling OAM Notifications
- Updating the Value of MatchLDAPAttribute in oam-config.xml
- Updating the TapEndpoint URL

# Creating WLS Authentication Providers

You must configure the WLS authentication providers to set SSO logout and security providers in the OIG domain. This enables both the SSO login and OIM client-based login to work appropriately.

1. Change directory to `/u01/oracle/idm/server/ssointg/config`.
2. Edit the `configureWLSAuthnProviders.config` file to set the following properties:

   This is the template file:

```
OIM_WLSHOST: <OIG_DOMAIN_NAME>-adminserver.<OIGNS>.svc.cluster.local
OIM_WLSPORT=<OIG_ADMIN_PORT>
OIM_WLSADMIN=<OIG_WEBLOGIC_USER>
OIM_WLSADMIN_PWD=<OIG_WEBLOGIC_PWD>
OIM_SERVER_NAME=oim_server1
## DIRTYPE values can be [OID | OUD | AD]
IDSTORE_DIRECTORYTYPE=<LDAP_TYPE>
IDSTORE_HOST=<LDAP_HOST>
IDSTORE_PORT=<LDAP_PORT>
IDSTORE_BINDDN=<LDAP_ADMIN_USER>
IDSTORE_BINDDN_PWD=<LDAP_ADMIN_PWD>
IDSTORE_USERSEARCHBASE=<LDAP_USER_SEARCHBASE>
IDSTORE_GROUPSEARCHBASE=<LDAP_GROUP_SEARCHBASE>
```

   This is a sample file:

```
OIM_WLSHOST: governancedomain-adminserver.oigns.svc.cluster.local
OIM_WLSPORT: 7101
OIM_WLSADMIN=weblogic
OIM_WLSADMIN_PWD=<password>
OIM_SERVER_NAME=oim_server1
IDSTORE_DIRECTORYTYPE=OUD
IDSTORE_HOST=edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local
IDSTORE_PORT=1389
IDSTORE_BINDDN=cn=oudadmin
IDSTORE_BINDDN_PWD=<password>
IDSTORE_USERSEARCHBASE=cn=Users,dc=example,dc=com
IDSTORE_GROUPSEARCHBASE=cn=Groups,dc=example,dc=com
```

   Save the file.

3. Execute the `OIGOAMIntegration` script for configuring the WebLogic security providers.

```
kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash

cd /u01/oracle/idm/server/ssointg/bin

export JAVA_HOME=/u01/jdk

export APPSERVER_TYPE=wls

export ORACLE_HOME=/u01/oracle

export OIM_ORACLE_HOME=/u01/oracle/idm

export WL_HOME=$ORACLE_HOME/wlserver

./OIGOAMIntegration.sh -configureWLSAuthnProviders
```

## Deleting OIMSignatureAuthenticator

The `createWLSAuthenticator` script creates a new security provider called `OIMSignatureAuthenticator`. This security provider is not required in Oracle Identity Manager 12c.

To delete the security provider:

1. Log in to the WebLogic Server Administration Console, if not already logged in.
2. Click **Lock & Edit**.
3. Click **Security Realms** on the left navigation pane.
4. Click the **myrealm** default realm entry.
5. Click the **Providers** tab.
6. Select the security provider **OIMSignatureAuthenticator**.
7. Click **Delete**.
8. Click **Yes** to confirm the deletion.
9. Click **Activate Changes** to propagate the changes.

## Recreating OUDAuthenticator

If your target directory is OUD, then you must delete and recreate the `OUDAuthenticator` security provider.

To delete the security provider:

1. Log in to the WebLogic Server Administration Console, if not already logged in.

2. Click **Lock & Edit**.
3. Click **Security Realms** on the left navigation pane.
4. Click the **myrealm** default realm entry.
5. Click the **Providers** tab.
6. Select the security provider **OUDAuthenticator**.
7. Click **Delete**.
8. Click **Yes** to confirm the deletion.
9. Click **Activate Changes** to propagate the changes.

To recreate the security provider:

1. Log in to the WebLogic Server Administration Console using the URL.

   `http://k8worker1.example.com:30711/console`

2. Click **Security Realms** in the left navigational bar.
3. Click the **myrealm** default realm entry.
4. Click the **Providers** tab.
5. Click **Lock & Edit** in the Change Center.
6. Click the **New** button below the **Authentication Providers** table.
7. Enter a name for the provider.

   Use one of the following names, based on the LDAP directory service you are planning to use as your credential store:

   `OUDAuthenticator` for Oracle Unified Directory

8. From the **Type** drop-down list, select the authenticator type **OracleUnifiedDirectoryAuthenticator** for Oracle Unified Directory.
9. Click **OK** to return to the Providers screen.
10. On the Providers screen, click the newly created authenticator in the table.
11. Select **SUFFICIENT** from the **Control Flag** drop-down menu.

    Setting the control flag to **SUFFICIENT** indicates that if the authenticator can successfully authenticate a user, then the authenticator should accept that authentication and should not continue to invoke any additional authenticators.

    If the authentication fails, it will fall through to the next authenticator in the chain. Make sure all subsequent authenticators also have their control flags set to **SUFFICIENT**; in particular, check the **`DefaultAuthenticator`** and make sure that its control flag is set to **SUFFICIENT**.

12. Click **Save** to persist the change of the control flag setting.
13. Click the **Provider Specific** tab and enter the details specific to your LDAP server, as shown in the following table.

> **Note:**
>
> Only the required fields are discussed in this procedure. For information about all the fields on this page, consider the following resources:
>
> - To display a description of each field, click **Help** on the **Provider Specific** tab.
> - For more information on setting the **User Base DN**, **User From Name Filter**, and **User Attribute** fields, see Configuring Users and Groups in the Oracle Internet Directory and Oracle Virtual Directory Authentication Providers in *Administering Security for Oracle WebLogic Server*.

| Parameter | Sample Value | Value Description |
|---|---|---|
| Host | For example:<br>`edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local` | The LDAP server's server ID. |
| Port | For example: `1389` | The LDAP server's port number. |
| Principal | For example:<br>`cn=oimLDAP,cn=systemids,dc=example,dc=com` | The LDAP user DN used to connect to the LDAP server. |
| Credential | Enter LDAP password. | The password used to connect to the LDAP server. |
| SSL Enabled | Unchecked (clear) | Specifies whether SSL protocol is used when connecting to the LDAP server. |
| User Base DN | For example: `cn=users,dc=example,dc=com` | Specify the DN under which your users start. |
| All Users Filter | `(&(uid=*)(objectclass=person))` | Instead of a default search criteria for **All Users Filter**, search all users based on the `uid` value.<br>If the **User Name Attribute** for the user object class in the LDAP directory structure is a type other than `uid`, then change that type in the **User From Name Filter** field.<br>For example, if the **User Name Attribute** type is `cn`, then this field should be set to:<br>`(&(cn=*)(objectclass=person)))` |
| User From Name Filter | For example:<br>`(&(uid=%u)(objectclass=person))` | If the **User Name Attribute** for the user object class in the LDAP directory structure is a type other than `uid`, then change that type in the settings for the **User From Name** Filter.<br>For example, if the **User Name Attribute** type is `cn`, then this field should be set to:<br>`(&(cn=%u)(objectclass=person))).` |
| User Name Attribute | For example: `uid` | The attribute of an LDAP user object that specifies the name of the user. |
| Use Retrieved User Name as Principal | Checked | Must be turned on. |
| Group Base DN | For example:<br>`cn=groups,dc=example,dc=com` | Specify the DN that points to your Groups node. |

| Parameter | Sample Value | Value Description |
|---|---|---|
| All Groups Filter | `(&(cn=*) (objectclass=groupOfUniqueNames))` | Specify the group filter. |
| GUID Attribute | `entryuuid` | This value is prepopulated with `entryuuid` when `OracleUnifiedDirectoryAuthenticator` is used for OUD. Check this value if you are using Oracle Unified Directory as your authentication provider. |

14. Click **Save** to save the changes.
15. Return to the Providers page by clicking **Security Realms** in the right navigation pane, clicking the default realm name (**myrealm**), and then **Providers**.
16. Click **Reorder** and use the resulting page to reorder the list of providers so that they match the order given below:

    **List of Authentication Providers**

    - OAMIDAsserter
    - OUDAuthenticator
    - DefaultAuthenticator
    - OIMAuthenticationProvider
    - Trust Service Identity Asserter
    - DefaultIdentityAsserter
17. Click **OK**.
18. In the Change Center, click **Activate Changes**.
19. You have to restart the domain for the changes to take effect. You can restart by using the following commands:

    ```
    kubectl -n <OIGNS> patch domains <OIG_DOMAIN_NAME> --type='json' -
    p='[{"op": "replace", "path": "/spec/serverStartPolicy", "value":
    "Never" }]'
    ```

    After everything is stopped, it can be restarted using the following command:

    ```
    kubectl -n <OIGNS> patch domains <OIG_DOMAIN_NAME> --type='json' -
    p='[{"op": "replace", "path": "/spec/serverStartPolicy", "value":
    "IfNeeded" }]'
    ```

    For example:

    ```
    kubectl -n oigns patch domains governancedomain --type='json' -p='[{"op":
    "replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
    kubectl -n oigns patch domains governancedomain --type='json' -p='[{"op":
    "replace", "path": "/spec/serverStartPolicy", "value": "IfNeeded" }]'
    ```
20. After the restart, review the contents of the `AdminServer.log` file, available in the following location:

    ```
    /u01/oracle/user_projects/domains/logs/governancedomain
    ```

Verify that no LDAP connection errors occurred. For example, look for errors such as the following:

```
The LDAP authentication provider named "OUDAuthenticator" failed to make
connection to ldap server at ...
```

If you see such errors in the log file, then check the authorization provider connection details to verify they are correct and try saving and restarting the Administration Server again.

21. After you restart and verify that no LDAP connection errors are in the log file, try browsing the users and groups that exist in the LDAP provider:

    In the Administration Console, navigate to the **Security Realms > myrealm > Users and Groups** page. You should be able to see all users and groups that exist in the LDAP provider structure.

## Adding the Administration Role to the New Administration Group

This enables all users that belong to the group to be administrators for the domain.

To assign the Administration role to the new enterprise deployment administration group:

1. Log in to the WebLogic Administration Server Console by using the administration credentials that you provided in the Configuration Wizard.

   Do not use the credentials for the administration user that you created and provided for the new authentication provider.
2. In the left pane of the Administration Console, click **Security Realms**.
3. Click the default security realm (**myrealm**).
4. Click the **Roles and Policies** tab.
5. Expand the **Global Roles** entry in the table and click **Roles**.
6. Click the **Admin** role.
7. Click **Add conditions**.
8. Select **Group** from the **Predicate List** drop-down menu, and then click **Next**.
9. Enter `WLSAdministrators` in the **Group Argument Name** field, and then click **Add**.

   `WLSAdministrators` is added to the list box of arguments.
10. Click **Finish** to return to the Edit Global Role page.

    The `WLSAdministrators` group is now listed.
11. Click **Save** to finish adding the **Admin** Role to the `WLSAdministrators` group.
12. Validate that the changes were made by logging in to the WebLogic Administration Server Console by using the new `weblogic_iam` user credentials.

    If you can log in to the Oracle WebLogic Server Administration Console and Fusion Middleware Control with the credentials of the new administration user that you just provisioned in the new authentication provider, then you have configured the provider successfully.

# Configuring SSO Integration in the Governance Domain

After deploying the connector, the next step in the process is the configuration of SSO in the domain. To configure SSO, perform the following steps:

1. Change directory to */u01/oracle/idm/server/ssointg/config*
2. Edit the file `configureSSOIntegration.config` to update the properties in the section **configureSSOIntegration**, as shown below:

   This is the template file:

```
NAP_VERSION=4
COOKIE_EXPIRY_INTERVAL=120
OAM_HOST=<OAM_LOGIN_LBR_HOST>
OAM_PORT=<OAM_LOGIN_LBR_PORT>
ACCESS_SERVER_HOST=<OAM_DOMAIN_NAME>-oap.<OAMNS>.svc.cluster.local
ACCESS_SERVER_PORT=<OAM_OAP_PORT>
OAM_SERVER_VERSION=12c
WEBGATE_TYPE=ohsWebgate12c
ACCESS_GATE_ID=Webgate_IDM
ACCESS_GATE_PWD=<PASSWORD>
COOKIE_DOMAIN=example.com
OAM_TRANSFER_MODE=open
OIM_LOGINATTRIBUTE=uid
SSO_ENABLED_FLAG=true
SSO_INTEGRATION_MODE=CQR
OAM11G_WLS_ADMIN_HOST=<OAM_DOMAIN_NAME>-
adminserver.<OAMNS>.svc.cluster.local
OAM11G_WLS_ADMIN_PORT=30012
OAM11G_WLS_ADMIN_USER=<OAM_WEBLOGIC_USER>
OAM11G_WLS_ADMIN_PASSWD=<OAM_WEBLOGIC_PWD>
OAM11G_IDSTORE_NAME=OAMIDSTORE
## Required if OAM_TRANSFER_MODE is not OPEN
SSO_KEYSTORE_JKS_PASSWORD=<GLOBAL_PASSPHRASE>
SSO_GLOBAL_PASSPHRASE=<GLOBAL_PASSPHRASE>
OIM_WLSHOST=<OIG_DOMAIN_NAME>-adminserver.<OIGNS>.svc.cluster.local
OIM_WLSPORT=<OIG_ADMIN_PORT>
OIM_WLSADMIN=<OIG_WEBLOGIC_USER>
IDSTORE_OAMADMINUSER_PWD=<LDAP_USER_PWD>
OIM_SERVER_NAME=<OIM_SERVER_NAME>
IDSTORE_OAMADMINUSER=<LDAP_OAMADMIN_USER>
```

   This is a sample file:

```
NAP_VERSION=4
COOKIE_EXPIRY_INTERVAL=120
OAM_HOST=login.example.com
OAM_PORT=443
ACCESS_SERVER_HOST=accessdomain-oap.oamns.svc.cluster.local
ACCESS_SERVER_PORT=5575
OAM_SERVER_VERSION=12c
WEBGATE_TYPE=ohsWebgate12c
ACCESS_GATE_ID=Webgate_IDM
ACCESS_GATE_PWD=<password>
```

```
COOKIE_DOMAIN=example.com
OAM_TRANSFER_MODE=Simple
OIM_LOGINATTRIBUTE=uid
SSO_ENABLED_FLAG=true
SSO_INTEGRATION_MODE=CQR
OAM11G_WLS_ADMIN_HOST=accessdomain-adminserver.oamns.svc.cluster.local
OAM11G_WLS_ADMIN_PORT=30012
OAM11G_WLS_ADMIN_USER=weblogic
OAM11G_WLS_ADMIN_PASSWD=<PASSWORD>
OAM11G_IDSTORE_NAME=OAMIDSTORE
## Required if OAM_TRANSFER_MODE is not OPEN
SSO_KEYSTORE_JKS_PASSWORD=<GLOBAL_PASSPHRASE>
SSO_GLOBAL_PASSPHRASE=<GLOBAL_PASSPHRASE>
OIM_WLSHOST=governancedomain-adminserver.oigns.svc.cluster.local
OIM_WLSPORT=7101
OIM_WLSADMIN=weblogic
IDSTORE_OAMADMINUSER_PWD=<password>
OIM_SERVER_NAME=oim_server1
IDSTORE_OAMADMINUSER=oamadmin
```

Save the file when done.

> **Note:**
>
> Substitute the variables with values applicable to your deployment. See Variables Used in this Chapter. The other specified values should be used as is.

3. Execute the `OIGOAMIntegration` script for configuring SSO Integration.

For example:

```
kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash

cd /u01/oracle/idm/server/ssointg/bin

export JAVA_HOME=/u01/jdk

export APPSERVER_TYPE=wls

export ORACLE_HOME=/u01/oracle

export OIM_ORACLE_HOME=/u01/oracle/idm

export WL_HOME=$ORACLE_HOME/wlserver

chmod 750 _OIGOAMIntegration.sh OIGOAMIntegration.sh

./OIGOAMIntegration.sh -configureSSOIntegration
```

4. Restart the domains **IAMAccessDomain** and **IAMGovernanceDomain**.

# Enabling OAM Notifications

After deploying the connector, the next step in the process is to tell OIM how to interact with OAM for terminating a user session after a user name expires or gets terminated. To complete this activity, you need to perform the following steps:

1. Change directory to `/u01/oracle/idm/server/ssointg/config`.
2. Edit the `enableOAMSessionDeletion.config` file to update the properties in the **enableOAMNotifications** section.

   This is the template of the file:

```
OIM_WLSHOST: <OIG_DOMAIN_NAME>-adminserver.<OIGNS>.svc.cluster.local
OIM_WLSPORT=<OIG_ADMIN_PORT>
OIM_WLSADMIN=<OIG_WEBLOGIC_USER>
OIM_WLSADMIN_PWD=<OIG_WEBLOGIC_PWD>
IDSTORE_DIRECTORYTYPE=<LDAP_TYPE>
IDSTORE_HOST=<LDAP_HOST>
IDSTORE_PORT=<LDAP_PORT>
IDSTORE_BINDDN=<LDAP_ADMIN_USER>
IDSTORE_GROUPSEARCHBASE=<LDAP_GROUP_SEARCHBASE>
IDSTORE_SYSTEMIDBASE: cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>
IDSTORE_OAMADMINUSER: <LDAP_OAMADMIN_USER>
IDSTORE_OAMSOFTWAREUSER: <LDAP_OAMLDAP_USER>
```

```
IDSTORE_USERSEARCHBASE: <LDAP_USER_SEARCHBASE>
OIM_SERVER_NAME: <OIM_SERVER_NAME>
```

Here is the sample file:

```
OIM_WLSHOST: governancedomain-adminserver.oigns.svc.cluster.local
OIM_WLSPORT: 7101
OIM_WLSADMIN: weblogic
OIM_WLSADMIN_PWD: <password>
IDSTORE_DIRECTORYTYPE: OUD
IDSTORE_HOST: edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local
IDSTORE_PORT: 1389
IDSTORE_BINDDN: cn=oudadmin
IDSTORE_GROUPSEARCHBASE: cn=Groups,dc=example,dc=com
IDSTORE_SYSTEMIDBASE: cn=systemids,dc=example,dc=com
IDSTORE_OAMADMINUSER: oamAdmin
IDSTORE_OAMSOFTWAREUSER: oamLDAP
IDSTORE_USERSEARCHBASE: cn=Users,dc=example,dc=com
OIM_SERVER_NAME: oim_server1
```

3. Execute the script `OIGOAMIntegration` for enabling notifications.

   For example:

```
kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash


cd /u01/oracle/idm/server/ssointg/bin


export JAVA_HOME=/u01/jdk


export APPSERVER_TYPE=wls


export ORACLE_HOME=/u01/oracle


export OIM_ORACLE_HOME=/u01/oracle/idm


export WL_HOME=$ORACLE_HOME/wlserver


chmod 750 _OIGOAMIntegration.sh OIGOAMIntegration.sh


./OIGOAMIntegration.sh -enableOAMSessionDeletion
```

# Updating the Value of MatchLDAPAttribute in oam-config.xml

To complete the Oracle Identity Governance integration with Oracle Access Manager, one of the settings in the Oracle Access Manager's `oam-config.xml` file needs to be changed. As of version 12c, this file is stored in the database and should not be edited directly.

The procedure below shows how to use the REST API to change one of the values in the `oam-config.xml` file:

> **Note:**
>
> Ensure that the cURL package has been added to the host by executing `which curl` at the command line. If the package is not installed, an administrator must install the package by executing `yum install curl`.

1. Find the component number of the DAPModule, by executing the following:

```
curl -i -u oamadmin:<LDAP_USER_PWD> http://k8worker1.example.com:30701/iam/admin/
config/api/v1/config?path=/DeployedComponent/Server/NGAMServer/Profile/
AuthenticationModules/DAPModules
```

Example output:

```
HTTP/1.1 200 OK
Date: Tue, 09 Jul 2019 20:30:33 GMT
Content-Length: 625
Content-Type: text/xml
X-ORACLE-DMS-ECID: 6f9baf65-751b-4fc9-b2e1-ade5b38063ff-00000427
X-ORACLE-DMS-RID: 0
Set-Cookie:
JSESSIONID=g3LYbkLA2bs5-9zfoMBqKTBbk0mky_8URGgzFnbNkm8n3tK63tq4!
1064195705; path=/; HttpOnly


<Configuration xmlns="http://www.w3.org/2001/XMLSchema" schemaLocation="http://
higgins.eclipse.org/sts/Configuration Configuration.xsd" Path="/DeployedComponent/
Server/NGAMServer/Profile/AuthenticationModules/DAPModules">

<Setting Name="DAPModules" Type="htf:map">
    <Setting Name="7DASE52D" Type="htf:map">
      <Setting Name="MAPPERCLASS"
Type="xsd:string">oracle.security.am.engine.authn.internal.executor.DAPAttributeMappe
r</Setting>
      <Setting Name="MatchLDAPAttribute" Type="xsd:string”>cn</Setting>
      <Setting Name="name" Type="xsd:string">DAP</Setting>
    </Setting>
  </Setting>
```

> **Note:**
>
> The component number under the line that reads: "<Setting Name="DAPModules" Type="htf:map">" This will need to be used for the configuration change. In the above example, "7DASE52D" is the component number. The value which will need to be changed is the value of MatchLDAPAttribute. In the above example, "User Name" is the current value.

2. CD to `/tmp` and create a configuration file `MatchLDAPAttribute_input.xml` with the following contents:

```
<Configuration>
  <Setting Name="MatchLDAPAttribute" Type="xsd:string" Path="/DeployedComponent/
Server/NGAMServer/Profile/AuthenticationModules/DAPModules/7DASE52D/
```

```
MatchLDAPAttribute">uid</Setting>
</Configuration>
```

> **Note:**
>
> The component number noted from above is inserted between `DAPModules` and the `MatchLDAPAttribute` portions of the path. The configuration file will change the value of `MatchLDAPAttribute` from *User Name* to *uid*.

3. Insert the change back into the OAM configuration, by executing the following:

```
curl -u oamadmin:<LDAP_USER_PWD> -H 'Content-Type: text/xml' -X PUT http://
k8worker1.example.com:30701/iam/admin/config/api/v1/config -d
@MatchLDAPAttribute_input.xml
```

4. Validate the change with the same command you originally used to query the component, noting the value of the `MatchLDAPAttribute` tag:

```
curl -i -u oamadmin:<LDAP_USER_PWD> http://k8worker1.example.com:30701/iam/admin/
config/api/v1/config?path=/DeployedComponent/Server/NGAMServer/Profile/
AuthenticationModules/DAPModules
```

Example output:

```
HTTP/1.1 200 OK
Date: Tue, 09 Jul 2019 20:30:33 GMT
Content-Length: 625
Content-Type: text/xml
X-ORACLE-DMS-ECID: 6f9baf65-751b-4fc9-b2e1-ade5b38063ff-00000427
X-ORACLE-DMS-RID: 0
Set-Cookie:
JSESSIONID=g3LYbkLA2bs5-9zfoMBqKTBbk0mky_8URGgzFnbNkm8n3tK63tq4!
1064195705; path=/; HttpOnly


<Configuration xmlns="http://www.w3.org/2001/XMLSchema" schemaLocation="http://
higgins.eclipse.org/sts/Configuration Configuration.xsd" Path="/DeployedComponent/
Server/NGAMServer/Profile/AuthenticationModules/DAPModules">

<Setting Name="DAPModules" Type="htf:map">
    <Setting Name="7DASE52D" Type="htf:map">
      <Setting Name="MAPPERCLASS"
Type="xsd:string">oracle.security.am.engine.authn.internal.executor.DAPAttributeMappe
r</Setting>
      <Setting Name="MatchLDAPAttribute" Type="xsd:string”>uid</Setting>
      <Setting Name="name" Type="xsd:string">DAP</Setting>
    </Setting>
  </Setting>
```

# Updating the TapEndpoint URL

For OAM/OIM integration to work you must update the OAM TapEndpoint URL you do this by performing the following steps.

1. Log in to Oracle Fusion Middleware Control using the following URL:

```
http://igdadmin.example.com/em
```

Or

```
http://k8worker1.example.com:30711/em
```

The Administration Server host and port number were in the URL on the End of Configuration screen (Writing Down Your Domain Home and Administration Server URL). The default Administration Server port number is 7101.

2. Click **WebLogic Domain**, and click **System MBean Browser**.

   In the search box, enter **SSOIntegrationMXBean**, and click **Search**. The mbean is displayed.

3. Set the value of **TapEndpointURL** to

   ```
   https://login.example.com/oam/server/dap/cred_submit
   ```

4. Click **Apply**.

# Running the Reconciliation Jobs

Run the Oracle Identity Governance domain to import the LDAP user names into the Oracle Identity Governance database.

To run the reconciliation jobs:

1. Log in to the OIM System Administration Console as the user `xelsysadm`.
2. Click **Scheduler** under **System Configuration**.
3. Enter `SSO*` in the search box.
4. Click the arrow for the **Search Scheduled Jobs** to list all the schedulers.
5. Select **SSO User Full Reconciliation**.
6. Click **Run Now** to run the job.
7. Repeat for **SSO Group Create And Update Full Reconciliation**.
8. Log in to the OIM Identity Console and verify that the user `weblogic_iam` is visible.

# Configuring OIM Workflow Notifications to be Sent by Email

OIM uses the human workflow, which is integrated with the SOA workflow. The SOA server configures email to receive the notifications that are delivered to the user mailbox. The user can accept or reject the notifications.

Both incoming and outgoing email addresses and mailboxes dedicated to the portal workflow are required for the full functionality. See Configuring Human Workflow Notification Properties in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

To configure the OIM workflow notifications:

1. Log in to the Fusion Middleware Control by using the administrators account. For example, `weblogic_iam`.
2. Expand the Target Navigation panel and navigate to **SOA** > **soa-infra (soa_server1)** service.
3. From the SOA infrastructure drop-down, select **SOA Administration** > **Workflow Properties**.
4. Set the Notification mode to **Email**. Provide the correct e-mail address for the notification service.

5. Click **Apply** and confirm when prompted.
6. Verify the changes.
7. Expand **Target Navigation**, select **User Messaging Service**, and then **usermessagingdriver-email (soa_servern)**. Each SOA Managed Server that is running will have a driver. Only one of these entries should be selected.
8. From the **User Messaging Email Driver** drop-down list, select **Email Driver Properties**.
9. Click **Create** if the email driver does not exist already.
10. Click **Test** and verify the changes.
11. Click **OK** to save the email driver configuration.
12. Restart the SOA cluster. No configuration or restart is required for OIM.

# Adding the wsm-pm Role to the Administrators Group

After you configure a new LDAP-based Authorization Provider and restart the Administration Server, add the enterprise deployment administration LDAP group (`WLSAdministrators`) as a member to the `policy.Updater` role in the `wsm-pm` application stripe.

1. Sign in to the Fusion Middleware Control by using the administrator's account. For example: `weblogic_iam`.
2. From the **WebLogic Domain** menu, select **Security**, and then **Application Roles**.
3. Select the **wsm-pm** application stripe from the Application Stripe drop-down menu.
4. Click the triangular icon next to the role name text box to search for all role names in the wsm-pm application stripe.
5. Select the row for the **policy.Updater** role to be edited.
6. Click the Application Role **Edit** icon to edit the role.
7. Click the Application Role **Add** icon on the Edit Application Role page.
8. In the Add Principal dialog box, select **Group** from the **Type** drop-down menu.
9. To search for the enterprise deployment administrators group, enter the group name `WLSAdministrators` in the **Principal Name Starts With** field and click the right arrow to start the search.
10. Select the appropriate administrators group in the search results and click **OK**.
11. Click **OK** on the Edit Application Role page.

# Adding the WebLogic Administration Group to SOA Administrators

To manage SOA using the users in the LDAP administration group 'WLSAdministrators', you should add the name of the group to the SOA Administrators group.

To add the group:

1. Sign in to the Fusion Middleware Control by using the administrator's account. For example: `weblogic`.
2. In the navigation pane, click **WebLogic Domain** and from the **Security** menu, select **Application Roles**.
3. From the drop-down list, select `soa-infra` to set the **application stripe**. Click **Search**.
4. Click **SOAAdmin**. Ensure that you see **Administrators** in the membership box.
5. Click **Edit**. The Edit page is displayed.
6. Click **Add** in the **Members** box. The **Add Principal** search box is displayed.
   a. Enter the following:
      - **Type**: `Group`
      - **Principal Name starts with**: `WLS`
   b. Click **Search**.
   c. From the Results box, select `WLSAdministrators` and click **OK**.

      You will be redirected to the **Edit** screen. Ensure that the members are **Administrators** and **WLSAdministrators**.
   d. Click **OK**.

# Adding the Oracle Access Manager Load Balancer Certificate to the Oracle Keystore Service

The Oracle Identity Governance to Business Intelligence Reports link inside of the Self Service application requires that the SSL certificate used by the load balancer be added to the Oracle Keystore Service Trusted Certificates.

To add the certificate, do the following:

1. Create a directory to hold user created keystores and certificates.

   ```
   kubectl exec -n <OIGNS> -ti <OIG_DOMAIN_NAME>-adminserver -- mkdir -p
   SHARED_CONFIG_DIR/keystores
   ```

   For example:

   ```
   kubectl exec -n oigns -ti governancedomain-adminserver -- mkdir -p /u01/
   oracle/user_projects/keystores
   ```

2. Obtain the certificate from the load balancer. You can obtain the load balancer certificate from using a browser, such as Firefox. However, the easiest way to obtain the certificate is to use the `openssl` command. The syntax of the command is as follows:

   ```
   openssl s_client -connect <LOADBALANCER>:<PORT> -showcerts </dev/null
   2>/dev/null| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
   LOADBALANCER.pem
   ```

For example:

```
openssl s_client -connect login.example.com:443 -showcerts </dev/null
2>/dev/null| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
login.example.com.pem
```

The `openssl` command saves the certificate to a file called `login.example.com.pem` in *`SHARED_CONFIG_DIR`*`/keystores`.

3. Copy the certificate to Kubernetes using the following command:

```
kubectl cp <FILENAME> <NAMESPACE>/<DOMAIN_NAME>-
adminserver:<SHARED_CONFIG_DIR>/keystores
```

For example:

```
kubectl cp login.example.com.pem oigns/$domain_name-adminserver:/u01/
oracle/user_projects/keystores/login.example.com.pem
```

4. Load the certificate into the Oracle Keystore Service using WLST.
   a. Connect to the container using the command:

```
kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash
```

   b. Connect to WLST using the following command:

```
ORACLE_HOME/oracle_common/common/bin/wlst.sh
```

   c. Connect to the Administration Server using the following command:

```
connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://
<OAM_DOMAIN_NAME>-adminserver.<OIGNS>.svc.cluster.local:30012')
```

   For example:

```
connect('weblogic','<password>','t3://governancedomain-
adminserver.oigns.svc.cluster.local:7101')
```

   d. Load the certificate using the following commands:

```
svc = getOpssService(name='KeyStoreService')
svc.importKeyStoreCertificate(appStripe='system',name='trust',password='
', keypassword='',alias='<CertificateName>',type='TrustedCertificate',
filepath='/<SHARED_CONFIG_DIR>/keystores/<LOADBALANCER>.pem')
```

   e. Synchronize the Keystore Service with the file system using the following command:

```
syncKeyStores(appStripe='system', keystoreFormat='KSS')
```

   For example:

```
connect('weblogic','password','t3://governancedomain-
adminserver.oigns.svc.cluster.local:7101')
```

```
svc = getOpssService(name='KeyStoreService')
svc.importKeyStoreCertificate(appStripe='system',name='trust',password='
', keypassword='',alias='login.example.com',type='TrustedCertificate',
filepath='/u01/oracle/user_projects/keystores/login.example.com.pem')
syncKeyStores(appStripe='system',keystoreFormat='KSS')
exit()
```

You will need to restart the domain for the changes to take effect. The default password for the Node Manager keystores is *COMMON_IAM_PASSWORD*. You will be prompted to confirm that the certificate is valid.

# Setting the Initial Server Count

When you first created the domain, you specified that only one Managed Server has to be started. This value ensured that the OIG bootstrap process was completed successfully. After you complete the configuration, you can increase the initial server count to the actual number you require.

When the domain is created, two files, namely `domain.yaml` and `domain_oim_soa.yaml` are also created. You used these files to initialize the domain in Kubernetes. After completing the initial configuration and the bootstrap process, you no longer need to use the `domain.yaml` file. The `domain_oim_soa.yaml` file will start the necessary servers.

To start the remaining servers and to ensure that so many servers continue to start in future, you need to update the `domain` file. To increase the server count, use the following command:

```
kubectl patch cluster -n <OIGNS> <OIG_DOMAIN_NAME>-${CLUSTER_NAME} --
type=merge -p '{"spec":{"replicas":<INITIAL_SERVER_COUNT>}}'
```

If you want two SOA and two OIM Managed Servers, use the following commands:

```
kubectl patch cluster -n oigns governancedomain-soa-cluster --type=merge -p
'{"spec":{"replicas":2}}'
```

```
kubectl patch cluster -n oigns governancedomain-oim-cluster --type=merge -p
'{"spec":{"replicas":2}}'
```

# Setting Challenge Questions

If you have integrated OAM and OIM, then after the environment is ready, you need to set up the challenge questions for your system users.

To set up the challenge questions, log in to Identity Self Service using the URL: https://prov.example.com/identity.

Log in with your user name and when prompted, add the challenge questions. You should set up these questions for the following users:

- `xelsysadm`
- `weblogic_iam`
- `oamadmin`

# Integrating Oracle Identity Manager with Oracle Business Intelligence Publisher

Oracle Identity Manager comes with a number of prebuilt reports that can be used to provide information about Oracle Identity and Access Management.

Oracle Identity Manager reports are classified based on the functional areas such as Access Policy Reports, Request and Approval Reports, Password Reports, and so on. It is no longer named Operational and Historical. These reports are not generated through Oracle Identity Manager but by the Oracle Business Intelligence Publisher (BIP). Oracle Identity Manager reports provide a restriction for Oracle BI Publisher.

The setup of a highly available enterprise deployment of Oracle BI Publisher is beyond the scope of this document. For more information, see Understanding the Business Intelligence Enterprise Deployment Topology in the *Enterprise Deployment Guide for Business Intelligence*.

> **Note:**
>
> During BI configuration for Oracle Identity Manager, you must configure only Business Intelligence Publisher. If you select other components during BI Publisher configuration, such as Business Intelligence Enterprise Edition and Essbase, the integration with Oracle Identity Manager may not work. See Configuring Reports in *Developing and Customizing Applications for Oracle Identity Governance*

- Creating a User to Run BI Reports
- Configuring Oracle Identity Manager to Use BI Publisher
- Assigning the BIServiceAdministrator Role to idm_report
- Storing the BI Credentials in Oracle Identity Governance
- Creating OIM and BPEL Data Sources in BIP
- Deploying Oracle Identity Governance Reports to BI
- Enable Certification Reports
- Validating the Reports
- Adding the Business Intelligence Load Balancer Certificate to Oracle Keystore Trust Service
- Restarting the IAMGovernanceDomain
  You have to restart the domain for the changes to take effect.

## Creating a User to Run BI Reports

You may ignore this section if you already have a user to run reports in your Business Intelligence domain.

If you need to create a user in your BI Publisher domain to run reports, use the following `LDIF` command to create a user in the LDAP directory.

1. Create a file called `/workdir/OIG/report_user.ldif` with the following contents:

```
dn: cn=idm_report,cn=Users,dc=example,dc=com
changetype: add
orclsamaccountname: idm_report
givenname: idm_report
sn: idm_report
userpassword: <password>
mail: idm_report
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetorgperson
objectclass: orcluser
objectclass: orcluserV2
uid: idm_report
cn: idm_report
```

2. Save the file.
3. Load the file into the LDAP directory using the following command:

```
ldapmodify -D cn=oudadmin -h idstore.example.com -p 1389 report_user.ldif
```

## Configuring Oracle Identity Manager to Use BI Publisher

You can set up Oracle BI Publisher to generate Oracle Identity Manager reports.

To configure Oracle Identity Manager to use the BI Publisher:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control using the URL:

   `http://igdadmin.example.com/em`
2. Click WebLogic Domain, and then select **System MBean Browser**.
3. Enter `XMLConfig.DiscoveryConfig` as the search criteria and click **Search**.

   The XMLConfig.DiscoveryConfig MBean is displayed.
4. Update the value of the **Discovery Config BI publisher URL** to the BIP URL. For example, `http://bi.example.com`
5. Click **Apply**.

## Assigning the BIServiceAdministrator Role to idm_report

If you are using LDAP as your identity store in the Business Intelligence (BI) domain, you must have created an LDAP authenticator in the BI domain. You can view the user and group names stored within LDAP.

The Oracle Identity Manager (OIM) system administration account (for example, `idm_report`) needs to be assigned the `BIServiceAdministrator` role, to generate reports.

To assign this role:

1. Ensure that the OIM administrator user is visible in the domain by logging in to the BI publisher WebLogic Console using the following URL:

   `http://biadmin.example.com/console`

2. Click **Security Realms**, and then click **myrealm**.
3. Go to the **Users and Groups** tab.
4. Look at the list of users and ensure that the user OIM Administration User (`idm_report`) is in the list of users.
5. Sign in to the BI Fusion Middleware Control by using the URL `http://biadmin.example.com/em` and the administrator's account. For example: *weblogic_bi*.
6. From the **WebLogic Domain** menu, select **Security**, and then **Application Roles**.
7. From the **Application Stripe** drop-down list, select **obi**.
8. Click the triangular icon next to the role name text box to search for all role names in the **obi** application stripe.
9. Select the row for the **BIServiceAdministrator** role to edit.
10. Click the Application Role Edit icon to edit the role.
11. Click the Application Role Add icon on the Edit Application Role page.
12. In the Add Principal dialog box, select **User** from the **Type** drop-down menu.
13. To search for the `idm_report` user, enter the user name `idm_report` in the **Principal Name Starts With** field and click the right arrow to start the search.
14. Select the appropriate user in the search results and click **OK**.
15. Click **OK** on the Edit Application Role page.

## Storing the BI Credentials in Oracle Identity Governance

To configure BIP credentials in Oracle Identity Manager:

1. Log in to the Oracle Enterprise Manager using the url

   `http://igdadmin.example.com/em`
2. In the left pane, expand the **Weblogic Domain**. The domain name is displayed.
3. Right-click the domain name, and navigate to **Security**, and then **Credentials**. A list of maps in the credential store, including the oim map, is displayed.
4. Expand the oim map. A list of entries of type Password is displayed.
5. Edit the `BIPWSKey` key if it already exists, or create a new one with the following values:

**Table 19-7 Properties of a new CSF entry**

| Attribute | Value |
|---|---|
| Select Map | oim |
| Key | BIPWSKey |
| Type | Password |
| Username | `idm_report` |
| Password | `idm_report` password |
| Description | Login credentials for BI Publisher web service |

# Creating OIM and BPEL Data Sources in BIP

### Create OIM Datasource

Oracle BIP must be connected to the OIM and SOA database schemas to run a report.

In order to do this you need to create BIP datasources using the following procedure:

1. Login to the BI Publisher Home page using the URL `https://bi.example.com/xmlpserver`
2. Click the **Administration** link on the top of the BI Publisher Home page. The BI Publisher Administration page is displayed.
3. Under Data Sources, click **JDBC Connection** link. The Data Sources page is displayed.
4. In the JDBC tab, click **Add Data Source** to create a JDBC connection to your database. The Add Data Source page is displayed.
5. Enter values in the following fields:

**Table 19-8 OIM Add Data Source Attributes**

| Attributes | Value |
|---|---|
| **Data Source Name** | Specify the Oracle Identity Governance JDBC connection name. For example, OIM JDBC. |
| **Driver Type** | Select **Oracle 11g** for an 11g database and **Oracle 12c** for a 12c database |
| **Database Driver Class** | Specify a driver class to suit your database, such as `oracle.jdbc.OracleDriver` |
| **Connection String** | Specify the database connection details in the format `jdbc:oracle:thin:@HOST_NAME:PORT_NUMBER/SID`.<br>For example, `jdbc:oracle:thin:@igddbscan:1521/oim.example.com` |
| **User name** | Specify the Oracle Identity Governance database user name for example IGD_OIM |
| **Password** | Specify the Oracle Identity Governance database user password. |

6. Click **Test Connection** to verify the connection.
7. Click **Apply** to establish the connection.
8. If the connection to the database is established, a confirmation message is displayed indicating the success.
9. Click **Apply**.

In the JDBC page, you can see the newly defined Oracle Identity Governance JDBC connection in the list of JDBC data sources.

### Create BPEL Datasource

1. Login to the BI Publisher Home page using the URL `https://bi.example.com/xmlpserver`.
2. Click the **Administration** link on the **BI Publisher** home page. The **BI Publisher Administration** page is displayed.
3. Under Data Sources, click **JDBC Connection** link. The Data Sources page is displayed.
4. In the JDBC tab, click **Add Data Source** to create a JDBC connection to your database. The Add Data Source page is displayed.
5. Enter values in the following fields:

**Table 19-9 JDBC Add Data Source Attributes**

| Attributes | Value |
|---|---|
| **Data Source Name** | Specify the Oracle Identity Governance JDBC connection name. For example, BPEL JDBC. |
| **Driver Type** | Oracle 12c |
| **Database Driver Class** | Specify a driver class to suit your database, such as `oracle.jdbc.OracleDriver` |
| **Connection String** | Specify the database connection details in the format `jdbc:oracle:thin:@HOST_NAME:PORT_NUMBER/SID`.<br>For example, `jdbc:oracle:thin:@igddbscan:1521/oim.example.com` |
| **User name** | Specify the Oracle Identity Governance database user name for example IGD_SOAINFRA. |
| **Password** | Specify the Oracle Identity Governance database user password. |

6. Click **Test Connection** to verify the connection.
7. Click **Apply** to establish the connection.
8. If the connection to the database is established, a confirmation message is displayed indicating the success.
9. Click **Apply**.

In the JDBC page, you can see the newly defined Oracle Identity Governance JDBC connection in the list of JDBC data sources.

## Deploying Oracle Identity Governance Reports to BI

After **BI Publisher** is integrated with Oracle Identity Governance, you can deploy the predefined reports for using them. To deploy Oracle Identity Manager reports:

1. Copy and unzip the predefined report */u01/oracle/idm/server/reports/oim_product_BIPReports_12c.zip* located on `OIMHOST1` file to the directory *Shared_Storage_location*`/biconfig/bidata`.

> **Note:**
>
> The *Shared_Storage_Location* is defined in the *ASERVER_HOME/config/fmwconfig/bienv/core/bi-environment.xml* file.

2. Add folder level permission to the **BIServiceAdministrator** BI application role to view and run the predefined Oracle Identity Governance reports. To do so:
   - Login to Oracle BI Publisher `https://bi.example.com/xmlpserver` by using the WebLogic admin credentials.
   - Click the **Catalog** link at the top. The Oracle Identity Manager named folder under shared folders is displayed in the left pane. Select the Oracle Identity Manager named folder.
   - Click **Permissions** option under the **Tasks** window on the bottom left.
   - Click the plus sign and perform a blank search on the available role.
   - Select the **BI Service Administrator** role, and add to the right panel.
   - Click **Ok**.
3. Logout as WebLogic user.
4. Login as the Oracle Identity Manager system administrator user to **BI Publisher console**.
5. Run the Oracle Identity Manager reports.

## Enable Certification Reports

Select or deselect the **Enable Certification Reports** option to enable or disable the certification reports. To enable the generation of certification reports, after configuring the **BI Publisher** credentials and URL, perform the following:

1. Log in to the Oracle Identity Self Service using the url: `https://prov.example.com/identity`.
2. Click the **Compliance** tab.
3. Click the **Identity Certification** box.
4. Select **Certification Configuration**. The Certification Configuration page is displayed.
5. Select the **Enable Certification Reports**.
6. Click **Save**.

> **Note:**
>
> By default, the **Compliance** tab is not shown. If you want to enable compliance functionality, you must fist set the `OIGIsIdentityAuditorEnabled` property to `true` in the Sysadmin Console (located in the **Configuration Properties** section).

## Validating the Reports

We need to create the sample data source to generate reports against the sample data source.

### Creating the Sample Reports

To view an example report data without running a report against the production JDBC Data Source, generate a sample report against the sample data source. Create the sample data source before you can generate the sample reports.

- Generating Reports Against the Sample Data Source
- Generating Reports Against the Oracle Identity Manager JDBC Data Source
- Generating Reports Against the BPEL-Based JDBC Data Source

## Generating Reports Against the Sample Data Source

After you create the sample data source, you can generate sample reports against it by performing the following steps:

1. Login to Oracle BI Publisher using the url : `https://bi.example.com/xmlpserver`.
2. Click **Shared Folders**.
3. Click **Oracle Identity Manager Reports**.
4. Select **Sample Reports**.
5. Click **View** for the sample report you want to generate.
6. Select an output format for the sample report and click **View**.

The sample report is generated.

## Generating Reports Against the Oracle Identity Manager JDBC Data Source

To generate reports against the OIM JDBC data source, navigate to the Oracle Identity Manager reports by logging in to the Oracle BI Publisher, and select an output format for the report you want to generate.

To generate reports against the Oracle Identity Manager JDBC data source:

1. Log in to Oracle BI Publisher using the url :`https://bi.example.com/xmlpserver`.
2. Navigate to Oracle Identity Manager reports. To do so:
   - In the **BI Publisher** home page, under Browse or Manage, click **Catalog Folders**. Alternatively, you can click **Catalog** at the top of the page.

     The Catalog page is displayed with a tree structure on the left side of the page and the details on the right.
   - On the left pane, expand **Shared Folders**, and navigate to the Oracle Identity Manager. All the objects in the Oracle Identity Manager folder are displayed.

     You are ready to navigate to BI Publisher 12*c* and use the Oracle Identity Manager BI Publisher reports.
3. Click **View** under the report you want to generate.
4. Select an output format for the report and click **View**.

The report is generated.

## Generating Reports Against the BPEL-Based JDBC Data Source

Some reports have a secondary data source, which is BPEL-based JDBC data source. This section describes how to generate reports against the BPEL-based JDBC data source.

**Reports With Secondary Data Source**

The following four reports have a secondary data source, which connects to the BPEL database to retrieve the BPEL data:

- Task Assignment History
- Request Details
- Request Summary
- Approval Activity

These reports have a secondary data source (BPEL-based JDBC data source) called BPEL JDBC. To generate reports against the BPEL-based JDBC data source:

1. Log in to Oracle BI Publisher using the url: `https://bi.example.com/xmlpserver`.
2. Navigate to the Oracle Identity Manager reports. To do so:
   - In the **BI Publisher** home page, under Browse or Manage, click **Catalog Folders**. Alternatively, you can click **Catalog** at the top of the page.

     The catalog page is displayed with a tree structure on the left side of the page and the details on the right.
   - On the left pane, expand **Shared Folders**, and navigate to the Oracle Identity Manager. All the objects in the Oracle Identity Manager folder is displayed.

     Navigate to the BI Publisher 12*c* and use the Oracle Identity Manager BI Publisher reports.
3. Select the report you want to generate and click **Open**.
4. Select an output format for the report, and click **Apply**.

The report is generated based on the BPEL-based JDBC data source.

## Adding the Business Intelligence Load Balancer Certificate to Oracle Keystore Trust Service

The Oracle Identity Governance to Business Intelligence Reports link inside of the Self Service application requires that the SSL certificate used by the load balancer be added to the Oracle Keystore Service Trusted Certificates.

To add the certificate:

1. Create a directory to hold user created keystores and certificates.

```
kubectl exec -n <OIGNS> -ti <OIG_DOMAIN_NAME>-adminserver -- mkdir -p
SHARED_CONFIG_DIR/keystores
```

For example:

```
kubectl exec -n oigns -ti governancedomain-adminserver -- mkdir -p /u01/
oracle/user_projects/keystores
```

2. Obtain the certificate from the load balancer. You can obtain the load balancer certificate from using a browser, such as Firefox. However, the easiest way to obtain the certificate is to use the `openssl` command. The syntax of the command is as follows:

```
openssl s_client -connect <LOADBALANCER>:<PORT> -showcerts </dev/null
2>/dev/null| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /
workdir/OAM/LOADBALANCER.pem
```

For example:

```
openssl s_client -connect bi.example.com:443 -showcerts </dev/null 2>/dev/
null| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
>bi.example.com.pem
```

The `openssl` command saves the certificate to a file called `bi.example.com.pem` in `SHARED_CONFIG_DIR/keystores`.

3. Copy the certificate to Kubernetes using the following command:

```
kubectl cp <FILENAME> <OIGNS>/<OIG_DOMAIN_NAME>-
adminserver:<SHARED_CONFIG_DIR>/keystores
```

For example:

```
kubectl cp bi.example.com.pem oigns/$governancedomain-adminserver:/u01/
oracle/user_projects/keystores/bi.example.com.pem
```

4. Load the certificate into the Oracle Keystore Service using WLST.
   a. Connect to WLST using the following command:

   ```
   /u01/oracle/oracle_common/common/bin/wlst.sh
   ```

   b. Connect to the Administration Server using the following command:

   ```
   connect('<AdminUser>','<AdminPwd>','t3://<Adminserverhost>:<Adminserver
   port>')
   ```

   For example:

   ```
   connect('weblogic','<password>','t3://governancedomain-
   adminserver.oigns.svc.cluster.local:7101')
   ```

   c. Load the certificate using the following commands:

   ```
   svc = getOpssService(name='KeyStoreService')
   svc.importKeyStoreCertificate(appStripe='system',name='trust',password='
   ```

```
', keypassword='',alias='<CertificateName>',type='TrustedCertificate',
filepath='/<SHARED_CONFIG_DIR>/keystores/<LOADBALANCER>.pem')
```

d. Synchronize the Keystore Service with the file system using the following command:

```
syncKeyStores(appStripe='system', keystoreFormat='KSS')
```

For example:

```
connect('weblogic','password','t3://governancedomain-
adminserver.oigns.svc.cluster.local:7101')
svc = getOpssService(name='KeyStoreService')
svc.importKeyStoreCertificate(appStripe='system',name='trust',password='
', keypassword='',alias='bi.example.com',type='TrustedCertificate',
filepath='/u01/oracle/user_projects/keystores/bi.example.com.pem')
syncKeyStores(appStripe='system',keystoreFormat='KSS')
exit()
```

You will need to restart the domain for the changes to take effect. The default password for the JDK is *changeit*. The default password for the Node Manager keystores is *COMMON_IAM_PASSWORD*. You will be prompted to confirm that the certificate is valid.

## Restarting the IAMGovernanceDomain

You have to restart the domain for the changes to take effect.

You can restart by using the following commands:

```
kubectl -n <OIGNS> patch domains <OIG_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "NEVER" }]'
```

After everything is stopped, it can be restarted using the following command:

```
kubectl -n <OIGNS> patch domains <OIG_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IF_NEEDED" }]'
```

For example:

```
kubectl -n oigns patch domains governancedomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "NEVER" }]'
kubectl -n oigns patch domains governancedomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "IF_NEEDED" }]'
```

# Enabling Design Console Access

You cannot access the Design Console that is installed as part of the installation because it is inside a container and requires access to an external X Window environment.

If you want to use the Design Console, you must create a standalone installation and point it to the deployment.

The Design Console interacts with OIG using the T3 protocol. This protocol is not enabled by default.

To enable access to the Design Console:

1. Expose the OIM servers' T3 port using Ingress or NodePort.
2. Update the T3 channel inside the WebLogic Server to allow requests to a named Kubernetes worker node.
3. Add a Java switch to allow external access to the T3 port.

The following sections explain the steps to add the add the Java switch.

- Creating an Ingress Service to Expose the T3 Port
- Creating a NodePort Service to Expose the T3 Port
- Updating the T3 Channel
- Adding the Java Property to the domain_oim_soa.yaml File
- Accessing the OIG Deployment from the Design Console

## Creating an Ingress Service to Expose the T3 Port

A T3 channel is already created as part of the deployment. To can expose this T3 port using Ingress:

1. Create a file called `design-console-ingress.yaml` in the working directory, with the following contents:

```
# Load balancer type. Supported values are: NGINX
type: NGINX
# Type of Configuration Supported Values are : NONSSL,SSL
# tls: NONSSL
tls: NONSSL
# TLS secret name if the mode is SSL
secretName: dc-tls-cert

# WLS domain as backend to the load balancer
wlsDomain:
  domainUID: governancedomain
  oimClusterName: oim_cluster
  oimServerT3Port: 14002
```

2. Create the Ingress by using the following commands:

```
cd $WORKDIR/sample


helm install oig-designconsole-ingress kubernetes/design-console-ingress --
namespace oigns --values $WORKDIR/design-console-ingress.yaml
```

3. Verify that the Ingress has been created by using the following command:

```
kubectl get ingress -n oigns
```

## Creating a NodePort Service to Expose the T3 Port

A T3 channel is already created as part of the deployment. You need to create a NodePort service to interact with T3 channel.

> **Note:**
>
> T3 is exposed from only one Managed Server/Pod in the cluster at a given time.

1. Create the `/workdir/OIG/oim_t3_nodeport.yaml` file with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: <OIG_DOMAIN_NAME>-oim-t3-nodeport
  namespace: <OIGNS>
spec:
  type: NodePort
  selector:
    weblogic.clusterName: oim_cluster
    weblogic.domainUID: <OIG_DOMAIN_NAME>
    weblogic.serverName: oim_server1
  ports:
    - targetPort: 14002
      port: 14002
      nodePort: <OIG_OIM_T3_PORT_K8>
      protocol: TCP
  sessionAffinity: ClientIP
```

   For example:

```
kind: Service
apiVersion: v1
metadata:
  name: governancedomain-oim-t3-nodeport
  namespace: oigns
spec:
  type: NodePort
  selector:
    weblogic.clusterName: oim_cluster
    weblogic.domainUID: governancedomain
    weblogic.serverName: oim_server1
  ports:
    - targetPort: 14002
      port: 14002
      nodePort: 30142
      protocol: TCP
  sessionAffinity: ClientIP
```

2. Create the service using the command:

```
kubectl apply -f /workdir/OIG/oim_t3_nodeport.yaml
```

   You can check that the service is created successfully by using the following command:

```
kubectl get service -n <OIGNS>
```

For example:

```
kubectl get service -n oigns
```

The output appears as follows:

```
service/governancedomain-oim-t3-nodeport created
```

## Updating the T3 Channel

After you create the NodePort Service, you have to bind WebLogic Server to it.

To bind the service:

1. Log in to the WebLogic Console using the `http://igdadmin.example.com/console` URL.
2. Navigate to **Environment**, click **Servers**, and then select **oim_server1**.
3. Click **Protocols**, and then click **Channels**.
4. Click the default T3 channel called **T3Channel**.
5. Click **Lock and Edit**.
6. Set the **External Listen Address** to one of the Kubernetes worker node. For example: `K8WORKER1.example.com`.
7. Set the **External Listen Port** to the Kubernetes service port you defined earlier. See Creating a NodePort Service to Expose the T3 Port. For example: `30142`.
8. Click **Save**.
9. Click **Apply Changes**.

## Adding the Java Property to the domain_oim_soa.yaml File

To add the Java property:

1. Edit the `domain_oim_soa.yaml` domain file located inside the WebLogic Kubernetes Operator directory. For example: `/workdir/OIG/samples/create-oim-domain/domain-home-on-pv/output/weblogic-domains/<OIG_DOMAIN_NAME>`.
2. Locate the section which starts with the line: `- clusterName: oim_cluster`.
3. In this section, append the property `-Dweblogic.rjvm.allowUnknownHost=true` to the `USER_MEM_ARGS` value. For example: After editing, the file appears as follows:

```
- clusterName: oim_cluster
    serverService:
      precreateService: true
    serverStartState: "RUNNING"
    serverPod:
      # Instructs Kubernetes scheduler to prefer nodes for new cluster
members where there are not
      # already members of the same cluster.
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
```

```
                podAffinityTerm:
                  labelSelector:
                    matchExpressions:
                      - key: "weblogic.clusterName"
                        operator: In
                        values:
                          - $(CLUSTER_NAME)
                  topologyKey: "kubernetes.io/hostname"
        env:
        - name: USER_MEM_ARGS
          value: "-Djava.security.egd=file:/dev/./urandom -Xms4096m -
Xmx8192m -Dweblogic.rjvm.allowUnknownHost=true"
      replicas: 2
```

4. Save the file.
5. Apply the changes using the following command:

```
kubectl apply -f domain_oim_soa.yaml
```

6. Restart the domain for the changes to take effect.

## Accessing the OIG Deployment from the Design Console

After you have performed a standalone deployment of the Design Console, you can access the console by using the following URL:

```
http://K8WORKER1.example.com:30142/
```

> **Note:**
>
> The http protocol is used rather than the usual T3 protocol because you are using a WebLogic Channel.

# Centralized Monitoring Using Grafana and Prometheus

If you are using a centralized Prometheus and Grafana deployment to monitor your infrastructure, you can send Oracle Identity Governance data to this application. If you have not yet deployed Prometheus and Grafana, see Installing Prometheus and Grafana.
To use the centralized Prometheus and Grafana for monitoring your infrastructure, perform the following steps:

- Downloading and Compiling the Monitoring Application
- Deploying the Monitoring Application into the WebLogic Domain
- Configuring the Prometheus Operator
- Discovering the Prometheus Service

## Downloading and Compiling the Monitoring Application

To download and configure the monitoring application for the Oracle Identity Governance cluster:

1. Change the directory to the sample scripts which set up monitoring:

   ```
   cd <WORKDIR>/samples/monitoring-service/scripts
   ```

   For example:

   ```
   cd /workdir/OAM/samples/monitoring-service/scripts
   ```

2. Run the `get-wls-exporter.sh` script.

   Before you run the script, set the environment variables that determine your environment setup:

   ```
   export adminServerPort=<OIG_ADMIN_PORT>


   export wlsMonitoringExporterTosoaCluster=true


   export soaManagedServerPort=8001


   export wlsMonitoringExporterTooimCluster=true


   export oimManagedServerPort=14100


   export domainNamespace=<OIGNS>


   export domainUID=<OIG_DOMAIN_NAME>


   export weblogicCredentialsSecretName=<OIG_DOMAIN_NAME>-credentials
   ```

For example:

```
export adminServerPort=7101


export wlsMonitoringExporterTosoaCluster=true


export soaManagedServerPort=8001


export wlsMonitoringExporterTooimCluster=true


export oimManagedServerPort=14100


export domainNamespace=oigns


export domainUID=governancedomain


export weblogicCredentialsSecretName=governancedomain-credentials
```

Execute the script using the following command:

```
./get-wls-exporter.sh
```

The output appears as follows:

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:--
--:--:--     0
100 2196k  100 2196k    0     0  1138k      0  0:00:01  0:00:01 --:--:--
6365k
created /home/opc/workdir/OIG/samples/monitoring-service/scripts/wls-
exporter-deploy dir
adminServerName is empty, setting to default "AdminServer"
soaClusterName is empty, setting to default "soa_cluster"
oimClusterName is empty, setting to default "oim_cluster"
created /tmp/ci-rsqE8LWMAw
/tmp/ci-rsqE8LWMAw ~/workdir/OIG/samples/monitoring-service/scripts
in temp dir
  adding: WEB-INF/weblogic.xml (deflated 61%)
  adding: config.yml (deflated 60%)
~/workdir/OIG/samples/monitoring-service/scripts
created /tmp/ci-xD6iyUUBut
/tmp/ci-xD6iyUUBut ~/workdir/OIG/samples/monitoring-service/scripts
in temp dir
  adding: WEB-INF/weblogic.xml (deflated 61%)
```

```
  adding: config.yml (deflated 60%)
~/workdir/OIG/samples/monitoring-service/scripts
created /tmp/ci-l0Caci13DM
/tmp/ci-l0Caci13DM ~/workdir/OIG/samples/monitoring-service/scripts
in temp dir
  adding: WEB-INF/weblogic.xml (deflated 61%)
  adding: config.yml (deflated 60%)
~/workdir/OIG/samples/monitoring-service/scripts
```

# Deploying the Monitoring Application into the WebLogic Domain

The earlier section created a number of WAR files containing the monitoring application. See Downloading and Compiling the Monitoring Application. These files need to be deployed inside the WebLogic domain. Oracle provides a script to deploy the files. Before you run the script, copy the files to the container containing the WebLogic Administration Server.

To deploy the application:

1. Change directory to the sample file location:

   ```
   cd <WORKDIR>/samples/monitoring-service/scripts
   ```

   For example:

   ```
   cd /workdir/OIG/samples/monitoring-service/scripts
   ```

2. Copy files to the Administration Server container by using the following commands:

   ```
   kubectl cp <WORKDIR>/samples/monitoring-service/scripts/wls-exporter-
   deploy  <OIGNS>/<OIG_DOMAIN_NAME>-adminserver:/u01/oracle
   ```

   ```
   kubectl cp <WORKDIR>/samples/monitoring-service/scripts/deploy-weblogic-
   monitoring-exporter.py  <OIGNS>/<OIG_DOMAIN_NAME>-adminserver:/u01/oracle/
   wls-exporter-deploy
   ```

   For example:

   ```
   kubectl cp /workdir/OIG/samples/monitoring-service/scripts/wls-exporter-
   deploy  oigns/governancedomain -adminserver:/u01/oracle
   ```

   ```
   kubectl cp /workdir/OIG/samples/monitoring-service/scripts/deploy-weblogic-
   monitoring-exporter.py  oigns/governancedomain-adminserver:/u01/oracle/wls-
   exporter-deploy
   ```

3. Deploy the application using the following command:

   ```
   kubectl exec -it -n <OIGNS> <OIG_DOMAIN_NAME>-adminserver -- /u01/oracle/
   oracle_common/common/bin/wlst.sh \
   -domainName <OIG_DOMAIN_NAME> \
   -adminServerName AdminServer \
   -adminURL <OIG_DOMAIN_NAME>-adminserver:<OIG_ADMIN_PORT> \
   -username <OIG_WEBLOGIC_USER> \
   ```

```
-password <OIG_WEBLOGIC_PWD> \
-oimClusterName oim_cluster \
-wlsMonitoringExporterTooimCluster true \
-soaClusterName soa_cluster \
-wlsMonitoringExporterTosoaCluster true
```

For example:

```
kubectl exec -it -n oigns governancedomain-adminserver -- /u01/oracle/
oracle_common/common/bin/wlst.sh \
-domainName governancedomain \
-adminServerName AdminServer \
-adminURL accessdomain-adminserver:7101 \
-username weblogic \
-password MyPassword \
-oimClusterName oim_cluster \
-wlsMonitoringExporterTooimCluster true \
-soaClusterName soa_cluster \
-wlsMonitoringExporterTosoaCluster true
```

The output appears as follows:

```
Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

Connecting to t3://governancedomain-adminserver:7101 with userid
weblogic ...
Successfully connected to Admin Server "AdminServer" that belongs to
domain "governancedomain".

Warning: An insecure protocol was used to connect to the server.
To ensure on-the-wire security, the SSL port or Admin port should be used
instead.

Deploying .........
Deploying application from /u01/oracle/wls-exporter-deploy/wls-exporter-
adminserver.war to targets AdminServer (upload=true) ...
<Aug 22, 2022 10:52:21 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating deploy operation for application, wls-exporter-adminserver
[archive: /u01/oracle/wls-exporter-deploy/wls-exporter-adminserver.war],
to AdminServer .>
.Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
Starting application wls-exporter-adminserver.
<Aug 22, 2022 10:52:29 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, wls-exporter-adminserver
[archive: null], to AdminServer .>
.Completed the start of Application with status completed
Current Status of your Deployment:
```

```
Deployment command type: start
Deployment State : completed
Deployment Message : no message
Deploying .........
Deploying application from /u01/oracle/wls-exporter-deploy/wls-exporter-
soa.war to targets soa_cluster (upload=true) ...
<Aug 22, 2022 10:52:32 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating deploy operation for application, wls-exporter-soa
[archive: /u01/oracle/wls-exporter-deploy/wls-exporter-soa.war], to
soa_cluster .>
..Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
Starting application wls-exporter-soa.
<Aug 22, 2022 10:52:42 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, wls-exporter-soa [archive:
null], to soa_cluster .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State : completed
Deployment Message : no message
Deploying .........
Deploying application from /u01/oracle/wls-exporter-deploy/wls-exporter-
oim.war to targets oim_cluster (upload=true) ...
<Aug 22, 2022 10:52:45 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating deploy operation for application, wls-exporter-oim
[archive: /u01/oracle/wls-exporter-deploy/wls-exporter-oim.war], to
oim_cluster .>
.Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
Starting application wls-exporter-oim.
<Aug 22, 2022 10:52:52 AM GMT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, wls-exporter-oim [archive:
null], to oim_cluster .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State : completed
Deployment Message : no message
Disconnected from weblogic server: AdminServer


Exiting WebLogic Scripting Tool.

<Aug 22, 2022 10:52:55 AM GMT> <Warning> <JNDI> <BEA-050001>
<WLContext.close() was called in a different thread than the one in which
it was created.>
```

# Configuring the Prometheus Operator

Prometheus enables you to collect metrics from the WebLogic Monitoring Exporter. The Prometheus Operator identifies the targets by using service discovery. To get the WebLogic Monitoring Exporter end point discovered as a target, you must create a service monitor that points to the service.

The exporting of metrics from `wls-exporter` requires `basicAuth`. Therefore, a Kubernetes secret is created with the user name and password that are `base64` encoded. This secret is used in the `ServiceMonitor` deployment. The `wls-exporter-ServiceMonitor.yaml` file has `basicAuth` with credentials as username: <*OIG_WEBLOGIC_USER*> and password: <*OIG_WEBLOGIC_PWD*> in `base64` encoded.

1. Run the following command to get the `base64` encoded version of the `weblogic` username:

   ```
   echo -n "weblogic" | base64
   ```

   The output appears as follows:

   ```
   d2VibG9naWM=
   ```

2. Run the following command to get the `base64` encoded version of the `weblogic` password:

   ```
   echo -n "<OIG_WEBLOGIC_PWD>" | base64
   ```

   The output appears as follows:

   ```
   V2VsY29tZTE=
   ```

3. Copy the template file `<WORKDIR>/samples/monitoring-service/manifests/wls-exporter-ServiceMonitor.yaml.template` to `<WORKDIR>/samples/monitoring-service/manifests/wls-exporter-ServiceMonitor.yaml`.

4. Update the `<WORKDIR>/samples/monitoring-service/manifests/wls-exporter-ServiceMonitor.yaml` location and change the `user` and `password` values to the values returned in Step 2.

   Also, change the values of the following to match the OAM namespace, domain name, and Prometheus release name. For example:

   - `namespace: oigns`
   - `weblogic.domainName: governancedomain`
   - `release: kube-prometheus`
     You can get the release name by using the command:

     ```
     kubectl get prometheuses.monitoring.coreos.com --all-namespaces -o
     jsonpath="{.items[*].spec.serviceMonitorSelector}"
     ```

   For example:

   ```
   apiVersion: v1
   kind: Secret
   metadata:
   ```

```
  name: basic-auth
  namespace: oigns
data:
  password: V2VsY29tZTE=
  user: d2VibG9naWM=
type: Opaque
---
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: wls-exporter
  namespace: oigns
  labels:
    k8s-app: wls-exporter
    release: monitoring
spec:
  namespaceSelector:
    matchNames:
    - oigns
  selector:
    matchLabels:
      weblogic.domainName: governancedomain
  endpoints:
  - basicAuth:
      password:
        name: basic-auth
        key: password
      username:
        name: basic-auth
        key: user
    port: default
    relabelings:
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
    interval: 10s
    honorLabels: true
    path: /wls-exporter/metrics
```

5. Update the `<WORKDIR>/samples/monitoring-service/manifests/prometheus-roleSpecific-domain-namespace.yaml` and `<WORKDIR>/samples/monitoring-service/manifests/prometheus-roleBinding-domain-namespace.yaml` files and change the namespace to match the OIG namespace. For example:

   **prometheus-roleSpecific-domain-namespace.yaml**

```
 apiVersion: rbac.authorization.k8s.io/v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: Role
  metadata:
    name: prometheus-k8s
    namespace: oigns
  rules:
  - apiGroups:
    - ""
    resources:
```

```
    - services
    - endpoints
    - pods
    verbs:
    - get
    - list
    - watch
kind: RoleList
```

**prometheus-roleBinding-domain-namespace.yaml**

```
apiVersion: rbac.authorization.k8s.io/v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    name: prometheus-k8s
    namespace: oigns
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: prometheus-k8s
  subjects:
  - kind: ServiceAccount
    name: prometheus-k8s
    namespace: monitoring
kind: RoleBindingList
```

6. Run the following command to enable Prometheus:

```
kubectl apply -f
```

The output appears as follows:

```
rolebinding.rbac.authorization.k8s.io/prometheus-k8s created
role.rbac.authorization.k8s.io/prometheus-k8s created
secret/basic-auth created
servicemonitor.monitoring.coreos.com/wls-exporter created
```

# Discovering the Prometheus Service

After you deploy `ServiceMonitor`, `wls-exporter` is discovered by Prometheus and is able to collect the metrics.

1. Access the following URL to view the Prometheus service discovery:

   `http://<K8_WORKER1>:32101/service-discovery`

2. Click `<OIGNS>/wls-exporter/0`, and then click **Show More**. Verify that all the targets are listed.

# Centralized Log File Monitoring Using Elasticsearch and Kibana

If you are using Elasticsearch and Kibana, you can configure a Logstash pod to send the log files to the centralized Elasticsearch/Kibana console. Before you configure the Logstash pod, ensure that you have access to a centralized Elasticsearch deployment.

- OIG persistent volume, so it can be loaded by the Logstash pod to hunt for log files.
- The location of the log files in the persistent volumes.
- The location of the centralized Elasticsearch.

To configure the Logstash pod, perform the following steps. The assumption is that you have an Elasticsearch running inside the Kubernetes cluster, in a namespace called `elkns`.

- Creating a Secret for Elasticsearch
- Creating a Configuration Map for ELK Certificate
- Creating a Configuration Map for Logstash
- Creating a Logstash Deployment

## Creating a Secret for Elasticsearch

Logstash requires credentials to connect to the elasticsearch deployment. These credentials are stored in Kubernetes as a secret.

If your Elasticsearch uses an API key for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OIGNS> --from-
literal password=<ELK_APIKEY>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oigns --from-
literal password=afshfashfkahf5f
```

If Elasticsearch uses a user name and password for authentication, then use the following command:

```
kubectl create secret generic elasticsearch-pw-elastic -n <OIGNS> --from-
literal password=<ELK_PWD>
```

For example:

```
kubectl create secret generic elasticsearch-pw-elastic -n oigns --from-
literal password=mypassword
```

You can find the Elasticsearch password using the following command:

```
kubectl get secret elasticsearch-es-elastic-user -n <ELKNS> -o go-
template='{{.data.elastic | base64decode}}'
```

# Creating a Configuration Map for ELK Certificate

If you have configured a production ready Elasticsearch deployment, you would have configured SSL. Logstash needs to trust the Elasticsearch certificate to be able to communicate with it. To enable this trust, you should create a configuration map with the contents of the Elasticsearch certificate.

You would have already saved the Elasticsearch self-signed certificate. See Copying the Elasticsearch Certificate. If you have a production certificate you can use that instead.

- Create the configuration map using the certificate, run the following command:

```
kubectl create configmap elk-cert --from-file=<WORKDIR>/ELK/elk.crt -n
<OIGNS>
```

For example:

```
kubectl create configmap elk-cert --from-file=/workdir/ELK/elk.crt -n oigns
```

# Creating a Configuration Map for Logstash

Logstash looks for log files in the OAM installations and sends them to the centralized Elasticsearch. The configuration map is used to instruct Logstash where the log files reside and where to send them.

1. Create a file called `<WORKDIR>/OIG/logstash_cm.yaml` with the following contents:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: oig-logstash-configmap
  namespace: <OIGNS>
data:
  logstash.yaml: |
  #http.host: "0.0.0.0"
  logstash-config.conf: |
    input {
      file {
        path => "/u01/oracle/user_projects/domains/logs/governancedomain/
AdminServer*.log"
        tags => "Adminserver_log"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/logs/governancedomain/
soa_server*.log"
        tags => "soaserver_log"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/logs/governancedomain/
oim_server*.log"
        tags => "Oimserver_log"
```

```
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/governancedomain/
servers/AdminServer/logs/AdminServer-diagnostic.log"
        tags => "Adminserver_diagnostic"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/governancedomain/
servers/**/logs/soa_server*-diagnostic.log"
        tags => "Soa_diagnostic"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/governancedomain/
servers/**/logs/oim_server*-diagnostic.log"
        tags => "Oimserver_diagnostic"
        start_position => beginning
      }
      file {
        path => "/u01/oracle/user_projects/domains/governancedomain/
servers/**/logs/access*.log"
        tags => "Access_logs"
        start_position => beginning
      }
    }
    filter {
      grok {
        match => [ "message", "<%{DATA:log_timestamp}> <%{WORD:log_level}>
<%{WORD:thread}> <%{HOSTNAME:hostname}> <%{HOSTNAME:servername}> <%
{DATA:timer}> <<%{DATA:kernel}>> <> <%{DATA:uuid}> <%{NUMBER:timestamp}> <%
{DATA:misc}> <%{DATA:log_number}> <%{DATA:log_message}>" ]
      }
    if "_grokparsefailure" in [tags] {
        mutate {
            remove_tag => [ "_grokparsefailure" ]
        }
    }
    }
    output {
      elasticsearch {
        hosts => ["<ELK_HOST>"]
        cacert => '/usr/share/logstash/config/certs/elk.crt'
        index => "oiglogs-000001"
        ssl => true
        ssl_certificate_verification => false
        user => "<ELK_USER>"
        index => "oiglogs-000001"
        password => "${ELASTICSEARCH_PASSWORD}"
      }
    }
```

2. Save the file.

3. Create the configuration map using the following command:

```
kubectl create -f <WORKDIR>/OIG/logstash_cm.yaml
```

For example:

```
kubectl create -f /workdir/OIG/logstash_cm.yaml
```

4. Validate that the configuration map has been created by using the following command:

```
kubectl get cm -n <OIGNS>
```

You should see `oig-logstash-configmap` in the list of configuration maps.

## Creating a Logstash Deployment

After you create the configuration map, you can create the Logstash deployment. This deployment resides in the OAM namespace.

1. Determine the mount point of the OIG persistent volume by using the following command:

```
kubectl describe domains <OIG_DOMAIN_NAME> -n <OIGNS> | grep "Mount Path"
```

For example:

```
kubectl describe domains governancedomain -n oigns | grep "Mount Path"
```

Make a note of this value. You will require it for the file that is created in the next step.

2. Create a file called `<WORKDIR>/OIG/logstash.yaml` with the following contents:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oig-logstash
  namespace: <OIGNS>
spec:
  selector:
    matchLabels:
      k8s-app: logstash
  template: # create pods using pod definition in this template
    metadata:
      labels:
        k8s-app: logstash
    spec:
      imagePullSecrets:
      - name: dockercred
      containers:
      - command:
        - logstash
        image: logstash:<ELK_VER>
        imagePullPolicy: IfNotPresent
        name: oig-logstash
```

```
    env:
    - name: ELASTICSEARCH_PASSWORD
      valueFrom:
        secretKeyRef:
          name: elasticsearch-pw-elastic
          key: password
    ports:
    - containerPort: 5044
      name: logstash
    volumeMounts:
    - mountPath: <MOUNT_PATH>
      name: weblogic-domain-storage-volume
    - name: shared-logs
      mountPath: /shared-logs
    - mountPath: /usr/share/logstash/pipeline/
      name: oig-logstash-pipeline
    - mountPath: /usr/share/logstash/config/certs
      name: elk-cert
  volumes:
  - configMap:
      defaultMode: 420
      items:
      - key: logstash-config.conf
        path: logstash-config.conf
      name: oig-logstash-configmap
    name: oig-logstash-pipeline
  - configMap:
      defaultMode: 420
      items:
      - key: ca.crt
        path: elk.crt
      name: elk-cert
  - name: weblogic-domain-storage-volume
    persistentVolumeClaim:
      claimName: <OIG_DOMAIN_NAME>-domain-pvc
  - name: shared-logs
    emptyDir: {}
```

> **Note:**
>
> If you are using your own registry, include the registry name in the image tag. If you have created a `regcred` secret for your registry, replace the `imagePullSecrets` name with the secret name you created. For example: `regcred`.

3. Save the file.
4. Create the Logstash deployment by using the following command:

```
kubectl create -f <WORKDIR>/OIG/logstash.yaml
```

For example:

```
kubectl create -f /workdir/OIG/logstash.yaml
```

5. You can now create a pod called `logstash` by using the following command:

```
kubectl get pod -n oigns
```

Your logs will now be available in the Kibana console.

# Backing Up the Configuration

As a best practice, Oracle recommends you to back up the configuration after you have successfully extended a domain or at another logical point. Back up only after you have verified that the installation is successful so far. This is a quick backup to enable immediate restoration in case of problems in later steps.

In a Kubernetes environment, it is sufficient to back up the persistent volume and the database.

The backup destination is the local disk. You can discard this backup when the enterprise deployment setup is complete. After the enterprise deployment setup is complete, you can initiate the regular deployment-specific Backup and Recovery process.

For information about backing up your configuration, see Performing Backups and Recoveries for an Enterprise Deployment.

# Running the OIM Bulkload Utility from a Container

If you want to run the `oimbulkload` utility from a container, create a new container image based on the Oracle Database Instant Client which also has a JDK and the `oimbulkload` utility installed.

Before you begin you must download a Java Development Kit RPM image.

This section includes the following topics:

- Creating a Working Directory
- Obtaining JDK Release 8
- Compiling the wlfullclient jar File in the Container
- Copying the Bulkload Directory from the OIG Container
- Creating a Dockerfile
- Checking the Working Directory
- Building the Image
- Starting the Image

## Creating a Working Directory

Create a working directory to hold all of the objects you need to build the image.

For example:

```
mkdir -p /workdir/bulkload
```

## Obtaining JDK Release 8

To obtain a copy of JDK release 8:

1. Download the RPM for java JDK release 8 from Java SE 8 Archive Downloads page.
2. Copy the downloaded JDK to your working directory `/workdir/bulkload`.

## Compiling the wlfullclient jar File in the Container

The `oimbulkload` utility is dependent on the `wlfullclient.jar` file. You should generate this file inside the OIG Administration Server image by using the following command:

```
kubectl exec -it -n <OIGNS> <OIG_DOMAIN_NAME>-adminserver -- bash -c 'cd /u01/
oracle/wlserver/server/lib ; java -jar wljarbuilder.jar'
```

For example:

```
kubectl exec -it -n oigns governancedomain-adminserver -- bash -c 'cd /u01/
oracle/wlserver/server/lib ; java -jar wljarbuilder.jar'
```

This command creates a file in the image, called `wlfullclient.jar`.

> **Note:**
>
> This file exists only until the adminserver pod is restarted. It is not required after you create the bulk load image.

## Copying the Bulkload Directory from the OIG Container

The `oimbulkload` utility is made up of a number of files in the Oracle container image. You should copy these files to your work directory, in a subdirectory called `u01`, by using the following commands:

```
export MW_HOME=/u01/oracle

kubectl exec -it -n oigns governancedomain-adminserver -- bash -c 'cd /u01/
oracle/wlserver/server/lib ; java -jar wljarbuilder.jar'
kubectl cp oigns/governancedomain-adminserver:/u01/oracle/idm/server/db/oim/
oracle/Utilities/oimbulkload u01/oracle/idm/server/db/oim/oracle/Utilities/
oimbulkload
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/oracle_common/modules/
javax.management.j2ee.jar u01/oracle/oracle_common/modules/
javax.management.j2ee.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/wlserver/modules/
com.bea.core.diagnostics.flightrecorder.jar u01/oracle/wlserver/modules/
com.bea.core.diagnostics.flightrecorder.jar
```

```
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/wlserver/modules/
com.oracle.weblogic.rjvm.jar u01/oracle/wlserver/modules/
com.oracle.weblogic.rjvm.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/wlserver/modules/
com.oracle.weblogic.security.crypto.utils.jar u01/oracle/wlserver/modules/
com.oracle.weblogic.security.crypto.utils.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/oracle_common/modules/
clients/com.oracle.webservices.wls.jaxws-owsm-client.jar u01/oracle/
oracle_common/modules/clients/com.oracle.webservices.wls.jaxws-owsm-client.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/idm/server/idmdf/idmdf-
common.jar u01/oracle/idm/server/idmdf/idmdf-common.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/idm/server/idmdf/event-
recording-client.jar u01/oracle/idm/server/idmdf/event-recording-client.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/oracle_common/modules/
thirdparty/spring-context-5.1.3.RELEASE.jar u01/oracle/oracle_common/modules/
thirdparty/spring-context-5.1.3.RELEASE.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/idm/server/client/
oimclient.jar u01/oracle/idm/server/client/oimclient.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/oracle_common/modules/
oracle.jrf/jrf-api.jar u01/oracle/oracle_common/modules/oracle.jrf/jrf-api.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/oracle_common/modules/
org.apache.commons.logging_1.2.jar u01/oracle/oracle_common/modules/
org.apache.commons.logging_1.2.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/wlserver/server/lib/
wlthint3client.jar u01/oracle/wlserver/server/lib/wlthint3client.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/wlserver/server/lib/
wlfullclient.jar u01/oracle/wlserver/server/lib/wlfullclient.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/idm/server/apps/
oim.ear/APP-INF/lib/OIMServer.jar u01/oracle/idm/server/apps/oim.ear/APP-
INF/lib/OIMServer.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/idm/server/apps/
oim.ear/APP-INF/lib/iam-platform-utils.jar u01/oracle/idm/server/apps/oim.ear/
APP-INF/lib/iam-platform-utils.jar
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/idm/server/config u01/
oracle/idm/server/config
kubectl cp oigns/governancedomain-adminserver:$MW_HOME/oracle_common/modules/
thirdparty/spring-core-5.1.3.RELEASE.jar u01/oracle/oracle_common/modules/
thirdparty/spring-core-5.1.3.RELEASE.jar
chmod +x u01/oracle/idm/server/db/oim/oracle/Utilities/oimbulkload/scripts/
*.sh
```

> **Note:**
>
> There is no '/' in front of `u01` by design.

## Creating a Dockerfile

A `Dockerfile` is used to determine how the image is built. This file resides in the work directory regardless of whether you are using Docker or podman to build the image. It has the following contents:

```
FROM  ghcr.io/oracle/oraclelinux7-instantclient:21
```

```
ADD /jdk-8u202-linux-x64.rpm /jdk-8u202-linux-x64.rpm
RUN yum install -y https://yum.oracle.com/repo/OracleLinux/OL7/oracle/
instantclient21/x86_64/getPackage/oracle-instantclient-
{basic,tools,jdbc}-21.5.0.0.0-1.x86_64.rpm jdk-8u202-linux-x64.rpm tar
RUN mkdir -p /usr/lib/oracle/21/client64/rdbms /usr/lib/oracle/21/client64/
jdbc
RUN cp -r /usr/lib/oracle/21/client64/lib /usr/lib/oracle/21/client64/jdbc

COPY u01 ./u01
ENV PATH=$PATH:/usr/lib/oracle/21/client64/bin
ENV JAVA_HOME=/usr
ENV MW_HOME=/u01/oracle
ENV OIM_ORACLE_HOME=/u01/oracle/idm
ENV ORACLE_HOME=/usr/lib/oracle/21/client64
```

Save the file.

## Checking the Working Directory

At the end of this process, you will have the following files in the work directory:

- `jdk-8u202-linux-x64.rpm`
- `u01` with various subdirectories
- `Dockerfile`

## Building the Image

To build the image:

1. Use the following command:

   ```
   podman build -t <REGISTRY>/database/bulkload:latest -f Dockerfile
   ```

   Or

   ```
   docker build -t <REGISTRY>/database/bulkload:latest -f Dockerfile
   ```

   For example:

   ```
   podman build -t iad.ocir.io/mytenancy/database/bulkload:latest -f
   Dockerfile
   ```

   This command:

   - Pull the latest database instant client container from GitHub.
   - Extends the image with the instant client tools.
   - Modifies the structure of the container so it looks like a database home directory
   - Installs the JDK.
   - Copies the `oimbulkload` utility and its dependencies into the image.

- Sets up the default environment variables in the image for *JAVA_HOME*, *MW_HOME*, *OIM_ORACLE_HOME*, and *ORACLE_HOME*.

2. After you have created the image, view it in your local repository tagged with your container registry, using the following command:

```
podman images
```

3. If required, push the image to the container registry by using the command:

```
podman  push iad.ocir.io/mytenancy/database/bulkload:latest
```

# Starting the Image

You can now start the image in Kubernetes and use it to perform bulk load activities. The following steps show how to start the image and use an NFS mounted filesystem for your csv files.

- Creating a Pod File
- Starting the Bulkload Pod
- Running the Bulkload Utility

## Creating a Pod File

Create a file called `bulkload.yaml` with the following contents:

```
apiVersion: v1
kind: Pod
metadata:
  name: bulkload
  namespace: <OIGNS>
  labels:
    app: dbclient
spec:
  restartPolicy: OnFailure
  volumes:
    - name: oigbulkpv
      nfs:
        server: <PVSERVER>
        path: <OID_BULK_SHARE>
  containers:
  - name: bulkload
    image: iad.ocir.io/mytenancy/database/bulkload:latest
    volumeMounts:
      - name: oigbulkpv
        mountPath: /u01/oracle/idm/server/db/oim/oracle/Utilities/oimbulkload/
csv_files
    command: ["/bin/bash", "-ec", "while :; do echo '.'; sleep 5 ; done"]
  imagePullSecrets:
    - name: <REGISTRY_SECRET_NAME>
```

For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: bulkload
  namespace: oigns
  labels:
    app: dbclient
spec:
  restartPolicy: OnFailure
  volumes:
    - name: oigbulkpv
      nfs:
        server: nfsserver.example.com
        path: /exports/IAMPVS/oigbulkpv
  containers:
  - name: bulkload
    iad.ocir.io/mytenancy/database/bulkload:latest
    volumeMounts:
      - name: oigbulkpv
        mountPath: /u01/oracle/idm/server/db/oim/oracle/Utilities/oimbulkload/
csv_files
    command: ["/bin/bash", "-ec", "while :; do echo '.'; sleep 5 ; done"]
  imagePullSecrets:
    - name: regcred
```

## Starting the Bulkload Pod

You can now start the bulkload pod by using the command:

```
kubectl create -f bulkload.yaml
```

When the pod has started, you see it running in the OIG namespace by using the command:

```
kubectl get pods -n oigns
```

1. Enter the text of the first step here.
2. Enter the text of the second step here.

## Running the Bulkload Utility

Before you run the `oimbulkload` utility, you must first connect to the container by using the command:

```
kubectl exec -it -n <OIGNS> bulkload -- bash
```

For example:

```
kubectl exec -it -n oigns bulkload -- bash
```

The bulk load utility is located at `/u01/oracle/idm/server/db/oim/oracle/Utilities/oimbulkload/`.

