Oracle® Cloud

Using Oracle WebLogic Server for OKE

F48259-50
May 2024

Oracle Cloud Using Oracle WebLogic Server for OKE,

F48259-50

Copyright © 2020, 2024, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

3 Manage WebLogic Domains

4 Manage WebLogic Domains in Verrazzano

5 Managing Resources

Preface

Using Oracle WebLogic Server for OKE explains how to create a stack in Oracle Cloud Infrastructure Marketplace, and then use the Jenkins CI/CD pipeline jobs to create a domain for developing Java Enterprise Edition (Java EE) applications, and deploy and integrate these applications to build the domain.

Topics:

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs 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.

1

Get Started

Learn about the architecture and features of Oracle WebLogic Server for Oracle Cloud Infrastructure Container Engine for Kubernetes (Oracle WebLogic Server for OKE), and perform any prerequisite tasks.

> **Note:**
>
> If you are using Oracle WebLogic Server for OKE (**Release 21.3.2 or earlier**), see Using Oracle WebLogic Server for OKE (Release 21.3.2 or earlier).

Topics:

About Oracle WebLogic Server for OKE

Use Oracle WebLogic Server for OKE to quickly create your Java Enterprise Edition (Java EE) application environment in Oracle Cloud Infrastructure in a fraction of the time it would normally take on-premises.

Oracle WebLogic Server for OKE is available as a set of applications in the Oracle Cloud Infrastructure Marketplace. After launching one of these applications, you use a simple wizard interface to configure a stack along with any supporting cloud resources like Kubernetes clusters, file systems, administration instances, networks and private load balancers. Then, use a Jenkins CI/CD pipeline job to create a domain with the required resources.

Oracle WebLogic Server for OKE uses multi-domain source type. Multi-domain enables you to create multiple domains in a single Kubernetes cluster with the ability to automatically patch the domains. You can create additional domains in the existing Kubernetes cluster using Jenkins CI/CD pipeline jobs

You can track and monitor the progress of an Oracle WebLogic Server for OKE stack in Resource Manager. A stack also provides a convenient method of deleting the cloud resources for a domain when you no longer require them.

After creating an Oracle WebLogic Server stack, you can use various tools in Oracle WebLogic Server for OKE to update the domain configuration and deploy your applications. When you apply any domain changes to a Kubernetes cluster, it deletes the existing pods and creates a new one.

Oracle WebLogic Server for OKE can install Verrazzano on the Kubernetes cluster node pool. See Configure Verrazzano.

Oracle WebLogic Server for OKE uses Jenkins to automate the creation of custom images for your WebLogic Server domain, and the deployment of these images to the Kubernetes cluster. See Jenkins.

Oracle WebLogic Server for OKE supports these Oracle WebLogic Server editions:

- Oracle WebLogic Server Enterprise Edition
- Oracle WebLogic Suite

Oracle WebLogic Server for OKE supports these Oracle WebLogic Server releases:

- Oracle WebLogic Server 14c (14.1.1.0) - See Understanding Oracle WebLogic Server
- Oracle WebLogic Server 12c (12.2.1.4) - See Understanding Oracle WebLogic Server

Oracle WebLogic Server for OKE supports these billing options:

- Universal Credits (also called UCM), where you are billed for the cost of the Oracle WebLogic Server Enterprise Edition or Oracle WebLogic Suite license (based on OCPUs per hour), for VMs running in the WebLogic node pool, in addition to the cost of the compute resources.
- Bring Your Own License (BYOL), which allows you to reuse your existing on-premise Oracle WebLogic Server Enterprise Edition and Oracle WebLogic Suite licenses in Oracle Cloud.

About the Components of Oracle WebLogic Server for OKE

Learn about the Oracle Cloud Infrastructure components that comprise Oracle WebLogic Server for OKE.

Topics

- Oracle WebLogic Server
- JRF Domain
- Marketplace
- Resource Manager
- Container Engine for Kubernetes
- Registry
- WebLogic Server Kubernetes Operator
- Helm
- Verrazzano
- Jenkins
- Compute
- Storage
- Virtual Cloud Network
- Load Balancer
- Database
- Vault
- Identity

Figure 1-1 Components of a typical Oracle WebLogic Server for OKE deployment

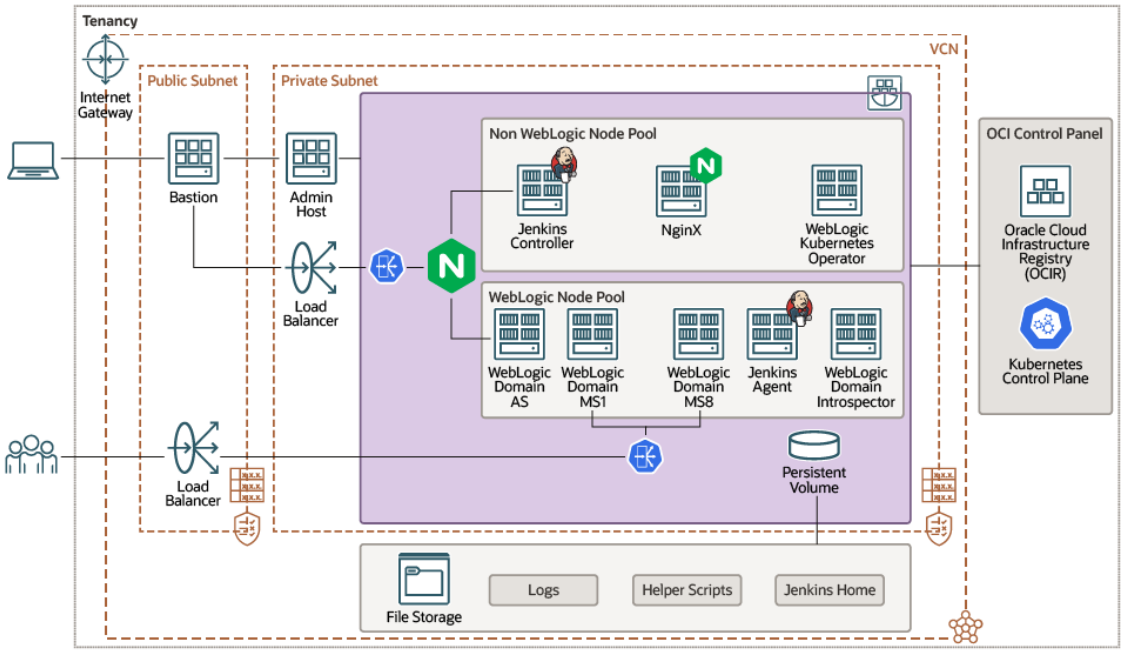
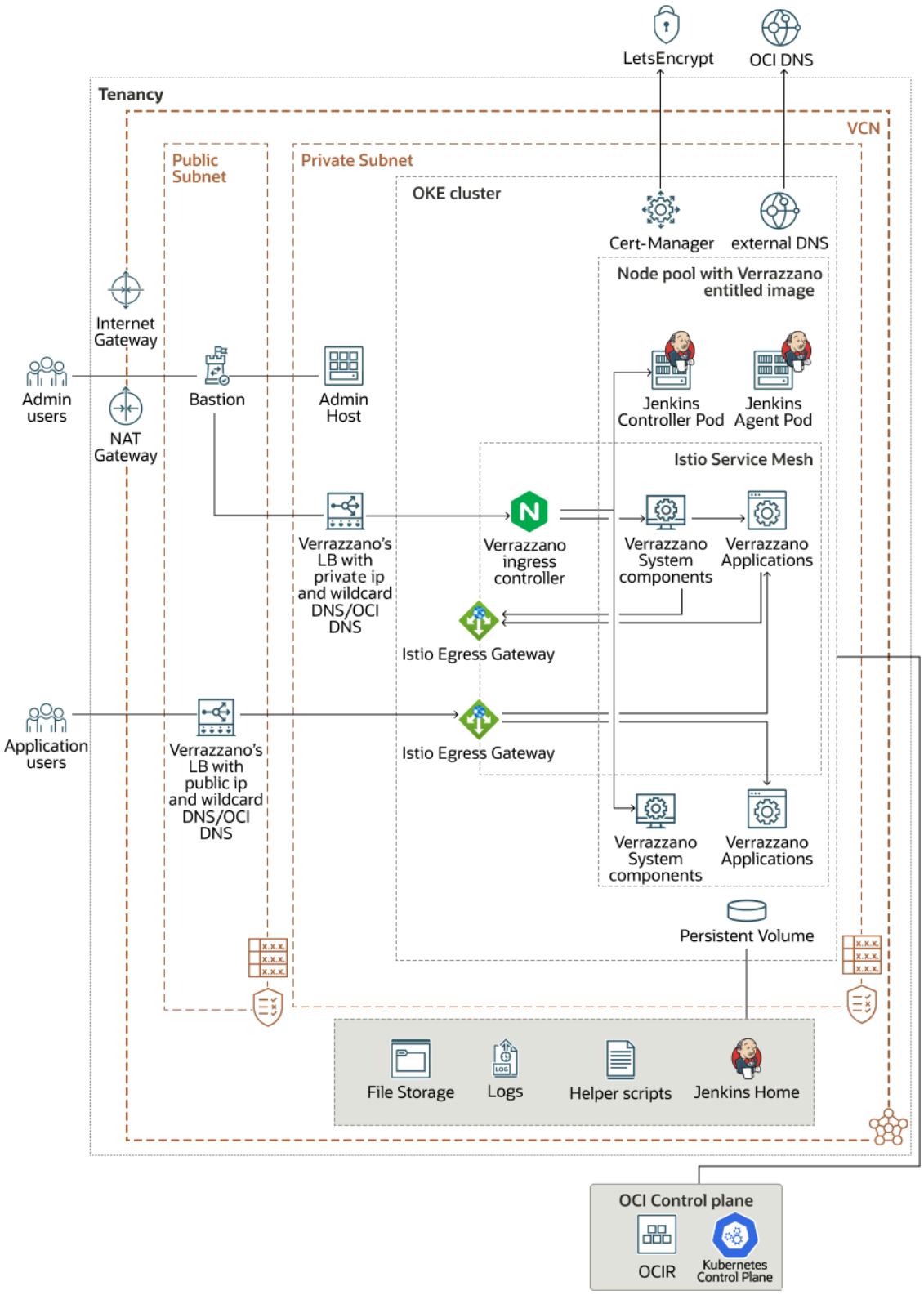


Figure 1-2 Components of a typical Oracle WebLogic Server for OKE deployment in Verrazzano



Oracle WebLogic Server

An Oracle WebLogic Server domain consists of one administration server and one or more managed servers to host your Java application deployments.

Oracle WebLogic Server for OKE supports these Oracle WebLogic Server editions:

- Oracle WebLogic Server Enterprise Edition
  - Includes clustering for high availability and scalability of Java resources and applications
  - Includes Oracle Java SE Advanced (Java Mission Control and Java Flight Recorder) for diagnosing problems in development and production
- Oracle WebLogic Suite
  - Includes all features and benefits of Oracle WebLogic Server Enterprise Edition
  - Includes Verrazzano for managing WebLogic workloads
  - Includes Oracle Coherence for increased performance and scalability
  - Includes Active Gridlink for RAC for advanced database connectivity

Oracle WebLogic Server for OKE supports Oracle WebLogic Server 12.2.1.4.0 and 14.1.1.0.0 releases. See About Oracle WebLogic Server for OKE for specific version information.

Oracle WebLogic Server for OKE can create these domain configurations:

- A basic domain that does not require a database.
- A domain that includes the Java Required Files (JRF) components and also requires a database.

Domains created with Oracle WebLogic Server for OKE do not utilize the Node Manager. Server health monitoring and lifecycle operations are performed by the WebLogic Server Kubernetes Operator.

JRF Domain

The Java Required Files (JRF) option must be selected if your applications were developed with Oracle Application Development Framework (ADF) or use Oracle Web Services Manager (WSM) to access and secure REST and SOAP endpoints.

For more information about ADF, see Faster and Simpler Java-based Application Development. For more information about WSM, see the documentation for Oracle Web Services Manager.

> **Note:**
>
> These Fusion Middleware components are considered part of Oracle JRF: Oracle Application Development Framework, Oracle Fusion Middleware Audit Framework, Fabric Common, Infrastructure Security, Java Object Cache, JMX Framework, JPS, MDS, OJSP.Next, Oracle Web Services, Oracle Web Services Manager, Oracle TopLink, UCP, and XDK.

Marketplace

Oracle WebLogic Server for OKE is accessed as a collection of applications in the Oracle Cloud Infrastructure Marketplace.

Oracle Cloud Infrastructure Marketplace is an online store that's available in the Oracle Cloud Infrastructure console. When you launch an Oracle WebLogic Server for OKE application from Marketplace, it prompts you for some basic information, and then directs you to Resource Manager to complete the configuration of your Oracle WebLogic Server stack and supporting cloud resources.

Choose an Oracle WebLogic Server for OKE application that meets your functional and licensing requirements.

See Overview of Marketplace in the Oracle Cloud Infrastructure documentation.

Resource Manager

Oracle WebLogic Server for OKE uses Resource Manager in Oracle Cloud Infrastructure to provision the Kubernetes cluster, networks and other cloud resources that support your Oracle WebLogic Server stack.

Resource Manager is an Oracle Cloud Infrastructure service that uses Terraform to provision, update, and destroy a collection of related cloud resources as a single unit called a stack. Resource Manager supports most resource types in Oracle Cloud Infrastructure, but a stack in Oracle WebLogic Server for OKE is comprised of these components:

- A Kubernetes cluster running the WebLogic Server stack and Jenkins
- An administration compute instance that includes `kubectl` and other domain management tools
- A bastion compute instance that provides public access to the administration compute instance
- A virtual cloud network (VCN), including subnets, route tables, and security lists (optional)
- Load balancers

See Overview of Resource Manager in the Oracle Cloud Infrastructure documentation.

Container Engine for Kubernetes

Oracle WebLogic Server for OKE uses Oracle Container Engine for Kubernetes for container management and orchestration.

Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications across a cluster of hosts. A Kubernetes cluster is comprised of a controller node and one or more agent nodes. The worker nodes create and manage containers. Kubernetes groups the containers that make up an application into logical units called pods for easy management and discovery.

Oracle Container Engine for Kubernetes is an Oracle Cloud Infrastructure service that allows you to easily create, manage, and deploy applications to Kubernetes clusters. The nodes in a Kubernetes cluster are Oracle Cloud Infrastructure compute instances.

You can access the Kubernetes API on the cluster control plane through a private endpoint hosted in a subnet of an existing VCN. This Kubernetes API endpoint subnet is assigned a private IP address. See Kubernetes Cluster Control Plane and Kubernetes API.

When you create a stack and then a domain with Oracle WebLogic Server for OKE, it provisions two node pools: WebLogic node pool and non-WebLogic node pool. By default, each node pool is created with one worker node. However, during provisioning, you can specify the number of worker nodes you want in each node pool.

It also creates and deploys the following pods to the Kubernetes cluster:

> **Note:**
>
> All other pods can run on either of the two node pools and are not restricted to a node pool. Only the following listed pods are restricted to run on the specified node pool.

- WebLogic node pool:
  - A pod running the domain's administration server
  - A pod running each managed server in the domain (maximum is 9)
  - A pod running the Jenkins agent
- Non-WebLogic node pool:
  - A pod running the Jenkins controller

When you scale a WebLogic cluster:

- You can add a maximum of four managed servers in the node pool for the WebLogic Server node pods that does not contain an administration server. If you want to add another managed server, you must add a node in the node pool for the WebLogic Server node pods.

  > **Note:**
  >
  > If you set the Java Virtual Machine (JVM) heap size in the WebLogic Server pods, you must decide on the number of managed servers to be added in the node pool. See Set the JVM Arguments Definition to set the JVM heap size.

- You cannot add more than three managed servers in the node pool for the WebLogic Server node pods that contains an administration server.

Oracle WebLogic Server for OKE also creates a separate compute instance that includes the `kubectl` command line utility. You can use `kubectl` to manage and monitor the cluster and your pods.

See Overview of Container Engine for Kubernetes in the Oracle Cloud Infrastructure documentation.

Registry

Oracle WebLogic Server for OKE manages the container images for your domain in Oracle Cloud Infrastructure Registry.

Oracle Cloud Infrastructure Registry lets developers store, share, and manage development artifacts like container images. An image is a read-only template with instructions for creating a container.

During the deployment of an application to a Kubernetes cluster, each pod's configuration can specify which images to pull from the registry. You provide the credentials that Kubernetes uses to access the registry.

The images in the registry are organized into named repositories. Repositories can be private or public. Any user with Internet access and knowledge of the appropriate URL can pull images from a public repository. When an image is pushed to the registry, a new private repository is created automatically if it doesn't already exist.

When you create a domain, Oracle WebLogic Server for OKE pushes a default image to the registry, which is used to provision the pods for your domain. From the administration compute instance, you can update this default image and then apply those changes using Kubernetes.

See Overview of Registry in the Oracle Cloud Infrastructure documentation.

WebLogic Server Kubernetes Operator

Your Oracle WebLogic Server for OKE domain includes the open-source WebLogic Server Kubernetes Operator, which has several key features to assist you with managing domains in a Kubernetes environment.

A WebLogic Server domain is modeled as a custom resource in the Kubernetes configuration file. The operator uses this configuration and the Kubernetes API to automate WebLogic Server operations such as provisioning, starting or stopping servers, patching, scaling, and security.

Oracle WebLogic Server for OKE installs and configures the operator in the Kubernetes cluster, and you can use the operator with `kubectl` on the administration compute instance.

The operator supports the use of Kubernetes persistent volumes to store your domain files in an external file system. However, in Oracle WebLogic Server for OKE all of the files that are required to run your domain are stored in the container image for your domain. With this approach, you can easily share the domain with your entire development team, and also ensure that everyone uses a consistent configuration. You also don't need to manually replicate changes in different environments, like testing and production.

See *WebLogic Kubernetes Operator* documentation.

Helm

Helm is a package manager for Kubernetes. Use it to quickly install and manage Kubernetes applications, tools, and services for a Kubernetes cluster.

A chart is a package in Helm. A release is a running instance of a chart in a Kubernetes cluster.

When you create your Oracle WebLogic Server for OKE instance, the Helm client is installed on the administration compute instance, and uses Helm to install the chart for the Oracle WebLogic Server Kubernetes Operator.

See the Helm Documentation.

Verrazzano

Oracle WebLogic Server for OKE allows easy installation of Verrazzano on the Kubernetes cluster for both BYOL and UCM billing modes. The Verrazzano version installed is 1.5.2.

> **Note:**
>
> You can configure Verrazzano installation for WebLogic Suite Edition only.

Oracle Verrazzano is a container deployment and management platform that allows you to deploy and manage container workloads in Kubernetes. It leverages the Open Application Model (OAM) specification for building platform-agnostic container applications. See Verrazzano documentation.

Oracle WebLogic Server for OKE installs Verrazzano and creates a private load balancer to access the Verrazzano consoles. You can also create and manage your WebLogic domains in Verrazzano using the Jenkins Pipeline jobs. Verrazzano is configured to use Jenkins with Kubernetes plugin.

See Configure Verrazzano, Pipeline Jobs With Verrazzano Installation, and Manage WebLogic Domains in Verrazzano.

Jenkins

Oracle WebLogic Server for OKE uses Jenkins to automate the creation of custom images for your WebLogic Server domain, and the deployment of these images to the Kubernetes cluster.

If Verrazzano is installed on the Kubernetes cluster, Oracle WebLogic Server for OKE uses Jenkins to deploy WebLogic Open Application Model (OAM) components on Verrazzano. See Pipeline Jobs in Verrazzano.

Jenkins is an open-source automation engine that facilitates a development workflow based on Continuous Integration and Continuous Delivery (CI/CD). You create projects that perform a series of steps like checking out files from a source control system, compiling code, or running a script. Pipelines are a type of project that organize complex activities into stages, like building, testing, and deploying applications.

Oracle WebLogic Server for OKE provisions the Jenkins primary server on a pod in the Kubernetes cluster. Jenkins is also configured to use the Kubernetes plugin. When you launch or schedule a job, the Jenkins server creates another pod in the Kubernetes cluster, and this *agent* pod is used to run the job.

> **Note:**
>
> The *agent* pod runs in the WebLogic node pool.

See Jenkins User Documentation.

Compute

In addition to the Kubernetes cluster, Oracle WebLogic Server for OKE creates Oracle Cloud Infrastructure Compute instances to provide access to the cluster and for other administration tasks.

A Oracle WebLogic Server for OKE instance is comprised of these compute instances:

- The Kubernetes cluster compute instances host the worker nodes.

- The administration compute instance hosts `kubectl` and other tools to update and manage your domain in Kubernetes.
- The bastion compute instance provides external network access to the Kubernetes cluster and the administration instance, which are provisioned on private subnets.

During domain creation, the administration compute instance is also used to configure the new Kubernetes cluster and to deploy the pods for the domain.

When you create a domain, you assign a shape to each of the compute instances. The shape determines the number of CPUs and the amount of memory allocated to the compute instance. Oracle Cloud Infrastructure offers a variety of bare metal (BM) and virtual machine (VM) shapes. However, Oracle WebLogic Server for OKE only supports the standard shapes, `VM.Standard2.x`, `VM.Standard.E2.x`, `BM.Standard2.x`, `BM.Standard.E2.x`, and `BM.Standard3.64`, and the flexible shapes, `VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`, and `VM.Optimized3.Flex`. Some shapes might not be available in all regions.

For the flexible shapes, the maximum number of OCPUs are:

- `VM.Standard.E3.Flex` and `VM.Standard.E4.Flex` - 64
- `VM.Standard3.Flex` - 32
- `VM.Optimized3.Flex` - 18

The memory, network bandwidth, and number of Virtual Network Interface Cards (VNICs) scale proportionately with the number of OCPUs. See Flexible Shapes.

You also assign a secure shell (SSH) public key to the compute instances for a domain. You can access and administer the operating system on the compute instances by using an SSH client and the matching private key.

An availability domain (AD) represents a data center within an Oracle Cloud Infrastructure region. Each availability domain contains three fault domains. The administration and bastion compute instances are created in a single availability domain. Oracle Container Engine for Kubernetes automatically distributes the worker nodes across all availability domains and fault domains in a region for high availability.

See Overview of the Compute Service and Regions and Availability Domains in the Oracle Cloud Infrastructure documentation.

Storage

Your domain's files are stored locally within each pod in the Kubernetes cluster, but Oracle WebLogic Server for OKE also uses Oracle Cloud Infrastructure File Storage to support certain administration use cases.

When you create a stack, Oracle WebLogic Server for OKE also creates a shared file system and mounts it to the following components:

- The WebLogic Server pods in the Kubernetes cluster use it to store WebLogic Server log files.
- The Jenkins pods in the Kubernetes cluster use it to store pipeline data.
- The administration compute instance uses it to access the Jenkins pipeline data.
- The administration compute instance uses it during the creation of a domain to deploy the WebLogic Server operator to the Kubernetes cluster.

Oracle WebLogic Server for OKE exports the file system to a mount target in a specified availability domain, which can be a different availability domain than the one used for the domain's compute instances. If you don't have a mount target in the selected availability domain, the File Storage service creates one automatically. Also, the mount target and compute instances can be in different compartments or in a different compartment where the stack is available.

Clients access the file system using the Network File System version 3.0 (NFSv3) protocol. The File Storage service uses synchronous replication to provide high availability for all file systems.

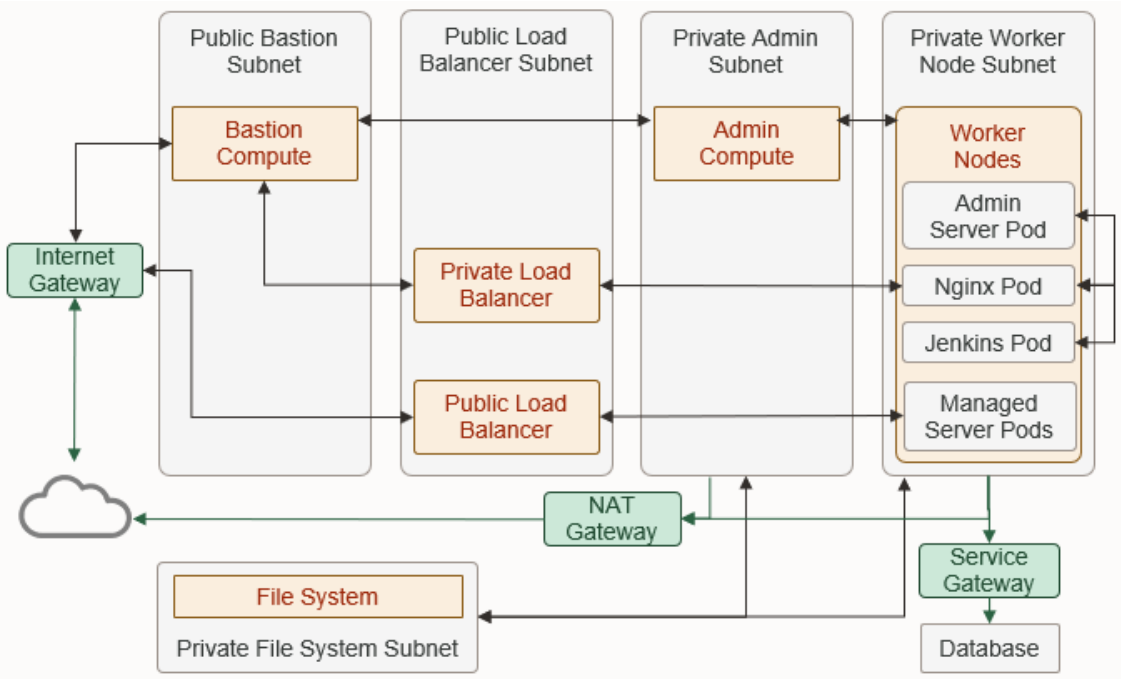
See Overview of File Storage in the Oracle Cloud Infrastructure documentation.

Virtual Cloud Network

Oracle WebLogic Server for OKE assigns a domain's resources to specific subnets in a virtual cloud network (VCN).

A VCN in Oracle Cloud Infrastructure covers a single, contiguous CIDR block of your choice. A VCN includes one or more subnets, route tables, security lists, gateways, and DHCP options. A subnet is a subdivision of a VCN that consists of a contiguous range of IP addresses and does not overlap with other subnets in the VCN.

The following diagram illustrates the VCN for a domain created with Oracle WebLogic Server for OKE.



A subnet can be public or private. Any resources assigned to a private subnet can not be directly accessed from outside of Oracle Cloud. A service gateway allows resources in a private subnet to access other cloud services like Key Management and Autonomous Database, without using the public Internet. A NAT gateway allows outbound access to services that are not in Oracle Cloud.

A domain in Oracle WebLogic Server for OKE consists of the following subnets:

- A private subnet for the worker nodes in the Kubernetes cluster
- A private subnet for the administration compute instance
- A private subnet for the shared file system
- A public subnet for the bastion compute instance
- A public subnet for the load balancers

Oracle WebLogic Server for OKE can automatically create a VCN and subnets for a new domain, or you can create your own VCN and subnets before creating a domain. By default subnets span an entire region in Oracle Cloud Infrastructure. Alternatively, you can create subnets that are specific to one availability domain (AD) in a region.

See Overview of Networking in the Oracle Cloud Infrastructure documentation.

Load Balancer

Oracle WebLogic Server for OKE uses the load balancing capabilities of Oracle Cloud Infrastructure Load Balancing and Oracle Container Engine for Kubernetes.

When you create a domain, Oracle WebLogic Server for OKE creates and configures two load balancers in Oracle Cloud Infrastructure:

- The public load balancer distributes traffic across the managed servers in your domain.
- The private load balancer provides access to the WebLogic Server administration console and the Jenkins console.

A load balancer consists of primary and standby instances but it is accessible from a single IP address. If the primary instance fails, traffic is automatically routed to the standby instance.

A private load balancer is not assigned a public IP address and cannot be accessed from outside of Oracle Cloud. You use the bastion compute instance to get access to the private load balancer for your domain.

> **Note:**
>
> By default, the reserved public IP address that you specify as the `loadBalancerIP` property of the `LoadBalancer` service in the manifest file is expected to be a resource in the same compartment as the cluster. If you want to specify a reserved public IP address in a different compartment, add the following policy to the tenancy:
>
> ```
> Allow any-user to read public-ips in tenancy where
> request.principal.type = 'cluster'
> Allow any-user to manage floating-ips in tenancy where
> request.principal.type = 'cluster'
> ```
>
> See Specifying Load Balancer Reserved Public IP Addresses.

If your region includes multiple availability domains (AD), the load balancer supports two networking options:

- Assign the load balancer to one regional subnet
- Assign the load balancer to two AD-specific subnets

Oracle WebLogic Server for OKE also creates an NGINX ingress controller in the Kubernetes cluster. NGINX is an open-source reverse proxy that controls the flow of traffic to pods within the Kubernetes cluster.

See the following topics in the Oracle Cloud Infrastructure documentation:

- Overview of Load Balancing
- Setting Up an Ingress Controller on a Cluster

Database

To create an Oracle WebLogic Server domain that includes the Java Required Files (JRF) components, you must provide an existing database in Oracle Cloud Infrastructure.

> **Note:**
>
> You cannot create an Oracle WebLogic Server domain that includes the Java Required Files (JRF) components for Oracle WebLogic Server 14.1.1.0 as this version does not support JRF.

When you create a domain and associate it with an existing database, Oracle WebLogic Server for OKE does the following:

- Provisions the schemas to support the JRF components in the selected database
- Provisions data sources in the domain that provide connectivity to the selected database
- Deploys the JRF components and libraries to the domain

Oracle WebLogic Server for OKE supports the following database options for a JRF-enabled domain:

- Oracle Autonomous Database (ATP)
- Oracle Cloud Infrastructure Database (bare metal, virtual machine, and Exadata DB systems)
- Shared Infrastructure (ATP-S), which is accessible from all public IPs or VCNs.
  ATP with VCN support is not supported, where the database is accessible with traffic only from the VCN. See About Network Access Options in *Using Oracle Autonomous Database on Shared Exadata Infrastructure*.

> **Note:**
>
> Free-Tier autonomous database is not recommended for production environments.

For Autonomous Database, Oracle WebLogic Server for OKE supports serverless databases only. Dedicated deployment databases are not supported.

For a 1-node VM DB system, you cannot use the fast provisioning option to create the database.

Oracle WebLogic Server for OKE supports the same database versions and drivers as those for on-premise WebLogic Server installations. Refer to the following documents at Oracle Fusion Middleware Supported System Configurations:

- System Requirements and Supported Platforms for Oracle Fusion Middleware 14c (14.1.1.0.0)
- System Requirements and Supported Platforms for Oracle Fusion Middleware 12c (12.2.1.4.0)

If you use an Oracle Cloud Infrastructure Database, the type of data sources that are created in the domain depend on the WebLogic Server edition and the number of database nodes.

- GridLink data sources for Oracle WebLogic Suite and a 2-node RAC DB system
- Multi data sources for Oracle WebLogic Server Enterprise Edition and a 2-node RAC DB system
- Generic data sources for all other configurations

The service gateway or NAT gateway in your VCN is used by the pods in the Kubernetes cluster to access the database. For an existing VCN, at least a NAT or service gateway is required.

See these topics in the Oracle Cloud Infrastructure documentation:

- Overview of the Autonomous Database
- Overview of the Database Service
- Access to Oracle Services: Service Gateway

See Understanding JDBC Resources in WebLogic Server in *Administering JDBC Data Sources for Oracle WebLogic Server*.

Vault

Oracle Cloud Infrastructure Vault enables you to manage sensitive information when creating an Oracle WebLogic Server domain.

A vault is a container for encryption keys and secrets. You create secrets for a domain's required passwords, and then Oracle WebLogic Server for OKE uses the same vault to decrypt the secrets when creating the domain.

Parameters for a new domain include the password for the default Oracle WebLogic Server administrator.

A standard vault is hosted on a hardware security module (HSM) partition with multiple tenants, and uses a more cost-efficient, key-based metric for billing purposes. A virtual private vault provides greater isolation and performance by allocating a dedicated partition on an HSM.

In order for the domain's Kubernetes cluster, compute instances, and file system to use your secrets, Oracle WebLogic Server for OKE automatically creates a dynamic group and policies in Oracle Cloud Infrastructure.

See these topics in the Oracle Cloud Infrastructure documentation:

- Overview of Vault
- Oracle Cloud Infrastructure Vault FAQ

Identity

Oracle Identity Cloud Service provides Oracle Cloud administrators with a central security platform to manage the relationships that users have with your applications.

By default, the Oracle WebLogic Server domain is configured to use the local WebLogic Server identity store to maintain administrators, application users, groups, and roles. These security elements are used to authenticate users, and to also authorize access to your applications and to tools like the WebLogic Server Administration Console.

Oracle WebLogic Server for OKE can configure a domain running WebLogic Server 12c to use Oracle Identity Cloud Service for authentication.

This configuration is supported only for Oracle Cloud accounts that include Oracle Identity Cloud Service 19.2.1 or later.

Oracle WebLogic Server for OKE configures an App Gateway in Oracle Identity Cloud Service. The App Gateway acts as a reverse proxy, intercepts HTTP requests to the domain, and ensures that the users are authenticated with Oracle Identity Cloud Service.

Oracle WebLogic Server for OKE creates two security applications in Oracle Identity Cloud Service to support the domain. A confidential application allows the domain to securely access the identity provider using the OAuth protocol. An enterprise application defines the URLs that are protected by the App Gateway.

See About Oracle Identity Cloud Service Concepts in *Administering Oracle Identity Cloud Service*.

About the Application Lifecycle with Oracle WebLogic Server for OKE

Learn about deploying and managing applications for a domain that was created with Oracle WebLogic Server for OKE.

A common practice is to create separate Oracle WebLogic Server domains to support development, testing, and production. A traditional development workflow typically includes the following tasks:

1. Update the development domain, including patches, data sources, and applications.
2. Apply the same changes to the test domain. You might use a combination of OPatch, the WebLogic Server administration console, and the WebLogic Scripting Tool (WLST).
3. After testing, apply the same changes to the production domain using the same tools.

Oracle WebLogic Server for OKE promotes a different workflow based on the principles of Continuous Integration and Continuous Delivery (CI/CD).

When you create a Model in Image domain with Oracle WebLogic Server for OKE, all the files required to run the domain in Kubernetes (binaries, patches, configuration, applications, and so on) are stored in the container image for the Model in Image domain. If you want to change the domain, you must update the container image. Any temporary changes you make to the running domain will be lost if you restart the pods in the Kubernetes cluster.

When you create a domain on a Persistent Volume (PV) with Oracle WebLogic Server for OKE, all the binary files required to run the domain in Kubernetes (binaries, patches) are stored in the container image for your PV domain. The domain configuration (applications, and so on) are stored on the persistent volume. If you want to change the binaries, you must update the container image. Any changes you make to the domain configuration using tools such as the WebLogic Server Administration Console, WLST script, and so on, are stored on PV. Therefore, the domain configuration will be persisted if you restart the pods in the Kubernetes cluster.

Oracle WebLogic Server for OKE deploys Jenkins to the Kubernetes cluster along with the domain, and configures a sample project in Jenkins to implement the recommended development workflow. This workflow includes the following tasks:

1. Updating the domain image with applications, resources, libraries, WebLogic patches, and JDK updates.
2. Building a test domain with the updated domain image, and then validating the test domain.
3. Applying the updated domain image to replace the current domain, and then validating the domain.
4. If applying the update domain image to the current running domain fails, then it automatically rolls back to the previous working domain image.

With this approach, you can easily share the latest domain with your entire development team, and also ensure that everyone uses a consistent configuration. You also don't need to manually replicate changes in different environments, such as testing and production.

You can customize the sample Jenkins project to meet your specific CI/CD requirements.

About Oracle WebLogic Server for OKE Versions and Retirement Policy

Oracle WebLogic Server for OKE versions adopt the standard Oracle multiple digits system for version numbering.

A version number for Oracle WebLogic Server for OKE has the following format:

```
YY.Q.M-OKEKubernetesVersion-iteration
```

where, `YY` is the calendar year, `Q` is the quarter, `M` is the month, `OKEKubernetesVersion` is the Kubernetes version supported for OKE.

For example:

```
22.4.3-oke_v1.24.1-1
```

Oracle WebLogic Server for OKE releases do not retire based on any fixed time range. Instead, only the last two product release versions are retained. That is, if *n* is the current version, then *n* and *n-1* versions are retained.

Note the following about all retained Oracle WebLogic Server for OKE versions and images:

- The Oracle WebLogic Server for OKE image of a version is retained as-is, including the WebLogic Server binaries, the VM image contents, and any bugs that were inherent to the Oracle WebLogic Server for OKE scripts on the VM.
- Bugs fixed in a later version are not back ported into an earlier version.
- A version may be pulled without notice if there is a very serious security or functional issue.

About Jenkins Pipeline

Oracle WebLogic Server for OKE deploys Jenkins to the Kubernetes cluster along with your stack, and uses the Jenkins Pipeline to provision a domain, deploy applications, automatic patches, and update the domain images.

Jenkins Pipeline, also referred to as Pipeline, is a suite of plug-ins that supports implementation and integration of continuous delivery (CI/CD) pipelines into Jenkins. The definition of the Pipeline is written in a `Jenkinsfile`, a text file that can be committed to a project's source control repository.

The CI/CD Pipeline uses Oracle WebLogic Deploy Tooling (WDT) and Oracle WebLogic Image Tool (WIT) to update the domain to deploy applications, libraries, and resources; apply JDK and WebLogic Server patches; and update an existing image.

See Oracle WebLogic Server Deploy Tooling and Oracle WebLogic Image Tool.

The images are pushed and pulled within the same regions as deployments using the Oracle Cloud Infrastructure Registry (OCIR) service. See Overview of Registry.

The features of Pipeline are:

- Pipelines can be implemented in code and version controlled.
- Pipelines can be paused at any stage for user inputs or approval.
- Pipelines support complex CD requirements that enables forking or joining, looping and executing stages in parallel.
- Pipelines support domain-specific language (DSL) extension.

Jenkins Pipeline jobs are used to deploy applications and patches. See:

- Pipeline Jobs
- Pipeline Jobs With Verrazzano Installation

Pipeline Jobs

Oracle WebLogic Server for OKE has a set of preconfigured jobs to deploy applications and patches.

Table 1-1 Preconfigured Jobs

Job Type	Job Name	Description
Main	`create domain on pv`	Creates an Oracle WebLogic Server domain home in a Kubernetes Persistent Volume (PV) using the domain creation images. You can use the Domain on PV domain home source type to create a non-Java Required Files (JRF) domain or a Java Required Files domain.
Main	`create mii domain`	Creates an Oracle WebLogic Server domain using the Model in Image (MII) domain home source type with auxillary images. You can use the Model in Image domain home source type to create a non-Java Required Files (JRF) domain.

Table 1-1 (Cont.) Preconfigured Jobs

Job Type	Job Name	Description
Main	`update mii domain`	Updates the domain configuration for the Model in Image domain. Deploys and undeploys applications, shared libraries and resources such as Java Messaging Service (JMS) and datasources to a Model in Image domain. **Note**: If you are using the Domain on PV approach to set up a WebLogic domain, you can update the domain using the WebLogic Server Administration Console or the WebLogic Scripting Tool (WLST).
Main	`automatic patching`	Automatically applies patches on the selected domains, as per the schedule.
Main	`apply patch`	Applies OPatches on the base image of the domain for both Model in Image domain and Domain on PV.
Main	`create base image`	Creates a new base image from Fusion Middleware installer, JDK installer, and JDK patches. Used to create the base image for both Model in Image domain and Domain on PV.
Main	`update base image`	Updates a base image of a domain with a different base image. Used to update the base image for both Model in Image domain and Domain on PV.
Main	`terminate mii domain`	Deletes the IDCS, external load balancer, WebLogic domain, and the Kubernetes namespaces that were created for the Model in Image domain. Additionally, the job also deletes the node pool associated with the domain if you opt to delete the nodepool at the time of terminating the domain. See Terminate a WebLogic Domain.
Main	`terminate pv domain`	Deletes the IDCS, external load balancer, WebLogic domain, and the Kubernetes namespaces that were created for the Domain on PV. Additionally, the job also deletes the node pool associated with the domain if you opt to delete the nodepool at the time of terminating the domain. See Terminate a WebLogic Domain.