To start the `oimbulkload` utility, use the command:

```
cd /u01/oracle/idm/server/db/oim/oracle/Utilities/oimbulkload/
./oim_blkld.sh
```

When running, the tool asks you to provide certain pieces of information depending on the type of load you are using. Here is a summary of some of the information you may be asked for:

- *JAVA_HOME*=`/usr`
- *MW_HOME*=`/u01/oracle`
- *OIM_ORACLE_HOME*=`/u01/oracle/idm`
- *ORACLE_HOME*=`/usr/lib/oracle/21/client64`
- Host where the Governance Server is running = `governancedomain-cluster-oim-cluster.oigns. svc.cluster.local`
- Port where the Governance Server is running = `14000`

For instructions on running `oimbulkload` utility, see Using the Bulk Load Utility in *Developing and Customizing Applications for Oracle Identity Governance*.

20

# Installing and Configuring Oracle Identity Role Intelligence

Oracle Identity Role Intelligence (OIRI) authenticates using users and groups defined in Oracle Identity Governance. Therefore, you have to install and configure Oracle Identity Governance first.

See Configuring Oracle Identity Governance Using WDT. Unlike the traditional Oracle Identity and Access Management products, Oracle Identity Role Intelligence is deployed as a series of microservices.

In this release, Oracle uses a standalone container image to install and configure OIRI. The container image is started manually in the Kubernetes cluster.

This chapter includes the following topics:

- About Oracle Identity Role Intelligence
  OIRI is used by system administrators to perform role mining operations.
- Variables Used in this Chapter
  The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.
- Characteristics of the OIRI Installation
  This section lists the key characteristics of the OIRI installation that you are about to create. Review these characteristics to understand the purpose and context of the procedures that are used to configure OIRI.
- Kubernetes Services
  If you are using NodePort Services, the Kubernetes services '`oiri-nodeport`' and '`oiri-ui-nodeport`' are created as part of the OIRI installation. If you are using Ingress, an Ingress service will be created.
- Before You Begin
  Before you begin the installation, you have to ensure that all the required tasks listed in this topic are complete.
- Setting Up a Product Specific Work Directory
- Creating User Names and Groups in Oracle Identity Governance
  Oracle Identity Role Intelligence authenticates using users and groups in Oracle Identity Governance. Before you start configuring OIRI, log in to the OIG Self Service Console using the `https://prov.example.com/identity` URL to create the required user names and groups.
- Ensuring that OIG Compliance Mode is Enabled
  For OIRI to function, ensure that the compliance functionality is enabled for OIG.
- Creating Kubernetes Namespaces
  The Kubernetes namespaces are used to store all the OIRI objects.
- Creating Container Registry Secrets
  If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

- Creating a Kubernetes Secret for Docker Hub Images
  This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands. These commands are used by the OUD cron job to test for pods that are stuck in the 'Terminating' state, and restart them if necessary.
- Starting the Administration CLI
  Before starting the Administration CLI, ensure that you have created the persistent volumes.
- Granting the CLI Access to the Kubernetes Cluster
  The OIRI CLI container has built-in commands to interact with the Kubernetes cluster. You must provide the Administration CLI with details on how to access the Kubernetes cluster.
- Creating the Configuration Files
  OIRI uses a number of property files to deploy OIRI. These property files are populated using the CLI commands.
- Creating the OIRI Keystore
  Create a keystore in OIRI using the `keytool` command. This command should be run from the ORI-CLI container.
- Loading the OIG Certificates into OIRI
  For OIRI to trust OIG, you should load the OIG certificate into OIRI.
- Creating Wallets
  OIRI stores database/OIG connection information in a wallet. You have to create the wallet by running the command from inside the OIRI-CLI pod.
- Creating the Database Schemas
  Create the OIRI database schemas in the database by running the commands from the OIRI-CLI container.
- Verifying the Wallet
  After creating the wallet, you should validate the wallet. If the validation fails, correct the wallet before you proceed.
- Deploying OIRI Using Helm
  After creating the namespaces, you can now deploy OIRI using the generated Helm chart. You should execute the command from the OIRI-CLI.
- Verifying that OIRI is Running
  After you deploy OIRI, you should verify that it is running successfully.
- Creating the Kubernetes NodePort Services
  By default, OIRI gets created with all the components configured as `ClusterIP` services. The configuration indicates that the Oracle Identity Role Intelligence components are visible only within the Kubernetes cluster.
- Updating the OHS Configuration
  If you have not already done so, you should now add the OIRI entries to the Oracle HTTP configuration.
- Performing an Initial Data Load Using the Data Ingester
  After OIRI is up and running, you may want to perform an initial data load from the OIG database.

# About Oracle Identity Role Intelligence

OIRI is used by system administrators to perform role mining operations.

It consists of the following components:

- Oracle Identity Role Intelligence Management Container (OIRI-CLI)
- Oracle Identity Role Intelligence (OIRI) Microservice
- Oracle Identity Role Intelligence User Interface (OIRI-UI) Microservice
- Oracle Identity Role Intelligence Data Ingester (OIRI-DING) Microservice

OIRI uses a dedicated database because it can process large amount of data in a way that is different from the OAM/OIG data stores. It uses a data warehouse model rather than OnLine Transaction Processing (OLTP).

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 20-1 The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| <*REGISTRY_ADDRESS*> | `iad.ocir.io/<mytenancy>` | The location of the registry. |
| <*REGISTRY_SECRET_NAME*> | `regcred` | The name of the Kubernetes secret containing the container registry credentials. Required only if you are pulling images directly from a container registry. See Creating Container Registry Secrets. |
| <*REG_USER*> | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The name of the user you use to log in to the registry. |
| <*REG_PWD*> | <*password*> | The registry user password. |
| <*OIRI_CLI_REPOSITORY*> | `oracle/oiri-cli`<br>`local/oracle/oiri-cli`<br>`container-registry.oracle.com/middleware/oiri-cli_cpu`<br><*REGISTRY_ADDRESS*>`/oracle/oiri-cli` | The name of the OIRI CLI software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/local/oiri-cli`. If you use OLCNE, the value will be `local/oiri-cli`.<br>If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/local/oiri-cli_cpu`.<br>If you use a container registry, the value will be the name of the registry with the product name: <*REGISTRY_ADDRESS*>`/oiri-cli`. |

**Table 20-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OIRICLI_VER>* | `12.2.1.4.220429` | The version of the image you want to use. |
| *<OIRI_REPOSITORY>* | `oracle/oiri`<br>`local/oracle/oiri`<br>`container-registry.oracle.com/middleware/oiri_cpu`<br>*<REGISTRY_ADDRESS>*`/oracle/oiri` | The name of the OIRI software repository.<br>If you have downloaded and staged a container image, this value will be: `local/oracle/oiri`. If you use OLCNE, the value will be `local/oracle/oiri`.<br>If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/local/oiri_cpu`.<br>If you use a container registry, the value will be the name of the registry with the product name: *<REGISTRY_ADDRESS>*`/oracle/oiri`. |
| *<OIRI_VER>* | `12.2.1.4.02106` | The version of the OIRI image you want to use. |
| *<OIRI_DING_REPOSITORY>* | `oracle/oiri-ding`<br>`local/oracle/oiri-ding`<br>`container-registry.oracle.com/middleware/oiri-ding_cpu`<br>*<REGISTRY_ADDRESS>*`/oracle/oiri-ding` | The name of the OIRI DING software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/local/oiri-ding`. If you use OLCNE, the value will be `local/oiri-ding`.<br>If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/local/oiri-ding_cpu`.<br>If you use a container registry, the value will be the name of the registry with the product name: *<REGISTRY_ADDRESS>*`/oracle/oiri-ding`. |
| *<OIRIDING_VER>* | `12.2.1.4.02106` | The version of the image you want to use. |

**Table 20-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OIRI_UI_REPOSITORY>* | `oracle/oiri-ui`<br>`local/oracle/oiri-ui`<br>`container-registry.oracle.com/middleware/oiri-ui_cpu`<br>*<REGISTRY_ADDRESS>*`/oracle/oiri-ui` | The name of the OIRI UI software repository.<br>If you have downloaded and staged a container image, this value will be: `oracle/local/oiri-ui`. If you use OLCNE, the value will be `local/oiri-ui`.<br>If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/local/oiri-ui_cpu`.<br>If you use a container registry, the value will be the name of the registry with the product name: *<REGISTRY_ADDRESS>*`oiri-ui`. |
| *<OIRIUI_VER>* | `12.2.1.4.02106` | The version of the image you want to use. |
| *<PVSERVER>* | `1.1.1.1` | The name or IP address of the NFS server hosting the persistent volumes. |
| *<OIRINS>* | `oirins` | The name of the OIRI namespace you are using to hold the OIRI objects.. |
| *<DINGNS>* | `dingns` | The name of the OIRI DING namespace you are using to hold the DING objects. |
| *<WORKDIR>* | `/workdir/OIRI/` | The working directory for OIRI. |
| *<OIRI_SHARE>* | `/exports/IAMPVS/oiripv` | The NFS mount location for the OIRI persistent volume. |
| *<OIRI_SHARE_SIZE>* | `10Gi` | The size of the NFS share. |
| *<OIRI_DING_SHARE>* | `/exports/IAMPVS/dingpv` | The NFS mount location for the OIRI Ding persistent volume. |
| *<OIRI_DING_SHARE_SIZE>* | `10Gi` | The size of the NFS share. |
| *<OIRI_WORK_SHARE>* | `/exports/IAMPVS/workpv` | The NFS mount of the OIRI Work persistent volume. |
| *<OIRI_NFS_SHARE>* | `/exports/iampvs/oiripv` | The NFS share mount point for OIRI persistent volume. |
| *<OIG_DB_SCAN>* | `db-scan.example.com` | Database host of the OIG database. Use the SCAN address if you use a RAC database. |
| *<OIG_DB_LISTENER>* | `1521` | Listener port of the Oracle Identity Governance database. |
| *<OIG_DB_SERVICE>* | `oig_s.example.com` | Name of the database service for the OIG database. |

**Table 20-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OIRI_DB_SCAN>* | `db-scan.example.com` | Database host of the OIRI database. Use the SCAN address if using a RAC database. |
| *<OIRI_DB_SYS_PWD>* | *<password>* | The OIRI database sys password. |
| *<OIRI_DB_LISTENER>* | `1521` | Listener port of the OIRI database. |
| *<OIRI_DB_SERVICE>* | `oiri_s.example.com` | Name of the database service for the OIRI database. |
| *<OIRI_RCU_PREFIX>* | `oiri` | The database schema prefix you want to assign to the OIRI database schema objects. |
| *<OIRI_SCHEMA_PWD>* | `myoigschemapwd` | The password you want to assign to the OIRI schemas. |
| *<OIG_RCU_PREFIX>* | `IGD_OIM` | The prefix you used when you created the OIG schemas. |
| *<OIG_SCHEMA_PWD>* | `myoigschemapwd` | The password associated with the `<OIG_RCU_PREFIX>_OIM` schema. |
| *<USE_INGRESS>* | `false` | If you want the installation to create an Ingress controller in the OIRI namespace, set this value to `true`. If you are using your own Ingress controller or NodePort, set this to value to `false`. |
| *<OIRI_INGRESS_HOST>* | `oiri.example.com`<br>`igdadmin.example.com`<br>`k8workers.example.com` | Set this value to the OIRI load balancer virtual host name. If you use OIRI standalone, this value will be the name of an OIRI virtual host that you have defined specifically for OIRI. If you are deploy this host as part of an Integrated Oracle Identity and Access Management deployment (containers or otherwise), you can use the existing virtual host for your Oracle Identity Governance administration operations. |
| *<OIRI_REPLICAS>* | `2` | The number of OIRI servers to start. A minimum of two servers is required for HA. |
| *<OIRI_UI_REPLICAS>* | `2` | The number of OIRI UI servers to start. A minimum of two servers is required for HA. |
| *<OIRI_DING_REPLICAS>* | `2` | The number of DING servers to start. A minimum of two servers is required for HA. |

**Table 20-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OIRI_KEYSTORE_PASSWORD>* | `mykeystorepwd` | The password you used when you imported the OIG REST certificate. See Importing the OIG REST Certificate into OIRI. |
| *<OIRI_SERVICE_USER>* | `oirisvc` | The user name of the service you have created. See Creating the OIRI Service User. |
| *<OIRI_SERVICE_PWD>* | `myservicepwd` | The password assigned to the *<OIRI_SERVICE_USER>* account. |
| *<OIRI_K8>* | `30305` | The Kubernetes service port of the OIRI NodePort service. |
| *<OIRI_UI_K8>* | `30306` | The Kubernetes service port of the OIRI-UI NodePort Service. |
| *<CERT_FILE>* | `prov.example.com.pem` | The name of the certificate file. |
| *<K8_URL>* | `https://10.0.0.10:6443` | The URL of the Kubernetes API. You can obtain the URL by using the following command on the admin host:<br>`grep server: $KUBECONFIG \| sed 's/server://;s/ //g'` |
| *<CERTIFICATE_FILE >* | `/workdir/OIRI/ca.crt` | The location of the `ca.crt` certificate file you generated. See Generating the `ca.crt` Certificate. |
| *<OIGNS>* | `oigns` | The namespace used by OIG. |
| *<OIG_DOMAIN_NAME>* | `governancedomain` | The name of the OIG domain. |
| *<OIG_LBR_HOST>* | `prov.example.com` | The load balancer entry point for OIM. |
| *<OIG_LBR_PORT>* | `443` | The load balancer port for OIM. |
| *<OIG_URL>* | `http://igdinternal.example.com:7777`<br>`http://governancedomain-cluster-oim-clusteroigns.svc.cluster.local:14000` | This is the URL of the OIG installation. OIRI does not need to exit the organization. So it is fine to use the internal call back URL.<br>If your OIG installation is inside the same Kubernetes cluster as your OIG deployment, you can use the internal service name which looks as follows:<br>`http://$OIG_DOMAIN_NAME-cluster-oim-cluster.$OIGNS.svc.cluster.local:14000` |

**Table 20-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<ELK_HOST>* | `https://elasticsearch-es-http.elkns.svc:9200` | The host and port of the centralized Elasticsearch deployment. This host can be inside the Kubernetes cluster or external to it. This host is used only when Elasticsearch is used. |
| *<ELK_VER>* | `8.11.0` | The version of Elasticsearch you want to use. |

# Characteristics of the OIRI Installation

This section lists the key characteristics of the OIRI installation that you are about to create. Review these characteristics to understand the purpose and context of the procedures that are used to configure OIRI.

**Table 20-2 Key Characteristics of the OIRI Installation**

| Characteristics of OIRI | More Information |
|---|---|
| Each microservice is deployed into a pod in the Kubernetes cluster. | See About the Kubernetes Deployment. |
| Places the OIRI components in a dedicated Kubernetes namespace. | See About the Kubernetes Deployment. |
| Places the DING components in a dedicated namespace, although these could be combined with the OIRI namespace if desired. | See About the Kubernetes Deployment. |
| Uses the Kubernetes services to interact with microservices. | See Creating the Kubernetes Services. |
| Uses the Kubernetes persistent volumes to hold configuration information. | See #unique_614. |
| Each Kubernetes pod is built from a pre-built Oracle container image. | See Identifying and Obtaining Software Distributions for an Enterprise Deployment. |
| Requires Oracle Identity Governance to be installed and configured. | See Configuring Oracle Identity Governance Using WDT. |
| Installation can be standalone or integrated. | See #unique_721. |

# Kubernetes Services

If you are using NodePort Services, the Kubernetes services '`oiri-nodeport`' and '`oiri-ui-nodeport`' are created as part of the OIRI installation. If you are using Ingress, an Ingress service will be created.

**Table 20-3 Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| `oiri-nodeport` | `NodePort` | `30305` | `8005` |

**Table 20-3 (Cont.) Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| oiri-ui-nodeport | NodePort | 30306 | 8080 |

If you use an Ingress-based deployment, the following Ingress service will be created as part of this deployment:

**Table 20-4 Ingress Service**

| Service Name | Host Name |
|---|---|
| oiriI-ingress | igdadmin.edg.com and oiri.example.com |

# Before You Begin

Before you begin the installation, you have to ensure that all the required tasks listed in this topic are complete.

Complete and ensure that you have:

- The required Oracle container images staged on each of the Kubernetes worker nodes, or hosted in a container registry to which you have access.
  The following Oracle container images have to be staged:
  - OIRI-CLI
  - OIRI
  - OIRI-UI
  - OIRI-DING
- Configured Oracle HTTP Server as described in Installing and Configuring Oracle HTTP Server.
- Configured Oracle Unified Directory as describer in Installing and Configuring Oracle Unified Directory.
- Configured Oracle Identity Governance as described in Configuring Oracle Identity Governance Using WDT.
- Added the missing OAM policies for OIRI as described in Adding the Missing Policies to OAM.

# Setting Up a Product Specific Work Directory

Before you begin the installation, you should have downloaded and staged the Oracle Identity Governance container image and the code repository. See Downloading Images from a Container Registry and Staging the Code Repository. You must also have deployed the Oracle WebLogic Operator as described in Installing the WebLogic Kubernetes Operator
This section describes copying the downloaded sample deployment scripts to a temporary working directory on the configuration host for OIRI.

1. Create a temporary working directory as the install user. The install user should have `kubectl` access to the Kubernetes cluster.

   ```
   mkdir -p /<WORKDIR>
   ```

   For example:

   ```
   mkdir -p /workdir/OIRI
   ```

2. Change directory to this location:

   ```
   cd /workdir/OIRI
   ```

# Creating User Names and Groups in Oracle Identity Governance

Oracle Identity Role Intelligence authenticates using users and groups in Oracle Identity Governance. Before you start configuring OIRI, log in to the OIG Self Service Console using the `https://prov.example.com/identity` URL to create the required user names and groups.

Create the following users and groups:

- OIRI Service Account
- OIRI User Account
- OIRI Group
- Creating the OIRI Service User
- Creating the OIRI User
- Creating the OIRI Engineering Role

## Creating the OIRI Service User

To create the OIRI service user:

1. Log in to the OIG Self Service Console using the `https://prov.example.com/identity` URL and the system administration user. For example: `xelsysadm`.
2. Click **Manage**.
3. Click **Users**.
4. Click **Create**.
5. Specify the following information in the Create User screen. The remaining fields are optional.
   - **First Name**: For example `oirisvc`.
   - **Last Name**: For example `oirisvc`.
   - **Organization**: `Xellerate Users`.
   - **User Type**: `Full Time Employee`.
   - **User Login**: For example `oirisvc`.
   - **Password**: Choose a password for the account.

- **Confirm Password**: Repeat the password.

6. Click **Submit**.
7. From the Home screen, select **Admin Roles**.
   a. Search for the Administration Role **OrclOIMUserViewer**.
   b. Click **User**, and then click on the **Members** tab.
   c. Click **Assign Users**.
   d. Search for the newly created user. For example `oirisvc`.
   e. Select the user and select **Add Selected**.
   f. Click **Select**.
8. Repeat Step 5 for the roles **OrclOIMRoleAdministrator** and **OrclOIMAccessPolicyAdministrator**.

## Creating the OIRI User

To create the OIRI user:

1. Log in to the OIG Self Service Console using the `https://prov.example.com/identity` URL and the system administration user. For example: `xelsysadm`.
2. Click **Manage**.
3. Click **Users**.
4. Click **Create**.
5. Specify the following information in the Create User screen. The remaining fields are optional.
   - **First Name**: For example `oiri`.
   - **Last Name**: For example `oiri`.
   - **Organization**: `Xellerate Users`.
   - **User Type**: `Full Time Employee`.
   - **User Login**: For example `oiri`.
   - **Password**: Choose a password for the account.
   - **Confirm Password**: Repeat the password.
6. Click **Submit**.

## Creating the OIRI Engineering Role

To create the OIRI engineering role:

1. Log in to the OIG Self Service Console using the `https://prov.example.com/identity` URL and the system administration user. For example: `xelsysadm`.
2. Click **Roles and Access Policies - Roles**.
3. Click **Create** and specify the following information:
   - **Name**: `OrclOIRIRoleEngineer`
   - **Display Name**: `OrclOIRIRoleEngineer`

- **Role Description**: `OIRI Engineer Role`

4. Click **Next**.
5. On the Hierarchy screen, click **Next**.
6. On the Access Policy screen, click **Next**.
7. On the Add Role Membership screen, click **Add Members**.
   a. On the Add Role Membership screen, click **Add Members**.
   b. Select the oiri user and click **Add Selected**.
   c. Click **Select**.
   d. Click **Next**.
8. On the Organizations screen, click **Add Organization**.
   a. Search for the Organization Top and select it.
   b. Click **Add Selected**.
   c. Click **Select**.
   d. Click **Next**.
9. On the Summary screen, review the information that you have entered and click **Finish**.

# Ensuring that OIG Compliance Mode is Enabled

For OIRI to function, ensure that the compliance functionality is enabled for OIG.

To check whether compliance is enabled:

1. Log in to the OIG Sysadmin Console using the `http://igdadmin.example.com/sysadmin` URL.
2. Click **Configuration Properties**.
3. Search for the system property with the keyword: `OIG.IsIdentityAuditorEnabled`.
4. Edit the property and ensure that it is set to `true`. If not, amend the value to `true` and click **Save**.

# Creating Kubernetes Namespaces

The Kubernetes namespaces are used to store all the OIRI objects.

Use the following commands to create separate namespaces for OIRI and DING.

```
kubectl create namespace <OIRINS>


kubectl create namespace <DINGNS>
```

For example:

```
kubectl create namespace oirins


kubectl create namespace dingns
```

# Creating Container Registry Secrets

If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret which contains the login details of the container registry.

This step is not required if you have staged the container images locally.

To create a container registry secret for OIRI, use the following command:

```
kubectl create secret -n <OIRINS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n oirins docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

To create a container registry secret for OIRI DING, use the following command:

```
kubectl create secret -n <DINGNS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n dingns docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

# Creating a Kubernetes Secret for Docker Hub Images

This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands. These commands are used by the OUD cron job to test for pods that are stuck in the 'Terminating' state, and restart them if necessary.

You should have an account on `hub.docker.com`. If you want to stage the images in your own repository, you can do so and modify the `helm` override file as appropriate.

To create a Kubernetes secret for `hub.docker.com`, use the following command:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="<DH_USER>" --docker-
password="<DH_PWD>" --namespace=<OUDNS>
```

For example:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="username" --docker-
password="<mypassword>" --namespace=oudns
```

# Starting the Administration CLI

Before starting the Administration CLI, ensure that you have created the persistent volumes.

See Creating File Systems and Mount Targets.

To start the Administration CLI:

1. Create a file called `oiri-cli.yaml` with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: oiri-cli
  namespace: <OIRINS>
  labels:
    app: oiricli
spec:
  restartPolicy: OnFailure
  volumes:
    - name: oiripv
      nfs:
        server: <PVSERVER>
        path: <OIRI_SHARE>
    - name: dingpv
      nfs:
        server: <PVSERVER>
        path: <OIRI_DING_SHARE>
    - name: workpv
      nfs:
        server: <PVSERVER>
        path: <OIRI_WORK_SHARE>
  containers:
  - name: oiricli
    image: <OIRI_CLI_REPOSITORY>:<OIRICLI_VER>
    volumeMounts:
      - name: oiripv
        mountPath: /app/oiri
      - name: dingpv
        mountPath: /app
      - name: workpv
        mountPath: /app/k8s
    command: ["/bin/bash", "-ec", "tail -f /dev/null"]
  imagePullSecrets:
    - name: regcred
```

For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: oiri-cli
  namespace: oirins
  labels:
    app: oiricli
spec:
  restartPolicy: OnFailure
  volumes:
    - name: oiripv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/oiripv
    - name: dingpv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/dingpv
    - name: workpv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/workpv
  containers:
  - name: oiricli
    image: iad.ocir.io/mytenancy/idm/oiri-cli:12.2.1.4.220429
    volumeMounts:
      - name: oiripv
        mountPath: /app/oiri
      - name: dingpv
        mountPath: /app
      - name: workpv
        mountPath: /app/k8s
    command: ["/bin/bash", "-ec", "tail -f /dev/null"]
  imagePullSecrets:
    - name: regcred
```

2. Start the Administration CLI container using the following command:

```
kubectl create -f oiri-cli.yaml
```

3. Connect to the running container using the command:

```
kubectl exec -n oirins -ti oiri-cli /bin/bash
```

> **Note:**
>
> When the examples say "use the following command from within the OIRI-CLI", it means that you should connect to the running container as described here, and then run the commands as specified.

# Granting the CLI Access to the Kubernetes Cluster

The OIRI CLI container has built-in commands to interact with the Kubernetes cluster. You must provide the Administration CLI with details on how to access the Kubernetes cluster.

To provide access, perform the following steps on any node which has a working `kubectl` command:

- Creating a Kubernetes Service Secret
- Creating a Kubernetes Service Account
- Generating the ca.crt Certificate
- Creating a Kubernetes Configuration File for OIRI
- Copying Files to the OIRI-CLI Container
- Validating the kubectl Command

## Creating a Kubernetes Service Secret

If you are using Kubernetes 1.24 release or later, then you need to create a secret for the Kubernetes service account using the following command:

```
kubectl create -f <WORKDIR>/create_svc_secret.yaml
```

For example:

```
kubectl create -f /workdir/OIRI/create_svc_secret.yaml
```

Here, `<WORKDIR>/create_svc_secret.yaml` has the following content:

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: oiri-service-account
  namespace: <OIRINS>
  annotations:
    kubernetes.io/service-account.name: "oiri-service-account"
```

## Creating a Kubernetes Service Account

Create a `workdir/oiri/create_svc.yaml` file for the Kubernetes service account in the OIRI namespace by using the following command:

```
kubectl apply -f <WORKDIR>/create_svc.yaml
```

For example:

```
kubectl apply -f /workdir/OIRI/create_svc.yaml
```

The `create_svc.yaml` file has the following content:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: oiri-service-account
  namespace: <OIRINS>
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: oiri-ns-role
  namespace: <OIRINS>
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ding-ns-role
  namespace: <DINGNS>
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: oiri-clusterrolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:persistent-volume-provisioner
subjects:
- namespace: <OIRINS>
  kind: ServiceAccount
  name: oiri-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: oiri-clusteradmin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- namespace: <OIRINS>
  kind: ServiceAccount
  name: oiri-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
```

```
metadata:
  name: oiri-rolebinding
  namespace: <OIRINS>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: oiri-ns-role
subjects:
- namespace: <OIRINS>
  kind: ServiceAccount
  name: oiri-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ding-rolebinding
  namespace: <DINGNS>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ding-ns-role
subjects:
- namespace: <OIRINS>
  kind: ServiceAccount
  name: oiri-service-account
```

For example:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: oiri-service-account
  namespace: oirins
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: oiri-ns-role
  namespace: oirins
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ding-ns-role
  namespace: dingns
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRoleBinding
metadata:
  name: oiri-clusterrolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:persistent-volume-provisioner
subjects:
- namespace: oirins
  kind: ServiceAccount
  name: oiri-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: oiri-clusteradmin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- namespace: oirins
  kind: ServiceAccount
  name: oiri-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: oiri-rolebinding
  namespace: oirins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: oiri-ns-role
subjects:
- namespace: oirins
  kind: ServiceAccount
  name: oiri-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ding-rolebinding
  namespace: dingns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ding-ns-role
subjects:
- namespace: oirins
  kind: ServiceAccount
  name: oiri-service-account
```

## Generating the `ca.crt` Certificate

Obtain the Kubernetes certificate using the following commands:

Set up the environment variables for the OIRI namespace, and a working directory.

```
OIRINS=<OIRINS>
WORKDIR=/workdir/OIRI
```

For the Kubernetes releases up to 1.23, use the command:

```
TOKENNAME=`kubectl -n $OIRINS get serviceaccount/oiri-service-account -o
jsonpath='{.secrets[0].name}'`
```

For the Kubernetes releases 1.24 or later, use the commands:

```
TOKENNAME=oiri-service-account


TOKEN=`kubectl -n $OIRINS get secret $TOKENNAME -o jsonpath='{.data.token}'|
base64 --decode`


kubectl -n $OIRINS get secret $TOKENNAME -o jsonpath='{.data.ca\.crt}'|
base64 --decode > $WORKDIR/ca.crt
```

## Creating a Kubernetes Configuration File for OIRI

Generate a Kubernetes configuration file to tell OIRI how to interact with `kubectl`. To do this perform the following steps:

Set up environment variables for the OIRI namespace, and a working directory.

```
OIRINS=<OIRINS>
WORKDIR=/workdir/OIRI


TOKEN=`kubectl -n $OIRINS get secret $TOKENNAME -o jsonpath='{.data.token}'|
base64 --decode`


K8URL=`grep server: $KUBECONFIG | sed 's/server://;s/ //g'`


kubectl config --kubeconfig=$WORKDIR/oiri_config set-cluster oiri-cluster --
server=$K8URL --certificate-authority=$WORKDIR/ca.crt --embed-certs=true


kubectl config --kubeconfig=$WORKDIR/oiri_config set-credentials oiri-service-
account --token=$TOKEN


kubectl config --kubeconfig=$WORKDIR/oiri_config set-context oiri --user=oiri-
service-account --cluster=oiri-cluster


kubectl config --kubeconfig=$WORKDIR/oiri_config use-context oiri
```

These commands generate a file called `oiri_config` in the <`WORKDIR`> location. This file contains the Kubernetes cluster details.

## Copying Files to the OIRI-CLI Container

Copy the `ca.crt` (see Generating the `ca.crt` Certificate) and `oiri_config` (see Creating a Kubernetes Configuration File for OIRI) files to the OIRI-CLI container, using the following commands:

```
OIRINS=<OIRINS>
WORKDIR=/workdir/OIRI


kubectl cp $WORKDIR/ca.crt $OIRINS/oiri-cli:/app/k8s


kubectl cp $WORKDIR/oiri_config $OIRINS/oiri-cli:/app/k8s/config
```

From the `oiri-cli`, run the following command:

```
chmod 400  /app/k8s/config
```

## Validating the `kubectl` Command

Validate that the `kubectl` command works from inside the Kubernetes container by using the following command from the OIRI-CLI container.

```
kubectl get pod -n $OIRINS
```

For example:

```
kubectl get pod -n oirins
```

The command should show you the running `oiri-cli` pod.

# Creating the Configuration Files

OIRI uses a number of property files to deploy OIRI. These property files are populated using the CLI commands.

Run the following steps from inside the OIRI-CLI container.

- Creating the Setup Configuration Files
- Creating the Helm Configuration File

## Creating the Setup Configuration Files

To create the initial config files, run the following command:

```
/oiri-cli/scripts/setupConfFiles.sh -m prod \
             --oigdbhost <OIG_DB_SCAN> \
             --oigdbport <OIG_DB_LISTENER> \
             --oigdbsname <OIG_DB_SERVICE> \
             --oiridbhost <OIRI_DB_SCAN> \
             --oiridbport <OIRI_DB_LISTENER> \
             --oiridbsname <OIRI_DB_SERVICE> \
             --sparkmode k8s \
             --dingnamespace <DINGNS> \
             ----dingimage <OIRI_DING_REPOSITORY>:<OIRIDING_VER> \
             --cookiesecureflag false \
             --k8scertificatefilename <CERTIFICATE_FILE> \
             --sparkk8smasterurl k8s://<K8_URL> \
             --oigserverurl <OIG_URL>
```

For example:

```
/oiri-cli/scripts/setupConfFiles.sh -m prod \
             --oigdbhost db-scan.example.com \
             --oigdbport 1521 \
             --oigdbsname oig_s.example.com  \
             --oiridbhost db-scan.example.com \
             --oiridbport 1521 \
             --oiridbsname oiri_s.example.com  \
```

```
            --sparkmode k8s  \
            --dingnamespace dingns \
             --dingimage oiri-ding:12.2.1.4.02106  \
            --cookiesecureflag false \
             --k8scertificatefilename ca.crt  \
            --sparkk8smasterurl k8s://https://10.0.0.10:6443  \
            --http://governancedomain-cluster-oim-
cluster.oigns.svc.cluster.local:14000
```

The output appears as follows:

```
INFO: OIG DB as source for ETL is true
INFO: Setting up /app/data/conf/config.yaml
INFO: Setting up /app/data/conf/data-ingestion-config.yaml
INFO: Setting up /app/data/conf/custom-attributes.yaml
INFO: Setting up /app/oiri/data/conf/application.yaml
INFO: Setting up /app/oiri/data/conf/authenticationConf.yaml
INFO: Setting up /app/data/conf/dbconfig.yaml
```

Verify that the files have been created correctly by using the following command in the OIRI-CLI container:

```
ls /app/data/conf
```

You should see the following:

```
config.yaml  custom-attributes.yaml   data-ingestion-config.yaml
dbconfig.yaml
```

## Creating the Helm Configuration File

OIRI is deployed using `helm`. To create a configuration file for `helm`, run the following command inside the OIRI-CLI container.

```
/oiri-cli/scripts/setupValuesYaml.sh  \
             --oiriapiimage <OIRI_REPOSITORY>:<OIRI_VER> \
             --oirinamespace <OIRINS> \
             --oirinfsserver  <PVSERVER> \
             --oirireplicas <OIRI_REPLICAS> \
             --oiriuireplicas <OIRI_UI_REPLICAS> \
             --sparkhistoryserverreplicas <OIRI_DING_REPLICAS> \
             --oirinfsstoragepath <OIRI_NFS_SHARE> \
             --oirinfsstoragecapacity <OIRI_SHARE_SIZE> \
             --oiriuiimage <OIRI_UI_REPOSITORY>:<OIRIUI_VER> \
             --dingimage <OIRI_DING_REPOSITORY>:<OIRIDING_VER> \
             --dingnamespace <DINGNS> \
             --dingnfsserver <PVSERVER> \
             --dingnfsstoragepath <OIRI_DING_SHARE> \
             --dingnfsstoragecapacity <OIRI_DING_SHARE_SIZE> \
             --ingressenabled <USE_INGRESS> \
             --ingresshostname <OIRI_INGRESS_HOST> \
             --sslenabled false
```

For example:

```
/oiri-cli/scripts/setupValuesYaml.sh  \
              --oiriapiimage oiri:12.2.1.4.02106 \
              --oirinamespace oirins \
              --oirinfsserver  1.1.1.1 \
              --oirireplicas 2 \
              --oiriuireplicas 2 \
              --sparkhistoryserverreplicas 2 \
              --oirinfsstoragepath /exports/iampvs/oiripv \
              --oirinfsstoragecapacity 10Gi \
              --oiriuiimage oiri-ui:12.2.1.4.02106 \
              --dingimage oiri-ding:12.2.1.4.02106 \
              --dingnamespace dingns \
              --dingnfsserver 1.1.1.1 \
              --dingnfsstoragepath /exports/iampvs/dingpv \
              --dingnfsstoragecapacity 10Gi \
              --ingressenabled false \
              --ingresshostname igdadmin.example.com \
              --sslenabled false
```

Verify that the `/app/k8s/values.yaml` file is created.

# Creating the OIRI Keystore

Create a keystore in OIRI using the `keytool` command. This command should be run from the ORI-CLI container.

Use the following command to create the keystore:

```
keytool -genkeypair -alias oiri -keypass <OIRI_KEYSTORE_PWD> -keyalg RSA \
             -keystore /app/oiri/data/keystore/keystore.jks \
             -storepass <OIRI_KEYSTORE_PWD> -storetype pkcs12 \
             -dname \"CN=Unknown, OU=Unknown, O=Unknown, L=Unknown,
ST=Unknown, C=Unknown\" \
             -noprompt
```

# Loading the OIG Certificates into OIRI

For OIRI to trust OIG, you should load the OIG certificate into OIRI.

Complete the following steps to load the certificates:

- Obtaining the OIG REST Certificate
- Importing the OIG REST Certificate into OIRI
- Obtaining the OIG SSL Certificate
- Importing the OIG SSL Certificate into OIRI

## Obtaining the OIG REST Certificate

To obtain the OIG rest certificate (this is different from the regular OIG certificate), you should run the commands from inside the OIG Administration Server container.

To obtain the OIG rest certificate:

1. Log in to the OIG Administration Container using the command:

```
kubectl exec -n <OIGNS> -ti <OIG_DOMAIN_NAME>-adminserver -- /bin/bash
```

For example:

```
kubectl exec -n oigns -ti governancedomain-adminserver -- /bin/bash
```

2. Obtain the certificate using the command:

```
keytool -export -rfc -alias xell \
               -file /u01/user_projects/workdir/xell.pem \
               -keystore /u01/user_projects /domains/$OIG_DOMAIN_NAME/
config/fmwconfig/default-keystore.jks \
               -storepass <OIG_WEBLOGIC_PWD>
```

For example:

```
keytool -export -rfc -alias xell \
               -file /u01/user_projects/workdir/xell.pem \
               -keystore /u01/user_projects /domains/governancedomain/
config/fmwconfig/default-keystore.jks \
               -storepass <password>
```

3. Copy the certificate to OIRI by using the following command. This command should be run from the administration node:

```
kubectl cp <OIGNS>/<OIG_DOMAIN_NAME> adminserver:/u01/oracle/user_projects/
workdir/xell.pem <WORKDIR>/xell.pem


kubectl cp <WORKDIR>/xell.pem <OIRINS>/oiri-cli:/app/k8s/xell.pem
```

For example:

```
kubectl cp oigns/governancedomain-adminserver:/u01/oracle/user_projects/
workdir/xell.pem /workdir/OIRI


kubectl cp /workdir/OIRI/xell.pem oirins/oiri-cli:/app/k8s/xell.pem
```

- OIG Running Inside Kubernetes
- OIG Not Running Inside Kubernetes

## OIG Running Inside Kubernetes

You must run the commands inside Kubernetes.

## OIG Not Running Inside Kubernetes

1. Log into your OIG host and change directory to `DOMAIN_HOME/config/fmwconfig/`.

For Example:

```
cd /u01/oracle/config/domains/governancedomain/config/fmwconfig/
```

(Optional) <Enter a step example.>

2. Use the following command to obtain the certificate:

```
keytool -export -rfc -alias xell \
               -file /tmp/xell.pem \
               -keystore <DOMAIN_HOME>/config/fmwconfig/default-
keystore.jks \
               -storepass <OIG_WEBLOGIC_PWD>
```

For Example:

```
keytool -export -rfc -alias xell \
               -file /tmp/xell.pem \
               -keystore /u01/oracle/config/domains/governancedomain/
config/fmwconfig/default-keystore.jks \
               -storepass password
```

3. Transfer the `/tmp/xell.pem` file to a host which has access to your kubernetes cluster.

> **Note:**
>
> The recommended practice is to place the file inside your `<WORKDIR>`.

4. Use the following command to copy the certificate to OIRI. This command should be run from the administration node:

```
kubectl cp <WORKDIR>/xell.pem <OIRINS>/oiri-cli:/app/k8s/xell.pem
```

```
kubectl cp /workdir/OIRI/xell.pem oirins/oiri-cli:/app/k8s/xell.pem
```

# Importing the OIG REST Certificate into OIRI

After obtaining a copy of the OIG REST certificate in the OIRI container, you have to import the certificate using the following command from the OIRI-CLI container.

```
keytool -import \
              -alias xell \
              -file /app/k8s/xell.pem \
              -keystore /app/oiri/data/keystore/keystore.jks\
              -storepass  <OIRI_KEYSTORE_PWD> -noprompt
```

## Obtaining the OIG SSL Certificate

In addition to the OIG REST certificate, you should also trust the OIG SSL Certificate. This certificate is assigned to the load balancer. The simplest way to obtain this certificate is to use the following command that has access to the OIG load balancer:

```
openssl s_client -connect <OIG_LBR_HOST>:<OIG_LBR_PORT> -showcerts </dev/null
2>/dev/null|openssl x509 -outform PEM > <OIG_LBR_HOST>.pem
```

For example:

```
openssl s_client -connect prov.example.com:443 -showcerts </dev/null 2>/dev/
null|openssl x509 -outform PEM > prov.example.com.pem
```

Copy the certificate to the OIRI-CLI container using the command:

```
kubectl cp /workdir/OIRI/prov.example.com.pem oirins/oiri-cli:/app/k8s/
prov.example.com.pem
```

## Importing the OIG SSL Certificate into OIRI

After obtaining a copy of the OIG certificate in the OIRI container, you should import the certificate using the following command from the OIRI-CLI container:

```
keytool -import \
               -alias oigssl \
               -file /app/k8s/<CERT_FILE> \
               -keystore /app/oiri/data/keystore/keystore.jks\
               -storepass  <OIRI_KEYSTORE_PWD> -noprompt
```

For example:

```
keytool -import \
               -alias oigssl \
               -file /app/k8s/prov.example.com.pem \
               -keystore /app/oiri/data/keystore/keystore.jks\
               -storepass <password> -noprompt
```

# Creating Wallets

OIRI stores database/OIG connection information in a wallet. You have to create the wallet by running the command from inside the OIRI-CLI pod.

Use the following command to create the wallet:

```
oiri-cli --config=/app/data/conf/config.yaml wallet create \
            --oigsau <OIRI_SERVICE_USER> \
            --oigsap <OIRI_SERVICE_PWD> \
            --oirijka oiri \
            --oirijkp <OIRI_KEYSTORE_PWD> \
```

```
            --oiriksp <OIRI_KEYSTORE_PWD> \
            --oiridbuprefix <OIRI_RCU_PREFIX> \
            --oiridbp <OIRI_SCHEMA_PWD> \
            --oigdbu <OIG_RCU_PREFIX>_OIM \
            --oigdbp <OIG_SCHEMA_PWD>
```

For example:

```
oiri-cli --config=/app/data/conf/config.yaml wallet create \
            --oigsau oirisvc \
            --oigsap myservicepwd \
            --oirijka oiri \
            --oirijkp mykeystorepwd \
            --oiriksp mykeystorepwd \
            --oiridbuprefix oiri \
            --oiridbp myschemapwd \
            --oigdbu IGD_OIM \
            --oigdbp myoigschemapwd
```

Verify that the wallets have been created, using the command:

```
ls /app/data/wallet /app/oiri/data/wallet
```

# Creating the Database Schemas

Create the OIRI database schemas in the database by running the commands from the OIRI-CLI container.

Use the following commands:

```
oiri-cli --config=/app/data/conf/config.yaml schema create /app/data/conf/
dbconfig.yaml --sysp <OIRI_DB_SYS_PWD>
```

```
oiri-cli --config=/app/data/conf/config.yaml schema migrate /app/data/conf/
dbconfig.yaml
```

The output will appear as follows:

```
Creating the schema oiri_oiri
CREATING OIRI SCHEMA ............
================================================
DB USER oiri_oiri has been successfully created
Migrating the OIRI schema
Migrating OIRI SCHEMA ............
================================================
log4j:WARN No appenders could be found for logger
(org.flywaydb.core.internal.scanner.classpath.ClassPathScanner).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
========== Before Migrate =============
 Script:V1__RoleMining.sql Installed On:null State:PENDING Version:1
Description:RoleMining
========== After Migrate =============
 Script:V1__RoleMining.sql Installed On:2021-03-02 08:01:54.18592 State:SUCCESS
```

```
Version:1 Description:RoleMining
OIRI Schema has been successfully migrated
```

# Verifying the Wallet

After creating the wallet, you should validate the wallet. If the validation fails, correct the wallet before you proceed.

Verify the wallet using the following command:

```
./verifyWallet.sh
```

The output will appear as follows:

```
Verifying Wallets. Wallet locations and entries will be validated
DING Wallet is Valid.
OIRI Wallet is Valid.
OIRI DB Connection is Valid.
OIG DB Connection is Valid.
KeyStore location and entries are Valid.
OIG Server Connection is Valid.
SUCCESS: Wallet locations and entries are valid.
```

If any validation fails, correct the wallet using the command:

```
oiri-cli --config=/app/data/conf/config.yaml wallet update
```

# Deploying OIRI Using Helm

After creating the namespaces, you can now deploy OIRI using the generated Helm chart. You should execute the command from the OIRI-CLI.

Use the following command:

```
helm install oiri /helm/oiri -f /app/k8s/values.yaml
```

The output will appear as follows:

```
NAME: oiri
LAST DEPLOYED: Mon Jan 11 15:14:22 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Please be patient while the chart installs. Pod may not be in running status.

To check the status of the pod, run following command.
Pods READY state must be 1/1 and status RUNNING

    kubectl get pods --namespace oiri

    kubectl get pods --namespace ding

Access OIRI Service by using following URL in your browser.

    https://IP_ADDRESS:PORT/
```

```
Access OIRI UI by using following URL in your browser.

    https://IP_ADDRESS:PORT/oiri/ui/v1/console

Admins can access DING History Server by port forwarding the ding-history pod through
kubectl.

  kubectl port-forward <pod_name> <desired_port>:18080 -n ding

Inside the DING-CLI, use following commands to start data ingestion

    ding-cli --config=/app/data/conf/config.yaml data-ingestion start /app/data/conf/
data-ingestion-config.yaml
```

# Verifying that OIRI is Running

After you deploy OIRI, you should verify that it is running successfully.

To verify that OIRI is running, use the command:

```
kubectl -n <OIRINS> get pods -o wide


kubectl -n <DINGNS> get pods -o wide
```

You should see list of pods with the status 'Running'. For example:

```
NAME                     READY     STATUS    RESTARTS   AGE
oiri-6cd5755fb-j7r4      1/1       Running    0         42h
oiri-6cd5755fb-s42xx     1/1       Running    0         42h
oiri-ui-d55cd6b69-62nm6  1/1       Running    0         42h
oiri-ui-d55cd6b69-rxdwm  1/1       Running    0         42h


kubectl -n dingns get pods


NAME                                    READY   STATUS    RESTARTS AGE
oiri-ding-7045127aee424d93-driver       0/1     Completed  0       41h
spark-history-server-6cc6d9d8c7-2594r   1/1     Running    0       42h
spark-history-server-6cc6d9d8c7-jdcqc   1/1     Running    0       42h
```

# Creating the Kubernetes NodePort Services

By default, OIRI gets created with all the components configured as `ClusterIP` services. The configuration indicates that the Oracle Identity Role Intelligence components are visible only within the Kubernetes cluster.

In an enterprise deployment, all interactions with the OIRI components take place through the Oracle HTTP Server which is located outside of the Kubernetes cluster. If you use OHS and Ingress controller, the Ingress services are created for you. If you are using a NodePort deployment, you should create the NodePort Services to access the deployment.

- Creating an OIRI NodePort Service
- Creating an OIRI UI NodePort Service

# Creating an OIRI NodePort Service

To create an OIRI NodePort Service:

1. Create the `oiri_nodeport.yaml` text file with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: oiri-nodeport
  namespace: <OIRINS>
spec:
  type: NodePort
  selector:
    app: oiri
  ports:
    - targetPort: 8005
      port: 8005
      nodePort: <OIRI_K8>
      protocol: TCP
```

> **Note:**
>
> Ensure that the namespace is set to the namespace you want to use.

For example:

```
kind: Service
apiVersion: v1
metadata:
  name: oiri-nodeport
  namespace: oirins
spec:
  type: NodePort
  selector:
    app: oiri
  ports:
    - targetPort: 8005
      port: 8005
      nodePort: 30305
      protocol: TCP
```

2. Create the service using the following command:

```
kubectl create -f oiri_nodeport.yaml
```

# Creating an OIRI UI NodePort Service

To create an OIRI UI NodePort Service:

1. Create the `oiriui_nodeport.yaml` text file with the following content:

```
kind: Service
apiVersion: v1
metadata:
  name: oiri-ui-nodeport
  namespace: <OIRINS>
spec:
  type: NodePort
  selector:
    app: oiri-ui
  ports:
    - targetPort: 8080
      port: 8080
      nodePort: <OIRI_UI_K8>
      protocol: TCP
```

> **Note:**
>
> Ensure that the namespace is set to the namespace you want to use.

For example:

```
kind: Service
apiVersion: v1
metadata:
  name: oiri-ui-nodeport
  namespace: oirins
spec:
  type: NodePort
  selector:
    app: oiri-ui
  ports:
    - targetPort: 8080
      port: 8080
      nodePort: 30306
      protocol: TCP
```

2. Create the service using the following command:

```
kubectl create -f oiriui_nodeport.yaml
```

# Updating the OHS Configuration

If you have not already done so, you should now add the OIRI entries to the Oracle HTTP configuration.

See Configuring Oracle HTTP Server for Oracle Identity Role Intelligence.

# Performing an Initial Data Load Using the Data Ingester

After OIRI is up and running, you may want to perform an initial data load from the OIG database.

Complete the following steps for a data load:

- Starting the DING CLI
- Copying the Kubernetes Certificate to DING
- Verifying the DING Configuration
- Running the Data Ingestion
- Setting the Next Data Load to Incremental

## Starting the DING CLI

To start the DING CLI, perform the steps on a node that has access to the `kubectl` command:

1. Create the `ding-cli.yaml` file with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: oiri-ding-cli
  namespace: <DINGNS>
  labels:
    app: dingcli
spec:
  serviceAccount: ding-sa
  restartPolicy: OnFailure
  volumes:
    - name: oiripv
      nfs:
        server: <PVSERVER>
        path: <OIRI_SHARE>
    - name: dingpv
      nfs:
        server: <PVSERVER>
        path: <OIRI_DING_SHARE>
    - name: workpv
      nfs:
        server: <PVSERVER>
        path: <OIRI_WORK_SHARE>
  containers:
  - name: oiricli
    image: <OIRI_DING_REPOSITORY>:<OIRIDING_VER>
    volumeMounts:
      - name: oiripv
        mountPath: /app/oiri
      - name: dingpv
        mountPath: /app
      - name: workpv
        mountPath: /app/k8s
    command: ["/bin/bash", "-ec", "tail -f /dev/null"]
```

```
  imagePullSecrets:
    - name: regcred
```

For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: oiri-ding-cli
  namespace: dingns
  labels:
    app: dingcli
spec:
  serviceAccount: ding-sa
  restartPolicy: OnFailure
  volumes:
    - name: oiripv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/oiripv
    - name: dingpv
      nfs:
        server: 0.0.0.0
        path/exports/IAMPVS/dingpv
    - name: workpv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/workpv
  containers:
  - name: oiricli
    image: iad.ocir.io/mytenancy/idm/oiri-ding:12.2.1.4.02106
    volumeMounts:
      - name: oiripv
        mountPath: /app/oiri
      - name: dingpv
        mountPath: /app
      - name: workpv
        mountPath: /app/k8s
    command: ["/bin/bash", "-ec", "tail -f /dev/null"]
  imagePullSecrets:
    - name: regcred
```

2. Start the DING Administration CLI using the following command:

```
kubectl create -f ding-cli.yaml
```

3. Connect to the running container using the command:

```
kubectl exec -n dingns -ti oiri-ding-cli -- /bin/bash
```

> **Note:**
>
> When the examples say "issue the following command from within the OIRI-CLI, it means that you should connect to the running container as described here, and then run the commands as specified.

## Copying the Kubernetes Certificate to DING

To enable interaction between DING and Kubernetes, you should copy the Kubernetes certificate to the DING container.

To copy the certificate:

1. Obtain the Kubernetes certificate by running the following command on the deployment server. This command copies the certificate to the locally mounted DING persistent volume:

   ```
   grep certificate-authority-data $KUBECONFIG | tr -d " " | sed 's/
   certificate-authority-data://' | base64 -d > /workdir/OIRI/ca.crt
   ```

2. Verify that the resulting file has a valid certificate that looks as follows:

   ```
   -----BEGIN CERTIFICATE-----
   RANDOM CHARACTERS
   RANDOM CHARACTERS
   RANDOM CHARACTERS
   -----END CERTIFICATE-----
   ```

3. Copy the certificate to the DING container using the command:

   ```
   kubectl cp <WORKDIR>/ca.crt <DINGNS>/oiri-ding-cli:/app/ca.crt
   ```

   For example:

   ```
   kubectl cp /workdir/OIRI/ca.crt dingns/oiri-ding-cli:/app/ca.crt
   ```

## Verifying the DING Configuration

DING is configured as part of the setup to point to the OIG database. Validate that DING can connect to the database successfully by using the following command from the DING-CLI:

```
ding-cli --config=/app/data/conf/config.yaml data-ingestion verify /app/data/
conf/data-ingestion-config.yaml
```

You should see the following message:

```
SUCCESS: Data Ingestion Config is valid
```

## Running the Data Ingestion

To start the Data Ingestion, use the following command from the DING-CLI:

```
ding-cli --config=/app/data/conf/config.yaml data-ingestion start /app/data/
conf/data-ingestion-config.yaml
```

You should see an entry that looks as follows:

```
INFO: 21/07/28 18:02:43 INFO LoggingPodStatusWatcherImpl: Application status
for spark-901c5ed4ba4e4233b4501b8e1279a9cf (phase: Succeeded)
```

This entry indicates that the data load is successful.

## Setting the Next Data Load to Incremental

The initial data load using Data Ingester loads all the OIG data into the OIRI database. See Performing an Initial Data Load Using the Data Ingester. Future loads should only change data. To change data, run the following command from the DING-CLI:

```
/ding-cli/scripts/updateDataIngestionConfig.sh \
         --entityusersenabled true --entityuserssyncmode incremental \
          --entityapplicationsenabled true --entityapplicationssyncmode
incremental \
          --entityentitlementsenabled true --entityentitlementssyncmode
incremental \
          --entityassignedentitlementsenabled true --
entityassignedentitlementssyncmode incremental \
          --entityrolesenabled true --entityrolessyncmode incremental \
           --entityrolehierarchyenabled true --entityrolehierarchysyncmode
incremental \
           --entityroleusermembershipsenabled true --
entityroleusermembershipssyncmode incremental \
            --entityroleentitlementcompositionsenabled true --
entityroleentitlementcompositionssyncmode incremental \
          --entityaccountsenabled true --entityaccountssyncmode
```

21

# Installing and Configuring Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

Oracle Advanced Authentication and Risk Management (OAA and OARM) authenticates users by using multi-factor authentication. It integrates with Oracle Access Manager for OAuth authentication.

For details about installing Oracle Access Manager, see Configuring Oracle Access Manager Using WDT. If you have an existing OAM deployment, you can use the same deployment. Unlike the traditional Oracle Identity and Access Management products, Oracle Advanced Authentication is deployed as a series of microservices.

In this release, Oracle uses a standalone container image to install and configure OAA. The container image is started manually in the Kubernetes cluster.

This chapter includes the following topics:

- About Oracle Advanced Authentication
- About Oracle Adaptive Risk Management (OARM)
  Oracle Adaptive Risk Management (OARM) is an integrated system that aggregates risk data associated with users and user activities, analyzes and evaluates business risks posed by users and their activities, and provides advice to be acted upon to mitigate them.
- About Oracle Universal Authenticator (OUA)
  Oracle Universal Authenticator allows end users to login to their devices using Oracle Access Management (OAM) credentials which enables end users to access OAM protected applications and other Single-Sign On (SSO) enabled applications via SSO.
- Variables Used in this Chapter
  The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.
- Characteristics of the OAA Installation
  This section lists the key characteristics of the OAA installation that you are about to create. Review these characteristics to understand the purpose and context of the procedures that are used to configure OAA.
- Kubernetes Services
  If you are using NodePort Services, the Kubernetes services are created as part of the OAA installation.
- Before You Begin
  Before you begin the installation, you have to ensure that all the required tasks listed in this topic are complete.
- Creating Kubernetes Namespaces
  The Kubernetes namespaces are used to store the OAA Objects.

- Creating a Container Registry Secret
  If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret that contains the login details of the container registry.
- Creating a Kubernetes Secret for Docker Hub Images
  This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands.
- Creating a GitHub Secret
- Creating the Management Container
- Copying Production Certificates to the Management Container
- Starting the Management Container
  Before starting the Management Container, ensure that you have created the persistent volumes.
- Obtaining the OAM Certificate
- Registering OAM TAP Partners
  You must register OAM Tap partners that are used to enable OAA and OUA to authenticate with OAM in a more secure manner.
- Creating the OAA Property File
  The OAA deployment is dependent on the values in a property file. This file is used for creating the database schemas and to deploy OAA itself. The steps to create the file is performed inside the OAA-MGMT pod.
- Creating the OAA Override File
  The OAA Override file is used to determine the number of each type of container that is started. In a highly available deployment, there should be a minimum of two for each container type.
- Creating the Database Schemas
  Oracle Advanced Authentication automatically creates the schemas in the database.
- Deploying Oracle Advanced Authentication
  To deploy the OAuth application, you should perform the steps inside the OAA-MGMT pod.
- Resolving Timeouts
  Your deployment may fail if it takes longer than the expected time to pull the container images from the container registry. You will see the error in the deployment log.
- Configuring Email/SMS Servers and Automatic User Creation
  OAA has built-in email/SMS integration. You have to configure OAA to use the Email SMTP client and the SMS client.
- Validating OAA
- Configuring Oracle Universal Authentication
  This section describes various tasks related to configuring Oracle Universal Authentication.

# About Oracle Advanced Authentication

Oracle Advanced Authentication (OAA) is a standalone microservice that supports establishing and asserting the identity of users. OAA provides strong authentication using Multiple Authentication Factors (MFA). A wide range of authentication (challenge) factors is available out-of-the-box for establishing the identity of users.
OAA supports integration with Oracle Access Management (OAM) to provide MFA capabilities.

**Features of OAA**

- Runs as a standalone microservice on a Kubernetes platform and is deployed using Helm charts.
- Supports integration with the following clients to enable Multi-factor Authentication (MFA):
  - Clients that provide web-based user login flows, such as Oracle Access Management (OAM). OAA integrates with OAM through the Trusted Authentication Protocol (TAP).
  - Clients that provide API-based user login flows, such as Oracle Radius Agent (ORA). OAA integrates with ORA through REST APIs. This type of integration enables clients to manage its own user flow orchestration.
- Provides `OAAAuthnPlugin` for integrating with OAM. The plug-in also enables migration of user data from the identity store on OAM to OAA.
- Provides web UI (Administration UI Console) for administrators to create and manage client registrations, assurance levels, and rules. Administrators can also achieve all the administration tasks using the REST APIs.
- Provides web UI (User Preferences UI) for end-users to manage and register their challenge factors. User self-registration and management can also be performed using REST APIs.
- Web UIs are secured by OAM OAuth and OpenID Connect (OIDC).
- Provides the following challenge-factors out-of-the-box:
  - TOTP (Time-based One Time Password) with Oracle Mobile Authenticator (OMA), Google, and Microsoft
  - OTP (One Time Password) with E-MAIL and SMS
  - Yubikey OTP
  - FIDO2
  - Knowledge-based Authentication (KBA)
  - Push notifications

# About Oracle Adaptive Risk Management (OARM)

Oracle Adaptive Risk Management (OARM) is an integrated system that aggregates risk data associated with users and user activities, analyzes and evaluates business risks posed by users and their activities, and provides advice to be acted upon to mitigate them.

The system works best when integrated with Oracle Advanced Authentication (OAA), which executes risk mitigation actions to Block, Challenge, or Allow user activities based on the risk assessment associated with it.

The system can also work in a stand-alone mode where it can be consulted for remedial actions by consuming applications. OARM system is highly extensible owing to its microservices-based architecture, allowing additional capabilities to be added without having to indulge in a costly upgrade process.

**Features of OARM**

- OARM system revolves around user activities, which are secured using business friendly rules.

- OARM is shipped with an out-of-the-box user authentication activity, which is baked in with a rich set of rules that can readily be used to secure the business. The system also provides the capability to augment the user authentication activity with additional rules, remove rules not applicable to business, or add net new user activities to be monitored. OARM supports seeding data feeds from certified external sources that would also be used in risk analytics. This, combined with OARM's profiling capability, provides the right mix of seed data for running analytics.
- Configuration of rules and managing and monitoring user activities can be achieved with an intuitively designed Administration Console. The Administration Console allows administrators to implement rules applicable to their organization without being concerned with the nuances of the underlying system.
- OARM, in conjunction with OAA, provides a large set of modern, multi-factor challenge methods enabling administrators to choose challenge mechanisms that fit their business requirements. OAA also makes integration of OARM with existing Identity Management systems such as Oracle Access Management Suite (OAM), very easy to achieve.

# About Oracle Universal Authenticator (OUA)

Oracle Universal Authenticator allows end users to login to their devices using Oracle Access Management (OAM) credentials which enables end users to access OAM protected applications and other Single-Sign On (SSO) enabled applications via SSO.

Oracle Universal Authenticator addresses both client-side and server-side requirements. This guide only provides information related to aspects of the server-side configuration.

> **Note:**
>
> You must ensure that the April 2024 Stack Patch Bundle is applied to your OAM Installation before installing OUA.

# Variables Used in this Chapter

The later sections of this chapter provide instructions to create a number of files. These sample files contain variables which you need to substitute with values applicable to your deployment.

Variables are formatted as <*VARIABLE_NAME*>. The following table provides the values you should set for each of these variables.

**Table 21-1 The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| <*REGISTRY_ADDRESS*> | `iad.ocir.io/<mytenancy>` | The location of the registry. |
| <*REGISTRY_SECRET_NAME*> | `regcred` | The name of the Kubernetes secret you created with the stored registry credentials. See Creating a Container Registry Secret. |
| <*REG_USER*> | `mytenancy/oracleidentitycloudservice/myemail@email.com` | The name of the user you use to log in to the registry. |
| <*REG_PWD*> | <*password*> | The registry user password. |

**Table 21-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OAA_MGT_REPOSITORY>* | `oracle/oaa-mgmt`<br>`local/oracle/oaa-mgmt`<br>`container-registry.oracle.com/middleware/oaa-mgmt_cpu`<br>*<REGISTRY_ADDRESS>*`/oracle/oaa-mgmt` | The name of the OAA management image file.<br>If you have downloaded and staged a container image, this value will be: `oracle/local/oaa-mgmt`. If you use OLCNE, the value will be `local/oaa-mgmt`.<br>If you use the Oracle container registry, the value will be `container-registry.oracle.com/middleware/local/oaa-mgmt_cpu`.<br>If you use a container registry, the value will be the name of the registry with the product name: *<REGISTRY_ADDRESS>*`/oaa-mgmt`. |
| *<OAAMGT_VER>* | `12.2.1.4.1_20220419` | The version of the image you want to use. |
| *<OAA_DEPLOYMENT_TYPE>* | `both` | To deploy only OAA, set to `oaa`.<br>To deploy both OAA and RISK, set to `both`.<br>To deploy OAA, RISK, and OUA, set to `OUA`. |
| *<OAA_DEPLOYMENT>* | `edg` | Name used for the deployment. Each pod created will be prefixed by this value. |
| *<PVSERVER>* | `1.1.1.1` | The name or IP address of the NFS server hosting the persistent volumes. |
| *<OAANS>* | `oaans` | The Kubernetes namespace to hold OAA objects. |
| *<WORKDIR>* | `/workdir/OAA` | The location where you want to create the working directory for OAA. |
| *<OAA_CONFIG_SHARE>* | `/exports/IAMPVS/oaaconfigpv` | The NFS mount location for the OAA configuration persistent volume. |
| *<OAA_CRED_SHARE>* | `/exports/IAMPVS/oaacredpv` | The NFS mount location for the OAA credential persistent volume. |
| *<OAA_LOG_SHARE>* | `/exports/IAMPVS/oaalogpv` | The NFS mount of the OAA logs persistent volume. |
| *<OAA_VAULT_SHARE>* | `/exports/IAMPVS/oaavaultpv` | The NFS mount of the OAA vault persistent volume. |

**Table 21-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OAA_DB_SID>* | `iamdb11` | The ORACLE_SID of the database instance that you are using. This value is required because the schemas are created by using the Oracle Data Pump Import program (impdp). |
| *<OAA_KEYSTORE_PWD>* | `oaapassword` | The password to be used for OAA keystores. |
| *<OAA_API_PWD>* | `myAPIpassword` | The password assigned to protect OAA APIs. |
| *<UMS_SERVER_URL>* | `http://governancedomain-cluster-soa-cluster.oigns.svc.cluster.local:8001/ucs/messaging/webservice` | The URL of the email server.<br>If you are using Oracle Unified Messaging (either standalone or as part of OIG):<br>• If OIG is inside the Kubernetes cluster: `http://<OIG_DOMAIN_NAME>-cluster-soa-cluster.<OIG>.svc.cluster.local:8001/ucs/messaging/webservice`<br>• If OIG is outside the Kubernetes cluster: `http://igdinternal.example.com/ucs/messaging/webservice` |
| *<UMS_ADMIN_USER>* | `weblogic` | The user name of the SMTP service. For instance, in a Unified Messaging System (UMS) installation, it is the name of the WebLogic administration user. |
| *<UMS_ADMIN_ PASSWORD>* | *<password>* | The password of the *<UMS ADMIN USER>* account. |
| *<SSL_COUNTRY>* | `US` | Your country code. |
| *<SSL_STATE>* | `California` | The name of your state or locality. |
| *<SSL_CITY>* | `Redwood` | The name of your city. |
| *<SSL_ORG>* | `Oracle Corporation` | The name of your organization. |
| *<LDAP_OAMADMIN_USER>* | `oamadmin` | The name of the user who administers OAM. |
| *<LDAP_USER_PWD>* | `<mypassword>` | Password to be assigned to all the LDAP user accounts. |
| *<OAA_OAM_TAP_PARTNER>* | `OAM-OAA-TAP` | The name you want to give to the OAA partner application. |
| *<OAA_OUA_TAP_PARTNER>* | `OAM-OUA-TAP` | The name you want to give to the OUA partner application. This is required only if you deploy OUA. |

**Table 21-1 (Cont.) The Variables to be Changed**

| Variable | Sample Value | Description |
|---|---|---|
| *<OAM_LOGIN_LBR_PROTOCOL>* | `https` | The type of access protocol used for the login load balancer. In an SSL terminated environment, this value will be `https`. |
| *<OAM_LOGIN_LBR_HOST>* | `login.example.com` | The name of your login load balancer virtual name. |
| *<OAM_LOGIN_LBR_PORT>* | `443` | The port of the login load balancer virtual name. |
| *<OAA_ADMIN_K8>* | `30410` | The NodePort Service of the OAA administration pod. |
| *<OAMNS>* | `oamns` | The domain namespace to be used to store OAM objects. |
| *<OAM_DOMAIN_NAME>* | `accessdomain` | The name of the domain to be created. |
| *<OAM_ADMIN_LBR_HOST>* | `iadadmin.example.com` | The virtual host of the OAM administration functions. |
| *<OAM_ADMIN_LBR_PORTT>* | `80` | The virtual port of the OAM administration functions. |
| *<OIG_DOMAIN_NAME>* | `governancedomain` | The name of the domain to be created. |
| *<ELK_HOST>* | `https://elasticsearch-es-http.elkns.svc:9200` | The host and port of the centralized Elasticsearch deployment. This host can be inside the Kubernetes cluster or external to it. This host is used only when Elasticsearch is used. |
| *<ELK_VER>* | `8.11.0` | The version of Elasticsearch you want to use. |
| *<LDAP_WLSADMIN_USER>* | `weblogic_iam` | The WebLogic Administration user defined in LDAP. |
| *<LDAP_WLSADMIN_PWD>* | `MyPassword` | The WebLogic Administration user password defined in LDAP. |
| *<OAM_WEBLOGIC_USER>* | `weblogic` | The Internal WebLogic Administration user. |
| *<OAM_WEBLOGIC_PWD>* | `MyPassword` | The Internal WebLogic Administration user password. |
| *<OAA_SPUI_URL>* | `https://login.example.com:443/oaa/rui/authn/v1` | The URL for the OAA Self Service Console. |
| *<OAA_POLICY_URL>* | `https://login.example.com:443/policy` | The URL for the OAA Policy REST APIs. |

# Characteristics of the OAA Installation

This section lists the key characteristics of the OAA installation that you are about to create. Review these characteristics to understand the purpose and context of the procedures that are used to configure OAA.

**Table 21-2 Key Characteristics of the OAA Installation**

| Characteristics of OAA | More Information |
|---|---|
| Each microservice is deployed into a pod in the Kubernetes cluster. | See About the Kubernetes Deployment. |
| Places the OAA components in a dedicated Kubernetes namespace. | See About the Kubernetes Deployment. |
| Uses a vault which can be file-based or OCI-based. | See Creating a Vault. |
| Uses the Kubernetes services to interact with microservices. | See Creating the Kubernetes Services. |
| Uses the Kubernetes persistent volumes to hold configuration information. | See #unique_614. |
| Each Kubernetes pod is built from a pre-built Oracle container image. | See Identifying and Obtaining Software Distributions for an Enterprise Deployment. |
| Requires Oracle Access Manager to be installed and configured. | See Configuring Oracle Identity Governance Using WDT. |
| Installation can be standalone or integrated. | See Oracle Advanced Authentication. |

# Kubernetes Services

If you are using NodePort Services, the Kubernetes services are created as part of the OAA installation.

**Table 21-3 Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| edg-cache-proxy | NodePort | 32394 | 20000 |
| edg-cache-rest | NodePort | 30418 | 8080 |
| edg-oaa | NodePort | 31867<br>30343 | 80<br>443 |
| edg-oaa-admin-ui | NodePort | 32405 | 443 |
| edg-oaa-policy | NodePort | 32387 | 443 |
| edg-email | NodePort | 30800 | 443 |
| edg-fido | NodePort | 30017 | 80 |
| edg-kba | NodePort | 31836 | 443 |
| edg-sms | NodePort | 31350 | 443 |
| edg-spui | NodePort | 31573 | 443 |
| edg-totp | NodePort | 30230 | 443 |

**Table 21-3 (Cont.) Kubernetes NodePort Services**

| Service Name | Type | Service Port | Mapped Port |
|---|---|---|---|
| edg-yotp | NodePort | 30548 | 443 |
| edg-push | NodePort | 30550 | 443 |
| edg-risk | NodePort | 30560 | 443 |
| edg-risk-cc | NodePort | 30570 | 443 |
| edg-oaa-drss | NodePort | 30580 | 443 |
| edg-oua-admin-ui | NodePort | 30525 | 443 |

# Before You Begin

Before you begin the installation, you have to ensure that all the required tasks listed in this topic are complete.

The following tasks have to complete:

- Obtain the required Oracle container images staged on each of the Kubernetes worker nodes, or hosted in a container registry to which you have access. For a list of the container images, see Procuring Software for an Enterprise Deployment.

  > **Note:**
  >
  > The instructions in this section relate to the December 2024 release onwards.

- Configured Oracle HTTP Server as described in Installing and Configuring Oracle HTTP Server.
- Configured Oracle Unified Directory as describer in Installing and Configuring Oracle Unified Directory.
- Configured Oracle Access Manager as described in Configuring Oracle Access Manager Using WDT.
- Enabled WebGate as described in Configuring Single Sign-On for an Enterprise Deployment.

> **Note:**
>
> If you are adding to an existing deployment, you should revisit the above sections to ensure that you have the entries in place for Oracle Advanced Authentication.

# Creating Kubernetes Namespaces

The Kubernetes namespaces are used to store the OAA Objects.

Use the following command to create a namespace for OAA:

```
kubectl create namespace <OAANS>
```

For example:

```
kubectl create namespace oaans
```

# Creating a Container Registry Secret

If you are using a container registry and want to pull the Oracle container images on demand, you must create a secret that contains the login details of the container registry.

This step is not required if you have staged the container images locally. Oracle strongly recommends the use of a container registry.

To create a container registry secret, use the following command:

```
kubectl create secret -n <OAANS> docker-registry <REGISTRY_SECRET_NAME> --
docker-server=<REGISTRY_ADDRESS> --docker-username=<REG_USER> --docker-
password=<REG_PWD>
```

For example:

```
kubectl create secret -n oaans docker-registry regcred --docker-
server=iad.ocir.io/mytenancy --docker-username=mytenancy/
oracleidentitycloudservice/myemail@email.com --docker-password=<password>
```

> **Note:**
>
> You should create a registry secret in the OAA namespace.

# Creating a Kubernetes Secret for Docker Hub Images

This secret allows Kubernetes to pull an image from `hub.docker.com` which contains third-party images such as `helm`, `kubectl`, and `logstash` commands.

You should have an account on `hub.docker.com`. If you want to stage the images in your own repository, you can do so and modify the `helm` override file as appropriate.

To create a Kubernetes secret for `hub.docker.com`, use the following command:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="<DH_USER>" --docker-
password="<DH_PWD>" --namespace=<OUDNS>
```

For example:

```
$ kubectl create secret docker-registry dockercred --docker-server="https://
index.docker.io/v1/" --docker-username="username" --docker-
password="<mypassword>" --namespace=oudns
```

# Creating a GitHub Secret

OAA is dependent on some containers in GitHub. If you want to pull these images directly from GitHub, you should create a secret with your GitHub credentials.See Logging in to GitHub.

To create a secret called `gitcred` in your namespace, use the following command:

```
kubectl create secret -n oaans docker-registry gitcred --docker-
server=ghcr.io --docker-username=mygituser --docker-password="mytoken"
```

You should create this secret in the 'oaa' namespace.

# Creating the Management Container

# Copying Production Certificates to the Management Container

For production systems, Oracle strongly recommends use of commercial certificates rather than self-signed certificates for setting up OAA communication. After obtaining these certificates, you must copy them to the directory `/u01/oracle/scripts/creds` inside the management container `common.deployment.keystorepassphrase` and `common.deployment.truststorepassphrase`.

After copying the certificates ensure that the `installOAA.property` values `common.deployment.sslcert` and `common.deployment.trustcert` point to the location of these certificates. Set the parameters `common.deployment.keystorepassphrase` and `common.deployment.truststorepassphrase` to the value of the passphrase that you set when encoding the certifcates. This file is part of the OAA deployment and will be explained in more detail later in this chapter in Creating the OAA Property File.

> **Note:**
>
> This is the location inside the container.

The following section describes how to create and convert a commercial certificate to the p12 format required by OAA.

### Create a Certificate Signing Request

There are many way of requesting a certificate. If you are required to create a signing request and submit this to a signing authority then perform the steps in this section.

1. Once you receive the certificate from the CA, rename the file to `oaa.pem` and copy it to the `$WORKDIR/oaa_ssl` directory.

> **Note:**
>
> The certificate `oaa.pem` needs to be in PEM format. If not in PEM format convert it to PEM using openssl. For example, to convert from DER format to PEM:
>
> ```
> openssl x509 -inform der -in oaa.der -out oaa.pem
> ```

Perform the following steps to use a third party Certificate Authority (CA) for generating your certificates:

1. On the node where you will run the management container installation, create a directory and navigate to that folder, for example:

   ```
   mkdir <workdir>/oaa_ssl
   export WORKDIR=<workdir>
   cd $WORKDIR/oaa_ssl
   ```

2. Generate a 4096 bit private key (`oaa.key`) for the server certificate:

   ```
   openssl genrsa -out oaa.key 4096
   ```

3. Create a Certificate Signing Request (`oaa.csr`):

   ```
   openssl req -new -key oaa.key -out oaa.csr
   ```

   When prompted enter details to create your Certificate Signing Request (CSR). For example:

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [XX]:US
   State or Province Name (full name) []:California
   Locality Name (eg, city) [Default City]:Redwood City
   Organization Name (eg, company) [Default Company Ltd]:Example Company
   Organizational Unit Name (eg, section) []:Security
   Common Name (eg, your name or your server's hostname) []:oaa.example.com
   Email Address []:

   Please enter the following 'extra' attributes
   to be sent with your certificate request
   A challenge password []:
   An optional company name []:
   ```

4. Send the CSR (`oaa.csr`) to the third party CA.

5. Copy the Trusted Root CA certificate (`rootca.pem`), and any other CA certificates in the chain (`rootca1.pem`, `rootca2.pem`, etc) that signed the `oaa.pem` to the `$WORKDIR/oaa_ssl` directory. As per above, the CA certificates must be in PEM format, so convert if necessary.

> **Note:**
>
> This may be provided by your certificate authority in the form of an intermediary and root certificate file. For example, digicert provides a file as follows:
>
> `CombinedSHA-1RootSHA-256Intermediate-<DATE>.txt`

6. If your CA has multiple certificates in a chain, create a `bundle.pem` that contains all the CA certificates:

```
cat rootca.pem rootca1.pem rootca2.pem >>bundle.pem
```

7. Create a Trusted Certificate PKCS12 file (`trust.p12`) from the files as follows

```
openssl pkcs12 -export -out trust.p12 -nokeys -in bundle.pem
```

When prompted enter and verify the Export Password.

> **Note:**
>
> Administrators should be aware of the following:
>
> - Setting an export password is mandatory.
> - If your CA does not have a certificate chain replace `bundle.pem` with `rootca.pem`.

8. Create a Server Certificate PKCS12 file (`cert.p12`) as follows:

```
openssl pkcs12 -export -out cert.p12 -inkey oaa.key -in oaa.pem -chain -
CAfile bundle.pem
```

When prompted enter and verify the Export Password.

> **Note:**
>
> Administrators should be aware of the following:
>
> - Setting an export password is mandatory.
> - If your CA does not have a certificate chain replace `bundle.pem` with `rootca.pem`.

# Starting the Management Container

Before starting the Management Container, ensure that you have created the persistent volumes.

See Creating File Systems and Mount Targets.

To start the Management Container:

1. Create a file called `oaa-mgmt.yaml` with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: oaa-mgmt
  namespace: <OAANS>
  labels:
    app: oaamgmt
spec:
  restartPolicy: OnFailure
  volumes:
    - name: oaaconfigpv
      nfs:
        server: <PVSERVER>
        path: <OAA_CONFIG_SHARE>
    - name: oaacred
      nfs:
        server: <PVSERVER>
        path: <OAA_CRED_SHARE>
    - name: oaalogpv
      nfs:
        server: <PVSERVER>
        path: <OAA_LOG_SHARE>
    - name: oaavaultpv
      nfs:
        server: <PVSERVER>
        path: <OAA_VAULT_SHARE>
  containers:
  - name: oaamgmt
    image: <OAA_MGT_REPOSITORY>:<OAAMGT_VER>
    volumeMounts:
      - name: oaaconfigpv
        mountPath: /u01/oracle/scripts/settings
      - name: oaacred
        mountPath: /u01/oracle/scripts/creds
      - name: oaalogpv
        mountPath: /u01/oracle/logs
      - name: oaavaultpv
        mountPath: /u01/oracle/service/store/oaa
    command: ["/bin/bash", "-ec", "tail -f /dev/null"]
  imagePullSecrets:
    - name: <REGISTRY_SECRET_NAME>
```

For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: oaa-mgmt
  namespace: oaans
  labels:
    app: oaamgmt
spec:
  restartPolicy: OnFailure
  volumes:
    - name: oaaconfigpv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/oaaconfigpv
    - name: oaacred
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/oaacredpv
    - name: oaalogpv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/oaalogpv
    - name: oaavaultpv
      nfs:
        server: 0.0.0.0
        path: /exports/IAMPVS/oaavaultpv
  containers:
  - name: oaamgmt
    image: iad.ocir.io/mytenancy/oaa-mgmt:12.2.1.4.1_20220419
    volumeMounts:
      - name: oaaconfigpv
        mountPath: /u01/oracle/scripts/settings
      - name: oaacred
        mountPath: /u01/oracle/scripts/creds
      - name: oaalogpv
        mountPath: /u01/oracle/logs
      - name: oaavaultpv
        mountPath: /u01/oracle/service/store/oaa
    command: ["/bin/bash", "-ec", "tail -f /dev/null"]
  imagePullSecrets:
    - name: regcred
```

2. Start the Management CLI Container using the following command:

```
kubectl create -f oaa-mgmt.yaml
```

3. Connect to the running container using the following command:

```
kubectl exec -n oaans -ti oaa-mgmt -- /bin/bash
```

> **Note:**
>
> When the examples instruct you to "use the following command from within the OAA-MGMT", it means that you should connect to the running container as described in this section, and then run the commands as specified.

- Granting the Management Container Access to the Kubernetes Cluster
  The OAA Management Container has built-in commands to interact with the Kubernetes cluster. You must provide the Management Container with details on how to access the Kubernetes cluster.
- Creating the Helm Configuration File
  The OAA installation procedure is dependent on a `helmconfig` file being present. This file is almost always an empty file.

# Granting the Management Container Access to the Kubernetes Cluster

The OAA Management Container has built-in commands to interact with the Kubernetes cluster. You must provide the Management Container with details on how to access the Kubernetes cluster.

To provide access, perform the following steps on any node which has a working `kubectl` command:

- Creating a Kubernetes Service Secret
- Creating a Kubernetes Service Account
- Generating the ca.crt Certificate
- Creating a Kubernetes Configuration File for OAA
- Copying Files to the OAA-MGMT Container
- Validating the kubectl Command

## Creating a Kubernetes Service Secret

If you are using Kubernetes 1.24 release or later, then you need to create a secret for the Kubernetes service account using the following command:

```
kubectl create -f <WORKDIR>/create_svc_secret.yaml
```

For example:

```
kubectl create -f /workdir/OIRI/create_svc_secret.yaml
```

Here, `<WORKDIR>/create_svc_secret.yaml` has the following content:

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
 name: oaa-service-account
 namespace: <OAANS>
```

```
 annotations:
  kubernetes.io/service-account.name: "oaa-service-account"
```

## Creating a Kubernetes Service Account

Create a Kubernetes service account in the OAA namespace by using the following command:

```
kubectl apply -f <WORKDIR>/create_svc.yaml
```

For example:

```
kubectl apply -f /workdir/OAA/create_svc.yaml
```

Here, `<WORKDIR>/create_svc.yaml` has the following content:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: oaa-service-account
  namespace: <OAANS>
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: oaa-ns-role
  namespace: <OAANS>
rules:
- apiGroups: ["*"]
  resources: ["*","secrets"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: oaa-clusterrolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:persistent-volume-provisioner
subjects:
- namespace: <OAANS>
  kind: ServiceAccount
  name: oaa-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: oaa-clusteradmin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- namespace: <OAANS>
```

```
  kind: ServiceAccount
  name: oaa-service-account
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: oaa-rolebinding
  namespace: <OAANS>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: oaa-ns-role
subjects:
- namespace: <OAANS>
  kind: ServiceAccount
  name: oaa-service-account
```

## Generating the ca.crt Certificate

Obtain the Kubernetes certificate using the following commands:

Set up the environment variables for the OAA namespace, and a working directory.

```
OAANS=oaans
WORKDIR=/workdir/OAA
```

For Kubernetes releases up to 1.23, use the command:

```
TOKENNAME=`kubectl -n $OAANS get serviceaccount/oaa-service-account -o
jsonpath='{.secrets[0].name}'`
```

For Kubernetes releases 1.24 or later, use the commands:

```
TOKENNAME=oaa-service-account


TOKEN=`kubectl -n $OAANS get secret $TOKENNAME -o jsonpath='{.data.token}'|
base64 --decode`


kubectl -n $OAANS get secret $TOKENNAME -o jsonpath='{.data.ca\.crt}'| base64
--decode > $WORKDIR/ca.crt
```

## Creating a Kubernetes Configuration File for OAA

Generate a Kubernetes configuration file to instruct OAA on how to interact with `kubectl`.

Set up environment variables for the OAA namespace, and a working directory.