Table 1-1 (Cont.) Preconfigured Jobs

Job Type	Job Name	Description
Child	`backup and deploy domain`	Creates a backup of the Yaml file of the running domain and deploys the changes on the running domain. This job is used by other pipeline jobs and cannot be activated manually. This is a child job for the `update mii domain`, `update base image`, and `apply patch` jobs.
Child	`create wls nodepool`	This is a child job for the `create mii domain` and `create pv domain` jobs.

Pipeline Jobs With Verrazzano Installation

Oracle WebLogic Server for OKE has a set of preconfigured jobs to deploy applications.

The following table lists the preconfigured jobs for Oracle WebLogic Server for OKE stack with Verrazzano. These preconfigured jobs are `main` job type.

Table 1-2 Preconfigured Jobs in Verrazzano

Job Name	Description
`create domain as component`	Creates an empty WebLogic domain with a container image and component.yaml file using the selected base image. This image can be used to register the Verrazzano WebLogic component and deploy applications.
`update domain in component`	Deploys and undeploys applications, shared libraries and resources such as Java Messaging Service (JMS) and datasources to a domain.
`create base image`	Creates new image from Fusion Middleware installer, JDK installer, and JDK patches. This new image can be used to create a new domain or update an existing domain..
`rebase component image`	Updates a base image of a Verrazzano WebLogic component with a different base image.
`register component`	Creates a domain definition using component.yaml. This job is used to create new empty WebLogic domain or update an existing WebLogic domain.
`configure application`	Deploys a Verrazzano application with WebLogic workloads.
`terminate component`	Deletes the Verrazzano component, RCU, WebLogic domain, Kubernetes namespaces, persistent volume that we created for the domain.
`unregister component`	Deletes the Verrazzano WebLogic component.
`unregister application`	Deletes the application configuration of the Verrazzano WebLogic component.

2

Create a Stack

Learn how to create a stack with Oracle WebLogic Server for OKE.

Topics:

About Creating a Stack

Learn about the options you have when creating a stack with Oracle WebLogic Server for OKE.

You have several options to choose from when you create a stack:

- Virtual Cloud Network (VCN)

  Oracle WebLogic Server for OKE can create a VCN for you when you create a stack, or you can specify a VCN that you have already created.

  If you let Oracle WebLogic Server for OKE create a new VCN, you must specify a contiguous CIDR block of your choice when you create the stack.

  If you use an existing VCN, you can let Oracle WebLogic Server for OKE create the subnets for you, or you can specify subnets that you have already created.

- Subnets

  Oracle WebLogic Server for OKE can create regional public and private subnets for the domain resources, or you can specify subnets that you have already created.

  If you let Oracle WebLogic Server for OKE create new subnets, you must specify a contiguous CIDR block of your choice for each subnet when you create the stack.

  If using existing subnets, you can specify either regional or availability domain-specific subnets that are scoped to one availability domain in the region.

- Network Access

  Oracle WebLogic Server for OKE creates private subnets for the Kubernetes cluster, administration compute instance, and the file system and mount target, and creates public subnets for the bastion instance and the WebLogic cluster load balancer. If using existing subnets, we recommend that you use the same architecture.

- Load Balancers

  When you create a stack, Oracle WebLogic Server for OKE creates a private load balancer to access administration consoles.

The load balancers are assigned to a public subnet, for which you must specify a CIDR block if you let Oracle WebLogic Server for OKE create new subnets during stack provisioning. You must also specify shapes for the private load balancer.

Prerequisites to Create a Stack

Before you create a stack with Oracle WebLogic Server for OKE, you must complete one or more prerequisite tasks.

Some tasks are required for any type of Oracle WebLogic Server stack that you create with Oracle WebLogic Server for OKE. Other tasks are optional or only applicable for specific domain configurations.

> **Note:**
>
> Before you create a stack, you can estimate the cost of the resources and services to use in your instance. See Oracle Cloud Cost Estimator.

Required Tasks

- Understand Service Requirements
- Create a Compartment
- Create Compartment Policies
- Create Root Policies
- Create an Auth Token
- Create an Encryption Key
- Create an SSH Key

Optional Tasks

- Create a Dynamic Group
- Create Policies for the Dynamic Group
- Create a Virtual Cloud Network
- Create a Subnet for the Kubernetes Cluster
- Create a Subnet for the Administration Host
- Create a Subnet for the Bastion Host
- Create a Subnet for the Load Balancer
- Create a Subnet for the File System
- Validate Existing Network Setup

Understand Service Requirements

You require access to several Oracle Cloud Infrastructure services in order to use Oracle WebLogic Server for OKE.

- Identity and Access Management (dynamic groups and policies)
- Compute

- Network
- Block Storage
- File Storage and Mount targets
- Container Engine
- Registry
- Vault
- Resource Manager
- Load Balancing
- Database (optional)
- Cloud Shell (optional)
- Tagging (optional)

To use Oracle WebLogic Server for OKE, you need at least the following limits available in your tenancy or region or availability domain as applicable:

- 1 OKE Cluster
- 4 Compute instances
- 2 Load Balancers
- 1 File System Service
- 1 Mount Target

Check the service limits for these components in your Oracle Cloud Infrastructure tenancy and, if necessary, request a service limit increase. See Service Limits in the Oracle Cloud Infrastructure documentation.

Create a Compartment

Create compartments in Oracle Cloud Infrastructure for your Oracle WebLogic Server for OKE resources, or use existing compartments.

When you create a stack with Oracle WebLogic Server for OKE, by default the Kubernetes cluster, compute instances, networks, and load balancers are all created within a single compartment. You can, however, choose to use a separate compartment for the network resources that are created for the stack, including load balancers, virtual cloud network, subnets, security lists, route tables and gateways.

See Managing Compartments in the Oracle Cloud Infrastructure documentation.

Create Compartment Policies

If you are not an Oracle Cloud Infrastructure administrator, you must be given management access to resources in the compartment in which you want to create a stack using Oracle WebLogic Server for OKE.

Access to Oracle Cloud Infrastructure resources in a compartment is controlled through policies. Your Oracle Cloud Infrastructure user must have management access for Marketplace applications, Resource Manager stacks and jobs, Kubernetes clusters, compute instances, file systems, block storage volumes, load balancers, Key Management vaults and keys, and IAM policies. If you want Oracle WebLogic Server for OKE to create network resources for a stack, then you must also have management access for these network resources.

A sample policy is shown below:

Where, *MyCompartment* is the compartment in which you created the stack.

```
Allow group MyGroup to manage instance-family in compartment MyCompartment
Allow group MyGroup to manage orm-family in compartment MyCompartment
Allow group MyGroup to manage mount-targets in compartment MyCompartment
Allow group MyGroup to manage file-systems in compartment MyCompartment
Allow group MyGroup to manage export-sets in compartment MyCompartment
Allow group MyGroup to manage cluster-family in compartment MyCompartment
Allow group MyGroup to use subnets in compartment MyCompartment
Allow group MyGroup to use vnics in compartment MyCompartment
Allow group MyGroup to inspect compartments in compartment MyCompartment
Allow group MyGroup to read metrics in compartment MyCompartment
Allow group MyGroup to manage virtual-network-family in compartment
MyCompartment
```

If you need to allow a user who is not an administrator to create secrets with the passwords required during provisioning, ensure that you grant `manage` access to the vaults, keys, and secret-family. For example:

```
Allow group MyGroup to manage vaults in compartment MyCompartment
Allow group MyGroup to manage keys in compartment MyCompartment
Allow group MyGroup to manage secret-family in compartment MyCompartment
```

If you need to allow a user who is not an administrator to use the secrets created by administrator, make sure you grant the following policy. For example:

```
Allow group MyGroup to inspect secrets in compartment id <Compartment OCID>
```

If you use a separate compartment for network resources, ensure that you set up the appropriate policy for the network compartment. For example:

```
Allow group MyGroup to manage virtual-network-family in compartment
MyNetworkCompartment
```

If you use a separate compartment for FSS resources, ensure that you set up the appropriate policy for the FSS compartment. For example:

```
Allow group MyGroup to manage mount-targets in compartment MyFSScompartment
Allow group MyGroup to manage file-systems in compartment MyFSScompartment
Allow group MyGroup to manage export-sets in compartment MyFSScompartment
```

If you intend to create a domain that includes the Java Required Files (JRF) components, then you must set up the policy for the database compartment. For example:

```
Allow group MyGroup to inspect autonomous-transaction-processing-family in
compartment MyDBCompartment
Allow group MyGroup to inspect database-family in compartment MyDBCompartment
```

See Common Policies in the Oracle Cloud Infrastructure documentation.

Create Root Policies

Certain root-level policies must exist in order to use Oracle WebLogic Server for OKE.

Identity and Access Management (IAM) policies let you control what type of access a group of users has and to which specific resources. Your Oracle Cloud Infrastructure administrator sets up the groups, compartments, and policies. Most IAM policies are set at the compartment level, while some are at the tenancy (root) level:

- Delegate IAM tasks, including the creation of dynamic groups
- Use the Cloud Shell to quickly run the Oracle Cloud Infrastructure command line interface (CLI)
- Inspect tag namespaces and apply defined tags from those namespaces to cloud resources

The following sample root policy grants other relevant permissions to a group of users who are not administrators:

```
Allow group MyGroup to inspect tenancies in tenancy
Allow group MyGroup to use tag-namespaces in tenancy
```

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Managing Dynamic Groups
- Cloud Shell
- Managing Tags and Tag Namespaces
- Policy Configuration for Cluster Creation and Deployment

Create an Auth Token

In order for Oracle WebLogic Server for OKE to push and pull container images to and from Oracle Cloud Infrastructure Registry, you must provide an auth token.

Oracle WebLogic Server for OKE can access the registry as the same user that creates a stack, or as a different user.

Every user in Oracle Cloud Infrastructure can be associated with up to two auth tokens. You can create a new auth token for a user with access to Oracle Cloud Infrastructure Registry, or use an existing auth token. When creating an auth token, be sure copy the token string immediately. You can't retrieve it again later using the console.

See Managing User Credentials in the Oracle Cloud Infrastructure documentation.

Create an Encryption Key

Create an encryption key in Oracle Cloud Infrastructure Vault. This will allow you to encrypt the passwords required for Oracle WebLogic Server for OKE.

Oracle WebLogic Server for OKE uses a single key to decrypt all passwords for a single stack.

Create an SSH Key

Create a secure shell (SSH) key pair so that you can access the compute instances in your Oracle WebLogic Server instances.

A key pair consists of a public key and a corresponding private key. When you create a stack using Oracle WebLogic Server for OKE, you specify the public key. You then access the compute instances from an SSH client using the private key.

On a UNIX or UNIX-like platform, use the `ssh-keygen` utility. For example:

```
ssh-keygen -b 2048 -t rsa -f mykey
cat mykey.pub
```

On a Windows platform, you can use the PuTTY Key Generator utility. See Creating a Key Pair in the Oracle Cloud Infrastructure documentation.

Create a Virtual Cloud Network

Oracle WebLogic Server for OKE can create a Virtual Cloud Network (VCN) in Oracle Cloud Infrastructure for a new Oracle WebLogic Server instance, or you can create your own VCN before creating a stack.

A VCN includes one or more subnets, route tables, security lists, gateways, and DHCP options.

By default subnets are public. Any resources assigned to a private subnet cannot be directly accessed from outside of Oracle Cloud. We recommend that you use private subnets for the Kubernetes cluster, administration compute instance, and file system.

If you create a VCN before creating a stack, then the VCN must meet the following requirements:

- The VCN must use DNS for hostnames.
- The VCN must include an Internet gateway.
- If you want to create a public subnet for the stack, then the VCN must include a route table that directs traffic to the Internet gateway.
- The VCN must include a service gateway so that resources in private subnets can access other cloud services like Key Management, Oracle Cloud Infrastructure Registry, and Oracle Autonomous Database.
- If you want to create a private subnet for the stack, then the VCN must include a route table that directs traffic to the service gateway.
- If you want resources in private subnets to access services outside of Oracle Cloud, then the VCN must include a Network Address Translation (NAT) gateway.
- If your VCN includes a NAT gateway and you want to create a private subnet for the stack, then the VCN must include a route table that directs traffic to the NAT gateway.

If you use an existing VCN for a stack, and also choose for Oracle WebLogic Server for OKE to create new subnets for the stack, then Oracle WebLogic Server for OKE will also create the required route tables in the VCN.

If you use an existing VCN and existing subnets in Oracle WebLogic Server for OKE, you can certify the existing network setup using helper scripts. See Validate Existing Network Setup and Script File To Validate Network Setup.

See these topics in the Oracle Cloud Infrastructure documentation:

- VCNs and Subnets
- Access to Oracle Services: Service Gateway

Create a Subnet for the Kubernetes Cluster

Oracle WebLogic Server for OKE can create a subnet for the Kubernetes cluster that hosts your Oracle WebLogic Server instance, or you can create your own subnet before creating a stack.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a stack with Oracle WebLogic Server for OKE, the worker nodes in the Kubernetes cluster are assigned to a subnet. We recommend that you use a private subnet for the Kubernetes cluster.

By default subnets span an entire region in Oracle Cloud Infrastructure. Alternatively, you can create multiple subnets that are each specific to one availability domain (AD) in a region. Oracle WebLogic Server for OKE supports both regional and AD-scoped subnets.

If you want to use an existing subnet for the Kubernetes cluster when creating a stack, the subnet must meet the following requirements:

- The subnet must use DNS for hostnames.
- The subnet must have a security list that enables inbound access to the SSH port (22) from the subnet that you plan to use for the administration compute instance.
- The subnet must have a security list that enables inbound access to all ports from the subnet that you plan to use for the load balancer.
- The subnet must have a security list that enables outbound access to the NFS port (2049) on the file system subnet.

If you use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.

Network security groups are an alternative to security lists. After creating a stack with an existing subnet, you can update the compute instances and assign them to a security group that has the required rules (inbound access to port 22, and so on).

If you use an existing subnet, you can certify the existing network setup using helper scripts. See Validate Existing Network Setup and Script File To Validate Network Setup.

See VCNs and Subnets and Network Resource Configuration for Cluster Creation and Deployment in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the Administration Host

Oracle WebLogic Server for OKE can create a subnet for your stack's administration compute instance, or you can create your own subnet before creating a stack.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a stack with Oracle WebLogic Server for OKE, the administration compute instance is assigned to a subnet. We recommend that you use a private subnet.

By default subnets span an entire region in Oracle Cloud Infrastructure. Alternatively, you can create multiple subnets that are each specific to one availability domain (AD) in a region. Oracle WebLogic Server for OKE supports both regional and AD-scoped subnets.

If you want to use an existing subnet for the administration compute instance when creating a stack, the subnet must meet the following requirements:

- The subnet must use DNS for hostnames.
- The subnet must have a security list that enables inbound access to the SSH port (22) from the subnet that you plan to use for the bastion compute instance.
- The subnet must have a security list that enables outbound access to the SSH port (22) on the subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables outbound access to the WebLogic administration server ports (by default, 7001 and 7002) on the subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables outbound access to the NFS port (2049) on the file system subnet.

If you use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.

Network security groups are an alternative to security lists. After creating a stack with an existing subnet, you can update the compute instances and assign them to a security group that has the required rules (inbound access to port 22, and so on).

If you use an existing subnet, you can certify the existing network setup using helper scripts. See Validate Existing Network Setup and Script File To Validate Network Setup.

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the Bastion Host

Oracle WebLogic Server for OKE can create a public subnet in Oracle Cloud Infrastructure for the bastion compute instance that is used to access your private Oracle WebLogic Server instance, or you can create your own subnet before creating a stack.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a stack in Oracle WebLogic Server for OKE, we recommend that you use a private subnet for the Kubernetes cluster and the administration compute instance. Because these resources can not be directly accessed from outside of Oracle Cloud, Oracle WebLogic Server for OKE creates a bastion compute instance on a public subnet.

By default subnets span an entire region in Oracle Cloud Infrastructure. Alternatively, you can create subnets that are specific to one availability domain (AD) in a region. Oracle WebLogic Server for OKE supports both regional and AD-scoped subnets.

If you want to use an existing subnet for the bastion compute instance when creating a stack, then the subnet must meet the following requirements:

- The subnet must use DNS for hostnames.
- The subnet must be public.
- The subnet must have a security list that enables inbound access to the SSH port (22).
- The subnet must have a security list that enables outbound access to the SSH port (22) on the subnet that you plan to use for the administration compute instance.

If you use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.

Network security groups are an alternative to security lists. After creating a stack with an existing subnet, you can update the bastion compute instance and assign it to a security group that has the required rules (inbound access to port 22, and so on).

If you use an existing subnet, you can certify the existing network setup using helper scripts. See Validate Existing Network Setup and Script File To Validate Network Setup.

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the Load Balancer

Oracle WebLogic Server for OKE can create a subnet for the load balancer that is used to access an Oracle WebLogic Server instance, or you can create your own subnet before creating a stack.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a stack, Oracle WebLogic Server for OKE creates a private load balancer and assigns them to a public subnet. The private load balancer is used to access the WebLogic Server administration console and the Jenkins console. It is not assigned a public IP address from the subnet.

By default subnets span an entire region in Oracle Cloud Infrastructure. Alternatively, you can create subnets that are specific to one availability domain (AD) in a region. Oracle WebLogic Server for OKE supports both regional and AD-scoped subnets.

If you want to use an existing subnet for the load balancer when creating a stack, then the subnet must meet the following requirements:

- The subnet must use DNS for hostnames.
- The subnet can be public or private.
- The subnet must have a security list that enables inbound access to ports 80 and 443.
- The subnet must have a security list that enables outbound access to the WebLogic administration server ports (by default, 7001 and 7002) on the subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables outbound access to the WebLogic managed server ports (by default, 7003 and 7004) on the subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables outbound access to the Jenkins port (80) on the subnet that you plan to use for the Kubernetes cluster.

If you use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.
Network security groups are an alternative to security lists. After creating a stack with an existing subnet, you can update the load balancer and assign it to a security group that has the required rules (inbound access to port 80, and so on).

If you use an existing subnet, you can certify the existing network setup using helper scripts. See Validate Existing Network Setup and Script File To Validate Network Setup.

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the File System

Oracle WebLogic Server for OKE can create a subnet for the shared file system that you use to manage your Oracle WebLogic Server instance, or you can create your own subnet before creating a stack.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a stack with Oracle WebLogic Server for OKE, the file system is assigned to a subnet. We recommend that you use a private subnet for the file system.

By default subnets span an entire region in Oracle Cloud Infrastructure. Alternatively, you can create multiple subnets that are each specific to one availability domain (AD) in a region. Oracle WebLogic Server for OKE supports both regional and AD-scoped subnets.

If you want to use an existing subnet for the file system when creating a stack, the subnet must meet the following requirements:

- The subnet must use DNS for hostnames.
- The subnet must have a security list that enables inbound access to the NFS port (2049) from the private subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables inbound access to the NFS port from the private subnet that you plan to use for the administration compute instance.

If you use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.

Network security groups are an alternative to security lists. After creating a stack with an existing subnet, you can update the file system and assign it to a security group that has the required rules (inbound access to port 2049, and so on).

If you use an existing subnet, you can certify the existing network setup using helper scripts. See Validate Existing Network Setup and Script File To Validate Network Setup.

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Validate Existing Network Setup

You can use helper scripts from the Oracle Cloud Infrastructure Cloud shell to certify the existing network setup (existing VCN and existing WebLogic Server subnet) in Oracle WebLogic Server for OKE. See Using Cloud Shell in Oracle Cloud Infrastructure documentation.

The helper scripts perform the following validations and functions:

- Validates if the service gateway or the NAT gateway is created for the administration instance private subnet and the worker nodes private subnets.
- Validates if internet gateway is created for public bastion, file shared system and load balancer subnets.
- Checks if port 22 in WebLogic Server Subnet is open for access to the CIDR of the bastion instance subnet or bastion host IP.
- Checks if the private subnet for the Oracle WebLogic Server compute instances using the service gateway route rule has **All <*Region*> Services In Oracle Services Network** as the destination.
- Checks if the existing subnet for the load balancer has a security list that enables inbound access to ports 80 and 443.