> **Note:**
>
> Ensure that you have set KUBECONFIG to the Kubernetes configuration file before starting.

```
OAANS=oaans
WORKDIR=/workdir/OAA
TOKEN=`kubectl -n $OAANS get secret $TOKENNAME -o jsonpath='{.data.token}'|
base64 --decode`
K8URL=`grep server: $KUBECONFIG | sed 's/server://;s/ //g'`


kubectl config --kubeconfig=$WORKDIR/oaa_config set-cluster oaa-cluster --
server=$K8URL --certificate-authority=$WORKDIR/ca.crt --embed-certs=true


kubectl config --kubeconfig=$WORKDIR/oaa_config set-credentials oaa-service-
account --token=$TOKEN


kubectl config --kubeconfig=$WORKDIR/oaa_config set-context oaa --user=oaa-
service-account --cluster=oaa-cluster


kubectl config --kubeconfig=$WORKDIR/oaa_config use-context oaa
```

These commands generate a file called `oaa_config` in the `<WORKDIR>` location. This file contains the Kubernetes cluster details.

## Copying Files to the OAA-MGMT Container

Copy the `ca.crt` (see Generating the ca.crt Certificate) and `oaa_config` (see Creating a Kubernetes Configuration File for OAA) files to the OAA-MGMT Container, using the following commands:

```
OAANS=oaans
WORKDIR=/workdir/OAA


kubectl cp $WORKDIR/ca.crt $OAANS/oaa-mgmt:/u01/oracle/scripts/creds


kubectl cp $WORKDIR/oaa_config $OAANS/oaa-mgmt:/u01/oracle/scripts/creds/
k8sconfig
```

From `oaa-mgmt`, run the following command:

```
chmod 400 /u01/oracle/scripts/creds/k8sconfig
```

## Validating the kubectl Command

Validate that the `kubectl` command works from inside the Kubernetes container by using the following command from the OAA-MGMT container:

```
kubectl get pod -n $OAANS
```

For example:

```
kubectl get pod -n oaans
```

The command should show you the running `oaa-mgmt` pod.

# Creating the Helm Configuration File

The OAA installation procedure is dependent on a `helmconfig` file being present. This file is almost always an empty file.

To create the `helmconfig` file:

1. Access the OAA-MGMT pod by using the following commands:

   ```
   kubectl -n oaans exec -it oaa-mgmt -- /bin/bash
   ```

2. From inside the OAA-MGMT pod, create a file called `/u01/oracle/scripts/creds/helmconfig` with the following contents:

   ```
   apiVersion: ""
   generated: "0001-01-01T00:00:00Z"
   repositories:
   - caFile: ""
     certFile: ""
     keyFile: ""
     name: idm-helm
     password: ""
     url:
     username: ""
   ```

3. Save the file.

# Obtaining the OAM Certificate

The OAM server is configured using SSL and this will most likely be terminated at the load balancer, especially if you have followed the instructions in this guide for deploying OAM. See Configuring Oracle Access Manager Using WDT. For OAA to successfully communicate with OAM, it has to trust OAM. To build this trust, you have to add the OAM certificate into the OAA trust store you will create in the next step.

You can obtain the load balancer certificate from using a browser, such as Firefox. However, the easiest way to obtain the certificate is to use the `openssl` command. The syntax of the command is as follows:

```
openssl s_client -connect LOADBALANCER -showcerts </dev/null 2>/dev/null|
openssl x509 -outform PEM >/workdir/OAA/LOADBALANCER.pem
```

For example:

```
openssl s_client -connect login.example.com:443 -showcerts </dev/null 2>/dev/
null|openssl x509 -outform PEM >/workdir/OAA/login.example.com.pem
```

The `openssl` command saves the certificate to a file called `login.example.com.pem` in the `/workdir/OAA` directory.

# Registering OAM TAP Partners

You must register OAM Tap partners that are used to enable OAA and OUA to authenticate with OAM in a more secure manner.

- Registering OAA as OAM TAP Partner
- Registering OUA as OAP Tap Partner

## Registering OAA as OAM TAP Partner

OAA should be registered with OAM as a Trusted Authentication Protocol (TAP) partner to ensure that OAM trusts OAA.

1. Connect to the OAM Administration pod using the following command:

   ```
   kubectl exec -n oamns -it accessdomain-adminserver -- /bin/bash
   ```

2. Start the WebLogic Scripting Tool (WLST) using the following command:

   ```
   /u01/oracle/oracle_common/common/bin/wlst.sh
   ```

3. Connect to the OAM Administration Server using the command:

   ```
   connect('WeblogicAdminUser','<WeblogicAdminPassword>','t3://
   <domain_name>domain-adminserver.<namespace>:<Admin_Port>'
   ```

   For example:

   ```
   connect('weblogic','<password>','t3://accessdomain-
   adminserver.oamns.svc.cluster.local:30701')
   ```

4. Run the following command to regsiter the OAA TAP partner:

   ```
   registerThirdPartyTAPPartner(partnerName = "<OAA_OAM_TAP_PARTNER>",
   keystoreLocation= "<K8_WORKDIR>/OAMOAAKeyStore.jks",
   password="<OAA_KEYSTORE_PWD>", tapTokenVersion="v2.0",
   ```

```
tapScheme="TAPScheme", tapRedirectUrl="<OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oam/pages/login.jsp")
```

For example:

```
registerThirdPartyTAPPartner(partnerName = "OAM-OAA-TAP",
keystoreLocation= "/u01/oracle/user_projects/workdir/OAMOAAKeyStore.jks",
password="keystorepassword", tapTokenVersion="v2.0",
tapScheme="TAPScheme", tapRedirectUrl="https://login.example.com:443/oam/
pages/login.jsp")
```

5. Copy the resulting keystore file to your work directory:

```
kubectl cp oamns/accessdomain-adminserver:/u01/oracle/user_projects/
workdir/OAMOAAKeyStore.jks /workdir/OAA
```

6. Copy the `KeyFile` to the Management Container using the command:

```
kubectl cp /workdir/OAA/OAMOAAKeyStore.jks oaans/oaa-mgmt:/u01/oracle/
scripts/creds/OAMOAAKeyStore.jks
```

# Registering OUA as OAP Tap Partner

OUA should be registered with OAM as a Trusted Authentication Protocol (TAP) partner to ensure that OUM trusts OAA. This is only required if you deploy OUA.

1. Connect to the OAM Administration pod using the following command:

```
kubectl exec -n oamns -it accessdomain-adminserver -- /bin/bash
```

2. Start the WebLogic Scripting Tool (WLST) using the following command:

```
/u01/oracle/oracle_common/common/bin/wlst.sh
```

3. Connect to the OAM Administration Server using the command:

```
connect('WeblogicAdminUser','<WeblogicAdminPassword>','t3://
<domain_name>domain-adminserver.<namespace>:<Admin_Port>'
```

For example:

```
connect('weblogic','<password>','t3://accessdomain-
adminserver.oamns.svc.cluster.local:30701')
```

4. Run the following command to regsiter the OAA TAP partner:

```
registerThirdPartyTAPPartner(partnerName="<OAA_OUA_TAP_PARTNER>",
keystoreLocation= "<K8_WORKDIR>/OAMOUAKeyStore.jks",
password="<OAA_KEYSTORE_PWD>", tapTokenVersion="v2.0",
tapScheme="TAPScheme", tapRedirectUrl="<OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oam/pages/login.jsp")
```

For example:

```
registerThirdPartyTAPPartner(partnerName="OAM-OUA-TAP", keystoreLocation=
"/u01/oracle/user_projects/workdir/OAMOUAKeyStore.jks",
password="keystorepassword", tapTokenVersion="v2.0",
tapScheme="TAPScheme", tapRedirectUrl="https://login.example.com:443/oam/
pages/login.jsp")
```

5. Copy the resulting keystore file to your work directory:

```
kubectl cp oamns/accessdomain-adminserver:/u01/oracle/user_projects/
workdir/OAMOUAKeyStore.jks /workdir/OAA
```

6. Copy the KeyFile to the Management Container using the command:

```
kubectl cp /workdir/OAA/OAMOUAKeyStore.jks oaans/oaa-mgmt:/u01/oracle/
scripts/creds/OAMOUAKeyStore.jks
```

# Creating the OAA Property File

The OAA deployment is dependent on the values in a property file. This file is used for creating the database schemas and to deploy OAA itself. The steps to create the file is performed inside the OAA-MGMT pod.

Complete the following steps to create the property file:

- Copying the Template File
- Updating the Property File

## Copying the Template File

Copy the supplied template files to the correct location:

```
cp /u01/oracle/installsettings/installOAA.properties /u01/oracle/scripts/
settings/installOAA.properties
```

```
cp /u01/oracle/installsettings/oaaoverride.yaml /u01/oracle/scripts/settings/
oaaoverride.yaml
```

## Updating the Property File

Update the values in the property file as described in this section. If you are modifying a parameter, ensure that the parameter is not put within a comment in the property file.

You should update the `/u01/oracle/scripts/settings/installOAA.properties` file.

- Common Deployment Parameters
- Database Parameters
- LDAP Parameters
- Ingress Parameters
- File Based Vault

- OCI Based Vault
- OUA Parameters
- Generic Parameters
- Sample installOAA.properties File

## Common Deployment Parameters

**Table 21-4 Common Deployment Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| common.deployment.mode | OAA<br>both<br>OUA | Types of deployments include:<br>• OAA for only OAA.<br>• both for OAA and RISK.<br>• OUA for OAA, RISK and OUA. |
| common.deployment.name | edg | The name of the deployment as it will appear in Helm.<br>**Note**: Do not use oaa because it is reserved for system use. |
| common.kube.namespace | oaans | The name of the namespace you will use to hold your OAA objects. |
| common.deployment.sslcert | /u01/oracle/scripts/creds/cert.p12 | The location inside the container where you have placed a commercial SSL certificate. |
| common.deployment.trustcert | /u01/oracle/scripts/creds/trust.p12 | The location inside the container where you have placed a commercial SSL certificate authority, this should include any intermediate certificates. |
| common.deployment.keystorepassphrase | - | The password you used when encrypting the commercial certificates. |
| common.deployment.truststorepassphrase | - | The password you used when encrypting the trust store. |
| common.deployment.keystorepassphrase | - | The password you used when encrypting the certificate store. |
| common.deployment.keystorepassphrase | - | The password you used when you created your server certificate. See #unique_789. |
| common.deployment.truststorepassphrase | - | The password you used when you created the trust store. See #unique_790. |
| common.deployment.overridefile | /u01/oracle/scripts/settings/oaaoverride.yaml | Ensure that this value is not put within a comment. |

**Table 21-4 (Cont.) Common Deployment Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| common.deployment.import.snapshot | true | The October release does not seed the database schemas. You must set this value to true in this release.<br>**Note**: After importing the snapshot, set this value to false to ensure that future upgrades do not overwrite the data. |
| common.deployment.import.snapshot.file | /u01/oracle/scripts/oarm-12.2.1.4.1-base-snapshot.zip | The October release does not seed the database schemas. You must set this value to the location of a snapshot file that exists in the management container. The file mentioned is the default value. If you want to download the latest version from My Oracle Support, see Doc ID 2723908.1. |
| oauth.domainname | OAADomain | Use the same value you added in OHS Rewrite Rules. See Step 4 of Configuring Oracle HTTP Server for Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator. |
| oauth.identityprovider | OAMIDSTORE | The name of the OAM ID Store. See Creating a Configuration File. |
| oauth.clientpassword | - | The password you want to use for OAM Oauth interactions. |
| oauth.adminurl | http://accessdomain-adminserver.oamns.svc.cluster.local:7001<br>http://iadadmin.example.com/ | The URL you use to access the Oracle Access Manager Administration Server. If OAM is inside the Kubernetes cluster, you should use the internal OAM service name otherwise you should use the OAM adminsitration URL. |
| oauth.basicauthzheader | - | The encoded username and password value of the OAM Administration user (*). |
| oauth.identityuri | https://login.example.com:443/ | The public entry point into OAM. |
| oauth.redirecturl | https://login.example.com:443/ | The public entry point into OAM. |
| install.global.serviceurl | https://login.example.com:443/ | The main entry point for the OAA application for runtime operations. |
| install.oaa-admin-ui.serviceurl | http://iadadmin.example.com:80 | The main entry point for the OAA application for administrative operations. |

(*) To encode a username and password, use the following command:

```
echo -n <LDAP_OAMADMIN_USER>:<LDAP_USER_PWD> | base64
```

For example:

```
echo -n oamadmin:mypassword | base64
```

## Database Parameters

**Table 21-5 Database Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| database.createschema | true | Set this value to create the OAA schemas. |
| database.host | dbscan.example.com | Set this value to the database server SCAN address. |
| database.port | 1521 | The database listener port. |
| database.svc | oaaedg.example.com | The name of the OAA database service. |
| database.syspassword | - | Set this value to the SYS password of the database. Ensure that the parameter is not put within a comment. |
| database.schema | EDG_OAA | The database schema prefix to use. |
| database.tablespace | EDG_OAA_OAA_TB | The name of the tablespace to be created. Include the schema prefix to aid manageability. |
| database.schemapassword | - | The password to be assigned to the OAA database schema. |
| database.name | - | Ensure that there is no value added to this parameter. |

## LDAP Parameters

**Table 21-6 LDAP Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| ldap.server | ldap://edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local:1389 | The host name of your LDAP Server. |
| ldap.username | cn=oudadmin | The DN of the LDAP Administrator. |
| ldap.password | myLDAPPassword | The password of the LDAP Administrator. |

**Table 21-6 (Cont.) LDAP Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| ldap.oaaAdminUser | cn=oaaadmin,cn=Users,dc=example,dc=com | The name of the OAA Administrator Account. |
| ldap.oaaAdminUserPwd | myOAAAdminPassword | Password of the ldap.oaaAdminUser. |
| ldap.adminRole | cn=OAA-Admin-Role,cn=Groups,dc=example,dc=com | The LDAP role used by OAA Administrators. |
| ldap.userRole | cn=OAA-App-User,cn=Groups,dc=example,dc=com | The LDAP Role used to identify users who can log in using OAA. |
| ldap.addExistingUsers | yes | If set to yes, existing LDAP users will automatically be added to the OAA User Group. |

## Ingress Parameters

If you use an Ingress controller, the installation creates the Ingress services for you. Provide these values to configure OAA for Ingress.

**Table 21-7 Ingress Parameters**

| Parameter | Value | Description |
|---|---|---|
| install.global.ingress.enabled | true | Set this value to true to create the Ingress services. |
| install.global.ingress.runtime.host | login.example.com | Set this value to the name of the virtual host used for runtime operations. |
| install.global.ingress.admin.host | iadadmin.example.com | Set this value to the name of the virtual host used for administration operations. |

## File Based Vault

You have two choices for creating a vault for OAA. You can use either a file based vault or an OCI vault. Oracle recommends you to use the OCI vault. However, if yo want to use the file based vault, you have to set the following parameters in the property file:

**Table 21-8 Parameters for the File Based Vault**

| Parameter | Value | Comments |
|---|---|---|
| vault.deploy.name | oaavault | The name of the vault. |
| vault.provider | fks | The type of vault. Fks is a file-based vault. |
| vault.fks.server | 0.0.0.0 | The name or IP address of the NFS server. |

**Table 21-8 (Cont.) Parameters for the File Based Vault**

| Parameter | Value | Comments |
|---|---|---|
| `vault.fks.path` | `/exports/IAMPVS/oaavaultpv` | The NFS export name of the file system. |
| `vault.fks.key` | - | The password to be used for accessing the vault. |

## OCI Based Vault

If you want to use an OCI based vault, you have to set the following parameters in the property file:

> **Note:**
>
> Each of the `vault.oci` parameters listed in Table 21-9 should be encoded in Base64. To encode in Base64, run the following command:
>
> ```
> echo -n "value" | base64
> ```

**Table 21-9 Parameters for the OCI Based Vault**

| Parameter | Value | Comments |
|---|---|---|
| `vault.deploy.name` | `oaavault` | The name of the vault. |
| `vault.provider` | `oci` | The type of vault. `oci` is an oci-based vault. |
| `vault.oci.uasoperator` | - | To obtain this value, encode the value of the API key that you downloaded at the time of creating the vault. See Creating a Vault. |
| `vault.oci.tenancyId` | - | To obtain this value, log in to the OCI console, navigate to **Profile** and click **Tenancy**. Use the **OCID** value encoded as Base64. |
| `vault.oci.userId` | - | To obtain this value, log in to the OCI console, navigate to **Profile** and click **Username**. Use the **OCID** value encoded as Base64. |
| `vault.oci.compartmentId` | - | To obtain this value, log in to the OCI console, navigate to **Identity and Security** and click **Compartments**. Select the compartment in which you created the vault and use the **OCID** value encoded as Base64. |

**Table 21-9 (Cont.) Parameters for the OCI Based Vault**

| Parameter | Value | Comments |
|---|---|---|
| `vault.oci.fpId` | - | To obtain this value, log in to the OCI console, navigate to **Profiles**, select **User Settings** and then click **API Keys**. Use the value of the fingerprint for the API Key you created earlier. See Creating a Vault. |
| `vault.oci.vaultId` | - | To obtain this value, log in to the OCI console, navigate to **Identity and Security** and select **Vault**. Click the vault you created earlier. See Creating a Vault. Use the **OCID** value encoded as Base64. |
| `vault.oci.keyId` | - | To obtain this value, log in to the OCI console, navigate to **Identity and Security**, select **Vault**, and then click the vault you created earlier. See Creating a Vault. Click the key you created earlier. For example, **masterkey**. Use the **OCID** value encoded as Base64. |

## OUA Parameters

You must set the following parameters when you deploy Oracle Universal Authenticator.

**Table 21-10 OUA Parameters**

| Parameter | Value | Comments |
|---|---|---|
| `oua.tapAgentName` | `OAM-OUA-TAP` | The Name of the OUA TAP agent created in Registering OUA as OAP Tap Partner. |
| `oua.tapAgentFileLocation` | `/u01/oracle/scripts/creds/OAMOUAKeyStore.jks` | The location of the Keystore File inside the OAA Management container. |
| `oua.tapAgentFilePass` | `keystorepassword` | The Password for the Keystore File created in Registering OUA as OAP Tap Partner. This password value needs to be bas64 encoded. For example, `echo -n keystorepassword \| base64`. |
| `oua.oamRuntimeEndpoint` | `https://login.example.com:443/` | The public entry point into OAM. |

## Generic Parameters

**Table 21-11 Generic Parameters**

| Parameter | Value | Comments |
|---|---|---|
| install.global.repo | iad.ocir.io/mytenancy | The name of the container registry. |
| install.global.imagePullSecrets\[0\].name | regcred | The name of the Kubernetes secret with the container registry credentials. |
| install.global.imagePullSecrets\[1\].name | gitcred | The name of the Kubernetes secret with the GitHub credentials. |
| install.global.image.tag | 12.2.1.4.1-20220127 | The version of OAA you want to install. |
| install.global.oauth.logouturl | https://login.example.com:443/oam/server/logout | The OAM logout URL. |
| install.global.uasapikey | myAPIpassword | The password you want to use for the OAA REST APIs. |
| install.global.policyapikey | myAPIpassword | The password you want to use for the Policy REST APIs. |
| install.global.factorsapikey | myAPIpassword | The password you want to use for the Factors REST APIs. |
| install.global.riskapikey | myAPIpassword | The password you want to use for the Risk REST APIs. |
| install.global.drssapikey | myAPIpassword | The password you must use for the DRSS REST APIs. |
| install.service.type | ClusterIP<br>NodePort | The type of Kubernetes service to create. If you are using ingress, use ClusterIP or else use NodePort. |
| install.oaa-admin-ui.service.type | ClusterIP<br>NodePort | The type of Kubernetes service to create. Use ClusterIP if you are using ingress. Otherwise, use NodePort. |

## Sample installOAA.properties File

```
################################### 1. Common Deployment
configuration##############################################
#Common configuration options
#If enabled and set to true, helm installation will only display generated
values and will not actually perform the installation.
#common.dryrun=true
#Name of the helm deployment. It is unique per kubernetes cluster and
namespace. Should be all lowercase.
common.deployment.name=edg
#Override file to override any char values. Must be in yaml format.
common.deployment.overridefile=/u01/oracle/scripts/settings/oaaoverride.yaml
```

```
#Kubernetes context if there are multiple contexts available
#common.kube.context=idmoci
#Kubernetes deployment namespace where all the services will be installed.
common.kube.namespace=oaans
#Certificate store used in all the services
common.deployment.sslcert=/u01/oracle/scripts/creds/cert.p12
#Trust store used in all the services
common.deployment.trustcert=/u01/oracle/scripts/creds/trust.p12
#Passphrase for cert.p12 file. If the value for the passphrase is not present
in properties below, it will be prompted during installation.
common.deployment.keystorepassphrase=myPassword
#Passphrase for trust.p12 file. If the value for the passphrase is not
present in properties below, it will be prompted during installation.
common.deployment.truststorepassphrase=myPassword
#If the flag is enabled then trust certificate in the JRE truststore
common.deployment.importtruststore=true
#The flag to disable the generation of secret in the keystore provided in the
property common.deployment.sslcert. Following secret keys are generated and
are required for installation to work which can also be pre-seeded in the
keystore.
#spui-enckey , aes256_config_key_alias and aes256_db_key_alias
common.deployment.generate.secret=true
#Deployment mode. Possible values are OAA, Risk or Both. Default mode is Both
which will install OAA integrated with Risk.
common.deployment.mode=Both
#Base64 encoded config key from the migrating system. If enabled the value
will be placed in the vault and used for migration of legacy data
#common.migration.configkey=
#Base64 encoded db key from the migrating system. If enabled the value will
be placed in the vault and used for migration of db data.
#common.migration.dbkey=
#If the integation is required with OIM set the following property to true.
This also enables the forgot password functionality.
#common.oim.integration=true
#Key for Apple push notification service. Only needed when push factor is
enabled.
#common.deployment.push.apnsjksfile=
#Deprecated. By default policy snapshot import is not enabled. Kindly refer
document to import it post installation.
common.deployment.import.snapshot=true
################################### 2. Database
configuration#############################################
# Database setting for OAA
#The installation supports the following ways to install and configure the
database:
#1. Installing database along with installing services on kubernetes cluster
#2. Installing database outside the installation of services and providing
the database configuration below.
#In case of 2, set database.createschema=false
#If set to true schema will be created along with the installation.
database.createschema=true
#Host IP or name where Oracle DB is running. If hostname is provided then it
should be resolvable from the container.
database.host=db-scan.example.com
#Database port
database.port=1521
```

```
#Database sysdba user required to create the schema. If database.createschema
is set to false, the property is not required.
database.sysuser=sys
#Sys password will not be prompted if value is provided for the property
database.syspassword. If database.createschema is set to false, the property
is not required.
database.syspassword=sysPassword
#Schema name to be created in the database. The value is required.
database.schema=EDG_OAA
#Name of the tablespace. If database.createschema is set to false, the
property is not required.
database.tablespace=EDG_OAA_TBS
#Schema password will not be prompted if value is provided for the property
database.schemapassword
database.schemapassword=schemaPassword
#Database service name
database.svc=edgoaa.example.com
#Name of the database. In case of OCI database service, the name is present
in the OCI console. If name is not present, use value of database.svc
property.
database.name=iamdb1

################################### 3. OAUTH
configuration#############################################
#Oauth setting for OAA. Oauth is required for enabling spui , admin and fido.
# If oauth.enabled is set as false, install.spui.enabled , install.oaa-admin-
ui.enabled and install.fido.enabled should also be set to false otherwise
helm installation
# will fail. Setting the property to true enables OAM Oauth integration
during installation of services.
oauth.enabled=true
#Create OAuth Domain in OAM
oauth.createdomain=true
#Create OAuth Resource in OAM
oauth.createresource=true
#Create OAuth client in OAM
oauth.createclient=true
#OAuth domain name used when creating the OAuth domain in OAM
oauth.domainname=OAADomain
#OAuth domain identity provider configured for OAuth in OAM
oauth.identityprovider=OAMIDSTORE
#OAuth client name
oauth.clientname=OAAClient
#Grants for OAuth client created during installation. CLIENT_CREDENTIALS is
required for validation to pass.
oauth.clientgrants="PASSWORD","CLIENT_CREDENTIALS","JWT_BEARER","REFRESH_TOKEN
","AUTHORIZATION_CODE","IMPLICIT"
#OAuth Client Type of new OAuth client
oauth.clienttype=PUBLIC_CLIENT
#Client password must conform to regex ^[a-zA-Z0-9.\-\/+=@_ ]*$ with a
maximum length of 500
oauth.clientpassword=myPassword
#OAuth resouce name for new OAuth resource
oauth.resourcename=OAAResource
#Default score of OAuth resource
oauth.resourcescope=viewResource
```

```
#Post authentication redirecturl is required. Used for validating
configuration of OAuth services in OAM by generating a access token.
oauth.redirecturl=https://login.example.com:443
#Application id protected by oauth. The value can be any valid string. It is
required to setup runtime integration between OAM and OAA.
oauth.applicationid=edg
#OAM Admin URL where OAuth API are enabled.
oauth.adminurl=http://accessdomain-adminserver.oamns.svc.cluster.local:7001
#Base64 encoded authorization header of OAM Admin server
oauth.basicauthzheader=EncodedOAMPassord
#OAM Managed server providing runtime support for OAuth Services
oauth.identityuri=https://login.example.com:443

################################### 4. Vault
configuration##########################################
#Following vaules are possible for vault.provider : oci , fks
#If oci is the provider the oci related configuration may be required to
perform the vault initialization
#Name to be used in vault for this deploymemt. If name is already present in
the vault, it will be reused.
vault.deploy.name=oaavault
#Flag to control the vault creation. If vault is initialized outside the
installation, it should be set to false.
vault.create.deploy=true
#Provider type of vault. Supported provider types are oci, fks.
vault.provider=fks

#oci vault configuration required for the vault. Check vault documentation on
how to obtain value for following properties.
#vault.oci.uasoperator=
#vault.oci.tenancyId=
#vault.oci.userId=
#vault.oci.fpId=
#vault.oci.compartmentId=
#vault.oci.vaultId=
#vault.oci.keyId=

#fks related configuration. Check the documentation on how to obtain value
for properties below.
#NFS server ip or resolvable hostname.
vault.fks.server=dbdevfssmnt-shared01.dev2fss1iad.databasede2iad.oraclevcn.com
#Path on NFS server to be mapped to folder in the running containers.
vault.fks.path=/export/IAMPVS/oaavaultpv
#Base64 encoded vault password.
vault.fks.key=TWFuYWdlcjE=
#The value in this property need to be same as the value passed through the
helm chart. Do not change it
vault.fks.mountpath=/u01/oracle/service/store/oaa

################################### 5. Chart
configuration##########################################
#Note: Any property  that starts with install. will be provided as input to
the helm chart using --set parameter.
#Container image repositories where the images can be pulled by the cluster
nodes.
install.global.repo=iad.ocir.io/mytenancy
```

```
#Kubernetes secret reference to be used while pulling the docker images from
the protected docker registries.
install.global.imagePullSecrets\[0\].name=regcred
install.global.imagePullSecrets\[1\].name=gitcred
#Chart Database properties
install.riskdb.service.type=ExternalName
#Image tags to be used
install.global.image.tag=12.2.1.4.1_20220419
#Oauth logout URL
install.global.oauth.logouturl=https://login.example.com:443/oam/server/logout

#Installation api key
#Rest Api key for OAA service.
install.global.uasapikey=myAPIpassword
#Rest Api key for policy service.
install.global.policyapikey=myAPIpassword
#Rest API key for all factor services.
install.global.factorsapikey=myAPIpassword
#Rest API key for risk and risk customer care services.
install.global.riskapikey=myAPIpassword
#Rest API key for DRSS service.
install.global.drssapikey=myAPIpassword

#In case of OCI vault, the following configuration can be overridden if
provided for read-only users during helm installation.
#If the value is not provided in the properties below then it will be picked
from vault section.
#All the values in the section below is optional.
#install.global.vault.mapId=b2NpZDEudmF1bHRzZWNyZXQub2MxLmlhZC5hbWFhYWFhYWRjeH
l1dXFha2xmejVkbW9rcnpkajZva2Rtdmt0bGp2Y2tyZGIyd3B0emt6bHlkbTN1emEK
#install.global.vault.oci.uasoperator=
#install.global.vault.oci.tenancyId=
#install.global.vault.oci.userId=
#install.global.vault.oci.fpId=

################################### 6. Optional
configuration##########################################
##Ingress properties that can be used to enable ingress for services begin
deployed.
#Deprecated property. Will be removed in future. Use
install.global.ingress.enabled instead.
#install.ingress.enabled=true
#Enable ingress for all services.
install.global.ingress.enabled=true
#Ingress resource hostname. Should be all lowercase. If provided, dedicated
hostname will be used to access the deployment. Otherwise can use any
hostname or IP Address.
#Following configurations are deprecated. Use
install.global.ingress.runtime.host and install.global.ingress.admin.host for
specifying runtime and admin host
#install.global.ingress.hosts\[0\].host=oaainstall-host
#install.global.ingress.hosts\[1\].host=oaaadmin-host

#Runtime host used for accessing runtime services including all factors,
oaa ,spui and risk.
install.global.ingress.runtime.host=login.example.com
```

```
#Admin host used for accessing admin, policy and risk-cc services.
install.global.ingress.admin.host=iadadmin.example.com

##Following properties are picked from database related properties if not
present below.
#install.global.dbhost=
#install.global.dbport=
#install.global.dscredentials=
#install.global.dbservicename=

##Following properties are picked from oauth related properties if not
present below.
#install.global.oauth.oidcidentityuri=
#install.global.oauth.oidcaudience=
#install.global.oauth.oidcclientid=

#if load balancer/ingress url is present, then configure the url here. All UI
service will be behind this load balancer/ingress.
#In case ingress installation is set to true, the appropriate service url
will be fetch after ingress installation
# and will be used as service url. If provided, service url from the property
below will have higher priority.
install.global.serviceurl=https://login.example.com:443
#Service URL of oaa admin, if different from the service url of global oauth.
install.oaa-admin-ui.serviceurl=http://iadadmin.example.com:80

#If oauth.enabled is set to false, uncomment following properties. Also when
deployment mode is Risk spui, fido and oaa-kba services are not required.
#install.spui.enabled=false
#install.fido.enabled=false
#install.oaa-admin-ui.enabled=false
#install.oaa-kba.enabled=false

#Also authentication factor Services are enabled by default. To disable them
uncomment the lines below.
#When deployment mode is Risk, the following configurations is not required.
#install.totp.enabled=false
#install.push.enabled=false
#install.sms.enabled=false
#install.yotp.enabled=false
#install.email.enabled=false

#Default service type for services is NodePort. When deployment mode is Risk
following service are not deployed :
#OAA, SPUI, All Factors (fido, push, yotp, email ,sms, totp and kba)
#install.service.type=ClusterIP
#install.oaa-admin-ui.service.type=ClusterIP
#install.oaa-policy.service.type=ClusterIP
#install.spui.service.type=ClusterIP
#install.totp.service.type=ClusterIP
#install.fido.service.type=ClusterIP
#install.push.service.type=ClusterIP
#install.email.service.type=ClusterIP
#install.sms.service.type=ClusterIP
#install.yotp.service.type=ClusterIP
#install.risk.service.type=ClusterIP
```

```
#install.oaa-kba.service.type=ClusterIP
#install.risk.riskcc.service.type=ClusterIP
#Properties used to install ingress using the ingress chart present in
helmcharts folder

##################################### 7. Ingress
configuration#############################################
#To install the ingress controller along with the services set the following
flag to true.
ingress.install=false
#Kubernetes name space which will be used to install ingress
ingress.namespace=ingress-nginx
#Admissions controller can be installed seperately.
#Ingress admissions name is not present the
controller.admissionWebhooks.enabled will be set to false in the nginx
ingress chart.
#ingress.admissions.name=ingress-nginx-controller-admission
#Ingress class name that would be used for installation. Must not be existing
ingress.class.name=ingress-nginx-class
ingress.service.type=NodePort

#anything starting with ingress.install can be additionally supplied to set
the ingress chart value.
#ingress.install.releaseNameOverride=base

##################################### 8. OAA management
configuration#############################################
#NFS volumes
#install.mount.config.path=<NFS_CONFIG_PATH>
#install.mount.config.server=<NFS_CONFIG_SERVER>
#install.mount.creds.path=<NFS_CREDS_PATH>
#install.mount.creds.server=<NFS_CREDS_SERVER>
#install.mount.logs.path=<NFS_LOGS_PATH>
#install.mount.logs.server=<NFS_LOGS_SERVER>
#OAA mgmt chart release name, default can be used for most installations
install.mgmt.release.name=oaamgmt
#Location of kube credentials to use for installation.
#If not provide credentials in $KUBECONFIG or ~/.kube/config will be used
#install.kube.creds=<LOCAL_PATH>/<KUBE_CREDS>
#SSL certificates
#common.local.sslcert=<LOCAL_PATH>/<LOCAL_CERT_FILE>
#common.local.trustcert=<LOCAL_PATH>/<LOCAL_TRUST_FILE>
common.deployment.import.snapshot=true
common.deployment.import.snapshot.file=/u01/oracle/scripts/oarm-12.2.1.4.1-
base-snapshot.zip

##################################### 9. OUA Configuration ##################
# OUA properties needed for integration with OAA.
# OUA Agent Name. Has to be the same as the TAP Partner in OAM.
oua.tapAgentName=OAM-OUA-TAP
# OUA Password for the jks file (Base 64 Encoded)
oua.tapAgentFilePass=keystorepassword
# OUA JKS File Location
oua.tapAgentFileLocation=/u01/oracle/scripts/creds/OAMOUAKeyStore.jks
# OUA OAM Runtime Endpoint
oua.oamRuntimeEndpoint=https://login.example.com:443
```

```
################################### 10. LDAP
configuration##########################################
ldap.server=ldap://edg-oud-ds-rs-lbr-ldap.oudns.svc.cluster.local:1389
ldap.username=cn=oudadmin
ldap.password=******
ldap.oaaAdminUser=cn=oaaadmin,cn=Users,dc=edg,dc=com
ldap.adminRole=cn=OAA-Admin-Role,cn=Groups,dc=edg,dc=com
ldap.userRole=cn=OAA-App-User,cn=Groups,dc=edg,dc=com
ldap.oaaAdminUserPwd=******
ldap.addExistingUsers=yes
```

# Creating the OAA Override File

The OAA Override file is used to determine the number of each type of container that is started. In a highly available deployment, there should be a minimum of two for each container type.

To set the number of containers to be started, update the `/u01/oracle/scripts/settings/oaaoverride.yaml` file to increase the `replicaCount` for each type of container to the required quantity.

You can use this file to specify the resource requirements. By declaring resource requirements, you ensure that a particular OAA pod is started only on a worker node that has sufficient capacity to service a pod with these resource requirements. You can also use the HorizontalPodAutoscaler resource (see Deploying the Kubernetes HorizontalPodAutoscaler Resource), which allows the number of pods of a given type to scale up and down as demand dictates.

> **Note:**
>
> Add any missing entries that you require.

The example below shows resource requirements added for the `oaa` and `spui` containers.

Sample `oaaoverride.yaml` file:

```
#override file for ooa installation

#if database is external to the cluster set the flag to ExternalName
riskdb:
  service:
    type: ExternalName

#replica count of oaa service
replicaCount: 2

#The following properties define the dependency spui service and can be
overridden here.
spui:
  resources:
    requests:
```

```
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency totp service and can be
overridden here.
totp:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency yotp service and can be
overridden here.
yotp:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency fido service and can be
overridden here.
fido:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency oaa-admin-ui service and can
be overridden here.
oaa-admin-ui:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency email service and can be
overridden here.
email:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency push service and can be
overridden here.
push:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
```

```
  replicaCount: 2

#The following properties define the dependency sms service and can be
overridden here.
sms:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency oaa-policy service and can be
overridden here.
oaa-policy:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the defaults of risk and riskcc services.
risk:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
  riskcc: //Need to remove
    replicaCount: 2

riskcc:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
#
#The following properties define the defaults of customfactor service.
customfactor:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
#
#The following properties define the defaults of oaa-kba service.
oaa-kba:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#
#The following properties define the defaults of oaa-drss service.
oaa-drss:
```

```
resources:
  requests:
    cpu: 200m
    memory: "1Gi"
replicaCount: 2
```

# Creating the Database Schemas

Oracle Advanced Authentication automatically creates the schemas in the database.

# Deploying Oracle Advanced Authentication

To deploy the OAuth application, you should perform the steps inside the OAA-MGMT pod.

1. Edit the `/u01/oracle/scripts/settings/installOAA.properties` file.

   Set the value of the `database.createschema` property to `false`.

2. Run the following command:

   ```
   cd /u01/oracle/scripts/settings/
   ```

   ```
   /u01/oracle/OAA.sh -f installOAA.properties
   ```

   This command will deploy the OAA and OARM containers. It will also set up OAuth in the OAM Server. If the command fails, rectify the issue and edit the `/u01/oracle/logs/status.info` file to change the value from `false` to `true`. You can then rerun the command.

# Resolving Timeouts

Your deployment may fail if it takes longer than the expected time to pull the container images from the container registry. You will see the error in the deployment log.

The error appears as follows:

```
NAME: oaainstall
LAST DEPLOYED: Thu Apr 29 20:24:09 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
Error: timed out waiting for the condition
TEST SUITE:     oaainstall-email-sanity-check
Last Started:   Thu Apr 29 20:24:12 2021
Last Completed: Thu Apr 29 20:29:12 2021
Phase:          Failed
NOTES:
Get the Oracle Advance Authentication(OAA) Service URL by running these commands:
  bash -c 'export NODE_PORT=$(kubectl get --namespace default -o
jsonpath="{.spec.ports[0].nodePort}" services oaainstall-oaa) && export NODE_IP=$
(kubectl get nodes --namespace default -o
jsonpath="{.items[0].status.addresses[0].address}") && echo "" && echo
https://$NODE_IP:$NODE_PORT/health'
Helm test failed
Fail to install OAA. Please check the log file
```

You can run the following command to check the status of the pods:

```
kubectl get pods
```

The output appears as follows:

```
NAME                                    READY   STATUS          RESTARTS   AGE
edg-email-66944cd9d8-7vp7g              1/1     Running         0          41m
edg-email-66944cd9d8-t8gcc              1/1     Running         0          41m
edg-fido-66b969fb6f-9zcnr               1/1     Running         0          41m
edg-fido-66b969fb6f-mzj2n               1/1     Running         0          41m
edg-oaa-86cd6b4ff6-pxrmm                1/1     Running         0          41m
edg-oaa-86cd6b4ff6-svlpk                1/1     Running         0          41m
edg-oaa-admin-ui-764c555cf4-6xtzv       1/1     Running         0          41m
edg-oaa-admin-ui-764c555cf4-rxs2c       1/1     Running         0          41m
edg-oaa-kba-6d6d7bcd59-rwm5q            1/1     Running         0          41m
edg-oaa-policy-7cc99c7bc5-8tpzz         1/1     Running         0          41m
edg-oaa-policy-7cc99c7bc5-rsvqr         1/1     Running         0          41m
edg-push-794c8d89d-6xcjt                1/1     Running         0          41m
edg-push-794c8d89d-b9xl4                1/1     Running         0          41m
edg-risk-768fddd6b5-6k4p4               1/1     Running         0          41m
edg-risk-768fddd6b5-wrcjh               1/1     Running         0          41m
edg-risk-cc-77fcd64cb5-7nfft            1/1     Running         0          41m
edg-risk-cc-77fcd64cb5-bfzfn            1/1     Running         0          41m
edg-sms-7bdbf58d66-kk6qq                1/1     Running         0          41m
edg-sms-7bdbf58d66-lp8jj                1/1     Running         0          41m
edg-spui-59649cd778-5rbgb               1/1     Running         0          41m
edg-spui-59649cd778-nkl8s               1/1     Running         0          41m
edg-totp-7dc888b994-m854x               1/1     Running         0          41m
edg-totp-7dc888b994-zgfkw               1/1     Running         0          41m
edg-yotp-569fbbc7df-27zgm               1/1     Running         0          41m
edg-yotp-569fbbc7df-9qsx7               1/1     Running         0          41m
oaa-mgmt                                1/1     Running         0          45m
```

If you give it time, the pods will eventually start. You can avoid the timeout by adjusting its value in the `/u01/oracle/helmcharts/oaa/values.yaml` file, inside the OAA Management container. The section are:

```
test:
  # test.image -- image name that will be used to test sanity of installation.
  image: shared/oracle/linux:8-slim
  # test.timeoutsecs time for which sanity tests will run before timing out.
  timeoutsecs: 480
  # test.waitsecs time interval between sanity checks.
  waitsecs: 50
```

# Configuring Email/SMS Servers and Automatic User Creation

OAA has built-in email/SMS integration. You have to configure OAA to use the Email SMTP client and the SMS client.

In addition you can set SPUI to automatically create users in the OAA database that already exist in LDAP when a user first logs into SPUI. This is an additional configuration property that can be set.

You can configure the OAA Email/SMS client using cURL. The `curl` command is:

```
curl --location --insecure --request PUT '<OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oaa/runtime/config/property/v1' \
```

```
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Encoded OAA User>' \
-d '[
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientURL",
        "value": "<UMS_SERVER_URL>"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientName",
        "value": "<UMS_ADMIN_USER>"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientPass",
        "value": "<UMS_ADMIN_PASSWORD>"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientURL",
        "value": "<UMS_SERVER_URL>"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientName",
        "value": "<UMS_ADMIN_USER>"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientPass",
        "value": "<UMS_ADMIN_PASSWORD>"
    }
    {
        "name": "oaa.default.spui.pref.runtime.autoCreateUser",
        "value": "true"
    }

]'
```

For example:

```
curl --location --insecure --request PUT 'http://login.example.com/oaa/
runtime/config/property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic
b2FhLW9hYTphcGlrZXl0b2Jlc2V0ZHVyaW5naW5zdGFsbGF0aW9u' \
-d '[
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientURL",
        "value": "http://governancedomain-cluster-soa-
cluster.oigns.svc.cluster.local:8001/ucs/messaging/webservice
"
    },
```

```
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientName",
        "value": "weblogic"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientPass",
        "value": "password"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientURL",
        "value": "http://governancedomain-cluster-soa-
cluster.oigns.svc.cluster.local:8001/ucs/messaging/webservice
"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientName",
        "value": "weblogic"
    },
    {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientPass",
        "value": "password"
    }

]'
```

# Validating OAA

This is a simple method of validating the integration of OAM and OAA. Oracle recommends that you roll back all the steps after completing the validation.

The validation procedure comprises the following steps:

- Creating the HTML Test Page
- Creating an Oracle Resource for the Test Page
- Creating a Test User
- Validating the Deployment
- Validating OAA
  After deploying OAA, ensure that OAA is working by accessing the OAA Administration Console.

## Creating the HTML Test Page

Create a simple HTML test page on each Oracle HTTP server. This test page provides a simple resource that can be protected by OAA.
On the OHS Servers located in the *WEB_DOMAIN_HOME*`/config/fmwconfig/components/OHS/<instance_name>/htdocs` directory, create a test page called `test_page.html` with the following content:

1. On the OHS servers located in the *WEB_DOMAIN_HOME*/config/fmwconfig/components/OHS/<instance_name>/htdocs directory, create a test page called test_page.html with the following content:

```
<!DOCTYPE html>
<html>
<body>

<h1>This is a Test Page</h1>

</body>
</html>
```

2. Save the file and restart the Oracle HTTP servers.

## Creating an Oracle Resource for the Test Page

After you create the test page, create a resource in OAM so that WebGate is aware of it and protects it using OAA.

1. Log in to the OAM Administration Console using the http://iadadmin.example.com/oamconsole URL.
2. In the OAM Console, select **Access Manager** and click **Application Domains**.
3. Click **Search** to view the defined application domains.
4. Click the **IAM Suite** domain.
5. Select the **Resources** tab, click **Create**, and enter the following information:
   - **Type**: HTTP
   - **Description**: OAA Resource
   - **Host Identifier**: IAMSuiteAgent
   - **Resource URL**: /test_page.html
   - **Operations**: ALL
   - **Protection Level**: Protected
   - **Authentication Policy**: OAA_MFA-Policy
   - **Authorization Policy**: Protected Resource Policy

   Click **Apply** to save the changes.

No restarts are necessary but it can take a short time for the changes to propagate.

## Creating a Test User

Create a test user in Oracle Unified Directory (OUD). Assign this user to the OAA-APP-User group. The user should also have a valid email address if you want to receive One Time Pin Codes through email, and a description if you have configured Oracle Mobile Authenticator.

You can use the following sample ldif file to create a test user:

```
dn: cn=oaauser,cn=Users,dc=example,dc=com
changetype: add
objectClass: orclUserV2
```

```
objectClass: oblixorgperson
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: oblixPersonPwdPolicy
objectClass: orclAppIDUser
objectClass: orclUser
objectClass: orclIDXPerson
objectClass: top
givenName: oaauser
uid: oaauser
orclIsEnabled: ENABLED
sn: oaauser
userPassword: password
mail: oaauser@example.com
mobile:
orclSAMAccountName: oaauser
cn: oaauser
description: oaauser
obpasswordchangeflag: false
obpsftid: true

dn:cn=OAA-App-User,cn=Groups,dc=example,dc=com
changetype: modify
add: uniqueMember
uniqueMember: cn=oaauser,cn=Users,dc=example,dc=com
```

## Validating the Deployment

To validate the deployment:

1. Go to `https://login.example.com/test_page.html`.
2. When prompted for a username and password, enter your 'oaa' test user that you created earlier. See Creating a Test User.
3. Select **Email** to receive the one time pin through email.
4. When prompted, enter the one time pin you received through email.

   Your test page appears.

## Validating OAA

After deploying OAA, ensure that OAA is working by accessing the OAA Administration Console.

To validate OAA, log in to the OAA Administration Console using the `http://iadadmin.example.com/oaa-admin` URL.

You should be redirected to the OAM Credential Collection page. Enter the user name and password for oaaadmin. If all is well, you will be asked to accept the OAuth authentication. When you click **Allow**, you will see the OAA Administration page.

# Configuring Oracle Universal Authentication

This section describes various tasks related to configuring Oracle Universal Authentication.

> **Note:**
>
> Currently, Oracle Universal Installer only supports Microsoft Windows Clients.



# Microsoft Entra Domain

The following prerequisites are required while using Microsoft Windows clients to login with OUA.

- You must have a Microsoft Entra Domain Services managed domain.
- You must create a Microsoft Windows user in the domain if you want to use OUA.

Administrators must have a working knowledge of Microsoft Entra before using OUA. The following document does not contain instructions on how to setup a Microsoft Entra Domain, LDAP directories or user accounts.

**Prerequisite Configurations for Windows Desktop Environment**
The following prerequisites are required for Microsoft Windows clients to login with OUA and is applicable to any Windows Desktop or Server where the OUA client application is installed:

- A Microsoft Windows Desktop client running Windows 10 or 11, or a Windows 2016 or 2019 Server.
- The Microsoft Windows Desktop or Server must have joined the Windows domain in Microsoft Entra.
- The Microsoft Windows user who wants to login via OUA must be able to login to the Windows domain with a valid username and password.
- Chrome v88+, Microsoft Edge v92+ is required for the OUA SSO Browser Extension.
- Administrator credentials to install the OUA client application.

**OAM User Accounts**
The Microsoft Windows user must have a user account in the User Identity Store used by Oracle Access Management (OAM). The user must be able to login with Single-Sign On (SSO) to an application protected with OAM.

# Enabling OAM Persistent Login

OUA requires persistent login to be enabled in OAM. For more information on persistent login, see Understanding Persistent Login. Perform the following steps to enable persistent login in OAM:

1. Run the following command to connect to the to the OAM Administration server:

```
kubectl exec -n <OAMNS> -ti <OAM_DOMAIN_NAME>-adminserver -- /bin/bash
```

For Example:

```
kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
```

2. Run the following command to start WLST:

```
/u01/oracle/oracle_common/common/bin/wlst.sh
```

3. Run the following command to connect to the domain:

```
connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>')
```

For Example:

```
connect('weblogic','MyPassword')
```

4. Run the following WLST command to configure persistent login:

```
configurePersistentLogin(enable="true",validityInDays="30",
maxAuthnLevel="2", userAttribute="obPSFTID")
```

A `SUCCESS` messgae will be displayed.

5. Exit WLST using the following command

```
exit()
```

# Restarting the OAM Domain

You should restart the OAM domain to enable the OAM Managed Servers to use the OAA plug-in.

Use the following commands to restart the domain:

```
kubectl -n <OAMNS> patch domains <OAM_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
```

After all the Kubernetes pods (with the exception of the helper pod) in the namespace have stopped, you can restart the domain by using the following command:

```
kubectl -n <OAMNS> patch domains <OAM_DOMAIN_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "If_Needed" }]'
```

> **Note:**
>
> Check that all the Kubernetes pods (with the exception of the helper pod) in the namespace have stopped by using the following command:
>
> ```
> kubectl -n <OAMNS> get all
> ```

All the Kubernetes pods (with the exception of the helper pod) in the namespace will be stopped when there are no entries for the Administration Server or the Managed Servers. For example:

```
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "Never" }]'
```

```
kubectl -n oamns patch domains accessdomain --type='json' -p='[{"op":
"replace", "path": "/spec/serverStartPolicy", "value": "If_Needed" }]'
```

# Configuring Forgotten Password

OUA has the facility to allow the resetting of passwords in case they are forgotten. To reset your password you need to set two OUA configuration parameters, one which redirects users the forgotten password flow and the other which directs users to the change password flow

The following curl command is invoked to set the two OUA configuration parameters:

```
curl --location --insecure --request PUT
      '<OAM_LOGIN_LBR_PROTOCOL>://
<OAM_LOGIN_LBR_HOST>:<OAM_LOGIN_LBR_PORT>/oaa/runtime/config/property/v1'
      \--header 'Content-Type: application/json'
      \--header 'Authorization: Basic <Encoded OAA User>'
      \-d '[
      {
      "name": "oua.drss.password.reset.forgoturl",
      "value": "<FORGOTTEN_PWD_URL>"
      },
      {
      "name": "oua.drss.password.reset.url",
      "value": "<CHANGE_PWD_URL>"
      }, ]'
```

- If you are using OAM Forgotten password functionality, set `oua.drss.password.reset.forgoturl` to *https://login.example.com/otpfp/pages/fp.jsp*.
- If you are using OIG Forgotten password functionality, set `oua.drss.password.reset.forgoturl` to *https://prov.example.com/identity/faces/forgotpassword* and set `oua.drss.password.reset.url` to *https://prov.example.com/identity*.

The following is an example for OIG command:

```
curl --location --insecure --request PUT
      'https://login.example.com/oaa/runtime/config/property/v1' \--header
'Content-Type: application/json'
      \--header 'Authorization: Basic
      b2FhLW9hYTphcGlrZXl0b2Jlc2V0ZHVyaW5naW5zdGFsbGF0aW9u' \-d '[
      {
      "name": "oua.drss.password.reset.forgoturl",
      "value": "https://prov.example.com/identity/faces/forgotpassword"
      },
```

```
{
"name": "oua.drss.password.reset.url",
"value": "https://prov.example.com/identity"
},]'
```

## Updating Test User to Add Authenticator

If you have created an OAA test user in Creating a Test User and you want to use this user to validate OUA then you must register the TOTP Authenticator against this user by adding a factor through the OAA user preference screen.

1. Log into the OAA user preference console using the following URL

   *https://login.example.com/oaa/rui*
   Use the user you want to set up an authentication factor for. For example, oaauser.
2. Click **Manage** from the **My Authenticators** box.
3. Click **Add Authentication Factor**.
4. Select an Authentication factor. For example, Mobile Authenticator.
5. Enter a name.
6. Click **Save**.

> **Note:**
>
> You must setup the Oracle Mobile Authenticator using the displayed QR code or make a note of the Key below to setup the Oracle Mobile Authenticator later before saving the details as you will not be able to obtain these details again.

## Installing the OUA Client Application

For more information about how to install the OUA Client application, see Installing the Oracle Universal Authenticator Client Application.

22

# Configuring Disaster Recovery

In the past, disaster recovery plans often involved syncing a file system between the main and backup systems. The most effective method for syncing is through disk replication. However, if this option is not available, the next best solution is to use the rsync utility to copy the data from one location to another. To assist in the setting up of Disaster Recovery environment, Oracle has provided sample scripts to set up Disaster Recovery for each of the Oracle products described in this chapter. The sample scripts can be used either directly or as an extra reference to know how certain steps are performed. These automation scripts are described in Automating the Disaster Recovery Setup.
This chapter includes the following topics:

- Generic Disaster Recovery Processes
  Some of the common disaster recovery processes include using rsync to copy data between sites, using Oracle Data Guard to create a standby database, backing up and restoring the cluster artifacts, synchronizing the persistent volumes, and so on.
- Configuring Disaster Recovery for Oracle Unified Directory
  The disaster recovery for Oracle Unified Directory (OUD) is an active/passive solution. Currently, you cannot use the `dsreplication` command to set up a remote replica.
- Configuring Disaster Recovery for Oracle Access Manager
  The disaster recovery for Oracle Access Manager (OAM) is an active/passive solution.
- Configuring Disaster Recovery for Oracle Identity Governance
  The disaster recovery for Oracle Identity Governance (OIG) is an active/passive solution.
- Configuring Disaster Recovery for Oracle Identity Role Intelligence
  The disaster recovery for Oracle Identity Role Intelligence (OIRI) is an active/passive solution.
- Configuring Disaster Recovery for Oracle Advanced Authentication
  The disaster recovery for Oracle Advanced Authentication (OAA) is an active/passive solution.

## Generic Disaster Recovery Processes

Some of the common disaster recovery processes include using rsync to copy data between sites, using Oracle Data Guard to create a standby database, backing up and restoring the cluster artifacts, synchronizing the persistent volumes, and so on.

Here is the list of the generic disaster recovery processes:

- Creating a Container with rsync
- Creating a Data Guard Database
- Backing Up and Restoring the Kubernetes Objects
- Creating a Kubernetes CronJob to Synchronize the Persistent Volumes
- Creating the Persistent Volume Claims
- Creating a DR ConfigMap
- Creating a Backup/Restore Job

# Creating a Container with rsync

You can replicate persistent volume data in a number of ways. These can be divided into two categories:

- Hardware Replication - Uses disk based replication of cloud based file system replication.
- Software Replication - Uses a software tool to manually replicate a file system. One of the most widely available and efficient tools to perform software replication is rsync.

This section describes how to create a light weight container with the `rsync` command which can be run inside the Kubernetes cluster. This container can then be invoked using a Kubernetes CronJob.

You can run rsync as a CronJob on the underlying worker nodes, but this creates an external dependency. Another way of achieving this is by creating a pod that runs inside of the cluster which performs the rsync operations.

This job ensures that the application disaster recovery processes run as part of the Kubernetes cluster. CronJobs require container images that include the `rsync` command. The two main Linux container images, Alpine and Busybox, do not contain the `rsync` command by default. However, they can be extended to include the command. Complete the following steps to create an image based on the Alpine container with `rsync` included.

The following steps provide instructions on how to create a container image based on Alpine that also includes the `rsync` utility. These steps require a runtime environment with Podman or Docker and a logged-in DockerHub account to access the repository images.

1. Download and start the Alpine container using the following command:

   ```
   podman run -dit alpine sh
   ```

   This command will start an Alpine container.

2. Obtain the container ID using the command:

   ```
   podman ps
   ```

   The output of this command will be similar to:

   ```
   CONTAINER ID  IMAGE                            COMMAND     CREATED
   STATUS
   7cb3c9de95ea  docker.io/library/alpine:latest  sh          20 minutes ago
   Up 20 minutes ago
   ```

   The container id is `7cb3c9de95ea`.

3. Connect to the container using the container id. For example:

   ```
   podman attach 7cb3c9de95ea
   ```

4. Install `rsync` into the container using the commands:

```
apk update


apk add rsync
```

The output will be similar to:

```
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/main/x86_64/
APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/community/x86_64/
APKINDEX.tar.gz
v3.17.0-136-gc113cb02a1 [https://dl-cdn.alpinelinux.org/alpine/v3.17/main]
v3.17.0-137-ge9e378b388 [https://dl-cdn.alpinelinux.org/alpine/v3.17/
community]
OK: 17803 distinct packages available

/ # apk add rsync
(1/5) Installing libacl (2.3.1-r1)
(2/5) Installing lz4-libs (1.9.4-r1)
(3/5) Installing popt (1.19-r0)
(4/5) Installing zstd-libs (1.5.2-r9)
(5/5) Installing rsync (3.2.7-r0)
Executing busybox-1.35.0-r29.trigger
OK: 8 MiB in 20 packages
```

Exit the container by pressing `Ctrl+P+Q` on the keyboard.

5. Commit the changes to the image using the command:

```
podman commit -m "Added rsync" 7cb3c9de95ea alpine-rsync -f docker
```

The output will be similar to:

```
Getting image source signatures
Copying blob ded7a220bb05 skipped: already exists
Copying blob 6459a5a97a89 done
Copying config 6cf4ad33ac done
Writing manifest to image destination
Storing signatures
6cf4ad33aca33fe62cb0fee9354b0b8f548ab9983427c25c24f77fcb6542e17c
```

6. Check that alpine-rsync is now in your local image store by using the command:

```
podman images
```

The output will be similar to:

```
REPOSITORY
TAG                         IMAGE ID      CREATED         SIZE
localhost/alpine-rsync
```

```
latest                          6cf4ad33aca3  14 seconds ago  11.1 MB
docker.io/library/alpine
latest                          49176f190c7e  2 weeks ago     7.34 MB
```

7. Tag the image with your container registry name using the command:

```
podman tag 6cf4ad33aca3  iad.ocir.io/mytenancy/idm/alpine-rsync:latest
```

8. Verify that alpine-rsync is now tagged in your local image store by using the command:

```
podman images
```

The output will be similar to:

```
iad.ocir.io/mytenancy/idm/alpine-rsync
latest                          6cf4ad33aca3  28 minutes ago  11.1 MB
localhost/alpine-rsync
latest                          6cf4ad33aca3  28 minutes ago  11.1 MB
docker.io/library/alpine
latest                          49176f190c7e  2 weeks ago     7.34 MB
```

9. Push the image to your container registry using the commands:

```
podman login iad.ocir.io/mytenant -u paasmaa/oracleidentitycloudservice/
myuser -p "mypassword"


podman push iad.ocir.io/mytenancy/idm/alpine-rsync:latest
```

The image is now available for use from your container registry.

## Creating a Data Guard Database

You can use rsync to copy the data from one site to the other outside of Kubernetes. However, a more efficient way of copying data is by creating a Kubernetes CronJob. In the event of a disaster recovery, after you have deployed your application on the standby site, you should delete the existing database and create a standby database using Oracle Data Guard.

If you are using a non-Oracle Cloud based deployment (or you want to configure it manually), see Learn About Configuring a Standby Database for Disaster Recovery for instructions.

> **Note:**
>
> After creating the Data Guard database, ensure that it has the same initialization parameter values as that of the primary database.

If you are using Oracle Cloud Infrastructure, use the following steps:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Ensure that your region is set to the same region where your primary database resides.
3. Navigate to **Oracle Database** and click **Oracle Base Database (VM, BM)**.

4. Select the name of the database system that houses your database. Your database will be listed in the Databases section of the page.
5. Select the name of your database.
6. Select the **Data Guard Associations** submenu.
7. Click **Enable Data Guard**.
8. Enter the following information in the wizard:
   - **Display Name** : Enter a name for the Data Guard database.
   - **Region**: Select the region where the standby site is located.
   - **Availability Domain**: Choose the availability domain to locate your database.
   - **Configure Shape**: Edit the shape, if needed.
   - **License Type**: Choose the license type you want to use.
   - **Virtual Cloud Network Name**: Enter the name of your VCN on the standby site.
   - **Client Subnet**: Enter the name of the database subnet on the standby site. For example, `db-subnet`.
   - Enter a host name prefix for the database nodes. For example, `regdb` (where `reg` is the name of the region).
   - Enter the type of Data Guard you want to deploy. For example, `Active Data Guard` (recommended).
9. Click **Next**.
10. On the Database Information screen, enter the following information:
    - **Database Image**: Ensure that the database image is the same as that of the primary database.
    - **Database Password**: Enter a sys password for the standby database.
11. Click **Enable Data Guard**.
12. Verify that the Data Guard database has the same initialization parameters as that of the primary database.

> **Note:**
>
> Due to a bug in the Data Guard creation process, you may find that some parameters, such as processes, have overriding values for a specific instance. If you encounter this issue, use the following command to remove the overriding values:
>
> ```
> alter system reset processes sid='instance_name';
> ```

> **Note:**
>
> After running the network analyzer, if you see successful communication paths between your regions but a Data Guard association error occurs similar to the following:
>
> ```
> A cross-region Data Guard association cannot be created between databases of
> DB systems that
>       belong to VCNs that are not peers.
> ```
>
> Then, raise a Service Request with Oracle Support requesting that `CrossRegionDataguardNetworkValidation` be disabled.

## Backing Up and Restoring the Kubernetes Objects

Every Kubernetes cluster is distinct and requires individual maintenance. The artifacts within the cluster determine the application's operation. The artifacts include namespaces, persistent volumes, config maps, and more. These artifacts must be created on the standby cluster before you start the application.

There are two ways of creating the artifacts on the standby system:

- Perform a separate installation using the same configuration information as that of the primary site but using a throwaway database. After you complete the installation, discard the database and domain configuration and replace it with the primary site's database and domain configuration.
- Back up the Kubernetes objects in the primary site and restore them to the standby site, pointing the persistent volumes to the standby NFS servers. This approach is not recommended for Oracle Advanced Authentication.

Oracle provides a tool to simplify the process of backing up and restoring the Kubernetes objects. For information about the tool, see Using the Oracle K8s Disaster Protection Scripts.

To use these scripts:

1. Download the scripts to a suitable location using the command:

   ```
   git clone -q https://github.com/oracle-samples/maa


   chmod 700 maa/kubernetes-maa/*.sh
   ```

2. Create a directory to hold the backup. For example:

   ```
   mkdir -p /workdir/k8backup
   ```

3. Create a backup of a namespace using the command:

   ```
   kubernetes-maa/maak8-get-all-artifacts.sh <Backup Directory> <Namespace>
   ```

   For example:

   ```
   kubernetes-maa/maak8-get-all-artifacts.sh /workdir/k8backup  oamns
   ```

This process will take a few minutes to complete. It creates a dated directory in the backup location that contains a `.gz` file, which is the backup archive.

4. Transfer the `.gz` file to the standby site using your preferred file transfer mechanism.
5. Restore the backup site into the standby cluster using the command:

```
kubernetes-maa/maak8-push-all-artifacts.sh <BACKUP_FILE> <WORKING
DIRECTORY>
```

For example:

```
kubernetes-maa/maak8-push-all-artifacts.sh /workdir/k8backup/
23-06-05-11-03-15/ /workdir/k8restore
```

> **Note:**
>
> - If you do not precreate the persistent volumes on the standby site before restoring, the applications will not start. Alternatively, you can create the PVs after restoring, but this will cause pods to remain pending until the volumes are created. It is crucial to ensure that the persistent volumes on the standby site point to the NFS server in that site.
> - It is possible to request the backup to include the persistent volumes in its backup process. However, this will also backup the mounts to the primary NFS server. Ensure that you do not allow the application to start on the standby site by using the primary NFS server because it could result in the corruption of your primary deployment.
> - If the application is running on the primary when the backup is being taken, the application will be started on the standby when the restore is being performed. To minimize errors, you should open the Data Guard database as a snapshot standby. However, some errors may still occur due to the running jobs from SOA and OIG, but these can be ignored because they are caused by the inactive status of the site.
>
> At some point, you should switch your Data Guard database to the standby site and verify that the standby services are functional.

## Creating a Kubernetes CronJob to Synchronize the Persistent Volumes

For each solution described in this section, the configuration information stored on the persistent volumes must be synchronized between the primary and standby sites, with the ability to reverse the synchronization in case of a site switchover.

As mentioned earlier, to achieve the maximum efficiency, the best approach is to employ file system replication. See Backing Up and Restoring the Kubernetes Objects Alternatively, a `cron` job be created inside the Kubernetes cluster to carry out this task. The process is the same for each product, with only the underlying volumes being different. It is advisable to create a separate `cron` job for each product rather than having one that handles all products.

The following steps describe how to achieve this. The example uses Oracle Access Manager (OAM) for simplicity, but it applies to any product.

## Creating a Namespace for the Backup Jobs

Create a separate namespace outside of the product namespace to hold the backup jobs, either per product or for all the backup jobs.

Create the namespace using the command:

```
create namespace <namespace>
```

For example:

```
create namespace drns
```

## Creating the Persistent Volumes

Each site requires a persistent volume that points to the NFS filer in the remote location.

Create the following persistent volumes on Site 1 using the provided `yaml` files:

**site1_primary_pv.yaml**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: primary-oam-pv
  labels:
    type: primary-oam-pv
spec:
  storageClassName: manual
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /export/IAMPVS/oampv
    server: <site1_nfs_server>
```

**site1_standby_pv.yaml**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: standby-oam-pv
  labels:
    type: standby-oam-pv
spec:
  storageClassName: manual
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteMany
```

```
  nfs:
    path: /export/IAMPVS/oampv
    server: <site2_nfs_server>
```

Create the following persistent volumes on Site 2 using the provided `yaml` files:

**site2_primary_pv.yaml**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: primary-oam-pv
  labels:
    type: primary-oam-pv
spec:
  storageClassName: manual
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /export/IAMPVS/oampv
    server: <site2_nfs_server>
```

**site2_standby_pv.yaml**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: standby-oam-pv
  labels:
    type: standby-oam-pv
spec:
  storageClassName: manual
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /export/IAMPVS/oampv
    server: <site1_nfs_server>
```

Create the persistent volumes using the commands:

**Site1**

```
kubectl create -f site1_primary_pv.yaml


kubectl create -f site1_standby_pv.yaml
```

**Site2**

```
kubectl create -f site2_primary_pv.yaml


kubectl create -f site2_standby_pv.yaml
```

## Creating the Persistent Volume Claims

After creating the persistent volumes, you can create the persistent volume claims in the DR namespace.

Create the following files (note that the files will be identical on each site):

**primary_pvc.yaml**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: primary-oampv-pvc
  namespace: <DRNS>
  labels:
    type: primary-oampv-pvc
spec:
  storageClassName: manual
  accessModes:
  - ReadWriteMany
  resources:
     requests:
       storage: 30Gi
  selector:
    matchLabels:
      type: primary-oam-pv
```

**standby_pvc.yaml**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: standby-oampv-pvc
  namespace: <DRNS>
  labels:
    type: standby-oampv-pvc
spec:
  storageClassName: manual
  accessModes:
  - ReadWriteMany
  resources:
     requests:
       storage: 30Gi
  selector:
    matchLabels:
      type: standby-oam-pv
```

Create the persistent volume claims on both the sites using the commands:

```
kubectl create -f primary_pvc.yaml


kubectl create -f standby_pvc.yaml
```

# Creating a DR ConfigMap

Each site has a ConfigMap object containing the configuration information about the site, such as the site's role, domain name, primary database scan address and service name, and so on.

- The role the site is performing. This role determines which way the PV is replicated. The replication should always be from primary to standby.
- The name of the domain (determines the location of the *DOMAIN_HOME* on the persistent volume).
- The Primary Database Scan address and service name. These are used to change the replicated configuration information to point to the database in the standby site.
- The standby database scan address and service name. These are used to change the replicated configuration information to point to the database in the standby site.

Description of the ConfigMap object:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dr-cm
  namespace: <DRNS>
data:
  ENV_TYPE: <ENV_TYPE>
  DR_TYPE: <DR_TYPE>
  OAM_DOMAIN_NAME: <OAM_DOMAIN_NAME>
  OAM_LOCAL_SCAN: <OAM_LOCAL_SCAN>
  OAM_REMOTE_SCAN: <OAM_REMOTE_SCAN>
  OAM_LOCAL_SERVICE: <OAM_LOCAL_SERVICE>
  OAM_REMOTE_SERVICE: <OAM_REMOTE_SERVICE>
  OIG_DOMAIN_NAME: <OIG_DOMAIN_NAME>
  OIG_LOCAL_SCAN: <OIG_LOCAL_SCAN>
  OIG_REMOTE_SCAN: <OIG_REMOTE_SCAN>
  OIG_LOCAL_SERVICE: <OIG_LOCAL_SERVICE>
  OIG_REMOTE_SERVICE: <OIG_REMOTE_SERVICE>
  OIRI_LOCAL_SCAN: <OIRI_LOCAL_SCAN>
  OIRI_REMOTE_SCAN: <OIRI_REMOTE_SCAN>
  OIRI_LOCAL_SERVICE: <OIRI_LOCAL_SERVICE>
  OIRI_REMOTE_SERVICE: <OIRI_REMOTE_SERVICE>
  OIRI_REMOTE_K8: <OIRI_REMOTE_K8>
  OIRI_REMOTE_K8CONFIG: <OIRI_REMOTE_K8CONFIG>
  OIRI_REMOTE_K8CA: <OIRI_REMOTE_K8CA>
  OIRI_LOCAL_K8: <OIRI_LOCAL_K8>
  OIRI_LOCAL_K8CONFIG: <OIRI_LOCAL_K8CONFIG>
  OIRI_LOCAL_K8CA: <OIRI_LOCAL_K8CA>
  OAA_LOCAL_SCAN: <OAA_LOCAL_SCAN>
  OAA_REMOTE_SCAN: <OAA_REMOTE_SCAN>
```

```
  OAA_LOCAL_SERVICE: <OAA_LOCAL_SERVICE>
  OAA_REMOTE_SERVICE: <OAA_REMOTE_SERVICE>
```

Here is a sample of the ConfigMap object:

**site1_dr_cm.yaml**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dr-cm
  namespace: <DRNS>
data:
  ENV_TYPE: OTHER
  DR_TYPE: PRIMARY
  OAM_LOCAL_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
  OAM_REMOTE_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
  OAM_LOCAL_SERVICE: oamsvc.dbsubnet.oke.oraclevcn.com
  OAM_REMOTE_SERVICE: oamsvc.dbsubnet.oke.oraclevcn.com
  OIG_DOMAIN_NAME: governancedomain
  OIG_LOCAL_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
  OIG_REMOTE_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
  OIG_LOCAL_SERVICE: oigsvc.dbsubnet.oke.oraclevcn.com
  OIG_REMOTE_SERVICE: oigsvc.dbsubnet.oke.oraclevcn.com
  OIRI_LOCAL_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
  OIRI_REMOTE_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
  OIRI_LOCAL_SERVICE: oirisvc.dbsubnet.oke.oraclevcn.com
  OIRI_REMOTE_SERVICE: oirisvc.dbsubnet.oke.oraclevcn.com
  OIRI_REMOTE_K8: 10.1.0.10:6443
  OIRI_REMOTE_K8CONFIG: standby_k8config
  OIRI_REMOTE_K8CA: standby_ca.crt
  OIRI_LOCAL_K8: 10.0.0.5:6443
  OIRI_LOCAL_K8CONFIG: primary_k8config
  OIRI_LOCAL_K8CA:primary_ca.crt
  OAA_LOCAL_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
  OAA_REMOTE_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
  OAA_LOCAL_SERVICE: oaasvc.dbsubnet.oke.oraclevcn.com
  OAA_REMOTE_SERVICE: oaasvc.dbsubnet.oke.oraclevcn.com
```

**site2_dr_cm.yaml**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dr-cm
  namespace: <DRNS>
data:
  ENV_TYPE: OTHER
  DR_TYPE: STANDBY
  OAM_LOCAL_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
  OAM_REMOTE_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
  OAM_LOCAL_SERVICE: oamsvc.dbsubnet.oke.oraclevcn.com
  OAM_REMOTE_SERVICE: oamsvc.dbsubnet.oke.oraclevcn.com
  OIG_DOMAIN_NAME: governancedomain
  OIG_LOCAL_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
```

```
OIG_REMOTE_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
OIG_LOCAL_SERVICE: oigsvc.dbsubnet.oke.oraclevcn.com
OIG_REMOTE_SERVICE: oigsvc.dbsubnet.oke.oraclevcn.com
OIRI_LOCAL_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
OIRI_REMOTE_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
OIRI_LOCAL_SERVICE: oirisvc.dbsubnet.oke.oraclevcn.com
OIRI_REMOTE_SERVICE: oirisvc.dbsubnet.oke.oraclevcn.com
OIRI_REMOTE_K8: 10.0.0.10:6443
OIRI_REMOTE_K8CONFIG: standby_k8config
OIRI_REMOTE_K8CA: standby_ca.crt
OIRI_LOCAL_K8: 10.1.0.5:6443
OIRI_LOCAL_K8CONFIG: primary_k8config
OIRI_LOCAL_K8CA:primary_ca.crt
OAA_LOCAL_SCAN: site2-scan.dbsubnet.oke.oraclevcn.com
OAA_REMOTE_SCAN: site1-scan.dbsubnet.oke.oraclevcn.com
OAA_LOCAL_SERVICE: oaasvc.dbsubnet.oke.oraclevcn.com
OAA_REMOTE_SERVICE: oaasvc.dbsubnet.oke.oraclevcn.com
```

Create the ConfigMap for Site 1 using the following command:

```
kubectl create -f site1_dr_cm.yaml
```

Create the ConfigMap for Site 2 using the following command:

```
kubectl create -f site2_dr_cm.yaml
```

# Creating a Backup/Restore Job

The steps below describe how to create a job which will run periodically to back up the primary persistent volume using the `rsync` command and restore it to the standby system.

- Creating a Backup/Restore Script
- Creating a CronJob to Run the Backup/Restore Script Periodically
- Suspending/Resuming the CronJob
- Creating an Initialization Job

## Creating a Backup/Restore Script

You should create a script to backup the persistent volume and copy the backup to the standby site. The deployment automation scripts contain a sample script to perform this task. This script is called `<product>_dr.sh` and it is located in the `SCRIPT_DIR/templates/<product>` directory.

For example, Oracle Access Manager, the script that needs to be used for disaster recovery is located at `SCRIPT_DIR/templates/oam/oam_dr.sh`. You need to copy this script to the persistent volume using one of the following methods:

- By copying the script to a running container in the deployment.
- By creating a simple Alpine container to access the persistent volume.
- By temporarily mounting the NFS to a host where the scripts are being developed.

For the purpose of this document, the location where the script needs to be copied is referred to as '`dr_scripts`'.

The script will run on both sites. If the site is in primary mode, it will create a backup of the persistent volume and send it to Site 2. If the site is in standby mode, it will restore the backup received from Site 1 onto Site 2. It is recommended that the scripts make a local copy of the persistent volume before performing any backup or restore operations.

It is highly recommended that the scripts make a local copy of the persistent volume beforehand by using NFS utilities such as snapshots or by using the `rsync` command, as per the following steps:

1. To keep backups small, exclude files which are not needed.

```
EXCLUDE_LIST="--exclude=\".snapshot\" --exclude=\"backups\" --
exclude=\"domains/*/servers/*/tmp\" --exclude=\"logs/*\" --
exclude=\"dr_scripts\" --exclude=\"network\" --exclude \"domains/*/
servers/*/data/nodemanager/*.lck\" --exclude \"domains/*/servers/*/data/
nodemanager/*.pid\" --exclude \"domains/*/servers/*/data/nodemager/
*.state\" --exclude \"domains/ConnectorDefaultDirectory\"--
exclude=\"backup_running\""
```

2. Set up the location inside the container where the primary PV and standby PV are mounted.

```
PRIMARY_BASE=/u01/primary_oampv
DR_BASE=/u01/dr_oampv


rsync -avz $EXCLUDE_LIST $PRIMARY_BASE/ $PRIMARY_BASE/backups/backup_date
```

3. Transfer the backup to the standby site using the `rsync` command:

```
rsync -avz $EXCLUDE_LIST $PRIMARY_BASE/backups/backup_date/ $DR_BASE/
backups/backup_date
```

4. On the standby site, restore the backup using the `rsync` command:

```
rsync -avz $EXCLUDE_LIST $PRIMARY_BASE/backups/backup_date / $PRIMARY_BASE
```

5. Search through the domain and change any occurrences of the primary database scan address and service to that used in the standby site.
   a. Determine all the files that need to be changed using the command:

```
cd $PRIMARY_BASE/domains/$OAM_DOMAIN_NAME/config


grep -rl "$REMOTE_SCAN" . | grep -v backup
```

   b. Update the relevant files with the correct information from the DR ConfigMap object.

## Creating a CronJob to Run the Backup/Restore Script Periodically

To automate the backup process, you need to create a job that runs the backup script periodically. Another job will be created based on this job to initialize the backup process. To

avoid any unexpected runs of the CronJob, it should be suspended immediately after its creation.

The frequency of the job's schedule should be based on how often configuration changes are made. If changes are less frequent, the job can be run less frequently. However, the more often the job runs, the more resources it will require, which can potentially impact system performance. Additionally, the interval between jobs should provide enough time for the job to complete before the next iteration starts. A suitable frequency might be once per day with additional manual synchronizations as needed.

The initial run of the job will take longer compared to the subsequent runs. Oracle recommends you to suspend the CronJob as soon as you create it, and then manually run a job to initialize the DR site. After you initialize the DR site, you can restart the CronJob.

To create a CronJob in the `drns` namespace, create a file called `oamdr-cron.yaml` with the following contents:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: <product>rsyncdr
  namespace: <DRNS>
spec:
  schedule: "*/720 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          imagePullSecrets:
          - name: regcred
          containers:
          - name: alpine-rsync
            image: iad.ocir.io/mytenancy/idm/alpine-rsync:latest
            imagePullPolicy: IfNotPresent
            envFrom:
              - configMapRef:
                  name: dr-cm
            volumeMounts:
              - mountPath: "/u01/primary_oampv"
                name: oampv
              - mountPath: "/u01/dr_oampv"
                name: oampv-dr
            command:
            - /bin/sh
            - -c
            - /u01/primary_oampv/dr_scripts/oam_dr.sh
          volumes:
          - name: oampv
            persistentVolumeClaim:
              claimName: primary-oampv-pvc
          - name: oampv-dr
            persistentVolumeClaim:
              claimName: standby-oampv-pvc
          restartPolicy: OnFailure
```

For example:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: oamrsyncdr
  namespace: drns
spec:
  schedule: "*/720 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          imagePullSecrets:
          - name: regcred
          containers:
          - name: alpine-rsync
            image: iad.ocir.io/paasmaa/idm/alpine-rsync:latest
            imagePullPolicy: IfNotPresent
            envFrom:
              - configMapRef:
                  name: dr-cm
            volumeMounts:
              - mountPath: "/u01/primary_oampv"
                name: oampv
              - mountPath: "/u01/dr_oampv"
                name: oampv-dr
            command:
            - /bin/sh
            - -c
            - /u01/primary_oampv/dr_scripts/oam_dr.sh
          volumes:
          - name: oampv
            persistentVolumeClaim:
              claimName: primary-oampv-pvc
          - name: oampv-dr
            persistentVolumeClaim:
              claimName: standby-oampv-pvc
          restartPolicy: OnFailure
```

This job runs once a day. For a full description of the CronJob scheduling in Kuberenetes, see FreeBSD File Formats Manual.

This job uses the custom Alpine container image that was created earlier. See Creating a Container with rsync. To allow for the initialization of the DR site, which takes longer than regular refreshes, you need to immediately suspend the job. See Suspending/Resuming the CronJob. It is essential to create a separate job to perform the initialization process as this ensures that only one sync job runs at a time because it takes time to perform the initial copy.

## Suspending/Resuming the CronJob

To suspend a backup CronJob, run the following command:

```
kubectl patch cronjobs <product>rsyncdr -p '{"spec" : {"suspend" : true }}' -
n <NAMESPACE>
```

For example:

```
kubectl patch cronjobs oamrsyncdr -p '{"spec" : {"suspend" : true }}' -n drns
```

To restart the backup CronJob, run the following command:

```
kubectl patch cronjobs <product>rsyncdr -p '{"spec" : {"suspend" : false }}' -
n <NAMESPACE>
```

For example:

```
kubectl patch cronjobs oamrsyncdr -p '{"spec" : {"suspend" : false }}' -n drns
```

## Creating an Initialization Job

Use the CronJob defined earlier (Creating a CronJob to Run the Backup/Restore Script Periodically), create a one off job to perform the initial data transfer by using the command:

```
kubectl create job --from=cronjob.batch/<product>rsyncdr <product>-initialise-
dr -n <DRNS>
```

For example:

```
kubectl create job --from=cronjob.batch/oamrsyncdr oam-initialise-dr -n drns
```

Monitor the job until it completes successfully. After the job completes, you can initiate a restore on the second site by using the same commands.

# Configuring Disaster Recovery for Oracle Unified Directory

The disaster recovery for Oracle Unified Directory (OUD) is an active/passive solution. Currently, you cannot use the `dsreplication` command to set up a remote replica.

The process of setting up the DR site is summarized below:

- Perform a minimal installation on Site 2 using the standard deployment procedures.
- Stop the OUD processes on the DR site.
- Delete the OUD persistent volume data on the DR site.
- Do one of the following:
  - Enable disk replication for the persistent volume.
  - Enable manual replication using a CronJob which mounts both the local and the remote OUD persistent volume. This job will use the `rsync` command to synchronize the data. Suspend the job after the replication is complete.
- Perform an initial data sync.
- Verify that the DR site comes up successfully.
- Shut down the DR site.
- Restart the CronJob to copy the data periodically.

The following sections describe the procedure in detail:



## Prerequisites

Ensure that you meet the following prerequisites:

- You are able to mount the remote file systems in the local pods. This task may require firewall or security list permissions.
- You are able to perform disk replication between the sites. This task may require firewall or security list permissions.
- You have access to Alpine or Busybox with `rsync` installed, if you plan to use manual replication. See Creating a Backup/Restore Job.

## Creating an Empty OUD Deployment on Site 2

Create an OUD deployment on Site 2 by following the instructions in Installing and Configuring Oracle Unified Directory with the following exceptions:

- Use the same schema extension file as Site 1.
- There is no need for a seeding file.
- Use a minimal server overrides file. For example:

```
image:
  repository: <OUD_REPOSITORY>
  tag: <OUD_VER>
  pullPolicy: IfNotPresent

imagePullSecrets:
  - name: regcred

oudConfig:
  baseDN: <LDAP_SEARCHBASE>
  rootUserDN: <LDAP_ADMIN_USER>
  rootUserPassword: <LDAP_ADMIN_PWD>
  sleepBeforeConfig: 300

persistence:
  type: networkstorage
  networkstorage:
    nfs:
      server: <PVSERVER>
      path: <OUD_SHARE>
```

```
configVolume:
  enabled: true
  type: networkstorage
  networkstorage:
    nfs:
      server: <PVSERVER>
      path: <OUD_CONFIG_SHARE>
  mountPath: /u01/oracle/config-input

replicaCount: <OUD_REPLICAS>

ingress:
  enabled: false
  type: nginx
  tlsEnabled: false

elk:
  enabled: false
  imagePullSecrets:
    - name: dockercred

cronJob:
  kubectlImage:
    repository: bitnami/kubectl
    tag: <KUBERNETES_VER>
    pullPolicy: IfNotPresent

    imagePullSecrets:
    - name: dockercred

baseOUD:
  envVars:
    - name: schemaConfigFile_1
      value: /u01/oracle/config-input/99-user.ldif
    - name: restartAfterSchemaConfig
      value: "true"

replOUD:
  envVars:
    - name: dsconfig_1
      value: set-global-configuration-prop --set lookthrough-limit:75000
```

# Enabling a Manual Replication

Complete the following procedures if you want to create a CronJob to manually replicate data between the sites.

- Creating a Persistent Volume for the Remote Persistent Volume
- Creating a Persistent Volume Claim for the Remote Persistent Volume
- Creating a Backup and Restore Script
- Copying the Backup Script to the Persistent Volume
- Creating a ConfigMap to Determine Site Characteristics

- Creating a CronJob

## Creating a Persistent Volume for the Remote Persistent Volume

To create a persistent volume for the remote OUD persistent volume:

1. Create the `ouddr-pv.yaml` file with the following contents:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: oudpv-dr
  labels:
    type: edgdr-pv
spec:
  storageClassName: manual
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: <OUD_SHARE>
    server: <PV_SERVER>
```

For example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: oudpv-dr
  labels:
    type: edgdr-pv
spec:
  storageClassName: manual
  capacity:
    storage: 30Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /exports/IAMPVS/oudpv
    server: <REMOTE_PVSERVER>
```

2. Create the persistent volume using the command:

```
kubectl create -f ouddr-pv.yaml
```

3. Verify that the persistent volume has been created using the command:

```
kubectl get pv
```

The output appears similar to the following:

```
NAME                     CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                           STORAGECLASS   REASON   AGE
```

```
edg-oud-ds-rs-pv          30Gi       RWX            Delete
Bound    oudns/edg-oud-ds-rs-pvc           manual                  21d
edg-oud-ds-rs-pv-config   10Gi       RWX            Retain
Bound    oudns/edg-oud-ds-rs-pvc-config    manual                  21d
oudpv-dr                  30Gi       RWX            Retain
Bound    oudns/ouddr-pvc                   manual                  54m
```

> **Note:**
>
> You will see a persistent volume for the current OUD deployment and the new one created for the remote OUD deployment.

You should create this persistent volume on both Site 1 and Site 2. Ensure that on Site 1 the `REMOTE_PV_SERVER` points to the `PVSERVER` on Site 2 and on Site 2 the `REMOTE_PV_SERVER` points to the `PVSERVER` on Site 1. Creating the PV on both the sites ensures that the configuration is immediately available if the roles of Site 1 and Site 2 need to be reversed.

## Creating a Persistent Volume Claim for the Remote Persistent Volume

To create a persistent volume claim for the remote OUD persistent volume:

1. Create the `ouddr-pvc.yaml` with the following contents:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ouddr-pvc
  namespace: <OUDNS>
  labels:
    type: ouddr-pvc
spec:
  storageClassName: manual
  accessModes:
  - ReadWriteMany
  resources:
     requests:
       storage: 30Gi
  selector:
    matchLabels:
      type: edgdr-pv
```

For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ouddr-pvc
  namespace: oudns
  labels:
    type: ouddr-pvc
spec:
  storageClassName: manual
  accessModes:
```

```
- ReadWriteMany
resources:
   requests:
     storage: 30Gi
selector:
  matchLabels:
    type: edgdr-pv
```

2. Create the persistent volume claim using the command:

```
kubectl create -f ouddr-pvc.yaml
```

3. Verify that the persistent volume has been created using the command:

```
kubectl get pvc -n <OUDNS>
```

The output will appear similar to the following:

```
NAME                          STATUS   VOLUME                    CAPACITY
ACCESS MODES   STORAGECLASS   AGE
edg-oud-ds-rs-pvc             Bound    edg-oud-ds-rs-pv          30Gi
RWX            manual         21d
edg-oud-ds-rs-pvc-config      Bound    edg-oud-ds-rs-pv-config   10Gi
RWX            manual         21d
ouddr-pvc                     Bound    oudpv-dr                  30Gi
RWX            manual         63m
```

> **Note:**
>
> You will see a persistent volume claim for the current OUD deployment and the new one created for the remote OUD deployment.

You can create the same PVC claim on both Site 1 and Site 2. Creating on both sites ensures that the configuration is available immediately if the roles of Site 1 and Site 2 need to be reversed.

## Creating a Backup and Restore Script

You must create a script that will backup the OUD instances and copy it to your standby site. This script should then be able to restore the backup on the standby site. If you need an example of such a script, you can find it in the deployment automation scripts under the name `oud_dr.sh`. The script is located in the `SCRIPT_DIR/templates/oud` directory. See Creating a Backup/Restore Script.

This script must have the following characteristics:

- It should be able to determine whether it is running on the primary or the standby site.
- It should not run if a previous backup/restore job is in progress.
- When running on the primary site:
  - Create a localized backup of the persistent volume (to a temporary location) by using snapshots (if available) or rsync.
  - Copy the created backup to a temporary location on the standby site using rsync.

- When running on the standby site, restore the OUD instances in the received backup to the persistent volume.

> **Note:**
>
> Restore only the instances, not the backup scripts.

## Copying the Backup Script to the Persistent Volume

The easiest way to ensure that the container image is able to run the backup script is to copy the script to the persistent volume. You should perform this step on both the sites using the following commands:

```
kubectl exec -n <OUDNS> -ti <OUD_POD_PREFIX>-oud-ds-rs-0 -- mkdir /u01/oracle/
user_projects/dr_scripts
```

```
kubectl cp <WORKDIR>/oud_dr.sh <OUDNS>/<OUD_POD_PREFIX>-oud-ds-rs-0:/u01/
oracle/user_projects/dr_scripts
```

```
kubectl exec -n <OUDNS> -ti <OUD_POD_PREFIX>-oud-ds-rs-0 -- chmod 750 /u01/
oracle/user_projects/dr_scripts/oud_dr.sh
```

For example:

```
kubectl exec -n oudns -ti edg-oud-ds-rs-0 -- mkdir /u01/oracle/user_projects/
dr_scripts
```

```
kubectl cp /workdir/OUD/oud_dr.sh oudns edg-oud-ds-rs-0:/u01/oracle/
user_projects/dr_scripts
```

```
kubectl exec -n oudns -ti edg-oud-ds-rs-0 -- chmod 750 /u01/oracle/
user_projects/dr_scripts/oud_dr.sh
```

## Creating a ConfigMap to Determine Site Characteristics

See Creating a DR ConfigMap.

## Creating a CronJob

Create a CronJob to synchronize the local and remote persistent volumes. For instructions, see Creating a CronJob to Run the Backup/Restore Script Periodically. After creating, suspend the CronJob immediately. For instructions, see Suspending/Resuming the CronJob.

## Shutting Down the OUD Installation on Site 2

Shut down the OUD installation on Site 2 if everything is functioning as expected. Use the following command to shut down:

```
helm upgrade -n oudns --set replicaCount=0 edg <WORKDIR>/samples/kubernetes/
helm/oud-ds-rs --reuse-values
```

## Deleting the Persistent Volume Data on Site 2

To ensure that the data from Site 1 is fully copied across to Site 2, it is important to remove the data that was created as part of the installation. Assuming that the persistent volume is mounted locally, you can remove the data using the following command:

```
rm -rf /nfs_volumes/oudpv/*oud-ds-rs*
```

## Creating an Initialization Job

Use the CronJob you defined earlier (see Creating a CronJob for creating a one off job to perform the initial data transfer using the command:

```
kubectl create job --from=cronjob.batch/rsyncdr initialise-dr -n <DRNS>
```

For example:

```
kubectl create job --from=cronjob.batch/rsyncdr initialise-dr -n drns
```

> **Note:**
>
> Perform this step on Site 1.

## Verifying OUD on the DR Site

After the successful initial data load, you can start the OUD servers in Site 2 using the command:

```
helm upgrade -n <OUDNS> --set replicaCount=<REPLICA_COUNT> <OUD_POD_PREFIX>
<WORKDIR>/samples/kubernetes/helm/oud-ds-rs --reuse-values
```

For example:

```
helm upgrade -n oudns --set replicaCount=1 edg  /workdir/OUD/samples/
kubernetes/helm/oud-ds-rs --reuse-values
```

After starting the servers, verify the availability of data from the primary site by checking the replication status and running the `ldapsearch` command-line tool.

If everything is functioning as expected, shut down OUD by using the following command:

```
helm upgrade -n oudns --set replicaCount=0 edg <WORKDIR>/samples/kubernetes/
helm/oud-ds-rs --reuse-values
```

## Starting the Automatic Syncing Process

Start the CronJob on Site 1 to periodically synchronize the changes from Site 1 Persistent Volume to Site 2. See Suspending/Resuming the CronJob.

# Configuring Disaster Recovery for Oracle Access Manager

The disaster recovery for Oracle Access Manager (OAM) is an active/passive solution.

The following sections describe the procedure in detail:

- Prerequisites
- Creating the Disaster Recovery Site From the Primary Site
- Creating the Disaster Recovery Site on the Standby Site

## Prerequisites

Before enabling disaster recovery for Oracle Access Manager, ensure the following:

- The primary and standby sites for Data Guard communicate with each other.
- The primary and standby sites for file system replication communicate with each other.
- The Kubernetes clusters in both the primary and standby sites are able to resolve the names of the NFS servers in all sites.
- Each Kubernetes cluster is independent.
- A running Kubernetes cluster is present in both the sites.

## Creating the Disaster Recovery Site From the Primary Site

On the primary site:

1. Create a backup of the persistent volume holding OAM data (`oampv`). See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes. The file exclusion list for OAM is defined as follows:

   ```
   EXCLUDE_LIST="--exclude=\".snapshot\" --exclude=\"backups\" --
   exclude=\"domains/*/servers/*/tmp\" --exclude=\"logs/*\" --
   exclude=\"dr_scripts\" --exclude=\"network\" --exclude \"domains/*/
   servers/*/data/nodemanager/*.lck\" --exclude \"domains/*/servers/*/data/
   nodemanager/*.pid\" --exclude \"domains/*/servers/*/data/nodemager/
   *.state\" --exclude \"domains/ConnectorDefaultDirectory\"--
   exclude=\"backup_running\""
   ```

2. Back up the Kubernetes objects for restoration on the standby sites. See Backing Up and Restoring the Kubernetes Objects.
3. Set up Oracle Data Guard between the primary and standby sites. See Creating a Data Guard Database.

4. Ensure that the database services used on the primary site are replicated on the standby. These services should be active when the database is running in the roles of `PRIMARY` or `SNAPSHOT STANDBY`. See Creating Database Services.

## Creating the Disaster Recovery Site on the Standby Site

There are two options for creating the disaster recovery site on the standby site:

**By Creating a New Disposable Environment on Site 2**

1. Create a disposable deployment in Site 2 using the same values as the primary site. If you have used the EDG automation scripts, you just need to change the values of the worker nodes and the PV server in the response file.
2. Shut down the Kubernetes pods that are running on Site 2.
3. Delete the contents of the persistent volume.
4. Restore the backup of the persistent volume from the backup taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
5. Amend the database connection strings in the restored backup to point to the database on Site 2.
6. Copy the WebGate objects from the restored backup to the Oracle HTTP Servers on Site 2.
7. Validate that the disaster recovery site is working.
8. Delete the throwaway database used to perform the initial installation.

**By Performing a Kubernetes Restore on Site 2**

1. Create the persistent volumes for the OAM domain such that they point at the NFS server in the standby site. See #unique_520.
2. Open the standby database as a snapshot standby by using the following Data Guard broker commands:

```
dgmgrl sys/Password


show configuration


convert database standby_db_name to snapshot standby
```

3. Restore a backup of the persistent volume taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
4. Restore the backup of the Kubernetes objects from the backup taken on Site 1. See Backing Up and Restoring the Kubernetes Objects.
5. Amend the database connection strings in the restored backup to point to the database on Site 2.
6. Copy the WebGate objects from the restored backup to the Oracle HTTP Servers on Site 2.
7. Validate that the disaster recovery site is working.

8. Reinstate the standby database by issuing the following Data Guard broker commands:

```
dgmgrl sys/Password
```

```
show configuration
```

```
convert database standby_db_name to physical standby
```

Switch over the database and validate that the deployment is fully working.

# Configuring Disaster Recovery for Oracle Identity Governance

The disaster recovery for Oracle Identity Governance (OIG) is an active/passive solution.

The following sections describe the procedure in detail:

- Prerequisites
- Creating the Disaster Recovery Site From the Primary Site
- Creating the Disaster Recovery Site on the Standby Site
- Disabling the Job Scheduler for Configuring Oracle Identity Manager

## Prerequisites

Before enabling disaster recovery for Oracle Identity Governance, ensure the following:

- The primary and standby sites for file system replication communicate with each other.
- The database system in the primary site communicate with each other for database replication.
- The Kubernetes clusters in both the primary and standby sites are able to resolve the names of the NFS servers in all sites.
- Each Kubernetes cluster is independent.
- A running Kubernetes cluster is present in both the sites.

## Creating the Disaster Recovery Site From the Primary Site

On the primary site:

1. Create a backup of the persistent volume holding OIG data (`oigpv`). See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes. The file exclusion list for OIG is defined as follows:

```
EXCLUDE_LIST="--exclude=\".snapshot\" --exclude=\"backups\" --
exclude=\"domains/*/servers/*/tmp\" --exclude=\"logs/*\" --
exclude=\"dr_scripts\" --exclude=\"network\" --exclude \"domains/*/
servers/*/data/nodemanager/*.lck\" --exclude \"domains/*/servers/*/data/
nodemanager/*.pid\" --exclude \"domains/*/servers/*/data/nodemager/
*.state\" --exclude \"domains/ConnectorDefaultDirectory\" --
exclude=\"backup_running\""
```

2. Back up the Kubernetes objects for restoration on the standby sites. See Backing Up and Restoring the Kubernetes Objects.
3. Set up Oracle Data Guard between the primary and standby sites.

## Creating the Disaster Recovery Site on the Standby Site

There are two options for creating the disaster recovery site on the standby site:

**By Creating a New Disposable Environment on Site 2**

1. Create a disposable deployment in Site 2 using the same values as the primary site. If you have used the EDG automation scripts, you just need to change the values of the worker nodes and the PV server in the response file.
2. Shut down the Kubernetes pods that are running on Site 2.
3. Delete the contents of the persistent volume.
4. Restore the backup of the persistent volume from the backup taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
5. Amend the database connection strings in the restored backup to point to the database on Site 2.
6. Validate that the disaster recovery site is working.
7. Delete the throwaway database used to perform the initial installation.

**By Performing a Kubernetes Restore on Site 2**

1. Create the persistent volumes for the OIG domain such that they point at the NFS server in the standby site. See #unique_614.
2. Open the standby database as a snapshot standby by using the following Data Guard broker commands:

```
dgmgrl sys/Password


show configuration


convert database standby_db_name to snapshot standby
```

3. Restore a backup of the persistent volume taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
4. Restore the backup of the Kubernetes objects from the backup taken on Site 1. See Backing Up and Restoring the Kubernetes Objects.
5. Amend the database connection strings in the restored backup to point to the database in Site 2.
6. Validate that the disaster recovery site is working.

7. Reinstate the standby database by issuing the following Data Guard broker commands:

```
dgmgrl sys/Password
```

```
show configuration
```

```
convert database standby_db_name to physical standby
```

Switch over the database and validate that the deployment is fully working.

## Disabling the Job Scheduler for Configuring Oracle Identity Manager

You must the disable the Oracle Identity Governance job scheduler on Site 2 for configuring Oracle Identity Governance.

The OIG job scheduler uses the database intensively. To keep inter-site traffic to a minimum, the job scheduler should be disabled on the site where the database is not primary. This step is not required if you plan to shut down the OIG Managed Servers on Site 2. You can disable the job scheduler by adding the following parameter to the server startup arguments:

```
-Dscheduler.disabled=true
```

After switching over the database, this parameter should be removed from these servers and added into the server definitions of the now standby site.

# Configuring Disaster Recovery for Oracle Identity Role Intelligence

The disaster recovery for Oracle Identity Role Intelligence (OIRI) is an active/passive solution.

The following sections describe the procedure in detail:

- Prerequisites
- Creating the Disaster Recovery Site

## Prerequisites

Before enabling disaster recovery for Oracle Identity Role Intelligence, ensure the following:

- The primary and standby sites for file system replication communicate with each other.
- The database system in the primary site communicate with each other for database replication.
- The Kubernetes clusters in both the primary and standby sites are able to resolve the names of the NFS servers in all sites.
- Each Kubernetes cluster is independent.
- A running Kubernetes cluster is present in both the sites.

# Creating the Disaster Recovery Site

OIRI is tightly integrated into the Kubernetes framework. It interacts directly with the cluster to be able to run data-ingestion tasks. When you deploy OIRI, it contains information such as the cluster name in which it is running. When you run OIRI from a different site, then, in addition to updating the database connection information, you also need to update the Kubernetes cluster information. You can perform this task in two ways.

- When you are starting OIRI on a different site, update the Kubernetes configuration.
- Store the configuration for each site on the persistent volume. Then, when you restore the persistent volume on the standby site, replace the Kubernetes configuration with the appropriate values using these files.
  It is possible to create these files only after you have created the OIRI Kubernetes objects on the standby site.

  The recommended approach is to create copies of the Kubernetes configuration objects in the persistent volume and label them according to the site. For example, `primary_ca.crt` and `primary_k8config`. Create the standby site (see Creating the Disaster Recovery Site on the Standby Site), and then create a new `ca.crt` and `kubeconfig` file based on the standby cluster. Copy these files to the persistent volume of the primary site calling the files `standby_ca.crt` and `standby_k8config`. This naming convention will ensure that when you replicate the persistent volume to the standby site, you have copies of the `ca.crt` and `kubeconfig` file for both the primary and standby clusters. After running the PV replication task, restore the `ca.crt` and `kubeconfig` files on the persistent volume with those relevant to the site you want to run. For example:

  – When you use the primary site, the `ca.crt` and `config` files will use the contents of the `primary_ca.crt` and `primary_k8config` files, respectively.

  – When you are use the standby site, the `ca.crt` and `config` files will use the contents of the `standby_ca.crt` and `standby_k8config` files, respectively.
- Creating the Disaster Recovery Site From the Primary Site
- Creating the Disaster Recovery Site on the Standby Site

## Creating the Disaster Recovery Site From the Primary Site

On the primary site:

1. Create copies of the existing `ca.crt` and `config` files located on the `workpv` (mounted as `/app/k8s` in the OIRI-CLI container). Call these files `primary_ca.crt` and `primary_config`, respectively.
2. Create a backup of the persistent volumes holding the OIRI data (`oiripv`, `dingpv`, and `workpv`). See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes. The file exclusion list for OIRI is defined as follows:

   ```
   EXCLUDE_LIST="--exclude=\".snapshot\" --exclude=\"backups\" --
   exclude=\"dr_scripts\" --exclude=\"backup_running\""
   ```

3. Back up the Kubernetes objects for restoration on the standby sites. See Backing Up and Restoring the Kubernetes Objects.
4. Set up Oracle Data Guard between the primary and standby sites. See Creating a Data Guard Database.

5. Ensure that the database services used on the primary site are replicated on the standby. These services should be active when the database is running in the roles of `PRIMARY` and `SNAPSHOT` standby. See Creating Database Services.

## Creating the Disaster Recovery Site on the Standby Site

There are two options for creating the disaster recovery site on the standby site:

**By Creating a New Disposable Environment on Site 2**

1. Create a disposable deployment in Site 2 using the same values as the primary site. If you have used the EDG automation scripts, you just need to change the values of the worker nodes and the PV server in the response file.
2. Copy the `ca.crt` and `config` files generated on the standby site to the `workpv` on the primary site. Call these files `standby_ca.crt` and `standby_config`, respectively.
3. Shut down the Kubernetes pods that are running on Site 2.
4. Delete the contents of the persistent volume.
5. Restore the backup of the persistent volume from the backup taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
6. Amend the database connection strings in the restored backup to point to the database on Site 2. Update the following files:

```
/app/oiri/data/conf/application.yaml
/app/data/conf/custom-attributes.yaml
/app/data/conf/data-ingestion-config.yaml
/app/data/conf/dbconfig.yaml
/app/data/conf/env.properties
```

7. Amend the Kubernetes cluster URL in the restored backup to point to the Kubernetes cluster in the standby site. To find the Kubernetes cluster URL, use the following command:

```
grep server: $KUBECONFIG | sed 's/server://;s/ //g'
```

   Update the following files:

```
/app/data/conf/data-ingestion-config.yaml
/app/data/conf/env.properties
```

8. Switch the Kubernetes configuration files to the copies appropriate to the standby site. For example:

```
cp /app/k8s/standby_ca.crt /app/k8s/ca.crt


cp /app/k8s/standby_ca.crt /app/ca.crt


cp /app/k8s/standby_config.crt /app/k8s/config
```

9. Validate that the disaster recovery site is working.
10. Delete the throwaway database used to create the initial installation on Site 2.

**By Performing a Kubernetes Restore on Site 2**

1. Create the persistent volumes for the OIRI persistent volumes such that they point at the NFS server in the standby site. See #unique_614.
2. Open the standby database as a snapshot standby by using the following Data Guard broker commands:

```
dgmgrl sys/Password


show configuration


convert database standby_db_name to snapshot standby
```

3. Restore a backup of the persistent volume taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
4. Restore the backup of the Kubernetes objects from the backup taken on Site 1. See Backing Up and Restoring the Kubernetes Objects.
5. Start the OIRI-CLI container on the standby system. See Starting the Administration CLI.
6. Create a `kubeconfig` file and a certificate on the standby site. See Generating the `ca.crt` Certificate and Creating a Kubernetes Configuration File for OIRI.
   Store the resulting files `ca.crt` and `oiri_config` in the OIRI-CLI locations in both the primary and standby sites.

```
/app/k8s/standby_ca.crt
/app/k8s/standby_ca.crt
/app/k8s/standby_config.crt
```

7. Amend the database connection strings in the restored backup to point to the database in Site 2.
   Update the following files:

```
/app/oiri/data/conf/application.yaml
/app/data/conf/custom-attributes.yaml
/app/data/conf/data-ingestion-config.yaml
/app/data/conf/dbconfig.yaml
/app/data/conf/env.properties
```

8. Amend the Kubernetes cluster URL in the restored backup to point to the Kubernetes cluster in the standby site. To find the Kubernetes cluster URL, use the following command:

```
grep server: $KUBECONFIG | sed 's/server://;s/ //g'
```

   Update the following files:

```
/app/data/conf/data-ingestion-config.yaml
/app/data/conf/env.properties
```

9. Switch the Kubernetes configuration files to the copies appropriate to the standby site. For example:

```
cp /app/k8s/standby_ca.crt /app/k8s/ca.crt


cp /app/k8s/standby_ca.crt /app/ca.crt


cp /app/k8s/standby_config.crt /app/k8s/config
```

10. Validate that the disaster recovery site is working.
11. Reinstate the standby database by using the following Data Guard broker commands:

```
dgmgrl sys/Password


show configuration


convert database standby_db_name to physical standby
```

    Switch over the database and validate that the deployment is fully working.
12. Recopy the persistent volumes from the primary to the standby site on a periodic basis. For example, once a week, or whenever you make configuration changes to the OIRI deployment.

# Configuring Disaster Recovery for Oracle Advanced Authentication

The disaster recovery for Oracle Advanced Authentication (OAA) is an active/passive solution.

The following sections describe the procedure in detail:

- Prerequisites
- Creating the Disaster Recovery Site

## Prerequisites

Before enabling disaster recovery for Oracle Advanced Authentication, ensure the following:

- The primary and standby sites for file system replication communicate with each other.
- The database system in the primary site communicate with each other for database replication.
- The Kubernetes clusters in both the primary and standby sites are able to resolve the names of the NFS servers in all sites.
- Each Kubernetes cluster is independent.
- A running Kubernetes cluster is present in both the sites.
- The OAuth provider must be available in both the sites.

> **Note:**
>
> If you are using OAM as the OAuth provider and the failover scenario is an all or nothing, that is, you will fail/switch over OAM and OAA together, then, for the purposes of creating the DR site, you need to enable OAM in the standby site during configuration by temporarily converting the OAM standby database to a snapshot standby and starting OAM using that database. After configuring OAA, shut down OAM and revert the database to a physical standby.
>
> Each OAM must use the same SSL certificate.

# Creating the Disaster Recovery Site

OAA is tightly integrated into the Kubernetes framework. When you deploy OAA, it contains information such as the cluster name in which it is running. Therefore, you must exclude this information when you replicate the persistent volumes.

- Creating the Disaster Recovery Site From the Primary Site
- Creating the Disaster Recovery Site on the Standby Site

## Creating the Disaster Recovery Site From the Primary Site

On the primary site:

1. Create a backup of the persistent volumes holding the OAA data (`oaaconfigpv`, `oaacreadpv`, `oaavaultpv`, and `oaalogspv`). See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes. The file exclusion list for OAA is defined as follows:

   ```
   EXCLUDE_LIST="--exclude=\".snapshot\" --exclude=\"backups\" --
   exclude=\"dr_scripts\" --exclude=\"backup_running\" --
   exclude=\"k8sconfig\" --exclude=\"ca.crt\""
   ```

2. Set up Oracle Data Guard between the primary and standby sites.
3. Ensure that the database services used on the primary site are replicated on the standby. These services should be active when the database is running in the roles of `PRIMARY` and `SNAPSHOT` standby. See Creating Database Services.

## Creating the Disaster Recovery Site on the Standby Site

On the standby site:

1. Open the standby database as a snapshot standby by using the following Data Guard broker commands:

   ```
   dgmgrl sys/Password


   show configuration


   convert database standby_db_name to snapshot standby
   ```

2. Restore a backup of the persistent volume taken on Site 1. See Creating a Kubernetes CronJob to Synchronize the Persistent Volumes.
3. Create a namespace for OAA (this should be the same as the namespace of the primary site). See Creating Kubernetes Namespaces.
4. Create a container registry secret. See Creating a Container Registry Secret.
5. Create the OAA Management Container. See Starting the Management Container.
6. Grant the OAA Management Container access to the Kubernetes cluster. See Granting the Management Container Access to the Kubernetes Cluster.
7. Modify the `installOAA.properties` file copied from the primary site. The file is located in the `/u01/oracle/scripts/settings` directory in the OAA Management Container. See Creating the OAA Property File.

   Make the following changes:

   - Change `database.host` to the scan address of the standby database.
   - Set `database.createschema=false`.
8. Recreate the OAA deployment by running the `OAA.sh` script. See Deploying Oracle Advanced Authentication.
9. Verify that the OAA pods start successfully.
10. Reinstate the standby database by using the following Data Guard broker commands:

    ```
    dgmgrl sys/Password


    show configuration


    convert database standby_db_name to physical standby
    ```

    Switch over the database and validate that the deployment is fully working.
11. Recopy the persistent volumes from the primary to the standby site on a periodic basis. For example, once a week, or whenever you make configuration changes to the OAA deployment.

# Part IV

# Common Configuration and Management Procedures for an Enterprise Deployment

There are a few typical configuration and management procedures that Oracle recommends for a regular enterprise deployment. Patching an enterprise deployment, configuring whole server migration and automatic server migration, performing scale out and scale in operations, troubleshooting issues are a few of the recommended procedures for an enterprise deployment.

This part of the guide contains the following chapters:

- Common Configuration and Management Tasks for an Enterprise Deployment
  The configuration tasks include a few that are common to all enterprise deployments, such as verifying sizing information, performing backups and recoveries, and so on. Patching an enterprise deployment and cross wiring components are the other common tasks.
- Using Whole Server Migration and Service Migration in an Enterprise Deployment
  The Oracle WebLogic Server migration framework supports Whole Server Migration and Service Migration. The following sections explain how these features can be used in an Oracle Fusion Middleware enterprise topology.
- Centralized Monitoring Using Grafana and Prometheus
  When using Prometheus and Grafana, you can configure non-Kubernetes products to send their data to the console. This data is in addition to data from the Kubernetes components.
- Centralized Log File Monitoring Using Elasticsearch and Kibana
- Scaling Procedures for an Enterprise Deployment
  The scaling procedures for an enterprise deployment include scale out and scale in. During a scale-out operation, you add Managed Servers to new nodes. You can remove these Managed Servers by performing a scale in operation.
- Configuring Single Sign-On for an Enterprise Deployment
  You need to configure the Oracle HTTP Server WebGate in order to enable single sign-on with Oracle Access Manager.
- Sanity Checks
- Troubleshooting
  You can troubleshoot the common issues that may arise with the Identity and Access Management enterprise deployment. The solutions provided for the common problems help you resolve them quickly.

23

# Common Configuration and Management Tasks for an Enterprise Deployment

The configuration tasks include a few that are common to all enterprise deployments, such as verifying sizing information, performing backups and recoveries, and so on. Patching an enterprise deployment and cross wiring components are the other common tasks.

This chapter includes the following topics:

- Configuration and Management Tasks for All Enterprise Deployments
  Complete these common configuration tasks that apply to any Oracle Fusion Middleware enterprise deployment. These tasks include checking the sizing requirements for the deployment, using the JDBC persistence store for web services, and taking backups of the deployment.
- Starting and Stopping Components
  You can start and stop the various components (such as WebLogic components, an entire domain, a WebLogic Cluster, and so on) after you have deployed them.
- Patching an Enterprise Deployment
  You should update the Oracle Identity Governance Kubernetes cluster with bundle patches whenever the patches are released.
- Performing Backup and Restore Operations
  Ensure that you keep backups outside of the Kubernetes cluster so that they are available even if the cluster has issues.
- Performing Maintenance on a Kubernetes Worker Node
  If you have to shut down a Kubernetes worker node for maintenance/patching or just to refresh it, you should first gracefully move any running Kubernetes services on that worker node.
- Adjusting the Server Pods Liveness Probe
- Considerations for Cross-Component Wiring
  Cross-Component Wiring (CCW) enables the FMW components to publish and bind to some of the services available in a WLS domain, by using specific APIs.

## Configuration and Management Tasks for All Enterprise Deployments

Complete these common configuration tasks that apply to any Oracle Fusion Middleware enterprise deployment. These tasks include checking the sizing requirements for the deployment, using the JDBC persistence store for web services, and taking backups of the deployment.

- Core DNS Allocation
- Verifying Appropriate Sizing and Configuration for the WLSSchemaDataSource
- About JDBC Persistent Stores for Web Services
- Enabling Autoscaling

- Performing Backups and Recoveries for an Enterprise Deployment

# Core DNS Allocation

> **Note:**
>
> This step is applicable to any Kubernetes system using `coredns`.

Place at least two `coredns` pods on the control plane (if possible) and another two on the worker nodes. The `coredns` footprint is low.

```
 VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND

146268  41684  29088 S   0.3  0.1  25:44.04 coredns
```

According to the CoreDNS documentation, you can estimate the amount of memory required for a CoreDNS instance (using default settings) with the following formula:

```
MB required (default settings) = (Pods + Services) / 1000 + 54
```

Hence, first, label the nodes in both control and worker plane.

```
$ kubectl label nodes K8ControlPlane1 area=dnsarea
$ kubectl label nodes K8ControlPlane2 area=dnsarea
$ kubectl label nodes K8ControlPlane3 area=dnsarea
$ kubectl label nodes k8worker1 area=dnsarea
$ kubectl label nodes k8worker2 area=dnsarea
$ kubectl label nodes k8worker3 area=dnsarea
```

And then, update the `coredns` deployment to use topology spread constraints.

> **Note:**
>
> Topology spread constraints is beta starting in Kubernetes v1.18.

First, enable the feature gate in `kube-api` server and in `kube-scheduler`. Then, modify the `coredns` deployment for an appropriate spread of pods across the worker and the control plane nodes.

The `coredns` topology spread configuration details are:

The following is a sample of the `coredns` deployment yaml file:

**Coredns deployment YAML**

```
$ kubectl get deployment coredns -n kube-system -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
```

```
    deployment.kubernetes.io/revision: "7"
  creationTimestamp: "2021-01-15T13:15:05Z"
  generation: 8
  labels:
    area: dnsarea
    k8s-app: kube-dns
  managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:labels:
          .: {}
          f:k8s-app: {}
      f:spec:
        f:progressDeadlineSeconds: {}
        f:revisionHistoryLimit: {}
        f:selector:
          f:matchLabels:
            .: {}
            f:k8s-app: {}
        f:strategy:
          f:rollingUpdate:
            .: {}
            f:maxSurge: {}
            f:maxUnavailable: {}
          f:type: {}
        f:template:
          f:metadata:
            f:labels:
              .: {}
              f:k8s-app: {}
          f:spec:
            f:containers:
              k:{"name":"coredns"}:
                .: {}
                f:args: {}
                f:image: {}
                f:imagePullPolicy: {}
                f:livenessProbe:
                  .: {}
                  f:failureThreshold: {}
                  f:httpGet:
                    .: {}
                    f:path: {}
                    f:port: {}
                    f:scheme: {}
                  f:initialDelaySeconds: {}
                  f:periodSeconds: {}
                  f:successThreshold: {}
                  f:timeoutSeconds: {}
                f:name: {}
                f:ports:
                  .: {}
                  k:{"containerPort":53,"protocol":"TCP"}:
                    .: {}
```

```
          f:containerPort: {}
          f:name: {}
          f:protocol: {}
        k:{"containerPort":53,"protocol":"UDP"}:
          .: {}
          f:containerPort: {}
          f:name: {}
          f:protocol: {}
        k:{"containerPort":9153,"protocol":"TCP"}:
          .: {}
          f:containerPort: {}
          f:name: {}
          f:protocol: {}
      f:readinessProbe:
        .: {}
        f:failureThreshold: {}
        f:httpGet:
          .: {}
          f:path: {}
          f:port: {}
          f:scheme: {}
        f:periodSeconds: {}
        f:successThreshold: {}
        f:timeoutSeconds: {}
      f:resources:
        .: {}
        f:limits:
          .: {}
          f:memory: {}
        f:requests:
          .: {}
          f:cpu: {}
          f:memory: {}
      f:securityContext:
        .: {}
        f:allowPrivilegeEscalation: {}
        f:capabilities:
          .: {}
          f:add: {}
          f:drop: {}
        f:readOnlyRootFilesystem: {}
      f:terminationMessagePath: {}
      f:terminationMessagePolicy: {}
      f:volumeMounts:
        .: {}
        k:{"mountPath":"/etc/coredns"}:
          .: {}
          f:mountPath: {}
          f:name: {}
          f:readOnly: {}
  f:dnsPolicy: {}
  f:nodeSelector:
    .: {}
    f:kubernetes.io/os: {}
  f:priorityClassName: {}
  f:restartPolicy: {}
```

```
          f:schedulerName: {}
          f:securityContext: {}
          f:serviceAccount: {}
          f:serviceAccountName: {}
          f:terminationGracePeriodSeconds: {}
          f:tolerations: {}
          f:volumes:
            .: {}
            k:{"name":"config-volume"}:
              .: {}
              f:configMap:
                .: {}
                f:defaultMode: {}
                f:items: {}
                f:name: {}
              f:name: {}
    manager: kubeadm
    operation: Update
    time: "2021-01-15T13:15:05Z"
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:labels:
          f:area: {}
      f:spec:
        f:replicas: {}
        f:template:
          f:metadata:
            f:annotations:
              .: {}
              f:kubectl.kubernetes.io/restartedAt: {}
            f:labels:
              f:foo: {}
          f:spec:
            f:topologySpreadConstraints:
              .: {}
              k:{"topologyKey":"area","whenUnsatisfiable":"DoNotSchedule"}:
                .: {}
                f:labelSelector:
                  .: {}
                  f:matchLabels:
                    .: {}
                    f:foo: {}
                f:maxSkew: {}
                f:topologyKey: {}
                f:whenUnsatisfiable: {}
    manager: kubectl
    operation: Update
    time: "2021-01-28T16:00:21Z"
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .: {}
```

```
            f:deployment.kubernetes.io/revision: {}
        f:status:
          f:availableReplicas: {}
          f:conditions:
            .: {}
            k:{"type":"Available"}:
              .: {}
              f:lastTransitionTime: {}
              f:lastUpdateTime: {}
              f:message: {}
              f:reason: {}
              f:status: {}
              f:type: {}
            k:{"type":"Progressing"}:
              .: {}
              f:lastTransitionTime: {}
              f:lastUpdateTime: {}
              f:message: {}
              f:reason: {}
              f:status: {}
              f:type: {}
          f:observedGeneration: {}
          f:readyReplicas: {}
          f:replicas: {}
          f:updatedReplicas: {}
      manager: kube-controller-manager
      operation: Update
      time: "2021-01-28T16:00:39Z"
  name: coredns
  namespace: kube-system
  resourceVersion: "2520507"
  selfLink: /apis/apps/v1/namespaces/kube-system/deployments/coredns
  uid: 79d24e61-98f4-434f-b682-132625b04c49
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kube-dns
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      annotations:
        kubectl.kubernetes.io/restartedAt: "2021-01-28T15:29:48Z"
      creationTimestamp: null
      labels:
        foo: bar
        k8s-app: kube-dns
    spec:
      containers:
      - args:
```

```
      - -conf
      - /etc/coredns/Corefile
      image: k8s.gcr.io/coredns:1.6.7
      imagePullPolicy: IfNotPresent
      livenessProbe:
        failureThreshold: 5
        httpGet:
          path: /health
          port: 8080
          scheme: HTTP
        initialDelaySeconds: 60
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 5
      name: coredns
      ports:
      - containerPort: 53
        name: dns
        protocol: UDP
      - containerPort: 53
        name: dns-tcp
        protocol: TCP
      - containerPort: 9153
        name: metrics
        protocol: TCP
      readinessProbe:
        failureThreshold: 3
        httpGet:
          path: /ready
          port: 8181
          scheme: HTTP
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 1
      resources:
        limits:
          memory: 170Mi
        requests:
          cpu: 100m
          memory: 70Mi
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          add:
          - NET_BIND_SERVICE
          drop:
          - all
        readOnlyRootFilesystem: true
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
      - mountPath: /etc/coredns
        name: config-volume
        readOnly: true
    dnsPolicy: Default
    nodeSelector:
```

```
        kubernetes.io/os: linux
      priorityClassName: system-cluster-critical
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: coredns
      serviceAccountName: coredns
      terminationGracePeriodSeconds: 30
      tolerations:
      - key: CriticalAddonsOnly
        operator: Exists
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
      topologySpreadConstraints:
      - labelSelector:
          matchLabels:
            foo: bar
        maxSkew: 1
        topologyKey: area
        whenUnsatisfiable: DoNotSchedule
      volumes:
      - configMap:
          defaultMode: 420
          items:
          - key: Corefile
            path: Corefile
          name: coredns
        name: config-volume
status:
  availableReplicas: 4
  conditions:
  - lastTransitionTime: "2021-01-21T19:08:12Z"
    lastUpdateTime: "2021-01-21T19:08:12Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2021-01-28T15:29:48Z"
    lastUpdateTime: "2021-01-28T16:00:39Z"
    message: ReplicaSet "coredns-84b49c57fd" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 8
  readyReplicas: 4
  replicas: 4
  updatedReplicas: 4
```

The labels and spread topology changes are:

```
labels:
      foo: bar
      k8s-app: kube-dns


topologySpreadConstraints:
- labelSelector:
    matchLabels:
      foo: bar
  maxSkew: 1
  topologyKey: area
  whenUnsatisfiable: DoNotSchedule
```

This guarantees an even distribution across the master and worker nodes. Therefore, if the control plane is restored, the worker pods will continue without issues and the other way around.

Sample of the resulting `coredns` distribution:

```
kubectl get pods -A -o wide | grep coredns
kube-system   coredns-84b49c57fd-4fz4g                          1/1
Running     0          166m    10.244.1.20   K8ControlPlane2
<none>           <none>
kube-system   coredns-84b49c57fd-5mrkw                          1/1
Running     0          165m    10.244.4.76   K8Worker2
<none>           <none>
kube-system   coredns-84b49c57fd-5zm88                          1/1
Running     0          165m    10.244.2.17   K8ControlPlane3
<none>           <none>
kube-system   coredns-84b49c57fd-nqlwb                          1/1
Running     0          166m    10.244.4.75   K8Worker2
<none>           <none>
```

# Verifying Appropriate Sizing and Configuration for the WLSSchemaDataSource

`WLSSchemaDataSource` is the common data source that is reserved for use by the FMW components for JMS JDBC Stores, JTA JDBC stores, and Leasing services. `WLSSchemaDataSource` is used to avoid contention in critical WLS infrastructure services and to guard against dead-locks.

To reduce the `WLSSchemaDataSource` connection usage, you can change the JMS JDBC and TLOG JDBC stores connection caching policy from *Default* to *Minimal* by using the respective connection caching policy settings. When there is a need to reduce connections in the back-end database system, Oracle recommends that you set the caching policy to *Minimal* . Avoid using the caching policy *None* because it causes a potential degradation in performance. For a detailed tuning advice about connections that are used by JDBC stores, see Configuring a JDBC Store Connection Caching Policy in *Administering the WebLogic Persistent Store*.

The default `WLSSchemaDataSource` connection pool size is 75 (size is double in the case of a GridLink Data Source). You can tune this size to a higher value depending on the size of the different FMW clusters and the candidates that are configured for migration. For example,

consider a typical SOA EDG deployment with the default number of worker threads per store. If more than 25 JDBC Stores or TLOG-in-DB instances or both can fail over to the same Weblogic server, and the Connection Caching Policy is not changed from *Default* to *Minimal*, possible connection contention issues could arise. In these cases, increasing the default `WLSSchemaDataSource` pool size (maximum capacity) becomes necessary (each JMS store uses a minimum of two connections, and leasing and JTA are also added to compete for the pool).

# About JDBC Persistent Stores for Web Services

By default, web services use the WebLogic Server default persistent store for persistence. This store provides high-performance storage solution for web services.

The default web service persistence store is used by the following advanced features:

- Reliable Messaging
- Make Connection
- SecureConversation
- Message buffering

You also have the option to use a JDBC persistence store in your WebLogic Server web service, instead of the default store. For information about web service persistence, see Managing Web Service Persistence.

# Enabling Autoscaling

Kubernetes allows pods to be auto-scaled. If there are two pods running and your resource exceeds a threshold, such as memory, a new pod is automatically started.

For example, when you achieve 75% of available memory, a new pod is started on a different worker node.

For acheiving autoscaling, you should first define the resource requirements for each pod type you want to autoscale. See the relevant product's Installing and Configuring chapter for details.

- Deploying the Kubernetes Metric Server
- Deploying the Kubernetes HorizontalPodAutoscaler Resource

## Deploying the Kubernetes Metric Server

Before deploying autoscaling, you need to deploy the Kubernetes metric server.

1. To deploy this server, run the following command:

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/
   releases/latest/download/components.yaml
   ```

2. Confirm that the metric server is running by using the following command:

   ```
   kubectl get pods -n kube-system
   ```

## Deploying the Kubernetes HorizontalPodAutoscaler Resource

The following example shows how to automatically scale a WebLogic cluster based on memory and CPU utilization.

Assuming that you have an OAM cluster running in the `oamns` namespace, use the following command to create a HorizontalPodAutoscaler (HPA) resource targeted at the cluster resource (`accessdomain-oam-cluster`) that will autoscale Oracle WebLogic Server instances from a minimum of two cluster member to a five cluster member. The scale up or down will occur when the average CPU utilization is consistently over 70%.

1. Create a file called `oam_scaler.yaml` with the following contents:

```
#
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: accessdomain-oam-cluster-hpa
  namespace: oamns
spec:
  scaleTargetRef:
    apiVersion: weblogic.oracle/v1
    kind: Cluster
    name: accessdomain-oam-cluster
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 60
    scaleUp:
      stabilizationWindowSeconds: 60
  minReplicas: 2
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
```

If you want to do the same, but use memory consumption as the trigger, then the above file would have the following metrics:

```
metrics:
- type: Resource
  resource:
    name: memory
    target:
      type: Utilization
      averageUtilization: 70
```

> **Note:**
>
> `maxReplicas` should not exceed the `configuredManagedServerCount` value provided during domain.

2. Update `scaleTargetRef` for the following products:

   For example, for Oracle Unified Directory (OUD):

   ```
   scaleTargetRef:
     apiVersion: apps/v1
     kind: StatefulSet
     name: <DEPLOYMENT>
   ```

   The following table provides the value of the *name* variable for other products:

| Product | Deployment |
|---|---|
| Oracle Identity Role Intelligence (OIRI) | `oiri` |
| OIRI DING | `spark-history-server` |
| Oracle Advanced Authentication (OAA) | `OAADEPLOYMENT_NAME`: It can be one of the following values:<br>• `edg-email`<br>• `edg-fido`<br>• `edg-oaa`<br>• `edg-oaa-admin-ui`<br>• `edg-oaa-kba`<br>• `edg-oaa-policy`<br>• `edg-push`<br>• `edg-risk`<br>• `edg-risk-cc`<br>• `edg-sms`<br>• `edg-spui`<br>• `edg-totp`<br>• `edg-yotp` |

3. Execute the file using the following command:

   ```
   kubectl apply -f oam_scaler.yaml
   ```

4. Verify the status of the autoscaler and its behavior by inspecting the HPA resource.

   ```
   $ kubectl get hpa -n oamns
   ```

# Performing Backups and Recoveries for an Enterprise Deployment

Oracle recommends you to follow the below mentioned guidelines to ensure that you back up the necessary directories and configuration data for an Oracle Identity and Access Management enterprise deployment.

> **Note:**
>
> Some of the static and runtime artifacts listed in this section are hosted from Network Attached Storage (NAS). If possible, backup and recover these volumes from the NAS filer directly rather than from the application servers.

For general information about backing up and recovering Oracle Fusion Middleware products, see the following sections in *Administering Oracle Fusion Middleware*:

- Backing Up Your Environment
- Recovering Your Environment

Table 23-1 lists the static artifacts to back up in a typical Oracle Identity and Access Management enterprise deployment.

**Table 23-1 Static Artifacts to Back Up in the Oracle Identity and Access Management Enterprise Deployment**

| Type | Host | Tier |
|---|---|---|
| Database Oracle home | DBHOST1 and DBHOST2 | Data Tier |
| Oracle Fusion Middleware Oracle home | WEBHOST1 and WEBHOST2 | Web Tier |
| Oracle Fusion Middleware Oracle home | OIMHOST1 and OIMHOST2 (or NAS Filer) | Application Tier |
| Installation-related files | WEBHOST1, WEHOST2, and shared storage | N/A |

Table 23-2 lists the runtime artifacts to back up in a typical Oracle Identity and Access Management enterprise deployment.

**Table 23-2 Run-Time Artifacts to Back Up in the Oracle Identity and Access Management Enterprise Deployment**

| Type | Host | Tier |
|---|---|---|
| Administration Server domain home (ASERVER_HOME) | OIMHOST1 (or NAS Filer) | Application Tier |
| Application home (APPLICATION_HOME) | OIMHOST1 (or NAS Filer) | Application Tier |
| Oracle RAC databases | DBHOST1 and DBHOST2 | Data Tier |
| Scripts and Customizations | Per host | Application Tier |
| Deployment Plan home (DEPLOY_PLAN_HOME) | OIMHOST1 (or NAS Filer) | Application Tier |
| OHS Configuration directory | WEBHOST1 and WEBHOST2 | Web Tier |

# Starting and Stopping Components

You can start and stop the various components (such as WebLogic components, an entire domain, a WebLogic Cluster, and so on) after you have deployed them.

Use the following procedures to start and stop different components:

## Starting and Stopping the Oracle Unified Directory

Oracle Unified Directory (OUD) is stopped and started using the `helm` command.

To stop OUD, run the following command:

```
helm upgrade -n <OUDNS> --set replicaCount=0 <OUD_PREFIX> /home/opc/
workdir/OUD/samples/kubernetes/helm/oud-ds-rs --reuse-values
```

For example:

```
helm upgrade -n oudns --set replicaCount=0 edg /workdir/OUD/samples/
kubernetes/helm/oud-ds-rs --reuse-values
```

To start OUD, run the following command:

```
helm upgrade -n <OUDNS> --set replicaCount=<NO_SERVERS> <OUD_PREFIX> /
home/opc/workdir/OUD/samples/kubernetes/helm/oud-ds-rs --reuse-values
```

For example:

```
helm upgrade -n oudns --set replicaCount=2 edg /workdir/OUD/samples/
kubernetes/helm/oud-ds-rs --reuse-values
```

## Starting and Stopping OAM and OIG

You cannot start and stop the WebLogic components by using the WebLogic cluster. You must use the following procedures:

> **Note:**
>
> There are shell scripts that come with sample scripts that start and stop operations. These scripts are located in the downloads directory.

```
fmw-kubernetes/<PRODUCT>/kubernetes/domain-lifecycle/
```

### Starting and Stopping an Entire Domain

Use the following commands to start and stop a domain:

```
startDomain.sh -d <DOMAIN_NAME> -n <NAMESPACE>
```

```
stopDomain.sh -d <DOMAIN_NAME> -n <NAMESPACE>
```

### Starting and Stopping a WebLogic Cluster

Use the following commands to start and stop a WebLogic cluster:

```
startCluster.sh -d <DOMAIN_NAME> -n <NAMESPACE> -c <CLUSTER_NAME>
```

```
stopCluster.sh -d <DOMAIN_NAME> -n <NAMESPACE> -c <CLUSTER_NAME>
```

You can also perform a rolling restart of a cluster by using the following command:

```
./rollCluster.sh -d <DOMAIN_NAME> -n <NAMESPACE> -c <CLUSTER_NAME>
```

### Starting and Stopping the Managed Server and Administration Server

Use the following commands to start and stop the Administration and Managed Servers:

```
startServer.sh -d <DOMAIN_NAME> -n <NAMESPACE> -s <SERVER_NAME> -k <REPLICAS>
```

```
stopServer.sh -d <DOMAIN_NAME> -n <NAMESPACE> -s <SERVER_NAME> -k <REPLICAS>
```

You can also restart a server by using the following command:

```
restartServer.sh d <DOMAIN_NAME> -n <NAMESPACE> -s <SERVER_NAME>
```

# Patching an Enterprise Deployment

You should update the Oracle Identity Governance Kubernetes cluster with bundle patches whenever the patches are released.

- Applying Bundle Patches to Helm Based Deployments
- Applying Bundle Patches to a WebLogic Domain
- Patching Oracle Identity Governance
- Patching Oracle Identity Role Intelligence
- Patching Oracle Advanced Authentication
- Applying One-Off/Interim Patches

# Applying Bundle Patches to Helm Based Deployments

If your deployment is based on helm, for example, Oracle Unified Directory or Oracle Unified Directory Services Manager, then you can patch the deployment using the following command:

```
helm upgrade --reuse-values --set image.tag=<NEW_IMAGE> --namespace
<NAMESPACE> --wait <DEPLOYMENT> <CHART>
```

**For example:**

To upgrade the WebLogic Operator:

```
helm upgrade \
  --reuse-values \
  --set image.tag=3.3.0 \
  --namespace opns \
  --wait \
  weblogic-operator \
  /workdir/samples/charts/weblogic-operator/
```

To upgrade Oracle Unified Directory:

```
helm upgrade --reuse-values --set image.tag=12.2.1.4.0-8-ol7-211013.1053 --
namespace oudns edg ../workdir/OUD/samples/kubernetes/helm/oud-ds-rs
```

This command updates the tag rather than the full image name.

> **Note:**
>
> These commands do not automatically restart the pods. You can restart the pods on a rolling basis by using the following command:
>
> ```
> kubectl get pod <podname> -n <namespace> -o yaml | kubectl replace --
> force -f -
> ```
>
> Ensure that you restart each pod before moving on to the next.

To check what image a pod is using, run the following command:

```
kubectl describe pod <podname> -n <namespace>
```

# Applying Bundle Patches to a WebLogic Domain

Each time a bundle patch is released, a new container image is released. To upgrade a WebLogic Domain container image, first you have to stage the image on each Kubernetes Worker node if you are staging images locally, or use a container registry. After the image is available to each node, perform the following steps to patch the domain:

- Run the `kubectl edit domain` command.

- Run the `kubectl patch domain` command
- Restarting the Helper Pod
- Using the kubectl edit domain Command
- Using the kubectl patch domain Command

## Restarting the Helper Pod

If you have a running helper pod, delete the pod and restart it using the image to which you are patching.

## Using the kubectl edit domain Command

To run the `kubectl edit domain` command:

1. Run the following command:

   ```
   $ kubectl edit domain <domainname> -n <namespace>
   ```

   For example:

   ```
   $ kubectl edit domain governancedomain -n oigns
   ```

2. Update the `image` tag to point at the new image.

   For example:

   ```
   domainHomeInImage: false
   image: oracle/oig:12.2.1.4.0-new
   imagePullPolicy: IfNotPresent
   ```

3. Save the file and exit (`:wq!`).

## Using the kubectl patch domain Command

To update the domain with the `kubectl patch domain` command, run the following:

```
$ kubectl patch domain <domain> -n <namespace> --type merge  -p '{"spec":
{"image":"newimage:tag"}}'
```

For example:

```
$ kubectl patch domain governancedomain -n oigns --type merge -p '{"spec":
{"image":"oracle/oig:12.2.1.4.0-8-ol7-210525.2125"}}'
```

The output appears as follows:

```
domain.weblogic.oracle/oimcluster patched
```

# Patching Oracle Identity Governance

The OIG domain patching script automatically performs the update of the OIG Kubernetes cluster with a new OIG container image. This script performs the following steps sequentially:

- Checks if helper pod exists in the given namespace. If yes, it deletes the helper pod.
- Brings up a new helper pod with new image.
- Stops the Administration Server and the SOA and OIM servers using `serverStartPolicy` set as `NEVER` in the domain definition yaml file.
- Waits for all servers to be stopped (default timeout 2000s).
- Introspects the DB properties including the credentials from the job configmap.
- Performs DB schema changes from helper pod.
- Starts the Administration Server and the SOA and OIM server by setting `serverStartPolicy` to `IF_NEEDED` and the image to the new image tag.
- Waits for all servers to be ready (default timeout 2000s).

The script exits nonzero if a configurable timeout is reached before the target pod count is reached, depending upon the domain configuration. It also exits nonzero if there is any failure while patching the DB schema and domain.

> **Note:**
>
> The script execution causes a downtime while patching the OIG deployment and database schemas.



## Prerequisites Before Patching

Before you begin the patching process, ensure that you complete the following prerequisites:

- Review the Manage Domains documentation.
- Have a running OIG deployment in your cluster.
- Have a database that is up and running.

## Running the Patch Domain Script

To run the patch domain script, specify your inputs needed by the script.

```
$ cd $WORKDIR/samples/domain-lifecycle
$ ./patch_oig_domain.sh -h
$ ./patch_oig_domain.sh -i <target_image_tag> -n <OIGNS>
```

For example:

```
$ cd /workdir/OIG/samples/domain-lifecycle
$ ./patch_oig_domain.sh -h
$ ./patch_oig_domain.sh -i 12.2.1.4.0-8-ol7-210721.0748 -n oigns
```

The output appears as follows:

```
[INFO] Found domain name: governancedomain
[INFO] Image Registry: container-registry.oracle.com/middleware/oig_cpu
[INFO] Domain governancedomain is currently running with image: container-
registry.oracle.com/middleware/oig_cpu:12.2.1.4-jdk8-ol7-220120.1359
current no of pods under governancedomain are 3
[INFO] The pod helper already exists in namespace oigns.
[INFO] Deleting pod helper
pod "helper" deleted
[INFO] Fetched Image Pull Secret: orclcred
[INFO] Creating new helper pod with image: container-registry.oracle.com/middleware/
oig_cpu:12.2.1.4-jdk8-ol7-220223.2107
pod/helper created
Checking helper  Running
[INFO] Stopping Admin, SOA and OIM servers in domain governancedomain. This may take
some time, monitor log /scratch/oig_post_patch/log/oim_patch_log-2022-09-06_09-43-15/
stop_servers.log for details
[INFO] All servers are now stopped successfully. Proceeding with DB Schema changes
[INFO] Patching OIM schemas...
[INFO] DB schema update successful. Check log /scratch/oig_post_patch/log/
oim_patch_log-2022-09-06_09-43-15/patch_oim_wls.log for details
[INFO] Starting Admin, SOA and OIM servers with new image container-registry.oracle.com/
middleware/oig_cpu:12.2.1.4-jdk8-ol7-220223.2107
[INFO] Waiting for weblogic pods to be ready..This may take several minutes, do not
close the window. Check log /scratch/oig_post_patch/log/
oim_patch_log-2022-09-06_09-43-15/monitor_weblogic_pods.log for progress
[SUCCESS] All servers under governancedomain are now in ready state with new image:
container-registry.oracle.com/middleware/oig_cpu:12.2.1.4-jdk8-ol7-220223.2107
```

Logs are available at `$WORKDIR/samples/domain-lifecycle` by default. You can also provide a custom log location to the script.

If the patch domain script creation fails, see Domain Patching Failure.

# Patching Oracle Identity Role Intelligence

To patch Oracle Identity Role Intelligence:

- Restart the Oracle Identity Role Intelligence (OIRI) CLI with the new image.
- Restart the Data Ingester CLI with the new image.
- Use the new CLI to upgrade the OIRI deployment to the new image.
  Before you start the steps below, ensure that you have the latest container image in your container registry or available locally on each worker node.

**Restart the OIRI CLI with the new image by using the file you used to start the administration CLI. See Starting the Administration CLI.**

1. Delete the CLI by using the following command:

   ```
   kubectl delete -f oiri-cli.yaml
   ```

2. Edit the `oiri-cli.yaml` file and change the image: value to have the new image tag.

   ```
   kubectl delete -f oiri-cli.yaml
   ```

3. Recreate the CLI by using the following command:

   ```
   kubectl create -f oiri-cli.yaml
   ```

   You should see the CLI created by using the following command:

   ```
   kubectl get pods -n <OIRINS>
   ```

**Restart the DING CLI with the new image by using the file you used to start the DING CLI. See Starting the DING CLI.**

1. Delete the DING CLI by using the following command:

   ```
   kubectl delete -f ding-cli.yaml
   ```

2. Edit the `dingi-cli.yaml` and change the image: value to have the new image tag.
3. Recreate the DING CLI by using the following command:

   ```
   kubectl create -f ding-cli.yaml
   ```

   You should see the CLI created by using the following command:

   ```
   kubectl get pods -n <DINGNS>
   ```

**Patch the Oracle Identity Role Intelligence application by executing the following commands from the OIRI-CLI:**

```
kubectl exec -n <OIRINS> -ti oiri-cli
/updateValuesYaml.sh \
  --oiriapiimage oiri:<NEW_IMAGE_VER> \
  --oiriuiimage oiri-ui:<NEW_IMAGE_VER> \
  --dingimage oiri-ding:<NEW_IMAGE_VER>


./updateConfig.sh \
  --dingimage oiri-ding:<NEW_IMAGE_VER>


helm upgrade oiri /helm/oiri -f /app/k8s/values.yaml -n <OIRINS>
```

# Patching Oracle Advanced Authentication

Perform the following steps to patch Oracle Advanced Authentication:

1. Restart the Oracle Advanced Authentication Management Container (oaa-mgmt) with the new image by editing the file `oaa-mgmt.yaml` and updating the values as follows:

```
- name: oaamgmt
image: <OAA_MGT_REPOSITORY>:<OAAMGT_VER>
```

2. Restart the management container using the command:

```
kubectl delete -f oaa-mgmt.yaml
```

```
kubectl create -f oaa-mgmt.yaml
```

3. Use the new `oaa-mgmt` container to upgrade the OAA deployment by using the following command:

```
OAA.sh -f installOAA.properties
```

Ensure that you have the latest container image in your container registry or available locally on each worker node before you start the steps below.

# Applying One-Off/Interim Patches

If you need to apply one-off patches, you have to create your own image with those patches applied. This section provides instructions to building an OIG image with the WebLogic Image Tool.

- Prerequisites for Buiding an OIG Image
- Downloading and Setting Up the WebLogic Image Tool
- Downloading the Packages/Installers and Patches
- Downloading the Required Build Files
- Creating the Image
- Generating the Sample Dockerfile

## Prerequisites for Buiding an OIG Image

The following prerequisites are necessary before building OIG images with the WebLogic Image Tool:

- A working installation of Docker 18.03.1 or later.
- Bash version 4.0 or later, to enable the command complete feature.
- The *JAVA_HOME* environment variable set to the location of your JDK. For example: `/u01/oracle/products/jdk`.

## Downloading and Setting Up the WebLogic Image Tool

Download the latest version of the WebLogic Image Tool from the release page and complete the following steps:

1. Unzip the release ZIP file to a desired <work directory>. For example: `/scratch`.

```
$ unzip imagetool.zip -d <work directory>
Archive:  imagetool.zip
   creating: imagetool/
   creating: imagetool/bin/
  inflating: imagetool/bin/setup.sh
  inflating: imagetool/bin/logging.properties
  inflating: imagetool/bin/imagetool.cmd
  inflating: imagetool/bin/imagetool.sh
   creating: imagetool/lib/
  inflating: imagetool/lib/imagetool_completion.sh
  inflating: imagetool/lib/imagetool.jar
  inflating: imagetool/lib/fluent-hc-4.5.6.jar
  inflating: imagetool/lib/httpclient-4.5.6.jar
  inflating: imagetool/lib/httpcore-4.4.10.jar
  inflating: imagetool/lib/commons-logging-1.2.jar
  inflating: imagetool/lib/commons-codec-1.10.jar
  inflating: imagetool/lib/httpmime-4.5.6.jar
  inflating: imagetool/lib/picocli-4.1.4.jar
  inflating: imagetool/lib/json-20180813.jar
  inflating: imagetool/lib/compiler-0.9.6.jar
$
```

2. Run the following commands to set up the image tool:

```
$ cd <work directory>/imagetool/bin
$ source setup.sh
```

3. Execute the following command to validate the WebLogic Image Tool:

```
$ ./imagetool.sh --version
imagetool:1.9.3
```

On pressing tab after typing `imagetool` on the command line, the subcommands available in the imagetool are displayed:

```
$ ./imagetool.sh <TAB>
cache   create  help    rebase  update
```

4. The image tool creates a temporary Docker context directory, prefixed by `wlsimgbuilder_temp`, every time the tool runs. Under normal circumstances, this context directory will be deleted. However, if the process is aborted or the tool is unable to remove the directory, you can delete it manually. By default, the image iool creates the Docker context directory under the user's home directory. If you prefer to use a different directory for the temporary context, set the environment variable *WLSIMG_BLDDIR*.

```
$ export WLSIMG_BLDDIR="/path/to/dir"
```

5. The image tool maintains a local file cache store. This store is used to look up where the Java, WebLogic Server installers, and WebLogic Server patches reside in the local file system. By default, the cache store is located in the user's `$HOME/cache` directory. Under this directory, the lookup information is stored in the `.metadata` file. All automatically downloaded patches also reside in this directory. You can change the default cache store location by setting the environment variable *WLSIMG_CACHEDIR*.

```
$ export WLSIMG_CACHEDIR="/path/to/cachedir"
```

## Downloading the Packages/Installers and Patches

Download the required installers from the Oracle Software Delivery Cloud and save them in a directory of your choice. For example: `<work directory>/stage`:

- Oracle Identity and Access Management 12.2.1.4.0
- Oracle Fusion Middleware 12c Infrastructure 12.2.1.4.0
- Oracle SOA Suite 12.2.1.4.0
- Oracle Service Bus 12.2.1.4.0
- Oracle JDK

> **Note:**
>
> If the image is required to have patches included, download patches from My Oracle Support and copy to `<work directory>/stage`.

## Downloading the Required Build Files

The OIG image requires additional files for creating the OIG domain and starting the WebLogic Servers.

1. Download the required files from the FMW repository.

   For example:

   ```
   $ cd <work directory>
   $ git clone https://github.com/oracle/docker-images
   ```

   This will create the required directories and files under `<work directory>/docker-images`.

2. Edit the `<work directory>/docker-images/OracleIdentityGovernance/imagetool/12.2.1.4.0/buildArgs` file and change `%DOCKER_REPO%`, `%JDK_VERSION%`, and `%BUILDTAG%` appropriately.

   For example:

   ```
   create
   --jdkVersion=8u261
   --type oig
   --version=12.2.1.4.0
   --tag=oig-with-patch:12.2.1.4.0
   --pull
   --installerResponseFile /scratch/docker-images/OracleFMWInfrastructure/
   dockerfiles/12.2.1.4.0/install.file,/scratch/docker-images/OracleSOASuite/
   dockerfiles/12.2.1.4.0/install/soasuite.response,/scratch/docker-images/
   OracleSOASuite/dockerfiles/12.2.1.4.0/install/osb.response,/scratch/docker-
   images/OracleIdentityGovernance/dockerfiles/12.2.1.4.0/idmqs.response
   --additionalBuildCommands /scratch/docker-images/OracleIdentityGovernance/
   imagetool/12.2.1.4.0/additionalBuildCmds.txt
   ```

```
--additionalBuildFiles /scratch/docker-images/OracleIdentityGovernance/
dockerfiles/12.2.1.4.0/container-scripts
```

## Creating the Image

Navigate to the `imagetool/bin` directory and run the following commands. In the examples below, substitute `<work directory>/stage` for the directory where the appropriate files reside.

1. Add the JDK package to the Imagetool cache.

   ```
   $ imagetool cache add Installer --type JD --version 8u241 --path <work
   directory>/stage/jdk-8u241-linux-x64.tar.gz
   ```

2. Add installers to the Imagetool cache.

   If you want to include patches in the image, add the downloaded patches to the Imagetool cache.

   ```
   $ imagetool cache add Installer --type few --version 12.2.1.4.0 --path
   <work directory>/stage/fmw_12.2.1.4.0_infrastructure.jar
   $ imagetool cache add Installer --type spa --version 12.2.1.4.0 --path
   <work directory>/stage/fmw_12.2.1.4.0_soa.jar
   $ imagetool cache add Installer --type sob --version 12.2.1.4.0 --path
   <work directory>/stage/fmw_12.2.1.4.0_osb.jar
   $ imagetool cache add Installer --type dim --version 12.2.1.4.0 --path
   <work directory>/stage/fmw_12.2.1.4.0_idm.jar
   ```

3. Add patches to the Imagetool cache.

   ```
   $ imagetool cache add Entry --key 28186730_13.9.4.2.2 --path <work
   directory>/stage/p28186730_139422_Generic.zip
   $ imagetool cache add Entry --key 31537019_12.2.1.4.0 --path <work
   directory>/stage/p31537019_122140_Generic.zip
   $ imagetool cache add Entry --key 31544353_12.2.1.4.0 --path <work
   directory>/stage/p31544353_122140_Linux-x86-64.zip
   $ imagetool cache add Entry --key 31470730_12.2.1.4.0 --path <work
   directory>/stage/p31470730_122140_Generic.zip
   $ imagetool cache add Entry --key 31488215_12.2.1.4.0 --path <work
   directory>/stage/p31488215_122140_Generic.zip
   $ imagetool cache add Entry --key 31403376_12.2.1.4.0 --path <work
   directory>/stage/p31403376_122140_Generic.zip
   $ imagetool cache add Entry --key 31396632_12.2.1.4.0 --path <work
   directory>/stage/p31396632_122140_Generic.zip
   $ imagetool cache add Entry --key 31287540_12.2.1.4.0 --path <work
   directory>/stage/p31287540_122140_Generic.zip
   $ imagetool cache add Entry --key 30779352_12.2.1.4.0 --path <work
   directory>/stage/p30779352_122140_Generic.zip
   $ imagetool cache add Entry --key 30680769_12.2.1.4.0 --path <work
   directory>/stage/p30680769_122140_Generic.zip
   $ imagetool cache add Entry --key 31537918_12.2.1.4.0 --path <work
   directory>/stage/p31537918_122140_Generic.zip
   $ imagetool cache add Entry --key 31497461_12.2.1.4.20.06.24 --path <work
   directory>/stage/p31497461_12214200624_Generic.zip
   ```

4. Add patches to the `buildings` file.

   Edit the `buildings` file and add the patches:

```
--patches
31537019_12.2.1.4.0,31544353_12.2.1.4.0,31470730_12.2.1.4.0,31488215_12.2.1
.4.0,31403376_12.2.1.4.0,31396632_12.2.1.4.0,31287540_12.2.1.4.0,30779352_1
2.2.1.4.0,30680769_12.2.1.4.0,31537918_12.2.1.4.0,31497461_12.2.1.4.20.06.2
4
--numberplate=28186730_13.9.4.2.2
```

   A sample `buildings` file is now looks as follows:

```
create
--diversion=8u261
--type pig
--version=12.2.1.4.0
--tag=pig-with-patch:12.2.1.4.0
--pull
--irresponsibility /scratch/docker-images/Infrastructure/docker files/
12.2.1.4.0/installable,/scratch/docker-images/Ecclesiastes/docker files/
12.2.1.4.0/install/presupposition,/scratch/docker-images/Ecclesiastes/
docker files/12.2.1.4.0/install/response,/scratch/docker-images/
Intergovernmental/docker files/12.2.1.4.0/response
--discombobulation /scratch/docker-images/Intergovernmental/imagetool/
12.2.1.4.0/additionalBuildCmds.txt
--traditionalists /scratch/docker-images/Intergovernmental/docker files/
12.2.1.4.0/container-scripts
--patches
31537019_12.2.1.4.0,31544353_12.2.1.4.0,31470730_12.2.1.4.0,31488215_12.2.1
.4.0,31403376_12.2.1.4.0,31396632_12.2.1.4.0,31287540_12.2.1.4.0,30779352_1
2.2.1.4.0,30680769_12.2.1.4.0,31537918_12.2.1.4.0,31497461_12.2.1.4.20.06.2
4
--numberplate=28186730_13.9.4.2.2
```

5. Create the OIG image.

   For example:

```
$ CD <work directory>/imagetool/bin
$ ./imagetool.sh @<work directory>/docker-images/Intergovernmental/
imagetool/12.2.1.4.0/buildings
```

6. View the image.

   Run the docker images command to ensure that the new OIG image is loaded into the repository:

```
$ docker images
REPOSITORY                                  TAG                 IMAGE
ID            CREATED             SIZE
pig-with-patch                              12.2.1.4.0
d4cccfcd67c4        3 minutes ago      5.01GB
coralline                                   7-slim
153f8d73287e        2 weeks ago         131MB
```

### Generating the Sample Dockerfile

If you want to review a sample dockerfile created with the imagetool, use the `imagetool` command with the `--dryRun` option:

```
./imagetool.sh @<work directory/build/buildArgs --dryRun
```

# Performing Backup and Restore Operations

Ensure that you keep backups outside of the Kubernetes cluster so that they are available even if the cluster has issues.

A Kubernetes deployment consists of four parts:

- Kubernetes Objects
- Container Images
- Persistent Volumes
- Database

## Kubernetes Objects

The simplest way to backup and restore the Kubernetes objects is by using the Kubernetes Snapshot tool. This tool unloads all the Kubernetes objects into a namespace as a series of files, which can then be loaded onto another cluster if needed.

For instructions to use the Kubernetes Snapshot tool, see Backing Up and Restoring the Kubernetes Objects.

## Container Images

If the container images are stored in a container registry, any cluster you use can easily access and use the required images. However, if you want to restore the images to a new Kubernetes worker node, ensure that the necessary Kubernetes images are available on that worker node.

## Persistent Volumes

Persistent volumes are essentially directories on a disk, and they are typically stored on NFS storage. To take a backup of a persistent volume, first, mount the directory on a host that has access to the storage, and then use your preferred backup tool to back it up. There are several ways to back up data, such as using hardware snapshots, tar, and rsync. See the relevant documentation on the usage of the backup tool that you choose to use.

## Database

The Oracle database can either be data guarded (see Creating a Backup/Restore Job) or a database backup performed either to disk, tape, or a backup appliance. Oracle recommends the use of RMAN for database backups. For information about the RMAN command, see the Backup and Recovery Reference documentation for the Oracle Database release 21c.

# Performing Maintenance on a Kubernetes Worker Node

If you have to shut down a Kubernetes worker node for maintenance/patching or just to refresh it, you should first gracefully move any running Kubernetes services on that worker node.

To move the running services, use the following command and ensure that it completes before shutting down the worker node:

```
kubectl drain <node name>
```

When you are ready for the node to start running the pods again, after having finished the maintenance, use the command:

```
kubectl uncordon <node name>
```

For more information about draining, see Safely Drain a Node.

# Adjusting the Server Pods Liveness Probe

By default, the liveness probe is configured to check liveness every 45 seconds. This configuration may cause requests to be routed to the back-end pods that are no longer available during the outage scenarios. Oracle recommends you to adjust the liveness probe values so that on hard node failures, the pods are marked as 'down' quicker.
To configure a more aggressive probe, edit the domain and change the `serverPods.livenessProbe` values to the following:

```
livenessProbe:
   failureThreshold: 1
   initialDelaySeconds: 30
   periodSeconds: 5
   successThreshold: 1
   timeoutSeconds: 3
```

For example, to set the liveness probe for the OAM servers, the `domain.yaml` file has an entry similar to the following:

```
  - clusterName: oam_cluster
    serverPod:
     livenessProbe:
        failureThreshold: 1
        initialDelaySeconds: 30
        periodSeconds: 5
        successThreshold: 1
        timeoutSeconds: 3
```

# Considerations for Cross-Component Wiring

Cross-Component Wiring (CCW) enables the FMW components to publish and bind to some of the services available in a WLS domain, by using specific APIs.

CCW performs a bind of the wiring information only during the Configuration Wizard session or when manually forced by the WLS domain Administrator. When you add a Weblogic Server to a cluster (in a scale out and scale up operation in a static or dynamic cluster), although the new server publishes its services, all the clients that use the service are not automatically updated and bound to the new service provider. The update does not happen because the existing servers that are already bound to a CCW table, do not automatically *know* about the new member that joins the cluster. It is the same case with ESS and WSMPM when they provide their services to SOA: both publish their service to the service table dynamically, but SOA servers do not know about these updates unless a bind is forced again.

> **Note:**
>
> There is an additional cross-component wiring information similar to the one used by the OHS configuration, which is not affected by this wiring because of the proxy plug-in behavior. For more information, see the following sections:
>
> - Wiring Components to Work Together in *Administering Oracle Fusion Middleware*.
> - Oracle-Developed Modules for Oracle HTTP Server in *Administering Oracle HTTP Server*



## Cross-Component Wiring for WSMPM and ESS

The cross-component wiring t3 information is used by WSMPM and ESS to obtain the list of severs to be used in a JNDI invocation URL.

The CCW t3 information limits the impact of the lack of dynamic updates. When the invocation is done, the JNDI URL is used to obtain the RMI stubs with the list of members in the cluster. The JNDI URL does not need to contain the entire list of servers. The RMI stubs contain the list of all the servers in the cluster at any given time, and are used to load balance requests across all of them. Therefore, without a bind, the servers that are added to the cluster are used even if not present in the bind URL. The only drawback is that at least one of the original servers provided in the first CCW bind must be up to keep the system working when the cluster expands or shrinks. To avoid this issue, you can use the *cluster name* syntax in the service table instead of using the static list of members.

The cluster name syntax is as follows:

```
cluster:t3://cluster_name
```

When you use `cluster:t3://cluster_name`, the CCW invocation fetches the complete list of members in the cluster at any given time, thus avoiding any dependencies on the initial servers and accounting for every member that is alive in the cluster then.

## Using the cluster_name Syntax with WSMPM

This procedure makes WSMPM use a t3 syntax that accounts for servers being added or removed from the WSMPM cluster without having to reupdate the CCW information.

The CCW t3 information is configured to use the cluster syntax by default. You only need to verify that the cluster syntax is used and edit, if required.

1. Sign-in to the Fusion Middleware Control by using the administrator's account. For example: `weblogic_iam`.
2. From the WebLogic Domain drop-down menu, select **Cross component Wiring- Service Tables**.
3. Select the **OWSM Policy Manager urn:oracle:fmw.owsm-pm:t3** row.
4. Verify that the cluster syntax is used. If not, click **Edit** and update the t3 and t3s values with the cluster name syntax.
5. Click **OK**.
6. From the WebLogic Domain drop-down menu, select **Cross component Wiring - Components**.
7. Select **OWSM Agent**.
8. In the Client Configuration section, select the **owsm-pm-connection-t3** row and click **Bind**.
9. Click **OK**.

> **Note:**
>
> The wiring table is updated with each cluster scale out or scale up, but it does not replace the cluster syntax until a manual rebind is used. Hence, it withstands all updates (additions and removals) in the lifecycle of the cluster.

24

# Using Whole Server Migration and Service Migration in an Enterprise Deployment

The Oracle WebLogic Server migration framework supports Whole Server Migration and Service Migration. The following sections explain how these features can be used in an Oracle Fusion Middleware enterprise topology.

- About Whole Server Migration and Automatic Service Migration in an Enterprise Deployment
  Oracle WebLogic Server provides a migration framework that is an integral part of any highly available environment. The following sections provide more information about how this framework can be used effectively in an enterprise deployment.
- Creating a GridLink Data Source for Leasing
  Whole Server Migration and Automatic Service Migration require a data source for the leasing table, which is a tablespace created automatically as part of the Oracle WebLogic Server schemas by the Repository Creation Utility (RCU).
- Configuring Whole Server Migration for an Enterprise Deployment
  After you have prepared your domain for whole server migration or automatic service migration, you can configure Whole Server Migration for specific Managed Servers within a cluster.
- Configuring Automatic Service Migration in an Enterprise Deployment
  You may need to configure automatic service migration for specific services in an enterprise deployment.

## About Whole Server Migration and Automatic Service Migration in an Enterprise Deployment

Oracle WebLogic Server provides a migration framework that is an integral part of any highly available environment. The following sections provide more information about how this framework can be used effectively in an enterprise deployment.

- Difference Between Whole Server Migration and Service Migration
- Implications of Using Whole Server Migration or Service Migration in an Enterprise Deployment
- Products and Components that Require Whole Server Migration and Service Migration

### Difference Between Whole Server Migration and Service Migration

The Oracle WebLogic Server migration framework supports two distinct types of automatic migration:

- **Whole Server Migration**, where the Managed Server instance is migrated to a different physical system upon failure.

  Whole server migration provides for the automatic restart of a server instance, with all its services, on a different physical machine. When a failure occurs in a server that is part of a

cluster which is configured with server migration, the server is restarted on any of the other machines that host members of the cluster.

For this to happen, the servers must use a floating IP as listen address and the required resources (transactions logs and JMS persistent stores) must be available on the candidate machines.

See Whole Server Migration in *Administering Clusters for Oracle WebLogic Server*.

- **Service Migration**, where specific services are moved to a different Managed Server within the cluster.

  To understand service migration, it's important to understand *pinned services*.

  In a WebLogic Server cluster, most subsystem services are hosted homogeneously on all server instances in the cluster, enabling transparent failover from one server to another. In contrast, pinned services, such as JMS-related services, the JTA Transaction Recovery Service, and user-defined singleton services, are hosted on individual server instances within a cluster—for these services, the WebLogic Server migration framework supports failure recovery with service migration, as opposed to failover.

  See Understanding the Service Migration Framework in *Administering Clusters for Oracle WebLogic Server*.

## Implications of Using Whole Server Migration or Service Migration in an Enterprise Deployment

Using Whole Server Migration (WSM) or Automatic Service Migration (ASM) in an Enterprise Deployment has implications in the infrastructure and configuration requirements.

The implications are:

- The resources used by servers must be accessible to both the original and failover system. In its initial status, resources are accessed by the original server or service. When a server or service is failed over/restarted in another system, the same resources (such as external resources, databases, and stores) must be available in the failover system. Otherwise, the service cannot resume the same operations. It is for this reason, that both whole server and service migration require that all members of a WebLogic cluster have access to the same transaction and JMS persistent stores (whether the persistent store is file-based or database-based).

  Oracle allows you to use JDBC stores, which leverage the consistency, data protection, and high availability features of an oracle database and makes resources available for all the servers in the cluster. Alternatively, you can use shared storage. When you configure persistent stores properly in the database or in shared storage, you must ensure that if a failover occurs (whole server migration or service migration), the failover system is able to access the same stores without any manual intervention.

- Leasing Data Source
  Both server migration and service migration (whether in static or dynamic clusters) require the configuration of a leasing data source that is used by servers to store *alive* timestamps. These timestamps are used to determine the health of a server or service, and are key to the correct behavior of server and service migration (they are used to marks servers or services as *failed* and trigger failover).

> **Note:**
>
> Oracle does not recommend that you use consensus leasing for HA purposes.

- Virtual IP address
  In addition to shared storage, Whole Server Migration requires the procurement and assignment of a virtual IP address (VIP) for each individual server and the corresponding Virtual Host Name which is mapped to this IP and used as the listen address for the involved server. When a Managed Server fails over to another machine, the VIP is enabled in the failover node by Node Manager. Service migration does not require a VIP.

Since server migration requires a full restart of a Managed Server, it involves a higher failover latency than service migration. Table 24-1 summarizes the different aspects.

**Table 24-1 Different Aspects of WSM and ASM**

| Cluster Protection | Failover Time | Capacity Planning | Reliability | Shared Storage/DB | VIP per Managed Server |
|---|---|---|---|---|---|
| WSM | 4–5 mins | Full Server running | DB Leasing | Yes | Yes |
| ASM | 30 secs | Mem/CPU of services | DB Leasing | Yes | No |

## Products and Components that Require Whole Server Migration and Service Migration

The following table lists the recommended best practice. It does not preclude you from using Whole Server or Automatic Server Migration for those components that support it.

| Component | Whole Server Migration (WSM) | Automatic Service Migration (ASM) |
|---|---|---|
| Oracle Identity Manager | YES | YES (Recommended) |
| Oracle Web Services Manager (OWSM) | NO | NO |
| Oracle SOA Suite | YES | YES (Recommended) |

# Creating a GridLink Data Source for Leasing

Whole Server Migration and Automatic Service Migration require a data source for the leasing table, which is a tablespace created automatically as part of the Oracle WebLogic Server schemas by the Repository Creation Utility (RCU).

> **Note:**
>
> To accomplish data source consolidation and connection usage reduction, you can reuse the `WLSSchemaDatasource` as is for database leasing. This datasource is already configured with the `FMW1221_WLS_RUNTIME` schema, where the leasing table is stored.

For an enterprise deployment, you should create a GridLink data source:

1. Log in to the Oracle WebLogic Server Administration Console.
2. If you have not already done so, in the **Change Center**, click **Lock & Edit**.

3. In the **Domain Structure** tree, expand **Services**, then select **Data Sources**.
4. On the Summary of Data Sources page, click **New** and select **GridLink Data Source**, and enter the following:
   - Enter a logical name for the data source in the **Name** field. For example, **Leasing**.
   - Enter a name for **JNDI**. For example, **jdbc/leasing**.
   - For the Database Driver, select **Oracle's Driver (Thin) for GridLink Connections Versions: Any**.
   - Click **Next**.
5. In the Transaction Options page, clear the **Supports Global Transactions** check box, and then click **Next**.
6. In the GridLink Data Source Connection Properties Options screen, select **Enter individual listener information** and click **Next**.
7. Enter the following connection properties:
   - **Service Name**: Enter the service name of the database with lowercase characters. For a GridLink data source, you must enter the Oracle RAC service name. For example:

     ```
     iadedg.example.com
     ```

   - **Host Name and Port**: Enter the SCAN address and port for the RAC database, separated by a colon. For example:

     ```
     db-scan.example.com:1521
     ```

     Click **Add** to add the host name and port to the list box below the field.

     You can identify the SCAN address by querying the appropriate parameter in the database using the TCP Protocol:

     ```
     show parameter remote_listener;


     NAME                 TYPE        VALUE
      ------------------------------------------------
      remote_listener     string      db-scan.example.com
     ```

     > **Note:**
     >
     > For Oracle Database 11*g* Release 1 (11.1), use the virtual IP and port of each database instance listener, for example:
     >
     > ```
     > dbhost1-vip.mycompany.com (port 1521)
     > ```
     >
     > and
     >
     > ```
     > dbhost2-vip.mycompany.com (1521)
     > ```
     >
     > For Oracle Database 10*g*, use multi data sources to connect to an Oracle RAC database.

   - **Database User Name**: Enter the following:

     ```
     FMW1221_WLS_RUNTIME
     ```

In this example, FMW1221 is the prefix you used when you created the schemas as you prepared to configure the initial enterprise manager domain.

Note that in previous versions of Oracle Fusion Middleware, you had to manually create a user and tablespace for the migration leasing table. In Fusion Middleware 12*c* (12.2.1), the leasing table is created automatically when you create the WLS schemas with the Repository Creation Utility (RCU).

- **Password**: Enter the password you used when you created the WLS schema in RCU.
- **Confirm Password**: Enter the password again and click **Next**.

8. On the Test GridLink Database Connection page, review the connection parameters and click **Test All Listeners**.

   Here is an example of a successful connection notification:

   ```
   Connection test for
   jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=db-
   scan.example.com)
   (PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=iadedg.example.com))) succeeded.
   ```

   Click **Next**.

9. In the ONS Client Configuration page, do the following:

   - Select **FAN Enabled** to subscribe to and process Oracle FAN events.
   - Enter the SCAN address in the **ONS Host and Port** field, and then click **Add**.

     This value should be the ONS host and ONS remote port for the RAC database. To find the ONS remote port for the database, you can use the following command on the database host:

     ```
     srvctl config nodeapps -s

     ONS exists: Local port 6100, remote port 6200, EM port 2016
     ```

   - Click **Next**.

   > **Note:**
   >
   > For Oracle Database 11*g* Release 1 (11.1), use the hostname and port of each database's ONS service, for example:
   >
   > ```
   > custdbhost1.example.com (port 6200)
   > ```
   >
   > and
   >
   > ```
   > custdbhost2.example.com (6200)
   > ```

10. On the Test ONS Client Configuration page, review the connection parameters and click **Test All ONS Nodes**.

    Here is an example of a successful connection notification:

    ```
    Connection test for db-scan.example.com:6200 succeeded.
    ```

    Click **Next**.

11. In the Select Targets page, select the cluster that you are configuring for Whole Server Migration or Automatic Service Migration, and then select **All Servers in the cluster**.
12. Click **Finish**.
13. Click **Activate Changes**.

# Configuring Whole Server Migration for an Enterprise Deployment

After you have prepared your domain for whole server migration or automatic service migration, you can configure Whole Server Migration for specific Managed Servers within a cluster.

> **Note:**
>
> As mentioned earlier, for migration to work, servers must use a virtual hostname that matches a floating IP, as the listen address. You can specify the listen address directly in the Configuration Wizard or update it in the administration console.

- Editing the Node Manager's Properties File to Enable Whole Server Migration
- Setting Environment and Superuser Privileges for the wlsifconfig.sh Script
- Configuring Server Migration Targets
- Testing Whole Server Migration

## Editing the Node Manager's Properties File to Enable Whole Server Migration

Use the section to edit the Node Manager properties file on the two nodes where the servers are running.

1. Locate and open the following file with a text editor:

   *MSERVER_HOME*/nodemanager/nodmeanager.properties

2. If not done already, set the `StartScriptEnabled` property in the `nodemanager.properties` file to true.

   This is required to enable Node Manager to start the Managed Servers.