- Validates if all protocols are open in private subnet for Kubernetes worker node for the Worker CIDR range.
- Validates if all protocols are open in private subnet for Kubernetes worker node for the VCN CIDR range.
- Validates if the file shared system has a security list that enables outbound access to ports 111 and 2048 (both TCP and UDP).
- Validates if the database port is accessible from WebLogic Server subnets.

Using the Validation Script

You can run the helper scripts to perform validations for existing private subnets, existing public subnets, and existing VCN peered subnets.

You must run the commands on the validation script file to check the existing network setup. For example, in this case, let's run the commands on the validation script file named `validateoke.sh`. See Script File To Validate Network Setup to create the `validateoke.sh` file.

1. Set execute permission to the `validateoke.sh` file.

   ```
   chmod +x validateoke.sh
   ```

2. Run the `validateoke.sh` command.

   ```
   ./validateoke.sh [OPTIONS]
   ```

   The following table lists the options that can be used with the `validateoke.sh` comand.

Parameter		Description
Short Form	Long Form	
-b	--bastionsubnet	Bastion Subnet OCID
-a	--adminsubnet	Administration Host Subnet OCID
-w	--workersubnet	Worker Subnet OCID
-f	--fsssubnet	File Shared System Subnet OCID
-l	--lbsubnet	Load Balancer Subnet OCID
-i	--bastionipcidr	Bastion Host IP CIDR The bastion host IP CIDR must have `/32` suffix.
-	--debug	Runs script in BASH debug mode (set -x)
-h	--help	Displays help and exits
-	--version	Displays output version information and exits

3. Run the following command prior to creating a stack:

   ```
   ./validateoke.sh -b <Bastion Subnet OCID> -a <Administration Host Subnet OCID> -w <Worker Subnet OCID>  -f <File Shared System Subnet OCID> -l <Load Balancer Subnet OCID>
   ```

Create Dynamic Groups and Policies

When you create a stack, by default the **OCI Policies** check box is selected and Oracle WebLogic Server for OKE creates the dynamic groups and policies.

The following policies are required when **OCI Policies** check box is selected:

```
Allow group MyGroup to manage dynamic-groups in tenancy
Allow group MyGroup to manage policies in tenancy
```

If you do not belong to a group that has the policies listed above, then you need to clear the **OCI Policies** check box and create a dynamic group and the required polices.

These tasks are typically performed by any user that belongs to a group that has the policies listed above or a tenancy administrator:

- Create a Dynamic Group
- Create Policies for the Dynamic Group

Create a Dynamic Group

Create a dynamic group in Oracle Cloud Infrastructure whose members are the compute instances that Oracle WebLogic Server for OKE will create for a stack.

The dynamic group is necessary for the compute instances to access encryption keys in Key Management, and also to access the database wallet if you're using Oracle Autonomous Database.

During stack creation for a domain, Oracle WebLogic Server for OKE creates compute instances in a compartment you select. This compartment's OCID must be listed in a dynamic group before users who are not administrators can create resources for the stack in the specified compartment.

One or more compartments can be listed in a dynamic group.

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Identity & Security**. Under the **Identity** group, click **Compartments**.
3. Copy the OCID for the compartment that you plan to use for the Oracle WebLogic Server compute instances.

   If you use another compartment just for network resources, copy also the OCID of the network compartment.
4. Click **Dynamic Groups**.
5. Click **Create Dynamic Group**.
6. Enter a **Name** and **Description**.
7. For **Rule 1**, create a rule that includes all instances in the selected compartment in this group.

   ```
   ALL {instance.compartment.id = 'WLS_Compartment_OCID'}
   ```

   Provide the OCID for the compartment you copied in step 3.
8. Click **Create Dynamic Group**.

See Managing Dynamic Groups in the Oracle Cloud Infrastructure documentation.

Create Policies for the Dynamic Group

Create policies in Oracle Cloud Infrastructure so that the compute instances in Oracle WebLogic Server for OKE can access your encryption key.

When you create a stack, compute instances in Oracle WebLogic Server for OKE need to access Oracle Cloud Infrastructure Vault secrets. If a load balancer is enabled, access to network resources is required.

The following sample policy grants the relevant permissions to a dynamic group:

```
Allow dynamic-group MyInstancesPrincipalGroup to manage all-resources in
compartment MyCompartment
Allow service oke to read app-catalog-listing in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
compartment VaultCompartment where target.secret.id = '<OCID for OCIR token
secret>'
Allow dynamic-group MyInstancesPrincipalGroup to use dynamic-groups in tenancy
```

The following sample policy grants the relevant permissions to a dynamic group, and is required if your network compartment is different than the stack compartment:

```
Allow dynamic-group MyInstancesPrincipalGroup to use subnets in compartment
MyNetworkCompartment
Allow dynamic-group MyInstancesPrincipalGroup to use vnics in compartment
MyNetworkCompartment
Allow dynamic-group MyInstancesPrincipalGroup to inspect instance-family in
compartment MyNetworkCompartment
```

The following sample policy grants access to the OS Management service:

```
Allow dynamic-group MyInstancesPrincipalGroup to use osms-managed-instances
in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to read instance-family in
compartment MyCompartment
```

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Writing Policies for Dynamic Groups

Create a Stack

Use Oracle WebLogic Server for OKE to create a stack that includes a basic Oracle WebLogic Server instance with network resources, Kubernetes cluster, compute instances, and private load balancers.

Launch a new stack from Marketplace.

Before you create a stack, ensure that all the prerequisites are completed. See Prerequisites to Create a Stack.

Tutorial

Topics:

Launch a Stack

Sign in to Marketplace and specify initial stack information.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  and select **Marketplace**.
3. Select an application that matches the edition of Oracle WebLogic Server that you want to provision.
   - **Oracle WebLogic Server Enterprise Edition for OKE BYOL**
   - **Oracle WebLogic Server Enterprise Edition for OKE UCM**
   - **Oracle WebLogic Suite for OKE BYOL**
   - **Oracle WebLogic Suite for OKE UCM**
4. Select a required Oracle WebLogic Server for OKE release version to use from the list.
5. Select the compartment in which to create the stack.

   By default the stack compartment is used to contain the compute instances and network resources. If later on you specify a network compartment on the Configure Variables page of the Create Stack wizard, then the compute instances and load balancers are created in the stack compartment that you select here.
6. Select the **Oracle Standard Terms and Restrictions** check box, and then click **Launch Stack**.

   The Create Stack wizard is displayed.

Configure Stack Information

Specify the name, description, and tags for the stack.

1. On the Stack Information page of the Create Stack wizard, enter a name for your stack.
2. Enter a description for the stack (optional).

3. Specify one or more tags for your stack (optional).
4. Click **Next**.

   The Configure Variables page opens.

Configure WebLogic Server on Container Cluster

Specify the parameters needed to configure the WebLogic Server in a Kubernetes cluster.

1. In the WebLogic Server on Container Cluster section of the Configure Variables page, enter the resource name prefix.

   The maximum character length is 16.

   This prefix is used by all the created resources, except load balancers.
2. Enter the SSH public key, by either uploading the SSH key file or copy-pasting the SSH key information.

Configure Verrazzano

Specify the parameters to configure Verrazzano installation based on the Verrazzano profile.

For Verrazzano, when you configure the Container Cluster, Oracle recommends you select `VM.Standard.E4.Flex` shape with four OCPU count and three nodes for the WebLogic Server node pool. See Configure the Container Cluster for Verrazzano Integration.

> **Note:**
>
> You can configure Verrazzano installation for WebLogic Suite Edition only.

When you enable Verrazzano, the default settings are:

- Free wildcard DNS service *nip.io*
- Self-signed CA certificate
- Flexible shape with minimum bandwidth size of 10 Mbps and maximum bandwidth of 100 Mbps for Administrative load balancer and Application Load Balancer
- Three master replicas (with 50Gi storage) and two data replicas (with 50Gi storage) for OpenSearch for *prod* profile, and one master replica (with 50Gi storage) and one data replica (with 50Gi storage) for OpenSearch for *dev* profile
- 50 Gi persistent volume storage for the components, OpenSearch, Prometheus, Grafana, and Keycloak

To configure Verrazzano:

1. Click **Enable Verrazzano**.
2. *Optional*: Enter the **Environment Name** for the Verrazzano installation.

   If specified, the environment name is displayed in the endpoint access URLs of the installation.

Examples of endpoint access URL formats:

```
http://jenkins.<environment_name>.<internal_lb_ip>.<dns_wildcard_service>/
jenkins
```

```
http://
verrazzano.<environment_name>.<internal_lb_ip>.<dns_wildcard_service>
```

See Access the Jenkins Console and Access the Verrazzano Consoles.

3. Select a profile for the Verrazzano installation. The default value is *prod*.

   For Verrazzano profiles, see Installation Profiles.

4. To customize the Verrazzano installation, select one or multiple options as required:

   - Click **Customize DNS**, and under DNS (Domain Name System) Configuration, select one of the **DNS type**:
     - Select *Wildcard*, and then select the wildcard DNS service in the **Wildcard DNS Type** field.
     - Select *OCI*, and then select the compartment in which you have defined the OCI DNS Service Zone and specify the OCID of the OCI DNS Service Zone.

     If you already provisioned an Oracle WebLogic Server for OKE stack with Verrazzano using the OCI DNS type, and if you create a new Oracle WebLogic Server for OKE stack with Verrazzano using the same environment name as the already provisioned stack, to avoid conflicts with the new records that are added for the OCI DNS instance during provisioning of the new stack, you must either remove the DNS instance records from the DNS zone of the already provisioned stack or use a different environment name in the new stack.

     If you use a private DNS zone, see Add a DNS view to the DNS Resolver.
   - Click **Customize Certificates**, and under Certificates Configuration, select one of the **Certificate Type**:
     - Select *Verrazzano self-signed CA*.
     - Select *Custom CA* (Certificate Authority) and then specify the OCID of the secret that contains the signing key and OCID of the secret that contains the custom CA certificate.
     - Select *LetsEncrypt* and specify a valid email address for the LetsEncrypt account and select the LetsEncrypt environment type.

     > **Note:**
     >
     > To use LetsEncrypt, you must configure OCI DNS type.

   - Click **Customize Administrative Load Balancer**, and under Administrative Load Balancer Configuration, do one of the following:
     - Select *flexible* load balancer shape, and then select a minimum and maximum flexible shape for the private load balancer.
       By default, the minimum bandwidth size is set to 10 Mbps and maximum to 100 Mbps.
     - Select the bandwidth size for the private load balancer shape.

- Click **Customize Application Load Balancer**, and under Application Load Balancer Configuration, do one of the following:
  - Select *flexible* load balancer shape, and then select a minimum and maximum flexible shape for the load balancer.
    By default, the minimum bandwidth size is set to 10 Mbps and maximum to 100 Mbps.
  - Select the bandwidth size for the load balancer shape.
    By default, the Application Load balancer is public. If you want to use a private load balancer to access applications, select **Private Application Load Balancer**.
- Click **Customize OpenSearch**, and under OpenSearch Configuration, select the following based on your configuration:
  - Select the number of master node replicas, ingest node replicas, and data replicas.
  - Select **Advanced OpenSearch Configuration** and specify the values for the following as a Quantity:
    * Select the memory request amount for master node, ingest node, and data replica.
    * Select the storage request amount.
- Click **Customize Persistent Storage**, and under Persistent Volume Configuration, specify the values for the following as a Quantity:
  - Specify the global persistent volume storage.
  - Specify the persistent volume storage for Keycloak.

Add a DNS view to the DNS Resolver

When you configure Verrazzano installation for your Oracle WebLogic Server for OKE stack using a private DNS zone and an existing VCN, you need to configure the DNS resolver of the VCN to use the DNS view of the DNS zone.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Networking**, and then select **Virtual Cloud Networks**.
3. From the list of Virtual Cloud Networks, click the name of the VCN.
4. On the **VCN Information** tab, click the name of the **DNS Resolver** for the VCN.
5. From the **Associated Private Views** section, click **Manage Private Views**.
6. In the **Private Manage Views** window, select the **Private View** from the compartment where the private view of the private DNS zone is located.
   If a **Private View** is already associated with the resolver, click **Additional Private View** to select the private view of the private DNS zone.
7. Click **Save Changes**.

Configure the Network

Define the Virtual Cloud Network (VCN) and the subnets configuration for the stack.

1. In the Network section of the Configure Variables page, select the **Network Compartment** in which to create the network resources for this stack.

   If you don't specify a network compartment, then all the network resources and compute instances are created in the stack compartment that you selected earlier upon launching

the stack. Select a network compartment if you want the network resources to be in a different compartment than the compute instances.

2. You can either create a new VCN, use an existing VCN but create *new* subnet resources, or an existing VCN and *existing* subnets.

   For an existing VCN and existing subnet, you can configure a bastion compute instance to provide access to the WebLogic Server compute instances on a private subnet. However, creating the bastion node on public subnet is optional.

   - To create resources in a new VCN, select **Create New VCN** from the **Virtual Cloud Network Strategy** dropdown, and then specify the following:
     - A CIDR for the new VCN
     - A shape for the private load balancer
   - To use an existing VCN but create *new* subnet resources, select **Use Existing VCN** from the **Virtual Cloud Network Strategy** dropdown, then do the following:
     a. From the **Existing Network** dropdown, select the name of an existing VCN.
     b. *Do not* select the **Use Existing Subnet** check box.
     c. Specify public subnet CIDRs for the bastion host and load balancers.
     d. Specify private subnet CIDRs for administration host, file system and mount target (storage) host, Kubernetes cluster and node pool, and Kubernetes cluster and API endpoint.
     e. Enter the Oracle Cloud Identifier (OCID) for an existing NAT gateway or service gateway.
     f. Select a minimum and maximum flexible shape for a private load balancer.
     By default, the minimum bandwidth size is set to 10 Mbps and maximum to 100 Mbps.

     > **Note:**
     >
     > You can update the shape to a maximum of 8000 Mbps. Before you select the maximum bandwidth, ensure to check the available service limit for the flexible private load balancer bandwidth.

   - To use an existing VCN and *existing* subnets with bastion configuration, select **Use Existing VCN** from the **Virtual Cloud Network Strategy** dropdown, then do the following:
     a. From the **Existing Network** dropdown, select the name of an existing VCN.
     b. Select the **Use Existing Subnet** check box.
     c. Select the **Subnet Compartment** to use for the existing subnet.
     The subnet compartment is different than the VCN compartment. The subnets for the bastion host, load balancers, Kubernetes cluster and node pool, Kubernetes cluster and API endpoint, administration host, and the file system and mount target host, use this same subnet compartment.

> **Note:**
>
> You can specify the subnet compartment only if you're using an existing subnet.

d. Keep the default selection for **Provision Bastion node on Public Subnet** check box.

e. Select the name of an existing public subnet for the bastion host.

f. Select the names of existing private subnets for the Kubernetes cluster and node pool, Kubernetes cluster and API endpoint, administration host, and the file system and mount target (storage) host.

g. Select the name of an existing subnet for the load balancer.

h. Enter the Oracle Cloud Identifier (OCID) for an existing NAT gateway or service gateway.

i. Select a minimum and maximum flexible shape for a private load balancer.
By default, the minimum bandwidth size is set to 10 Mbps and maximum to 100 Mbps.

> **Note:**
>
> You can update the shape to a maximum of 8000 Mbps. Before you select the maximum bandwidth, ensure to check the available service limit for the flexible private load balancer bandwidth.

- To use an existing VCN and *existing* subnets without bastion configuration, select **Use Existing VCN** from the **Virtual Cloud Network Strategy** dropdown, then do the following:

  a. From the **Existing Network** dropdown, select the name of an existing VCN.

  b. Select the **Use Existing Subnet** check box.

  c. Select the **Subnet Compartment** to use for the existing subnet.
  The subnet compartment is different than the VCN compartment. The subnets for the bastion host, load balancers, Kubernetes cluster and node pool, Kubernetes cluster and API endpoint, administration host, and the file system and mount target host, use this same subnet compartment.

  > **Note:**
  >
  > You can specify the subnet compartment only if you're using an existing subnet.

  d. Deselect the **Provision Bastion node on Public Subnet** check box.

> **Note:**
>
> - It is recommended to deselect the **Provision Bastion Node on Public Subnet** check box only in network with fast connect setup.
> - In this case, no status is returned for provisioning, then you must check the status of provisioning in the **Logs** under **Application Information** of the stack, and view the error or success messages in the `/u01/logs/provisioning.log` file on the administration instance.
> - To get the internal and external load balancer IP addressesfor accessing the Jenkins Console, WebLogic Console, and the WebLogic Cluster Load Balancer, run the following command:
>
>   ```
>   kubectl get svc -A
>   ```
>
>   The private load balancer is listed with the namespace `wlsoke-ingress-nginx` and name *okename*`-internal`.
>
>   The public load balancers are listed with the namespace `wlsoke-ingress-nginx` and name *<domain-name>*`-lb-external`.

e. Select the names of existing private subnets for the Kubernetes cluster and node pool, Kubernetes cluster and API endpoint, administration host, and the file system and mount target (storage) host.

f. Select the name of an existing subnet for the load balancer.

g. Enter the Oracle Cloud Identifier (OCID) for an existing NAT gateway or service gateway.

h. Select a minimum and maximum flexible shape for a private load balancer. By default, the minimum bandwidth size is set to 10 Mbps and maximum to 100 Mbps.

> **Note:**
>
> You can update the shape to a maximum of 8000 Mbps. Before you select the maximum bandwidth, ensure to check the available service limit for the flexible private load balancer bandwidth.

Configure the Container Cluster

You can specify the parameters needed to create a container cluster or configure the WebLogic Server domain to use an existing container cluster for an existing VCN and an existing subnet only.

> **Note:**
>
> If you configure Verrazzano installation, then to configure the container cluster, see Configure the Container Cluster with Verrazzano.

- Create a Container Cluster
- Use an Existing Cluster

Create a Container Cluster

1. In the Container Cluster Configuration section of the Configure Variables page, enter a **Kubernetes Version** to run on the cluster nodes.

   > **Note:**
   >
   > The latest Kubernetes version is displayed by default. Check the Kubernetes version that is certified and compatible with WebLogic Server Kubernetes Operator. See Oracle WebLogic Server Kubernetes Operator.
   >
   > If you enter a Kubernetes version that is not available, the stack provisioning fails.

2. Select a shape for each node in the Kubernetes cluster node pool, for non-WebLogic node pools.

   If you select a flexible shape, move the slider to specify the OCPU count and amount of memory for the non-WebLogic node pool shape.

   > **Note:**
   >
   > By default, for the flexible shape, the OCPU count is *1* and the amount of memory is *1* GB.

   > **WARNING:**
   >
   > If you do not select a shape, the stack creation is stuck on the Review page of the Create Stack wizard, or the stack creation fails.

3. Select the number of nodes in the node pool, for non-WebLogic node pools.
4. Specify a CIDR for the pods in the Kubernetes cluster.
5. Select the preferred WebLogic Server node pool shape.

   If you select a flexible shape, move the slider to specify the OCPU count and amount of memory for the WebLogic node pool shape.

   > **Note:**
   >
   > By default, for the flexible shape, the OCPU count is *1* and the amount of memory is *1* GB.

> **WARNING:**
>
> If you do not select a shape, the stack creation is stuck on the Review page of the Create Stack wizard, or the stack creation fails.

6. Specify the number of nodes required in the WebLogic Server node pool.
7. Specify a CIDR for the Kubernetes services that are exposed.
8. *Optional:* To encrypt the Kubernetes secrets at rest in `etcd` by using the master encryption key in the OCI vault service, select **Kubernetes Secret Encryption**. Then, select the compartment where you have the vault, the vault where you have the key, and the key.

   If you do not select this option, then the standard block storage encryption is used for `etcd`.

> **Caution:**
>
> - If you use **Kubernetes Secret Encryption**, then ensure that you do not disable or delete the vault key, which you used to encrypt the Kubernetes secrets.
> - If you disable or delete the vault key, you cannot perform any administrative commands on the administration server. like, `kubectl get pods -A`. The only option is to destroy and recreate the domain.
> - If you disable the vault key, the changes are immediate and you would not be able to access the stack.
> - If you have scheduled the key for deletion, it is in the `Pending Deletion` state until it is deleted permanently on the scheduled deletion date. You can cancel the key deletion schedule to restore access to the Kubernetes secrets. See Managing Secrets.