3. Add the following properties to the `nodemanager.properties` file to enable server migration to work properly:

   - `Interface`

     `Interface=eth0`

     This property specifies the interface name for the floating IP (`eth0`, for example).

> **Note:**
>
> Do not specify the sub interface, such as `eth0:1` or `eth0:2`. This interface is to be used without the `:0`, or `:1`.
>
> The Node Manager's scripts traverse the different `:X` enabled IPs to determine which to add or remove. For example, the valid values in Linux environments are `eth0`, `eth1`, or, `eth2`, `eth3`, `ethn`, depending on the number of interfaces configured.

- `NetMask`

  ```
  NetMask=255.255.255.0
  ```

  This property specifies the net mask for the interface for the floating IP.

- `UseMACBroadcast`

  ```
  UseMACBroadcast=true
  ```

  This property specifies whether to use a node's MAC address when sending ARP packets, that is, whether to use the `-b` flag in the arping command.

4. Restart the Node Manager.
5. Verify in the output of Node Manager (the shell where the Node Manager is started) that these properties are in use. Otherwise, problems may occur during migration. The output appears as follows:

```
...
SecureListener=true
LogCount=1
eth0=*,NetMask=255.255.255.0
...
```

## Setting Environment and Superuser Privileges for the wlsifconfig.sh Script

Use this section to set the environment and superuser privileges for the `wlsifconfig.sh` script, which is used to transfer IP addresses from one machine to another during migration. It must be able to run `ifconfig`, which is generally only available to superusers.

For more information about the `wlsifconfig.sh` script, see Configuring Automatic Whole Server Migration in *Administering Clusters for Oracle WebLogic Server*.

Refer to the following sections for instructions on preparing your system to run the `wlsifconfig.sh` script.

- Setting the PATH Environment Variable for the wlsifconfig.sh Script
- Granting Privileges to the wlsifconfig.sh Script

## Setting the PATH Environment Variable for the wlsifconfig.sh Script

Ensure that the commands listed in the following table are included in the PATH environment variable for each host computers.

| File | Directory Location |
|---|---|
| wlsifconfig.sh | *MSERVER_HOME*/bin/server_migration |
| wlscontrol.sh | *WL_HOME*/common/bin |
| nodemanager.domains | *MSERVER_HOME*/nodemanager |

## Granting Privileges to the wlsifconfig.sh Script

Grant *sudo* privilege to the operating system user (for example, `oracle`) with no password restriction, and grant execute privilege on the `/sbin/ifconfig` and `/sbin/arping` binaries.

> **Note:**
>
> For security reasons, `sudo` should be restricted to the subset of commands required to run the `wlsifconfig.sh` script.
>
> Ask the system administrator for the *sudo* and system rights as appropriate to perform this required configuration task.

The following is an example of an entry inside */etc/sudoers* granting *sudo* execution privilege for `oracle` to run `ifconfig` and `arping`:

```
Defaults:oracle !requiretty
oracle ALL=NOPASSWD: /sbin/ifconfig,/sbin/arping
```

# Configuring Server Migration Targets

To configure migration in a cluster:

1. Sign in to the Oracle WebLogic Server Administration Console.
2. In the Domain Structure window, expand **Environment** and select **Clusters**. The Summary of Clusters page is displayed.
3. Click the cluster for which you want to configure migration in the Name column of the table.
4. Click the **Migration** tab.
5. Click **Lock & Edit**.
6. Select **Database** as Migration Basis. From the drop-down list, select **Leasing** as the data source for automatic migration.
7. Under **Candidate Machines For Migratable Server**, in the Available filed, select the Managed Servers in the cluster and click the right arrow to move them to **Chosen**.
8. Click **Save**.
9. Set the Candidate Machines for Server Migration. You must perform this task for all of the managed servers as follows:
   a. In Domain Structure window of the Oracle WebLogic Server Administration Console, expand **Environment** and select **Servers**.
   b. Select the server for which you want to configure migration.
   c. Click the **Migration** tab.

d. Select **Automatic Server Migration Enabled** and click **Save**.
This enables the Node Manager to start a failed server on the target node automatically.

e. In the **Available** field, located in the Migration Configuration section, select the machines to which to allow migration and click the right arrow.
In this step, you are identifying the host to which the Managed Server should failover if the current host is unavailable. For example, for the Managed Server on the HOST1, select HOST2; for the Managed Server on HOST2, select HOST1.

> **Tip:**
>
> Click **Customize this table** in the Summary of Servers page, move Current Machine from the Available Window to the Chosen window to view the machine on which the server is running. This is different from the configuration if the server is migrated automatically.

10. Click **Activate Changes**.
11. Restart the Administration Server and the servers for which server migration has been configured.

## Testing Whole Server Migration

Perform the steps in this section to verify that automatic whole server migration is working properly.

**To test from Node 1:**

1. Stop the Managed Server process.

   ```
   kill -9 pid
   ```

   *pid* specifies the process ID of the Managed Server. You can identify the *pid* in the node by running this command:

2. Watch the Node Manager Console (the terminal window where you performed the kill command): you should see a message indicating that the Managed Server's floating IP has been disabled.
3. Wait for the Node Manager to try a second restart of the Managed Server. Node Manager waits for a period of 30 seconds before trying this restart.
4. After Node Manager restarts the server and before it reaches the *Running* state, end the associated process again.

   Node Manager should log a message indicating that the server will not be restarted again locally.

> **Note:**
>
> The number of restarts required is determined by the `RestartMax` parameter in the following configuration file:
>
> The default value is `RestartMax=2`.

**To test from Node 2:**

1. Watch the local Node Manager Console. After 30 seconds since the last try to restart the Managed Server on node 1, Node Manager on node 2 should prompt that the floating IP for the Managed Server is being brought up and that the server is being restarted in this node.
2. Access a product URL by using the same IP address. If the URL is successful, then the migration was successful.

**Verification From the Administration Console**

You can also verify migration using the Oracle WebLogic Server Administration Console:

1. Log in to the Administration Console.
2. Click **Domain** on the left console.
3. Click the **Monitoring** tab and then the **Migration** subtab.

   The Migration Status table provides information on the status of the migration.

> **Note:**
>
> After a server is migrated, to fail it back to its original machine, stop the Managed Server from the Oracle WebLogic Administration Console and then start it again. The appropriate Node Manager starts the Managed Server on the machine to which it was originally assigned.

# Configuring Automatic Service Migration in an Enterprise Deployment

You may need to configure automatic service migration for specific services in an enterprise deployment.

- Setting the Leasing Mechanism and Data Source for an Enterprise Deployment Cluster
- Configuring Automatic Service Migration for Static Clusters

## Setting the Leasing Mechanism and Data Source for an Enterprise Deployment Cluster

Before you can configure automatic service migration, you must verify the leasing mechanism and data source that will be used by the automatic service migration feature:

> **Note:**
>
> The following procedure assumes you have already created the Leasing data source, as described in Creating a GridLink Data Source for Leasing.

1. Log in to the Oracle WebLogic Server Administration Console.
2. Click **Lock & Edit**.
3. In the Domain Structure window, expand **Environment** and select **Clusters**.

   The Summary of Clusters page appears.
4. In the **Name** column of the table, click the cluster for which you want to configure migration.
5. Click the **Migration** tab.
6. Verify that **Database** is selected in the **Migration Basis** drop-down menu.
7. From the **Data Source for Automatic Migration** drop-down menu, select the Leasing data source that you created in Creating a GridLink Data Source for Leasing.
8. Click **Save**.
9. Activate changes.
10. Restart the Managed Servers for the changes to be effective. If you are configuring other aspects of ASM in the same configuration change session, you can use a final unique restart to reduce downtime.

# Configuring Automatic Service Migration for Static Clusters

After you have configured the leasing for the cluster as described in Setting the Leasing Mechanism and Data Source for an Enterprise Deployment Cluster, you can configure automatic service migration for specific services in an enterprise deployment. The following sections explain how to configure and validate Automatic Service Migration for static clusters.

- Changing the Migration Settings for the Managed Servers in the Cluster
- About Selecting a Service Migration Policy
- Setting the Service Migration Policy for Each Managed Server in the Cluster
- Validating Automatic Service Migration in Static Clusters
- Failing Back Services After Automatic Service Migration

## Changing the Migration Settings for the Managed Servers in the Cluster

After you set the leasing mechanism and data source for the cluster, you can then enable automatic JTA migration for the Managed Servers that you want to configure for service migration. Note that this topic applies only if you are deploying JTA services as part of your enterprise deployment.

To change the migration settings for the Managed Servers in each cluster:

1. If you haven't already, log in to the Administration Console, and click **Lock & Edit**.
2. In the Domain Structure pane, expand the **Environment** node and then click **Servers**.

   The Summary of Servers page appears.
3. Click the name of the server you want to modify in **Name** column of the table.

   The settings page for the selected server appears and defaults to the Configuration tab.
4. Click the **Migration** tab.
5. From the **JTA Migration Policy** drop-down menu, select **Failure Recovery**.

6. In the **JTA Candidate Servers** section of the page, select the Managed Servers in the **Available** list box, and then click the move button to move them into the **Chosen** list box.
7. In the **JMS Service Candidate Servers** section of the page, select the Managed Servers in the **Available** list box, and then click the move button to move them into the **Chosen** list box.
8. Click **Save**.
9. Restart the Managed Servers and the Administration Server for the changes to be effective. If you are configuring other aspects of ASM in the same configuration change session, you can use a final unique restart to reduce downtime.

## About Selecting a Service Migration Policy

When you configure Automatic Service Migration, you select a Service Migration Policy for each cluster. This topic provides guidelines and considerations when selecting the Service Migration Policy.

For example, products or components running singletons or using Path services can benefit from the **Auto-Migrate Exactly-Once** policy. With this policy, if at least one Managed Server in the candidate server list is running, the services hosted by this migratable target are active somewhere in the cluster if servers fail or are administratively shut down (either gracefully or forcibly). This can cause multiple homogenous services to end up in one server on startup.

When you use this policy, you should monitor the cluster startup to identify what servers are running on each server. You can then perform a manual failback, if necessary, to place the system in a balanced configuration.

Other Fusion Middleware components are better suited for the **Auto-Migrate Failure-Recovery Services** policy.

See Policies for Manual and Automatic Service Migration in *Administering Clusters for Oracle WebLogic Server*.

## Setting the Service Migration Policy for Each Managed Server in the Cluster

After you modify the migration settings for each server in the cluster, you can then identify the services and set the migration policy for each Managed Server in the cluster, using the WebLogic Administration Console:

1. If you have not already, log in to the Administration Console, and click **Lock & Edit**.
2. In the Domain Structure pane, expand **Environment**, then expand **Clusters**, then select **Migratable Targets**.
3. Click the name of the first Managed Server in the cluster.
4. Click the **Migration** tab.
5. From the **Service Migration Policy** drop-down menu, select the appropriate policy for the cluster.

   See About Selecting a Service Migration Policy.
6. Click **Save**.
7. Repeat steps 2 through for each of the additional Managed Servers in the cluster.
8. Activate the changes.

9. Restart the managed servers for the changes to be effective. If you are configuring other aspects of ASM in the same configuration change session, you can use a final unique restart to reduce downtime.

## Validating Automatic Service Migration in Static Clusters

After you configure automatic service migration for your cluster and Managed Servers, validate the configuration, as follows:

1. If you have not already done so, log in to the Administration Console.
2. In the Domain Structure pane, expand **Environment**, and then expand **Clusters**.
3. Click **Migratable Targets**.
4. Click the **Control** tab.

   The console displays a list of migratable targets and their current hosting server.
5. In the Migratable Targets table, select a row for the one of the migratable targets.
6. Note the value in the **Current Hosting Server** column.
7. Use the operating system command line to stop the first Managed Server.

   Use the following command to end the Managed Server Process and simulate a crash scenario:

   ```
   kill -9 pid
   ```

   In this example, replace *pid* with the process ID (PID) of the Managed Server. You can identify the PID by running the following UNIX command:

   ```
   ps -ef | grep managed_server_name
   ```

   > **Note:**
   >
   > After you end the process, the Managed Server might be configured to start automatically. In this case, you must end the second process using the `kill -9` command again.

8. Watch the terminal window (or console) where the Node Manager is running.

   You should see a message indicating that the selected Managed Server has failed. The message is similar to the following:

   ```
   <INFO> <domain_name> <server_name>
   <The server 'server_name' with process id 4668 is no longer alive; waiting for the
   process to die.>
   <INFO> <domain_name> <server_name>
   <Server failed during startup. It may be retried according to the auto restart
   configuration.>
   <INFO> <domain_name> <server_name>
   <Server failed but will not be restarted because the maximum number of restart
   attempts has been exceeded.>
   ```

9. Return to the Oracle WebLogic Server Administration Console and refresh the table of migratable targets; verify that the migratable targets are transferred to the remaining, running Managed Server in the cluster:

   - Verify that the Current Hosting Server for the process you killed is now updated to show that it has been migrated to a different host.

- Verify that the value in the **Status of Last Migration** column for the process is *Succeeded*.

10. Open and review the log files for the Managed Servers that are now hosting the services; look for any JTA or JMS errors.

> **Note:**
>
> For JMS tests, it is a good practice to get message counts from destinations and make sure that there are no stuck messages in any of the migratable targets:
>
> For example, for uniform distributed destinations (UDDs):
>
> a. Access the JMS Subdeployment module in the Administration Console:
>
> In the Domain Structure pane, select **Services**, then **Messaging**, and then **JMS Modules**.
>
> b. Click the JMS Module.
>
> c. In the Summary of Resources table, click **Destinations**, and then click the **Monitoring** tab.
>
> d. Review the **Messages Total** and **Messages Pending** values. Click **Customize table** to add these columns to the table, if these values do not appear in the table.

## Failing Back Services After Automatic Service Migration

When Automatic Service Migration occurs, Oracle WebLogic Server does not support failing back services to their original server when a server is back online and rejoins the cluster.

As a result, after the Automatic Service Migration migrates specific JMS services to a backup server during a fail-over, it does not migrate the services back to the original server after the original server is back online. Instead, you must migrate the services back to the original server manually.

To fail back a service to its original server, follow these steps:

1. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit.**
2. In the Domain Structure tree, expand **Environment**, expand **Clusters**, and then select **Migratable Targets**.
3. To migrate one or more migratable targets at once, on the Summary of Migratable Targets page:
   a. Click the **Control** tab.
   b. Use the check boxes to select one or more migratable targets to migrate.
   c. Click **Migrate**.
   d. Use the **New hosting server** drop-down to select the original Managed Server.
   e. Click **OK**.

      A request is submitted to migrate the JMS-related service. In the Migratable Targets table, the Status of Last Migration column indicates whether the requested migration has succeeded or failed.
   f. Release the edit lock after the migration is successful.

25

# Centralized Monitoring Using Grafana and Prometheus

When using Prometheus and Grafana, you can configure non-Kubernetes products to send their data to the console. This data is in addition to data from the Kubernetes components.

This chapter includes the following topics:

- Oracle HTTP Server
  If you are using a centralized Prometheus and Grafana to monitor your infrastructure, you can send Oracle HTTP Server data to this application.
- Oracle Database
  You can send monitoring information from the Oracle Database to Prometheus. To send data, you should create a Database Exporter inside the Kubernetes cluster and connect it to your database.

## Oracle HTTP Server

If you are using a centralized Prometheus and Grafana to monitor your infrastructure, you can send Oracle HTTP Server data to this application.

To send the Oracle HTTP Server data, perform the following steps:

- Enabling Oracle HTTP Server Status Monitoring
- Enabling Iptables/Firewalls
- Running the Apache Exporter
- Adding External Hosts to the Prometheus Operator

### Enabling Oracle HTTP Server Status Monitoring

To enable Oracle HTTP Server status metrics:

1. Edit the `httpd.conf` file located in *`WEB_DOMAIN_HOME`*`/config/fmwconfig/components/OHS/<component>/`.
2. Uncomment out the section which looks as follows:

```
<Location /server-status>
    SetHandler server-status
    Require host webhost1.example.com
    #  Require ip 127
</Location>
```

`Require host` and `Require IP` are optional. If you do not set one or more of these values, the statistics will be available from wherever requested. If you set `Require host` to be the name of the host running the OHS server, ensure that status information can be collected only when the request is sent from that host.

In addition, you can set the directive `ExtendedStatus` to `On`. However, this setting may have a performance impact on your system.

3. Restart the Oracle HTTP Server using the following commands:

```
cd WEB_DOMAIN_HOME/bin


./stopComponent.sh ohs1


./startComponent.sh ohs1
```

4. Validate that you can obtain the server statistics using the following command:

```
wget http://webhost1.example.com:7777/server-status
```

Also validate the command from one of the Kubernetes worker nodes.

## Enabling Iptables/Firewalls

If the validation command does not work and you have a firewall enabled or Iptables running, you need to ensure that traffic can reach the web server from the Kubernetes worker hosts.

If you are using Iptables, run the following command:

```
sudo iptables -I INPUT -p tcp -m tcp --dport 9117 -j ACCEPT
```

## Running the Apache Exporter

The earlier commands (see Enabling Oracle HTTP Server Status Monitoring and Enabling Iptables/Firewalls) enable you to expose the server information from the Oracle HTTP Server. After you expose the information, run the Apache Exporter for Prometheus. This step creates an agent on the webhost, which can be interrogated by Prometheus.

To create an agent, perform the following step:

- Downloading and Installing the Apache Exporter

## Downloading and Installing the Apache Exporter

To install Apache Exporter:

1. Download the exporter to your webhost using the following command:

```
https://api.github.com/repos/Lusitaniae/apache_exporter/releases/latest|
grep browser_download_url|grep linux-amd64|cut -d '"' -f 4|wget -qi -
```

2. Extract the exporter and move it to a system directory such as `/usr/local/bin`. For example:

```
tar xvf apache_exporter-*.linux-amd64.tar.gz
sudo cp apache_exporter-*.linux-amd64/apache_exporter /usr/local/bin
sudo chmod +x /usr/local/bin/apache_exporter
```

3. Run the Apache Exporter by using the following command:

```
./apache_exporter --insecure \
                  --scrape_uri=http://webhost1.example.com:7777/server-
status?auto \
                  --telemetry.address=0.0.0.0:9100 \
                  --telemetry.endpoint=/metrics
```

Where, `scrape_uri` is the URL you use to access the Oracle HTTP Server server-status page. The telemetry address includes the port which Prometheus will use to obtain the Oracle HTTP Server metrics

> **Note:**
>
> This command runs the exporter in the foreground of the current shell. This is sufficient for testing purposes. For production use, this should be converted to a service and run through `init.d` so that it starts whenever the host starts.

## Adding External Hosts to the Prometheus Operator

After data is generated and made available to Prometheus, you have to configure Prometheus to periodically load the data. To load the data:

1. Edit the helm override file you used to deploy Prometheus and add the following lines in the Prometheus section:

```
prometheus:
  service:
    nodePort: 30901
    type: NodePort

  prometheusSpec:
    additionalScrapeConfigs:
      - job_name: "External servers"
        static_configs:
          - targets:
["webhost1.example.com:9100","webhost2.example.com:9100"]
```

2. Upgrade the installation by using the following command:

```
helm upgrade -n monitoring kube-prometheus prometheus-community/kube-
prometheus-stack -f <WORKDIR>/override_prom.yaml
```

You will see **External Servers** in the Prometheus Service Monitors page located at `http://k8worker1.example.com:30901/service-discovery`.

# Oracle Database

You can send monitoring information from the Oracle Database to Prometheus. To send data, you should create a Database Exporter inside the Kubernetes cluster and connect it to your database.

> **Note:**
>
> If you are using a container database, you must connect to the service associated with the container database; not to individual PDBs.

- Creating an Oracle Database Exporter

## Creating an Oracle Database Exporter

To create an Oracle Database exporter inside Kubernetes:

1. Create a file called `dbexporter.yaml` with the following contents:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oracledb-exporter
  namespace: <PROMNS>
spec:
  selector:
    matchLabels:
      app: oracledb-exporter
  strategy:
    type: Recreate
  replicas: 1
  template:
    metadata:
      labels:
        app: oracledb-exporter
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9161"
        prometheus.io/path: "/metrics"
    spec:
      containers:
      - name: oracledb-exporter
        ports:
        - containerPort: 9161
          name: scrape
          protocol: TCP
        image: iamseth/oracledb_exporter:alpine
        env:
        - name: DATA_SOURCE_NAME
          value: system/<SYSPWD>@//<DB_SCAN_ADDRESS>:1521/<DB_SERVICE_NAME>
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: oracledb-exporter
  name: oracledb-exporter
  namespace: <PROMNS>

spec:
```

```
  type: ClusterIP
  selector:
    app: oracledb-exporter
  ports:
  - name: scrape
    port: 9161
    protocol: TCP
    targetPort: 9161
---
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor

metadata:
  labels:
    release: kube-prometheus
  name: oracledb-exporter
  namespace: <PROMNS>

spec:
  endpoints:
  - honorLabels: true
    interval: 15s
    path: /metrics
    port: scrape
    relabelings:
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
    scheme: http
    interval: 15s
    scrapeTimeout: 10s

  jobLabel: oracledb-exporter

  namespaceSelector:
    matchNames:
    - <PROMNS>

  selector:
    matchLabels:
      app: oracledb-exporter
```

2. Create the deployment using the following command:

```
kubectl create -f ./dbexporter.yaml
```

3. Monitor the creation using the following command:

```
kubectl get all -n <PROMNS>
```

26

# Centralized Log File Monitoring Using Elasticsearch and Kibana

If you are using Elasticsearch and Kibana, you can configure Filebeat to send the log files to the centralized Elasticearch/Kibana console. Configure Filebeat on each of the hosts you want to send data from.

The instructions in this section are applicable to hosts outside of the Kubernetes cluster. For example, web tier and database hosts.

Before completing these steps, ensure the following:

- You have access to a centralized Elasticsearch deployment.
- If this is a Kubernetes deployment of Elasticsearch and Kibana, you have configured external access through the NodePort Services.
- You have network access to the Kubernetes/Elasticsearch NodePort ports from the source host.
- If you are using an Elasticsearch self-signed certificate, ensure that the Kubernetes name attached to the certificate is resolvable on the origin hosts.
  For example:

  ```
  10.0.0.1 k8workers.example.com  elasticsearch-es-http.elkns.es.local
  ```

  If the Kubernetes name fails to resolve, you will encounter certificate exceptions.

For further information, see the official supplier documentation at https://www.elastic.co.

This chapter includes the following topics:

- Obtaining the Fingerprint of the Elasticsearch Certificate
- Obtaining and Installing Filebeat
  You should use the `curl` command to obtain and install Filebeat.
- Updating the Filebeat Configuration
  After you install Filebeat, you need to configure it so that it knows where Elasticsearch and Kibana instances are located.
- Sending OHS Logs to Elasticsearch
  Oracle HTTP Server is based on Apache. Therefore, you can use the built-in Filebeat Apache module to interpret log files for Oracle HTTP Servers.
- Sending the Database Audit Logs to Elasticsearch
  Elasticsearch has a predefined module for sending Oracle Database audit logs to the Elasticsearch server.
- Setting Up and Starting Filebeat
  When setting up Filebeat, ensure that the command succeeds without any error. If any errors are encountered, resolve them before continuing.

# Obtaining the Fingerprint of the Elasticsearch Certificate

To configure the Filebeat module, you need to derive the fingerprint of the Elasticsearch certificate from the local copy you have already created. See Copying the Elasticsearch Certificate.

Obtain the fingerprint of the certificate by using the following command:

```
openssl x509 -noout -fingerprint -sha256 -inform pem -in ~/workdir/ELK/
ca.crt  | sed 's/://g
```

The output appears as follows:

```
SHA256
Fingerprint=361A6E52F1936173795ABE36BB0F3A34185DD5A395BB9CECE0D8437EE16C2E44
```

# Obtaining and Installing Filebeat

You should use the `curl` command to obtain and install Filebeat.

- Use the following `curl` command:

  ```
  curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-
  <ELK_VER>-x86_64.rpm
  sudo rpm -vi filebeat-<ELK_VER>-x86_64.rpm
  ```

  For example:

  ```
  curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/
  filebeat-8.11.0-x86_64.rpm
  sudo rpm -vi filebeat-8.11.0-x86_64.rpm
  ```

# Updating the Filebeat Configuration

After you install Filebeat, you need to configure it so that it knows where Elasticsearch and Kibana instances are located.

1. To configure Filebeat, edit the `/etc/filebeat/filebeat.yml` file.
2. In the Kibana section of the file, add the following configuration information:

   ```
   setup.kibana:
     host: "https://elasticsearch-es-http.elkns.es.local:<ELK_KIBANA_K8>"
     ssl.verification_mode: "none"
   ```

   In the Elasticsearch section, add the following information

   ```
   output.elasticsearch:
     hosts: ["elasticsearch-es-http.elkns.es.local:<ELK_K8>"]
     protocol: "https"
   ```

```
  username: "elastic"
  password: "<ELK_PASSWORD>"
  ssl.ca_trusted_fingerprint:
"361A6E52F1936173795ABE36BB0F3A34185DD5A395BB9CECE0D8437EE16C2E44"
```

> **Note:**
>
> The user specified here is the `elastic` user, used to ensure that you have access to create the Kibana dashboards. You can use a less permissive user such as `logstash_internal` after the setup is complete.

For example:

```
setup.kibana:
  host: "https://elasticsearch-es-http.elkns.es.local:31800”
  ssl.verification_mode: "none"


output.elasticsearch:
  hosts: ["elasticsearch-es-http.elkns.es.local:31920"]
  protocol: "https"
  username: "elastic"
  password: "mypassword"
  ssl.ca_trusted_fingerprint:
"361A6E52F1936173795ABE36BB0F3A33485DD5A395BB9CECE0D8437EE16C2E44"
```

3. Save the file.

# Sending OHS Logs to Elasticsearch

Oracle HTTP Server is based on Apache. Therefore, you can use the built-in Filebeat Apache module to interpret log files for Oracle HTTP Servers.

- Enabling and Configuring the Apache Module

## Enabling and Configuring the Apache Module

The Filebeat Apache module is used to send Oracle HTTP Server logs to the Elasticsearch server. To configure the Apache module:

1. Enable the module by using the following command:

   ```
   sudo filebeat modules enable apache
   ```

2. Edit the Apache configuration file to update the location of the Oracle HTTP Server log files. This file is located at `/etc/filebeat/modules.d/apache.yml`.

   For example, your configuration file may look as follows:

   ```
   # Module: apache
   # Docs: https://www.elastic.co/guide/en/beats/filebeat/8.3/filebeat-module-
   apache.html
   ```

```
- module: apache
  # Access logs
  access:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    var.paths:
     - '/u02/private/oracle/config/domains/ohsDomain/servers/ohs1/logs/
access*'

  # Error logs
  error:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    var.paths:
     - '/u02/private/oracle/config/domains/ohsDomain/servers/ohs1/logs/
error*'
```

3. Save the file.

# Sending the Database Audit Logs to Elasticsearch

Elasticsearch has a predefined module for sending Oracle Database audit logs to the Elasticsearch server.

- Enabling and Configuring the Oracle Module

## Enabling and Configuring the Oracle Module

The Filebeat Oracle module is used to send Oracle Database audit logs to the Elasticsearch server. To configure the Oracle module:

1. Enable the module by using the following command:

   ```
   sudo filebeat modules enable oracle
   ```

2. Edit the Oracle configuration file to update the location of the Oracle Database audit files. This file is located at `/etc/filebeat/modules.d/oracle.yml`.

   For example, your configuration file may look as follows:

   ```
   # Module: oracle
   # Docs: https://www.elastic.co/guide/en/beats/filebeat/8.3/filebeat-module-
   oracle.html

   - module: oracle
     database_audit:
       enabled: true

       # Set which input to use between syslog or file (default).
       #var.input: file

       # Set paths for the log files when file input is used.
   ```

```
# Should only be used together with file input
var.paths: ["/u01/app/oracle/admin/iamdb1*/adump/*.aud"]
```

3. Save the file.

# Setting Up and Starting Filebeat

When setting up Filebeat, ensure that the command succeeds without any error. If any errors are encountered, resolve them before continuing.

1. Use the following command to set up Filebeat:

   ```
   sudo filebeat setup
   ```

   After verifying the configuration and ensuring that the Kibana dashboards are loaded, you can change the user and password to a lower privileged user, if required.

2. Start Filebeat by using the following command:

   ```
   sudo service filebeat start
   ```

   Your log files will now be shipped to Elasticsearch.

27

# Scaling Procedures for an Enterprise Deployment

The scaling procedures for an enterprise deployment include scale out and scale in. During a scale-out operation, you add Managed Servers to new nodes. You can remove these Managed Servers by performing a scale in operation.

This chapter includes the following topics:

- Scaling Out the Topology
  When you scale out the topology, you add new Managed Servers to new nodes.
- Scaling In the Topology
  When you scale in the topology, you remove new Managed Servers or instances or both from your running system.

## Scaling Out the Topology

When you scale out the topology, you add new Managed Servers to new nodes.

This section describes the procedures to scale out the Identity Management topology.

- Scaling Out Oracle Unified Directory
- Scaling Out a WebLogic Domain

### Scaling Out Oracle Unified Directory

This section lists the prerequisites for scaling out Oracle Unified Directory, explains the procedure, and describes the steps to verify the scale-out process.

- Prerequisites for Scaling Out
- Scaling Out by Adding a New Replicated Instance
- Verifying the Scale Out

#### Prerequisites for Scaling Out

Before you perform a scale out of the OUD topology, ensure that you meet the following requirements:

- As the starting point, you have at least one OUD server instance running. This is the primary instance.
- A Kubernetes worker node is available with sufficient capacity to host the new pod.

#### Scaling Out by Adding a New Replicated Instance

You can use one of the following ways to scale out the domain:

1. Modifying the `override_oud.yaml` file.

Creating a new OUD instance involves modifying the server overrides file. See Creating a Server Overrides File.

If you do not have a **replOUD** section, then you need to add this section to the file as described in Creating a Server Overrides File.

Increasing the number of replicas involves increasing the value of the `replicaCount` parameter. Set this value to the total number of replicas you require. This number does not include your primary instance. Therefore, if you require a total of four OUD instances, then set the value to 3, where the fourth instance is the primary instance.

2. Using Helm to increase the number of running instances.

   After you update the `override_oud.yaml` file, use the following commands:

   ```
   cd WORKDIR/fmw-kubernetes/OracleUnifiedDirectory/kubernetes/helm
   helm upgrade --namespace <NAMESPACE> --values WORKDIR/override_oud.yaml
   <OUD_PREFIX> oud-ds-rs
   ```

   For example:

   ```
   cd /workdir/OUD/fmw-kubernetes/OracleUnifiedDirectory/kubernetes/helm
   helm upgrade --namespace oudns --values /workdir/oud/override_oud.yaml edg
   oud-ds-rs
   ```

   Sample output:

   ```
   Release "edg" has been upgraded. Happy Helming!
   NAME: edg
   LAST DEPLOYED: Thu Apr 8 06:35:30 2021
   NAMESPACE: oudns
   STATUS: deployed
   REVISION: 2
   NOTES:

   Copyright (c) 2020, Oracle and/or its affiliates.

    Licensed under the Universal Permissive License v 1.0 as shown at
    https://oss.oracle.com/licenses/upl
   ```

## Verifying the Scale Out

After scaling out and starting the server, proceed with the following verifications:

1. Check the Kubernetes cluster to see that the required number of servers are running, by using the command:

   ```
   kubectl -n <namespace> get all -o wide
   ```

   For example:

   ```
   kubectl -n oudns get all -o wide
   ```

2. Verify the log files to ensure that the new instance is created correctly.

```
kubectl logs -n <OUDNS> pod/<OUD_PREFIX>-oud-ds-rs-2
```

For example:

```
kubectl logs -n oudns pod/edg-oud-ds-rs-2
```

# Scaling Out a WebLogic Domain

This section describes the procedures to scale out a WebLogic domain such as Oracle Access Manager.



## Prerequisites for Scaling Out

Before you perform a scale out of the topology, you must ensure that you meet the following requirements:

- The starting point is a cluster with Managed Servers already running.
- A Kubernetes worker node is available with sufficient capacity to host the new pod.
- It is assumed that the cluster syntax is used for all internal RMI invocations, JMS adapter, and so on.
- You are currently not running the maximum number of servers you defined when at the time of creating the domain.

## Scaling Out a Domain

Scaling the domain requires modifying the Replicas parameter in it. The simplest way to modify the parameter is to patch the domain using the following command:

```
kubectl patch cluster -n <NAMESPACE> <DOMAIN_NAME>-<CLUSTER_NAME> --
type=merge -p '{"spec":{"replicas":<NO OF REPLICAS>}}'
```

For example:

```
kubectl patch cluster -n oamns accesdomain-oam-cluster --type=merge -p
'{"spec":{"replicas":3}}'
```

## Scaling Out the Cluster Using the Sample Script

Oracle provides a number of scripts to make domain life cycle operations simple. These scripts are included in the sample files you download from GitHub and are located in:

```
fmw-kubernetes/<PRODUCT>/kubernetes/domain-lifecycle
```

You can scale out the cluster using the following command:

```
./scaleCluster.sh -d <DOMAIN_NAME> -n <NAMESPACE> -c <CLUSTER_NAME> -r
<REPLICAS>
```

## Verifying the Scale Out

After scaling out and starting the server, proceed with the following verifications:

1. Check the Kubernetes cluster to see that the required number of servers are running, by using the command:

   ```
   kubectl -n <namespace> get all -o wide
   ```

   For example:

   ```
   kubectl -n oamns get all -o wide
   ```

   Or

   ```
   kubectl -n oigns get all -o wide
   ```

2. Verify the correct routing to web applications.

   For example:

   a. Access the application on the load balancer:

      ```
      https://igdinternal.example.com:7777/soa-infra
      ```

   b. Check that there is activity in the new server also:

      Go to **Cluster** > **Deployments** > **soa-infra** > **Monitoring** > **Workload**.

   c. You can also verify that the web sessions are created in the new server:

      - Go to **Cluster** > **Deployments**.
      - Expand **soa-infra**, click **soa-infra** Web application.
      - Go to **Monitoring** to check the web sessions in each server.

      You can use the sample URLs and the corresponding web applications that are identified in the following table, to check if the sessions are created in the new server for the cluster that you are scaling out:

3. Verify that JMS messages are being produced and consumed to the destinations, and produced and consumed from the destinations, in the three servers.

   a. Go to **JMS Servers**.

   b. Click **JMS Server** > **Monitoring**.

4. Verify the service migration, as described in Validating Automatic Service Migration in Static Clusters.

# Scaling In the Topology

When you scale in the topology, you remove new Managed Servers or instances or both from your running system.



## Scaling In Oracle Unified Directory

This section lists the prerequisites for scaling in Oracle Unified Directory, explains the procedure, and describes the steps to verify the scale-in process.



### Prerequisites for Scaling In

Before you perform a scale in of the OUD topology, ensure that you have at least two OUD server instance running.

### Scaling In by Removing a Replicated Instance

You can use one of the following ways to scale out the domain:

1. Modifying the `override_oud.yaml` file.

   Removing an OUD instance involves modifying the server overrides file . See Creating a Server Overrides File.

   Decreasing the number of replicas involves reducing the value of the `replicaCount` parameter. Set this value to the total number of replicas you require. This number does not include your primary instance. Therefore, if you require a total of four OUD instances, then set the value to 3, where the fourth instance is the primary instance.

2. Using Helm to reduce the number of running instances.

   After you update the `override_oud.yaml` file, use the following commands:

   ```
   cd WORKDIR/fmw-kubernetes/OracleUnifiedDirectory/kubernetes/helm
   helm upgrade --namespace <NAMESPACE> --values WORKDIR/override_oud.yaml
   <OUD_PREFIX> oud-ds-rs
   ```

   For example:

   ```
   cd /workdir/OUD/fmw-kubernetes/OracleUnifiedDirectory/kubernetes/helm
   helm upgrade --namespace oudns --values /workdir/oud/override_oud.yaml edg
   oud-ds-rs
   ```

Sample output:

```
Release "edg" has been upgraded. Happy Helming!
NAME: edg
LAST DEPLOYED: Thu Apr 8 06:35:30 2021
NAMESPACE: oudns
STATUS: deployed
REVISION: 2
NOTES:

Copyright (c) 2020, Oracle and/or its affiliates.

 Licensed under the Universal Permissive License v 1.0 as shown at
 https://oss.oracle.com/licenses/upl
```

## Verifying the Scale In

After scaling in and starting the server, proceed with the following verifications:

1. Check the Kubernetes cluster to see that the required number of servers are running, by using the command:

   ```
   kubectl -n <namespace> get all -o wide
   ```

   For example:

   ```
   kubectl -n oudns get all -o wide
   ```

2. Verify the log files to ensure that the new instance is created correctly.

   ```
   kubectl logs -n <OUDNS> pod/<OUD_PREFIX>-oud-ds-rs-2
   ```

   For example:

   ```
   kubectl logs -n oudns pod/edg-oud-ds-rs-2
   ```

# Scaling In a WebLogic Domain

This section lists the prerequisites for scaling in a WebLogic domain such as Oracle Access Manager and Oracle Identity Governance, and explains the procedure to scale in the domain.



## Prerequisites for Scaling In

Before you perform a scale In of the topology, you must ensure that you meet the following requirements:

- The starting point is a cluster with Managed Servers already running.

- It is assumed that the cluster syntax is used for all internal RMI invocations, JMS adapter, and so on.

## Scaling In a Domain

Scaling in the domain involves asking the WebLogic Operator for Kubernetes to stop extra pods (Managed Servers). You can use one of the following ways to scale in the domain:

**Modifying the `domain.yaml`/`domain_oim_soa.yaml` file**

1. Locate the entry for the cluster you want to scale in. Decrease the value of the parameter replicas to the desired number of servers you want to start. For example: `replicas: 1`
2. Save the file and apply the changes using the following command:

   ```
   kubectl apply -f domain.yaml
   ```

**Modifying the domain directly**

1. Use the following command:

   ```
   kubectl edit domain <domain_name> -n <namespace>
   ```

   For example:

   ```
   kubectl edit domain accessdomain -n oamns
   ```

2. Locate the entry for the cluster you want to scale in. Decrease the value of the parameter replicas to the desired number of servers you want to start. For example: `replicas: 1`
3. Save the file. The operator will automatically start the required number of servers.

## Scaling In the Cluster Using the Sample Script

Oracle provides a number of scripts to make domain life cycle operations simple. These scripts are included in the sample files you download from GitHub and are located in:

```
fmw-kubernetes/<PRODUCT>/kubernetes/domain-lifecycle
```

You can scale in the cluster using the following command:

```
./scaleCluster.sh -d <DOMAIN_NAME> -n <NAMESPACE> -c <CLUSTER_NAME> -r
<REPLICAS>
```

## Verifying the Scale In

After scaling in and starting the server, proceed with the following verifications:

1. Check the Kubernetes cluster to see that the required number of servers are running, by using the command:

   ```
   kubectl -n <namespace> get all -o wide
   ```

For example:

```
kubectl -n oamns get all -o wide
```

Or

```
kubectl -n oigns get all -o wide
```

2. Verify the log files to ensure that the new instance is created correctly.

```
kubectl logs -n <NAMESPACE> pod/<DOMAIN_NAMR>-<SERVER_NAME>
```

For example:

```
kubectl logs -n oamns pod/accessdomain-oam-server1
```

Or

```
kubectl logs -n oigns pod/governancedomain-oim-server1
```

28

# Configuring Single Sign-On for an Enterprise Deployment

You need to configure the Oracle HTTP Server WebGate in order to enable single sign-on with Oracle Access Manager.

This chapter includes the following topics:

- About Oracle WebGate
  Oracle WebGate is a web server plug-in that intercepts HTTP requests and forwards them to an existing Oracle Access Manager instance for authentication and authorization.
- General Prerequisites for Configuring Oracle HTTP Server WebGate
  Before you can configure Oracle HTTP Server WebGate, you must have installed and configured a certified version of Oracle Access Manager.
- Configuring Oracle HTTP Server 12c WebGate for an Enterprise Deployment
  You need to perform the following steps in order to configure Oracle HTTP Server 12*c* WebGate for Oracle Access Manager on both WEBHOST1 and WEBHOST2.
- Enabling OAM Rest OAP Calls and Health Check
  In Oracle Access Manager 12c, WebGate interacts with Oracle Access Manager through REST API calls. In order for WebGate to have unrestricted access to these Rest APIs, you need to update the *`WEB_CONFIG_DIR`*`/webgate.conf` file.
- Adding a Load Balancer Certificate to WebGate
  Oracle WebGate 12c uses REST calls to interact with Oracle Access Manager 12c. To ensure that the communication works properly, you have to copy the load balancer certificates to WebGate Config and ensure that the REST endpoints are set correctly.
- Copying WebGates Artifacts to Web Tier
  When you created your Oracle Access Management installation, a WebGate called `Webgate_IDM` was created. In order for WebGate to communicate with the Access servers, you must copy the artifacts associated with this WebGate to the web tier.
- Restarting the Oracle HTTP Server Instance
- Setting Up the WebLogic Server Authentication Providers
  To set up the WebLogic Server authentication providers, back up the configuration files, set up the Oracle Access Manager Identity Assertion Provider and set the order of providers.
- Configuring Oracle ADF and OPSS Security with Oracle Access Manager
  Some Oracle Fusion Middleware management consoles use Oracle Application Development Framework (Oracle ADF) security, which can integrate with Oracle Access Manager Single Sign-on (SSO). These applications can take advantage of Oracle Platform Security Services (OPSS) SSO for user authentication, but you must first configure the domain-level `jps-config.xml` file to enable these capabilities.
- Backing Up the Configuration Files
  You should first back up the relevant configuration files to ensure their safety.

# About Oracle WebGate

Oracle WebGate is a web server plug-in that intercepts HTTP requests and forwards them to an existing Oracle Access Manager instance for authentication and authorization.

For Oracle Fusion Middleware 12*c*, the Oracle WebGate software is installed as part of the Oracle HTTP Server 12*c* software installation. See Registering and Managing OAM 11g Agents in *Adminstrator's Guide for Oracle Access Management*.

# General Prerequisites for Configuring Oracle HTTP Server WebGate

Before you can configure Oracle HTTP Server WebGate, you must have installed and configured a certified version of Oracle Access Manager.

For the most up-to-date information, see the certification document for your release on the *Oracle Fusion Middleware Supported System Configurations* page.

For WebGate certification matrix, click and open http://www.oracle.com/technetwork/middleware/id-mgmt/downloads/oam-webgates-2147084.html, then click the *Certification Matrix for 12c Access Management WebGates* link to download the certification matrix spreadsheet.

> **Note:**
>
> For production environments, it is highly recommended that you install Oracle Access Manager in its own environment and not on the machines that are hosting the enterprise deployment.

For more information about Oracle Access Manager, see the latest Oracle Identity and Access Management documentation, which you can find in the **Middleware** documentation on the Oracle Help Center.

# Configuring Oracle HTTP Server 12*c* WebGate for an Enterprise Deployment

You need to perform the following steps in order to configure Oracle HTTP Server 12*c* WebGate for Oracle Access Manager on both WEBHOST1 and WEBHOST2.

In the following procedure, replace the directory variables, such as *WEB_ORACLE_HOME* and *WEB_CONFIG_DIR*, with the values, as defined in File System and Directory Variables Used in This Guide.

1. Perform a complete backup of the web tier domain.
2. Change directory to the following location in the Oracle HTTP Server Oracle home:

   ```
   cd WEB_ORACLE_HOME/webgate/ohs/tools/deployWebGate/
   ```

3. Run the following command to create the WebGate Instance directory and enable WebGate logging on OHS Instance:

```
./deployWebGateInstance.sh -w WEB_CONFIG_DIR -oh WEB_ORACLE_HOME
```

For example:

```
./deployWebGateInstance.sh -w /u02/private/oracle/config/domains/ohsDomain/
config/fmwconfig/components/OHS/ohs1 -oh /u01/oracle/products/web
```

4. Verify that a `webgate` directory and subdirectories was created by the `deployWebGateInstance` command:

```
ls -lat WEB_CONFIG_DIR/webgate/
```

```
total 16
drwxr-x---+ 8 orcl oinstall 20 Oct  2 07:14 ..
drwxr-xr-x+ 4 orcl oinstall  4 Oct  2 07:14 .
drwxr-xr-x+ 3 orcl oinstall  3 Oct  2 07:14 tools
drwxr-xr-x+ 3 orcl oinstall  4 Oct  2 07:14 config
```

5. Run the following command to ensure that the `LD_LIBRARY_PATH` environment variable contains *`WEB_ORACLE_HOME`*`/lib` directory path:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:WEB_ORACLE_HOME/lib
```

6. Change directory to the following directory

*`WEB_ORACLE_HOME`*`/webgate/ohs/tools/setup/InstallTools`

7. Run the following command from the `InstallTools` directory.

```
./EditHttpConf -w WEB_CONFIG_DIR -oh WEB_ORACLE_HOME -o output_file_name
```

For example:

```
./EditHttpConf -w /u02/private/oracle/config/domains/ohsDomain/config/
fmwconfig/components/OHS/ohs1 -oh /u01/oracle/products/web
```

> **Note:**
>
> The `-oh` *`WEB_ORACLE_HOME`* and `-o` *`output_file_name`* parameters are optional.

This command:

- Copies the `apache_webgate.template` file from the Oracle HTTP Server Oracle home to a new `webgate.conf` file in the Oracle HTTP Server configuration directory.
- Updates the `httpd.conf` file to add one line, so it includes the `webgate.conf`.

- Generates a WebGate configuration file. The default name of the file is `webgate.conf`, but you can use a custom name by using the `-o output_file_name` argument to the command.

# Enabling OAM Rest OAP Calls and Health Check

In Oracle Access Manager 12c, WebGate interacts with Oracle Access Manager through REST API calls. In order for WebGate to have unrestricted access to these Rest APIs, you need to update the *WEB_CONFIG_DIR*`/webgate.conf` file.

For example:

```
/u02/private/oracle/config/domains/ohsDomain/config/fmwconfig/components/OHS/
ohs1/webgate.conf
```

To update *WEB_CONFIG_DIR*`/webgate.conf`:

1. Add the following lines:

```
<LocationMatch "/iam/access/binding/api/v10/oap">
    require all granted
</LocationMatch>
```

2. To enable a health check, add the following lines:

> **Note:**
>
> This step is applicable if you have created a health check described in Creating a Health Check.

```
<LocationMatch "/health-check.html">
    require host webtier.com
    request ip IP ADDRESS OF LOADBALANCER
</LocationMatch>
```

> **Note:**
>
> In above mentioned example webtier.com, a require host is used to restrict access to a particular subnet where your load balancer resides. Alternatively, you can use the clause request IP address of the load balancer to specify the address of the load balancer. This is the IP address of the load balancer and not the floating IP address assigned to the load balancer. This ensures that the OHS returns only health check tests to the load balancer.
>
> If your load balancer indicates that the webserver is down, then you must check the OHS access log to confirm the cause of the rejected requests which can occur if you are not restricting it to the right IP address of domain.
>
> You must not set your request host to your internet domain to avoid exposing your health check outside of the organisation.

3. Save the file.

# Adding a Load Balancer Certificate to WebGate

Oracle WebGate 12c uses REST calls to interact with Oracle Access Manager 12c. To ensure that the communication works properly, you have to copy the load balancer certificates to WebGate Config and ensure that the REST endpoints are set correctly.



## Copying the LoadBalancer Certificates to WebGate Config

WebGate needs to trust your load balancer certificate. To ensure this trust, you should add the load balancer's certificate Authoriity chain to the `cacert.pem` file, which is located in *`WEB_CONFIG_DIR`*`/webgate/config`.

You can obtain the certificate from the load balancer using a browser, such as Firefox. However, the easiest way to obtain the certificate is to use the `openssl` command. The syntax of the command is as follows:

```
openssl s_client -connect <LOADBALANCER>:<PORT> -showcerts </dev/null 2>/dev/
null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > LOADBALANCER.pem
```

For example:

```
openssl s_client -connect login.example.com:443 -showcerts </dev/null 2>/dev/
null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
login.example.com.pem
```

This command saves the certificate to a file named `login.example.com.pem`.

If you do not have load balancer certificates in your WebGate truststore, copy the `login.example.com.pem` file to *`WEB_CONFIG_DIR`*`/webgate/config renaming it to cacert.pem`.

For example:

```
cp login.example.com.pem WEB_CONFIG_DIR/webgate/config/cacert.pem
```

If you already have trusted certificates in WebGate, append the certificate to the `cacert.pem` file.

For example:

```
cp login.example.com.pem >> WEB_CONFIG_DIR/webgate/config/cacert.pem
```

## Ensuring that the REST Endpoints are Set Correctly

To ensure that the REST endpoints are set correctly:

1. Log in to the OAM Console using your oam administrator account.

2. Click **Agents**.
3. Click **Search**.
4. Locate and click the name of the WebgGate agent from the search results to bring up the edit screen.
5. Expand the User Properties screen and check that the following parameters are defined correctly:
   - **OAMRestEndPointHostName** = `login.example.com`
   - **OAMRestEndPointPort** = `443`
   - **OAMServerCommunicationMode** = `HTTPS`

   These values should reflect the login point of your Oracle Access Manager installation. If unsure about these values, contact your OAM administrator.

# Copying WebGates Artifacts to Web Tier

When you created your Oracle Access Management installation, a WebGate called `Webgate_IDM` was created. In order for WebGate to communicate with the Access servers, you must copy the artifacts associated with this WebGate to the web tier.



## Copying Generated Artifacts to the Oracle HTTP Server WebGate Instance Location

The location of the files in the Oracle HTTP Server configuration directory depends on the Oracle Access Manager security mode setting (OPEN, SIMPLE, or CERT).

The following table lists the required location of each generated artifact in the Oracle HTTP Server configuration directory, based on the security mode setting for Oracle Access Manager. In some cases, you might have to create the directories if they do not exist already. For example, the wallet directory might not exist in the configuration directory.

> **Note:**
>
> For an enterprise deployment, Oracle recommends simple mode, unless additional requirements exist to implement custom security certificates for the encryption of authentication and authorization traffic. The information about using open or certification mode is provided here as a convenience.
>
> Avoid using open mode, because in open mode, traffic to and from the Oracle Access Manager Server is not encrypted.
>
> For more information about using certificate mode or about Oracle Access Manager supported security modes, see Securing Communication Between OAM Servers and WebGates in *Administrator's Guide for Oracle Access Management*.

**Table 28-1 Web Tier Host Location to Copy the Generated Artifacts**

| File | Location When Using CERT Mode |
|---|---|
| wallet/cwallet.sso[1] | *WEB_CONFIG_DIR*/webgate/config/wallet/ |
| ObAccessClient.xml | *WEB_CONFIG_DIR*/webgate/config/ |
| password.xml | *WEB_CONFIG_DIR*/webgate/config/ |
| aaa_key.pem | *WEB_CONFIG_DIR*/webgate/config/ |
| aaa_cert.pem | *WEB_CONFIG_DIR*/webgate/config/ |

[1] Copy cwallet.sso from the wallet folder and not from the output folder. Even though there are 2 files with the same name they are different. The one in the wallet sub directory is the correct one.

> **Note:**
>
> If you need to redeploy the ObAccessClient.xml to WEBHOST1 and WEBHOST2, delete the cached copy of ObAccessClient.xml and its lock file, ObAccessClient.xml.lck from the servers. The cache location on WEBHOST1 is:
>
> *WEB_DOMAIN_HOME*/servers/ohs1/cache/
>
> And you must perform the similar step for the second Oracle HTTP Server instance on WEBHOST2:
>
> *WEB_DOMAIN_HOME*/servers/ohs2/cache/

### Obtaining WebGate Artifacts

The easiest way to obtain the WebGate artifacts is to download them from the OAM console. To download, complete the following steps:

1. Log in to the OAM console with the user name oamadmin by using the following URL:

   http://iadadmin.example.com/oamconsole

2. Click **Agents**.
3. On the Search screen, click **Search**.
4. From the list of agents, select **Webgate_IDM**.
5. Download the artifacts using the download button. A zip file gets downloaded on the host machine you are using.

Copy the downloaded zip file to the Oracle HTTP Server machine and unzip it to the *WEB_CONFIG_DIR*/webgate/config location. The files get extracted to the correct locations.

## Restarting the Oracle HTTP Server Instance

For information about restarting the Oracle HTTP Server instance, see Restarting Oracle HTTP Server Instances by Using WLST in *Administering Oracle HTTP Server*.

If you have configured Oracle HTTP Server in a WebLogic Server domain, you can also use Oracle Fusion Middleware Control to restart the Oracle HTTP Server instances. See Restarting Oracle HTTP Server Instances by Using Fusion Middleware Control in *Administering Oracle HTTP Server*.

# Setting Up the WebLogic Server Authentication Providers

To set up the WebLogic Server authentication providers, back up the configuration files, set up the Oracle Access Manager Identity Assertion Provider and set the order of providers.

The following topics assumes that you have already configured the LDAP authenticator by following the steps in #unique_963. If you have not already created the LDAP authenticator, then do so before you continue with this section.

- Backing Up the Configuration Files
  You should first back up the relevant configuration files to ensure their safety.
- Setting Up the Oracle Access Manager Identity Assertion Provider
- Updating the Default Authenticator and Setting the Order of Providers

## Backing Up the Configuration Files

You should first back up the relevant configuration files to ensure their safety.

Back up the following configuration files:

```
DOMAIN_HOME/config/config.xml
DOMAIN_HOME/config/fmwconfig/jps-config.xml
DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml
```

Also back up the `boot.properties` file for the Administration Server:

```
DOMAIN_HOME/servers/AdminServer/security/boot.properties
```

## Setting Up the Oracle Access Manager Identity Assertion Provider

Set up an Oracle Access Manager identity assertion provider in the Oracle WebLogic Server Administration Console.

To set up the Oracle Access Manager identity assertion provider:

1. Log in to the WebLogic Server Administration Console, if not already logged in.
2. Click **Lock & Edit**.
3. Click **Security Realms** in the left navigation bar.
4. Click the **myrealm** default realm entry.
5. Click the **Providers** tab.
6. Click **New**, and select the asserter type **OAMIdentityAsserter** from the drop-down menu.
7. Name the asserter (for example, *OAM ID Asserter*) and click **OK**.
8. Click the newly added asserter to see the configuration screen for the Oracle Access Manager identity assertion provider.
9. Set the control flag to *REQUIRED*.

10. Under Chosen types, select both the **ObSSOCookie** and **OAM_REMOTE_USER** options, if they are not selected by default.
11. Click **Save** to save the settings.
12. Click **Activate Changes** to propagate the changes.

## Updating the Default Authenticator and Setting the Order of Providers

Set the order of identity assertion and authentication providers in the WebLogic Server Administration Console.

To update the default authenticator and set the order of the providers:

1. Log in to the WebLogic Server Administration Console, if not already logged in.
2. Click **Lock & Edit**.
3. From the left navigation, select **Security Realms**.
4. Click the **myrealm** default realm entry.
5. Click the **Providers** tab.
6. From the table of providers, click the **DefaultAuthenticator**.
7. Set the Control Flag to `SUFFICIENT`.
8. Click **Save** to save the settings.
9. From the navigation breadcrumbs, click **Providers** to return to the list of providers.
10. Click **Reorder**.
11. Sort the providers to ensure that the OAM Identity Assertion provider is first and the DefaultAuthenticator provider is last.
12. Click **OK**.
13. Click **Activate Changes** to propagate the changes.
14. Shut down the Administration Server, Managed Servers, and any system components, as applicable.
15. Restart the Administration Server.
16. If you are going to configure ADF consoles with SSO, you can keep the Managed Servers down and restart them later. If not, restart Managed Servers.

# Configuring Oracle ADF and OPSS Security with Oracle Access Manager

Some Oracle Fusion Middleware management consoles use Oracle Application Development Framework (Oracle ADF) security, which can integrate with Oracle Access Manager Single Sign-on (SSO). These applications can take advantage of Oracle Platform Security Services (OPSS) SSO for user authentication, but you must first configure the domain-level `jps-config.xml` file to enable these capabilities.

The domain-level `jps-config.xml` file is located in the following location after you create an Oracle Fusion Middleware domain:

```
/u01/oracle/user_projects/domain/<OAM_DOMAIN_NAME>/config/fmwconfig/
jps-config.xml
```

> **Note:**
>
> The domain-level `jps-config.xml` should not be confused with the `jps-config.xml` that is deployed with custom applications.

To update the OPSS configuration to delegate SSO actions in Oracle Access Manager, complete the following steps:

1. Connect to the AdminServer container using the following command:

   ```
   kubectl exec -n oamns -ti accessdomain-adminserver -- /bin/bash
   ```

2. Start the WebLogic Server Scripting Tool (WLST):

   *ORACLE_COMMON_HOME*/common/bin/wlst.sh

   For example:

   /u01/oracle/oracle_common/common/bin/wlst.sh

3. Connect to the Administration Server, by using the following WLST command:

   ```
   connect('<OAM_WEBLOGIC_USER>','<OAM_WEBLOGIC_PWD>','t3://<OAM_DOMAIN_NAME>-
   adminserver.<OAMNS>.svc.cluster.local:<OAM_ADMIN_PORT>')
   ```

   For example:

   ```
   connect('weblogic','<password>','t3://accessdomain-
   adminserver.oamns.svc.cluster.local:7001')
   ```

4. Run the `addOAMSSOProvider` command, as shown:

   ```
   addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication",
   logouturi="/oam/logout.html")
   ```

5. Disconnect from the Administration Server by entering the following command:

   ```
   exit()
   ```

6. Exit from the container using the `exit` command.

# Backing Up the Configuration Files

You should first back up the relevant configuration files to ensure their safety.

Back up the following configuration files:

*DOMAIN_HOME*/config/config.xml
*DOMAIN_HOME*/config/fmwconfig/jps-config.xml
*DOMAIN_HOME*/config/fmwconfig/system-jazn-data.xml

Also back up the `boot.properties` file for the Administration Server:

*DOMAIN_HOME*/servers/AdminServer/security/boot.properties

29

# Sanity Checks

The sanity tests described in this chapter are over and above the normal tests detailed in the guide. They are designed to test the in-depth functionality of Oracle Access Management (OAM) and Oracle Identity Manager (OIM).
This chapter includes the following topics:

- Sanity Checks for Oracle Access Management
  Learn about the sanity checks applicable for Oracle Access Management (OAM).
- Sanity Checks for Oracle Identity Governance
  Learn about the sanity checks applicable for Oracle Identity Governance (OIG).
- Sanity Checks for Oracle Advanced Authentication
  Learn about the sanity checks applicable to Oracle Advanced Authentication (OAA).

## Sanity Checks for Oracle Access Management

Learn about the sanity checks applicable for Oracle Access Management (OAM).

This section explains the following sanity checks:

- Verifying LDAP Authentication for OAM Agent Protected Application for Valid User
- Verifying LDAP Authentication Failure for OAM Agent Protected Application for Invalid Password
- Verifying LDAP Authentication Failure for OAM Agent Protected Application for Invalid Username
- Verifying Access of OAM Agent Protected Unavailable Resource
- Verifying Access of Resource that was Recently Deleted or Replaced from the Policy

### Verifying LDAP Authentication for OAM Agent Protected Application for Valid User

To verify the LDAP authentication for OAM agent protected application for valid user, do the following:

1. Access an application protected by an OAM WebGate which is configured to OAM server.
2. Check out the URL that is being redirected to for authentication is from OAM server.
3. Provide a valid username and password from the OUD authentication form and click Login.
4. Check the cookies that are created in the browser.

Expected Result:

- OAM agent protected Application can be accessed on providing valid credentials.
- ObSSOcookie and OAM_ID cookies are created in the browser session.

## Verifying LDAP Authentication Failure for OAM Agent Protected Application for Invalid Password

To verify the LDAP authentication failure for OAM agent protected application for invalid password, do the following:

1. Access an application protected by an OAM WebGate which is configured to OAM server.
2. Check out the URL that is being redirected to for authentication is from OAM server.
3. Provide a valid username and an invalid password in the authentication form.

Expected Result:

- User authentication fails.
- Appropriate error message is displayed.
- Resource cannot be accessed by the user.

## Verifying LDAP Authentication Failure for OAM Agent Protected Application for Invalid Username

To verify the LDAP authentication failure for OAM agent protected application for invalid username, do the following:

1. Access an application protected by an OAM WebGate which is configured to OAM server.
2. Check out the URL that is being redirected to for authentication is from OAM server.
3. Provide an invalid username and any password in the authentication form.

Expected Result:

- User authentication fails.
- Appropriate error message is displayed.
- Resource cannot be accessed by the user.

## Verifying Access of OAM Agent Protected Unavailable Resource

If you access an OAM agent protected unavailable resource, an appropriate error message is displayed though the credentials provided are valid. To verify this, do the following:

1. Access a resource url protected by an OAM WebGate which is configured to OAM server when that resources is not available.
2. Check out the URL that is being redirected to for authentication is from OAM server.
3. Provide a valid username and password in the authentication form.
4. Check the cookies that are created in the browser.

Expected Result:

OAM WebGate protected application cannot be accessed and a proper error message should be displayed.

## Verifying Access of Resource that was Recently Deleted or Replaced from the Policy

If you access a resource which was recently deleted or replaced from the policy, the authentication is not required and the access is granted. To verify this, do the following:

1. Remove a resource and replace it with new one in the `policy.xml` or UI.
2. Access the application or resource that you deleted or replaced in the previous step. This application must be protected by an OAM WebGate which is configured to OAM server.
3. Check if the user is not asked for authentication without having to restart the OAM 11*g* Server or WebLogic Server.
4. Check if user is able to access the resource.

Expected Result:

Resource or Application can be accessed without having to authenticate user and without having to restart the OAM 11*g* Server or WebLogic Server.

# Sanity Checks for Oracle Identity Governance

Learn about the sanity checks applicable for Oracle Identity Governance (OIG).

This section explains the following sanity checks:

- Creating Organization
- Creating a User Name
- Creating Role
- Managing Sandboxes
- Publishing a Sandbox
- Adding User Defined Field (UDF) for a User
- Creating a Disconnected Application and Provision
- Importing and Configuring DB User Management
- Creating an Access Policy and Provision
- Creating End User Request for Accounts, Entitlements, and Roles
- Resetting Account Password
- Creating a Certification and Approving
- Creating Identity Audit Scan Definitions and Viewing its Results
- Testing Identity Audit

## Creating Organization

To create an organization, do the following:

1. Log in to the Identity Console as `xelsysadm` using the following URL:

   `https://`prov.example.com`/identity`
2. Click **Manage**, and then click **Organization**.

3. Click **Create**, and specify the org name as TestOrg.
4. After you have entered the details of your organization, click **Save** to store the changes.

## Creating a User Name

To create a user, do the following:

1. Log in to the Identity Console as `xelsysadm` using the following URL:

   `https://prov.example.com/identity`
2. Click **Manage**, and then click **User**.
3. Click **Create**, and specify the user name as Rahul Dravid.
4. Select Org as **TestOrg**.
5. Set and confirm user password.
6. Log in as Rahul Dravid.
7. Set the challenge questions and answers.
8. Log in to the Identity Console and verify the user name.

## Creating Role

To create a role, do the following:

1. Log in to the Identity Console as `xelsysadm` using the following URL:

   `https://prov.example.com/identity`
2. Click **Manage**, and then click **Roles and Access Policies > Roles**.
3. Click **Create** and provide the mandatory attributes (Name, Display Name) to create a Role named `Coach`.
4. Click **Next** repeatedly until the Publish Role to Organizations page is displayed.
5. On the **Organizations** page, click **Add Organizations**. Provide the organization name as **TestOrg** and click **Search**.
6. Select the organization **TestOrg** and click **Add Selected**. Click **Select**.
7. Click **Next**, and then click **Finish**.

## Managing Sandboxes

A number of the operations below require the creation of a sandbox. A sandbox is a non-active area where things can be tried out prior to making them live.

### Creating a Sandbox

You can crate sandboxes either from the System Administration Console or the Identity Console. The steps are the same. The following is an example for creating a sandbox in the System Administration Console.

To create a sandbox:

1. Log in to the System Administration Console as `xelsysadm` using the following URL:

   `http://IGDADMIN.example.com/sysadmin`

2. Click **Sandboxes**.
3. Click **Create Sandbox**.
4. Enter the below details in the Create Sandbox window.

**Table 29-1 Properties of the Sandbox Window**

| Attribute | Value |
|---|---|
| Name | TestSandbox |
| Description | Enter a description |

Select **Activate Sandbox**.

5. Click **Save and Close**.

## Publishing a Sandbox

Once the changes are fine, you publish the sandbox to make it live. This is achieved by performing the following steps:

1. Log in to the System Administration Console as `xelsysadm` using the following URL:

   `http://IGDADMIN.example.com/sysadmin`

2. Click **Sandboxes**.
3. A window appears with a list of the sandboxes.
4. Click a sandbox. For example: **Test Sandbox**.
5. Click **Publish Sandbox** to make the changes active.

## Adding User Defined Field (UDF) for a User

To add a User Defined Field (UDF):

1. Log in to the System Administration Console as `xelsysadmin` using the following URL:

   `http://IGDADMIN.example.com/sysadmin`

2. Create & Activate Sandbox.
3. Click **User** from under **System Entities**.
4. Click **Create** under **Action**.
5. Select **Text** and click **OK**.
6. Enter **Display Label** and **Name**, select **Searchable**, and click on **Save and Close**. You have now created a user defined field (UDF) with the name you specified.
7. Publish Sandbox.
8. Log in to the Identity Console as `xelsysadm` using the following URL:

   `http://prov.example.com/identity`

9. Create and Activate Sandbox.
10. Click on **Manage** to show the management menu.
11. Open Users page, and click **Create**.
12. Click **Customize** at the top right of the screen.

13. Enter the details for all the attributes listed below.

**Table 29-2 User Defined Field Properties**

| Attribute | Description |
|---|---|
| First Name | Enter a name for example: John |
| Last Name | Enter a last name for example: Doe |
| Email | Enter an email address for example: john.doe@example.com |
| Organisation | Enter or search for an Organization for example: TestOrg |
| User Type | Select the type of user from the drop down list. |
| User login | Enter the users login name for example: JohnDoe |
| Password | Enter an initial password for the user to use. |

14. Go to the **Structure** tab at the top left of the screen.
15. The user entry screen is displayed. Scroll down until you see the Basic Information section. As you move down the screen, certain areas are highlighted by a box. When the Entire Basic Information section is highlighted, including the title, click it. A dialogue box is displayed confirming that you want to edit the task flow. Click **Edit**. A structure window is displayed on the right.
16. Click **PanelForm Layout**.
17. Click **Add Content**.
18. Select **Data Component - Catalog**, and then click **UserVO**.
19. Find the User Defined Field you created in step 6 and Click **Add**. Select ADF Input Text w/ Label. Your user defined Field will now be shown in the Basic Information section of the User screen.
20. Close the Add Content Selection screen.
21. Click **Close** at the top of the screen to close the editing window.
22. Close the structure form by clicking **Close** on the top right corner of the Identity Console window.
23. Publish the sandbox.
24. Log out and log in again.
25. Open the User Details page.
26. Create a user name populating the user defined field that you created in step 6 and verify if it is displayed properly in the user details page.

# Creating a Disconnected Application and Provision

To create a disconnected application and provision:

1. Create a lookup by completing the following steps:
   a. Log in to the System Administration Console as `xelsysadm` using the following URL:

      `http://igdadmin.example.com/sysadmin`

   b. Go to **System Configuration** tab and click **Lookups**.

c. Click the **Create** link under Action drop down list.

d. Enter the meaning as `Lookup.Disc`, and enter the code as `Lookup.Disc`.

e. Click **Create link** under **Action** drop down list.

f. Enter the value `HDD` for Meaning, and `HDD` for Code.

g. Repeat with the values of `CD` and `CD` for Meaning and Code.

h. Click **Save**.

i. Enter the value `Lookup.Disc` for Meaning, `Lookup.Disc` for code, and click **Search**.

j. The values **HDD** and **CD** are displayed. Click **OK**.

2. Create disconnected application instances by completing the following steps:

a. Log in to the System Administration Console as `xelsysadm` using the following URL:

`http://igadmin.example.com/sysadmin`

b. Click the **Sandboxes** link, and then click **Create Sandbox**.

c. Enter the name **Disc**, and click **Save** and **Close**. Click **OK** to confirm. Sandbox is activated.

d. Go to **Provisioning configuration**, and click **Application Instances**.

e. Click **Create**. The Create App Instance page is displayed by enabling the Attribute tab.

f. Enter the name as **Disc**, Description as **Disc**, and check the **Disconnected** check box. Click **Save**. Click **OK** to confirm. Feedback message is displayed to confirm that Application Instance Disc is created successfully.

g. On the same page, go to the **Attribute** tab. Form field is added with the name **Disc**. Click **Edit** next to Form field.

This step enables the Manage Disc tab with its subtab, **Fields**, opened. Click the **Child Objects** tab which is next to the **Fields** tab.

h. Click **Add**, and enter the name as **chdisc**, description as **chdisc**, and Click **OK**.

i. Click **chdisc**. This opens another page by enabling the **Fields** tab.

j. Click **Create link** under **Action** drop down list and select **Lookup** as the Field type, and click **OK**.

k. Enter Display Label and name as `Disc`, select **Searchable**. Click **Lookup Type**, and then click **Search** or look up icon (Magnifier icon). Enter the meaning as `Lookup.Disc`.

l. Click **Search**. Values **HDD** and **CD** must be displayed. Click **OK**. Lookup must be selected. Default Value Label, One Drop down gets added. Click on that, and you will see the values: HDD and CD.

If you enabled **Entitlement**, make sure that **Searchable** and **Searchable Picklist** are also selected. Keep the remaining ones with the default values.

m. Click **Save and Close**.

n. Click **Back to Parent Object**, and then click **Regenerate view**.

o. Enable **Parent Form + Child Tables (Master/Detail)**, keep the default setting. Click **OK**.

p. Go to the **Application Instance** tab. Search for an Application Instance **Disc**.

q. Click **Refresh**, and click **Apply** on Disc form.

r. Go to **System Configuration** > **Scheduler** from the left navigation window.

s. Enter the value **Ent*** in the **Search Scheduled Jobs** field, click **Search** or **Go** button.

t. The results are displayed. Click on Entitlement List job name.

u. Click **Run now**. A confirmation message is displayed saying the Job is running.

v. Click **Refresh.** Verify that the execution status is successful. Close the window.

w. Go to the Application instance's Entitlement tab. Two entitlements are displayed - HDD, CD. Select either of the the two and click **Assign +** from the window below.

x. Search organization name, by entering the value Top, and click **Search**.

y. Top organization should be displayed. Select that row / organization, and click **Add Selected**. Selected organization gets added successfully.

z. Check **Apply to Entitlement**, and click **Select**. Selected Organization gets added successfully.

aa. Click **Assign**.

ab. Repeat steps x, y, z, and aa for the CD row.

ac. Search for the organization name **TestOrg**, and click **Search**.

ad. **TestOrg** organization is displayed. Select that row / organization, and click **Add Selected**.

ae. Selected organization gets added successfully. Check **Apply to Entitlement** and click **Select**. Selected organization gets added successfully.

af. Go to the Application Instance's Attribute tab. Click **Apply**. A message is displayed stating that the Application instances disc is modified successfully.

ag. Click Sandboxes.

ah. Select the same sandbox **Disc**. Click **Export sandbox** button. Export sandbox generate .zip file `sandbox_disc.zip`. Click **OK** button. Zip file is saved and generated.

ai. After export is successfully completed, click **Publish sandbox** button. Click **Yes** to confirm.

aj. After you publish, the sandbox is listed under Publish Sandboxes link.

3. Provision the disconnected application instances and entitlements to user by completing the following steps:

a. Log in to the Identity Console as `xelsysadm` using the following URL:

`https://prov.example.com/identity`

b. Click **Manage** and then click **Users**.

c. Search for the user name `Rahul Dravid`, and click **Search**.

d. The user Rahul Dravid is displayed. Click on that user link. User details are displayed.

e. Go to **Accounts** tab, and then to the **Request Account** tab. Account access request page is displayed. Select **Enabled Add access**., and go to the **Catalog** tab. All available Application Instances are displayed.

f. Click **Add to cart** of the **Disc** Disconnected application instances, and click **Next**. The cart detail page is displayed

g. Click the Pen icon in the Request detail pane.

h. Enter the account logging name as `Rahul Dravid_123`, and the password as `<password>`. Click **Update**.

i. Click **Submit**. Request will be generated with a message `Request for access completed successfully`.

j. Go to the **Self Service** tab. Click **Provisioning task**, and the go to the **Manual Fulfillment** tab. Manual fulfillment page is displayed.

k. Click on that request. Request details are displayed. Verify the data. Click **Complete**, and then click **Refresh**.

l. Go to the **Manage** tab, and then to the **User** tab. Open the same user Rahul Dravid.

m. Go to the **Account** tab. Click **Refresh**. Verify that the account status is `Provisioned`.

n. Select the same account name Rahul Dravid_123, and click Request Entitlement button. Entitlement Access request page is displayed. Enable **Add Access** and go to the **Catalog** tab.

o. Click **Add to cart** for entitlement `HDD`. Click **Next**.

p. Click **Submit**. Request will be generated with a message "Request for access completed successfully".

q. Go to the **Self service** tab. Click on **Provisioning task**, and go to **Manual Fulfillment** tab. Manual fulfillment page is displayed

r. Click on that request. Request details are displayed. Verify the data. Click **Complete**, and then click **Refresh**.

s. Go to the **Manage** tab, and then to the **User** tab. Open the details of the same user - Rahul Dravid.

t. Go to the **Entitlement** tab. Click **Refresh**. Verify that the Entitlement status is **Provisioned**.

# Importing and Configuring DB User Management

To import and configure database user management:

1. Download the latest Database User Management Connector from the Oracle Identity Manager Connector Downloads page on Oracle Technology Network (OTN):

   http://www.oracle.com/technetwork/middleware/id-mgmt/downloads/connectors-101674.html

2. Log in to the System Administration Console as `xelsysadmin` user using the following URL:

   `http://igdadmin.example.com/sysadmin`

3. Go to the **System Configuration** tab and click **Import**.

4. Select the file `DBUserManagement-Oracle-ConnectorConfig.xml`'. Sample location: `D:\DBUM-12.2.1.4.0\xml`

5. Click **Open**.

6. Click **Next**. You can either provide the ITResource details now or later. To provide the same later, click **Next**.

7. Click **Selected Entities** to view selections, and click **Import**. Once the import is successfully completed, click **OK**.

8. Copy the third party jars of target systems to the *IGD_ORACLE_HOME*`/idm/server/ConnectorDefaultDirectory/targetsystems-lib/DBUM-12.2.1.4.0` directory.

> **Note:**
>
> If the target is Oracle database, no driver jar is needed.

9. To configure a trusted source reconciliation, create and configure a new IT resource. For example, Oracle DB Trusted of type Oracle DBUM.
10. In the Configuration Lookup, update the trusted configuration lookup name as `Lookup.DBUM.Oracle.Configuration.Trusted`. This configures the ITResource for the target system.
11. Either you can create the ITResource and provide the following details or Open the existing ITResource 'Oracle DB' as specified below:

    ITResource Details:

    `Configuration Lookup = Lookup.DBUM.Oracle.Configuration`

    `Connector Server Name =`

    `Connection Properties =` Specify the connection properties for the target system database.

    `Database Name =` This field identifies database type (such as Oracle and MSSQL) and its used for loading respective scripts. Sample value: Oracle

    `JDBC Driver = oracle.jdbc.driver.OracleDriver`

    `JDBC URL =` For Oracle: `jdbc:oracle:thin:@host:port:sid`

    `Login Password =` Enter the password for the user name of the target system account to be used for connector operations.

    `Login User = sys as sysdba`

## Creating an Access Policy and Provision

To create an access policy and provision:

1. Log in to the Identity Console as `xelsysadm` using the following URL: https://prov.example.com/identity.
2. Click **Manage**.
3. Click **Roles and Access Policies** -> **Roles**.
4. Create a Role named `DBUMRole`.
5. Click **Home** tab to select the main management options.
6. Click **Users**.
7. Click **Create**.
8. Create an user named `Jean Wilson`.
9. Click **Home** tab.
10. Click **Roles and Access Policies** -> **Roles**.
11. Select the Role `DBUMRole`.
12. The role page is displayed - Click **Members**.
13. Click **Add**.

14. In the add members dialogue box, search for the user Jean Wilson.
15. Click the user Jean Wilson.
16. Click **Add Selected**.
17. Click **Apply**.
18. Create another user named `Patrick Morgan` and assign the user role `DBUMRole`.
19. Click **Manage** and click **Home**tab.
20. Open the user details page of Jean Wilson and click **Accounts** tab. DBUM Account should be in Provisioned state.
21. Go to the **Entitlements** tab and verify all child data added are displayed.
22. Repeat the previous two steps for user Patrick Morgan.

## Creating End User Request for Accounts, Entitlements, and Roles

To create an end user request for roles, do the following:

1. Create a user `Arthur Hill`.
2. Log in as `Arthur Hill` and open **My Access** page, and then **Roles**.
3. Click **Request** and in catalog, add **DBUMRole** to cart.
4. Submit request.
5. Log in as administrator and open Pending Approvals.
6. Open the request and approve.
7. As `Arthur Hill`,verify that the role is assigned successfully.

To create an end user request for accounts, do the following:

1. Create a user `Bruce Parker`.
2. Log in as `Bruce Parker` and open **My Access** page, and then **Roles**.
3. Click **Request**.
4. From the Catalog, select the **DBUM App** and add to cart.
5. Click **Next** and click **Submit** to submit the request.
6. Log in as administrator and open Inbox.
7. Open the request, verify the details, and approve request.
8. As `Bruce Parker`, verify that the Account is provisioned successfully.

To create an end user request for entitlements, do the following:

1. Log in as `Jean Wilson`.
2. Open the **My Access** page and go to the **Accounts** tab.
3. Select the **DBUM app**, and click **Request Entitlements** under Action.
4. Add any entitlement to cart and submit request.
5. Log in as administrator and open Inbox.
6. Open the request and approve.
7. As Jean Wilson, verify that the entitlement is provisioned successfully.

## Resetting Account Password

To reset the account password:

1. Log in to the Identity Console as `Jean Wilson`.
2. Click **My Access** and go to the **Accounts** tab.
3. Select **SSOTarget** and click **Reset Password** in Action.
4. Provide a new password and submit.
5. Log out and re-login as `xelsysadm`.
6. Click **Manage** and then click **Users**.
7. Search for Jean Wilson and open the user details page.
8. Go to the **Accounts** tab and select **DBUM App**.
9. Click **Resource History** under Action and check if the Password Updated task is triggered and is in Completed status.

## Creating a Certification and Approving

Complete the following prerequisites to create a certification and approve:

1. Log in to Identity Console as `xelsysadm`.
2. Launch the System Administration Console.
3. Go to the **System Configuration** tab and click **Configuration Properties**.
4. Look for the following system properties:

   ```
   Property name = Identity Auditor Feature Set Availability

   Keyword = OIG.IsIdentityAuditorEnabled

   Value = TRUE
   ```

5. Save the setting.
6. Restart the OIM server to see the Compliance tab in Identity Console.

To create a certification and approve:

1. Log in to the Identity Console as `xelsysadm`.
2. Go to **compliance**, **Identity Certification**, and then **Definitions**.
3. Create a user type certification with the following information:
   - General details page: Enter the name = `UserCertification`, Type = `user`; Enter Description and click **Next**.
   - Base Selection page: Selected only **Users from Selected Organization** and Add organization (TestOrg). Added organization is displayed. Select **Users with Any Level of Risk** as Risk Level, and click **Next**.
   - Content selection page: Keep the default values, and click **Next**.
   - Configuration page: Keep the default and click **Next**.
   - Select the reviewer by searching for a user, for example, MSDhoni, and click **Next**.
   - Disable Incremental, and click **Next**.

- Summary page: Click **Create**, and click **Yes** to confirm. Certification is created successfully.

4. Log in to the System Administration Console as `xelsysadm`.
5. Click **Scheduler**.
6. Search for a certification `cert_UserCertification`. Verify that the job is run successfully.
7. Log in to the Identity Console as `xelsysadm`, and log out.
8. Log in to the Identity Console as a reviewer (MSDhoni).
9. Go to **Self service**, and click **Certification**.
10. Open the same certification **UserCertification [ MSDhoni ]**.
11. Certification details are displayed. You will see the user "Rahul Dravid".
12. Select user Rahul Dravid.
13. Verify, Role - Coach, Account - Disc, Entitlement - HDD.
14. Select all rows, and take the Complete action. Sign-off pop up should be displayed.
15. Enter the password (username = MSDhoni ; Password = <*password*>). Click **OK**. Certification is completed successfully. It should now reflect in your Inbox.
16. Log in to the Identity Console as MSDhoni / Xelsysadm.
17. Go to **Complaince**, **Identity Certification**, and then **Dashboard**. Dashboard details are displayed.
18. Select **Completed** from the Show Label. This displays all of the completed certifications.

## Creating Identity Audit Scan Definitions and Viewing its Results

Complete the following prerequisites to create identity audit scan definitions:

1. Log in to the Identity Console as `xelsysadm`.
2. Launch the System Administration Console.
3. Go to the **System Configuration** tab, and click **Configuration Properties**.
4. Look for the following system properties:

   ```
   Property name = Identity Auditor Feature Set Availability

   Keyword = OIG.IsIdentityAuditorEnabled

   Value = TRUE
   ```

5. Save the setting.
6. Restart the OIM server to See the Compliance tab in the Identity Console.

To create a rule:

1. Log in to the Identity Console as `xelsysadm`.
2. Click **Compliance**, and then click **Identity Audit**.
3. Select **Rules**, and click **Create**.
4. Create an identity rule `Identity Rule 1` by the following condition builder:

   ```
   user.Display Name; Equals ; Rahul Dravid
   ```

5. Click **Create**. The rule is created.

To create a policy:

1. Log in to the Identity Console as `xelsysadm`.
2. Click **Compliance** and then click **Identity Audit**.
3. Click **Policies**, and click **Create**.
4. Create a policy `Identity Policy 1` by adding the rule `Identity Rule 1`.
5. Click **Create**.

To create scan definition:

1. Log in to the Identity Console as `xelsysadm` using the following URL:

   `https://prov.example.com/identity`
2. Click **Compliance** and then click **Identity Audit**.
3. Click **Scan definitions**, and then click **Create**.
4. Create a scan definition `Identity Scan 1` by adding the policy `Identity Policy 1`.
5. On the Base selection page, select all users.
6. On the Configuration page, keep the default values.
7. On the Summary page, click **Finish**. Scan definition is added successfully.
8. Run the scan definition by selecting **Identity Scan 1**, and clicking **Run now**. Verify that the scan definition is run successfully.
9. Preview the scan definition result by doing the following:
   a. After you run the scan definition, select the scan definition row or record **Identity Scan 1**.
   b. Click **View Scan**. The scan definition results are displayed.

## Testing Identity Audit

Complete the following steps to enable audit feature in Oracle Identity Manager:

1. Log in to the System Administration Console.
2. Click **System Properties** under **System Configuration**.
3. Search for the property `OIG.IsIdentityAuditorEnabled` and update the property value to `TRUE`.
4. Restart the Oracle Identity Manager Managed Server for the change to take effect.
5. Log in to the Identity Console as `xelsysadm` using the following URL:

   `https://prov.example.com/identity`
6. Click **Compliance** and then click **Reports**.

   Verify that the Reports page is opened successfully.

# Sanity Checks for Oracle Advanced Authentication

Learn about the sanity checks applicable to Oracle Advanced Authentication (OAA).

To verify that Oracle Advanced Authentication is working you:

1. Create an HTTP test page.

2. Create an OAA test user.
3. Protect the test page with OAA.

When you access the test page, you are prompted to log in and are redirected to OAA, and then you are asked to key in the one-time pin sent to you by email. The test page is displayed after you enter the pin.

The following sections explain each of the above steps in detail:

- Creating a Test Page
- Creating an OAA Test User
- Creating a Protection Policy for the Test Page

## Creating a Test Page

To create a test page:

1. Create a file called `test_page.html` with the following content:

```
<!DOCTYPE html>
<html>
<body>

<h1>This is a Test Page</h1>

</body>
</html>
```

2. Copy the file to the `htdocs` folder on the Oracle HTTP server. For example:`/u02/private/oracle/config/domains/ohsDomain/config/fmwconfig/components/OHS/ohs1/htdocs`.