Use an Existing Cluster

1. In the Container Cluster Configuration section of the Configure Variables page, select the **Use existing cluster** check box.
2. Enter the OCID of the existing Kubernetes cluster.

   Ensure that this Kubernetes cluster exists in the compartment that you selected upon launching the stack, and in the specified existing VCN.

   You must not use the same Kubernetes cluster to create multiple Oracle WebLogic Server for OKE instances. If you want to use the cluster for multiple instances, you must delete the resources and the stack. See Delete a Stack .

Configure the Container Cluster with Verrazzano

Specify the parameters needed to create a container cluster if you configured Verrazzano installation.

1. In the Container Cluster Configuration section of the Configure Variables page, enter a **Kubernetes Version** to run on the cluster nodes.

> **Note:**
>
> The latest Kubernetes version is displayed by default. Check the Kubernetes version that is certified and compatible with WebLogic Server Kubernetes Operator. See Oracle WebLogic Server Kubernetes Operator.
>
> If you enter a Kubernetes version that is not available, the stack provisioning fails.

2. Select the WebLogic Server node pool shape.

   If you select a flexible shape, move the slider to specify the OCPU count and amount of memory for the WebLogic node pool shape.

   > **Note:**
   >
   > By default, for the flexible shape, the OCPU count is *1* and the amount of memory is *1* GB.
   >
   > Oracle recommends you select `VM.Standard.E4.Flex` shape with four OCPU counts.
   >
   > > **WARNING:**
   > >
   > > If you do not select a shape, the stack creation is stuck on the Review page of the Create Stack wizard, or the stack creation fails.

3. Specify the number of nodes required in the WebLogic Server node pool.

   > **Note:**
   >
   > You must select a minimum of three nodes for the selected shape.

4. Specify a CIDR for the pods in the Kubernetes cluster.
5. Specify a CIDR for the Kubernetes services that are exposed.
6. *Optional:* To encrypt the Kubernetes secrets at rest in `etcd` by using the master encryption key in the OCI vault service, select **Kubernetes Secret Encryption**. Then, select the compartment where you have the vault, the vault where you have the key, and the key.

   If you do not select this option, then the standard block storage encryption is used for `etcd`.

> **Caution:**
>
> - If you use **Kubernetes Secret Encryption**, then ensure that you do not disable or delete the vault key, which you used to encrypt the Kubernetes secrets.
> - If you disable or delete the vault key, you cannot perform any administrative commands on the administration server. like, `kubectl get pods -A`. The only option is to destroy and recreate the domain.
> - If you disable the vault key, the changes are immediate and you would not be able to access the stack.
> - If you have scheduled the key for deletion, it is in the `Pending Deletion` state until it is deleted permanently on the scheduled deletion date. You can cancel the key deletion schedule to restore access to the Kubernetes secrets. See Managing Secrets.

Configure the Administration Instances

Specify where you want to create the administration instances and select the shapes to use.

1. In the Administration Instances section of the Configure Variables page, select the availability domain in which to create the bastion and Kubernetes administration compute instances.
2. Select a shape for the Kubernetes administration compute instance.

   If you select a flexible shape, move the slider to specify the OCPU count and amount of memory for the administration compute instance.

   > **Note:**
   >
   > By default, for the flexible shape, the OCPU count is *1* and the amount of memory is *1* GB.

   > **WARNING:**
   >
   > If you do not select a shape, the stack creation is stuck on the Review page of the Create Stack wizard, or the stack creation fails.

3. Select a shape for the bastion compute instance.

   If you select a flexible shape, move the slider to specify the OCPU count and amount of memory for the bastion compute instance.

> **Note:**
>
> By default, for the flexible shape, the OCPU count is *1* and the amount of memory is *1* GB.
>
> You cannot select a shape for the bastion compute instance if you deselect the **Provision Bastion Node** check box.

> **WARNING:**
>
> If you do not select a shape, the stack creation is stuck on the Review page of the Create Stack wizard, or the stack creation fails.

Configure the File System

Specify where you want to create the shared file system.

1. Select the availability domain where you want to create the shared file system and the mount target.

   > **Note:**
   >
   > Shared file system and mount target can be in a different availability domain than the WebLogic instances.

2. Select the compartment for the mount target.

   If you want to use an existing subnet to set up an Oracle WebLogic Server for OKE cluster, you have the option to create the mount target in a compartment different from that of the stack compartment. Similarly, you can also use an existing mount target from a different compartment than that of the stack compartment. If you are provisioning a new subnet, the mount target is created in the stack compartment by default.

3. *Optional*: If you want to use an existing subnet to provision an Oracle WebLogic Server for OKE cluster, you have the option to select **Add Existing Mount Target**, and then select an existing mount target from the list of mount targets available for the selected availability domain and compartment. This mount target should be in the same subnet where the new file system is created.

   If you do not select an existing mount target, a new mount target is automatically created for the file system in the chosen compartment for the mount target.

Configure the Registry

Specify the credentials that Oracle WebLogic Server for OKE uses to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

1. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.

   The registry user name format is `tenancy_namespace/<username>`. If your tenancy is federated with Oracle Identity Cloud Service, then the registry user name format is `tenancy_namespace/oracleidentitycloudservice/<username>`.

You can choose either to include the `tenancy_namespace` or remove the `tenancy_namespace` in the user name format. For example, you can either use `tenancy_namespace/<username>` or `<username>`.

> **Note:**
>
> If you choose to include `tenancy_namespace` in the user format, ensure that you use the correct namespace for your tenancy.

2. In the **Registry_Authentication_Token** field, select the compartment where you have the OCI Secret that contains the auth token.
3. In the **Validated Secret for OCIR Auth Token** field, select the secret that contains the OCIR auth token. To generate an auth token, see Getting an Auth Token in the Oracle Cloud Infrastructure documentation.

For information about how to create a container registry, see Overview of Registry in the Oracle Cloud Infrastructure documentation.

Create OCI Policies

When you create stack, by default the **OCI Policies** check box is selected and Oracle WebLogic Server for OKE creates a dynamic group and relevant root-level (tenancy) policies for you.

If you are not an administrator, the necessary groups and policies must be in place before you can create a stack.

Before you deselect the check box, ask your administrator to create the required dynamic group and relevant policies, as described in Create a Dynamic Group and Create Policies for the Dynamic Group.

Create the Stack

After you have specified the parameters for your stack, finish creating the stack.

On the Review page of the Create Stack wizard, review the information you have provided, and then click **Create**. This runs the stack creation job.

The Job Details page of the stack in Resource Manager is displayed. A stack creation job name has the format `ormjobyyyymmddnnnnnn`. For example, `ormjob20200922125850`. Periodically monitor the progress of the job until it is finished. If an email address is associated with your user profile, you will receive an email notification.

Troubleshoot a Stack

Identify common problems in a Oracle WebLogic Server for OKE stack and learn how to diagnose to solve them.

Topics

- Stack Creation Failed
- Load Balancer Creation Failed
- Get Additional Help and Contact Support

View the Cloud Resources for a Stack

Use Resource Manager to view the Oracle Cloud Infrastructure compute instances, networks, and other resources that are provisioned by Oracle WebLogic Server for OKE for your stack.

> **Note:**
>
> Load balancers created for your stack are not listed in the Resource Manager. See Load Balancers.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** that contains your stack.
4. Click the name of your stack.
5. From the Jobs section, click the latest successful job that has the Apply type.
6. Click **Application Information**.

   A list of details displays, including Kubernetes cluster, bastion and administration instance OCIDs, bastion and administration instance IP addresses, and administration console URLs.

   You can click **Logs** to find the private load balancer IP addresses.
7. Click a resource OCID to view the compute instance page for the resource. You can manage the resource on the Instance Details page (for example, reboot the instance).
8. Under Resources on the left, click **Associated Resources**.

   A list of resources in this stack displays. The list includes compute instances, virtual cloud networks (VCN), subnets, security lists, gateways, dynamic groups, and policies.
9. Click **Show** next to a resource name to show the resource attributes.

You can also find your stack's resources by using the search field at the top of the console. For example, if you assigned tags to the resources in the stack, you can enter these tags in the search field.

About the Resources in a Stack

Learn about the compute instances, load balancers, network, and other resources in a stack created by Oracle WebLogic Server for OKE for an Oracle WebLogic Server domain.

To obtain a list of associated resources created for a specific stack, see View the Cloud Resources for a Stack.

Topics:

- Compute Instances
- Network Resources
- Load Balancers

- Kubernetes Resources
- File System Resources
- Registry Resources
- Identity Resources for Dynamic Group and Root Policies

Compute Instances

Oracle WebLogic Server for OKE creates Oracle Cloud Infrastructure compute instances for your Oracle WebLogic Server domain and Kubernetes cluster.

In the Oracle Cloud Infrastructure Console, use the navigation menu and select **Compute**. Under the **Compute** group, click **Instances**. When you select the compartment you specified to use for Oracle WebLogic Server when you created the stack, you'll see the following compute instances provisioned for your stack and Kubernetes cluster:

- Bastion instance - Has the name *resourceprefix*-bastion
- Administration instance - Has the name *resourceprefix*-admin
- A Kubernetes worker node - Has the name oke-*generated-alphanumeric-string-n*

Note: *resourceprefix* is the resource name prefix you provided during stack creation. *n* is the number 0 or 1.

Network Resources

Oracle WebLogic Server for OKE creates several network resources such as route tables, security lists, and gateways for your Oracle WebLogic Server stack and Kubernetes cluster in Oracle Cloud Infrastructure.

Additional network resources are created if you specify a new virtual cloud network (VCN) or new subnets for an existing VCN during stack creation.

In the Oracle Cloud Infrastructure Console, click **Networking** and select a compartment to view network resources. For example, click **Virtual Cloud Networks** to view all the virtual cloud networks (VCN) created in a compartment. If you created a new VCN for your stack during stack creation you'll find the VCN and its related resources in the compartment you specified to use for network resources.

Your stack configuration determines the type and number of network resources created. With the exception of load balancers, the names of those network resources begin with the resource name prefix you provided during stack creation. For example, *resourceprefix*-admin and *resourceprefix*-bastion.

The following table provides a summary of the resources that can be created for your domain.

Resource Name	Type
resourceprefix-vcn	WebLogic VCN (if create a new VCN)
resourceprefix-lb	Subnet for public and private load balancers
resourceprefix-workers	Private subnet for Kubernetes worker nodes
resourceprefix-admin	Private subnet for Kubernetes administration instance
resourceprefix-fss	Private subnet for file shared system
resourceprefix-bastion	Public subnet for bastion instance

Resource Name	Type
resourceprefix-admin-seclist	Security list for the administration instance private subnet
resourceprefix-pub-lb	Security list for the load balancer public subnet
resourceprefix-private-workers	Security list for the worker nodes private subnet
resourceprefix-fss-seclist	Security list for the file shared system private subnet
resourceprefix-bastion	Security list for the bastion instance public subnet
Default Security List for *resourceprefix*-vcn	Default security list for the WebLogic VCN
Default Route Table for *resourceprefix*-vcn	Default route rules in the WebLogic VCN
resourceprefix-nat-route	Route rules table in the WebLogic VCN for NAT and service gateways
resourceprefix-ig-route	Route rules table in the WebLogic VCN for internet gateway
resourceprefix-ig-gw	Internet gateway in the WebLogic VCN
Default DHCP Options for *resourceprefix*-vcn	Default set of Dynamic Host Configuration Protocol (DHCP) options for the WebLogic VCN
resourceprefix-nat-gateway-gw	NAT gateway in the WebLogic VCN
resourceprefix-service-gateway-gw	Service gateway in the WebLogic VCN

Load Balancers

Oracle WebLogic Server for OKE creates a private load balancer for your Oracle WebLogic Server stack and Kubernetes cluster in Oracle Cloud Infrastructure.

Private load balancer is provisioned when you create a stack.

In the Oracle Cloud Infrastructure Console, use the navigation menu under the Core Infrastructure group to go to **Networking** and click **Load Balancers**. When you select the compartment you specified to use for the stack, you'll see the private load balancer provisioned for your WebLogic Server stack and Kubernetes cluster.

Unlike network resources, note that the names of load balancers created by Oracle WebLogic Server for OKE do not begin with the resource name prefix you provided during stack creation. Oracle WebLogic Server for OKE load balancer names are generated, hyphenated alphanumeric strings. For example, `1x1x1x1x-1x1x-1x1x-1x1x1x1x1x1x`.

The private load balancer provides access to the WebLogic Server administration console and the Jenkins console. The private load balancer resource is provisioned with the following:

- A private IP address
- A backend set, which is identified by the name `TCP-80`. The backend set configures the load balancing policy.
- A listener named `TCP-80`. The listener handles traffic on port 80.

Kubernetes Resources

Oracle WebLogic Server for OKE provisions a Kubernetes cluster for your Oracle WebLogic Server stack in Oracle Cloud Infrastructure.

To view the Kubernetes cluster provisioned for your WebLogic Server stack:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  and select **Developer Services**.
3. Under the **Containers** group, click **Kubernetes Clusters**.
4. Select the compartment you specified to use for the stack.

The cluster and node resource names are as follows:

- The Kubernetes cluster name begins with the resource name prefix you provided during stack creation. For example, *resourceprefix*-cluster.
- A node pool named resourceprefix-non-wls-np, with one or more worker nodes for each node pool
- The worker nodes are compute instances with the names oke-*generated-alphanumeric-string*-0 and oke-*generated-alphanumeric-string*-1.

File System Resources

Oracle WebLogic Server for OKE creates a shared file system that is made available through a mount target.

In the Oracle Cloud Infrastructure Console, use the navigation menu and select **Storage**. Under the **File Storage** group, click **File Systems** or **Mount Targets**. When you select the compartment you specified to use during stack creation, you'll see the resources created for the shared file system and mount target:

- *resourceprefix*-fss
- *resourceprefix*-mntTarget

Note that both resource names begin with the resource name prefix you provided during stack creation.

Registry Resources

During stack creation, Oracle WebLogic Server for OKE pushes a default image to the registry. The default image is used to provision the WebLogic Server and Jenkins pods for your domain.

After the stack is created, you can use Kubernetes in the administration compute instance to apply any changes you make to the default image.

In the Oracle Cloud Infrastructure Console, use the navigation menu and select **Developer Services**. Under the **Containers and Artifacts** group, click **Container Registry**, and then select the required **Compartment**. The registry resources for your stack begin with the resource name prefix you provided during stack creation.

The list of registry resources provisioned include:

- *resourceprefix*/infra/cisystem-jenkins-controller
- *resourceprefix*/infra/cisystem-jenkins-agent
- *resourceprefix*/infra/nginx-ingress-controller
- *resourceprefix*/infra/oraclelinux
- *resourceprefix*/infra/weblogic-kubernetes-operator
- *resourceprefix*/wls-base-image/12214

Identity Resources for Dynamic Group and Root Policies

Oracle WebLogic Server for OKE creates a dynamic group and one policy for your domain when you create a stack.

The dynamic group and root-level (tenancy) policy allows compute instances in the domain to access keys and secrets in Oracle Cloud Infrastructure Vault.

The name of the dynamic group and root-level policy are:

- *servicename*-admin-instance-principal-group (dynamic group)
- servicename-oke-encryption-key-principal-group
- *servicename*-oke-policy

Where *servicename* is the resource name prefix you provided during stack creation.

For a single compartment, the matching rule created in the dynamic group is:

```
instance.compartment.id='ocid1.compartment.oc1..alongstring'
```

The rule states that all instances created in the compartment (identified by the compartment OCID) are members of the dynamic group.

The `osms` policy has the following statement:

```
Allow dynamic-group servicename-admin-instance-principal-group to use osms-managed-instances in tenancy
```

The `oke-policy` policy at the root level (tenancy) has the following statements that are scoped to the compartent IDs, resource IDs, or both compartment and resource IDs:

- `Allow dynamic-group servicename-admin-instance-principal-group to use dynamic-groups in tenancy where target.group.id = <dynamic_group_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to manage all-resources in compartment id <stack_compartment_ocid>`
- `Allow service oke to read app-catalog-listing in compartment id <stack_compartment_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to read secret-bundles in tenancy where target.secret.id = <OCID for OCIR token secret>`
- `Allow dynamic-group servicename-admin-instance-principal-group to use vnics in compartment id <network_compartment_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to inspect instance-family in compartment id <network_compartment_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to use subnets in compartment id <network_compartment_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to use keys in tenancy where target.key.id = <oke_encryption_key_ocid>`
  This policy applies if cluster encryption is selected.

3
Manage WebLogic Domains

Learn how to manage a WebLogic domain after creating it with Oracle WebLogic Server for OKE.

Topics:

About Managing a WebLogic Domain

Learn how to create and deploy custom domain images, and access the different consoles.

In general, you configure, manage, and maintain an Oracle WebLogic Server for OKE domain just like an on-premise domain. For example, to deploy an application, see Roadmap for Deploying Applications in WebLogic Server (12.2.1.4)

About WebLogic Deploy Tooling

Learn about the WebLogic Deploying Tool (WDT) to manage a WebLogic domain in Oracle WebLogic Server for OKE.

WebLogic Deploy Tooling automates the domain creation and application deployment tasks using the metadata model that describes the domain and applications (with their dependent resources). WebLogic Deploy Tooling also provides single-purpose tools that perform domain lifecycle operations based on the content of the model.

WebLogic Deploy Tooling is used to create the container image and represent the WebLogic configuration. It makes use of a model Yaml file, an archive zip, and a properties file.

When you have multiple WDT files, the WebLogic Deploy Tooling merges them together and create a single WebLogic domain configuration represented by a single model Yaml file.

The model Yaml file that represents your WebLogic configuration is located in `/u01/shared/` location.

Following is an example of a model Yaml file that has the name of the application as `my-application` and target cluster as `my-WebLogic-domain-cluster`:

```
appDeployments:
    Application:
      my-application :
        SourcePath: wlsdeploy/applications/my-application
        Target: my-WebLogic-domain-cluster
        ModuleType: ear
```

Following is an example of a model Yaml file that has only WebLogic credentials:

```
domainInfo:
    AdminUserName: '@@SECRET:__weblogic-credentials__:username@@'
    AdminPassword: '@@SECRET:__weblogic-credentials__:password@@'
```

Following is an example of a model Yaml file that has RCU database connection information:

```
domainInfo:
    RCUDbInfo:
      rcu_prefix: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-access:rcu_prefix@@'
      rcu_schema_password: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-
access:rcu_schema_password@@'
      rcu_db_conn_string: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-
access:rcu_db_conn_string@@'
      rcu_db_user: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-access:rcu_db_user@@'
      rcu_admin_password: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-
access:rcu_admin_password@@'
```

To know more, see Metadata Model in *WebLogic Deploy Tooling*.

Project Components

Learn about the different tools and files that you can use to build Jenkins projects for your domain.

Oracle WebLogic Server for OKE creates a private load balancer for your domain, which you use to access the Jenkins console running on the Kubernetes cluster. An NGINX ingress controller is used to route traffic from the private load balancer to Jenkins.

All Jenkins files are found on the shared file system.

- `/u01/shared/var/jenkins_home` - The Jenkins controller server configuration
- `/u01/shared/scripts/pipeline` - The resources used to run the sample Jenkins job, including scripts and metadata files
- `/u01/shared/scripts/pipeline/samples` - Sample domain metadata files for deploying applications, applying patches, and so on.

To access or modify these files, use the administration compute instance for your domain. This compute instance also includes the following software:

- `kubectl` - Deploy and manage pods in the Kubernetes cluster for this domain.

- `docker` - Download, modify, and update container images in Oracle Cloud Infrastructure Registry. Use the `login` command to connect to the registry.
- `oci` - View, create and update resources in Oracle Cloud Infrastructure.

Access Resources

Learn how to access the administration console using the tools provided with Oracle WebLogic Server for OKE.

Topics:

Access the Administration Instance

Access the administration compute instance for a Oracle WebLogic Server for OKE.

From the administration compute instance, you can access the shared file system at `/u01/shared`. It also includes the following software:

- `kubectl` - Deploy and manage pods in the Kubernetes cluster for this stack.
- `docker` - Download, modify and update container images in Oracle Cloud Infrastructure Registry. Use the `login` command to connect to the registry.
- `oci` - View, create and update resources in Oracle Cloud Infrastructure.

This compute instance is on a private subnet and cannot be directly accessed from the public Internet. You can use the bastion instance, which is on a public subnet, and the proxy option of a secure shell (SSH) utility.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.
5. Click **Jobs**.
6. In the **Jobs** table, click the Apply job for the stack.
7. Click **Application Information**.
8. Identify and make a note of the following IP addresses:
   - `Bastion Instance Public IP` - The public IP address of the bastion compute instance
   - `Admin Instance Private IP` - The private IP address of the administration compute instance
9. From your computer, create an SSH connection to the administration instance's IP address, and also specify the bastion instance's IP address as a proxy.

   Connect as the `opc` user.

Provide the path to the private key that corresponds to the public key that you specified when you created the stack.

The SSH command format is:

```
ssh -i <path_to_private_key> -o ProxyCommand="ssh -W %h:%p -i
<path_to_private_key> opc@<bastion_public_ip>" opc@<admin_ip>
```

For example:

```
ssh -i ~/.ssh/mykey.openssh -o ProxyCommand="ssh -W %h:%p -i ~/.ssh/
mykey.openssh opc@203.0.113.13" opc@198.51.100.1
```

On a Windows platform, you can use Windows PowerShell to run the SSH command.

10. If prompted, enter the passphrase for the private key.

Access the Jenkins Console

Access the Jenkins build engine for a stack that you created with Oracle WebLogic Server for OKE.

Jenkins runs as a pod in the Kubernetes cluster and is accessible from a private load balancer. This load balancer cannot be directly accessed from the public Internet. You can use the bastion instance on a public subnet and dynamic port forwarding with a secure shell (SSH) utility.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.
5. Click **Jobs**.
6. Click the Apply job for the stack.
7. Click **Outputs**.
8. Identify the public IP address of the bastion compute instance, `bastion_instance_public_ip`.
9. Click **Logs**.
10. Search for the attribute `jenkins_console_url`. Copy the URL.

    The URL format is:

    ```
    jenkins_console_url=http://<internal_lb_ip>/jenkins
    ```

    where, `internal_lb_ip` is the internal load balancer IP address.

> **Note:**
>
> If you provision a domain without a bastion instance, you must obtain the internal load balancer IP address to use in the Jenkins URL.
>
> To get the internal load balancer IP address, access the administration instance for your stack (see Access the Administration Instance) and run the following command:
>
> ```
> kubectl get svc -A
> ```
>
> The private load balancer is listed with the namespace `wlsoke-ingress-nginx` and name *okename*`-internal`.

11. From your computer, open an SSH tunnel to an unused port on the bastion compute instance as the `opc` user.

    For example, you can use port `1088` for SOCKS proxy.

    Specify the `-D` option to use dynamic port forwarding. Provide the path to the private key that corresponds to the public key that you specified when you created the domain.

    The SSH command format is:

    ```
    ssh -D <port_for_socks_proxy> -fCqN -i <path_to_private_key>
    opc@<bastion_public_ip>
    ```

    For example:

    ```
    ssh -D 1088 -fCqN -i ~/.ssh/mykey.openssh opc@198.51.100.1
    ```

    On a Windows platform, you can use Windows PowerShell to run the SSH command.

12. In your browser settings, set up the SOCKS (version 5) proxy configuration. Specify your local computer and the same SOCKS port that you used in your SSH command.
13. Browse to the Jenkins console URL.
14. If this is the first time using the Jenkins console, you are prompted to create a new admin user.

Access the WebLogic Console

Access the WebLogic Server Administration Console for a domain that you created with Oracle WebLogic Server for OKE.

> **Note:**
>
> - Before you access the WebLogic console, you must have created a WebLogic domain. See Create a WebLogic Domain.
> - Do not use the WebLogic console to make any configuration changes. All configuration changes should be should done through jobs, this ensure that the changes are persistent.
> - Security check warnings are displayed at the top of the console. See About the Security Checkup Tool for the warnings and how to handle them.

The domain's administration server runs as a pod in the Kubernetes cluster and is accessible from a private load balancer. This load balancer cannot be directly accessed from the public Internet. You can use the bastion instance on a public subnet and dynamic port forwarding with a secure shell (SSH) utility.

> **Note:**
>
> Any modifications you make to a running domain will be lost when you redeploy the pods for the domain in the Kubernetes cluster.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.
5. Click **Jobs**.
6. In the **Jobs** table, click the Apply job for the stack.
7. Click **Outputs**.
8. Identify and make a note of the public IP address of the bastion compute instance, `bastion_instance_public_ip`.
9. Click **Logs**.
10. Search for the attribute `weblogic_console_url`. Copy the URL.

    The URL format is:

    `weblogic_console_url=http://<internal_lb_ip2>/<domain-name>/console`

where, `<internal_lb_ip2>` is the internal load balancer IP address that is used to access the Jenkins console.

> **Note:**
>
> If you provision a domain without a bastion instance, you must obtain the internal load balancer IP address to use in the WebLogic Console URL.
>
> To get the internal load balancer IP address, access the administration instance for your stack (see Access the Administration Instance) and run the following command:
>
> ```
> kubectl get svc -A
> ```
>
> The private load balancer is listed with the namespace `wlsoke-ingress-nginx` and name *okename*`-internal`.

Example: This example will help you obtain the WebLogic Console URL.

After you access the administration instance for your stack, run the following command:

```
kubectl get svc -A
```

The output appears as follows:

```
NAMESPACE              NAME                                TYPE
CLUSTER-IP      EXTERNAL-IP      PORT(S)                     AGE
default                kubernetes                          ClusterIP
10.96.0.1       <none>           443/TCP,12250/TCP           9d
service1-operator-ns   weblogic-operator-webhook-svc       ClusterIP
10.96.30.88     <none>           8083/TCP,8084/TCP           9d
domain1-ns             domain1-cluster-domain1-cluster     ClusterIP
10.96.40.171    <none>           8001/TCP                    2d23h
domain1-ns             domain1-domain1-adminserver         ClusterIP
None            <none>           30012/TCP,7001/TCP          2d23h
domain1-ns             domain1-domain1-managed-server1     ClusterIP
None            <none>           8001/TCP                    2d23h
domain1-ns             domain1-domain1-managed-server2     ClusterIP
None            <none>           8001/TCP                    2d23h
jenkins-ns             jenkins-service                     ClusterIP
10.96.13.193    <none>           8080/TCP,50000/TCP          9d
kube-system            kube-dns                            ClusterIP
10.96.5.5       <none>           53/UDP,53/TCP,9153/TCP      9d
wlsoke-ingress-nginx   service1-internal                   LoadBalancer
10.96.190.50    10.10.6.13       80:31149/TCP                9d
```

Now, look for the **EXTERNAL-IP** field for the service with the following details:

- **NAMESPACE** = `wlsoke-ingress-nginx`
- **NAME** = `<`*service-name*`>-internal` = `service1_internal`
- **TYPE** = `LoadBalancer`

For this example, `service1` is the service name and `domain1` is the domain name. The WebLogic Console URL format is `http://<internal_lb_ip>/<domain-name>/console`. Therefore, as per this example, the console URL is `http://10.10.6.13/domain1/console`.

11. From your computer, open an SSH tunnel to an unused port on the bastion compute instance as the `opc` user.

    For example, you can use port `1088` for SOCKS proxy.

    Specify the `-D` option to use dynamic port forwarding. Provide the path to the private key that corresponds to the public key that you specified when you created the stack.

    The SSH command format is:

    ```
    ssh -C -D port_for_socks_proxy -i path_to_private_key opc@bastion_public_ip
    ```

    For example:

    ```
    ssh -C -D 1088 -i ~/.ssh/mykey.openssh opc@198.51.100.1
    ```

    On a Windows platform, you can use Windows PowerShell to run the SSH command.

12. In your browser settings, set up the SOCKS (version 5) proxy configuration. Specify your local computer and the same SOCKS port that you used in your SSH command.
13. Browse to the console URL.
14. Sign in using the administrator credentials for your domain.

Create a WebLogic Domain

Create a WebLogic domain by using the tools provided with Oracle WebLogic Server for OKE.

Topics:

About Creating a Domain

Learn about the options you have when creating a domain.

- Domain Type

  A Non-JRF domain does not require an existing database.

  A JRF-enabled domain includes the Java Required Files (JRF) components and requires access to an existing database in Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System). If using a DB System database, note that the DB System and the Kubernetes cluster must be in the same Virtual Cloud Network (VCN). See Create a Database.

- Load Balancers

  When you create a domain, Oracle WebLogic Server for OKE creates a public load balancer to distribute application traffic to the WebLogic cluster.

The public load balancer consists of primary and standby nodes but it is accessible from a single IP address. If the primary node fails, traffic is automatically routed to the standby node. The public load balancer is configured for SSL connections (the HTTPS protocol) that terminate at the load balancer.

The load balancers are assigned to a public subnet, for which you must specify a CIDR block if you let Oracle WebLogic Server for OKE create new subnets during stack provisioning. You must also specify shapes for the private and public load balancers.

> **Note:**
>
> By default, the reserved public IP address that you specify as the `loadBalancerIP` property of the `LoadBalancer` service in the manifest file is expected to be a resource in the same compartment as the cluster. If you want to specify a reserved public IP address in a different compartment, add the following policy to the tenancy:
>
> ```
> Allow any-user to read public-ips in tenancy where
> request.principal.type = 'cluster'
> Allow any-user to manage floating-ips in tenancy where
> request.principal.type = 'cluster'
> ```
>
> See Specifying Load Balancer Reserved Public IP Addresses.

Prerequisites to Create a Domain

Complete the prerequisites before you create a domain.

Required Tasks

- Create Policies for the Dynamic Group
- Create a Database
- Create a Confidential Application
- Approve Scripts to View Parameters
- Validate Existing Network Setup

Create Policies for the Dynamic Group

Create policies in Oracle Cloud Infrastructure so that the compute instances in Oracle WebLogic Server for OKE can access your encryption key.

When you create a domain, compute instances in Oracle WebLogic Server for OKE need to access specific components in Oracle Cloud Infrastructure.

The following sample policy grants the relevant database permissions to a dynamic group:

```
Allow dynamic-group MyInstancesPrincipalGroup to use autonomous-transaction-
processing-family in compartment ATP_Database_Compartment
Allow dynamic-group MyInstancesPrincipalGroup to inspect db-systems in
compartment id OCIDBCompartmentID
Allow dynamic-group MyInstancesPrincipalGroup to inspect databases in
compartment id OCIDBCompartmentID
```

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Writing Policies for Dynamic Groups

Create a Database

If you are using Domain on PV, a WebLogic Kubernetes Operator (WKO) domain home source type, to create an Oracle WebLogic Server Java Required Files (JRF) domain, then, before creating the domain that includes the JRF components, you must create a database in Oracle Cloud Infrastructure.

A JRF-enabled domain is used by Fusion Middleware products. The JRF domain has database requirements. The database components are created using the Repository Creation Utility (RCU); a new RCU schema is created before creating a JRF-based domain.

A JRF-enabled domain includes the Java Required Files (JRF) components and requires access to an existing database in Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System). If using a DB System database, ensure that the DB System and the Kubernetes cluster are in the same Virtual Cloud Network (VCN).

> **Note:**
>
> For each schema that is created in the database, a data source is created in WebLogic Server. These data sources should not be used by applications deployed to the WebLogic domain after provisioning is complete. Instead, you must create independent data sources. See About Data Sources.

Choose one of these database options:

- Oracle Autonomous Database
  - Create a serverless database. Oracle WebLogic Server for OKE does not yet support using a dedicated deployment database.
  - See Creating an Autonomous Database in the Oracle Cloud Infrastructure documentation.

    > **Note:**
    >
    > Oracle Application Express (APEX) autonomous database is not supported.

- Oracle Cloud Infrastructure Database
  - Create a bare metal, virtual machine (VM), or Exadata DB system.
  - The Virtual Cloud Network (VCN) of the Oracle Cloud Infrastructure database must be same as the WebLogic Server VCN.
  - See Creating Bare Metal and Virtual Machine DB Systems or Managing Exadata DB Systems in the Oracle Cloud Infrastructure documentation.

The database must allow your domain to access its listen port (1521 by default):

- Oracle Autonomous Database - Update your access control list (ACL), if necessary.

- Oracle Cloud Infrastructure Database - Update the network security group that is assigned to the database, or update the security lists for the subnet on which the database was created, if necessary.

To create a JRF-enabled domain with Oracle WebLogic Server for OKE, you need the following information about the database:

- Administrator credentials
- Oracle Cloud Identifier (OCID) of the Oracle Cloud Infrastructure Database database system or the Autonomous Transaction Processing (ATP) database. This information is optional if you use a database connection string.

Oracle WebLogic Server for OKE supports the same database versions and drivers as those for on-premise WebLogic Server installations. Refer to the following documents at Oracle Fusion Middleware Supported System Configurations:

- System Requirements and Supported Platforms for Oracle Fusion Middleware 14c (14.1.1.0.0)
- System Requirements and Supported Platforms for Oracle Fusion Middleware 12c (12.2.1.4.0)

Create a Confidential Application

Before creating an Oracle WebLogic Server for OKE domain that integrates with Oracle Identity Cloud Service, you must create a confidential application, and then identify its client ID and client secret.

This configuration is supported only for Oracle Cloud accounts that include Oracle Identity Cloud Service 19.2.1 or later.

When creating a new domain, Oracle WebLogic Server for OKE provisions an App Gateway and other security components in Oracle Identity Cloud Service. In order for Oracle WebLogic Server for OKE to perform these tasks, you must provide the following information:

- Your Oracle Identity Cloud Service instance ID, which is also referred to as your tenant name. This ID is typically found in the URL you use to access the Oracle Identity Cloud Service console, and has the format `idcs-<GUID>`.
- The client ID of a confidential application in Oracle Identity Cloud Service
- The client secret of the confidential application.

Create a confidential application for Oracle WebLogic Server for OKE, or use an existing one. You can use a single confidential application in Oracle Identity Cloud Service to create multiple domains.

1. From the Oracle Identity Cloud Service Console, click the navigation menu, and then select **Applications**.
2. Click **Add**.
3. Select **Confidential Application**.
4. Enter a **Name**, and then click **Next**.
5. Click **Configure this application as a client now**.
6. For **Allowed Grant Types**, select **Client Credentials**.
7. Below **Grant the client access to Identity Cloud Service Admin APIs**, click **Add**.
8. Select **Identity Domain Administrator**, and then click **Add**.

9. (Optional) For a WebLogic Server 12.2.1.4 domain only, add **Cloud Gate App Role**. You can add this role after you create your WebLogic Server domain but you may need to restart the domain.

> **Caution:**
>
> Add Cloud Gate App Role only if you need to open and log in to the Fusion Middleware Control Console from the Internet. While enabling this role means the Fusion Middleware Control Console is accessible from the Internet, it also means any application would be allowed to look up users.

10. Complete the Add Confidential Application wizard. Record the values of **Client ID** and **Client Secret**.
11. Select the check box for your application, click **Activate**, and then click **OK**.
12. In the Oracle Cloud Infrastructure console, create a secret in a vault to store the client secret of your confidential application.

See Add a Confidential Application in *Administering Oracle Identity Cloud Service*.

Approve Scripts to View Parameters

At times, the Jenkins UI input parameters in a list are not rendered. So, you need to approve groovy scripts to view all the parameters in a list.

Complete the following steps to approve the scripts:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. Go to **Dashboard** > **Manage Jenkins**.
3. Under Security, click **In-process Script Approval**.
4. Click **Approve** against all the groovy scripts.
   All the parameters are now listed in the pipeline jobs.

Validate Existing Network Setup

You can use helper scripts from the Oracle Cloud Infrastructure Cloud shell to certify the existing network setup (existing VCN and existing WebLogic Server subnet) in Oracle WebLogic Server for OKE. See Using Cloud Shell in Oracle Cloud Infrastructure documentation.

The helper scripts perform the following validations and functions:

- Validates if the service gateway or the NAT gateway is created for the administration instance private subnet and the worker nodes private subnets.
- Validates if internet gateway is created for public bastion, file shared system and load balancer subnets.
- Checks if port 22 in WebLogic Server Subnet is open for access to the CIDR of the bastion instance subnet or bastion host IP.
- Checks if the private subnet for the Oracle WebLogic Server compute instances using the service gateway route rule has **All *<Region>* Services In Oracle Services Network** as the destination.
- Checks if the existing subnet for the load balancer has a security list that enables inbound access to ports 80 and 443.

- Validates if all protocols are open in private subnet for Kubernetes worker node for the Worker CIDR range.
- Validates if all protocols are open in private subnet for Kubernetes worker node for the VCN CIDR range.
- Validates if the file shared system has a security list that enables outbound access to ports 111 and 2048 (both TCP and UDP).
- Validates if the database port is accessible from WebLogic Server subnets.

Using the Validation Script

You can run the helper scripts to perform validations for existing private subnets, existing public subnets, and existing VCN peered subnets.

You must run the commands on the validation script file to check the existing network setup. For example, in this case, let's run the commands on the validation script file named `validateoke.sh`. See Script File To Validate Network Setup to create the `validateoke.sh` file.

1. Set execute permission to the `validateoke.sh` file.

   ```
   chmod +x validateoke.sh
   ```

2. Run the following command prior to creating a domain:

   - Basic domain

     ```
     ./validateoke.sh -b <Bastion Subnet OCID> -a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File Shared System Subnet OCID> -l <Load Balancer Subnet OCID>
     ```

> **Note:**
>
> If you restricted the bastion compute instance to access port 22 in WebLogic subnet, you can validate using the `Bastion Host IP CIDR` rather than the entire bastion subnet CIDR.
>
> ```
> ./validateoke.sh -b <Bastion Subnet OCID> -i <Bastion Host IP CIDR> -a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File Shared System Subnet OCID> -l <Load Balancer Subnet OCID>
> ```

`validateoke.sh`

```
example_user@cloudshell:~ (us-phoenix-1)$ ./validateoke.sh -b <Bastion Subnet OCID>
-a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File Shared System Subnet OCID>
-l <Load Balancer Subnet OCID>
ERROR: SSH port 22 is not open for access by [0.0.0.0/0] in <Bastion Subnet OCID>
WARNING: SSH port 22 is not open for access by Bastion Subnet CIDR
[10.0.0.0/24] in private Admin Host Subnet [<Administration Host Subnet OCID>]
ERROR: Missing Service or NAT gateway in the VCN of the private ADMIN_SUBNET
Host subnet ocid [<Administration Host Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the BASTION_SUBNET subnet
[<Bastion Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the LB_SUBNET subnet [<Load Balancer Subnet OCID>]
```

```
WARNING: Missing internet gateway in the VCN of the FSS_SUBNET_OCID subnet
[<File Shared System Subnet OCID>]
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 31474
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 10256
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 31804
WARNING: All Ports are not open for LB Subnet CIDR
WARNING: All Ports are not open for LB Subnet CIDR
WARNING: All Ports are not open for LB Subnet CIDR
ERROR: All Protocols are not open for WORKER's Subnet CIDR
ERROR: All Protocols are not open in WORKER's Subnet for VCN CIDR
ERROR: TCP -- 111 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2048 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2049 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2050 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 111 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2048 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2049 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2050 -- Port is not open in FSS Subnet for VCN CIDR
```

Create a Domain

After you create a stack, use the Jenkins job to create a domain for Oracle WebLogic Server for OKE.

You can locate a WebLogic domain either on a persistent volume (Domain on PV) or inside the container (Model in Image). For a comparison between the two types, see Choose a Domain Home Source Type.

- Model in Image:
  With Model in Image, you do not need to create your WebLogic domain home beforehand. This tool uses a WebLogic Deploy Tooling (WDT) model to define the WebLogic configuration. It supports standard WebLogic Server domains. For more information, see Model in Image.

  You can create a Model in Image within the container image using the **create mii domain** job.

  When you create a domain with the **create mii domain** job, a new non-Java Required Files (JRF) domain that has a basic configuration with no custom applications or libraries, called the primordial domain, is created. This domain contains:

  – Base WebLogic Server image that has the WebLogic installer, JDK installer, and WebLogic patches for Oracle WebLogic Server for OKE.

  – Primordial Auxiliary image that has WebLogic domain resources (server, cluster, JDBC, and other resources).