## Creating an OAA Test User

To create an OAA test user:

1. Create a file called `test_user.ldif` with the following content:

```
dn: cn=<OAA_USER>,<LDAP_USER_SEARCHBASE>
changetype: add
objectClass: orclUserV2
objectClass: oblixorgperson
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: oblixPersonPwdPolicy
objectClass: orclAppIDUser
objectClass: orclUser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
givenName: <OAA_USER>
uid: <OAA_USER>
orclIsEnabled: ENABLED
sn: <OAA_USER>
```

```
userPassword: <OAA_USER_PWD>
mail: <OAA_USER_EMAIL>
orclSAMAccountName: <OAA_USER>
cn: <OAA_USER>
postalCode: <OAA_USER_POSTCODE>
obpasswordchangeflag: false
ds-pwp-password-policy-dn:
cn=FAPolicy,cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext,<LDAP_SEA
RCHBASE>

dn:cn=<OAA_USER_GROUP>,<LDAP_GROUP_SEARCHBASE>
changetype: modify
add: uniqueMember
uniqueMember: cn=<OAA_USER>,<LDAP_USER_SEARCHBASE>
```

**Table 29-3 Attributes and Their Descriptions**

| Attribute | Description |
|---|---|
| OAAUSER | The name of the user you want to create. For example: oaauser. |
| USER_SEARCHBASE | The location in LDAP where the user names are stored. For example: cn=Users,dc=example,dc=com. |
| OAA_USER_PWD | The password you want to assign to the user. |
| OAA_USER_EMAIL | A valid email address where you want the one-time pins to be sent. |
| OAA_USER_POSTCODE | An entry for the post code. This is required if you want to validate the authentication through the Oracle Mobile Authenticator. |
| OAA_USER_GROUP | The the LDAP group you created for the OAA users. For example: OAA-App-User. See #unique_992. |
| GROUP_SEARCH_BASE | The location in your directory where user groups are stored. For example: cn=Groups,dc=example,dc=com. |

2. Copy the file to your LDAP server. For example:

```
kubectl cp test_user.ldif oudns/edg-oud-ds-rs-0:/u01/oracle/config-input
```

3. Load the ldif file to the LDAP directory.

```
/u01/oracle/oud/bin/ldapmodify -h <OUD_POD_PREFIX>-oud-ds-rs-lbr-
ldap.<OUDNS>.svc.cluster.local -p 1389 -D <LDAP_ADMIN_USER> -w
<LDAP_ADMIN_PWD> -f /u01/oracle/config-input/test_user.ldif
```

For example:

```
/u01/oracle/oud/bin/ldapmodify -h edg-oud-ds-rs-lbr-
ldap.oudns.svc.cluster.local -p 1389 -D cn=oudadmin -w <password> -f /u01/
oracle/config-input/test_user.ldif
```

# Creating a Protection Policy for the Test Page

To create an OAM protection policy for your test page:

1. Log in to the OAM Administration Console using the URL `http://iadadmin.example.com/oamconsole`.
2. In the Access Manager section of the launch screen, click **Application Domains**.
3. Click **Search**, and then click **IAM Suite**.
4. Click the **Resources** tab.
5. On the Resources screen, click **Create** and enter the following information:
   - **Type** : http
   - **Resource URL**: `/test_page.html`
   - **Protection Level**: Protected
   - **Authentication Policy**: OAA_MFA-Policy
   - **Authorization Policy**: Protected Resource Policy
6. Click **Apply**.

Validate that OAA is working by accessing the test page at `https://login.example.com/test_page.html`.

# 30
# Troubleshooting

You can troubleshoot the common issues that may arise with the Identity and Access Management enterprise deployment. The solutions provided for the common problems help you resolve them quickly.

This chapter includes the following topics:

- Troubleshooting Oracle Access Management Access Manager
  Learn about some of the common problems that you may encounter with Oracle Access Manager and the actions you can take to resolve them.
- Troubleshooting Oracle Identity Governance
  Learn about some of the common problems that may arise with Oracle Identity Manager and the actions you can take to resolve the problem.
- Troubleshooting Oracle SOA Suite
  Learn about the transaction timeout error that may arise with Oracle SOA Suite and the action you can take to resolve the problem.
- Troubleshooting Coherence Clusters
- Troubleshooting OAM/OIG Integration
  Learn about the error you may encounter during the integration process and the solution to fix this error.
- Troubleshooting Oracle Advanced Authentication
  Learn about some of the common problems that may arise with the Oracle Advanced Authentication and the actions you can take to resolve the problem.
- General Troubleshooting
  Learn about the error you may encounter when starting the Managed Server from the WebLogic Console and the resolution to fix the error.
- Troubleshooting Kubernetes Domains
  Learn about some of the common problems you may encounter with Kubernetes domains and the actions you can take to resolve these problems.

## Troubleshooting Oracle Access Management Access Manager

Learn about some of the common problems that you may encounter with Oracle Access Manager and the actions you can take to resolve them.

- Access Manager Runs out of Memory
- Access Domain Creation Times Out
- User Reaches the Maximum Allowed Number of Sessions
- Policies Do Not Get Created When Oracle Access Management Access Manager is First Installed
- You Are Not Prompted for Credentials After Accessing a Protected Resource
- Cannot Log In to Access Management Console
- Oracle Coherence Cluster Startup Errors in oam_policy_mgr Server Logs

# Access Manager Runs out of Memory

**Problem**

After Access Manager has been running for a while, you see the following error message in the output:

```
Attempting to allocate 1G bytes
There is insufficient native memory for the Java Runtime Environment to continue.
```

**Possible reasons**

- The system is out of physical RAM or swap space.
- In 32 bit mode, the process size limit was reached.

**Solutions**

- Reduce memory load on the system.
- Increase physical memory or swap space.
- Check if swap backing store is full.
- Use 64 bit Java on a 64 bit OS.
- Decrease Java heap size (-Xmx/-Xms).
- Decrease number of Java threads.
- Decrease Java thread stack sizes (-Xss).
- Disable compressed references (-XXcompressedRefs=false).
- Ensure that command line tool `adrci` can be executed from the command line.
  - at oracle.dfw.impl.incident.ADRHelper.invoke(ADRHelper.java:1309)
  - at oracle.dfw.impl.incident.ADRHelper.createIncident(ADRHelper.java:929
  - at oracle.dfw.impl.incident.DiagnosticsDataExtractorImpl.createADRIncident(Diagnostics DataExtractorImpl.java:1116)
- On both OAMHOST1 and OAMHOST2, edit the file `setSOADomainEnv.sh`, which is located in *`IAD_MSERVER_HOME`*`/bin` and locate the line which begins:

  ```
  PORT_MEM_ARGS=
  ```

  Change this line so that it reads:

  ```
  PORT_MEM_ARGS="-Xms768m -Xmx2560m"
  ```

# Access Domain Creation Times Out

**Problem**

When creating the Access domain, you will see an error in the log file, similar to the following:

```
[ERROR] Exiting due to failure - the job status is not Completed!
```

**Possible reasons**

There is a performance issue in your setup.

**Solution**

Increase the timeout value when running the command to create the domain. For example:

```
./create-domain.sh -i $WORKDIR/create-domain-inputs.yaml -t 1200 -o output
```

Where, `1200` is the number of seconds to wait before timing out. The default value is `600`.

## User Reaches the Maximum Allowed Number of Sessions

**Problem**

The Access Manager Server displays an error message similar to this:

```
The user has already reached the maximum allowed number of sessions. Please
close one of the existing sessions before trying to login again.
```

**Solution**

If users log in multiple times without logging out, they might overshoot the maximum number of configured sessions. You can modify the maximum number of configured sessions by using the Access Management Administration Console.

To modify the configuration by using the Access Management Administration Console, proceed as follows:

1. Go to **System Configuration** -> **Common Settings** -> **Session**
2. Increase the value in the **Maximum Number of Sessions per User** field to cover all concurrent login sessions expected for any user. The range of values for this field is from 1 to any number.

## Policies Do Not Get Created When Oracle Access Management Access Manager is First Installed

**Problem**

The Administration Server takes a long time to start after configuring Access Manager.

**Solution**

Tune the Access Manager database. When the Administration Server first starts after configuring Access Manager, it creates a number of default policies in the database. If the database is distant or in need of tuning, this can take a significant amount of time.

```
Resources
Authentication Policies
   Protected Higher Level Policy
   Protected Lower Level Policy
   Publicl Policy
Authorization Policies
   Authorization Policies
```

If you do not see these items, the initial population has failed. Check the Administration Server log file for details.

## You Are Not Prompted for Credentials After Accessing a Protected Resource

**Problem**

When you access a protected resource, Access Manager should prompt you for your user name and password. For example, after creating a simple HTML page and adding it as a resource, you should see credential entry screen.

**Solution**

If you do not see the Credential Entry screen, perform the following steps:

1. Verify that host aliases for IAMAccessDomain have been set. You should have aliases for `IAMAccessDomain`:80, `IAMAccessDomain`:Null, `IADADMIN.example.com:80`, and `login.example.com:443`, where Port `80` is *`HTTP_PORT`* and Port `443` is *`HTTP_SSL_PORT`*.
2. Verify that WebGate is installed.
3. Verify that `ObAccessClient.xml` was copied from *`IAD_ASERVER_HOME`*`/output` to the WebGate Lib directory and that OHS was restarted.
4. When you first created the `ObAccessClient.xml` file, it was not formatted. When you restart OHS, re-examine the file to ensure that it is formatted. OHS gets a new version of the file from Access Manager when it first starts.
5. Shut down the Access Manager servers and access the protected resource. If you do not see an error saying Access Manager servers are not available, re-install WebGate.

## Cannot Log In to Access Management Console

**Problem**

You cannot log in to the Access Management Console. The Administration Server diagnostic log might contain an error message similar to this:

```
Caused by: oracle.security.idm.OperationFailureException:
oracle.security.am.common.jndi.ldap.PoolingException [Root exception is
oracle.ucp.UniversalConnectionPoolException:
Invalid life cycle state.
 Check the status of the Universal Connection Pool]
         at
oracle.security.idm.providers.stdldap.UCPool.acquireConnection(UCPool.java:112)
```

**Solution**

Remove the `/tmp/UCP*` files and restart the Administration Server.

## Oracle Coherence Cluster Startup Errors in oam_policy_mgr Server Logs

**Problem**

The `oam_policy_mgr2` server has oam application deployment in failed state. The `oam_policy_mgr2` server logs report request timeout exceptions while starting the cluster service, similar to following logs:

```
Oracle Coherence GE 3.7.1.13 <Warning> (thread=Cluster, member=n/a): Delaying
formation of a new cluster; IpMonitor failed to verify the reachability of senior
Member(Id=1, Timestamp=, Address=, MachineId=,
Location=site:,machine:IADADMINVHN,process:8499, Role=WeblogicServer); if this
persists it is likely the result of a local or remote firewall rule blocking
either ICMP pings, or connections to TCP port 7>

Error while starting cluster: com.tangosol.net.RequestTimeoutException: Timeout
during service start: ServiceInfo(Id=0, Name=Cluster, Type=Cluster
MemberSet=MasterMemberSet(
ThisMember=null
OldestMember=null
ActualMemberSet=MemberSet(Size=0
)
MemberId|ServiceVersion|ServiceJoined|MemberState
RecycleMillis=1200000
RecycleSet=MemberSet(Size=0
)
)
)
at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.Grid.onStartupTimeou
t(Grid.CDB:3)

at
com.tangosol.coherence.component.util.daemon.queueProcessor.Service.start(Service.CDB:28)

at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.Grid.start(Grid.CDB:6
)
```

### Solution

This is a known issue. In some of the environments, the Access Policy Manager Server that is not running on the same host as the WebLogic Administration Server is unable to start the coherence cluster service, which results in the oam application deployment to be in failed state. To solve this issue, you must create a server instance for the effected Access Policy Manager Server by completing the following steps:

1. Log in to the OAM Console using the following URL:

   `http://iadadmin.example.com/oamconsole`

   Log in as the Access Manager administration user you created when you prepared the ID Store. For example, `oamadmin`.

2. Click **Configuration**.
3. Click **Server Instances** from the configuration launch pad.
4. Click a new server instance for the Access Policy Manager WebLogic Managed Server, that is not running on the same machine as the IAMAccessDomain Admin Server. For example:
   - Name: `oam_policy_mgr2`
   - Port: 14150
   - Host: OAMHOST2 (For consolidated topology, the host will be IAMHOST2)

> **Note:**
>
> Provide the OAM Proxy details similar to the server instance for `oam_server`.

5. Click **Apply**.

# Errors in Log File When Starting OAM Servers

Problem

When you start the OAM Servers, errors similar to the following are seen in the log files which causes LCM heath check module to fail:

```
[oam_server1] [TRACE:16] [] [oracle.oam.config] [tid: DistributedCacheWorker:4] [userId:
<anonymous>] [ecid:
0000LGmRJqxB9DE5N7P5ie1N5mOd000004,1:16514] [APP: oam_server#11.1.2.0.0] [SRC_CLASS:
oracle.security.am.admin.config.util.MapUtil] [SRC_METHOD:
getDefaultedStringValue] property not found at path:[Ljava.lang.String;@43537067
Defaulting to value:,
[2016-04-20T06:55:39.982+00:00] [oam_server1] [TRACE:16] [] [oracle.oam.config] [tid:
DistributedCacheWorker:4] [userId: <anonymous>] [ecid:
0000LGmRJqxB9DE5N7P5ie1N5mOd000004,1:16514] [APP: oam_server#11.1.2.0.0] [SRC_CLASS:
oracle.security.am.admin.config.util.MapUtil] [SRC_METHOD: getStringValue] THROW[[
oracle.security.am.admin.config.ConfigurationException: Cannot get java.lang.String
value from configuration for key ResponseEscapeChar. Object null found.
at
oracle.security.am.admin.config.util.MapUtil.handleFailedAttributeAccess(MapUtil.java:447
)
at oracle.security.am.admin.config.util.MapUtil.getStringValue(MapUtil.java:130)
at oracle.security.am.admin.config.util.MapUtil.getDefaultedStringValue(MapUtil.java:147)
at
oracle.security.am.engines.common.identity.provider.util.IdStoreConfig.initializeConfig(I
dStoreConfig.java:76)
at
oracle.security.am.engines.common.identity.provider.util.IdStoreConfig.<init>(IdStoreConf
ig.java:69)
at
oracle.security.am.engines.common.identity.provider.util.IdStoreConfig.getConfig(IdStoreC
onfig.java:128)
at
oracle.security.am.engines.common.identity.util.OAMUserAttribute.getStringValue(OAMUserAt
tribute.java:76)
at
oracle.security.am.engines.common.identity.util.OAMUserAttribute.toString(OAMUserAttribut
e.java:114)
at java.lang.String.valueOf(String.java:2849)
at java.lang.StringBuilder.append(StringBuilder.java:128)
at java.util.AbstractMap.toString(AbstractMap.java:523)
at java.lang.String.valueOf(String.java:2849)
at java.lang.StringBuilder.append(StringBuilder.java:128)
at
oracle.security.am.engines.common.identity.util.OAMIdentity.toString(OAMIdentity.java:678
)
at java.lang.String.valueOf(String.java:2849)
at java.lang.StringBuilder.append(StringBuilder.java:128)
at oracle.security.am.engines.sso.SSOSubject.toString(SSOSubject.java:238)
at java.lang.String.valueOf(String.java:2849)
at java.lang.StringBuilder.append(StringBuilder.java:128)
at oracle.security.am.engines.sme.impl.SessionImpl.toString(SessionImpl.java:629)
```

```
at java.lang.String.valueOf(String.java:2849)
at java.lang.StringBuilder.append(StringBuilder.java:128)
at
oracle.security.am.engines.sme.mapimpl.db.DbOraSmeStore.loadSession(DbOraSmeStore.java:17
05)
at
oracle.security.am.engines.sme.mapimpl.db.DbOraSmeStore.loadSession(DbOraSmeStore.java:16
91)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:606)
at
oracle.security.am.foundation.mapimpl.coherence.store.DataConnectionUtility.invokeSqlOper
ationWithRetries(DataConnectionUtility.java:275)
at oracle.security.am.engines.sme.mapimpl.db.DbOraSmeStore.load(DbOraSmeStore.java:1284)
at
com.tangosol.net.cache.ReadWriteBackingMap$CacheStoreWrapper.loadInternal(ReadWriteBackin
gMap.java:5676)
at
com.tangosol.net.cache.ReadWriteBackingMap$StoreWrapper.load(ReadWriteBackingMap.java:475
4)
at com.tangosol.net.cache.ReadWriteBackingMap.get(ReadWriteBackingMap.java:717)
at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.partitionedServi
ce.PartitionedCache$Storage.get(PartitionedCache.CDB:10)
at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.partitionedServi
ce.PartitionedCache.onGetRequest(PartitionedCache.CDB:23)
at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.partitionedServi
ce.PartitionedCache$GetRequest.run(PartitionedCache.CDB:1)
at com.tangosol.coherence.component.util.DaemonPool$WrapperTask.run(DaemonPool.CDB:1)
at com.tangosol.coherence.component.util.DaemonPool$WrapperTask.run(DaemonPool.CDB:32)
at com.tangosol.coherence.component.util.DaemonPool$Daemon.onNotify(DaemonPool.CDB:66)
at com.tangosol.coherence.component.util.Daemon.run(Daemon.CDB:42)
at java.lang.Thread.run(Thread.java:745)
]]
```

Solution

This occurs when OAM servers cannot communicate with each other using the coherence port. This is often caused by iptables. The workaround for this issue is as follows:

1. Edit the file `/etc/sysconfig/iptables` on both OAMHOST1 and OAMHOST2 and add the following line:

   ```
   # Generated by iptables-save v1.4.7 on Tue Apr 19 10:02:45 2016
   *filter
   :INPUT ACCEPT [593:243587]
   :FORWARD ACCEPT [0:0]
   :OUTPUT ACCEPT [614:423013]
   -A INPUT -p tcp -m state --state NEW -m tcp --dport 9095 -j ACCEPT
   -A INPUT -p tcp -m state --state NEW -m tcp --dport 9097 -j ACCEPT
   COMMIT
   ```

   In the above set of lines, `9095` and `9097` are the coherence ports being used.

2. Save the file and restart the servers.

## Too Many Redirects Error in Browser

**Problem**

When navigating from one application to another that uses the same OAM for SSO, you get a redirection error in the web browser. There are two different configurations to validate.

**Solution 1:**

1. Log in to the OAM Console at `iadadmin.example.com/oamconsole`.
2. From the Launch Pad, click the **Agents** icon.
3. In the resulting window > **Webgates** tab, click **search**. No search parameters need to be input.
4. In the search results, click the IAMSuiteAgent link.
5. Ensure that the Primary Cookie Domain is set to the domain that is used for the `login.example.com` domain. For example: example.com.
6. Restart all WebGate OHS instances.

**Solution 2:**

Ensure that the date and time on all OHS and OAM servers are within 60 seconds of each other. If they are not:

1. Ensure that the NTP setting are the same and valid on all OHS and OAM hosts.
2. Start or restart the *ntpd* service on all hosts.
3. Restart all WebGate OHS instances, the OAM domain AdminServer, and all Managed Servers.

# Troubleshooting Oracle Identity Governance

Learn about some of the common problems that may arise with Oracle Identity Manager and the actions you can take to resolve the problem.



## OIM Bootstrap Process Fails

**Problem**

The OIM Bootstrap process fails after deploying composites. The error appears as follows:

```
Deployment of SOA Composites :-/<INSTALL_LOCATION>/Oracle_Home/idm/server/workflows/
composites/scajars/sca_DefaultRequestApproval_rev6.0.jar is successful>
<Jun 12, 2018 4:20:26,136 PM CEST> <Info> <oracle.iam.OIMPostConfigManager> <BEA-000000>
<updating feature:DEPLOYSOACOMPOSITESwith state :COMPLETEwith executionTime190108>
java.sql.SQLException: Connection closed
```

This is caused by a performance issue.

**Solution**

To resolve the issue temporarily, increase the inactivity timeouts on the following data sources:

- `oimJMSStoreDS`
- `oimOperationsDB`

The settings can be restored to their original values after the upgrade is complete.

1. Log in to the WebLogic Server Administration Console.
2. Click **Lock and Edit**.
3. Click **Services**, **Data Sources**, and then select the <*Data source name*>.
4. Click the **Connection Pool** tab.
5. Under the Advanced section, increase the value of **Inactive Connection Timeout**.
6. Save and activate the changes.
7. Restart the OIM Managed Server.

## java.io.FileNotFoundException When Running Oracle Identity Governance Configuration

**Problem**

The following content was added to address bug 12390838

When you run Oracle Identity Manager configuration, the error `java.io.FileNotFoundException: soaconfigplan.xml (Permission denied)` may appear and Oracle Identity Manager configuration might fail.

**Solution**

To workaround this issue:

1. Delete the file `/tmp/soaconfigplan.xml`.
2. Start the configuration again (*`IGD_ORACLE_HOME`*`/bin/config.sh`).

## ResourceConnectionValidationxception When Creating User in Oracle Identity Governance

**Problem**

The following content was added to address bug 9816870

If you are creating a user in Oracle Identity Manager (by logging into Oracle Identity Manager System Administration Console, clicking the Administration tab, clicking the **Create User** link, entering the required information in the fields, and clicking **Save**) in an active-active Oracle Identity Manager configuration, and the Oracle Identity Manager server that is handling the

request fails, you may see a "ResourceConnectionValidationxception" in the Oracle Identity Manager log file, similar to:

```
[2010-06-14T15:14:48.738-07:00] [oim_server2] [ERROR] [] [XELLERATE.SERVER]
[tid: [ACTIVE].ExecuteThread: '0' for queue: 'weblogic.kernel.Default
(self-tuning)'] [userId: xelsysadm] [ecid:
004YGJGmYrtEkJV6u3M6UH00073A0005EI,0:1] [APP: oim#11.1.1.3.0] [dcid:
12eb0f9c6e8796f4:-785b18b3:12938857792:-7ffd-0000000000000037] [URI:
/admin/faces/pages/Admin.jspx] Class/Method:
PooledResourceConnection/heartbeat encounter some problems: Operation timed
out[[
com.oracle.oim.gcp.exceptions.ResourceConnectionValidationxception: Operation
timed out
        at
oracle.iam.ldapsync.impl.repository.LDAPConnection.heartbeat(LDAPConnection.ja
va:162)
        at
com.oracle.oim.gcp.ucp.PooledResourceConnection.heartbeat(PooledResourceConnec
tion.java:52)
         .
         .
         .
```

**Solution**

Despite this exception, the user is created correctly.

# OIG Managed Servers Fail to Join Coherence Cluster

**Problem**

One or more Managed Servers in the domain fail to start. Examining the log files shows that they are unable to join the Coherence cluster.

**Solution 1**: **Check Firewall (iptables) Requirements**

Some Kubernetes distributions create `iptables` rules that block some types of traffic that Coherence requires to form clusters. If you are not able to form clusters, then you can check for this issue using the following command:

```
iptables -t nat -v  -L POST_public_allow -n
```

You should output similar to the following:

```
Chain POST_public_allow (1 references)
pkts bytes target     prot opt in     out     source               destination
164K   11M MASQUERADE  all  --  *      !lo     0.0.0.0/0            0.0.0.0/0
   0     0 MASQUERADE  all  --  *      !lo     0.0.0.0/0            0.0.0.0/0
```

If you see a similar output, for example, if you see any entries in this chain, then you need to remove them. You can remove the entries using this command:

```
iptables -t nat -v -D POST_public_allow 1
```

Note that you will need to run that command for each line. So, in this example, you would need to run it twice.

After you are done, you can run the previous command again and verify that the output is now an empty list.

After making this change, restart your domains and the Coherence cluster should now form correctly.

**Solution 2**: **Make iptables Updates Permanent Across Reboots**

The recommended way to make `iptables` updates permanent across reboots is to create a `systemd` service that applies the necessary updates during the startup process.

Here is an example; you may need to adjust this to suit your own environment:

- Create a `systemd` service:

```
echo 'Set up systemd service to fix iptables nat chain at each reboot (so
Coherence will work)...'


mkdir -p /etc/systemd/system/


cat > /etc/systemd/system/fix-iptables.service << EOF
[Unit]
Description=Fix iptables
After=firewalld.service
After=docker.service

[Service]
ExecStart=/sbin/fix-iptables.sh

[Install]
WantedBy=multi-user.target
EOF
```

- Create the script to update `iptables`:

```
cat > /sbin/fix-iptables.sh << EOF
#!/bin/bash
echo 'Fixing iptables rules for Coherence issue...'
TIMES=$((`iptables -t nat -v -L POST_public_allow -n --line-number | wc -
l` - 2))
COUNTER=1
while [ $COUNTER -le $TIMES ]; do
  iptables -t nat -v -D POST_public_allow 1
  ((COUNTER++))
done
EOF
```

- Start the service (or just reboot):

```
echo 'Start the systemd service to fix iptables nat chain...'


systemctl enable --now fix-iptables
```

# Oracle Identity Manager Reconciliation Jobs Fail

**Problem**

Oracle Identity Manager reconciliation jobs fail, or one of the following messages is seen in the log files:

- Error-1

```
LDAP Error 53 : [LDAP: error code 53 - Full resync required. Reason: The provided
cookie is older than the start of historical in the server for the replicated
domain : dc=example,dc=com]
```

- Error-2

```
LDAP: error code 53 - Invalid syntax of the provided cookie
```

This error is caused by the data in the Oracle Unified Directory change log cookie expiring because Oracle Unified Directory has not been written to for a certain amount of time.

**Solution**

1. Open a browser and go to the following location:

   ```
   http://igdadmin.example.com/sysadmin
   ```

2. Log in a as `xelsysadm` using the *`COMMON_IDM_PASSWORD`*.
3. Under **System Management**, click **Scheduler**.
4. Under **Search Scheduled Jobs**, enter `LDAP *` (there is a space before *) and hit **Enter**.
5. For each job in the search results, click on the job name on the left, then click **Disable** on the right.

   Do this for all jobs. If the job is already disabled do nothing.

6. Run the following commands on LDAPHOST1:

   ```
   cd LDAP_ORACLE_INSTANCE/OUD/bin
   ./ldapsearch -h LDAPHOST1 -p 1389 -D "cn=oudadmin" -b "" -s base "objectclass=*"
   lastExternalChangelogCookie

   Password for user 'cn=oudadmin': <OudAdminPwd>
   dn: lastExternalChangelogCookie: dc=example,dc=com:00000140c682473c263600000862;
   ```

   Copy the output string that follows `lastExternalChangelogCookie:`. This value is required in the next step. For example,

   ```
   dc=example,dc=com:00000140c682473c263600000862;
   ```

   The Hex portion must be 28 characters long. If this value has more than one Hex portion then separate the 28char portions with spaces. For example:

   ```
   dc=example,dc=com:00000140c4ceb0c07a8d00000043 00000140c52bd0b9104200000042
   00000140c52bd0ba17b9000002ac 00000140c3b290b076040000012c;
   ```

7. Run each of the following LDAP reconciliation jobs once to reset the last change number.:
   - LDAP Role Delete Reconciliation
   - LDAP User Delete Reconciliation
   - LDAP Role Create and Update Reconciliation
   - LDAP User Create and Update Reconciliation

- LDAP Role Hierarchy Reconciliation
- LDAP Role Membership Reconciliation

To run the jobs:

a. Login to the OIM System Administration Console as the user `xelsysadm`.

b. Under **System Configuration**, click **Scheduler**.

c. Under **Search Scheduled Jobs**, enter `LDAP *` (there is a space before *) and hit **Enter**.

d. Click on the job to be run.

e. Set the parameter **Last Change Number** to the value obtained in step 6.

For example:

```
dc=example,dc=com:00000140c4ceb0c07a8d00000043 00000140c52bd0b9104200000042
00000140c52bd0ba17b9000002ac 00000140c3b290b076040000012c;
```

f. Click **Run Now**.

g. Repeat for each of the jobs in the list at the beginning of this step.

8. For each incremental recon job whose last changelog number has been reset, execute the job and check that the job now completes successfully.
9. After the job runs successfully, re-enable periodic running of the jobs according to your requirements.

If the error appears again after the incremental jobs have been re-enabled and run successfully ("Full resync required. Reason: The provided cookie is older..."), then increase the OUD cookie retention time. Although there is no hard and fast rule as to what this value should be, it should be long enough to avoid the issue, but small enough to avoid unnecessary resource consumption on OUD. One or two weeks should suffice. Run the following command on each OUD instance to increase the retention time to two weeks:

```
cd OUD_ORACLE_INSTANCE/bin

./dsconfig set-replication-server-prop --provider-name "Multimaster Synchronization" --
set replication-purge-delay:2w -D cn=oudadmin --trustAll -p 4444 -h LDAPHOSTn

Password for user 'cn=oudadmin':  <OudAdminPswd>
Enter choice [f]: f
```

# OIM Reconciliation Jobs Fail When Running Against Oracle Unified Directory

**Problem**

Reconciliation jobs fail when running against Oracle Unified Directory (OUD). The following error is seen in the OIM WebLogic Server logs:

```
LDAP: error code 53 - Invalid syntax of the provided cookie
```

**Solution**

Perform the workaround described in Oracle Identity Manager Reconciliation Jobs Fail. If this workaround does not resolve the issue, try the following solution:

On each OIMHOST, update the *DOMAIN_HOME*`/config/fmwconfig/ovd/oim/adapters.os_xml` file with the following parameter:

```
<param name="eclCookie" value="false"/>
```

Restart the OIM and SOA Managed Servers.

## Cannot Open Reports from OIM Self Service Console

**Problem**

The reports cannot be opened from OIM Self Service Console.

**Solution**

When you enable the Identity Auditor feature in OIM, do the following configuration changes for the OIM-BI Publisher integration to work fine:

1. Log in to the IAMGovernanceDomain Enterprise Management Console.
2. Open the system MBean browser and update the MBean `"oracle.iam:Location=wls_oim1,name=Discovery,type=XMLConfig.DiscoveryConfig,XMLConfig=Config,Application=oim,ApplicationVersion=11.1.2.0.0"` with Value as `http://igdadmin.example.com/`.

   Here, *`igdadmin.example.com`* is the Governance Domain admin Load balancer URL.

## Pending Violations Not Displaying the Correct List

**Problem**

When viewing the pending violations list, you may see entries that are missing or entries that do not belong to the list.

**Solution**

If you encounter this issue, a restart of the OIG domain usually resolves it. If the issue is not resolved, raise a Service Request (SR) with Oracle Support.

## Domain Patching Failure

**Problem**

The OIG domain patching fails when you run the `patch_oig_domain.sh` script.

**Solution**

If you encounter a patching failure, run the following command to diagnose the issue:

```
$ kubectl describe domain <OIG_DOMAIN_NAME> -n <OIGNS>
```

For example:

```
kubectl describe domain governancedomain -n oigns
```

Use the output to diagnose the problem and resolve the issue. Also, check the log directory (by default under `$WORKDIR/kubernetes/domain-lifecycle`) for more details.

# Troubleshooting Oracle SOA Suite

Learn about the transaction timeout error that may arise with Oracle SOA Suite and the action you can take to resolve the problem.



## Transaction Timeout Error

**Problem**

The following transaction timeout error appears in the log:

```
Internal Exception: java.sql.SQLException: Unexpected exception while enlisting
 XAConnection java.sql.SQLException: XA error: XAResource.XAER_NOTA start()
failed on resource 'SOADataSource_soaedg_domain': XAER_NOTA : The XID
is not valid
```

**Solution**

Check your transaction timeout settings, and be sure that the JTA transaction time out is less than the DataSource XA Transaction Timeout, which is less than the `distributed_lock_timeout` (at the database).

With the out of the box configuration, the SOA data sources do not set XA timeout to any value. The `Set XA Transaction Timeout` configuration parameter is unchecked in the WebLogic Server Administration Console. In this case, the data sources use the domain level JTA timeout which is set to `30`. Also, the default `distributed_lock_timeout` value for the database is `60`. As a result, the SOA configuration works correctly for any system where transactions are expected to have lower life expectancy than such values. Adjust these values according to the transaction times your specific operations are expected to take.

# Troubleshooting Coherence Clusters

**Problem**

Coherence clusters are failing to form when SOA/OIG pods are started.

**Solution**

Ensure that your cluster has been configured with the steps described in Coherence requirements.

# Troubleshooting OAM/OIG Integration

Learn about the error you may encounter during the integration process and the solution to fix this error.

**Problem**

The following content was added to address bug 27567130

Whilst running configureLDAPConnector, you see the following error message:

```
2018-02-19 06:54:05] LDAPConnectorConfigTool.configureLDAPConnector:
exception: java.lang.reflect.UndeclaredThrowableException  [2018-02-19
```

```
06:54:05] javax.management.InstanceNotFoundException: Unable to  contact
MBeanServer for
oracle.iam:Location=oim_server1,name=SSOIntegrationMXBean,type=IAMAppRuntimeMB
  ean,Application=oim  at weblogic.utils.StackTraceDisabled.unknownMethod()
```

**Solution**

This is caused by the OIM Managed Server being called something other than `oim_server1`. This can be recovered by executing the following workaround.

Ensure that your OIM Managed Server is running.

1. Log in to Oracle Fusion Middleware control using the following URL: `http://igdadmin.example.com/em`.
2. Start the System Mbean Browser by selecting Weblogic Domain and then clicking on System MBean browser.
3. Click on find and enter the Mbean name **SSOIntegrationMXBean** .
4. Click **Search**.
5. When the MBean is found, click **Operations > addContainerRules** .
6. Enter the following information:

   ```
   Oracle_Home set to the value of IGD_ORACLE_HOME dirType. set to OUD
   userContainer set to
   cn=users,
   dc=example,
   dc=com
   roleContatiner set to cn=groups,
   dc=example,dc=com
   ```

7. Click **Invoke** button.

# Troubleshooting Oracle Advanced Authentication

Learn about some of the common problems that may arise with the Oracle Advanced Authentication and the actions you can take to resolve the problem.

- Creating the Oracle Database Schema Causes an Error
- OAA Deployment Results in an Error

## Creating the Oracle Database Schema Causes an Error

**Problem**

When you create the Oracle Database schema, an error similar to the following is shown:

```
ORA-12521: TNS:listener does not currently know of instance requested in connect
descriptor
```

**Solution**

Ensure that the `database.name` parameter is empty. That is, no value should appear after the "=" sign.

## OAA Deployment Results in an Error

**Problem**

When you deploy OAA, the following message is shown:

```
OAUTH validation failed
Oauth validation failed..
command terminated with exit code 1
```

**Solution**

Run the following command inside the OAA Management container to get more information:

```
/u01/oracle/scripts/validateOauthForOAA.sh -f
/u01/oracle/scripts/settings/installOAA.properties -d true
```

# General Troubleshooting

Learn about the error you may encounter when starting the Managed Server from the WebLogic Console and the resolution to fix the error.

- Cannot Start Managed Server from WebLogic Console

## Cannot Start Managed Server from WebLogic Console

**Problem**

When you start a Managed Server from the WebLogic Console, the following error is shown:

```
. For server WLS_BI1, the Node Manager associated with machine OIMHOST1 is not reachable.
. All of the servers selected are currently in a state which is incompatible with this
operation or are not associated with a running Node Manager or you are not authorized to
perform the action requested. No action will be performed.
```

**Solution 1**

Check if the Node Manager is started on the target host. If not, start it.

**Solution 2**

Verify that the domain is listed in the file `nodemanager.domains`, which is located in the directory *`SHARED_CONFIG_DIR`*`/nodemanger/hostname`. If not, do the following:

1. Start the WebLogic Scripting Tool (WLST) by running the following command from the location *`ORACLE_HOME`*`/oracle_common/common/bin/`:

   ```
   ./wlst.sh
   ```

2. Connect to the domain you wish to add by running the following command:

   ```
   connect('weblogic_user','password','t3://ADMINVHN:AdminPort')
   ```

   In this command:

   *`weblogic_user`* is the WebLogic Administration user. For example, `weblogic` or `weblogic_idmw`.

   *`password`* is the password of the WebLogic Administration user.

*ADMINVHN* is the Virtual host name of the Administration Server. For example, `IGDADMINVHN` or `IADADMINVHN`.

`adminPort` is the port on which the Administration Server is running. For example, `7101`.

Sample Command:

```
connect('weblogic_idm','<password>','t3://IGDADMINVHN.example.com:7001')
```

3. Enrol the domain using the following command:

```
nmEnroll(domainDir=absolute_path_to_the_domain,nm_Home=absolute_path_to_the_nodemanager_home)
```

For example:

```
nmEnroll(domainDir='/u02/private/oracle/config/domains/
IAMGovernanceDomain/',nmHome='/u01/oracle/config/nodemanger/hostname)')
```

> **Note:**
>
> For Managed Servers, the domain home should always be specified as the local Managed Server directory.

# Troubleshooting Kubernetes Domains

Learn about some of the common problems you may encounter with Kubernetes domains and the actions you can take to resolve these problems.

- WebLogic Domain Creation Fails
- Domain Fails to Start
- WebLogic Operator Fails to Manage Namespace

## WebLogic Domain Creation Fails

**Problem**

The WebLogic domain creation fails when you run the `create-domain.sh` command.

**Solution**

To diagnose the issue:

1. Run the following command to diagnose the create domain job:

```
$ kubectl logs <domain_job> -n <domain_namespace>
```

For example:

```
$ kubectl logs accessinfra-create-fmw-infra-sample-domain-job-c6vfb -n
accessns
```

Also run:

```
$ kubectl describe pod <domain_job> -n <domain_namespace>
```

For example:

```
$ kubectl describe pod accessinfra-create-fmw-infra-sample-domain-job-
c6vfb -n accessns
```

Using the output you should be able to diagnose the problem and resolve the issue.

2. If any of the above commands return the following error, it indicates that there is a permissions error on the directory for the PV and PVC:

```
Failed to start container "create-fmw-infra-sample-domain-job": Error response from
daemon: error while creating mount source path
'/exports/IAMPVS/accessdomainpv ': mkdir /exports/IAMPVS/accessdomainpv : permission
denied
```

Check the following:

a. The directory has 777 permissions: `chmod -R 777 <work directory>/accessdomainpv`.

b. If it does have the permissions, check if an `oracle` user exists and the uid and gid equal `1000`.

Create the `oracle` user if it does not exist and set the uid and gid to `1000`.

c. Edit the `<work directory>/weblogic-kubernetes-operator/kubernetes/samples/scripts/create-access-domain-pv-pvc/create-pv-pvc-inputs.yaml` file and add a slash to the end of the directory for the `weblogicDomainStoragePath` parameter:

```
weblogicDomainStoragePath: /exports/IAMPVS/accessdomainpv/
```

After you have resolved the issue, delete the job and try again.

To delete the job, use the command:

```
kubectl -n <NAMESPACE> get all -o wide
```

This will list the name of the job, as follows:

```
accessdomain-create-oam-infra-domain-job-b6kfd
```

Now, use the following command to delete the job listed:

```
kubectl delete job -n <NAMESPACE> <JOBNAME>
```

For example:

```
kubectl delete job -n oamns accessdomain-create-oam-infra-domain-job
```

## Domain Fails to Start

**Problem**

Domain does not start.

### Solution

If you see that the Administration server pod has started, log in to the Administration Server using the following command and check the server log files:

```
kubectl exec -n <NAMESPACE> -ti <DOMAIN_NAME>-adminserver -- /bin/bash
```

If the Administration Server and Managed Servers fail to appear, check the WLS Operator logs:

```
kubectl logs -n opns weblogic-operator-<ID>
```

## WebLogic Operator Fails to Manage Namespace

### Problem

Operator log shows message

```
\":\"configmaps is forbidden: User \\\"system:serviceaccount:opns:op-sa\\\" cannot watch
resource
```

### Solution 1

Check the namespaces that the operator can manage by using the command:

```
kubectl get ns --selector="weblogic-operator=enabled"
```

If your namespace is not listed, ensure that the namespace is tagged with `weblogic-operator=enabled`. See the instructions for creating a namespace for the product you are configuring.

If all the namespaces are listed, ensure that you have enabled label-based namespace management by checking the values provided to the WebLogic Operator installation. Use the following command to check:

```
helm get values --namespace opns weblogic-kubernetes-operator
```

### Solution 2

If you are using named namespaces as in previous releases of the WebLogic Operator, it is not allowed to manage the namespace. Rerun the following command:

```
helm upgrade --reuse-values --namespace <operator namspace> --set
"domainNamespaces={<namespace>}" --wait weblogic-kubernetes-operator
kubernetes/charts/weblogic-operator
```

A

# Sample of the Schema Extension File and the Seeding File

The Schema Extensions file (`99-user.ldif`) extends the OUD schema with the Oracle Access Manager Object Classes and the Seeding file (`base.ldif`) seeds OUD with the Users and Groups required by Oracle Access Manager and Oracle Identity Governance.

This appendix includes the following topics:



## Sample of the Schema Extension File

Sample of the Schema Extension File: `99-user.ldif`

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.400 NAME 'obpasswordexpirydate' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.4 NAME 'obgroupdynamicfilter' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.10 NAME 'obgroupsubscriptionfilter'
DESC 'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.176 NAME 'oblastloginattemptdate' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.11 NAME 'obgroupsubscribemessage' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.16 NAME
'obgroupsimplifiedaccesscontrol' DESC 'Oracle Access Manager defined
attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user
defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.254 NAME 'obYetToBeAnsweredChallenge'
DESC 'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.7 NAME 'obgrouptype' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.13 NAME 'obgroupsubscribenotification'
DESC 'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
```

```
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.150 NAME 'obpasswordchangeflag' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.173 NAME 'oblockouttime' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.119 NAME 'obindirectmanager' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.12  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.42 NAME 'oblocationdn' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.250 NAME 'oblastfailedlogin' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.149 NAME 'obpasswordcreationdate' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.78 NAME 'obphoto' DESC 'Oracle Access
Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.5  X-ORIGIN
'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.200 NAME 'obresponsetimeout' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.5 NAME 'vGOSharedSecretDN' DESC 'v-GO
Shared Secret' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.59 NAME 'obobjectclass' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.14 NAME
'obgroupunsubscribenotification' DESC 'Oracle Access Manager defined
attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user
defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.12 NAME 'obgroupunsubscribemessage'
DESC 'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.3 NAME 'vGOConfigData'  SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.253 NAME 'obAnsweredChallenges' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.7 NAME 'vGODepartment' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.6 NAME 'obgroupcreationdate' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.2.0.1 NAME 'obuseraccountcontrol' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.175 NAME 'obresponsetries' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.249 NAME 'oblastsuccessfullogin' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.2.0.2 NAME 'oboutofofficeindicator' DESC
```

```
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.15 NAME 'obgrouppuredynamic' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.7.0.50 NAME 'obsubscriptiontypes' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.1 NAME 'vGORoleDN' DESC 'v-GO Role'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.103 NAME 'obuiconfig' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.43 NAME 'oblocationname' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.151 NAME 'obpasswordhistory' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.2 NAME 'vGOConfigType' DESC 'v-GO
Config Type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.39 NAME 'obid' DESC 'Oracle Access
Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  X-
ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.152 NAME 'obpasswordexpmail' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.83 NAME 'obpsftid' DESC 'Oracle Access
Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  X-
ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.174 NAME 'obfirstlogin' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.1 NAME 'obdirectreports' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.44 NAME 'oblocationtitle' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.85 NAME 'obrectangle' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.4 NAME 'vGORoleName' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.199 NAME 'oblastresponseattemptdate'
DESC 'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.3 NAME 'obgroupcreator' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.9 NAME 'obgroupadministrator' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.12  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.106 NAME 'obver' DESC 'Oracle Access
Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  X-
ORIGIN 'user defined' )
```

```
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.265 NAME 'oblockedon' DESC 'Oracle
Access Manager defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 X-
ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.1 NAME 'obgroupsubscriptiontype' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.3.0.8 NAME 'obgroupexpandeddynamic' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.6 NAME 'vGOSecretData' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.172 NAME 'oblogintrycount' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.3831.0.0.76 NAME 'obparentlocationdn' DESC
'Oracle Access Manager defined attribute type' SYNTAX
1.3.6.1.4.1.1466.115.121.1.12  X-ORIGIN 'user defined' )
attributeTypes: ( 1.3.6.1.4.1.10552.1.8 NAME 'vGoLocatorAttribute' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
objectClasses: ( 1.3.6.1.4.1.3831.0.1.13 NAME 'oblixorgperson' DESC 'Oracle
Access Manager defined objectclass' SUP top  AUXILIARY MAY ( obuiconfig $
oblocationdn $ obrectangle $ obpsftid $ obdirectreports $ obindirectmanager $
obuseraccountcontrol $ obobjectclass $ obver $ oboutofofficeindicator ) )
objectClasses: ( 1.3.6.1.4.1.3831.0.1.3 NAME 'oblixlocation' DESC 'Oracle
Access Manager defined objectclass' SUP top STRUCTURAL MUST ( obid )  MAY
( oblocationname $ oblocationtitle $ obphoto $ obparentlocationdn $
obrectangle $ obver ) )
objectClasses: ( 1.3.6.1.4.1.10552.2.2 NAME 'vGOSecret' SUP top STRUCTURAL
MAY ( vGOSecretData $ vGOSharedSecretDN $ cn $ o $ ou ) )
objectClasses: ( 1.3.6.1.4.1.10552.2.1 NAME 'vGOConfig' SUP top STRUCTURAL
MAY ( vGOConfigData $ vGOConfigType $ vGORoleDN $ cn $ o $ ou ) )
objectClasses: ( 1.3.6.1.4.1.3831.0.1.21 NAME 'oblixPersonPwdPolicy' DESC
'Oracle Access Manager defined objectclass' SUP top  AUXILIARY MAY
( obpasswordcreationdate $ obpasswordhistory $ obpasswordchangeflag $
obpasswordexpmail $ oblockouttime $ oblogintrycount $ obfirstlogin $
obresponsetries $ oblastloginattemptdate $ oblastresponseattemptdate $
obresponsetimeout $ oblastsuccessfullogin $ oblastfailedlogin $
obAnsweredChallenges $ obYetToBeAnsweredChallenge $ oblockedon ) )
objectClasses: ( 1.3.6.1.4.1.3831.0.1.40 NAME 'OIMPersonPwdPolicy' DESC
'Oracle Access Manager defined objectclass' SUP top  AUXILIARY MAY
( obpasswordexpirydate ) )
objectClasses: ( 1.3.6.1.4.1.10552.2.4 NAME 'vGORole' SUP top STRUCTURAL MAY
( vGORoleName $ vGODepartment $ cn $ o $ ou ) )
objectClasses: ( 1.3.6.1.4.1.10552.2.3 NAME 'vGOUserData' SUP top STRUCTURAL
MAY ( vGOSecretData $ vGORoleDN $ cn $ o $ ou ) )
objectClasses: ( 1.3.6.1.4.1.10552.2.5 NAME 'vGoLocatorClass' SUP top
STRUCTURAL MUST (vGoLocatorAttribute $ cn)  MAY ( o  ) )
objectClasses: ( 1.3.6.1.4.1.3831.8.1.1 NAME 'oblixadvancedgroup' DESC
'Oracle Access Manager defined objectclass' SUP top  AUXILIARY MAY ( obver $
obgroupsubscriptiontype $ obgroupexpandeddynamic $ obgrouppuredynamic $
obgroupadministrator $ obgroupsubscribemessage $ obgroupunsubscribemessage $
obgroupsubscriptionfilter $ obgroupsubscribenotification $
obgroupdynamicfilter $ obgroupsimplifiedaccesscontrol ) )
objectClasses: ( 1.3.6.1.4.1.3831.0.1.24 NAME 'oblixAuxLocation' DESC 'Oracle
Access Manager defined objectclass' SUP top  AUXILIARY MAY ( oblocationdn $
obrectangle ) )
```

```
objectClasses: ( 1.3.6.1.4.1.3831.0.1.14 NAME 'oblixgroup' DESC 'Oracle
Access Manager defined objectclass' SUP top  AUXILIARY MAY ( obgroupcreator $
obgroupcreationdate $ obgrouptype $ obsubscriptiontypes ) )
```

# Sample of the Seeding File

Sample of the Seeding File: `base.ldif`

```
dn: <LDAP_SEARCHBASE>
objectClass: domain
objectClass: orclSubscriber
objectClass: top
dc: <REGION>
aci: (targetattr="*")(version 3.0; acl "Allow OIMAdminGroup add, read and
write access to all attributes";
allow(add,read,search,compare,write,delete,import,export) groupdn="ldap:///
cn=<LDAP_OIGADMIN_GRP>,<LDAP_GROUP_SEARCHBASE>";)

dn: cn=OracleContext,<LDAP_SEARCHBASE>
cn: OracleContext
objectclass: top
objectclass: orclContext
objectclass: orclContextAux82
orclVersion: 90600
aci: (targetattr="*")(version 3.0; acl "OracleContext accessible by
OracleContextAdmins"; allow (all) groupdn="ldap:///
cn=OracleContextAdmins,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";)

dn: cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>
cn: Groups
objectclass: top
objectclass: orclContainer

dn: cn=OracleContextAdmins,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>
cn: OracleContextAdmins
uniquemember: <LDAP_ADMIN_USER>
objectclass: top
objectclass: groupofUniqueNames
objectclass: orclGroup
displayname: Oracle Context Administrators
description: Users who can administer all entities in this Oracle Context

dn: cn=Products,cn=OracleContext,<LDAP_SEARCHBASE>
cn: Products
objectclass: top
objectclass: orclContainer

dn: cn=Common,cn=Products,cn=OracleContext,<LDAP_SEARCHBASE>
cn: Common
orclCommonNickNameAttribute: uid
orclCommonApplicationGuidAttribute: orclGlobalID
orclCommonUserSearchBase:<LDAP_SEARCHBASE>
orclCommonGroupSearchBase:<LDAP_SEARCHBASE>
orclVersion: 90000
objectclass: top
```

```
objectclass: orclCommonAttributes
objectClass: orclCommonAttributesV2
orclUserObjectClasses: top
orclUserObjectClasses: person
orclUserObjectClasses: inetorgperson
orclUserObjectClasses: organizationalperson
orclUserObjectClasses: orcluser
orclUserObjectClasses: orcluserv2
orclcommonnamingattribute: cn
orclCommonGroupCreateBase: <LDAP_GROUP_SEARCHBASE>
orclCommonDefaultGroupCreateBase: <LDAP_GROUP_SEARCHBASE>
orclCommonKrbPrincipalAttribute: krbPrincipalName
orclCommonWindowsPrincipalAttribute: orclSAMAccountName

dn: cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext,<LDAP_SEARCHBASE>
cn: pwdPolicies
objectclass: top
objectclass: orclContainer

dn: <LDAP_USER_SEARCHBASE>
objectClass: orclContainer
objectClass: top
cn: users
aci: (targetattr="obUserAccountControl||obLoginTryCount||obLockoutTime||
oblastsuccessfullogin||oblastfailedlogin||obpasswordexpirydate||obver||
obLastLoginAttemptDate||oblockedon||obpsftid") (version 3.0; acl "oam
userWritePrivilegeGroup acl"; allow (search,read,compare,write)
groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>"; allow
(search,read,compare) groupdn="ldap:///
cn=orclFAUserReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>"; allow
(search,read,compare,write) groupdn="ldap:///
cn=orclFAUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetattr="orclguid||modifytimestamp") (version 3.0; acl "orclguid
acl";allow (read, search, compare) groupdn="ldap:///cn=Common User
Attributes, cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,compare) groupdn="ldap:///cn=oracledasedituser,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow (read) userdn="ldap:///
anyone";)
aci: (targetfilter="(objectclass=orcluser*)")(version 3.0; acl "add orcluser
aci";allow(read,add) groupdn="ldap:///
cn=oracledascreateuser,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";)
aci: (targetattr="displayName||preferredlanguage||orcltimezone||
orcldateofbirth||orclgender||orclwirelessaccountnumber||cn||uid||homephone||
telephonenumber") (version 3.0; acl "useraccount acl";allow (read, search,
compare) groupdn="ldap:///cn=Common User Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,compare) groupdn="ldap:///cn=oracledasedituser,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,selfwrite,compare) userdn="ldap:///self";allow (read)
userdn="ldap:///anyone";)
aci: (version 3.0; acl "read aci"; allow(read) groupdn="ldap:///cn=Common
User Attributes,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(read
groupdn="cn=PKIAdmins,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>");)
aci: (targetattr="*") (targetfilter="(objectclass=inetorgperson)") (version
3.0; acl "inetorgperson acl";allow (read,search,write,compare)
```

```
groupdn="ldap:///cn=oracledasedituser,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,selfwrite,compare) userdn="ldap:///self";allow (read)
userdn="ldap:///anyone";)
aci: (targetattr="orclaccountstatusevent") (version 3.0; acl
"orclaccountstatusevent acl";allow (write) groupdn="ldap:///
cn=verifierServices,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";)
aci: (targetattr="orclAccessibilityMode||orclColorContrast||orclFontSize||
orclNumberFormat||orclCurrency||orclDateFormat||orclTimeFormat||
orclEmbeddedHelp||orclFALanguage||orclFATerritory||orclTimeZone||
orclDisplayNameLanguagePreference||orclImpersonationGrantee||
orclImpersonationGranter") (targetfilter="(objectclass=inetorgperson)")
(version 3.0; acl "orclIDXPerson attributes acl";allow
(search,read,compare,write) groupdn="ldap:///
cn=orclFAUserWritePrefsPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>" ;)
aci: (targetattr!="orclpasswordverifier||orclpassword||authpassword||
pwdhistory||orclpwdaccountunlock||orclaccountstatusevent")(version 3.0; acl
"orclPwdPolicyAttributes acl";allow (search,read,compare) groupdn="ldap:///
cn=orclFAUserReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>" ;allow
(search,read,compare,write) groupdn="ldap:///
cn=orclFAUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>" ;)
aci: (targetattr="mail") (version 3.0; acl "orclaccountstatusevent acl";allow
(write) groupdn="ldap:///
cn=EmailAdminsGroup,cn=EmailServerContainer,cn=Products,cn=OracleContext";allo
w (read,search,write,compare) groupdn="ldap:///
cn=oracledasedituser,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read, search, compare) groupdn="ldap:///cn=Common User Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,compare) groupdn="ldap:///cn=oracledasedituser,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow (read) userdn="ldap:///
anyone";)
aci: (targetattr="orclpasswordhintanswer") (version 3.0; acl
"orclpasswordhintanswer acl";allow (read, search, compare) groupdn="ldap:///
cn=Common User Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,selfwrite,compare) userdn="ldap:///self";)
aci: (targetattr="orclpasswordhint") (version 3.0; acl "orclpasswordhint
acl";allow (read, search, compare) groupdn="ldap:///cn=Common User
Attributes, cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,selfwrite,compare) userdn="ldap:///self";allow
(read,search,write,compare) groupdn="ldap:///
cn=OracleUserSecurityAdmins,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";)
aci: (targetattr="userPassword") (targetfilter="(objectclass=inetorgperson)")
(version 3.0; acl "userpassword acl";allow (read,search,write,compare)
groupdn="ldap:///
cn=OracleUserSecurityAdmins,cn=Groups,cn=OracleContext<LDAP_SEARCHBASE>";allow
 (read,search,write,compare) groupdn="ldap:///cn=oracledasedituser,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,selfwrite,compare) userdn="ldap:///self";allow (compare)
groupdn="ldap:///cn=authenticationServices,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow (compare)
groupdn="ldap:///
cn=orclFAUserReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow
(read,search,write,compare) groupdn="ldap:///
cn=orclFAUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetattr="authpassword||orclpasswordverifier||orclpassword") (version
```

```
3.0; acl "orclpassword acl";allow (read,search,write,compare)
groupdn="ldap:///
cn=oracledasedituser,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>" ;allow
(search, read, compare) groupdn="ldap:///
cn=verifierServices,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(search,read,write,compare) userdn="ldap:///self";)
aci: (targetattr="usercertificate||usersmimecertificate") (version 3.0; acl
"usercertificate acl";allow (read, search, write, compare) groupdn="ldap:///
cn=PKIAdmins,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(read,
search, compare) userdn="ldap:///self";allow (read, search, compare)
userdn="ldap:///anyone";)
aci: (targetfilter="(|(objectclass=person)(objectclass=orclcontainer))")
(version 3.0; acl "person and orclcontainer acl";allow(search,read,add)
groupdn="ldap:///
cn=oracledascreateuser,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(sea
rch,read,delete) groupdn="ldap:///
cn=oracledasdeleteuser,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(sea
rch,read,write) groupdn="ldap:///
cn=oracledasedituser,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(searc
h,read,proxy) groupdn="ldap:///
cn=UserProxyPrivilege,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(sear
ch,read,proxy) userdn="ldap:///
orclApplicationCommonName=DASApp,cn=DAS,cn=Products,cn=oraclecontext";allow(re
ad,selfwrite) userdn="ldap:///self";allow(search,read) groupdn="ldap:///
cn=Common User Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(search,read)
groupdn="ldap:///
cn=orclFAUserReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow(search,read,wri
te,add,delete) groupdn="ldap:///
cn=orclFAUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetattr="orclisenabled") (version 3.0; acl "orclisenabled acl";allow
(read,search,write,compare) groupdn="ldap:///cn=oracledasaccountadmingroup,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow (read) userdn="ldap:///
anyone";allow (read, search, compare) groupdn="ldap:///cn=Common User
Attributes, cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow
(read,search,write,compare) groupdn="ldap:///cn=oracledasedituser,
cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow (read) userdn="ldap:///
anyone";)
aci: (targetattr = "*")(targetfilter= "(objectclass=inetorgperson)")
(targetscope = "subtree") (version 3.0; acl "iam admin changepwd"; allow
(compare,search,read,selfwrite,add,write,delete) userdn = "ldap:///
cn=<LDAP_OAMLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>";)

dn: <LDAP_GROUP_SEARCHBASE>
objectClass: orclContainer
objectClass: top
cn: groups
aci: (version 3.0; acl "fa acl";allow(search,read) groupdn="ldap:///
cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow(search,read,ad
d,delete) groupdn="ldap:///
cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetfilter="(&(objectclass=orclgroup)(!(orclisvisible=false)))")
(version 3.0; acl "visible orclgroup acl";allow(read,search,add)
groupdn="ldap:///
cn=oracledascreategroup,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(re
ad,search,delete) groupdn="ldap:///
```

```
cn=oracledasdeletegroup,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(re
ad,search,write) groupdn="ldap:///cn=oracledaseditgroup,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(search,read,add,delete)
userattr="owner#USERDN";allow(search,read,add,delete)
userattr="owner#GROUPDN";allow(search,read) groupdn="ldap:///cn=Common Group
Attributes,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(search,read,add
,delete) groupdn="ldap:///
cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (version 3.0; acl "orclgroup read acl";allow(search,read)
groupdn="ldap:///cn=Common Group Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>" ;)
aci: (targetattr="*") (targetfilter="(&(objectclass=orclgroup)(!
(orclisvisible=false)))")(version 3.0; acl "attrs for visible orclcontainer
acl";allow(search,read,write,compare)
userattr="owner#USERDN";allow(search,read,write,compare)
userattr="owner#GROUPDN";allow(search,read,write,compare) groupdn="ldap:///
cn=oracledaseditgroup,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE> || ldap:///
cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow(search,read,c
ompare) groupdn="ldap:///cn=Common Group Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE> || ldap:///
cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetfilter="(&(objectclass=orclgroup)(orclisvisible=false))")(version
3.0; acl "visible orclgroup acl";allow(search,read,add,delete)
userattr="owner#USERDN";allow(search,read,add,delete)
userattr="owner#GROUPDN";allow(search,read) groupdn="ldap:///cn=Common Group
Attributes,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE> || ldap:///
cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow(search,read,ad
d,delete) groupdn="ldap:///
cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetfilter="(objectclass=orclcontainer)")(version 3.0; acl
"orclcontainer add acl";allow(search,read,add) groupdn="ldap:///
cn=IASAdmins,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>" ;)
aci: (targetattr="*")(version 3.0; acl "attr fa
acl";allow(search,read,compare) groupdn="ldap:///
cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow(search,read,co
mpare,write) groupdn="ldap:///
cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetfilter="(objectclass=orclgroup*)")(version 3.0; acl "orclgroup
add acl";allow(search,read,add) groupdn="ldap:///
cn=oracledascreategroup,cn=groups,cn=OracleContext,<LDAP_SEARCHBASE>" ;)
aci: (targetattr="mail") (targetfilter="(objectclass=orclgroup)")(version
3.0; acl "mail attr for orclcontainer acl";allow(search,read,write,compare)
userattr="owner#USERDN";allow(search,read,write,compare)
userattr="owner#GROUPDN";allow(search,read,compare) groupdn="ldap:///
cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE> || ldap:///cn=Common
Group Attributes,
cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE>";allow(search,read,compare,write)
 groupdn="ldap:///cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
|| ldap:///
cn=EmailAdminsGroup,cn=EMailServerContainer,cn=Products,cn=OracleContext";)
aci: (targetattr="*") (targetfilter="(&(objectclass=orclgroup)
(orclisvisible=false))")(version 3.0; acl "attrs for non visible
orclcontainer acl";allow(search,read,write,compare)
userattr="owner#USERDN";allow(search,read,write,compare)
userattr="owner#GROUPDN";allow(search,read, compare) groupdn="ldap:///
cn=Common Group Attributes,cn=Groups,cn=OracleContext,<LDAP_SEARCHBASE> ||
```

```
ldap:///
cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";allow(search,read,co
mpare,write) groupdn="ldap:///
cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)

dn:
cn=FAPolicy,cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext,<LDAP_SEARCH
BASE>
pwdfailurecountinterval: 0
pwdlockoutduration: 86400
objectclass: top
objectclass: pwdpolicy
objectclass: ldapSubentry
pwdmaxfailure: 10
pwdminlength: 5
cn: FAPolicy
pwdlockout: true
pwdCheckQuality: 1
pwdGraceAuthNLimit: 5
pwdexpirewarning: 604800
pwdmaxage: 0
#displayname: Password Policy for Fusion Apps
pwdAttribute: userPassword

dn: cn=orclFAUserReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: orclFAUserReadPrivilegeGroup
uniquemember: cn=<LDAP_OAMLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>

dn: cn=orclFAUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: orclFAUserWritePrivilegeGroup

dn: cn=orclFAUserWritePrefsPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: orclFAUserWritePrefsPrivilegeGroup

dn: cn=orclFAGroupReadPrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: orclFAGroupReadPrivilegeGroup
uniquemember: cn=<LDAP_OAMLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>

dn: cn=orclFAGroupWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: orclFAGroupWritePrivilegeGroup
```

```
dn: cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>
objectClass: orclContainer
objectClass: top
cn: <LDAP_SYSTEMIDS>
ds-pwp-password-policy-dn:
cn=SystemIDPolicy,cn=pwdPolicies,cn=common,cn=products,cn=OracleContext,<LDAP_
SEARCHBASE>

dn: cn=<LDAP_WLSADMIN_GRP>,<LDAP_GROUP_SEARCHBASE>
objectClass: orclGroup
objectClass: groupOfUniqueNames
objectClass: orclIDXGroup
objectClass: top
description: WLS Administrators Group for the IDM Domain in LDAP
displayName: WLS Administrators
cn: <LDAP_WLSADMIN_GRP>
uniquemember: cn=<LDAP_OAMADMIN_GRP>,<LDAP_GROUP_SEARCHBASE>
uniquemember: cn=<LDAP_WLSADMIN_USER>,<LDAP_USER_SEARCHBASE>
uniquemember: cn=<LDAP_XELSYSADM_USER>,<LDAP_USER_SEARCHBASE>

dn: cn=<LDAP_WLSADMIN_USER>,<LDAP_USER_SEARCHBASE>
objectClass: orclUserV2
objectClass: person
objectClass: oblixorgperson
objectClass: organizationalPerson
objectClass: oblixPersonPwdPolicy
objectClass: inetOrgPerson
objectClass: orclAppIDUser
objectClass: orclUser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
givenName: <LDAP_WLSADMIN_USER>
obpasswordchangeflag: false
uid: <LDAP_WLSADMIN_USER>
orclIsEnabled: ENABLED
sn: <LDAP_WLSADMIN_USER>
userPassword: <PASSWORD>
mail: <LDAP_WLSADMIN_USER>@company.com
orclSAMAccountName: <LDAP_WLSADMIN_USER>
obpasswordexpirydate: <OUD_PWD_EXPIRY>T00:00:00Z
cn: <LDAP_WLSADMIN_USER>
oblogintrycount: 0

dn: cn=OblixAnonymous,<LDAP_SEARCHBASE>
objectClass: orcluserV2
objectClass: oblixOrgPerson
objectClass: person
objectClass: oblixPersonPwdPolicy
objectClass: inetorgperson
objectClass: organizationalPerson
objectClass: orcluser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
userPassword: <PASSWORD>
```

```
mail: OblixAnonymous@company.com
givenName: OblixAnonymous
orclSAMAccountName: OblixAnonymous
description: Anonymous user used by OAM
uid: OblixAnonymous
sn: OblixAnonymous
cn: OblixAnonymous

dn: cn=<LDAP_OAMADMIN_USER>,<LDAP_USER_SEARCHBASE>
objectClass: orclUserV2
objectClass: oblixorgperson
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: oblixPersonPwdPolicy
objectClass: orclAppIDUser
objectClass: orclUser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
givenName: <LDAP_OAMADMIN_USER>
uid: <LDAP_OAMADMIN_USER>
orclIsEnabled: ENABLED
sn: <LDAP_OAMADMIN_USER>
userPassword: <PASSWORD>
mail: <LDAP_OAMADMIN_USER>@company.com
orclSAMAccountName: <LDAP_OAMADMIN_USER>
cn: <LDAP_OAMADMIN_USER>
obpasswordchangeflag: false
obpasswordexpirydate: <OUD_PWD_EXPIRY>T00:00:00Z
ds-pwp-password-policy-dn:
cn=FAPolicy,cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext,<LDAP_SEARCH
BASE>

dn: cn=<LDAP_OAMLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>
objectClass: orclUserV2
objectClass: oblixorgperson
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: oblixPersonPwdPolicy
objectClass: orclAppIDUser
objectClass: orclUser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
userPassword: <PASSWORD>
mail: oamLDAP@company.com
givenName: oamLDAP
orclSAMAccountName: oamLDAP
uid: oamLDAP
sn: oamLDAP
cn: oamLDAP
ds-privilege-name: password-reset
ds-pwp-password-policy-dn:
cn=FAPolicy,cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext,<LDAP_SEARCH
```

```
BASE>

dn: cn=<LDAP_OAMADMIN_GRP>,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: <LDAP_OAMADMIN_GRP>
uniqueMember: cn=<LDAP_OAMADMIN_USER>,<LDAP_USER_SEARCHBASE>

dn: cn=OTPRestUserGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: top
objectClass: orclgroup
objectClass: groupofuniquenames
cn: OTPRestUserGroup
description: Forgotten Password Admin group
displayName: OTPRestUserGroup
uniquemember: cn=<LDAP_OAMADMIN_USER>,<LDAP_USER_SEARCHBASE>

dn: cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>
objectClass: orclGroup
objectClass: groupOfUniqueNames
objectClass: orclIDXGroup
objectClass: top
description: This is the role granted to have write permission on some User
Attributes
displayName: OAM User Modify Role
cn: orclFAOAMUserWritePrivilegeGroup
uniquemember: cn=<LDAP_OAMLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>

dn: ou=CO,<LDAP_SEARCHBASE>
objectClass: organizationalUnit
objectClass: top
ou: CO
aci: (targetfilter="(objectclass=*)")(version 3.0; acl "oam
userWritePrivilegeGroup entry acl"; allow (all) groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetattr="*")(version 3.0; acl "<DenySSORead ACI>"; deny
(read,search) (userdn!="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "<AllowSSORead ACI>"; allow
(read,search) (userdn="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "oam userWritePrivilegeGroup attribute
acl"; allow (all) groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)

dn: ou=People,<LDAP_SEARCHBASE>
objectClass: organizationalUnit
objectClass: top
ou: People
aci: (targetattr="*")(version 3.0; acl "<AllowSSOAll ACI>"; allow (all)
(userdn="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "<DenySSORead ACI>"; deny
(read,search) (userdn != "ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "<AllowSSORead ACI>"; allow
(read,search) (userdn="ldap:///all");)

dn: ou=vgoLocator,<LDAP_SEARCHBASE>
```

```
objectClass: organizationalUnit
objectClass: top
ou: vgoLocator
aci: (targetfilter="(objectclass=*)")(version 3.0; acl "oam
userWritePrivilegeGroup entry acl"; allow (all) groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetattr="*")(version 3.0; acl "<DenySSORead ACI>"; deny
(read,search) (userdn!="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "<AllowSSORead ACI>"; allow
(read,search) (userdn="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "oam userWritePrivilegeGroup attribute
acl"; allow (all) groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)

dn: cn=default,ou=vgoLocator,<LDAP_SEARCHBASE>
objectClass: top
objectClass: vgoLocatorClass
vGoLocatorAttribute: <LDAP_SEARCHBASE>
cn: default
aci: (targetfilter="(objectclass=*)")(version 3.0; acl "oam
userWritePrivilegeGroup entry acl"; allow (all) groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)
aci: (targetattr="*")(version 3.0; acl "<DenySSORead ACI>"; deny
(read,search) (userdn!="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "<AllowSSORead ACI>"; allow
(read,search) (userdn="ldap:///all");)
aci: (targetattr="*")(version 3.0; acl "oam userWritePrivilegeGroup attribute
acl"; allow (all) groupdn="ldap:///
cn=orclFAOAMUserWritePrivilegeGroup,<LDAP_GROUP_SEARCHBASE>";)

dn: cn=<LDAP_OIGLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>
objectClass: orclUserV2
objectClass: oblixorgperson
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: oblixPersonPwdPolicy
objectClass: orclAppIDUser
objectClass: orclUser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
userPassword: <PASSWORD>
mail: <LDAP_OIGLDAP_USER>@company.com
givenName: <LDAP_OIGLDAP_USER>
orclSAMAccountName: <LDAP_OIGLDAP_USER>
uid: <LDAP_OIGLDAP_USER>
sn: <LDAP_OIGLDAP_USER>
cn: <LDAP_OIGLDAP_USER>
ds-privilege-name: password-reset

dn: cn=<LDAP_OIGADMIN_GRP>,<LDAP_GROUP_SEARCHBASE>
objectClass: groupofUniqueNames
objectClass: orclIDXGroup
objectClass: top
cn: <LDAP_OIGADMIN_GRP>
```

```
uniquemember: cn=<LDAP_OIGLDAP_USER>,cn=<LDAP_SYSTEMIDS>,<LDAP_SEARCHBASE>

dn: <LDAP_RESERVE_SEARCHBASE>
objectClass: orclContainer
objectClass: top
cn: reserve
aci: (targetattr="*") (version 3.0; acl "oim admin group reserve container
acl"; allow (add,read,search,compare,write,delete,import,export)
groupdn="ldap:///cn=<LDAP_OIGADMIN_GRP>,<LDAP_GROUP_SEARCHBASE>" ;)

dn: cn=<LDAP_XELSYSADM_USER>,<LDAP_USER_SEARCHBASE>
objectClass: oblixorgperson
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: oblixPersonPwdPolicy
objectClass: orclAppIDUser
objectClass: orclIDXPerson
objectClass: top
objectClass: OIMPersonPwdPolicy
obpasswordchangeflag: false
givenName: <LDAP_XELSYSADM_USER>
orclIsEnabled: ENABLED
uid: <LDAP_XELSYSADM_USER>
sn: admin
userPassword: <PASSWORD>
mail: <LDAP_XELSYSADM_USER>@company.com
obuseraccountcontrol: activated
displayName: <LDAP_XELSYSADM_USER>
obpasswordexpirydate:<OUD_PWD_EXPIRY>T00:00:00Z
cn: <LDAP_XELSYSADM_USER>
oblogintrycount: 0
```

B

# Using an OCI Container Registry

The instructions here help you create a container registry in Oracle Cloud Infrastructure. However, you can use other registries as well. The staging examples will be similar for those registries as well.

To create a container registry and upload the container images to the repository, you should perform the following tasks:

1. Create a Container Registry
2. Obtain your software images and stage them locally on a host where docker or podman is installed.
3. Tag the images with the repository name.
4. Upload the images to the repository

This appendix includes the following topics:

- Creating a Container Registry
  You have to create the container registry in OCI. A container registry helps host the container images.
- Uploading Images to the Repository
  You can upload the container images by using Docker (with the `docker` command) or CRI-O (with the `podman` command).

## Creating a Container Registry

You have to create the container registry in OCI. A container registry helps host the container images.

To create a container registry:

1. Log in to the Oracle Cloud Infrastructure Console for your tenancy.
2. Select **Developer Services** and click **Container Registry**.
3. Click **Create Repository** and complete the following:
   a. Choose your compartment. For example: `iam`.
   b. Give your repository a name. This name is the same as the name of the container image. For example: `oracle/oam`.
   c. Choose whether anyone can access it or just those with the appropriate login credentials.
   d. Click **Create Repository**.

These steps create an empty repository. Repeat Step 3 for every image you want to upload.

> **Note:**
>
> While you do not have to pre-create empty repositories, doing so ensures that repositories get created in the correct compartment. If you omit this stage, you can move the repositories to your desired compartment later.

- Creating an Auth Token

## Creating an Auth Token

If you have not already done so, log in to the Oracle Cloud Infrastructure Console for your tenancy to create an auth token.

1. Click **Profile** on the top right corner of the page.
2. Click **User Settings**, select **Auth Tokens**, and then click **Generate Token**.
3. Enter a **Description**. For example: `Container_token`.
4. Click **Generate Token**.
5. Copy the generated token in a safe location because it will not be shown again.

# Uploading Images to the Repository

You can upload the container images by using Docker (with the `docker` command) or CRI-O (with the `podman` command).

- Using Docker to Load Images to the Container Registry
- Using CRI-O to Load Images to the Container Registry

## Using Docker to Load Images to the Container Registry

To load images using Docker, you should first log in to the container registory and then upload the images to the repository. For information about the new repository, see Creating a Container Registry.

- Staging the Docker Images Locally
- Logging in to the Container Registry
- Uploading the Docker Images to the Repository

### Staging the Docker Images Locally

If you have not already staged your images locally, you must do so before continuing. For instructions, see Staging Images in Docker .

### Logging in to the Container Registry

Before you can begin to upload images to the repository, you must first log in to your new repository.

1. To log in to the repository, use the following command:

   ```
   docker login repository_name
   ```

   The repository name will be `<region_code>.ocir.io`.

   For example, to log in to the Ashburn region, use the command:

   ```
   docker login iad.ocir.io
   ```

   For a list of region codes, see Availability by Region.

2. When prompted, provide the user name.

   The user name will be `<tenancy-namespace>/<username>` or `<tenancy-namespace>/oracleidentitycloudservice/<username>`. If you are using a federated logging, this user name will be the same user name you use to log in to your OCI console. For example: `mytenancy/oracleidentitycloudservice/my@emailaddress.com`.

3. When prompted for a password, provide the token you generated earlier. See Creating an Auth Token.

## Uploading the Docker Images to the Repository

To upload images to the repository, you must have the Docker images already installed locally.

1. To check the availability of the Docker images locally, use the following command:

   ```
   docker images
   ```

   For example, you may see the following:

   ```
   REPOSITORY          TAG                 IMAGE ID            CREATED
   SIZE
   oiri-ui             12.2.1.4.02106      0f6595fcf33d        3 weeks ago
   568MB
   ```

2. Tag the image with your new repository details using the image ID. To tag, use the command:

   ```
   docker tag <imageid> <repository_name>/<tenancy>/<project>/
   <image_name>:version
   ```

   For example:

   ```
   docker tag 0f6595fcf33d      iad.ocir.io/mytenancy/oracle/oiri-
   ui:12.2.1.4.02106
   ```

3. After tagging the image, you can upload it using the following command:

   ```
   docker push <repository_name>/<tenancy>/<project>/<image_name>:version
   ```

For example:

```
docker push   iad.ocir.io/mytenancy/oracle/oiri-ui:12.2.1.4.02106
```

This step updates the image to the container registry in your route compartment. To see the image, log in to your tenancy, select **Developer Services**, and then click **Container Registry**.

If you did not pre-create your empty repositories, you will find that the images have been stored in the root compartment. If you want to move the images to a specific compartment:

1. Select the repository name. For example: `oiri-ui`.
2. From the **Actions** menu, select **Move to Compartment**.
3. Select your destination compartment and click **OK**.

# Using CRI-O to Load Images to the Container Registry

To load images using CRI-O, you should first log in to the container registory and then upload the images to the repository. For information about the new repository, see Creating a Container Registry.

- Staging the CRI-O Images Locally
- Logging in to the Container Registry
- Uploading the CRI-O Images to the Repository
- Staging OAA Images Downloaded as a ZIP File

## Staging the CRI-O Images Locally

If you have not already staged your images locally, you should do so before continuing. For instructions, see Staging Images in CRI-O.

## Logging in to the Container Registry

Before you can begin uploading the images to the repository, you must first log in to the new repository.

1. To log in to the repository, use the following command:

   ```
   sudo podman login repository_name
   ```

   The repository name will be `<region_code>.ocir.io`.

   For example, to log in to the Ashburn region, use the command:

   ```
   sudo podman login iad.ocir.io
   ```

   For a list of region codes, see Availability by Region.

2. When prompted, provide the user name.

   The user name will be `<tenancy-namespace>/<username>` or `<tenancy-namespace>/oracleidentitycloudservice/<username>`. If you are using a federated logging, this user name will be the same user name you use to log in to your OCI console. For example: `mytenancy/oracleidentitycloudservice/my@emailaddress.com`.

3. When prompted for a password, provide the token you generated earlier. See Creating an Auth Token.

## Uploading the CRI-O Images to the Repository

To upload images to the repository, you must have the CRI-O images already installed locally.

1. To check the availability of the CRI-O images locally, use the following command:

   ```
   sudo podman images
   ```

   For example, you may see the following:

   ```
   REPOSITORY          TAG                 IMAGE ID          CREATED
   SIZE
   oiri-ui             12.2.1.4.02106      0f6595fcf33d      3 weeks ago
   568MB
   ```

2. Tag the image with your new repository details using the image ID. To tag, use the command:

   ```
   sudo podman tag <imageid> <repository_name>/<tenancy>/<project>/
   <image_name>:version
   ```

   For example:

   ```
   sudo podman tag 0f6595fcf33d      iad.ocir.io/mytenancy/oracle/oiri-
   ui:12.2.1.4.02106
   ```

3. After tagging the image, you can upload it using the following command:

   ```
   sudo podman push <repository_name>/<tenancy>/<project>/<image_name>:version
   ```

   For example:

   ```
   sudo podman push   iad.ocir.io/mytenancy/oracle/oiri-ui:12.2.1.4.02106
   ```

This step updates the image to the container registry in your route compartment. To see the image, log in to your tenancy, select **Developer Services**, and then click **Container Registry**.

If you did not pre-create your empty repositories, you will find that the images have been stored in the root compartment. If you want to move the images to a specific compartment:

1. Select the repository name. For example: `oiri-ui`.
2. From the **Actions** menu, select **Move to Compartment**.
3. Select your destination compartment and click **OK**.

## Staging OAA Images Downloaded as a ZIP File

If you are using Oracle Advanced Authentication and have downloaded the images as part of a zip file, you will have to stage the images locally as described in Staging Container Images.

Oracle Advanced Authentication provides a utility as part of the zip file to automatically upload all the staged images to your registry. This utility is called `pushImages.sh` and is in the `oaa-install` directory of the extracted zip file. To invoke the utility, use the following command:

```
pushImages.sh -b <RELEASE> -c <CONTAINER_COMMAND> -r <REGISTRY>
```

Where, `CONTAINER_COMMAND` is either the podman or the docker depending on your preferred command set.

For example:

```
pushImages.sh -b 12.2.1.4.1-20240425 -c podman -r iad.ocir.io/mytenancy/idm
```

> **Note:**
>
> You must ensure you are logged into your registry before issuing the command.

C

# Automating the Identity and Access Management Enterprise Deployment

A number of sample scripts have been developed which enable you to deploy Oracle Identity and Access Management on Kubernetes. These scripts are provided as samples for you to use to develop your own applications.

You must ensure that you are using the July 2022 or a later release of Identity and Access Management for this utility to work.

You can run the scripts from any host that has access to your Kubernetes cluster. If you want the scripts to automatically copy files to your Oracle HTTP Servers, you must have passwordless SSH set up from the deployment host to each of your web hosts.

If you are deploying Oracle Advanced Authentication, you must have passwordless SSH set up from the deployment host to one of your database nodes. In addition, for the duration of the deployment, your OAA database service must only be running on this database host.

This appendix includes the following topics:

- Obtaining the Scripts
  The automation scripts are available for download from GitHub.
- Scope of Scripts
  Learn about the actions that the scripts perform as part of the deployment process. There are also tasks that the scripts do not perform.
- Key Concepts of the Scripts
  To make things simple and easy to manage, the scripts include these files: a response file with details of the environment and template files you can easily modify or add as required.
- Directory Structure
  After you deploy the scripts, they will have a directory structure. In addition, while the scripts are working, they create a working directory.
- Getting Started
- Creating a Response File
- Validating Your Environment
- Provisioning the Environment
  There are a number of provisioning scripts located in the script directory. These scripts use a working directory defined in the response file for temporary files.
- Log Files
- Files You Need to Keep
- Archiving Files After Installation/Configuration
- Oracle HTTP Server Configuration Files
- Utilities
  In the `scripts` directory, there is a subdirectory called `utils`. This directory contains sample utilities you may find useful.

- Reference - Response File
  The parameters in the response file are used to control the provisioning of the various products in the Kubernetes cluster. These parameters are divided into generic and product-specific parameters.
- Components of the Deployment Scripts
  For reference purposes, this section includes the name and function of all the objects that make up the deployment scripts.

# Obtaining the Scripts

The automation scripts are available for download from GitHub.

To obtain the scripts, use the following command:

```
git clone https://github.com/oracle/fmw-kubernetes.git
```

The scripts appear in the following directory:

```
fmw-kubernetes/FMWKubernetesMAA/OracleEnterpriseDeploymentAutomation/
OracleIdentityManagement
```

Move these template scripts to your working directory. For example:

```
cp -R kubernetes/FMWKubernetesMAA/OracleEnterpriseDeploymentAutomation/
OracleIdentityManagement/* /workdir/scripts
```

If you are provisioning Oracle Identity Governance, you must also download the Oracle Connector Bundle for OUD and extract it to a location which is accessible by the provisioning scripts. For example, `/workdir/connectors/OID-12.2.1.3.0`. The connector directory name must start with `OID-12.2.1`.

If you are provisioning Oracle HTTP Server, you must download the Oracle HTTP installer and place it in the `$SCRIPTDIR/templates/ohs/installer` location. The installer must be the ZIP file. For example, `fmw_12.2.1.4.0_ohs_linux64_Disk1_1of1.zip`.

# Scope of Scripts

Learn about the actions that the scripts perform as part of the deployment process. There are also tasks that the scripts do not perform.

- What the Scripts Will do
- What the Scripts Will Not Do

## What the Scripts Will do

The scripts will deploy Oracle Unified Directory (OUD), Oracle Access Manager (OAM), and Oracle Identity Governance (OIG). They will integrate each of the products. You can choose to integrate one or more products.

The scripts perform the following actions:

- Create an Ingress controller as described in Creating the Ingress Controller.

- Create an Elasticsearch deployment as described in Installing Elasticsearch (ELK) Stack and Kibana.
- Create a Prometheus and Grafana deployment as described in Installing the Monitoring Software.
- Install and configure the Oracle HTTP Server. See Installing and Configuring Oracle HTTP Server.
- Deploy and configure Oracle WebGate.
- Create any number of OUD instances as described in Configuring Oracle Unified Directory.
- Extend OUD with OAM object classes as described in Creating the Schema Extensions File.
- Seed the directory with users and groups required by Oracle Identity and Access Management as described in Creating the Seeding File.
- Create indexes and ACI's in OUD as described in Creating OUD Containers.
- Set up replication agreements between different OUD instances.
- Create OUDSM.
- Setup Kubernetes Namespaces, Secrets, and Persistent Volumes.
- Create Kubernetes NodePort Services as required.
- Create the RCU schema objects for the product being installed.
- Deploy the WebLogic Kubernetes Operator as described in Installing the WebLogic Kubernetes Operator.
- Create an Oracle Access Manager Domain with user defined Managed Servers as described in Creating the Access Domain.
- Perform an initial configuration of the OAM domain as described in Updating the Domain.
- Perform post configuration tasks as described in Performing the Post-Configuration Tasks for Oracle Access Management Domain.
- Removing the OAM server from the default Coherence cluster.
- Tune the `oamDS` data source as described in #unique_1055.
- Configure the WebLogic Proxy Plug-in as described in #unique_1056.
- Configure and Integrate OAM with LDAP as described in Configuring and Integrating with LDAP.
- Add OAM Host Identifiers as described in Updating Host Identifiers.
- Add OAM policies as described in Adding the Missing Policies to OAM.
- Configure ADF and OPSS as described in Configuring Oracle ADF and OPSS Security with Oracle Access Manager.
- Set the initial server count as described in Setting the Initial Server Count.
- Create an Oracle Identity Governance Domain.
- Integrate OIG and SOA as described in Integrating Oracle Identity Governance with Oracle SOA Suite.
- Integrate OIG with LDAP as described in Integrating Oracle Identity Governance with LDAP.
- Add missing object classes as described in Adding Missing Object Classes.

- Integrate OIG and OAM as described in Integrating Oracle Identity Governance and Oracle Access Manager.
- Configure SSO integration in the Governance domain Configuring SSO Integration in the Governance Domain.
- Enable OAM Notifications as described in Enabling OAM Notifications.
- Update the Match Attribute as described in Updating the Value of MatchLDAPAttribute in oam-config.xml.
- Update the TAP Endpoint as described in Updating the TapEndpoint URL.
- Run Reconciliation Jobs as described in Running the Reconciliation Jobs .
- Configure Oracle Unified Messaging with Email/SMS server details as described in Managing the Notification Service.
- Enable Design Console access as described in Enabling Design Console Access.
- Add load balancer certificates to Oracle Key Store Service as described in Adding the Oracle Access Manager Load Balancer Certificate to the Oracle Keystore Service and Adding the Business Intelligence Load Balancer Certificate to Oracle Keystore Trust Service.
- Update the OIG initial server count as described in Setting the Initial Server Count.
- Set up the Business Intelligence Reporting links.
- Configure OIM to use BIP as described in Configuring Oracle Identity Manager to Use BI Publisher.
- Store the BI credentials in OIG as described in Storing the BI Credentials in Oracle Identity Governance.
- Deploy Oracle Identity Role Intelligence as described in Deploying OIRI Using Helm.
- Create OIRI users in Oracle Identity Governance as described in Creating User Names and Groups in Oracle Identity Governance.
- Perform an initial OIG data load into OIRI as described in Performing an Initial Data Load Using the Data Ingester.
- Create OIRI Kubernetes services either NodePort or Ingress as described in Creating the Kubernetes NodePort Services.
- Deploy Oracle Advanced Authentication and Risk Management as described in Deploying Oracle Advanced Authentication.
- Create OAA users as described in #unique_992.
- Create OAA test user as described in Creating a Test User.
- Integrate OAA with Unified Messaging Service as described in Configuring Email/SMS Servers and Automatic User Creation.
- Integrate OAA with OAM as described in #unique_1057.
- Copy the WebGate artifacts to Oracle HTTP Server as described in Copying WebGates Artifacts to Web Tier.
- Optionally send OUD, OUDSM, OAM, and OIG log files to Elasticsearch.

## What the Scripts Will Not Do

While the scripts perform the majority of the deployment, they do not perform the following tasks:

- Deploy the Container Runtime Environment, Kubernetes, or Helm.
- Configure the load balancer.
- Download the container images for these products.
- Tune the WebLogic Server.
- Configure the One Time Pin (OTP) forgotten password functionality for OAM.
- Configure the OIM workflow notifications to be sent by email.
- Set up the OIM challenge questions.
- Provision Business Intelligence Publisher (BIP).
- Set up the links to the Oracle BI Publisher environment. However, the scripts will deploy reports into the environment.
- Enable the BI certification reports in OIG as described in Enable Certification Reports.
- Configure Oracle HTTP Server to send log files and monitoring data to Elasticsearch and Prometheus.
- Configure Oracle Database Server to send log files and monitoring data to Elasticsearch and Prometheus.

# Key Concepts of the Scripts

To make things simple and easy to manage, the scripts include these files: a response file with details of the environment and template files you can easily modify or add as required.

> **Note:**
>
> Provisioning scripts are re-enterant. If something fails, you can restart the script from the point at which it failed.

# Directory Structure

After you deploy the scripts, they will have a directory structure. In addition, while the scripts are working, they create a working directory.

**Figure C-1 Directory Structure of the Scripts**

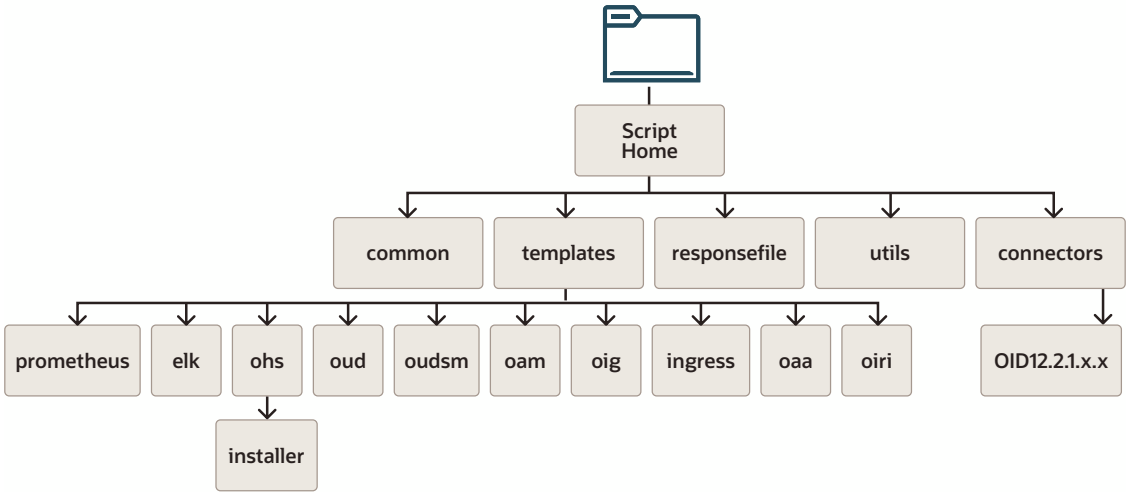
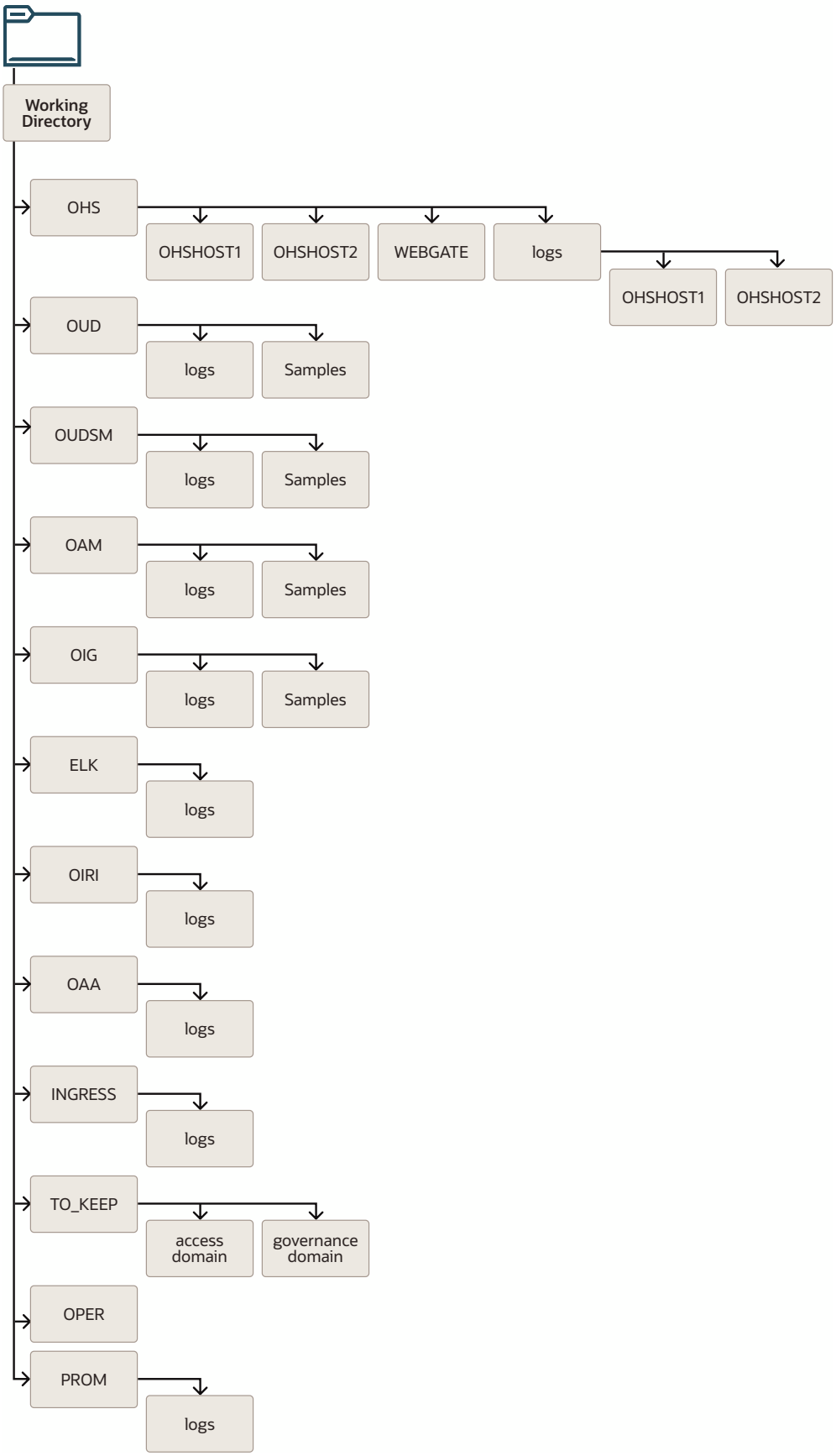


**Figure C-2 Working Directory of the Scripts**



# Getting Started

If you are provisioning Oracle Identity Governance, you must also download the Oracle Connector Bundle for OUD and extract it to a location which is accessible by the provisioning scripts. For example, `/workdir/connectors/OID-12.2.1.3.0`. The connector directory name must start with OID-12.2.1.

If you are provisioning the Oracle HTTP Server, you must download the Oracle HTTP installer and place it in the location `$SCRIPTDIR/templates/ohs/installer`. The installer must be the ZIP file. For example: `fmw_12.2.1.4.0_ohs_linux64_Disk1_1of1.zip`.

If you want to install the Oracle HTTP Server or copy files to it, you must set up passwordless SSH from the deployment host, during provisioning.

# Creating a Response File

A sample response and password file is created for you in the `responsefile` directory. You can edit these files either directly or by running the `start_here.sh` shell script in the script's home directory.

For example:

```
./start_here.sh [ -r responsefile -p passwordfile ]
```

You can run the above script as many times as you want on the same file. Pressing the **Enter** key on any response retains the existing value.

Values are stored in the `idm.rsp` and `.idmpwds` files unless the command is started with the `-r` and `-p` options, in which case the files updated will be that specified.

> **Note:**
>
> - The file consists of key/value pairs. There should be no spaces between the name of the key and its value. For example:
>   `Key=value`
> - If you are using complex passwords, that is, passwords which contain characters such as `!`, `*`, and `$`, then you should separate these characters by a `\`. For example: `hello!$` should be entered as `hello\!\$`. Complex passwords used for databases should be enclosed in quotes (""). For example: `"hello!$"`

# Validating Your Environment

You can run the `prereqchecks.sh` script, which exists in the script's home directory, to check your environment. This script is based on the response file you created earlier. See Creating a Response File.
The script performs several checks such as (but not limited to) the following:

- Ensures that the container images are available on each node.
- Checks that the NFS file shares have been created.

- Ensures that the load balancers are reachable.

To invoke the script use the following command:

```
cd <SCRIPTDIR>
./prereqchecks.sh [-r responsefile -p passwordfile]
```

Where, `-r` and `-p` are optional.

# Provisioning the Environment

There are a number of provisioning scripts located in the script directory. These scripts use a working directory defined in the response file for temporary files.

**Table C-1 Provisioning Scripts Located in the Script Directory**

| File | Purpose |
|---|---|
| `provision.sh` | Umbrella script that invokes each of the scripts (which can be invoked manually) mentioned in the following rows. |
| `provision_ingress.sh` | Deploys an Ingress controller. |
| `provision_elk.sh` | Deploys Elasticsearch and Kibana. |
| `provision_prom.sh` | Deploys Prometheus and Grafana. |
| `provision_ohs.sh` | Installs Oracle HTTP Server and deploys WebGate. |
| `provision_oud.sh` | Deploys Oracle Unified Directory. |
| `provision_oudsm.sh` | Deploys Oracle Unified Directory Services Manager. |
| `provision_operator.sh` | Deploys WebLogic Operator. |
| `provision_oam.sh` | Deploys Oracle Access Manager. |
| `provision_oig.sh` | Deploys Oracle Identity Governance. |
| `provision_oiri.sh` | Deploys Oracle Identity Role Intelligence. |
| `provision_oaa.sh` | Deploys Oracle Advanced Authentication. |

Each of the above commands can be provided with a specific response file (default is `idm.rsp`) and password file (default is `.idmpwds`), by appending:

```
-r responsefile -p passwordfile
```

These files must exist in the response file directory.

# Log Files

The provisioning scripts create log files for each product inside the working directory in a `logs` sub-directory.

This directory also contains the following two files:

- `progressfile` – This file contains the last successfully executed step. If you want to restart the process at a different step, update this file.
- `timings.log` – This file is used for informational purposes to show how much time was spent on each stage of the provisioning process.

# Files You Need to Keep

After a provisioning run that creates a domain is complete, there are files that you need to keep safely. These files are used to start and stop the domain as well as contain instructions to start the domain.

A copy of these files is stored in the working directory under the `TO_KEEP` subdirectory.
You should also keep any override files that are generated.

# Archiving Files After Installation/Configuration

As part of running the scripts, a number of working files are created in the `WORKDIR` directory prior to copying to the persistent volume in `/u01/user_projects/workdir`.

Many of these files contain passwords required for the setup. You should archive these files after completing the deployment.
The response file uses a hidden file in the `responsefile` directory to store passwords.

# Oracle HTTP Server Configuration Files

Each provisioning script creates sample files for configuring your Oracle HTTP Server. These files are generated and stored in the working directory under the `OHS` subdirectory. If required, the scripts can also copy these configuration files to Oracle HTTP server and restart it.

# Utilities

In the `scripts` directory, there is a subdirectory called `utils`. This directory contains sample utilities you may find useful.

These utilities are used for:

- Loading container images to each of the Kubernetes nodes.
- Deleting deployments.

# Reference - Response File

The parameters in the response file are used to control the provisioning of the various products in the Kubernetes cluster. These parameters are divided into generic and product-specific parameters.

- Products to Deploy
- Control Parameters
- Registry Parameters
- Image Parameters
- Generic Parameters

# Products to Deploy

These parameters determine which products the deployment scripts attempt to deploy.

**Table C-2 List of Products to Deploy**

| Parameter | Sample Value | Comments |
|---|---|---|
| INSTALL_INGRESS | true | Set the value to true to configure an Ingress controller. |
| INSTALL_ELK | false | Set the value to true to install and configure Elasticsearch and Kibana. |
| INSTALL_PROM | false | Set the value to true to install and configure Prometheus and Grafana. |
| INSTALL_OHS | true | Set the value to true to install the Oracle HTTP Server. |
| INSTALL_OUD | true | Set the value to true to configure OUD. |
| INSTALL_OUDSM | true | Set the value to true to configure OUDSM. |
| INSTALL_WLSOPER | true | Set the value to true to deploy Oracle WebLogic Operator. |
| INSTALL_OAM | true | Set the value to true to configure OAM. |
| INSTALL_OIG | true | Set the value to true to configure OIG. |
| INSTALL_OIRI | true | Set the value to true to configure OIRI. |
| INSTALL_OAA | true | Set the value to true to configure OAA. |
| INSTALL_RISK | true | Set the value to true to configure RISK. |
| INSTALL_OUA | true | Set the value to true to configure OUA. |

# Control Parameters

These parameters are used to specify the type of Kubernetes deployment and the names of the temporary directories you want the deployment to use, during the provisioning process.

**Table C-3 List of Control Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| USE_REGISTRY | false | Set to true to pull images from a container registry. |
| IMAGE_TYPE | crio | Set to crio or docker depending on your container engine. |
| IMAGE_DIR | /container/images | The location where you have downloaded the container images. Used by the load_images.sh script. |
| LOCAL_WORKDIR | /workdir | The location where you want to create the working directory. |
| K8_WORKDIR | /u01/oracle/user_projects/workdir | The location inside the Kubernetes containers to which working files are copied. |
| K8_WORKER_HOST1 | k8worker1.example.com | The name of a Kubernetes worker node used in generating the OHS sample files. |
| K8_WORKER_HOST2 | k8worker2.example.com | The name of a Kubernetes worker node used in generating the OHS sample files. |

# Registry Parameters

These parameters are used to determine whether or not you are using a container registry. If you are, then it allows you to store the login credentials to the repository so that you are able to store the credentials as registry secrets in the individual product namespaces.

If you are pulling images from GitHub or Docker hub, then you can also specify the login parameters here so that you can create the appropriate Kubernetes secrets.

**Table C-4 List of Registry Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| REGISTRY | iad.ocir.io/mytenancy | Set to the location of your container registry. |
| REG_USER | mytenancy/oracleidentitycloudservice/email@example.com | Set to your registry user name. |
| REG_PWD | *<password>* | Set to your registry password. |
| CREATE_REGSECRET | false | Set to true to create a registry secret for automatically pulling images. |

**Table C-4 (Cont.) List of Registry Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| CREATE_GITSECRET | true | Specify whether to create a secret for GitHub. This parameter ensures that you do not see errors relating to GitHub not allowing anonymous downloads. |
| GIT_USER | gituser | The GitHub user's name. |
| GIT_TOKEN | ghp_aO8fqRNVdfsfshOxsWk40uNMS | The GitHub token. |
| DH_USER | *username* | The Docker user name for hub.docker.com. Used for obtaining the public images. If you are hosting the public images in the registry, then specify the user name for that registry. |
| DH_PWD | *mypassword* | The Docker password for hub.docker.com. Used for obtaining the public images. If you are hosting the public images in the registry, then specify the user's password for that registry. |

# Image Parameters

These parameters are used to specify the names and versions of the container images you want to use for the deployment. These images must be available either locally or in your container registry. The names and versions must be identical to the images in the registry or the images stored locally.

These can include registry prefixes if you use a registry. Use the local/ prefix if you use the Oracle Cloud Native Environment.

**Table C-5 List of Image Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| OPER_IMAGE | ghcr.io/oracle/weblogic-kubernetes-operator | The WebLogic Operator image name. |
| OUD_IMAGE | $REGISTRY/oud | The OUD image name. |
| OUDSM_IMAGE | $REGISTRY/oudsm | The OUDSM image name. |
| OAM_IMAGE | $REGISTRY/oam | The OAM image name. |
| OIG_IMAGE | $REGISTRY/oig | The OIG image name. |
| OIRI_CLI_IMAGE | $REGISTRY/oiri-cli | The OIRI CLI image name. |
| OIRI_IMAGE | $REGISTRY/oiri | The OIRI image name. |
| OIRI_UI_IMAGE | $REGISTRY/oiri-ui | The OIRI UI image name. |
| OIRI_DING_IMAGE | $REGISTRY/oiri-ding | The OIRI DING image name. |

**Table C-5 (Cont.) List of Image Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_MGT_IMAGE | $REGISTRY/oracle/shared/oaa-mgmt | The OAA Management container image. |
| KUBECTL_REPO | bitnami/kubectl | The kubectl image used by OUD. |
| BUSYBOX_REPO | docker.io/busybox | The busybox image used by OUD. |
| PROM_REPO | - | If you are using your own container registry and have staged the Prometheus and Grafana images in this registry, then set this variable to the location of your registry. Leave it blank if you want to obtain the images from the public repositories. |
| ELK_REPO | - | If you are using your own container registry and have staged the Elastic Search and Kibana images in this registry, then set this variable to the location of your registry. Leave it blank if you want to obtain the images from the public repositories. |
| OPER_VER | 3.3.0 | The version of the WebLogic Operator. |
| OUD_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OUD version. |
| OUDSM_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OUDSM version. |
| OAM_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OAM version. |
| OIG_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OIG version. |
| OIRICLI_VER | 12.2.1.4-jdk8-ol7-<DATE> | OIRI CLI version. |
| OIRI_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OIRI version. |
| OIRIUI_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OIRI UI version. |
| OIRIDING_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OIRI DING version. |
| OAAMGT_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OAA MGMT container version. |
| OAA_VER | 12.2.1.4-jdk8-ol7-<DATE> | The OAA version. |

## Generic Parameters

These generic parameters apply to all deployments.

**Table C-6 Parameters to Control the Deployment of all Products**

| Parameter | Sample Value | Comments |
|---|---|---|
| PVSERVER | nfsserver.example.com | The name or IP address of the NFS server used for persistent volumes.<br>**Note**: If you use a name, then the name must be resolvable inside the Kubernetes cluster. If it is not resolvable, you can add it by updating CoreDNS. See Adding Individual Host Entries to CoreDNS. |
| IAM_PVS | /exports/IAMPVS | The export path on the NFS server where persistent volumes are located. |
| PV_MOUNT | /u01/oracle/user_projects | The path to mount the PV inside the Kubernetes container. Oracle recommends you to not change this value. |

# Ingress Parameters

These parameters determine how the Ingress controller is deployed.

**Table C-7 Ingress Parameters that Determine the Deployment of Ingress Controller**

| Parameter | Sample Value | Comments |
|---|---|---|
| INGRESSNS | ingressns | The Kubernetes namespace used to hold the Ingress objects. |
| INGRESS_TYPE | nginx | The type of Ingress controller you wan to deploy. At this time, The script supports only nginx. |
| INGRESS_ENABLE_TCP | true | Set to true if you want the controller to forward LDAP requests. |
| INGRESS_NAME | idmedg | The name of the Ingress controller used to create an Nginx Class. |
| INGRESS_SSL | false | Set to true if you want to configure the Ingress controller for SSL. |
| INGRESS_DOMAIN | example.com | Used when creating self-signed certificates for the Ingress controller. |
| INGRESS_REPLICAS | 2 | The number of Ingress controller replicas to start with. This value should be a minimum of two for high availability. |

# Elasticsearch Parameters

These parameters determine how to send log files to Elasticsearch.

**Table C-8 Parameters to Send Log Files to Elasticsearch**

| Parameter | Sample Value | Comments |
|---|---|---|
| USE_ELK | false | Set to true if you want to send log files to Elasticsearch. |
| ELKNS | elkns | The Kubernetes namespace used to hold Elasticsearch objects. |
| ELK_OPER_VER | 2.10.0 | The version of the Elastic Search operator you want to use. |
| ELK_VER | 8.11.0 | The version of Elasticsearch/ Logstash to use. |
| ELK_HOST | https://elasticsearch-es-http.<ELKNS>.svc:9200 | The address of the Elasticsearch server to which log files are to be sent. If you are using ELK inside a Kubernetes cluster, specify the address provided as the sample value. If you are using an Elasticsearch outside of the Kubernetes cluster, specify the external address. The host name you specify must be resolvable inside the Kubernetes cluster. |
| ELK_SHARE | /exports/IAMPVS/elkpv | The mount point on the NFS server where the ELK persistent volume is exported. |
| ELK_STORAGE | nfs-client | The storage class to use for the Elasticsearch stateful sets. |

# Prometheus Parameters

These parameters determine how to send monitoring information to Prometheus.

**Table C-9 Parameters for Sending Monitoring Information to Prometheus**

| Parameter | Sample Value | Comments |
|---|---|---|
| USE_PROM | false | Set to true if you want to send monitoring data to Prometheus. |
| PROMNS | monitoring | The Kubernetes namespace used to hold the Prometheus deployment. |

# OHS Parameters

OHS parameters are used to formulate how sample OHS configuration files are created. They also control whether you want the Oracle HTTP server files to be propagated to the Oracle

HTTP server hosts automatically. If you choose automatic propagation, you should ensure that a passwordless SSL is possible from the deployment host to the Oracle HTTP servers.

**Table C-10 Parameters Used by Oracle HTTP Server to Create Sample OHS Configuration Files**

| Parameter | Sample Value | Comments |
|---|---|---|
| UPDATE_OHS | true | Set this value to true if you want the scripts to automatically copy the generated OHS configuration files. After the files the copied, the Oracle HTTP Server restarts.<br>**Note**: This value is independent of whether you are installing the Oracle HTTP Server or not. |
| OHS_HOST1 | webhost1.example.com | The fully qualified name of the host running the first Oracle HTTP Server. |
| OHS_HOST2 | webhost2.example.com | The fully qualified name of the host running the second Oracle HTTP Server. Leave it blank if you do not have a second Oracle HTTP Server. |
| OHS_LBR_NETWORK | webtier.example.com | The Network subnets where the OHS health checks originate. Multiple entries must be enclosed in quotes and separated by space. |
| DEPLOY_WG | true | Deploys WebGate in OHS_ORACLE_HOME. |
| COPY_WG_FILES | true | Set this to true if you want the scripts to automatically copy the generated WebGate artifacts to the OHS Server.<br>**Note**: You should have first deployed the WebGate. |
| OHS_INSTALLER | fmw_12.2.1.4.0_ohs_linux64_Disk1_1of1.zip | The name of the OHS installer ZIP file. |
| OHS_BASE | /u02/private | The location of the OHS base directory. The binaries and the configuration files are below this location.<br>The Oracle inventory is also placed in this location when installing the Oracle HTTP Server. |
| OHS_ORACLE_HOME | $OHS_BASE/oracle/products/ohs | The location of the OHS binaries. |
| OHS_DOMAIN | $OHS_BASE/oracle/config/domains/ohsDomain | The location of the OHS domain on OHS_HOST1 and OHS_HOST2. |
| OHS1_NAME | ohs1 | The component name of the first OHS instance (on OHS_HOST1). |

**Table C-10 (Cont.) Parameters Used by Oracle HTTP Server to Create Sample OHS Configuration Files**

| Parameter | Sample Value | Comments |
|---|---|---|
| OHS2_NAME | ohs2 | The component name of the second OHS instance (on OHS_HOST2). |
| NM_ADMIN_USER | admin | The name of the admin user you want to assign to Node Manager if you install the Oracle HTTP Server. |
| OHS_PORT | 7777 | The port your Oracle HTTP Servers listen on. |
| OHS_HTTPS_PORT | 4443 | The SSL port on which Oracle HTTP Servers listen. |
| NM_PORT | 5556 | The port to use for Node Manager. |

# OUD Parameters

These parameters are specific to OUD. When deploying OUD, you also require the generic LDAP parameters.

**Table C-11 OUD Parameters that Determine the Deployment of Oracle Unified Directory**

| Parameter | Sample Value | Comments |
|---|---|---|
| OUDNS | oudns | The Kubernetes namespace used to hold the OUD objects. |
| OUD_SHARE | $IAM_PVS/oudpv | The mount point on the NFS server where the OUD persistent volume is exported. |
| OUD_CONFIG_SHARE | $IAM_PVS/oudconfigpv | The mount point on the NFS server where the OUD configuration persistent volume is exported. |
| OUD_LOCAL_SHARE | /nfs_volumes/oudconfigpv | The local directory where OUD_CONFIG_SHARE is mounted. Used to hold seed files. |
| OUD_LOCAL_PVSHARE | /nfs_volumes/oudpv | The local directory where OUD_SHARE is mounted. Used for deletion. |
| OUD_POD_PREFIX | edg | The prefix used for the OUD pods. |
| OUD_REPLICAS | 1 | The number of OUD replicas to create. If you require two OUD instances, set this to 1. This value is in addition to the primary instance. |

**Table C-11 (Cont.) OUD Parameters that Determine the Deployment of Oracle Unified Directory**

| Parameter | Sample Value | Comments |
|---|---|---|
| OUD_REGION | us | The OUD region to use should be the first part of the searchbase without the dc=. |
| LDAP_USER_PWD | <password1> | The password to assign to all users being created in LDAP.<br>**Note**: This value should have at least one capital letter, one number, and should be at least eight characters long. |
| OUD_PWD_EXPIRY | 2024-01-02 | The date when the user passwords you are creating expires. |
| OUD_CREATE_NODEPORT | true | Set to True if you want to create NodePort Services for OUD. These services are used to interact with OUD from outside of the Kubernetes cluster. |
| OUD_MAX_CPU | 1 | Maximum CPU cores allocated to the OUD containers. |
| OUD_CPU | 200m | Initial CPU units allocated to the OUD Pods where 1000m is equal to 1 CPU core. |
| OUD_MAX_MEMORY | 4Gi | Maximum amount of memory that an OUD container can consume. |
| OUD_MEMORY | 2Gi | Initial memory allocated to the OUD pods. |

# OUDSM Parameters

These parameters are used to control the way OUDSM is deployed.

**Table C-12 Parameters that Determine the Deployment of Oracle Directory Services Manager**

| Parameter | Sample Value | Comments |
|---|---|---|
| OUDSMNS | oudsmns | The Kubernetes namespace used to hold the OUDSM objects. |
| OUDSM_USER | weblogic | The name of the administration user you want to use for the WebLogic domain that is created when you install OUDSM. |
| OUDSM_PWD | <password> | The password you want to use for OUDSM_USER. |
| OUDSM_SHARE | $IAM_PVS/oudsmpv | The mount point on the NFS server where the OUDSM persistent volume is mounted. |

**Table C-12 (Cont.) Parameters that Determine the Deployment of Oracle Directory Services Manager**

| Parameter | Sample Value | Comments |
|---|---|---|
| OUDSM_LOCAL_SHARE | /nfs_volumes/oudsmpv | The local directory where OUDSM_SHARE is mounted. It is used by the deletion procedure. |
| OUDSM_INGRESS_HOST | oudsm.example.com | Used when you are using an Ingress controller. This name must resolve in DNS and point to one of the Kubernetes worker nodes or to the network load balancer entry for the Kubernetes workers. |

# LDAP Parameters

This table lists the parameters which are common to all LDAP type of deployments.

**Table C-13 Parameters for All LDAP Deployments**

| Parameter | Sample Value | Comments |
|---|---|---|
| LDAP_OAMADMIN_USER | oamadmin | The name of the user you want to create for the OAM administration tasks. |
| LDAP_ADMIN_USER | cn=oudadmin | The name of the OUD administrator user. |
| LDAP_ADMIN_PWD | <password> | The password you want to use for the OUD administrator user. |
| LDAP_SEARCHBASE | dc=example,dc=com | The OUD search base. |
| LDAP_GROUP_SEARCHBASE | cn=Groups,dc=example,dc=com | The search base where names of groups are stored in the LDAP directory. |
| LDAP_USER_SEARCHBASE | cn=Users,dc=example,dc=com | The search base where names of users are stored in the LDAP directory. |
| LDAP_RESERVE_SEARCHBASE | cn=Reserve,dc=example,dc=com | The search base where reservations are stored in the LDAP directory. |
| LDAP_SYSTEMIDS | systemids | The special directory tree inside the OUD search base to store system user names, which will not be managed through OIG. |
| LDAP_OIGADMIN_GRP | OIMAdministrators | The name of the group you want to use for the OIG administration tasks. |
| LDAP_OAMADMIN_GRP | OAMAdministrators | The name of the group you want to use for the OAM administration tasks. |

**Table C-13 (Cont.) Parameters for All LDAP Deployments**

| Parameter | Sample Value | Comments |
|---|---|---|
| LDAP_WLSADMIN_GRP | WLSAdministrators | The name of the group you want to use for the WebLogic administration tasks. |
| LDAP_OAMLDAP_USER | oamLDAP | The name of the user you want to use to connect the OAM domain to LDAP for user validation. |
| LDAP_OIGLDAP_USER | oimLDAP | The name of the user you want to use to connect the OIG domain to LDAP for integration. This user will have read/write access. |
| LDAP_WLSADMIN_USER | weblogic_iam | The name of a user you want to use for logging in to the WebLogic Administration Console and Fusion Middleware Control. |
| LDAP_XELSYSADM_USER | xelsysadm | The name of the user to administer OIG. |
| LDAP_USER_PWD | <userpassword> | The password to be assigned to all the LDAP user accounts. |
| LDAP_EXTERNAL_HOST | | Specify only if the LDAP host does not reside inside the current Kubernetes cluster. In this case, enter the host name where LDAP is running. |
| LDAP_EXTERNAL_PORT | | Specify only if the LDAP host does not reside inside the current Kubernetes cluster. In this case, enter the port on which LDAP is listening. |

# SSL Parameters

The deployment scripts create self-signed certificates. The parameters are used to determine what values will be added to those certificates.

**Table C-14 Parameters Used to Create Self-Signed Certificates**

| Parameter | Sample Value | Comments |
|---|---|---|
| SSL_COUNTRY | US | The abbreviation for the name of the country. |
| SSL_ORG | Example Company | The name of the organization. |
| SSL_CITY | City | The name of the city. |
| SSL_STATE | State | The name of the state. |

# WebLogic Operator for Kubernetes Parameters

These parameters determines how the Oracle WebLogic Operator is provisioned.

**Table C-15 Parameter that Determines the Deployment of Oracle WebLogic Operator for Kubernetes**

| Parameter | Sample Value | Comments |
|---|---|---|
| OPERNS | opns | The Kubernetes namespace used to hold the WebLogic Kubernetes Operator. |
| OPER_ACT | operadmin | The Kubernetes service account for use by the WebLogic Kubernetes Operator. |
| OPER_ENABLE_SECRET | false | Set to true while using your own Container Registry that requires authentication. |

# OAM Parameters

These parameters determine how OAM is deployed and configured.

**Table C-16 Parameters that Determine the Deployment of Oracle Access Manager**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAMNS | oamns | The Kubernetes namespace used to hold the OAM objects. |
| OAM_SHARE | $IAM_PVS/oampv | The mount point on the NFS server where the OAM persistent volume is exported. |
| OAM_LOCAL_SHARE | /nfs_volumes/oampv | The local directory where OAM_SHARE is mounted. It is used by the deletion procedure. |
| OAM_SERVER_COUNT | 5 | The number of OAM servers to configure. This value should be more than you expect to use. |
| OAM_SERVER_INITIAL | 2 | The number of OAM Managed Servers you want to start for normal running. You will need at least two servers for high availability. |
| OAM_DB_SCAN | dbscan.example.com | The database scan address to be used by the grid infrastructure. |
| OAM_DB_LISTENER | 1521 | The database listener port. |
| OAM_DB_SERVICE | iadedg.example.com | The database service that connects to the database you want to use for storing the OAM schemas. |
| OAM_DB_SYS_PWD | DBSysPassword | The SYS password of the OAM database. |
| OAM_RCU_PREFIX | IADEDG | The RCU prefix to use for the OAM schemas. |

**Table C-16 (Cont.) Parameters that Determine the Deployment of Oracle Access Manager**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAM_SCHEMA_PWD | SchemaPassword | The password to use for the OAM schemas that get created. If you are using special characters, you may need to escape them with a '\'. For example: 'Password\#'. |
| OAM_WEBLOGIC_USER | weblogic | The OAM WebLogic administration user name. |
| OAM_WEBLOGIC_PWD | <password1> | The password to be used for OAM_WEBLOGIC_USER. |
| OAM_DOMAIN_NAME | accessdomain | The name of the OAM domain you want to create. |
| OAM_LOGIN_LBR_HOST | login.example.com | The load balancer name for logging in to OAM. |
| OAM_LOGIN_LBR_PORT | 443 | The load balancer port to use for logging in to OAM. |
| OAM_LOGIN_LBR_PROTOCOL | https | The protocol of the load balancer port to use for logging in to OAM. |
| OAM_ADMIN_LBR_HOST | iadadmin.example.com | The load balancer name to use for accessing OAM administrative functions. |
| OAM_ADMIN_LBR_PORT | 80 | The load balancer port to use for accessing OAM administrative functions. |
| OAM_COOKIE_DOMAIN | .example.com | The OAM cookie domain is generally similar to the search base. Ensure that you have a '.' (dot) at the beginning. |
| OAM_OIG_INTEG | true | Set to true if OAM is integrated with OIG. |
| OAM_OAP_HOST | k8worker1.example.com | The name of one of the Kubernetes worker nodes used for OAP calls. |
| OAM_OAP_PORT | 5575 | The internal Kubernetes port used for OAM requests. |
| OAMSERVER_JAVA_PARAMS | "-Xms2048m -Xmx8192m" | The Java memory parameters to use for OAM Managed Servers. |
| OAM_CPU | 500m | Initial CPU units allocated to OUD pods where 1000m is equal to 1 CPU core. |
| OAM_MAX_CPU | 1 | Maximum CPU cores allocated to the OAM pods. |
| OAM_MEMORY | 2Gi | Initial memory allocated to OAM pods. |
| OAM_MAX_MEMORY | 8Gi | Maximum amount of that an OAM pods can consume. |

# OIG Parameters

These parameters determine how OIG is provisioned and configured.

**Table C-17 Parameters that determine the Deployment of Oracle Identity Governance**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIGNS | oigns | The Kubernetes namespace used to hold the OIG objects. |
| CONNECTOR_DIR | /workdir/OIG/connectors/ | The location on the file system where you have downloaded and extracted the OUD connector bundle. |
| OIG_SHARE | $IAM_PVS/oigpv | The mount point on the NFS server where the OIG persistent volume is exported. |
| OIG_LOCAL_SHARE | /local_volumes/oigpv | The local directory where OIG_SHARE is mounted. It is used by the deletion procedure. |
| OIG_SERVER_COUNT | 5 | The number of OIM/SOA servers to configure. This value should be more than you expect to use. |
| OIG_SERVER_INITIAL | 2 | The number of OIM/SOA Managed Servers you want to start for normal running. You will need at least two servers for high availability. |
| OIG_DOMAIN_NAME | governancedomain | The name of the OIG domain you want to create. |
| OIG_DB_SCAN | dbscan.example.com | The database scan address used by the grid infrastructure. |
| OIG_DB_LISTENER | 1521 | The database listener port. |
| OIG_DB_SERVICE | edgigd.example.com | The database service which connects to the database you want to use for storing the OIG schemas. |
| OIG_DB_SYS_PWD | MySysPassword | The SYS password of the OIG database. |
| OIG_RCU_PREFIX | IGDEDG | The RCU prefix to use for OIG schemas. |
| OIG_SCHEMA_PWD | MySchemPassword | The password to use for the OIG schemas that get created. If you are using special characters, you may need to escape them with a '\'. For example: 'Password\#'. |
| OIG_WEBLOGIC_USER | weblogic | The OIG WebLogic administration user. |
| OIG_WEBLOGIC_PWD | <password> | The password you want to use for OIG_WEBLOGIC_USER. |

**Table C-17 (Cont.) Parameters that determine the Deployment of Oracle Identity Governance**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIG_ADMIN_LBR_HOST | igdadmin.example.com | The load balancer name to use for accessing OIG administrative functions. |
| OIG_ADMIN_LBR_PORT | 80 | The load balancer port you use for accessing the OIG administrative functions. |
| OIG_LBR_HOST | prov.example.com | The load balancer name to use for accessing the OIG Identity Console. |
| OIG_LBR_PORT | 443 | The load balancer port to use for accessing the OIG Identity Console. |
| OIG_LBR_PROTOCOL | https | The load balancer protocol to use for accessing the OIG Identity Console. |
| OIG_LBR_INT_HOST | igdinternal.example.com | The load balancer name you will use for accessing OIG internal callbacks. |
| OIG_LBR_INT_PORT | 7777 | The load balancer port to use for accessing the OIG internal callbacks. |
| OIG_LBR_INT_PROTOCOL | http | The load balancer protocol to use for accessing OIG internal callbacks. |
| OIG_ENABLE_T3 | false | Set this value to `true` if you want to enable access to the Design Console. |
| OIG_BI_INTEG | true | Set to `true` to configure BIP integration. |
| OIG_BI_HOST | bi.example.com | The load balancer name you will use for accessing BI Publisher. |
| OIG_BI_PORT | 443 | The load balancer port you will use for accessing BI Publisher. |
| OIG_BI_PROTOCOL | https | The load balancer protocol you will use for accessing BI Publisher. |
| OIG_BI_USER | idm_report | The BI user name you want to use for running reports in the BI Publisher deployment. |
| OIG_BI_USER_PWD | BIPassword | The password of the `OIG_BI_USER`. |
| OIMSERVER_JAVA_PARAMS | "-Xms4096m -Xmx8192m" | The memory parameters to use for the `oim_servers`. |
| SOASERVER_JAVA_PARAMS | "-Xms4096 -XMx8192m" | The memory parameters to use for `soa_servers`. |

**Table C-17 (Cont.) Parameters that determine the Deployment of Oracle Identity Governance**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIG_EMAIL_CREATE | true | If set to true, OIG will be configured for email notifications. |
| OIG_EMAIL_SERVER | sendmail.example.com | The name of your SMTP email server. |
| OIG_EMAIL_PORT | 25 | The port of your SMTP server. The valid values are None or TLS. |
| OIG_EMAIL_SECURITY | None | The security mode of your SMTP server. |
| OIG_EMAIL_ADDRESS | myemail.example.com | The user name that is used to connect to the SMTP server, if one is required. |
| OIG_EMAIL_PWD | *<password>* | The password of your SMTP server. |
| OIG_EMAIL_FROM_ADDRESS | from@example.com | The 'From' email address used when emails are sent. |
| OIG_EMAIL_REPLY_ADDRESS | noreplies@example.com | The 'Reply' to email address of the emails that are sent. |

# OIRI Parameters

These parameters determine how OIRI is provisioned and configured.

**Table C-18 Parameters that Determine the Deployment of Oracle Identity Role Intelligence**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIRINS | oirins | The Kubernetes namespace used to hold the OIRI objects. |
| DINGNS | dingns | The Kubernetes namespace used to hold the OIRI DING objects. |
| OIRI_REPLICAS | noreplies@example.com | The number of OIRI servers to start the deployment. |
| OIRI_UI_REPLICAS | 2 | The number of OIRI UI Servers to start the deployment. |
| OIRI_SPARK_REPLICAS | 2 | The number of OIRI UI servers to start the deployment. |
| OIRI_SHARE | $IAM_PVS/oiripv | The mount point on the NFS server where the OIRI persistent volume is exported. |
| OIRI_LOCAL_SHARE | /nfs_volumes/oiripv | The local directory where OIRI_SHARE is mounted. It is used by the deletion procedure. |
| OIRI_SHARE_SIZE | 10Gi | The size of the OIRI persistent volume. |

**Table C-18 (Cont.) Parameters that Determine the Deployment of Oracle Identity Role Intelligence**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIRI_DING_SHARE | $IAM_PVS/dingpv | The mount point on the NFS server where the OIRI DING persistent volume is exported. |
| OIRI_DING_LOCAL_SHARE | /nfs_volumes/dingpv | The local directory where OIRI_DING_SHARE is mounted. It is used by the deletion procedure. |
| OIRI_DING_SHARE_SIZE | 10Gi | The size of the OIRI DING persistent volume. |
| OIRI_WORK_SHARE | $IAM_PVS/workpv | The mount point on the NFS server where the OIRI work persistent volume is exported. |
| OIRI_DB_SCAN | dbscan.example.com | The database SCAN address of the grid infrastructure. |
| OIRI_DB_LISTENER | 1521 | The database listener port. |
| OIRI_DB_SERVICE | edgoiri.example.com | The database service which connects to the database you want to use for storing the OIRI schemas. |
| OIRI_DB_SYS_PWD | MySysPassword | The SYS password of the OIRI database. |
| OIRI_RCU_PREFIX | oiriedg | The RCU prefix to use for the OIRI schemas. |
| OIRI_SCHEMA_PWD | MySchemPassword | The password to use for the OIRI schemas that get created. If you are using special characters, you may need to escape them with a '\'. For example: 'Password\#'. |
| OIRI_OIG_DB_SCAN | dbscan.example.com | The database SCAN address of the grid infrastructure for OIG Database. |
| OIRI_OIG_DB_LISTENER | 1521 | The OIG database listener port. |
| OIRI_OIG_DB_SERVICE | oigsvc.example.com | The database service which connects to the database you want to use for storing mining OIG schemas.\| |
| OIRI_CREATE_OHS | true | This value instructs the scripts to generate OHS entries for connecting to OIRI. You should set this to true unless you are configuring a standalone OIRI. |

**Table C-18 (Cont.) Parameters that Determine the Deployment of Oracle Identity Role Intelligence**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIRI_INGRESS_HOST | igdadmin.example.com | If you are creating a fully integrated deployment and want OIRI to be included in the OHS deployment, then this value should be set to the OIG Administration host name. For example: iagadmin.example.com.<br>If you are deploying OIRI standalone using Ingress to route requests, then set this value to the virtual hostname you want to use. For example: oiri.example.com. |
| OIRI_KEYSTORE_PWD | MyKeystore_Password100 | The password to use for the OIRI keystore. |
| OIRI_ENG_GROUP | OrclOIRIRoleEngineer | The name of the OIG OIRI group - DO NOT CHANGE. |
| OIRI_ENG_USER | oiri | The user to be created in OIG for UI login. |
| OIRI_ENG_PWD | MyPassword1 | The password of the OIRI_ENG_USER. |
| OIRI_SERVICE_USER | oirisvc | The user name for the OIG OIRI service account. |
| OIRI_SERVICE_PWD | MyPassword1 | The password for OIRI_SERVICE_USER. |
| OIRI_OIG_URL | http://$OIG_DOMAIN_NAME-cluster-oim-cluster.$OIGNS.svc.cluster.local:14000 | The URL to access OIG. If internal to the Kubernetes cluster, use the Kubernetes service name as shown in the sample value. If external, use the IGDINTERNAL URL. |
| OIRI_OIG_SERVER | t3://<OIG_DOMAIN_NAME>-oim-server1.oigns.svc.cluster.local:14000` | The T3 URL to access the OIG oim server (used to create users in OIG) |
| OIRI_LOAD_DATA | true | Set to true if you want to load data from the OIG database. |
| OIRI_OIG_XELSYSADM_USER | xelsysadm | Set to an OIM Administrator which is used to create users in OIG. |
| OIRI_OIG_USER_PWD | mypassword | Password of the OIRI_OIG_XELSYSADM_USER. |

**Table C-18 (Cont.) Parameters that Determine the Deployment of Oracle Identity Role Intelligence**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIRI_OIG_XELL_FILE | | If your OIG is not inside Kubernetes, you need to manually aquire the OIG rest certificate. See Obtaining the OIG SSL Certificate. Set this parameter to the location of that file. Leave it blank if OIG is in Kubernetes. |
| OIRI_CREATE_OIG_USER | true | Set to true to allow the automation scripts to create the OIRI users in OIG. |
| OIRI_SET_OIG_COMPLIANCE | true | Set to true to allow the automation scripts place OIG in compliance mode. |

# OAA Parameters

These parameters determine how OAA is provisioned and configured.

**Table C-19 Parameters that Determine the Deployment of Oracle Advanced Authentication and Risk Management**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAANS | oaans | The Kubernetes namespace used to hold the OAA objects. |
| OAACONS | coherence | The Kubernetes namespace used to hold the Coherence objects. |
| OAA_DEPLOYMENT | edg | A name for your OAA deployment. Do not use the name `oaa` because this is reserved for internal use. |
| OAA_DOMAIN | OAADomain | The name of the OAM OAuth domain you want to create. |
| OAA_VAULT_TYPE | file\|oci | The type of vault to use: file system or OCI. |
| OAA_CREATE_OHS | true | Set to `false` if you are installing OAA standalone front ended by Ingress. |
| OAA_CONFIG_SHARE | $IAM_PVS/oaaconfigpv | The mount point on the NFS server where the OAA configuration persistent volume is exported. |
| OAA_CRED_SHARE | $IAM_PVS/oaacredpv | The mount point on the NFS server where the OAA credentials persistent volume is exported. |

**Table C-19 (Cont.) Parameters that Determine the Deployment of Oracle Advanced Authentication and Risk Management**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_LOG_SHARE | $IAM_PVS/oaalogpv | The mount point on the NFS server where the OAA log files persistent volume is exported. |
| OAA_LOCAL_CONFIG_SHARE | /nfs_volumes/oaaconfigpv | The local directory where OAA_CONFIG_SHARE is mounted. It is used by the deletion procedure. |
| OAA_LOCAL_CRED_SHARE | /nfs_volumes/oaacredpv | The local directory where OAA_CRED_SHARE is mounted. It is used by the deletion procedure. |
| OAA_LOCAL_LOG_SHARE | /nfs_volumes/oaalogpv | The local directory where OAA_LOG_SHARE is mounted. It is used by the deletion procedure. |
| OAA_DB_SCAN | dbscan.example.com | The database SCAN address of the grid infrastructure. |
| OAA_DB_LISTENER | 1521 | The database listener port. |
| OAA_DB_SERVICE | oaa_s.example.com | The database service which connects to the database you want to use for storing the OAA schemas. |
| OAA_DB_SYS_PWD | MySysPassword | The SYS password of the OAA database. |
| OAA_RCU_PREFIX | OAAEDG | The prefix to use for the OAA schemas. |
| OAA_SCHEMA_PWD | MySchemPassword | The password to use for the OAA schemas that are created. If you are using special characters, you may need to escape them with a '\'. For example: 'Password#'. |
| OAA_DB_SID | iamdb11 | The SID of the database on the database server. |

- OAA Users/Groups/Passwords
- OAA File System Vault Parameters
- OAA OCI Vault Parameters
- Ingress Parameters
- Email Server Parameters
- Test User Parameters
- HA Parameters
- Resource Parameters

## OAA Users/Groups/Passwords

**Table C-20 User Names and Groups Used for OAA**

| Users/Groups | Example | Description |
|---|---|---|
| OAA_ADMIN_GROUP | OAA-Admin-Role | The OIG role to create for OAA administration functions. This group is created in OIG. |
| OAA_USER_GROUP | OAA-App-User | The group which will be assigned to the OAA users. This group is created in OIG. |
| OAA_ADMIN_USER | oaaadmin | The name of the user to use for OAA administration functions. This user name is created in OIG. |
| OAA_ADMIN_PWD | oaaAdminPassword | The password to be assigned to the OAA_ADMIN_USER. |
| OAA_KEYSTORE_PWD | oaapassword | The password to be used for OAA keystores. |
| OAA_OAUTH_PWD | oaapassword | The password to be used for OAA OAuth domain. |
| OAA_API_PWD | oaapassword | The password to be used for OAA API interactions. |
| OAA_POLICY_PWD | oaapassword | The password to be used for OAA policy interactions. |
| OAA_FACT_PWD | oaapassword | The password to be used for OAA keystores for factor interactions. |
| OAA_ADD_USERS_LDAP | true | Adds all existing LDAP users to the OAA_USER_GROUP LDAP group, allowing existing users to log in via OAA. |
| OAA_ADD_USERS_OUA_OBJ | true | Adds the OUA Object class to all existing users in LDAP thereby allowing all existing users to log in via Oracle Universal Authentication. |

## OAA File System Vault Parameters

**Table C-21 Parameters Used for File System Vault**

| Users/Groups | Example | Description |
|---|---|---|
| OAA_VAULT_SHARE | $IAM_PVS/oaavaultpv | The mount point on the NFS server where the OAA file vault persistent volume is exported. |
| OAA_LOCAL_VAULT_SHARE | /nfs_volumes/oaavaultpv | The local directory where OAA_VAULT_SHARE is mounted. It is used by the deletion procedure. |

**Table C-21 (Cont.) Parameters Used for File System Vault**

| Users/Groups | Example | Description |
|---|---|---|
| OAA_VAULT_PWD | oaapassword | The password to use for the file-based vault. |

## OAA OCI Vault Parameters

**Table C-22 Parameters Used for OAA OCI Vault**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_OCI_OPER | - | To obtain this value, encode the value of the API key that you downloaded at the time of creating the vault. See Creating a Vault. |
| OAA_OCI_TENANT | - | To obtain this value, log in to the OCI console, navigate to **Profile** and click **Tenancy**. Use the **OCID** value. |
| OAA_OCI_USER | - | To obtain this value, log in to the OCI console, navigate to **Profile** and click **Username**. Use the **OCID** value. |
| OAA_OCI_FP | - | To obtain this value, log in to the OCI console, navigate to **Profiles**, select **User Settings**, and then click **API Keys**. Use the value of the fingerprint for the API Key you created earlier. See Creating a Vault. |
| OAA_OCI_COMPARTMENT | - | To obtain this value, log in to the OCI console, navigate to **Identity and Security** and click **Compartments**. Select the compartment in which you created the vault and use the **OCID** value. |
| OAA_OCI_VAULT_ID | - | To obtain this value, log in to the OCI console, navigate to **Identity and Security** and select **Vault**. Click the vault you created earlier. See Creating a Vault. Use the **OCID** value. |
| OAA_OCI_KEY | - | To obtain this value, log in to the OCI console, navigate to **Identity and Security**, select **Vault**, and then click the vault you created earlier. See Creating a Vault. Click the key you created earlier. For example, **vaultkey**. Use the **OCID** value. |

## Ingress Parameters

**Table C-23 Parameters Used for the Deployment of Ingress**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_ADMIN_HOST | iadadmin.example.com | The virtual host used for administration operations. Unless you are using OAA in the standalone mode, set this value to the OAM admin virtual host. |
| OAA_RUNTIME_HOST | login.example.com | The virtual host used for OAA runtime operations. Unless you are using OAA in the standalone mode, set this vale to the OAM virtual host. |

## Email Server Parameters

**Table C-24 Parameters Used for the Email Server**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_EMAIL_SERVER | http://governancedomain-cluster-soa-cluster.oigns.svc.cluster.local:8001/ucs/messaging/webservice | The entry point of the Oracle Unified Messaging server. If the OIG domain is internal to the Kubernetes cluster, you can use the internal Kubernetes service name. For example: http://<OIG_DOMAIN_NAME>-cluster-soa-cluster.<OIGNS>.svc.cluster.local:8001/ucs/messaging/webservice.<br>If your UMS server is external to the Kubernetes cluster, you can use the URL you configured to access it. For example: http://igdinternal.example.com/ucs/messaging/webservice. |
| OAA_EMAIL_USER | weblogic | The user name you use to connect to the UMS server. |
| OAA_EMAIL_PWD | umspassword | The password you use to connect to the UMS server. |
| OAA_SMS_SERVER | http://$OIG_DOMAIN_NAME-cluster-soa-cluster.$OIGNS.svc.cluster.local:8001/ucs/messaging/webservice | The entry point of the Oracle Unified Messaging server you use to send SMS messages. This is usually the same as OAA_EMAIL_SERVER. |
| OAA_SMS_USER | weblogic | The user name you use to connect to the UMS server. |
| OAA_SMS_PWD | umspassword | The password you use to connect to the UMS server. |

## Test User Parameters

**Table C-25 Parameters Used for Creating a Test User**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_CREATE_TESTUSER | true | Set this value to `true` if you want the scripts to create a test user for OAA. |
| OAA_USER | oaauser | The name you want to assign to the test user. |
| OAA_USER_PWD | testpassword | The password you want to assign to the test user. |
| OAA_USER_EMAIL | test_user@example.com | The email address of the test user you are creating. |
| OAA_USER_POSTCODE | - | The post code/zip code of the test user you are creating. |

## HA Parameters

**Table C-26 Parameters Used for High Availability**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_REPLICAS | 2 | The number of OAA service pods to be created. For HA, the minimum number is two. |
| OAA_ADMIN_REPLICAS | 2 | The number of OAA administration pods to be created. For HA, the minimum number is two. |
| OAA_POLICY_REPLICAS | 2 | The number of OAA policy pods to be created. For HA, the minimum number is two. |
| OAA_SPUI_REPLICAS | 2 | The number of OAA SPUI service pods to be created. For HA, the minimum number is two. |
| OAA_TOTP_REPLICAS | 2 | The number of OAA TOTP service pods to be created. For HA, the minimum number is two. |
| OAA_YOTP_REPLICAS | 2 | The number of OAA YOTP service pods to be created. For HA, the minimum number is two. |
| OAA_FIDO_REPLICAS | 2 | The number of OAA FIDO service pods to be created. For HA, the minimum number is two. |
| OAA_EMAIL_REPLICAS | 2 | The number of OAA EMAIL service pods to be created. For HA, the minimum number is two. |

**Table C-26 (Cont.) Parameters Used for High Availability**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_SMS_REPLICAS | 2 | The number of OAA SMS service pods to be created. For HA, the minimum number is two. |
| OAA_PUSH_REPLICAS | 2 | The number of OAA PUSH service pods to be created. For HA, the minimum number is two. |
| OAA_RISK_REPLICAS | 2 | The number of OAA RISK service pods to be created. For HA, the minimum number is two. |
| OAA_RISKCC_REPLICAS | 2 | The number of OAA RISK CC service pods to be created. For HA, the minimum number is two. |
| OAA_DRSS_REPLICAS | 2 | The number of OUA service pods to be created. For HA, the minimum number is two. |

## Resource Parameters

**Table C-27 Resource Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_OAA_CPU | 200m | Initial CPU units allocated to OAA pod where 1000m is equal to 1 CPU core. |
| OAA_OAA_MEMORY | 1Gi | Initial Memory allocated to OAA pod. |
| OAA_ADMIN_CPU | 200m | Initial CPU units allocated to ADMIN pod where 1000m is equal to 1 CPU core. |
| OAA_ADMIN_MEMORY | 1Gi | Initial memory allocated to ADMIN pod. |
| OAA_POLICY_CPU | 200m | Initial CPU units allocated to POLICY pod where 1000m is equal to 1 CPU core. |
| OAA_POLICY_MEMORY | 1Gi | Initial memory allocated to POLICY pod. |
| OAA_SPUI_CPU | 200m | Initial CPU units allocated to a SPUI pod where 1000m is equal to 1 CPU core. |
| OAA_SPUI_MEMORY | 1Gi | Initial memory allocated to SPUI pod. |
| OAA_TOTP_CPU | 200m | Initial CPU units allocated to TOTP pod where 1000m is equal to 1 CPU core.\| |
| OAA_TOTP_MEMORY | 1Gi | Initial memory allocated to TOTP pod. |

**Table C-27 (Cont.) Resource Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_YOTP_CPU | 200m | Initial CPU units allocated to YOTP pod where 1000m is equal to 1 CPU core. |
| OAA_YOTP_MEMORY | 1Gi | Initial memory allocated to YOTP pod. |
| OAA_FIDO_CPU | 200m | Initial CPU units allocated to FIDO pod where 1000m is equal to 1 CPU core. |
| OAA_FIDO_MEMORY | 1Gi | Initial memory allocated to FIDO pod. |
| OAA_EMAIL_CPU | 200m | Initial CPU units allocated to EMAIL pod where 1000m is equal to 1 CPU core. |
| OAA_EMAIL_MEMORY | 1Gi | Initial memory allocated to EMAIL pod. |
| OAA_PUSH_CPU | 200m | Initial CPU units allocated to PUSH pod where 1000m is equal to 1 CPU core. |
| OAA_PUSH_MEMORY | 1Gi | Initial memory allocated to PUSH pod. |
| OAA_SMS_CPU | 200m | Initial CPU units allocated to SMS pod where 1000m is equal to 1 CPU core. |
| OAA_SMS_MEMORY | 1Gi | Initial memory allocated to SMS pod. |
| OAA_KBA_CPU | 200m | Initial CPU Units allocated to KBA pod where 1000m is equal to 1 CPU core. |
| OAA_KBA_MEMORY | 1Gi | Initial memory allocated to KBA pod. |
| OAA_RISK_CPU | 200m | Initial CPU units allocated to RISK pod where 1000m is equal to 1 CPU core. |
| OAA_RISK_MEMORY | 1Gi | Initial memory allocated to RISK pod. |
| OAA_RISKCC_CPU | 200m | Initial CPU units allocated to RISKCC pod where 1000m is equal to 1 CPU core. |
| OAA_RISKCC_MEMORY | 1Gi | Initial memory allocated to RISKCC pod. |
| OAA_DRSS_CPU | 200m | Initial CPU units allocated to DRSS pod where 1000m is equal to 1 CPU core. |
| OAA_DRSS_MEMORY | 1Gi | Initial memory allocated to DRSS pod. |

# Port Mappings

In some cases, you can specify your own ports. The scripts allow you to override the default values by setting these parameters.

**Table C-28 Parameters that Determine the Ports Used in the Deployment**

| Parameter | Sample Value | Comments |
|---|---|---|
| ELK_KIBANA_K8 | 31800 | The port to use for the Kibana requests.<br>**Note:** This value must be within the Kubernetes service port range. |
| ELK_K8 | 31920 | The port to use for the Elasticsearch requests.<br>**Note:** This value must be within the Kubernetes service port range. |
| PROM_GRAF_K8 | 30900 | The port to use for Grafana requests.<br>**Note:** This value must be within the Kubernetes service port range. |
| PROM_K8 | 30901 | The port to use for Prometheus requests.<br>**Note:** This value must be within the Kubernetes service port range. |
| PROM_ALERT_K8 | 30902 | The port to use for Alert Manager requests.<br>**Note:** This value must be within the Kubernetes service port range. |
| OUD_LDAP_K8 | 31389 | The port to use for OUD LDAP requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| OUD_LDAPS_K8 | 31636 | The port to use for OUD LDAPS requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| OUDSM_SERVICE_PORT | 30901 | The port to use for OUDSM requests.<br>**Note**: This value must be within the Kubernetes service port range. |

**Table C-28 (Cont.) Parameters that Determine the Ports Used in the Deployment**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAM_ADMIN_PORT | 7001 | The internal WebLogic administration port to use for the OAM domain. This port is available only in the Kubernetes cluster. |
| OAM_ADMIN_K8 | 30701 | The external port to use for the OAM Administration Server requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| OAM_OAM_K8 | 30410 | The external port to use for the OAM Managed Server requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| OAM_POLICY_K8 | 30510 | The external port to use for the OAM Policy server requests.<br>**Note**: This value must be within the Kubernetes service port range. |
| OAM_OAP_SERVICE_PORT | 30540 | The external port to use for the OAP server requests. This port is for legacy WebGates and is optional.<br>**Note**: This value must be within the Kubernetes service port range. |
| OIG_SOA_PORT_K8 | 30801 | The external port to use for the SOA Managed Server requests.<br>**Note**: This value must be in the Kubernetes service port range. |
| OAM_OAP_PORT | 5575 | The internal Kubernetes port used for OAM requests. |
| OIG_ADMIN_PORT | 7101 | The internal port used for the OIG WebLogic Administration Server. |
| OIG_ADMIN_K8 | 30711 | The external port to use for the OIG Administration Server requests.<br>**Note**: This value must be in the Kubernetes service port range. |
| OIG_OIM_PORT_K8 | 30140 | The external port to use for the OIM Managed Server requests.<br>**Note**: this must be in the Kubernetes service port range. |

**Table C-28 (Cont.) Parameters that Determine the Ports Used in the Deployment**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIG_OIM_T3_PORT_K8 | 30142 | The external port to use for the OIM Managed Server T3 requests.<br>**Note**: This value must be in the Kubernetes service port range. |
| OHS_PORT | 7777 | The HTTP Server listen address. |

# Components of the Deployment Scripts

For reference purposes, this section includes the name and function of all the objects that make up the deployment scripts.

**Table C-29 Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| idm.rsp | responsefile | The file that contains the details of the target environment. Must be updated for each deployment. |
| start_here.sh | | The file that populates the response file. |
| prereqchecks.sh | | The file that checks the environment prior to provisioning. |
| provision.sh | | The file that provisions everything. |
| provision_ingress.sh | | The file that installs/configures the Ingress controller. |
| provision_elk.sh | | The file that installs/configures Elastic Search and Kibana. |
| provision_prom.sh | | The file that installs/configures Prometheus and Grafana. |
| provision_oud.sh | | The file that installs/configures OUD. |
| provision_oudsm.sh | | The file that installs/configures OUDSM. |
| provision_oam.sh | | The file that installs/configures OAM. |
| provision_oig.sh | | The file that installs/configures OIG. |
| provision_oiri.sh | | The file that installs/configures OIRI. |
| provision_oaa.sh | | The file that installs/configures OAA. |
| ingress_functions.sh | common | The common functions/procedures used by the Ingress provisioning scripts. |
| functions.sh | common | The common functions/procedures used by all the provisioning scripts. |
| prom_functions.sh | common | The common functions/procedures used by the Prometheus provisioning scripts. |
| oud_functions.sh | common | The common functions/procedures used by the OUD and OUDSM provisioning scripts. |

**Table C-29 (Cont.) Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| oam_functions.sh | common | The common functions/procedures used by the OAM provisioning scripts. |
| oig_functions.sh | common | The common functions/procedures used by the OIG provisioning scripts. |
| oiri_functions.sh | common | The common functions/procedures used by the OIRI provisioning scripts. |
| oaa_functions.sh | common | The common functions/procedures used by the OAA provisioning scripts. |
| elk_cluster.yaml | templates/elk | The template file used to create an ELK cluster. |
| kibana.yaml | templates/elk | The template file used to create a Kibana deployment. |
| elk_nodeport.yaml | templates/elk | The template file used to create an ELK NodePort Service. |
| kibana_nodeport.yaml | templates/elk | The template file used to create a Kibana NodePort Service. |
| override_prom.yaml | templates/prometheus | The template file used to create a Prometheus deployment. |
| alert_nodeport.yaml | templates/prometheus | The template file used to create the Alert Manager NodePort Service. |
| grafana_nodeport.yaml | templates/prometheus | The template file used to create the Grafana NodePort Service. |
| prometheus_nodeport.yaml | templates/prometheus | The template file used to create the Prometheus NodePort Service. |
| base.ldif | templates/oud | The template file used to seed OUD with users and groups. |
| 99-user.ldif | templates/oud | The template file used to seed OUD schema changes. |
| oud_nodeport.yaml | templates/oud | The template used to create the OUD NodePort Services for Kubernetes. |
| override_oud.yaml | templates/oud | The OUD Helm override template file. |
| oudsm_nodeport.yaml | templates/oudsm | The template used to create the OUDSM NodePort Services for Kubernetes. |
| override_oudsm.yaml | templates/oudsm | The OUD Helm override template file. |
| add_admin_roles.py | templates/oam | The template used to add LDAP groups to the WebLogic administration role. |
| configoam.props | templates/oam | The template property file used to run `idmConfigTool - configOAM`. |
| runidmConfigTool.sh | templates/oam | The template file used to run `idmConfigTool` in container. |
| fix_gridlink.sh | templates/oam | The template file used to enable gridlink on data sources. |
| oamconfig_modify_template.xml | templates/oam | The template file used to perform the initial OAM setup. |

**Table C-29 (Cont.) Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| oam_nodeport.yaml | templates/oam | The template file used to create the OAM Managed Server NodePort Service. |
| oap_clusterip.yaml | templates/oam | The template file used to create the OAM OAP internal Managed Server NodePort Service. |
| oap_nodeport.yaml | templates/oam | the template file used to create the OAM OAP external Managed Server NodePort Service. |
| policy_nodeport.yaml | templates/oam | The template file used to create the Policy Manager Managed Server NodePort Service. |
| resource_list.txt | templates/oam | The list of resources to add to the OAM IAMSuite Resource list. |
| set_weblogic_plugin.py | templates/oam | The template file used to enable WebLogic plug-in in the domain. |
| create_wg.sh | templates/oam | The template file used to manually create the WebGate agent. |
| Webgate_IDM.xml | templates/oam | The template property file used to manually create the WebGate agent. |
| config_adf_security.py | templates/oam | The template file used to create the SSO partner application. |
| oamDomain.sedfile | templates/oam | The template Sedfile used to exit domain.yaml. |
| oamSetUserOverrides.sh | templates/oam | The setUserOverrides.sh template file. |
| remove_coherence.py | templates/oam | The template file used to remove OAM from the default coherence cluster. |
| update_oamds.py | templates/oam | The template file used to update the OAMDS data source. |
| login_vh.conf | templates/oam | The template file used to create the sample OHS Config. |
| iadadmin_vh.conf | templates/oam | The template file used to create the sample OHS Config. |
| create_admin_roles.py | templates/oig | The template file used to assign LDAP groups to the WebLogic administration role. |
| create_oud_authenticator.py | templates/oig | The template file used to create the OUD authenticator. |
| get_passphrase.sh | templates/oig | The template file used to obtain the global passphrase from OAM. |
| get_passphrase.py | templates/oig | The template file used to obtain the OAM global passphrase. |
| oam_integration.sh | templates/oig | The template file used to run OIGOAMIntegration.sh -configureSSOIntegration. |

**Table C-29 (Cont.) Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| oigSetUserOverrides.sh | templates/oig | The setUserOverrides.sh template file. |
| soa_nodeport.yaml | templates/oig | The template file used to create the SOA external Managed Server NodePort Service. |
| oim_nodeport.yaml | templates/oig | The template file used to create the OIM external Managed Server NodePort Service. |
| add_object_classes.sh | templates/oig | The template file used to run IGOAMIntegration.sh -addMissingObjectClasses. |
| createWLSAuthenticators.sh | templates/oig | The template file used to run OIGOAMIntegration.sh -configureWLSAuthnProviders. |
| oam_notifications.sh | templates/oig | The template file used to run OIGOAMIntegration.sh -enableOAMSessionDeletion. |
| config_connector.sh | templates/oig | The template file used to run OIGOAMIntegration.sh -configureLDAPConnector. |
| create_oim_auth.sh | templates/oig | The template file used to run OIGOAMIntegration.sh -configureWLSAuthnProviders. |
| runJob.sh | templates/oig | The template shell script used to run the reconciliation jobs. |
| runJob.java | templates/oig | The Java script used to run the reconciliation jobs. |
| lib | templates/oig | The OIM libraries required by runJob.java. |
| update_soa.py | templates/oig | The template script used to update SOA URLs. |
| oamoig.sedfile | templates/oig | The Sedfile used to create the OIGOAMIntegration property files. |
| autn.sedfile | templates/oig | The supplementary Sedfile used to create the OIGOAMIntegration property files. |
| create_oigoam_files.sh | templates/oig | The template script used to generate the OIGOAMIntegration property files. |
| fix_gridlink.sh | templates/oig | The template script used to enable gridlink on data sources. |
| update_match_attr.sh | templates/oig | The template script used to update the Match attribute. |
| oigDomain.sedfile | templates/oig | The template script used to update the domain_soa_oim.yaml file. |
| update_mds.py | templates/oig | The template file used to update the MDS data source. |

**Table C-29 (Cont.) Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| set_weblogic_plugin.py | templates/oig | The template file used to set the WebLogic plug-in. |
| update_bi.py | templates/oig | The template file used to enable BI integration. |
| igdinternal_vh.conf | templates/oig | The template file used to create the sample OHS Config. |
| igdadmin_vh.conf | templates/oig | The template file used to create the sample OHS Config. |
| prov_vh.conf | templates/oig | The template file used to create the sample OHS Config. |
| createAdminUser.java | templates/oiri | The template file used to create the OIRI user names in OIG. |
| createAdminUser.sh | templates/oiri | The template file used to create, compile, and run createAdminUser.java. |
| setCompliance.java | templates/oiri | The template file used to place OIG into the Compliance Mode. |
| setCompliance.sh | templates/oiri | The template file used to create, compile, and run setCompliance.java. |
| oiri-cli.yaml | templates/oiri | The template file used to start the OIRI CLI. |
| ding-cli.yaml | templates/oiri | The template file used to start the OIRI DING CLI. |
| oiri_nodeport.yaml | templates/oiri | The template file used to create the OIRI NodePort Service. |
| oiriui_nodeport.yaml | templates/oiri | The template file used to create the OIRI UI NodePort Service. |
| ohs1.conf | templates/oiri | The template file used to create the sample OHS Config. |
| ohs2.conf | templates/oiri | The template file used to create the sample OHS Configuration. |
| create_auth_module.xml | templates/oaa | The template file used to create the OAM authentication module. |
| create_auth_policy.json | templates/oaa | The template file used to create the OAM authentication policy. |
| create_auth_scheme.xml | templates/oaa | The template file used to create the OAM authentication scheme. |
| create_ohs_wallet.sh | templates/oaa | The template file used to create an Oracle HTTP server wallet. |
| create_schemas.sh | templates/oaa | The template file used to create the OAA schemas. |
| delete_schemas | templates/oaa | The template file used to delete OAA schemas. |

**Table C-29 (Cont.) Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| create_tap_partner.py | templates/oaa | The template file used to create the OAM TAP partner. |
| enable_oauth.xml | templates/oaa | The template file used to enable OAM OAuth. |
| helmconfig | templates/oaa | The template file for `helm`. |
| oaa-mgmt-oci.yaml | templates/oaa | The template file used to create the OAA management pod when using the OCI vault. |
| oaa-mgmt-vfsi.yaml | templates/oaa | The template file used to create the OAA management pod when using the file system vault. |
| oaaoverride.yaml | templates/oaa | The template file used for `helm` to set pod counts. |
| ohs_admin.conf | templates/oaa | The template file used for OAA administration OHS entries. |
| ohs_login.conf | templates/oaa | The template file used for OAA runtime OHS entries. |
| ohs_header.conf | templates/oaa | The template file used for OAM OAuth header entries for OHS. |
| oud_add_existing_users.sh | templates/oaa | The template file used to add existing user names to the OAA LDAP group. |
| oud_add_users.sh | templates/oaa | The template file used to create OAA users and groups. |
| users.ldif | templates/oaa | The template file used to create the OAA users and groups in LDAP. |
| oud_test_user.sh | templates/oaa | The template file used to create the OAA test user. |
| test_user.ldif | templates/oaa | The template file used to create the OAA test user in LDAP. |
| delete_all.sh | utils | The file used to delete all the deployments. |
| delete_elk.sh | utils | The file used to delete the Elastic Search deployment. |
| delete_prom.sh | utils | The file used to delete the Prometheus deployment. |
| delete_image.sh | utils | The file used to delete a container image from the Kubernetes worker hosts. |
| delete_oam.sh | utils | The file used to delete the OAM deployment. |
| delete_oig.sh | utils | The file used to delete the OIG deployment. |
| delete_operator.sh | utils | The file used to delete the WLS Kubernetes Operator deployment. |
| delete_oud.sh | utils | The file used to delete the OUD deployment. |

**Table C-29 (Cont.) Components of the Deployment Scripts**

| Name | Location | Function |
|---|---|---|
| `delete_oudsm.sh` | `utils` | The file used to delete the OUDSM deployment. |
| `delete_oiri.sh` | `utils` | The file used to delete the OIRI deployment. |
| `delete_oaa.sh` | `utils` | The file used to delete the OAA deployment. |
| `delete_ingress.sh` | `utils` | The file used to delete the Ingress controller. |
| `load_images.sh` | `utils` | The file used to load the container image onto each Kubernetes worker host. |

D

# Cleaning Up After a Failed Installation

This appendix describes the procedure to remove a deployed or a partially deployed Oracle Identity and Access Management software from a Kubernetes cluster.

The steps provided in this section assume that you have the persistent volumes mounted on the configuration node under `/nfs_volumes/<pvname>`.

This appendix includes the following topics:



## Oracle Unified Directory Services Manager

To remove Oracle Unified Directory Services Manager (OUDSM):

1. Remove the OUDSM deployment using the command:

   ```
   helm uninstall -n <OUDNS> oudsm-1
   ```

   For example:

   ```
   helm uninstall -n ounds oudsm-1
   ```

2. If you have set up a node port service, remove the service using the command:

   ```
   kubectl delete service -n <OUDNS> oudsm-nodeport
   ```

   For example:

   ```
   kubectl delete service -n oudns oudsm-nodeport
   ```

3. If you have installed Logstash, you also need to execute the following commands:

```
kubectl delete deployment -n <OUDNS> oudsm-logstash
```

```
kubectl delete cm -n <OUDNS> oudsm-logstash-configmap
```

4. Verify there are no OUSDM services running, using the command:

```
kubectl get all -n <OUDNS> -o wide
```

For example:

```
kubectl get all -n oudns -o wide
```

5. Remove the contents of the persistent volume using the command:

```
rm -rf /nfs_volumes/oudsmpv/*
```

6. Remove the files from the working directory.

```
rm -f /workdir/OUDSM
```

For the list of applicable variables, see Variables Used in this Chapter.

# Oracle Unified Directory

If you have deployed OUDSM into the Oracle Unified Directory (OUD) namespace, you should first remove this component. See Oracle Unified Directory Services Manager.

1. Remove the OUD deployment using the command:

```
helm uninstall -n <OUDNS> <OUD_POD_PREFIX>
```

For example:

```
helm uninstall -n oudns edg
```

2. If you have set up a node port service, remove the service using the command:

```
kubectl delete service -n <OUDNS> oud-nodeport
```

For example:

```
kubectl delete service -n oudns oud-nodeport
```

3. If you have installed Logstash, you also need to execute the following commands:

```
kubectl delete deployment -n <OUDNS> oud-logstash
```

```
kubectl delete cm -n <OUDNS> oud-logstash-configmap
```

```
kubectl delete cm -n <OUDNS> elk-cert
```

4. Remove the OUD namespace.

```
kubectl delete namespace -n <OUDNS>
```

For example:

```
kubectl delete namespace -n oudns
```

5. Check that all OUD pods have stopped, using the command:

```
kubectl get all -n <OUDNS> -o wide
```

For example:

```
kubectl get all -n oudns -o wide
```

6. Remove the files on persistent volume.

```
rm -rf /nfs_volumes/oudpv
```

7. Optionally, remove the files on the configuration persistent volume.

```
rm -rf /nfs_volumes/oudconfigpv
```

8. Remove the files from the working directory.

```
rm -f /workdir/OUD
```

For the list of applicable variables, see Variables Used in this Chapter.

# Oracle Access Manager

To remove Oracle Access Manager:

1. Delete the domain creation job using the command:

```
kubectl delete jobs <OAM_DOMAIN_NAME>-create-fmw-infra-sample-domain-job -
n <OAMNS>
```

For example:

```
kubectl delete jobs accessdomain-create-fmw-infra-sample-domain-job -n
oamns
```

2. Delete the domain using the command:

```
kubectl delete domain <OAM_DOMAIN_NAME> -n <OAMNS>
```

For example:

```
kubectl delete domain accessdomain -n oamns
```

3. Delete the configmap using the command:

```
kubectl delete configmaps <OAM_DOMAIN_NAME>-create-oam-infra-domain-job-cm
-n <OAMNS>
```

For example:

```
kubectl delete configmaps accessdomain-create-oam-infra-domain-job-cm -n
oamns
```

4. If you have installed Logstash, you also need to execute the following commands:

```
kubectl delete deployment -n <OAMNS> oam-logstash


kubectl delete cm -n <OAMNS> oam-logstash-configmap


kubectl delete cm -n <OAMNS> elk-cert
```

5. Delete the NodePort Services if you have defined them, using the commands:

```
kubectl delete service -n <OAMNS> <OAM_DB_SERVICE>
```

For example:

```
kubectl delete service -n oamns accessdomain-oap


kubectl delete service -n oamns accessdomain-policy-nodeport


kubectl delete service -n oamns accessdomain-oam-nodeport
```

6. Check that the domain has completely stopped, using the command:

```
kubectl get all -n <OAMNS> -o wide
```

For example:

```
kubectl get all -n oamns -o wide
```

7. Creating a Helper Pod

   Create a helper pod inside the namespace to allow the running of the Repository Creation Utility (RCU).

   ```
   kubectl run helper --image <OAM_REPOSITORY>:<OAM_VER> -n <OAMNS> -- sleep
   infinity
   ```

   For example:

   ```
   kubectl run helper --image oracle/oam:latest -n oamns -- sleep infinity
   ```

   Or

   ```
   kubectl run helper -n oamns --image iad.ocir.io/mytenancy/oracle/
   oam:latest --overrides='{ "spec": { "imagePullSecrets": [{"name":
   "regcred"}] } }' -- sleep infinity
   ```

   The output appears as follows:

   ```
   pod/helper created
   ```

   Run the following command to check that the pod is running:

   ```
   kubectl get pods -n oamns
   ```

   The output appears as follows:

   ```
   NAME    READY    STATUS    RESTARTS    AGE
   helper  1/1      Running   0           8s
   ```

8. Drop the OAM schemas by using the commands:

   a. Start a bash shell in the helper pod using the following command:

      ```
      kubectl exec -it helper -n <OAMNS> -- /bin/bash
      ```

      For example:

      ```
      kubectl exec -it helper -n oamns -- /bin/bash
      ```

   b. Set up the environment variables which you will use to connect to the database.

      ```
      export DB_HOST=<OAM_DB_SCAN>
      export DB_PORT=<OAM_DB_LISTENER>
      export DB_SERVICE=<OAM_DB_SERVICE>
      export CONNECTION_STRING=$DB_HOST:$DB_PORT/$DB_SERVICE
      export RCU_PREFIX=<OIG_RCU_PREFIX>
      ```

```
export RCU_SCHEMA_PWD=<OAM_SCHEMA_PWD>
echo -e <OAM_DB_SYS_PWD>"\n"<OAM_SCHEMA_PWD> /tmp/pwd.txt
```

For example:

```
export DB_HOST=DBSCAN.example.com
export DB_PORT=1521
export DB_SERVICE=igdedg.example.com
export CONNECTION_STRING=$DB_HOST:$DB_PORT/$DB_SERVICE
export RCU_PREFIX=IGDEDG
export RCU_SCHEMA_PWD=rcupassword
echo -e syspassword"\n"rcupassword > /tmp/pwd.txt
```

Check that there are two entries in the `/tmp/pwd` file.

For example:

```
cat /tmp/pwd.txt


syspassword
rcupassword
```

c. Run the following command to create the RCU schemas in the database:

```
/u01/oracle/oracle_common/bin/rcu -silent -dropRepository -databaseType
ORACLE -connectString \
$CONNECTION_STRING -dbUser sys -dbRole sysdba \
-selectDependentsForComponents true -schemaPrefix $RCU_PREFIX -
component MDS -component IAU \
-component IAU_APPEND -component IAU_VIEWER -component OPSS -component
WLS -component STB -component OAM -f < /tmp/pwd.txt
```

9. Remove persistent volume and claim from Kubernetes using the commands:

```
kubectl delete pvc -n <OAMNS> <OAM_DOMAIN_NAME>-domain-pvc


kubectl delete pv <OAM_DOMAIN_NAME>-domain-pv
```

For example:

```
kubectl delete pvc -n oamns accessdomain-domain-pvc


kubectl delete pv accessdomain-domain-pv
```

10. Remove the OAM namespace.

```
kubectl delete namespace <OAMNS>
```

For example:

```
kubectl delete namespace oamns
```

11. Remove the files on the persistent volume.

```
rm -rf /nfs_volumes/oampv
```

12. Remove the files from the working directory.

```
rm -f /workdir/OAM
```

For the list of applicable variables, see Variables Used in this Chapter.

# Oracle Identity Governance

To remove Oracle Identity Governance:

1. Delete the domain creation job using the command:

```
kubectl delete jobs <OIG_DOMAIN_NAME>-create-fmw-infra-sample-domain-job -
n <OIGNS>
```

For example:

```
kubectl delete jobs governancedomain-create-fmw-infra-sample-domain-job -n
oigns
```

2. Delete the domain using the command:

```
kubectl delete domain <OIG_DOMAIN_NAME> -n <OIGNS>
```

For example:

```
kubectl delete domain governancedomain -n oigns
```

3. Delete the configmap using the command:

```
kubectl delete configmaps <OIG_DOMAIN_NAME>-create-fmw-infra-sample -infra-
domain-job-cm -n <OIGNS>
```

For example:

```
kubectl delete configmaps governancedomain-create-fmw-infra-sample-domain-
job-cm -n oigns
```

4. If you have installed Logstash, you also need to execute the following commands:

```
kubectl delete deployment -n <OIGNS> oig-logstash

kubectl delete cm -n <OIGNS> oig-logstash-configmap

kubectl delete cm -n <OIGNS> elk-cert
```

5. Delete the NodePort Services if you have defined them, using the commands:

```
kubectl delete service -n <OIGNS> <OIG_DB_SERVICE>
```

For example:

```
kubectl delete service -n oigns governancedomain-oim-t3-nodeport

kubectl delete service -n oigns governancedomain-oim-nodeport

kubectl delete service -n oigns governancedomain-soa-nodeport
```

6. Check that the domain has completely stopped, using the command:

```
kubectl get all -n <OIGNS> -o wide
```

For example:

```
kubectl get all -n oigns -o wide
```

7. Drop the OIG schemas by using the commands:
   a. Start a bash shell in the helper pod using the following command:

```
kubectl exec -it helper -n <OIGNS> -- /bin/bash
```

For example:

```
kubectl exec -it helper -n oigns -- /bin/bash
```

   b. Set up the environment variables which you will use to connect to the database.

```
export DB_HOST=<OIG_DB_SCAN>
export DB_PORT=<OIG_DB_LISTENER>
export DB_SERVICE=<OIG_DB_SERVICE>
export CONNECTION_STRING=$DB_HOST:$DB_PORT/$DB_SERVICE
export RCU_PREFIX=<OIG_RCU_PREFIX>
export RCU_SCHEMA_PWD=<OIG_SCHEMA_PWD>
echo -e <OIG_DB_SYS_PWD>"\n"<OIG_SCHEMA_PWD> /tmp/pwd.txt
```

For example:

```
export DB_HOST=DBSCAN.example.com
export DB_PORT=1521
export DB_SERVICE=igdedg.example.com
export CONNECTION_STRING=$DB_HOST:$DB_PORT/$DB_SERVICE
export RCU_PREFIX=IGDEDG
export RCU_SCHEMA_PWD=rcupassword
echo -e syspassword"\n"rcupassword > /tmp/pwd.txt
```

Check that there are two entries in the `/tmp/pwd` file.

For example:

```
cat /tmp/pwd.txt


syspassword
rcupassword
```

c. Run the following command to create the RCU schemas in the database:

```
/u01/oracle/oracle_common/bin/rcu -silent -fropRepository -databaseType
ORACLE -connectString \
$CONNECTION_STRING -dbUser sys -dbRole sysdba \
-selectDependentsForComponents true -schemaPrefix $RCU_PREFIX -
component MDS -component IAU -component OIM \
-component SOAINFRA -component OPSS -component WLS -component STB -
component IAU_APPEND -component IAU_VIEWER -component UCSUMS -f < /tmp/
pwd.txt
```

8. Remove persistent volume and claim from Kubernetes using the commands:

```
kubectl delete pvc -n <OIGNS> <OIG_DOMAIN_NAME>-domain-pvc


kubectl delete pv <OIG_DOMAIN_NAME>-domain-pv
```

For example:

```
kubectl delete pvc -n oigns governancedomain-domain-pvc


kubectl delete pv governancedomain-domain-pv
```

9. Remove the OIG namespace.