  To know about the primordial domain and auxiliary images, see Mutate the Domain Layer and Auxiliary Images in *WebLogic Kubernetes Operator* documentation.

- Domain on Persistent Volume (PV):
  Domain on PV creates the domain on a persistent volume. Domain on PV is applicable for two types of domains: a standard Oracle WebLogic Server (WLS) domain and an Oracle Fusion Middleware Infrastructure, Java Required Files (JRF) domain. For more information, see Domain on Persistent Volume (PV).

  You can create and deploy a domain on persistent volume (Domain on PV) using the **create pv domain** job. With this job, you can create both a non-Java Required Files (JRF) domain or a Java Required Files (JRF) domain, depending on your requirement.

Before you create a domain, ensure that all the prerequisites are completed. See Prerequisites to Create a Domain.

Topics:

- Provision a Non-JRF Domain
- Provision a JRF Domain

Provision a Non-JRF Domain

You can use the Jenkins pipeline job `create mii domain` or `create domain on pv` to automate the deployment of a non-JRF WebLogic Server doamin. The job you choose will depend on whether you want to use Model in Image domain home source type or the Domain on Persistent Volume domain home source type for deploying a domain.

Complete the following steps to create a non-JRF domain using `create mii domain` or `create domain on pv`, based on your requirement.

Topics:

- Configure WebLogic Server
- Configure the Registry
- Configure the Container Cluster
- Configure the Load Balancer
- Configure the Domain
- Configure Identity Cloud Service Integration
- Create the Domain

Configure WebLogic Server

Specify the parameters required to configure a WebLogic server on a container cluster.

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **create domain**.
3. Click **Build with Parameters**.
4. For **Domain_Name**, specify a WebLogic name.
5. For **WebLogic_Version**, select a version of Oracle WebLogic Server.

   The available versions are 12.2.1.4.0, 14.1.1.0.0 running on JDK 8, and 14.1.1.0.0 running on JDK 11.
6. *Optional*: Select the required base image from the **Base_Image** list.

   The images are displayed based on the Weblogic Server version. For example, if you select 12.2.1.4.0, 12.2.1.4.0 images are displayed, and if you select 14.1.1.0.0, 14.1.1.0.0_jdk8 and 14.1.1.0.0_jdk11 images are displayed.

> **Note:**
>
> For 14.1.1.0.0., make sure that the base image you select should have the same WebLogic Server and JDK version as in step 5.
>
> The custom base images and uploaded images are also listed in the **Base_Image** list. See Create a Custom Base Image.

7. Enter a user name for the WebLogic Server administrator.
8. Enter the password for the WebLogic Server administrator.
9. Select the number of running managed servers in the domain you want to create. You can specify up to `9` manged servers.

   The number of running managed servers is also the number of WebLogic Server pods in the Kubernetes cluster. Each managed server runs in a separate pod in the Kubernetes cluster.

   Managed servers are members of a WebLogic Server cluster.
10. Specify the time in minutes to wait for the WebLogic Domain Server pods to start in the running state. The default wait time is 40 minutes.
11. Select **Patch_Automatically**, if you want the domain to be subscribed for automatic patching.

    Once subscribed, the domain is patched periodically with the latest patches available in the patching repository. See Automatic Patching.
12. If your previous `create domain` job failed, then select **Cleanup_Domain_Resources** to cleanup any existing domain resources.

Configure the Registry

Specify the credentials to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

> **Note:**
>
> If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

1. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.

   The registry user name format is `tenancy_namespace/<username>`. If your tenancy is federated with Oracle Identity Cloud Service, then the registry user name format is `tenancy_namespace/oracleidentitycloudservice/<username>`.

   You can choose either to include the `tenancy_namespace` or remove the `tenancy_namespace` in the user name format. For example, you can either use `tenancy_namespace/<username>` or `<username>`.

> **Note:**
>
> If you choose to include `tenancy_namespace` in the user format, ensure that you use the correct namespace for your tenancy.

2. In the **Registry_Authentication_Token** field, select the compartment where you have the OCI Secret that contains the auth token.

For information about how to create a container registry, see Overview of Registry in the Oracle Cloud Infrastructure documentation.

Configure the Container Cluster

Specify the parameters required to either create a node pool or select an existing node pool for the WebLogic nodes.

- Use an Existing Node Pool
- Create a Node Pool

Use an Existing Node Pool

To use an existing node pool:

1. From **WebLogic_Node_Pool_Type**, select **Existing_Node_Pool**.
2. From **Existing_Node_Pool**, select the required node pool.

> **Note:**
>
> In the **Existing_Node_Pool** list, the node pools, if any, created during stack creation, and the node pools that are idle, that is, do not have any domains running in them and which are created using the `create mii domain` or the `create domain on pv` job are listed.

Create a Node Pool

To create a node pool:

1. From **WebLogic_Node_Pool_Type**, select **Create_Node_Pool**.
2. For **Node_Count**, specify the number of nodes your want for the WebLogic node pool.
3. For **Node_Pool_Name**, specify the name of the node pool.
4. From **WebLogic_Node_Pool_Shape**, select a shape for each node in the Kubernetes cluster node pool, for the WebLogic node pools.

   For 2 or more running managed servers, select a shape with 2 or more OCPUs. For example, `VM.Standard2.2` instead of `VM.Standard2.1`.

   If you select a flexible shape, specify the OCPU count and the amount of memory for the WebLogic node pool shape. The amount of memory is based on the number of OCPUs.

> **Note:**
>
> If you specify the amount of memory that is not allowed for the number of OCPUs, the node pool creation fails. See Flexible Shapes.

5. *Optional*: For **SSH_Public_Key**, specify the contents of the SSH public key to access the nodes in the WebLogic server node pool in this domain.

   If you want to use another SSH public key for this domain, other than the one specified when creating a stack, then enter the SSH public key by copy-pasting the SSH key information.

> **Note:**
>
> If you use another SSH public key, the new SSH public key is used to access the nodes in the WebLogic server node pool. The SSH key for accessing the Administrator node is not changed, which you specified when creating a stack.

6. *Optional*: For **NodePool_Subnet_ID**, if you want the node pool to be created in a specific private subnet, then specify the Oracle Cloud Identifier (OCID) of that private subnet.

> **Note:**
>
> - Ensure that the private subnet exists in the same VCN as the Kubernetes cluster.
> - If you want the node pool in another subnet, then you must to set following additional security rules:
>   - In the `oke_endpoint` security list, allow access on ports `6443` and `12550` for the subnets where you want the nodepool created.
>   - In the `workers_subnet` security list, allow access for all protocols for the destination subnet. This must be an ingress rule with the destination subnet CIDR being the source CIDR.

Configure the Load Balancer

Specify the parameters required to create a public load balancer for the application. The public load balancer is used to access applications on the WebLogic managed servers.

1. For **External_Lb_Shape_Min** and **External_Lb_Shape_Max**, specify the minimum and maximum flexible shape for a public load balancer.

   By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

> **Note:**
>
> You can update the shape to a maximum of 8000 Mbps. Before you select the maximum bandwidth, ensure to check the available service limit for the flexible public load balancer bandwidth.

2. *Optional*: Enter the **LB_Subnet_ID** of the load balancer subnet.

> **Note:**
>
> Ensure that the subnet exists in the same VCN as the Kubernetes cluster. If you do not specify the OCID, the load balancer is created in the same subnet as the load balancer subnet you specified during stack creation.

3. Select **Private_Load_Balancer**, if you want to create a private load balancer for your applications.
4. If you want to use a public load balancer with a reserved public IP, then in **Reserved_Public_IP**, specify the public IP for the load balancer.

> **WARNING:**
>
> If you create a load balancer in a private subnet, you must not specify the reserved public IP address, else the domain creation fails.

> **Note:**
>
> By default, the reserved public IP address that you specify as the `loadBalancerIP` property of the `LoadBalancer` service in the manifest file is expected to be a resource in the same compartment as the cluster. If you want to specify a reserved public IP address in a different compartment, add the following policy to the tenancy:
>
> ```
> Allow any-user to read public-ips in tenancy where
> request.principal.type = 'cluster'
> Allow any-user to manage floating-ips in tenancy where
> request.principal.type = 'cluster'
> ```
>
> See Specifying Load Balancer Reserved Public IP Addresses.

Configure Identity Cloud Service Integration

You have the option to use IDCS to authenticate application users for your domain. To enable IDCS, specify the parameters required to configure WebLogic Authentication with Oracle Identity Cloud Service (IDCS).

To use Oracle Identity Cloud Service for authentication:

1. From **IDCS_Enabled**, select **YES**.
2. For **IDCS_Host_Name**, specify the required host name.

   The default value of the port name is displayed. If required, you can override the port that you use to access Oracle Identity Cloud Service.
3. For **IDCS_Tenant**, specify your IDCS tenant name, which is also referred to as the instance ID.

This ID is usually found in the URL that you use to access IDCS, and has the format `idcs-<GUID>`

4. For **IDCS_Client_ID** and **IDCS_Client_Secret**, specify the client ID and the password.

   The client ID and secret are from the confidential application that you created as a prerequisite to create a domain. See Create a Confidential Application.

5. In **IDCS_Redirect_Port**, the default port used for the IDCS App Gateway is displayed. If required, you can override the default port.

Configure the Domain

If you are using the `create domain on pv` job to deploy a non-JRF domain, in the Provision with JRF section of the page, keep the default selection for **Domain_Type** as **Non_JRF**.

Create the Domain

Click **Build** to run the job.

After the job is successful, you can access the WebLogic Console. See Access the WebLogic Console.

Provision a JRF Domain

You can use the `create domain on pv` job to provision a JRF domain. The Oracle WebLogic Server domain includes the Java Required Files (JRF) components, network resources, Kubernetes cluster, compute instances, and load balancers. For more information about JRF domains, see JRF Domain.

Creating a JRF-enabled domain is similar to creating a basic domain; however, a database in Oracle Cloud Infrastructure is required. You can specify a database in Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System). If you plan to use a DB System database, note that the DB System and the Oracle WebLogic Server for OKE compute instances must be in the same virtual cloud network (VCN).

> **Note:**
>
> - For each schema that is created in the database, a data source is created in WebLogic Server. These data sources should not be used by applications deployed to the WebLogic domain after provisioning is complete. Instead, you must create independent data sources. See About Data Sources.
> - Oracle WebLogic Server 14c does not support JRF, so you cannot create a JRF domain using Oracle WebLogic Server 14.1.1.0.0.

The steps to provision a JRF domain is the same as those used for provisioning a non-JRF domain. See Provision a Non-JRF Domain.

Additionally, you should specify a database. See:

- Provision a JRF Domain with an Autonomous Database
- Provision a JRF Domain with an OCI Database

Provision a JRF Domain with an Autonomous Database

To create a JRF domain with an autonomous database:

1. From **Domain_Type**, select **JRF_with_ATP**.
2. For **Database_OCID**, specify the OCID of the autonomous database.
3. For **Database_Password**, enter the database administrator password.
4. For **ATP_DB_LEVEL**, specify the service level that the domain should use to connect to the selected autonomous database.

Provision a JRF Domain with an OCI Database

To create a JRF domain with an OCI database:

1. From **Domain_Type**, select **JRF_with_OCIDB**.
2. For **Database_OCID**, specify the OCID of the OCI database.
3. For **Database_Password**, enter the database administrator password.
4. For **Database_Connection_String**, enter the connect string to connect to the database:

   If you use **Database_Connection_String**, then you can skip specifying **Database_OCID**.

> **WARNING:**
>
> Do not use the database connect string example provided in the **Oracle Database Connection String** field , instead use the format specified in the following table.

Table 3-1 Database Connect String for Database Version and Type

Database Version	Database Type	Database Connection String
12c and above	VM	`//<db_hostname>-scan.<db_domain>:<db_port>/<pdb_name>.<db_domain>`
12c and above	Bare Metal	`//<db_hostname>.<db_domain>:<db_port>/<pdb_name>.<db_domain>`

> **Note:**
>
> If you use database connect string, then Oracle WebLogic Server for OKE creates a single instance datasource. However, you can update the data source for Oracle WebLogic Suite with Active GridLink data source and data source for Oracle WebLogic Server Enterprise Edition with multi data source. See Configuring Active GridLink Connection Pool Features and Configuring JDBC Multi Data Sources.
>
> If using **Database System** with connect string, security list is not created to access the database. You must ensure that the ports are open to access the database.

Update a WebLogic Domain

Learn how to update a domain by using the tools provided with Oracle WebLogic Server for OKE.

You can update a Model in Image domain by using the Jenkins pipeline job `update mii domain`. To update a domain created with the Domain on PV approach, you should use the WebLogic Server Administration Console.

Topics:

Create a Custom Base Image

Use the Jenkins job `create base image` to create a custom base WebLogic image from WebLogic installer, JDK installer, and WebLogic patches for Oracle WebLogic Server for OKE.

You can use one of the following sources to specify the location of the JDK installer file, WebLogic installer file, and WebLogic patch file:

- Object Storage - Uses the pre-authenticated URL on the Object Storage.

  For JDK installer file, you must specify the location of a JDK that uses the Linux x64 compressed archive format (`.tar.gz`).

- Shared File System - Uses the path of the shared file storage.

  The NFS shared file system path is mounted on `/u01/shared` location on the administration host.

> **Important:**
>
> You can get your own preferred WebLogic base image and upload them to the OCIR. You must upload the images to the following location:
>
> `<region>/<tenancy>/<servicename>/wls-base-image/12214`

To create a custom base image:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **create base image**.
3. Click **Build with Parameters**.
4. Select the source of the JDK installer file from the **JDK_Installer** list.
5. For **JDK_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system.
6. From **FMW_Installer**, select the source of the Fusion Middleware installer file.
7. For **FMW_Installer_Location**, specify the http pre-authenticated URL or the path of the JAR file on the shared file system.
8. *Optional:* Select patches.
   a. Select the source of the Weblogic patch file from the **WLS_Opatches** list. **WLS_Opatches** refers to any patch that may be required to patch WLS, ADF/JRF, or Coherence product binaries (including ADR patches, JRF patches, OPSS, OWSM, OPatches).
   b. If you selected a Weblogic patch file: For **Opatches_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system. For multiple patches, specify the location in separate lines.
9. *Optional:* Configure the Registry.

   If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

   a. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.

      If your tenancy is federated with Oracle Identity Cloud Service, use the format: `oracleidentitycloudservice/<username>`, else use `<username>`.

   b. In the **Registry_Authentication_Token** field, enter the OCID of the secret for the auth token generated for the registry user.
10. Click **Build** to run the pipeline job.

    The custom base image is created and is available in the following location.

    `<region>/<tenancy>/<servicename>/wls-base-image/12214`

The created custom base images and uploaded images are listed when you create a domain, in the **Base_Image** list. See Create a Domain .

Update a Domain Configuration

For information about updating the two domain types, see:

- Update a Model in Image Domain
- Update a Domain on a Persistent Volume

Update a Model in Image Domain

You can use the `update mii domain` job to deploy or undeploy applications, shared libraries and resources to the new domain using a sample application or user-provided artifacts

When you run the `update mii domain` job, the primordial domain created using the `create mii domain` job is used as the base domain, and updates are applied to this domain.

Therefore, if you run the `update mii domain` job for the first time and deploy an application-A in this job, the resulting primordial domain will have application-A deployed. Then, if you run the `update mii domain` job for the second time and deploy only application-B in the job, the resulting primordial domain will have only application-B deployed. That is, the state on top of the primordial domain is not stored within the image and this allows you to use the source control system to store WebLogic Deploy Tooling (WDT) model to persist any configuration changes on top of the primordial domain.

To know about the primordial domain, see Mutate the Domain Layer in *WebLogic Kubernetes Operator* documentation.

You can extend your model Yaml file with new definitions using the `update mii domain` job. For model Yaml file, see About WebLogic Deploy Tooling.

> **Tip:**
>
> To deploy and undeploy an application, see Tutorial.

To update the WebLogic domain you can provide a WDT model file, a WDT properties file, or an archive file. The archive file can contain applications, libraries, model file along with other artifacts. For the structure of the archive file, see archive file in *WebLogic Deploy Tooling* documentation.

When you are updating the domain, if you provide a model through the WDT model field and within the archive, the model specified in the field takes precedence.

You can use one of the following sources to specify the location of the archive zip file, domain model Yaml file, and the variables properties file:

- File Upload - Uploads the file from the local system.
- Object Storage - Uses the pre-authenticated URL on the Object Storage.
- Shared File System - Uses the path of the shared file storage.
  The NFS shared file system path is mounted on `/u01/shared` location on the administration host.

> **Note:**
>
> Ensure that you specify the full path of the location of the zip file on the shared file system.

To update a domain and deploy sample applications, shared libraries, and resources to the domain:

Note:

Every time you update a domain, you have to provide a model containing all the resources (applications, libraries, and so on) you want your domain to have. If your domain currently have application-A installed, and you update your domain with a model with application-B, then the domain will have only application-B. If you want to have both applications, you have to use a model with both application-A and application-B.

If you are running the **update domain** job for the first time, then to deploy the application, you must manually approve `new java.io.File java.lang.String` and `method java.io.File` under `jenkins/scriptApproval/`. See Approve Scripts to View Parameters.

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **update mii domain**.
3. Click **Build with Parameters**.
4. From **Domain_Name**, select the domain that you want to update.
5. *Optional*: To update the base image, select the **Update_Base_Image** check box, and then from **Base_Image**, select the image that you want to apply to the selected domain.
6. Select the source of the archive zip file from the **Archive_Source** list.

   Note:

   In case of files larger than one MB, use the **Object Storage** or **Shared File System** option.

7. For **Archive_File_Location**, either browse to select the zip file, specify the http pre-authenticated URL, or specify the path of the zip file on the shared file system.
   For the structure of the archive file, see archive file in *WebLogic Deploy Tooling* documentation.
8. Select the source of the domain model Yaml file from the **Domain_Model_Source** list.
   If the archive zip file contains the domain model Yaml file, you can skip this step.
9. For **Model_File_Location**, either browse to select the Yaml file, specify the http pre-authenticated URL, or specify the path of the Yaml on the shared file system.
   To deploy the sample application, browse and select Yaml file: `deploy_sample_app.yaml` or specify the path, `/u01/shared/scripts/pipeline/samples/deploy_sample_app.yaml`.
   See model YAML file in *WebLogic Deploy Tooling* documentation.
10. Select the source of the variable properties file from the **Variable_Source** list.
11. For **Variable_File_Location**, either browse to select the file, specify the http pre-authenticated URL, or specify the path of the properties file on the shared file system.
12. Deselect the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).

If you deselected this check box, you can rollback to the previous image later from the backup.

> **Note:**
>
> The **Rollback_On_Failure** check box is selected by default.

13. Click **Build** to run the Pipeline job.

You can use the WebLogic Server Administration Console to verify that the domain is updated with all the specified parameters. See Access the WebLogic Console.

Update a Domain on a Persistent Volume

The operator creates the domain on a persistent volume when the domain resource is first deployed. Typically, after the initial domain is created, you use tools such as the following for subsequent domain lifecycle operations:

- Fusion Middleware Control Console
- WebLogic Server Administration Console
- WebLogic Remote Console
- Product-specific WLST functions
- JDeveloper

Update the Base Image

Use the Jenkins job `update base image` to update the base image of a domain with a different base image for Oracle WebLogic Server for OKE. You can either use an existing base image or update the base image for your domain. If you created a new base image to apply patches to the JDK and WebLogic binaries, or to apply new set of OS packages, you can update the existing base image in your domain. You can use the `update base image` job to update the base image for both Model in Image and Domain on PV domains.

Patch a Domain

Learn how to patch a domain by using the tools provided with Oracle WebLogic Server for OKE. You can either run a Jenkins job to apply patches to a specific domain or automatically schedule patching for the required domains.

When you apply patches to a domain, the base image of a domain is automatically updated to a different base image.

You can apply patches also by creating a custom base image from WebLogic installer, JDK installer, and any set of WebLogic patches. See Create a Custom Base Image.

Topics:

- Apply a WebLogic Server Patch
- Automatic Patching

Apply a WebLogic Server Patch

Use the Jenkins job to apply patches to a WebLogic Server installation that you created with Oracle WebLogic Server for OKE.