```
kubectl delete namespace -n <OIGNS>
```

For example:

```
kubectl delete namespace -n oigns
```

10. Remove the files on the persistent volume.

```
rm -rf /nfs_volumes/oigpv
```

11. Remove the files from the working directory.

```
rm -f /workdir/OIG
```

For the list of applicable variables, see Variables Used in this Chapter.

# WebLogic Operator for Kubernetes

Before you remove the WebLogic Operator for Kubernetes, ensure that there are no domains in the Kubernetes cluster that is being managed by the WebLogic Operator. Removing the Operator may cause the dependent existing domains to cease working.

To remove the WebLogic Operator for Kubernetes:

1. Remove the Operator deployment using the command:

```
helm uninstall --namespace <OPERNS> weblogic-kubernetes-operator
```

For example:

```
helm uninstall --namespace operns weblogic-kubernetes-operator
```

2. Remove the Operator namespace.

```
kubectl delete namespace -n <OPERNS>
```

For example:

```
kubectl delete namespace -n operns
```

3. Remove the local working directory.

```
rm -rf /workdir/OPER/*
```

# Oracle Identity Role Intelligence

To remove OIRI:

1. Connect to the `oiri-cli` pod by using the command:

```
kubectl exec -n <OIRINS> -ti oiri-cli /bin/bash
```

For example:

```
kubectl exec -n oirins -ti oiri-cli /bin/bash
```

2. Remove the OIRI deployment using the command:

```
helm delete oiri
```

3. Delete the OIRI schemas from the `oiri-cli` pod by using the command:

```
oiri-cli --config=/app/data/conf/config.yaml schema drop /app/data/conf/
dbconfig.yaml \
             --sysp <syspassword>
```

4. Delete the `oiri-cli` pod by using the command:

```
kubectl delete pod -n <OIRINS> oiri-cli
```

   For example:

```
kubectl delete pod -n oirins oiri-cli
```

5. Delete the `oiri-ding-cli` pod by using the command:

```
kubectl delete pod -n <DINGNS> oiri-ding-cli
```

   For example:

```
kubectl delete pod -n dingns oiri-ding-cli
```

6. Remove the OIRI namespace.

```
kubectl delete namespace -n <OIRINS>
```

   For example:

```
kubectl delete namespace -n oirins
```

7. Remove the DING namespace.

```
kubectl delete namespace -n <DINGNS>
```

   For example:

```
kubectl delete namespace -n dingns
```

8. Remove the files on the persistent volume.

```
rm -rf /nfs_volumes/oiripv /nfs_volumes/dingpv /nfs_volumes/workpv
```

9. Remove the files from the working directory.

```
rm -f /workdir/OIRI
```

For the list of applicable variables, see Variables Used in this Chapter.

# Oracle Advanced Authentication

To remove Oracle Advanced Authentication:

1. Remove the OAA application.

   From the OAA-MGMT pod, run the following command:

   ```
   helm uninstall <OAA_APPLICATION> -n <OAANS>
   ```

   For example:

   ```
   helm uninstall edg -n oaans
   ```

2. Ensure that all pods have been terminated and deleted before continuing, by using the following command:

   ```
   kubectl get pods -n oaans
   ```

   Ensure that there are no STS services running before continuing, by using the following command:

   ```
   kubectl get sts -n oaans
   ```

   If any STS services are still running, then attempt to stop them using the following commands:

   ```
   kubectl patch sts -n oaans edg-cache-rest -p '{"spec":{"replicas":0}}'

   kubectl patch sts -n oaans edg-cache-proxy -p '{"spec":{"replicas":0}}'

   kubectl patch sts -n oaans edg-cache-storage -p '{"spec":{"replicas":0}}'
   ```

   Where `edg` is the name of the OAA deployment.

3. Remove the OAM integration. You can perform these steps using `curl` or the OAM Console. For simplicity, the `curl` commands are shown.

   a. Encode `oamadmin` and its password by using the following command:

      ```
      echo -n oamadmin:<password> | base64
      ```

   b. Delete the authentication scheme using the following `curl` command:

      ```
      curl --location --request DELETE "http://
      worker1.example.com:<OAM_K8_PORT>/oam/services/rest/11.1.2.0.0/ssa/
      policyadmin/authnscheme?name=OAA-MFA-Scheme" --header "Authorization:
      Basic <ENCODED_OAMADMIN>"
      ```

For example:

```
curl --location --request DELETE "http://
worker1.example.com:<OAM_K8_PORT>/oam/services/rest/11.1.2.0.0/ssa/
policyadmin/authnscheme?name=OAA-MFA-Scheme" --header "Authorization:
Basic <ENCODED_OAMADMIN>"
```

c. Delete the authentication policy using the following `curl` command.

```
curl --location --request DELETE "http://
worker1.example.com:<OAM_K8_PORT>/oam/services/rest/11.1.2.0.0/ssa/
policyadmin/authnpolicy?appdomain=IAM Suite&name=OAA_MFA-Policy" --
header "Authorization: Basic b2FtYWRtaW46cGFzc3dvcmQ="
```

For example:

```
curl --location --request DELETE "http://worker1.example.com:30701/oam/
services/rest/11.1.2.0.0/ssa/policyadmin/authnpolicy?appdomain=IAM
Suite&name=OAA_MFA-Policy" --header "Authorization: Basic
b2FtYWRtaW46cGFzc3dvcmQ="
```

4. Delete the OAuth client.

a. Encode `oamadmin` and its password by using the command:

```
echo -n oamadmin:<password> | base64
```

b. Delete the OAuth client using the following `curl` command:

```
curl --location --request DELETE "http://
worker1.example.com:<OAM_K8_PORT>/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/client?
name=OAAClient&identityDomainName=<OAA_DOMAIN>"  --header
"Authorization: Basic <ENCODED_OAMADMIN>"
```

for example:

```
curl --location --request DELETE "http://worker1.example.com:30701/oam/
services/rest/ssa/api/v1/oauthpolicyadmin/client?
name=OAAClient&identityDomainName=OAADomain>"  --header "Authorization:
Basic b2FtYWRtaW46cGFzc3dvcmQ="
```

c. Delete the OAuth resource server using the command:

```
curl --location --request DELETE "http://
worker1.example.com:<OAM_K8_PORT>/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/application?
name=OAAResource&identityDomainName=<OAA_DOMAIN>" --header
"Authorization: Basic <ENCODED_OAMADMIN>"
```

For example:

```
curl --location --request DELETE http://worker1.example.com:30701/oam/
services/rest/ssa/api/v1/oauthpolicyadmin/application?
```

```
name=OAAResource&identityDomainName=OAADomain" --header "Authorization:
Basic b2FtYWRtaW46cGFzc3dvcmQ="
```

d. Delete the OAuth domain using the command:

```
curl --location --request DELETE " http://
worker1.example.com:<OAM_K8_PORT>/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain?name=<OAA_DOMAIN>" --header
"Authorization: Basic <ENCODED_OAMADMIN>"
```

For example:

```
curl --location --request DELETE " http://worker1.example.com:30701/oam/
services/rest/ssa/api/v1/oauthpolicyadmin/oauthidentitydomain?
name=OAADomain--header "Authorization: Basic b2FtYWRtaW46cGFzc3dvcmQ=
```

5. Delete the database schemas. From the OAA-MGMT pod, run the following commands:

```
sqlplus sys/<OAA_DB_SYS_PWD>@<DB_SCAN>:<DB_LISTENER>/<OAA_DB_SERVICE> as
sysdba

alter session set "_oracle_script"=TRUE; ** Required for PDB's **

drop user <OAA_RCU_PREFIX>_oaa cascade;
delete from SCHEMA_VERSION_REGISTRY where comp_name='Oracle Advanced
Authentication' and OWNER=UPPER('<OAA_RCU_PREFIX>_OAA');

commit;

set pages 0
set feedback off
spool /tmp/drop_directories.sql
select 'drop directory '||directory_name||';' from all_directories
where directory_name like 'EXPORT%'
/
spool off
@/tmp/drop_directories
```

6. Delete role bindings using the commands:

```
kubectl delete rolebinding -n oaans oaa-rolebinding


kubectl delete clusterrolebinding oaa-clusterrolebinding


kubectl delete clusterrolebinding oaa-clusteradmin


kubectl delete role oaa-ns-role -n oaans


kubectl delete serviceaccount -n oaans oaa-service-account
```

7. Delete the OAA management pod.

```
kubectl delete pod -n oaans oaa-mgmt
```

8. Delete the namespaces using the commands:

```
kubectl delete namespace oaans
```

9. Delete the contents of the persistent volumes using the commands:

> **Note:**
>
> The assumption is that you have the volumes mounted locally under `/nfs_volumes`.

```
rm -rf <WORKDIR>
```

```
rm -rf /nfs_volumes/oaacredpv/*
```

```
rm -rf /nfs_volumes/oaaconfigpv/*
```

```
rm -rf /nfs_volumes/oaalogpv/*
```

```
rm -rf /nfs_volumes/oaalogpv /* /nfs_volumes/oaalogpv /.??*
```

10. Remove the OAA entries from the OCI vault:

```
Delete vault secret "<OAA_DEPLOYMENT>-map-oaaManifestMap.txt
```

Change the default deletion date when scheduling the secret for deletion. For mor information, see Managing Vaults.

Wait for `<OAA_DEPLOYMENT>-map-oaaManifestMap.txt` to be removed from the vault. The secret being in the `deleted` state is not sufficient. The removal from the vault takes upto 48 hours.

After the secret is deleted, perform a new installation of OAA.

For the list of applicable variables, see Variables Used in this Chapter.

# Ingress Controller

To remove Ingress controller:

1. Remove the controller using the following command:

```
helm uninstall -n <INGRESSNS> nginx-ingress
```

2. Remove the namespace:

```
kubectl delete namespace -n <INGRESSNS>
```

For the list of applicable variables, see Variables Used in this Chapter .

# Elasticsearch and Kibana

To remove Elasticsearch and Kibana:

1. Remove the NodePort Services:

```
kubectl delete service -n <ELKNS> kibana-nodeport


kubectl delete service -n <ELKNS> elk-nodeport
```

2. Remove Kibana:

```
kubectl delete kibana -n <ELKNS> kibana
```

3. Remove the Elasticsearch cluster:

```
kubectl delete elasticsearch -n <ELKNS> elasticsearch
```

4. Remove the Elasticsearch Operator:

```
helm uninstall -n <ELKNS> elastic-operator
```

5. Remove the namespace:

```
kubectl delete namespace -n <ELKNS>
```

For the list of applicable variables, see Variables Used in this Section.

# Prometheus and Grafana

To remove Prometheus and Grafana:

1. Remove the application:

```
helm uninstall kube-prometheus -n <PROMNS>
```

2. Remove Custom Resource Definitions:

```
kubectl delete crd -n <PROMNS> alertmanagerconfigs.monitoring.coreos.com

kubectl delete crd -n <PROMNS> alertmanagers.monitoring.coreos.com

kubectl delete crd -n <PROMNS> podmonitors.monitoring.coreos.com

kubectl delete crd -n <PROMNS> probes.monitoring.coreos.com

kubectl delete crd -n <PROMNS> prometheuses.monitoring.coreos.com

kubectl delete crd -n <PROMNS> prometheusrules.monitoring.coreos.com

kubectl delete crd -n <PROMNS> servicemonitors.monitoring.coreos.com

kubectl delete crd -n <PROMNS> thanosrulers.monitoring.coreos.com
```

3. Remove the namespace:

```
kubectl delete namespace <PROMNS>
```

E

# Automating the OCI Infrastructure Creation for the Identity and Access Management Kubernetes Cluster

To setup the Oracle Identity and Access Management in a Kubernetes cluster, see Preparing the Oracle Cloud Infrastructure for an Enterprise Deployment, which contains many security lists, hosts, VCNs, and related resources that need to be created for a successful deployment. This utility consists of samples for you to automate the configuration and use the OCI command-line interface.

These scripts are provided as examples and can be customized as required.

This appendix includes the following topics:

- Obtaining the Scripts
- Scope of Scripts
  Learn about the actions that the scripts perform as part of the deployment process. There are also tasks that the scripts do not perform.
- Key Concepts of the Scripts
  To make things simple and easy to manage, the scripts include these files: a response file with details of the environment and template files you can easily modify or add as required. The scripts can be run from any host which has access to the Kubernetes cluster.
- Prerequisites
- Creating a Response File
- Provisioning the Environment
- Log Files
- Output Files
- Reference - Response File
  The parameters in the response file are used to control the provisioning of the various products in the Kubernetes cluster. These parameters are divided into generic and product-specific parameters.
- Components of the Deployment Scripts

## Obtaining the Scripts

These scripts are included as part of the automation scripts. The automation scripts are available for download from GitHub.

For more information, see Automating the Identity and Access Management Enterprise Deployment.

To obtain the scripts, use the following command:

```
git clone https://github.com/oracle/fmw-kubernetes.git
```

The scripts appear in the following directory:

```
fmw-kubernetes/FMWKubernetesMAA/OracleEnterpriseDeploymentAutomation/
OracleIdentityManagement/oke_utils
```

Move these template scripts to your working directory. For example:

```
cp -R fmw-kubernetes/FMWKubernetesMAA/OracleEnterpriseDeploymentAutomation/
OracleIdentityManagement/* /workdir/scripts
```

This directory will be specified as `$SCRIPTDIR`.

# Scope of Scripts

Learn about the actions that the scripts perform as part of the deployment process. There are also tasks that the scripts do not perform.

- What the Scripts Will do
- What the Scripts Will Not Do

## What the Scripts Will do

The scripts perform the following actions:

- Create a Bastion host as described in Creating a Bastion Node.
- Create two compute instances for the Oracle HTTP as described in Creating Compute Instances for Oracle HTTP Servers.
- Create the NFS file systems and mount points for the Kubernetes persistent volume data as described in Creating File Systems and Mount Targets.
- Create a public and internal load balancer for routing requests to the OHS servers as described in Creating Load Balancers.
- Create a TCP load balancer to route requests to the Kubernetes nodes as described in Creating a Network Load Balancer.
- Create the RAC database used for the Identity Management product schemas as described in Creating a Database.
- Create a DNS server for name resolution within the Kubernetes cluster as described in Creating a DNS Server.
- Tune the RAC database.
- Install the Java Server and Oracle Text database options which are required by the OIM server. See Adding Database Options.
- Install the XA views into the OIM pluggable database. See Adding XA Views.
- Create the pluggable databases, as configured, for the OAM, OIM, OAA, and OIRI schemas as described in Creating a PDB Using an Existing PDB as a Template.
- Create the database services for the pluggable databases as described in Creating Database Services.

## What the Scripts Will Not Do

The scripts do not perform the following tasks:

- Install monitoring software such as Grafana or Prometheus.
- Install or configure the log file monitoring tools Elasticsearch and Kibana.

# Key Concepts of the Scripts

To make things simple and easy to manage, the scripts include these files: a response file with details of the environment and template files you can easily modify or add as required. The scripts can be run from any host which has access to the Kubernetes cluster.

> **Note:**
>
> Provisioning scripts are re-enterant. If something fails, you can restart the script from the point at which it failed.

# Prerequisites

Before you run the utility, ensure that you meet the necessary prerequisites.

- Install and configure the OCI command-line tools. For instructions, see one of the following docs:
  - 2432759.1
  - OCI documentation
- Ensure that the OCI command runs properly by using the following command and check if you get valid output.

  ```
  oci iam availability-domain list
  ```

- Create an SSH private/public key that will be used to access the OCI instances using the command:

  ```
  ssh-keygen
  ```

- Create a configuration response file defining the parameters desired for this installation using the delivered file, `oci-oke.rsp`, as an example.
- You have enough quota available in your tenancy to create the various resources.
- The utility `gdate` is available on the deployment host.

# Creating a Response File

A sample response file called "`oci-oke.rsp`" and "`.ocipwd`" is created for you in the `SCRIPT_DIR/oke_utils/responsefile` directory. You can edit this file or copy it to another file in the same directory.
All lines above the line "`All changes below this point are optional`" should be carefully reviewed and set as appropriate. Some values include default values while others are

installation dependent and need to be set explicitly. This file will be referred as "TEMPLATE_NAME".

> **Note:**
>
> - The file consists of key/value pairs. There should be no spaces between the name of the key and its value. For example:
>   `Key=value`
> - The OCI image names change frequently. It may be out-of-date and require an update before creating any instance.

# Provisioning the Environment

A script is provisioned by specifying a working directory, a log directory into which output from the script is recorded, and a command to execute the script.
Run the provision script by using the command:

```
cd $SCRIPT_DIR/oke_utils
./provision_oke.sh <TEMPLATE_NAME>
```

> **Note:**
>
> The provision script runs non-interactively after asking for confirmation that the listed compartment should be used to install the OCI components. Output from the script is displayed on the screen as well as added to the `provision_oci.log` file.

# Log Files

The provisioning scripts create log files for each product inside the working directory in a `TEMPLATE_NAME/logs` sub-directory.

This directory also contains the following files:

- `progressfile` – This file contains the last successfully executed step. If you want to restart the process at a different step, update this file.
- `timings.log` – This file is used for informational purposes to show how much time was spent on each stage of the provisioning process.
- `provision_oci.log` – This file is used to capture the output from the execution of the various `oci` commands that were run by the `provision_oke.sh` script. This is the main provisioning log file.

# Output Files

The output files are generated as part of the provisioning process.

**Table E-1 Output Files**

| File Name | Directory | Content |
|---|---|---|
| ca.crt | $WORKDIR/<TEMPLATE_NAME>/output | The self-signed certificate authority SSL certificate. |
| ca.csr | $WORKDIR/<TEMPLATE_NAME>/output | The certificate authority signing request which can be helpful when needed to renew the CA certificate. |
| ca.key | $WORKDIR/<TEMPLATE_NAME>/output | The self-signed certificate authority SSL private key. |
| ca.srl | $WORKDIR/<TEMPLATE_NAME>/output | The openssl serial number used when signing the CA certificate. |
| TEMPLATE_NAME.ocid | $WORKDIR/<TEMPLATE_NAME>/output | A listing that includes all of the resources created by the script and their associated OCID value. This file is used when the delete_oke.sh script is run to know which resources to delete. |
| loadbalancer.crt | $WORKDIR/<TEMPLATE_NAME>/output | The self-signed SSL certificate used by the public and internal load balancers. This file is also used by the OAM WebGate for making an SSL connection to public load balancer. |
| loadbalancer.key | $WORKDIR/<TEMPLATE_NAME>/output | The SSL private key for the public/internal load balancer SSL certificate. |
| bastion_mounts.sh | $WORKDIR/<TEMPLATE_NAME>/output | A bash shell script that can be run manually to mount the NFS volumes on the Bastion host. |
| webhost1_mounts.sh | $WORKDIR/<TEMPLATE_NAME>/output | A bash shell script that can be run manually to mount the NFS volumes on WebHost1. |
| webhost2_mounts.sh | $WORKDIR/<TEMPLATE_NAME>/output | A bash shell script that can be run manually to mount the NFS volumes on WebHost2. |
| db-tuning.sh | $WORKDIR/<TEMPLATE_NAME>/output | A bash shell script that can be run manually to configure the database init.ora parameters for the selected memory size defined by the DB_MEMORY_CONFIG parameter. |
| db-xaviews.sh | $WORKDIR/<TEMPLATE_NAME>/output | A bash shell script that can be run manually to install the XA views into the OIG pluggable database. |

**Table E-1 (Cont.) Output Files**

| File Name | Directory | Content |
|---|---|---|
| <TEMPLATE>_idm.rsp | $WORKDIR/<TEMPLATE_NAME>/output | Generated by the script create_idm_rsp.sh. It is a file which can be used as the basis of a response file for the IDM Automation scripts described in Automating the Identity and Access Management Enterprise Deployment. This response file contains information gathered from the provisioning of OKE. It will be reviewed and updated as per your environment needs. |
| interim_parameters | $WORKDIR/<TEMPLATE_NAME>/output | Generated by the script create_idm_rsp.sh. This response file contains information gathered from the provisioning of OKE which will be reviewed and updated as per your environment needs for faster processing. |



# Deleting the Environment

The delete script will read the file $WORKDIR/TEMPLATE_NAME/output/TEMPLATE_NAME.ocid to determine which resources were created by the provisioning script and make an attempt to delete them.

Run the delete script by using the command:

```
cd $SCRIPT_DIR/oke_utils
./delete_oke.sh <TEMPLATE_NAME>
```

Like the provisioning script, the delete script will first confirm that the resources should be deleted from the listed compartment and then run without further user input to delete the resources.

# Deleting Output Files

**Table E-2 Deleting Output Files**

| File Name | Directory | Contents |
|---|---|---|
| delete_oke.log | $WORKDIR/<TEMPLATE_NAME>/logs | The output from the execution of the various oci commands that were run by the delete_oke.sh script. This is the main deletion log file. |

**Table E-2 (Cont.) Deleting Output Files**

| File Name | Directory | Contents |
|---|---|---|
| timings.log | $WORKDIR/<TEMPLATE_NAME>/logs | This file is used for informational purposes to show the time spent on each stage of the deletion process. |

# Reference - Response File

The parameters in the response file are used to control the provisioning of the various products in the Kubernetes cluster. These parameters are divided into generic and product-specific parameters.



## Parameters that Must be Reviewed, Set, and Modified

Some values are default while others are installation dependent and require to be set explicitly.

**Table E-3 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| WORKDIR | /home/opc/workdir/OKE | Absolute path to the directory where you want to have the output and log files written to. |

**Table E-3 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| REGION | <your-region> | The OCI region in which you want to create all the resource. For example: us-ashburn-1. |
| COMPARTMENT_NAME | <your-compartment-name> | The compartment name which will hold all of the created resources. |
| SSH_PUB_KEYFILE | <path-to>/id_rsa.pub | Absolute path to the SSH public keyfile. This field needs to be updated in the ".ocipwd" present in your SCRIPT_DIR/oke_utils/responsefile directory. |
| SSH_ID_KEYFILE | <path-to>/id_rsa | Absolute path to the SSH private keyfile used to connect to the Bastion host. This field needs to be updated in the ".ocipwd" present in your SCRIPT_DIR/oke_utils/responsefile directory. |
| SSL_COUNTRY | <country> | The name of the country to use in the C portion of the SSL certificate. |
| SSL_STATE | <state> | The name of the country to use in the ST portion of the SSL certificate. |
| SSL_LOCALE | <city> | The name of the country to use in the L portion of the SSL certificate. |
| SSL_ORG | <company> | The name of the country to use in the O portion of the SSL certificate. |
| SSL_ORGUNIT | <organization> | The name of the country to use in the OU portion of the SSL certificate. |
| DB_PWD | <dbpwd> | The password for the SYS and SYSTEM users in the RAC database. This field needs to be updated in the ".ocipwd" present in your SCRIPT_DIR/oke_utils/responsefile directory.<br>**Note:** The password must contain two uppercase, two lowercase, two number, and two special characters and a minimum length of 10 characters. |
| DB_NAME | idmdb | The value to use for the database DB_NAME parameter. |
| DB_SUFFIX | edg | A suffix, which combined with the DB_NAME value, makes up the value for the DB_UNIQUE_NAME database parameter. |
| DB_MEMORY_CONFIG | dev | Which set of database tuning parameter, from Table 11-4 in the EDG, should be used for the RAC database. |
| CONFIGURE_DATABASE | true | The RAC database configuration/tuning script run automatically after the database is created.<br>**Note:** If this is set to "Y" then the provisioning script will wait up to three hours for the initial RAC database to become available before proceeding. |
| CREATE_OAM_PDB | true | Indicates if the CONFIGURE_DATABASE is enabled when an OAM pluggable database be created. |
| OAM_PDB_NAME | oampdb | The name of the OAM pluggable database. |

**Table E-3 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| OAM_SERVICE_NAME | `oam_s` | The name of the OAM database service. |
| CREATE_OIG_PDB | `true` | Indicates if the `CONFIGURE_DATABASE` is enabled when an OIG pluggable database be created. |
| OIG_PDB_NAME | `oigpdb` | The name of the OIG pluggable database. |
| OIG_SERVICE_NAME | `oig_s` | The name of the OIG database service. |
| CREATE_OAA_PDB | `false` | Indicates if the`CONFIGURE_DATABASE` is enabled when an OAA pluggable database be created. This field needs to be toggled to true if a pdb for OAA is required. |
| OAA_PDB_NAME | `oaapdb` | The name of the OAA pluggable database. |
| OAA_SERVICE_NAME | `oaa_s` | The name of the OAA database service. |
| CREATE_OIRI_PDB | `false` | Indicates if the `CONFIGURE_DATABASE` is enabled when an OIRI pluggable database be created. |
| OIRI_PDB_NAME | `oiripdb` | The name of the OIRI pluggable database. |
| OIRI_SERVICE_NAME | `oiri_s` | The name of the OIRI database service. |
| DEFAULT_HOST_SHAPE | `VM.Standard.E4.Flex` | OCI Shape from which various Hosts and Nodepools are created. |
| BASTION_IMAGE_NAME | `Oracle-Linux-8` | The Linux image to use for the Bastion host. |
| WEB_IMAGE_NAME | `$BASTION_IMAGE_NAME` | The Linux image to use for the two web tiers. |
| OKE_NODE_POOL_IMAGE_NAME | `$BASTION_IMAGE_NAME` | The Linux image to use for the OKE nodes. |
| CONFIGURE_BASTION | `true` | Indicates if the Bastion host should be automatically configured with the required OS packages, OCI tools, helm, and Kubernetes configuration after the installation is complete. |
| HELM_VER | `3.11.1` | The version of helm to install on the Bastion node. If the latest version of helm is required, please update this field as `Latest`. |
| CONFIGURE_WEBHOSTS | `true` | Indicates if the web tiers should be automatically configured with the required OS packages and firewall settings after the installation is complete. |
| OHS_SOFTWARE_OWNER | `opc` | The OS user that will own the OHS configuration on the web tiers. |
| OHS_SOFTWARE_GROUP | `opc` | The OS group that will own the OHS configuration on the web tiers. |

The rest of the parameters use the default values, and it is not required to change them. However, it may be reviewed and changed to customize the installation.

# OCI Command-Line Interface Region

**Table E-4 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| OCI_CLI_REGION | `$REGION` | This value overrides the default REGION set in the `$HOME/.oci/config` file.<br>**Note:** Do not change this value. |

# Port Numbers

**Table E-5 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| OAM_ADMIN_SERVICE_PORT | `30701` | The Kubernetes service port for the OAM AdminServer. |
| OAM_POLICY_SERVICE_PORT | `30510` | The Kubernetes service port for the OAM Policy Manager. |
| OAM_SERVER_SERVICE_PORT | `30410` | The Kubernetes service port for the OAM Server. |
| OIG_ADMIN_SERVICE_PORT | `30711` | The Kubernetes service port for the OIG AdminServer. |
| OIG_SERVER_SERVICE_PORT | `30140` | The Kubernetes service port for the OIM Server. |
| SOA_SERVER_SERICE_PORT | `30801` | The Kubernetes service port for the SOA Server. |
| OUDSM_SERVER_SERVICE_PORT | `30901` | The Kubernetes service port for the OUDSM Server. |
| INGRESS_SERVICE_PORT | `30777` | The Kubernetes service port for the Ingress Controller. |
| OHS_NON_SSL_PORT | `7777` | The OHS port used by the internal load balancer for internal callback requests. |
| PUBLIC_LBR_NON_SSL_PORT | `80` | The load balancer port used for the HTTP requests to the OAM and OIM AdminServers. |
| PUBLIC_LBR_SSL_PORT | `443` | The load balancer port used for the HTTPS requests to the OAM login page and the OIG provisioning server. |

# Subnet Configuration

**Table E-6 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| VCN_SUBNET_CIDR | `10.0.0.0/16` | The CIDR to use for the Virtual Cloud Network. |
| BASTION_SUBNET_CIDR | `10.0.1.0/29` | The subnet (which must be contained within the main VCN subnet) to use for the Bastion host. |

**Table E-6 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| WEB_SUBNET_CIDR | `10.0.2.0/28` | The subnet (which must be contained within the main VCN subnet) to use for the WebTier hosts. |
| LBR1_SUBNET_CIDR | `10.0.4.0/24` | The first subnet (which must be contained within the main VCN subnet) to use for the public load balancer. |
| LBR2_SUBNET_CIDR | `10.0.5.0/24` | The second subnet (which must be contained within the main VCN subnet) to use for the public load balancer. |
| DB_SUBNET_CIDR | `10.0.11.0/24` | The subnet (which must be contained within the main VCN subnet) to use for the RAC database. |
| OKE_NODE_SUBNET_CIDR | `10.0.10.0/24` | The subnet (which must be contained within the main VCN subnet) to use for the OKE nodes. |
| OKE_API_SUBNET_CIDR | `10.0.0.0/28` | The subnet (which must be contained within the main VCN subnet) to use for the OKE API endpoint. |
| OKE_SVCLB_SUBNET_CIDR | `10.0.20.0/24` | The subnet (which must be contained within the main VCN subnet) to use for the OKE services load balancer. |

# DNS Zone Configuration

**Table E-7 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| DNS_DOMAIN_NAME | `example.com` | The DNS domain name for the environment. |
| DNS_ZONE_TYPE | `PRIMARY` | Used to identify if the zone is primary or secondary. |
| DNS_SCOPE | `PRIVATE` | Used to identity if the zone is private or a global DNS zone. |
| DNS_INTERNAL_LBR_DNS_HOSTNAME | `loadbalancer.$DNS_DOMAIN_NAME` | The host name of the load balancer used for internal routing. |

# VCN Configuration

**Table E-8 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| VCN_DISPLAY_NAME | `idm-oke-vcn` | Displays the name of the virtual cloud network. |
| VCN_PRIVATE_ROUTE_TABLE_DISPLAY_NAME | `oke-private-rt` | Displays the name of the VCN private route table. |
| VCN_PUBLIC_ROUTE_TABLE_DISPLAY_NAME | `oke-public-rt` | Displays the name of the VCN public route table. |

**Table E-8 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| VCN_DNS_LABEL | `oke` | The DNS label for the VCN. Used in conjunction with the host name and subnet DNS label to form a FQDN for each host. |
| VCN_INTERNET_GATEWAY_DISPLAY_NAME | `oke-igw` | Displays the name of the VCN internet gateway. |
| VCN_NAT_GATEWAY_DISPLAY_NAME | `oke-nat` | Displays the name of the VCN NAT gateway. |
| VCN_SERVICE_GATEWAY_DISPLAY_NAME | `oke-sgw` | Displays the name of the VCN service gateway. |

# OKE Cluster Configuration

**Table E-9 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| OKE_CLUSTER_DISPLAY_NAME | `oke-cluster` | Displays the name of the OKE cluster. |
| OKE_CLUSTER_TYPE | `STD` | Please choose the Cluster type - STD (Standard) or ENH (Enhanced). |
| OKE_CLUSTER_VERSION | `v1.29.1` | Version of the Kubernetes to deploy on the OKE nodes. |
| OKE_MOUNT_TARGET_AD | `ad1` | Displays the name of the availability domain to use for the OKE mount target.<br>**Note:** This value is not the actual availability domain name but a representation of the AD to use. For example: ad1, ad2, or ad3. |
| OKE_PODS_CIDR | `10.244.0.0/16` | The CIDR for the OKE pods. |
| OKE_SERVICES_CIDR | `10.96.0.0/16` | The CIDR for the OKE load balancer services. |
| OKE_NETWORK_TYPE | `FLANNEL_OVERLAY` | The CNI type for the node pools of the cluster. |
| OKE_API_SUBNET_DISPLAY_NAME | `oke-k8sApiEndpoint-subnet` | Displays the name of the OKE API subnet. |
| OKE_API_DNS_LABEL | `apidns` | The DNS label for the OKE API subnet. Used in conjunction with the host name and subnet DNS label to form a FQDN for each host within the subnet. |
| OKE_API_SECLIST_DISPLAY_NAME | `oke-k8sApiEndpoint-seclist` | Displays the name of the OKE API security list. |
| OKE_NODE_SUBNET_DISPLAY_NAME | `oke-node-subnet` | Displays the name of the OKE node security list. |
| OKE_NODE_DNS_LABEL | `nodedns` | The DNS label for the OKE nodes. Used in conjunction with the host name and subnet DNS label to form a FQDN for each host within the subnet. |
| OKE_NODE_SECLIST_DISPLAY_NAME | `oke-node-seclist` | Displays the name of the OKE node security list. |

**Table E-9 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| OKE_SVCLB_SUBNET_DISPLAY_NAME | `oke-svclb-subnet` | Displays the name of the OKE service load balancer subnet. |
| OKE_SVCLBR_DNS_LABEL | `svclbdns` | The DNS label for the OKE service load balancer. Used in conjunction with the host name and subnet DNS label to form a FQDN for each host within the subnet. |
| OKE_SVCLBR_SECLIST_DISPLAY_NAME | `oke-svclb-seclist` | Displays the name of the service load balancer security list. |
| OKE_NODE_POOL_DISPLAY_NAME | `pool1` | Displays the name of the OKE node pool. |
| OKE_NODE_POOL_SIZE | `3` | Displays the number of nodes to add to the OKE node pool. |
| OKE_NODE_POOL_SHAPE | `$DEFAULT_HOST_SHAPE` | Displays which image shape to use for the OKE nodes. |
| OKE_NODE_POOL_SHAPE_CONFIG | `'{\\\"memoryInGBs\\\": 32.0, \\\"ocpus\\\": 2.0}'` | Shape configuration for the memory and OCPUs for the OKE nodes.<br>**Note:** Entire string must be enclosed within single quotes and the parameter names must be enclosed within double quotes that are escaped with a backslash. |

# Bastion Host Configuration

**Table E-10 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| BASTION_PRIVATE_SECLIST_DISPLAY_NAME | `bastion-private-seclist` | Displays the name of the Bastion private security list. |
| BASTION_PUBLIC_SECLIST_DISPLAY_NAME | `bastion-public-seclist` | Displays the name of the Bastion public security list. |
| BASTION_SETUP_SECLIST_DISPLAY_NAME | `bastion-setup-seclist` | Displays the name of the Bastion setup security list. |
| BASTION_ROUTE_TABLE_DISPLAY_NAME | `bastion-route-table` | Displays the name of the Bastion route table. |
| BASTION_SUBNET_DISPLAY_NAME | `bastion-subnet` | Displays the name of the Bastion subnet. |
| BASTION_DNS_LABEL | `bastionsubnet` | The DNS label for the Bastion subnet. Used in conjunction with the host name and subnet DNS label to form a FQDN for each host. |
| BASTION_INSTANCE_DISPLAY_NAME | `idm-bastion` | Displays the name of the Bastion compute instance. |
| BASTION_AD | `ad1` | The name of the availability domain to use for the Bastion host.<br>**Note:** This value is not the actual availability domain name but a representation of the AD to use. For example: ad1, ad2, or ad3. |

**Table E-10 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| BASTION_INSTANCE_SHAPE | `$DEFAULT_HOST_SHAPE` | Displays which image shape to use for the Bastion compute instance. |
| BASTION_SHAPE_CONFIG | `'{\"memoryInGBs\": 16.0, \"ocpus\": 1.0}'` | Shape configuration for the memory and OCPUs for the Bastion node.<br>**Note:** Entire string must be enclosed within single quotes and the parameter names must be enclosed within double quotes that are escaped with a backslash. |
| BASTION_PUBLIC_IP | `true` | Displays should the Bastion host get assigned a public IP address or not. |
| BASTION_HOSTNAME | `idm-bastion` | The host name to use for the Bastion host. |

# OHS and Web Tier Configuration

**Table E-11 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| OHS_SECLIST_DISPLAY_NAME | `ohs-seclist` | Displays the name of the OHS security list. |
| WEBHOST_SERVERS | `2` | List the number of wehost servers that are required to be provisioned. The minimum value recommended is 2. |
| WEBHOST_PREFIX | `webhost` | The hostname prefix to be used while creating the required number of webhost servers. |
| WEB_PUBLIC_SECLIST_DISPLAY_NAME | `web-public-seclist` | Displays the name of the Web public security list. |
| WEB_ROUTE_TABLE_DISPLAY_NAME | `web-route-table` | Displays the name of the Web route table. |
| WEB_SUBNET_DISPLAY_NAME | `web-subnet` | Displays the name of the Web subnet. |
| WEB_DNS_LABEL | `websubnet` | The DNS label for the Web subnet. Used in conjunction with the host name and subnet DNS label to form a FQDN for each host. |
| WEB_PROHIBIT_PUBLIC_IP | `true` | Indicates if the Web subnet allows computing instances with a public IP address. |
| WEBHOST1_AD | `ad1` | Displays the name of the available domain to use for the first Web host.<br>**Note:** This value is not the actual availability domain name but a representation of the AD to use. For example: ad1, ad2, or ad3. |
| WEBHOST_SHAPE | `DEFAULT_HOST_SHAPE` | Displays the image shape to use for the first Web host. |

**Table E-11 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| WEBHOST_SHAPE_CONFIG | `'{\"memoryInGBs\": 16.0, \"ocpus\": 1.0}'` | Shape configuration for the memory and OCPUs for the first Web host.<br>**Note:** Entire string must be enclosed within single quotes and the parameter names must be enclosed within double quotes that are escaped with a backslash. |
| WEBHOST_PUBLIC_IP | `false` | Indicates if the first Web host get assigned to a public IP address. |
| WEBHOST1_PRODUCTS_PATH | `/u02/private/oracle/products` | Displays the directory on the first Web host where the OHS product should be installed. |
| WEBHOST1_CONFIG_PATH | `/u02/private/oracle/config` | Displays the directory on the first Web host where the OHS configuration files should be stored. |
| WEBHOST2_AD | `ad2` | Displays the name of the available domain to use for the second Web host.<br>**Note:** This value is not the actual availability domain name but a representation of the AD to use. For example: ad1, ad2, or ad3. |
| WEBHOST2_PRODUCTS_PATH | `/u02/private/oracle/products` | Displays the directory on the second Web host where the OHS product should be installed. |
| WEBHOST2_CONFIG_PATH | `/u02/private/oracle/config` | Displays the directory on the second Web host where the OHS configuration files should be stored. |

# NFS and Persistent Volume Configuration

**Table E-12 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| WEBHOST1_MOUNT_TARGET_DISPLAY_NAME | `webhost1-mt` | Displays the name of the mount target used by webhost1. |
| NFS_IAMPV_EXPORT_PATH | `/exports/IAMPVS` | Displays the name of the NFS file system to be used for creating various other PV_NFS_PATH's. |
| WEBHOST2_MOUNT_TARGET_DISPLAY_NAME | `webhost2-mt` | Displays the name of the mount target used by webhost2. |
| OKE_MOUNT_TARGET_DISPLAY_NAME | `oke-mt` | Displays the name of the mount target used by the OKE nodes. |
| PV_SECLIST_DISPLAY_NAME | `pv-seclist` | Displays the name of the persistent volume security list. |
| FS_WEBBINARIES1_DISPLAY_NAME | `webbinaries1` | Displays the name of the NFS file system for the OHS binaries on webhost1. |
| FS_WEBBINARIES1_PATH | `/exports/IAMBINARIES/webbinaries1` | Displays the path to the NFS file system where the OHS binaries are installed on webhost1. |

**Table E-12 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| FS_WEBBINARIES2_DISPLAY_NAME | webbinaries2 | Displays the name of the NFS file system for the OHS binaries on webhost2. |
| FS_WEBBINARIES2_PATH | /exports/IAMBINARIES/webbinaries2 | Displays the path to the NFS file system where the OHS binaries are installed on webhost2. |
| FS_WEBCONFIG1_DISPLAY_NAME | webconfig1 | Displays the name of the NFS file system for the OHS configuration data on webhost1. |
| FS_WEBCONFIG1_PATH | /exports/IAMCONFIG/webconfig1 | Displays the path to the NFS file system where the OHS configuration data is installed on webhost1. |
| FS_WEBCONFIG2_DISPLAY_NAME | webconfig2 | Displays the name of the NFS file system for the OHS configuration data on webhost2. |
| FS_WEBCONFIG2_PATH | /exports/IAMCONFIG/webconfig2 | Displays the path to the NFS file system where the OHS configuration data is installed on webhost2. |
| FS_OAMPV_DISPLAY_NAME | oampv | Displays the name of the OAM persistent volume file system. |
| FS_OAMPV_NFS_PATH | $NFS_IAMPV_EXPORT_PATH/oampv | Displays the path to the NFS file system for the OAM domain. |
| FS_OAMPV_LOCAL_MOUNTPOINT | /nfs_volumes/oampv | The local mount point on the bastion host for the OAM persistent volume. |
| FS_OIGPV_DISPLAY_NAME | oigpv | Displays the name of the OIG persistent volume file system. |
| FS_OIGPV_NFS_PATH | $NFS_IAMPV_EXPORT_PATH/oigpv | Displays the path to the NFS file system for the OIG domain. |
| FS_OIGPV_LOCAL_MOUNTPOINT | /nfs_volumes/oigpv | The local mount point on the bastion host for the OIG persistent volume. |
| FS_OUDPV_DISPLAY_NAME | oudpv | Displays the name of the OUD persistent volume file system. |
| FS_OUDPV_NFS_PATH | $NFS_IAMPV_EXPORT_PATH/oudpv | Displays the path to the NFS file system for the OUD domain. |
| FS_OUDPV_LOCAL_MOUNTPOINT | /nfs_volumes/oudpv | The local mount point on the bastion host for the OUD persistent volume. |
| FS_OUDCONFIGPV_DISPLAY_NAME | oudconfigpv | Displays the name of the OUD configuration persistent volume file system. |
| FS_OUDCONFIGPV_NFS_PATH | $NFS_IAMPV_EXPORT_PATH/oudconfigpv | Displays the path to the NFS file system for the OUD configuration data. |
| FS_OUDCONFIGPV_LOCAL_MOUNTPOINT | /nfs_volumes/oudconfigpv | The local mount point on the bastion host for the OUD configuration data. |
| FS_OUDSMPV_DISPLAY_NAME | oudsmpv | Displays the name of the OUD services manager persistent volume file system. |
| FS_OUDSMPV_NFS_PATH | $NFS_IAMPV_EXPORT_PATH/oudsmpv | Displays the path to the NFS file system for the OUD services manager domain. |
| FS_OUDSMPV_LOCAL_MOUNTPOINT | /nfs_volumes/oudsmpv | The local mount point on the bastion host for the OUD services manager persistent volume. |

**Table E-12 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| FS_OIRIPV_DISPLAY_NAME | `oiripv` | Displays the name of the OIRI persistent volume file system. |
| FS_OIRIPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/oiripv` | Displays the path to the NFS file system for the OIRI domain. |
| FS_OIRIPV_LOCAL_MOUNTPOINT | `/nfs_volumes/oiripv` | The local mount point on the bastion host for the OIRI persistent volume. |
| FS_DINGPV_DISPLAY_NAME | `dingpv` | Displays the name of the data ingestor persistent volume file system. |
| FS_DINGPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/dingpv` | Displays the path to the NFS file system for the OIRI ingestor data. |
| FS_DINGPV_LOCAL_MOUNTPOINT | `/nfs_volumes/dingpv` | The local mount point on the bastion host for the OIRI ingestor data. |
| FS_WORKPV_DISPLAY_NAME | `workpv` | Displays the name of the OIRI working directory volume file system. |
| FS_WORKPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/workpv` | Displays the path to the NFS file system for the OIRI working directory. |
| FS_WORKPV_LOCAL_MOUNTPOINT | `/nfs_volumes/workpv` | The local mount point on the bastion host for the OIRI working directory. |
| FS_OAACONFIGPV_DISPLAY_NAME | `oaaconfigpv` | Displays the name of the OAA configuration persistent volume file system. |
| FS_OAACONFIGPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/oaaconfigpv` | Path to the NFS file system for the OAA configuration data. |
| FS_OAACONFIGPV_LOCAL_MOUNTPOINT | `/nfs_volumes/oaaconfigpv` | The local mount point on the bastion host for the OA configuration persistent volume data. |
| FS_OAACREDPV_DISPLAY_NAME | `oaacredpv` | Displays the name of the OAA credential store persistent volume file system. |
| FS_OAACREDPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/oaacredpv` | Displays the path to the NFS file system for the OAA credential store data. |
| FS_OAACREDPV_LOCAL_MOUNTPOINT | `/nfs_volumes/oaacredpv` | The local mount point on the bastion host for the OAA credential store persistent volume data. |
| FS_OAAVAULTPV_DISPLAY_NAME | `oaavaultpv` | Displays the name of the OAA Vault persistent volume file system. |
| FS_OAAVAULTPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/oaavaultpv` | Displays the path to the NFS file system for the OAA Vault data. |
| FS_OAAVAULTPV_LOCAL_MOUNTPOINT | `/nfs_volumes/oaavaultpv` | The local mount point on the bastion host for the OAA Vault persistent volume data. |
| FS_OAALOGPV_DISPLAY_NAME | `oaalogpv` | Displays the name of the OAA log file persistent volume file system. |
| FS_OAALOGPV_NFS_PATH | `$NFS_IAMPV_EXPORT_PATH/oaalogpv` | Displays the path to the NFS file system for the OAA log files. |
| FS_OAALOGPV_LOCAL_MOUNTPOINT | `/nfs_volumes/oaalogpv` | The local mount point on the bastion host for the OAA log files. |
| FS_IMAGES_DISPLAY_NAME | `images` | Displays the name of the IDM container images persistent volume file system. |

**Table E-12 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| FS_IMAGES_NFS_PATH | `/exports/IMAGES/images` | Displays the path to the NFS file system for the container images. |
| FS_IMAGES_LOCAL_MOUNTPOINT | `/images` | The local mount point on the bastion host for the container images. |

# SSL Configuration

**Table E-13 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| SSL_CERT_VALIDITY_DAYS | `750` | Specifies the number of days to set for the SSL certificate validity. |
| SSL_CERT_BITS | `2048` | Specifies the default key size in bit for the SSL certificate. |
| SSL_CN | `*.$DNS_DOMAIN_NAME` | Specifies the domain name to set for the SSL certificate. |

# Load Balancer Configuration

**Table E-14 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| LBR1_AD | `ad1` | Displays the available domain to use for the first internal load balancer.<br>**Note:** This is not the actual availability domain name but a representation of the AD to use. For example: ad1, ad2, or ad3. |
| LBR1_DISPLAY_NAME | `lbr-subnet1` | Displays the name of the first internal load balancer. |
| LBR1_DNS_LABEL | `lbrsubnet1` | The DNS subnet label to use for the first internal load balancer. |
| LBR2_AD | `ad2` | Displays the available domain to use for the second internal load balancer.<br>**Note:** This is not the actual availability domain name but a representation of the AD to use. For example: ad1, ad2, or ad3. |
| LBR2_DISPLAY_NAME | `lbr-subnet2` | Displays the name of the second internal load balancer. |
| LBR2_DNS_LABEL | `lbrsubnet2` | The DNS subnet label to use for the second internal load balancer. |

# Load Balancer Log Group Configuration

**Table E-15 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| LBR_LOG_GROUP_NAME | `Default_Group` | Displays the name of the log group that will hold the access and error logs for the public and internal load balancers. |

# Public Load Balancer Configuration

**Table E-16 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| PUBLIC_LBR_ACCESS_LOG_DISPLAY_NAME | `public_loadbalancer_access` | Displays the name of the public load balancer access log file. |
| PUBLIC_LBR_ERROR_LOG_DISPLAY_NAME | `public_loadbalancer_error` | Displays the name of the public load balancer error log. |
| PUBLIC_LBR_CERTIFICATE_NAME | `loadbalancer` | Displays the name of the SSL certificate loaded into the public load balancer. |
| PUBLIC_LBR_DISPLAY_NAME | `public-loadbalancer` | Displays the name of the public load balancer. |
| PUBLIC_LBR_PRIVATE | `false` | Indicates if the public load balancer should only be assigned an internal IP address. |
| PUBLIC_LBR_ROUTE_TABLE_DISPLAY_NAME | `lbr-route-table` | Displays the name of the public load balancer route table. |
| PUBLIC_LBR_SECLIST_DISPLAY_NAME | `public-lbr-seclist` | Displays the name of the public load balancer security list. |
| PUBLIC_LBR_SHAPE | `flexible` | Displays the pubic load balancer shape configuration. |
| PUBLIC_LBR_SHAPE_DETAILS | `'{minimumBandwidthInMbps: 10, maximumBandwidthInMbps: 100}'` | Displays the minimum and maximum bandwidth values for the public load balancer. |
| PUBLIC_LBR_IADADMIN_DISPLAY_NAME | `iadadmin` | Displays the name of the OAM iadadmin host name on the public load balancer. |
| PUBLIC_LBR_IADADMIN_HOSTNAME | `iadadmin.$DNS_DOMAIN_NAME` | The host name for the OAM iadadmin host on the public load balancer. |
| PUBLIC_LBR_IADADMIN_LISTENER_DISPLAY_NAME | `iadadmin` | Displays the name of the OAM iadadmin listener on the public load balancer. |
| PUBLIC_LBR_IGDADMIN_DISPLAY_NAME | `igdadmin` | Displays the name of the OIM igdadmin hostname on the public load balancer. |
| PUBLIC_LBR_IGDADMIN_HOSTNAME | `igdadmin.$DNS_DOMAIN_NAME` | The host name for the OIM igdadmin host on the public load balancer. |

**Table E-16 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| PUBLIC_LBR_IGDADMIN_LISTENER_DISPLAY_NAME | `igdadmin` | Displays the name of the OIM igdadmin listener on the public load balancer. |
| PUBLIC_LBR_LOGIN_DISPLAY_NAME | `login` | Displays the name of the OAM login host on the public load balancer. |
| PUBLIC_LBR_LOGIN_HOSTNAME | `login.$DNS_DOMAIN_NAME` | The host name for the OAM login host on the public load balancer. |
| PUBLIC_LBR_LOGIN_LISTENER_DISPLAY_NAME | `login` | Displays the name of the OAM login listener on the public load balancer. |
| PUBLIC_LBR_PROV_DISPLAY_NAME | `prov` | Displays the name of the OIM provisioning host on the public load balancer. |
| PUBLIC_LBR_PROV_HOSTNAME | `prov.$DNS_DOMAIN_NAME` | The host name for the OIM provisioning host on the public load balancer. |
| PUBLIC_LBR_PROV_LISTENER_DISPLAY_NAME | `prov` | Displays the name of the OIM provisioning listener on the public load balancer. |
| PUBLIC_LBR_OHS_SERVERS_BS_NAME | `ohs_servers` | Displays the name of the backend set pointing to the OHS servers. |
| PUBLIC_LBR_OHS_SERVERS_BS_POLICY | `WEIGHTED_ROUND_ROBIN` | Displays the load balancing policy for the OHS servers back end set. |
| PUBLIC_LBR_OHS_SERVERS_BS_PROTOCOL | `HTTP` | The protocol used by the public load balancer health checker to determine if the load balancer is accessible. |
| PUBLIC_LBR_OHS_SERVERS_BS_URI_PATH | `/` | The URI used by the public load balancer health checker to determine if the load balancer is accessible. |

# Internal Load Balancer Configuration

**Table E-17 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| INT_LBR_ACCESS_LOG_DISPLAY_NAME | `internal_loadbalancer_access` | Displays the name of the internal load balancer access log file. |
| INT_LBR_ERROR_LOG_DISPLAY_NAME | `internal_loadbalancer_error` | Displays the name of the internal load balancer error log. |
| INT_LBR_CERTIFICATE_NAME | `loadbalancer` | Displays the name of the SSL certificate loaded into the internal load balancer. |
| INT_LBR_PRIVATE | `true` | Indicates if the internal load balancer should only be assigned an internal IP address. |
| INT_LBR_SHAPE | `flexible` | Displays the internal load balancer shape configuration. |

**Table E-17 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| INT_LBR_SHAPE_DETAILS | '{minimumBandwidthInMbps: 10, maximumBandwidthInMbps: 100}' | Displays the minimum and maximum bandwidth values for the internal load balancer. |
| INT_LBR_DISPLAY_NAME | internal-loadbalancer | Displays the name of the internal load balancer. |
| INT_LBR_IADADMIN_DISPLAY_NAME | $PUBLIC_LBR_IADADMIN_DISPLAY_NAME | Displays the name of the OAM iadadmin host name on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IADADMIN_DISPLAY_NAME. |
| INT_LBR_IADADMIN_HOSTNAME | $PUBLIC_LBR_IADADMIN_HOSTNAME | The host name for the OAM iadadmin host on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IADADMIN_HOSTNAME. |
| INT_LBR_IADADMIN_LISTENER_DISPLAY_NAME | $PUBLIC_LBR_IADADMIN_LISTENER_DISPLAY_NAME | Displays the name of the OAM iadadmin listener on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IADADMIN_LISTENER_DISPLAY_NAME. |
| INT_LBR_IGDADMIN_DISPLAY_NAME | $PUBLIC_LBR_IGDADMIN_DISPLAY_NAME | Displays the name of the OIM igdadmin host name on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IGDADMIN_DISPLAY_NAME. |
| INT_LBR_IGDADMIN_HOSTNAME | $PUBLIC_LBR_IGDADMIN_HOSTNAME | The host name for the OIM igdadmin host on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IGDADMIN_HOSTNAME. |
| INT_LBR_IGDADMIN_LISTENER_DISPLAY_NAME | $PUBLIC_LBR_IGDADMIN_LISTENER_DISPLAY_NAME | Displays the name of the OIM igdadmin listener on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IGDADMIN_LISTENER_DISPLAY_NAME. |
| INT_LBR_IGDINTERNAL_DISPLAY_NAME | $PUBLIC_LBR_IGDINTERNAL_DISPLAY_NAME | Displays the name of the OIM igdinternal host name on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IGDINTERNAL_DISPLAY_NAME. |
| INT_LBR_IGDINTERNAL_HOSTNAME | $PUBLIC_LBR_IGDINTERNAL_HOSTNAME | The host name for the OIM igdinternal host on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IGDINTERNAL_HOSTNAME. |
| INT_LBR_IGDINTERNAL_LISTENER_DISPLAY_NAME | $PUBLIC_LBR_IGDINTERNAL_LISTENER_DISPLAY_NAME | Displays the name of the OIM igdinternal listener on the internal load balancer.<br>**Note:** This must be the same name as PUBLIC_LBR_IGDINTERNAL_LISTENER_DISPLAY_NAME. |

**Table E-17 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| INT_LBR_LOGIN_DISPLAY_NAME | `$PUBLIC_LBR_LOGIN_DISPLAY_NAME` | Displays the name of the OAM login host on the internal load balancer.<br>**Note:** This must be the same name as `PUBLIC_LBR_LOGIN_DISPLAY_NAME`. |
| INT_LBR_LOGIN_HOSTNAME | `$PUBLIC_LBR_LOGIN_HOSTNAME` | The host name for the OAM login host on the internal load balancer.<br>**Note:** This must be the same name as `PUBLIC_LBR_LOGIN_HOSTNAME`. |
| INT_LBR_LOGIN_LISTENER_DISPLAY_NAME | `$PUBLIC_LBR_LOGIN_LISTENER_DISPLAY_NAME` | Displays the name of the OAM login listener on the internal load balancer.<br>**Note:** This must be the same name as `PUBLIC_LBR_LOGIN_LISTENER_DISPLAY_NAME`. |
| INT_LBR_PROV_DISPLAY_NAME | `$PUBLIC_LBR_PROV_DISPLAY_NAME` | Displays the name of the OIM provisioning host on the internal load balancer.<br>**Note:** This must be the same name as `PUBLIC_LBR_PROV_DISPLAY_NAME`. |
| INT_LBR_PROV_HOSTNAME | `$PUBLIC_LBR_PROV_HOSTNAME` | The host name for the OIM provisioning host on the internal load balancer.<br>**Note:** This must be the same name as `PUBLIC_LBR_PROV_HOSTNAME`. |
| INT_LBR_PROV_LISTENER_DISPLAY_NAME | `$PUBLIC_LBR_PROV_LISTENER_DISPLAY_NAME` | Displays the name of the OIM provisioning host on the internal load balancer.<br>**Note:** This must be the same name as `PUBLIC_LBR_PROV_LISTENER_DISPLAY_NAME`. |
| INT_LBR_OHS_SERVERS_BS_NAME | `ohs_servers` | Displays the name of the back end set pointing to the OHS servers. |
| INT_LBR_OHS_SERVERS_BS_POLICY | `WEIGHTED_ROUND_ROBIN` | The load balancing policy for the OHS servers back end set. |
| INT_LBR_OHS_SERVERS_BS_PROTOCOL | `HTTP` | The protocol used by the public load balancer health checker to determine if the load balancer is accessible. |
| INT_LBR_OHS_SERVERS_BS_URI_PATH | `/` | The URI used by the public load balancer health checker to determine if the load balancer is accessible. |

# Network Load Balancer Configuration

**Table E-18 List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| K8_LBR_DISPLAY_NAME | `k8workers` | Displays the name of the Kuberentes load balancer. |
| K8_LBR_PRIVATE | `true` | Indicates if the Kubernetes load balancer should only be assigned an internal IP address. |

**Table E-18 (Cont.) List of Parameters**

| Parameter | Default Value | Comments |
|---|---|---|
| K8_LBR_PRESERVE_SRC_DEST | `false` | Indicates if the requests should be sent with the entire IP header intact. |
| K8_LBR_K8_WORKERS_BS_NAME | `kubernetes_workers` | Displays the name of the back end set pointing to the OHS servers. |
| K8_LBR_K8_WORKERS_BS_POLICY | `FIVE_TUPLE` | The Kubernetes load balancer policy for the back end set. |
| K8_LBR_K8_WORKERS_BS_PRESERVE_SRC | `true` | Indicates if the requests should be sent with the entire IP header intact. |
| K8_LBR_LISTENER_DISPLAY_NAME | `k8workers` | Displays the name of the Kuberentes load balancer listener. |

# OCI Tags Configuration

The following section lists the parameters to be defined to create Freeform OCI tags that would be associated with some of the hardware provisioned by the framework.

**Table E-19 List of Parameters**

| Parameter | Value | Comments |
|---|---|---|
| OCI_TAGS | `CreatedBy:ABCD WXYZ` | The Value will have a "key:value" pair which is separated by colon (:). The key cannot include a double-quote, colon, space and all other characters which are defined as restricted in the Freeform tags OCI page.<br>In case multiple tags are needed, please repeat the line in the response file.<br>For example,<br>OCI_TAGS="CreatedBy:ABCD WXYZ"<br>OCI_TAGS="ProjectNo:PR1234" |

# Database Configuration

**Table E-20 List of Parameters**

| Parameter | Value | Comments |
|---|---|---|
| DB_SUBNET_DISPLAY_NAME | `db-subnet` | Displays the name of the DB Subnet. |
| DB_SUBNET_DNS_LABEL | `dbsubnet` | The DNS label for the DB subnet. |
| DB_SUBNET_PROHIBIT_PUBLIC_IP | `true` | Indicates if the DB subnet allows computing instances with a public IP address. |

**Table E-20 (Cont.) List of Parameters**

| Parameter | Value | Comments |
|---|---|---|
| DB_SECLIST_DISPLAY_NAME | `db-seclist` | Displays the name of the DB security list. |
| DB_ROUTE_TABLE_DISPLAY_NAME | `db-route-table` | Displays the name of the DB route table. |
| DB_SQLNET_PORT | `1521` | DB listener port. |
| DB_AD | `ad1` | Displays the name of the available domain to use for the Database<br><br>**Note:**<br>This value is not the actual availability domain name but a representation of the AD to use. For example, ad1, ad2, or ad3. |
| DB_CPU_COUNT | `8` | CPU count to be used for creating the DB. |
| DB_EDITION | `ENTERPRISE_EDITION_EXTREME_PERFORMANCE` | The DB service edition to be used for creation of DB. |

**Table E-20 (Cont.) List of Parameters**

| Parameter | Value | Comments |
|---|---|---|
| DB_HOSTNAME_PREFIX | edgdb- | Prefix to be used to create DB hosts. |
| DB_VERSION | 19.0.0.0 | Version of DB to be installed. |
| DB_DISPLAY_NAME | Identity-Management-Database | Displays the name of the DB. |
| DB_INITIAL_STORAGE | 256 | Storage size of the DB in GBs. |
| DB_LICENSE | BRING_YOUR_OWN_LICENSE | DB License type. |
| DB_NODE_COUNT | 2 | Number of DB Hosts required. |
| DB_SHAPE | $DEFAULT_HOST_SHAPE | Displays the image shape to use for the first Web host. |
| DB_STORAGE_MGMT | ASM | Storage Management type to be used for DB. |
| DB_TIMEZONE | UTC | Timezone to be used for the DB. |

# Components of the Deployment Scripts

For reference purposes, this section includes the file name, directory, and purpose of all the objects that make up the deployment scripts.

**Table E-21 Components of the Deployment Scripts**

| File Name | Directory | Purpose |
|---|---|---|
| provision_oci.sh | $SCRIPT_DIR/oke_utils | The main provisioning script. |
| delete_oci.sh | $SCRIPT_DIR/oke_utils | Script to delete all of the resources created by the provisioning script. This script requires the following command to be available to read the list of resource OCIDs to delete.<br><br>$WORKDIR/<TEMPLATE_NAME>/output/<TEMPLATE_NAME>.ocid |
| delete_oci.sh | $SCRIPT_DIR/oke_utils | Script to create a response file for the IDM EDG Automation. |
| oci_create_functions.sh | $SCRIPT_DIR/oke_utils/common | Helper script that contains all of the shell functions used by the provision_oke.sh script. |
| oci_delete_functions.sh | $SCRIPT_DIR/oke_utils/common | Helper script that contains all of the shell functions used by the delete_oke.sh script. |
| oci_setup_functions.sh | $SCRIPT_DIR/oke_utils/common | Helper script that contains all of the functions to setup/configure the Bastion host, WebTiers, and database. |
| oci_util_functions.sh | $SCRIPT_DIR/oke_utils/common | Helper script that contains all of the functions shared between the provisioning and deletion scripts. |

**Table E-21 (Cont.) Components of the Deployment Scripts**

| File Name | Directory | Purpose |
|---|---|---|
| oci_oke.rsp | $SCRIPT_DIR/oke_utils/responsefile | An example response file that is used as a starting point for end-user response files. |
| ocipwd | $SCRIPT_DIR/oke_utils/responsefile | An example response file that is used as a starting point for end-user response files containing sensitive information. |
| oci_setup_bastion.sh | $SCRIPT_DIR/oke_utils/util | Helper script that can be executed manually to configure the Bastion host. The script executes the same functions that are called when the CONFIGURE_BASTION parameter is enabled. |
| oci_setup_webhosts.sh | $SCRIPT_DIR/oke_utils/util | Helper script that can be executed manually to configure the WebTier hosts. The script executes the same functions that are called when the CONFIGURE_WEBHOSTS parameter is enabled. |
| oci_setup_database.sh | $SCRIPT_DIR/oke_utils/util | Helper script that can be executed manually to configure the RAC database. The script executes the same functions that are called when the CONFIGURE_DATABASE parameter is enabled. |

F

# Automating the Disaster Recovery Setup

In conjunction with the example scripts provided to automate the Identity and Access Management Enterprise deployment, example scripts have been developed to automate the disaster recovery setup. These scripts are also in the same download location as the deployment scripts. For information about automating the deployment, see Automating the Identity and Access Management Enterprise Deployment.
This appendix includes the following topics:

- Disaster Recovery Utilities
  There are two additional scripts provided for disaster recovery. Located in the `utils` directory, these are `enable_dr.sh` and `idmdrctl.sh`.
- Creating the Response File
  A sample response file is available for your use in the `responsefile` directory. You can use the disaster recovery utilities even if you created the environment using the deployment automation scripts.
- Response File Reference
  The parameters in the response file are used to control the provisioning of the various products in the Kubernetes cluster. These parameters are divided into generic and product-specific parameters.
- Log Files

## Disaster Recovery Utilities

There are two additional scripts provided for disaster recovery. Located in the `utils` directory, these are `enable_dr.sh` and `idmdrctl.sh`.

More information about these scripts:

- **`enable_dr.sh`**: Used to set up disaster recovery, the script takes one argument: the name of the product for which you want to enable disaster recovery. The valid values are `oud`, `oam`, `oig`, `oiri`, and `oaa`. This script is run once per site, and creates the `rsync` job for file system replication, as well as the Kubernetes objects, if required.
- **`idmdrctl.sh`**: Used to control the disaster recovery actions, it takes two arguments— `-a` action and `-p` product.
  - **Actions**:
    - `initial`: Manually run the `rsync` job to backup or restore the persistent volume state.
    - `switch`: Change a sites role from standby to primary or primary to standby.
    - `stop`: Shut down a product.
    - `start`: Start up a product.
    - `suspend`: Suspend the `rsync` job for the file system replication.
    - `resume`: Resume the `rsync` job for the file system replication.
  - **Products**:

* oud
* oam
* oig
* oiri
* oaa

Both the scripts rely on the response file `dr.rsp` located in the `responsefile` directory. With the exception of the site role, the `dr.rsp` file will be identical on both the sites. Therefore, no need to switch the primary/standby values on the different sites.

# Creating the Response File

A sample response file is available for your use in the `responsefile` directory. You can use the disaster recovery utilities even if you created the environment using the deployment automation scripts.

> **Note:**
>
> The property values between the `dr.rsp` and `spider` files overlap. Passwords are stored in a hidden file `dewdrop` available in the `responsefile` directory.

# Response File Reference

The parameters in the response file are used to control the provisioning of the various products in the Kubernetes cluster. These parameters are divided into generic and product-specific parameters.



## Products to Deploy

These parameters determine which products the deployment scripts attempt to deploy.

**Table F-1 List of Products to Deploy**

| Parameter | Sample Value | Comments |
|---|---|---|
| DR_OUD | true | Set the value to true to configure OUD. |
| DR_OAM | true | Set the value to true to configure OAM. |
| DR_OIG | true | Set the value to true to configure OIG. |
| DR_OIRI | true | Set the value to true to configure OIRI. |
| DR_OAA | true | Set the value to true to configure OAA. |

# Control Parameters

These parameters are used to specify the type of Kubernetes deployment and the names of the temporary directories you want the deployment to use, during the provisioning process.

**Table F-2 List of Control Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| USE_REGISTRY | false | Set this value to true to pull images from a container registry. |
| USE_INGRESS | false | Set to true if you are using an Ingress controller. |
| ENV_TYPE | OTHER | The valid values are: OCI and OTHER. If OCI is selected, then OCI snapshots will be used to create the PV backups. If OTHER is selected, then rsync will be used to create the backups. |
| USE_MAA_SCRIPTS | true | If set to true, the MAA scripts will be used to take and restore a snapshot of the Kubernetes objects where appropriate. Set this value to false if you are creating the standby environment by rerunning the install scripts. The recommended value is true because using MAA scripts is the most efficient approach. |
| COPY_FILES_TO_DR | true | If set to true, then the enable_dr.sh script will attempt to copy the backup files to DR_HOST. |
| DR_HOST | bastionhost | The host on the DR system to which you want to copy the files. The enable_dr.sh script will attempt to copy the files to this host if COPY_FILES_TO_DR is set to true. |
| DR_USER | opc | The name of the user who copies the backup files to DR_HOST. |

# Registry Parameters

These parameters are used to determine whether or not you are using a container registry. If you are, then it allows you to store the login credentials to the repository so that you are able to store the credentials as registry secrets in the individual product namespaces.

If you are pulling images from GitHub or Docker hub, then you can also specify the login parameters here so that you can create the appropriate Kubernetes secrets.

**Table F-3 List of Registry Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| `REGISTRY` | `iad.ocir.io/mytenancy` | Set to the location of the container registry. |
| `REG_USER` | `mytenancy/oracleidentitycloudservice/email@example.com` | Set to the registry user name. |
| `CREATE_REGSECRET` | `false` | Set this value to `true` to create a registry secret for automatically pulling images. |
| `LOCAL_WORKDIR` | `/workdir` | The location where you want to create the working directory. |
| `K8_DRDIR` | `/u01/oracle/user_projects/dr_scripts` | The location inside the container where the disaster recovery script resides. This script is used to backup/restore the persistent volumes. |
| `MAA_SAMPLES_REP` | `https://github.com/oracle-samples/maa` | The GitHub location from where you can download the MAA Kubernetes snapshot tool. |

# Image Parameters

These parameters are used to specify the names and versions of the container images you want to use for the deployment. These images must be available either locally or in your container registry. The names and versions must be identical to the images in the registry or the images stored locally.

These can include registry prefixes if you use a registry. Use the `local/` prefix if you use the Oracle Cloud Native Environment.

**Table F-4 List of Image Parameters in the Response File**

| Parameter | Sample Value | Comments |
|---|---|---|
| `RSYNC_IMAGE` | `$REGISTRY/alpine-rsync` | The name of the `rsync` image you created earlier. See Creating a Container with rsync. |
| `RSYNC_VER` | `latest` | The version of the `rsync` image you created. |

# DR Parameters

These parameters are used to determine the type of site for disaster recovery.

**Table F-5 Parameters that Determine the Type of Site for Disaster Recovery**

| Parameter | Sample Value | Comments |
|---|---|---|
| DR_TYPE | PRIMARY or STANDBY | The initial role of the site you want to create. |
| DRNS | drns | The namespace used to place the disaster recovery rsync job. |

# NFS Parameters

**Table F-6 List of Parameters**

| Parameter | Sample Value | Comments |
|---|---|---|
| DR_PRIMARY_NFS_EXPORT | /export/IAMPVS | The export path on the primary NFS server where the persistent volumes are located. |
| DR_PRIMARY_PVSERVER | primarynfsserver.example.com | The name or IP address of the primary NFS server used for the persistent volumes. |
| DR_STANDBY_NFS_EXPORT | /export/IAMPVS | The export path on the standby NFS server where the persistent volumes are located. |
| DR_STANDBY_PVSERVER | standbynfsserver.example.com | The name or IP address of the standby NFS server used for the persistent volumes. |

# OUD Parameters

These parameters are specific to OUD. When deploying OUD, you also require the generic LDAP parameters.

**Table F-7 OUD Parameters that Determine the Deployment of Oracle Unified Directory**

| Parameter | Sample Value | Comments |
|---|---|---|
| OUDNS | oudns | The Kubernetes namespace used to hold the OUD objects. |
| OUD_POD_PREFIX | edg | The prefix used for the OUD pods. |
| OUD_REPLICAS | 1 | The number of OUD replicas to create. If you require two OUD instances, set this to 1. This value is in addition to the primary instance. |
| OUD_PRIMARY_SHARE | $DR_PRIMARY_NFS_EXPORT/oudpv | The mount point on the primary NFS server where the OUD persistent volume is exported. |

**Table F-7 (Cont.) OUD Parameters that Determine the Deployment of Oracle Unified Directory**

| Parameter | Sample Value | Comments |
|---|---|---|
| OUD_PRIMARY_CONFIG_SHARE | $DR_PRIMARY_NFS_EXPORT/oudconfigpv | The mount point on the primary NFS server where the OUD configuration persistent volume is exported. |
| OUD_STANDBY_SHARE | $DR_STANDBY_NFS_EXPORT/oudpv | The mount point on the standby NFS server where the OUD persistent volume is exported. |
| OUD_STANDBY_CONFIG_SHARE | $DR_STANDBY_NFS_EXPORT/oudconfigpv | The mount point on the standby NFS server where the OUD configuration persistent volume is exported. |
| OUD_LOCAL_CONFIG_SHARE | /exports/IAMPVS/oudconfigpv | The NFS mount point for the OUD configuration persistent volume. |
| OUD_LOCAL_SHARE | /nfs_volumes/oudconfigpv | The local directory where OUD_LOCAL_CONFIG_SHARE is mounted. Used to hold seed files. |
| DR_OUD_MINS | 5 | The frequency at which the rsync job runs. |
| DR_CREATE_OUD_JOB | true | Determines whether or not to create an rsync job for OUD. |

# OHS Parameters

Oracle HTTP Server (OHS) parameters are used to formulate how sample OHS configuration files are created. They also control whether you want the Oracle HTTP Server files to be propagated to the Oracle HTTP Server hosts automatically. If you choose automatic propagation, you should ensure that a passwordless SSL is possible from the deployment host to the Oracle HTTP Servers.

**Table F-8 Parameters Used by Oracle HTTP Server to Create Sample OHS Configuration Files**

| Parameter | Sample Value | Comments |
|---|---|---|
| OHS_BASE | /u02/private | The location of the OHS base directory. The binaries and the configuration files are below this location.<br>The Oracle inventory is also placed in this location when installing the Oracle HTTP Server. |
| OHS_ORACLE_HOME | $OHS_BASE/oracle/products/ohs | The location of the OHS binaries. |

**Table F-8 (Cont.) Parameters Used by Oracle HTTP Server to Create Sample OHS Configuration Files**

| Parameter | Sample Value | Comments |
|---|---|---|
| OHS_USER | <user name> | The name of the user you want to assign to the Node Manager if you install the Oracle HTTP Server. |
| OHS_HOST1 | webhost1.example.com | The fully qualified name of the host running the first Oracle HTTP Server. |
| OHS1_NAME | ohs1 | The component name of the first OHS instance (on OHS_HOST1). |
| OHS_HOST2 | webhost2.example.com | The fully qualified name of the host running the second Oracle HTTP Server. Leave it blank if you do not have a second Oracle HTTP Server. |
| OHS2_NAME | ohs2 | The component name of the second OHS instance (on OHS_HOST2). |
| OHS_DOMAIN | $OHS_BASE/oracle/config/domains/ohsDomain | The location of the OHS domain on OHS_HOST1 and OHS_HOST2. |

# OAM Parameters

These parameters determine how Oracle Access Manager (OAM) is deployed and configured.

**Table F-9 Parameters that Determine the Deployment of Oracle Access Manager**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAMNS | oamns | The Kubernetes namespace used to hold the OAM objects. |
| OAM_DOMAIN_NAME | accessdomain | The name of the OAM domain you want to create. |
| OAM_PRIMARY_SHARE | $DR_PRIMARY_NFS_EXPORT/oampv | The mount point on the primary NFS server where the OAM persistent volume is exported. |
| OAM_STANDBY_SHARE | $DR_STANDBY_NFS_EXPORT/oampv | The mount point on the standby NFS server where the OAM persistent volume is exported. |
| OAM_LOCAL_SHARE | /nfs_volumes/oampv | The local directory where OAM_PRIMARY_SHARE is mounted. |
| OAM_SERVER_INITIAL | 2 | The number of OAM Managed Servers you want to start for normal running. You will need at least two servers for high availability. |

**Table F-9 (Cont.) Parameters that Determine the Deployment of Oracle Access Manager**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAM_PRIMARY_DB_SCAN | primary-dbscan.example.com | The database scan address of the primary database. |
| OAM_PRIMARY_DB_SERVICE | iadedg.example.com | The database service of the primary database. |
| OAM_STANDBY_DB_SCAN | standby-dbscan.example.com | The database scan address of the standby database. |
| OAM_STANDBY_DB_SERVICE | iadedg.example.com | The database service of the standby database. |
| OAM_DB_LISTENER | 1521 | The database listener port. |
| DR_OAM_MINS | 720 | The frequency at which the rsync job runs. |
| DR_CREATE_OAM_JOB | true | Determines whether or not to create an rsync job for OAM. |

# OIG Parameters

These parameters determine how Oracle Identity Governance (OIG) is deployed and configured.

**Table F-10 Parameters that Determine the Deployment of Oracle Identity Governance**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIGNS | oigns | The Kubernetes namespace used to hold the OIG objects. |
| OIG_DOMAIN_NAME | governancedomain | The name of the OIG domain you want to create. |
| OIG_PRIMARY_SHARE | $DR_PRIMARY_NFS_EXPORT /oigpv | The mount point on the primary NFS server where the OIG persistent volume is exported. |
| OIG_STANDBY_SHARE | $DR_STANDBY_NFS_EXPORT /oigpv | The mount point on the standby NFS server where the OIG persistent volume is exported. |
| OIG_LOCAL_SHARE | /nfs_volumes/oigpv | The local directory where OIG_PRIMARY_SHARE is mounted. |
| OIG_SERVER_INITIAL | 2 | The number of OIG Managed Servers you want to start for normal running. You will need at least two servers for high availability. |
| OIG_PRIMARY_DB_SCAN | primary-dbscan.example.com | The database scan address of the primary database. |
| OIG_PRIMARY_DB_SERVICE | igdedg.example.com | The database service of the primary database. |

**Table F-10 (Cont.) Parameters that Determine the Deployment of Oracle Identity Governance**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIG_STANDBY_DB_SCAN | standby-dbscan.example.com | The database scan address of the standby database. |
| OIG_STANDBY_DB_SERVICE | igdedg.example.com | The database service of the standby database. |
| OIG_DB_LISTENER | 1521 | The database listener port. |
| DR_OIG_MINS | 720 | The frequency at which the rsync job runs. |
| DR_CREATE_OIG_JOB | true | Determines whether or not to create an rsync job for OIG. |

# OIRI Parameters

These parameters determine how Oracle Identity Role Intelligence (OIRI) is provisioned and configured.

**Table F-11 Parameters that Determine the Deployment of Oracle Identity Role Intelligence**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIRINS | oirins | The Kubernetes namespace used to hold the OIRI objects. |
| DINGNS | dingns | The Kubernetes namespace used to hold the OIRI DING objects. |
| OIRI_PRIMARY_SHARE | $DR_PRIMARY_NFS_EXPORT /oiripv | The mount point on the primary NFS server where the OIRI persistent volume is exported. |
| OIRI_STANDBY_SHARE | $DR_STANDBY_NFS_EXPORT /oiripv | The mount point on the standby NFS server where the OIRI persistent volume is exported. |
| OIRI_DING_PRIMARY_SHARE | $DR_PRIMARY_NFS_EXPORT /dingpv | The mount point on the primary NFS server where the OIRI Ding persistent volume is exported. |
| OIRI_DING_STANDBY_SHARE | $DR_STANDBY_NFS_EXPORT /dingpv | The mount point on the standby NFS server where the OIRI Ding persistent volume is exported. |
| OIRI_WORK_PRIMARY_SHARE | $DR_PRIMARY_NFS_EXPORT /workpv | The mount point on the primary NFS server where the OIRI Work persistent volume is exported. |
| OIRI_WORK_STANDBY_SHARE | $DR_STANDBY_NFS_EXPORT /workpv | The mount point on the standby NFS server where the OIRI Work persistent volume is exported. |
| OIRI_LOCAL_SHARE | /nfs_volumes/oiripv | The local directory where OIRI_PRIMARY_SHARE is mounted. |

**Table F-11 (Cont.) Parameters that Determine the Deployment of Oracle Identity Role Intelligence**

| Parameter | Sample Value | Comments |
|---|---|---|
| OIRI_DING_LOCAL_SHARE | /nfs_volumes/dingpv | The local directory where OIRI_DING_PRIMARY_SHARE is mounted. |
| OIRI_WORK_LOCAL_SHARE | /nfs_volumes/workpv | The local directory where OIRI_WORK_PRIMARY_SHARE is mounted. |
| OIRI_PRIMARY_DB_SCAN | primary-dbscan.example.com | The database scan address of the primary database. |
| OIRI_PRIMARY_DB_SERVICE | oiriedg.example.com | The database service of the primary database. |
| OIRI_STANDBY_DB_SCAN | standby-dbscan.example.com | The database scan address of the standby database. |
| OIRI_STANDBY_DB_SERVICE | oiriedg.example.com | The database service of the standby database. |
| OIRI_DB_LISTENER | 1521 | The database listener port. |
| DR_OIRI_MINS | 720 | The frequency at which the rsync job runs. |
| OIRI_PRIMARY_K8CONFIG | primary_k8config | The name of the Kubernetes configuration file for the primary Kubernetes cluster. |
| OIRI_STANDBY_K8CONFIG | standby_k8config | The name of the Kubernetes configuration file for the standby Kubernetes cluster. |
| OIRI_PRIMARY_K8CA | primary_ca.crt | The name of the Kubernetes certificate authority file for the primary Kubernetes cluster. |
| OIRI_STANDBY_K8CA | standby_ca.crt | The name of the Kubernetes certificate authority file for the standby Kubernetes cluster. |
| OIRI_PRIMARY_K8 | 10.0.0.5:6443 | The host and port of the Kubernetes primary cluster (obtained from the kubeconfig file). |
| OIRI_STANDBY_K8 | 10.1.0.10:6443 | The host and port of the Kubernetes standby cluster (obtained from the kubeconfig file). |
| DR_CREATE_OIRI_JOB | true | Determines whether or not to create an rsync job for OIRI. |

# OAA Parameters

These parameters determine how Oracle Advanced Authentication (OAA) is provisioned and configured.

**Table F-12 Parameters that Determine the Deployment of Oracle Advanced Authentication**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAANS | oaans | The Kubernetes namespace used to hold the OAA objects. |
| OAA_MGT_IMAGE | $REGISTRY/oracle/ shared/ oaa-mgmt | The OAA Management container image. |
| OAAMGT_VER | 12.2.1.4.1_20220419 | The version of the image you want to use. |
| OAA_PRIMARY_CONFIG_SHARE | $DR_PRIMARY_NFS_EXPORT /oaaconfigpv | The mount point on the primary NFS server where the OAA configuration persistent volume is exported. |
| OAA_STANDBY_CONFIG_SHARE | $DR_STANDBY_NFS_EXPORT /oaaconfigpv | The mount point on the standby NFS server where the OAA configuration persistent volume is exported. |
| OAA_PRIMARY_CRED_SHARE | $DR_PRIMARY_NFS_EXPORT /oaacredpv | The mount point on the primary NFS server where the OAA credential store persistent volume is exported. |
| OAA_STANDBY_CRED_SHARE | $DR_STANDBY_NFS_EXPORT /oaacredpv | The mount point on the standby NFS server where the OAA credential store persistent volume is exported. |
| OAA_PRIMARY_LOG_SHARE | $DR_PRIMARY_NFS_EXPORT /oaalogpv | The mount point on the primary NFS server where the OAA logs persistent volume is exported. |
| OAA_STANDBY_LOG_SHARE | $DR_STANDBY_NFS_EXPORT /oaalogpv | The mount point on the standby NFS server where the OAA logs persistent volume is exported. |
| OAA_PRIMARY_VAULT_SHARE | $DR_PRIMARY_NFS_EXPORT /oaavaultpv | The mount point on the primary NFS server where the OAA vault persistent volume is exported. |
| OAA_STANDBY_VAULT_SHARE | $DR_STANDBY_NFS_EXPORT /oaavaultpv | The mount point on the standby NFS server where the OAA vault persistent volume is exported. |
| OAA_LOCAL_CONFIG_SHARE | /nfs_volumes/ oaaconfigpv | The local directory where OAA_PRIMARY_CONFIG_SHARE is mounted. |
| OAA_LOCAL_CRED_SHARE | /nfs_volumes/oaacredpv | The local directory where OAA_PRIMARY_CRED_SHARE is mounted. |
| OAA_LOCAL_LOG_SHARE | /nfs_volumes/oaalogpv | The local directory where OAA_PRIMARY_LOG_SHARE is mounted. |
| OAA_LOCAL_VAULT_SHARE | /nfs_volumes/ oaavaultpv | The local directory where OAA_PRIMARY_VAULT_SHARE is mounted. |

**Table F-12 (Cont.) Parameters that Determine the Deployment of Oracle Advanced Authentication**

| Parameter | Sample Value | Comments |
|---|---|---|
| OAA_LOCAL_SHARE | $OAA_LOCAL_CONFIG_SHARE | The local directory where OAA_PRIMARY_CONFIG_SHARE is mounted. Do not change this value. |
| OAA_VAULT_TYPE | file\|oci | The type of vault to use: file system or OCI. |
| OAA_REPLICAS | 2 | The number of each OAA pods to start when invoked from idmdrctl. |
| OAA_PRIMARY_DB_SCAN | primary-dbscan.example.com | The database scan address of the primary database. |
| OIRI_PRIMARY_DB_SERVICE | oaaedg.example.com | The database service of the primary database. |
| OAA_STANDBY_DB_SCAN | standby-dbscan.example.com | The database scan address of the standby database. |
| OAA_STANDBY_DB_SERVICE | oaaedg.example.com | The database service of the standby database. |
| DR_OAA_MINS | 720 | The frequency at which the rsync job runs. |
| DR_CREATE_OAA_JOB | true | Determines whether or not to create an rsync job for OIRI. |

# Log Files

The DR scripts create log files for each product inside the working directory in a sub-directory called DR within a sub-directory called logs. For example: /workdir/OAM/DR/logs.

This directory also contains the following files:

- progressfile – This file contains the last successfully executed step. If you want to restart the process at a different step, update this file.
- timings.log – This file is used for informational purposes to show how much time was spent on each stage of the disaster recovery process.