You can use one of the following sources to specify the location of the WebLogic patch file:

- Object Storage - Uses the pre-authenticated URL on the Object Storage.
- Shared File System - Uses the path of the shared file storage.

  The NFS shared file system path is mounted on `/u01/shared` location on the administration host.

To apply the WebLogic patch file:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **apply patch**.
3. Click **Build with Parameters**.
4. From **Domain_Name**, select the domain to which you want to apply patches.
5. In **Domain_Restart_Timeout**, specify the time in minutes to wait for the WebLogic Domain Server pods to restart and be back in the running state. The default wait time is 40 minutes.
6. Select the source of the Weblogic patch file from the **WLS_Opatches** list.

   WLS_Opatches refers to any patch that may be required to patch WLS or Coherence product binaries (including ADR patches, OPSS, OWSM, OPatches).
7. For **Opatches_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system.

   For multiple patches, specify the location in separate lines.
8. Deselect the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).

   If you deselected this check box, you can rollback to the previous image later from the backup.

   > **Note:**
   >
   > The **Rollback_On_Failure** check box is selected by default.
9. *Optional:* Configure the Registry.

   If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

   a. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.

      If your tenancy is federated with Oracle Identity Cloud Service, use the format: `oracleidentitycloudservice/<username>`, else use `<username>`.

   b. In the **Registry_Authentication_Token** field, enter the OCID of the secret for the auth token generated for the registry user.

10. Click **Build** to run the Pipeline job.

Automatic Patching

Use the Jenkins job to automatically schedule patching for the required domains.

Following are the steps that you need to complete to utilize all the advantages of using automatic patching:

1. Subscribe or unsubscribe domains for automatic patching
2. Schedule automatic patching

Step 1: Subscribe or unsubscribe domains for automatic patching

To select a domain for automatic patching:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **automatic patching**.
3. Click **Build with Parameters**.
4. From **Domain_Names**, select the domains for automatic patching.
5. Subscribe or unsubscribe domains.
   - *Subscribe* domains: Select **Patch_Automatically**, if you want to run automatic patching on the selected domains.

     > **Note:**
     >
     > The **Patch_Automatically** option is selected by default.

   - *Unsubscribe* domains: Deselect the **Patch_Automatically** option, if you do *not* want to run automatic patching on the selected domains.
6. Run the `automatic patching` job.
   - If you *subscribe* a domain and click **Build**, then:
     - If applicable, the latest patches are applied on the selected domain.
     - The default schedule for automatic patching is enabled. By default, your local time zone is used and the automatic patching schedule is set to midnight of that time zone. To change the default schedule, see Schedule automatic patching.
     - A corresponding *Topic* is created in the *Notifications* application in Oracle Cloud Infrastructure.
       By default, the *Topic* name is: `<domain-name>-patching-topic`

       After the *Topic* is created. You can create subscriptions as required. See Creating a Subscription.
   - If you *unsubscribe* a domain and click **Build**, then:
     - The selected domains are removed from the automatic patching list.
     - The related *Topic* is deleted in the *Notifications* application in Oracle Cloud Infrastructure.

Step 2: Schedule automatic patching

To schedule automatic patching:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **automatic patching**.
3. Click **Configure** > **Build Triggers**.
4. By default, the **Build periodically** is selected with the required parameters.

   Update the parameters to set the preferred patching schedule.

   > **Note:**
   >
   > - Click the help icon next to the **Build periodically** field, to view information about the parameters you can use to schedule automatic patching.
   > - When you schedule automatic patching, by default, your local time zone is used and the automatic patching schedule is set to midnight of that time zone.

5. Click **Save**.

Troubleshoot a WebLogic Domain

Learn about the common issues when creating and managing a domain and then how to diagnose to solve them.

Topics

- Patching Job Fails
- Provisioning Fails at a Specific Stage
- Unable to View Jenkins UI Input Parameters
- Cleanup Resources Manually for a Failed Domain
- Terminate Domain Job Is Stuck at Finish_cleanup Phase
- Introspection Failed when Running Pipeline Jobs
- New Data Source Incorrectly Deployed
- WebLogic Server Pod Fails to Start
- Unable to Access the Console or the Application
- Load Balancer Creation Failed
- Jenkins Installation Fails
- Security Checkup Tool Warnings

Terminate a WebLogic Domain

Depending on the domain type, you can use the Jenkins jobs `terminate mii domain` or `terminate pv domain` to terminate a domain that you created with Oracle WebLogic Server for

OKE. The terminate domain job deletes the IDCS, external load balancer, WebLogic domain, and the Kubernetes namespaces that were created for the domain.

For Oracle WebLogic Server for OKE domains created in release 22.2.2 (May, 2022), you must update the Jenkins node label before you terminate a domain. See Terminate Domain Job Is Stuck at Finish_cleanup Phase.

To terminate a domain:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **terminate mii domain** or **terminate pv domain**.
3. Click **Build with Parameters**.
4. From **Domain_Name**, select the domain that you want to terminate.
5. Select **Delete_Node_Pool**, if you want to delete the node pool in the domain you selected to terminate.

   If you do no select **Delete_Node_Pool**, then the node pool in the selected domain is not deleted. And, the node pool is set as idle, which is listed under **Existing_Node_Pool** when you create a domain. See Create a Domain .

> **Note:**
>
> - The WebLogic node pool is not deleted if any existing domain pods are running on the node pool.
> - If the domain you terminate is subscribed for automatic patching, then, if applicable, the domain is automatically unsubscribed from the automatic patching list and the corresponding notification topics are deleted.

After the terminate job is complete, it displays a log that includes information about the actions that were performed by the terminate job.

Create a JRF Domain on a Persistent Volume Manually

You can create a JRF WebLogic domain using the tools provided with Oracle WebLogic Server for OKE on an existing Kubernetes Persistent Volume with the Domain on PV WebLogic domain home source type.

> **Note:**
>
> The steps provided in this section do not apply to WebLogic Kubernetes Operator version 4.0 and later.

For creating JRF domain on a persistent volume using the Jenkins pipeline job, see Provision a JRF Domain.

Topics:

- Rebase the Existing Base Image for the JRF Domain
- Apply the Patched Images to the Running JRF Domain
- Delete the Generated Domain Home

For more information, see Domain Home on a PV.

About Domain on Persistent Volume

The Domain on Persistent Volume (PV) WebLogic domain home source type lets you locate a WebLogic domain on a persistent volume with the PV residing either in a Network File System (NFS) or in other Kubernetes volume types. With Domain on a PV, you should populate the PV with a domain home before deploying the domain resource.

Domain on PV is supported on two types of domains:

- A standard Oracle WebLogic Server domain.
- An Oracle Fusion Middleware Infrastructure, Java Required Files (JRF) domain.

For more information about this domain home source type, see Domain on Persistent Volume (PV) .

Prerequisites to Create a Domain on a Persistent Volume

Before you create a JRF domain on a persistent volume with Oracle WebLogic Server for OKE, you must complete the following prerequisite tasks.

- Ensure that the WebLogic Kubernetes Operator is in a running state.
- Obtain the Base Image to Create the JRF Domain
- Create a Kubernetes Namespace for the JRF Domain
- Create the Kubernetes Secrets for the JRF Domain
- Create the Persistent Volume and the Persistent Volume Claim

Obtain the Base Image to Create the JRF Domain

To create a JRF domain on PV, you will require a base image or a container image with WebLogic Server or Fusion Middleware infrastructure to run the WebLogic domains in Kubernetes. You can obtain the base image using one of the following options:

- From Oracle Container Registry (OCR). For example: `container-registry.oracle.com/middleware/weblogic:12.2.1.4`.
- From Oracle WebLogic Server for OKE administration host. For example, `<ocir_tenancy_name>/<namespace>/<domain_name>/wls-base-image/12214:12.2.1.4.230117-230117`.
- By creating a new base image using the Oracle WebLogic Server for OKE Jenkins Pipeline job `create base image`. See Pipeline Jobs.

Create a Kubernetes Namespace for the JRF Domain

> **Note:**
>
> Oracle recommends that you create a namespace and all the JRF domain resources in the format `${domain_name}-<resource_name>`. For example: `jrf00001-ns` (namespace), `jrf00001-property-configmap` (configmap), and so on.

You should create a Kubernetes namespace for the JRF domain resource if you do not want to use the default namespace. Let the WebLogic Kubernetes Operator manage this namespace.

To create a namespace, run the following command:

```
kubectl create namespace <domain_name>-ns
```

For example:

```
kubectl create namespace jrf00001-ns
```

To allow the Operator to manage the namespace:

```
kubectl label namespace <domain_name>-ns weblogic-operator=enabled
```

Create the Kubernetes Secrets for the JRF Domain

The WebLogic Kubernetes Operator expects the secret key names to be `username` and `password` of the administrative account in the same Kubernetes namespace as the domain resource.

You should create the following secret key names:

- A Kubernetes docker-registry secret containing the registry credentials.

```
$ kubectl create secret docker-registry ocirsecret1 \
  -n <domain_name>-ns \
  --docker-server=container-registry.oracle.com \
  --docker-username=<YOUR_USERNAME> \
  --docker-password=<YOUR_PASSWORD> \
```

- The WebLogic domain administrator credentials.

```
$ kubectl create secret generic <domain_name>-weblogic-credentials \
  --from-literal=username=<ADMIN_USERNAME> --from-
literal=password=<ADMIN_PASSWORD> \
  -n <domain_name>-ns
```

- The domain runtime encryption secret.

```
$ kubectl -n <domain_name>-ns create secret generic \
  <domain_name>-runtime-encryption-secret \
   --from-literal=password=<my_runtime_password>
```

- The secret to access the Repository Creation Utility (RCU):
  If you are using the Oracle Autonomous Database:

```
$ kubectl create secret generic  <domain_name>-rcu-access
      --from-literal=rcu_db_name=<RCU_DB_NAME>
      --from-literal=rcu_prefix=<RCU_PREFIX>
      --from-literal=rcu_schema_password=<RCU_SCHEMA_PASSWORD>
      --from-literal=rcu_wallet_password=<RCU_WALLET_PASSWORD>
      -n <domain_name>-ns
```

  If you are using a database deployed on Oracle Cloud Infrastructure (OCI):

```
$ kubectl create secret generic <domain_name>-rcu-access
      --from-literal=rcu_db_user=<RCU_DB_USER>
      --from-literal=rcu_prefix=<RCU_PREFIX>
      --from-literal=rcu_schema_password=<RCU_SCHEMA_PASSWORD>
      --from-literal=rcu_admin_password=<RCU_ADMIN_PASSWORD>
      --from-literal=rcu_db_conn_string=<RCU_CONN_STRING>
      -n <domain_name>-ns
```

- The wallet password secret for Oracle Platform Security Service (OPSS).

```
$ kubectl create secret generic <domain_name>-opss-wallet-password-secret
     --from-literal=walletPassword=<WALLET_PASSWORD>
     -n <domain_name>-ns
```

Create the Persistent Volume and the Persistent Volume Claim

The Kubernetes Persistent Volume (PV) and Persistent Volume Claim (PVC) are used as storage locations for the WebLogic domain homes and log files.

PVs and PVCs are described in YAML files. For each PV, you should create one PV YAML file and one PVC YAML file. In the following example, you will find two YAML file samples, one for the volume and one for the claim.

pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: <domain_name>-oke-pv
 namespace: <domain_name>-ns
spec:
 storageClassName: <service_name>-oke-fss
 capacity:
  storage: 100Gi
 accessModes:
  - ReadWriteMany
 mountOptions:
```

```
  - nosuid
 nfs:
  server: <MOUNT_IP>
  path: "/<service_name>"
  readOnly: false
```

Here:

- `<MOUNT_IP>` is the IP of NFS server that is created as part of the stack.
- `<service_name>` is the resource_prefix of the stack.

pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: <domain_name>-oke-pvc
 namespace: <domain_name>-ns
spec:
 storageClassName: <service_name>-oke-fss
 accessModes:
 - ReadWriteMany
  resources:
  requests:
  storage: 100Gi
  volumeName: <domain_name>-oke-pv
```

Use the `pv.yaml` file to create a PV by running the following command:

```
$ kubectl apply -f pv.yaml
```

Use the `pvc.yaml` file to create a PVC by running the following command:

```
kubectl apply -f pvc.yaml
```

Use the following commands to verify:

```
kubectl get pvc
```

```
kubectl get pv
```

To view the list of storage classes in your cluster, run the following command:

```
$ kubectl get storageclass
```

Create the JRF Domain

After you complete the prerequisite tasks, you can use the sample scripts provided by the WebLogic Kubernetes Operator (WKO) to create the JRF domain on an existing Kubernetes Persistent Volume (PV) and Persistent Volume Claim (PVC).

Topics:

Download the Scripts

Download and copy the sample scripts provided by WebLogic Kubernetes Operator to create the JRF domain on PV. The scripts are available at https://github.com/oracle/weblogic-kubernetes-operator/tree/release/3.4/kubernetes/samples/scripts/create-weblogic-domain/domain-home-on-pv.

Use the `git clone` command to copy the scripts into a newly created directory on your machine, as shown in the following example:

```
git clone git@github.com:oracle/weblogic-kubernetes-operator.git
```

You will find all the scripts ready for your use in this newly created directory `weblogic-kubernetes-operator`.

Create the RCU Schema

One of the sample scripts provided by the WebLogic Kubernetes Operator is the `create-rcu-schema.sh` script that creates the RCU (Repository Creation Utility) schema for the JRF domain on PV.

Use the following command to run this script on the administration VM:

```
$ scripts/create-rcu-schema/create-rcu-schema.sh -s <rcuSchemaPrefix> -d
<rcuDatabaseURL> -n <domain_name>-ns -c <domain_name>-rcu-access
```

Use the Scripts to Create the JRF Domain

To create the JRF domain, use the sample script `create-domain.sh` provided by WebLogic Kubernetes Operator. This script takes the `create-domain-inputs.yaml` file as the input.

Here is a sample of the `create-domain-inputs.yaml` file for your reference:

```
# Copyright (c) 2023, 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-domain-inputs-v1

# Port number for admin server
adminPort: 7001

# Name of the Admin Server
adminServerName: <domain_name>-adminserver
```

```
# Unique ID identifying a domain.
# This ID must not contain an underscore ("_"), and must be lowercase and
unique across all domains in a Kubernetes cluster.
domainUID: <domain_name>

# Home of the WebLogic domain
# If not specified, the value is derived from the domainUID as /shared/
domains/<domainUID>
domainHome: /u01/shared/weblogic-domains/<domain_name>

# Determines which OracleFMWInfrastructure Servers the operator will start up
# Legal values are "Never", "IfNeeded", or "AdminOnly"
serverStartPolicy: IfNeeded

# Cluster name
clusterName: cluster-1

# Number of managed servers to generate for the domain
configuredManagedServerCount: 5

# Number of managed servers to initially start for the domain
initialManagedServerReplicas: 2

# Base string used to generate managed server names
managedServerNameBase: <domain_name>-managedserver

# Port number for each managed server
managedServerPort: 8001

# WebLogic Server image.
# The operator requires WebLogic Server 12.2.1.3.0 with patch 29135930
applied or 12.2.1.4.0.
# The existing WebLogic Server image, `container-registry.oracle.com/
middleware/fmw-infrastructure:12.2.1.3`,
# has all the necessary patches applied.
#
# **NOTE**:
# This sample uses General Availability (GA) images. GA images are suitable
for demonstration and
# development purposes only where the environments are not available from the
public Internet;
# they are not acceptable for production use. In production, you should
always use CPU (patched)
# images from OCR or create your images using the WebLogic Image Tool.
# Please refer to the `OCR` and `Manage FMW infrastructure domains` pages in
the WebLogic
# Kubernetes Operator documentation for details.
image: container-registry.oracle.com/middleware/fmw-infrastructure:12.2.1.4

# Image pull policy
# Legal values are "IfNotPresent", "Always", or "Never"
imagePullPolicy: IfNotPresent

# Name of the Kubernetes secret to access the container registry to pull the
WebLogic Server image
```

```
# The presence of the secret will be validated when this parameter is enabled.
#imagePullSecretName:

# Boolean indicating if production mode is enabled for the domain
productionModeEnabled: true

# Name of the Kubernetes secret for the Admin Server's username and password
# The name must be lowercase.
# If not specified, the value is derived from the domainUID as <domainUID>-
weblogic-credentials
weblogicCredentialsSecretName: <domain_name>-weblogic-credentials

# Whether to include server .out to the pod's stdout.
# The default is true.
includeServerOutInPodLog: true

# The in-pod location for domain log, server logs, server out, Node Manager
log, introspector out, and
# HTTP access log files. If not specified, the value is derived from the
domainUID as
# /shared/logs/<domainUID>.
logHome: /u01/shared/logs/<domain_name>

# Set this value to 'false' to have HTTP access log files written to the
directory
# configured in the WebLogic domain home.
# The default is 'true', which means HTTP access log files will be written to
# the logHome directory.
httpAccessLogInLogHome: true

# Port for the T3Channel of the NetworkAccessPoint
t3ChannelPort: 30012

# Public address for T3Channel of the NetworkAccessPoint.  This value should
be set to the
# kubernetes server address, which you can get by running "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: false

# NodePort to expose for the admin server
adminNodePort: 30701

# Boolean to indicate if the adminNodePort will be exposed
exposeAdminNodePort: false

# Name of the domain namespace
namespace: <domain_name>-ns

#Java Option for WebLogic Server
javaOptions: -Dweblogic.StdoutDebugEnabled=false
```

```
# Name of the persistent volume claim
# If not specified, the value is derived from the domainUID as <domainUID>-
weblogic-sample-pvc
persistentVolumeClaimName: <domain_name>-oke-pvc

# Mount path of the domain persistent volume.
domainPVMountPath: /u01/shared

# Mount path where the create domain scripts are located inside a pod
#
# The `create-domain.sh` script creates a Kubernetes job to run the script
(specified in the
# `createDomainScriptName` property) in a Kubernetes pod to create a WebLogic
home. Files
# in the `createDomainFilesDir` directory are mounted to this location in the
pod, so that
# a Kubernetes pod can use the scripts and supporting files to create a
domain home.
createDomainScriptsMountPath: /u01/weblogic

# Script that the create domain script uses to create a WebLogic domain
#
# The `create-domain.sh` script creates a Kubernetes job to run this script
to create a
# domain home. The script is located in the in-pod directory that is
specified in the
# `createDomainScriptsMountPath` property.
#
# If you need to provide your own scripts to create the domain home, instead
of using the
# built-it scripts, you must use this property to set the name of the script
that you want
# the create domain job to run.
createDomainScriptName: create-domain-job.sh

# Directory on the host machine to locate all the files to create a WebLogic
domain
# It contains the script that is specified in the `createDomainScriptName`
property.
#
# By default, this directory is set to the relative path `wlst`, and the
create script will
# use the built-in WLST offline scripts in the `wlst` directory to create the
WebLogic domain.
# It can also be set to the relative path `wdt`, and then the built-in WDT
scripts will be
# used instead.
#
# An absolute path is also supported to point to an arbitrary directory in
the file system.
#
# The built-in scripts can be replaced by the user-provided scripts or model
files as long
# as those files are in the specified directory. Files in this directory are
put into a
```

```
# Kubernetes config map, which in turn is mounted to the
`createDomainScriptsMountPath`,
# so that the Kubernetes pod can use the scripts and supporting files to
create a domain home.
createDomainFilesDir: wlst

# Resource request for each server pod (Memory and CPU). This is minimum
amount of compute
# resources required for each server pod. Edit value(s) below as per pod
sizing requirements.
# These are optional.
# Please refer to the kubernetes documentation on Managing Compute
# Resources for Containers for details.
serverPodMemoryRequest: "1280Mi"
serverPodCpuRequest: "500m"

# Uncomment and edit value(s) below to specify the maximum amount of compute
resources allowed
# for each server pod.
# These are optional.
# Please refer to the kubernetes documentation on Managing Compute
# Resources for Containers for details.
# serverPodMemoryLimit: "2Gi"
# serverPodCpuLimit: "1000m"

#
# RCU configuration details
#
# The schema prefix to use in the database, for example `SOA1`.  You may want
to make this
# the same as the domainUID in order to simplify matching domains to their
RCU schemas.
rcuSchemaPrefix: <rcuSchemaPrefix>

# The database URL
rcuDatabaseURL: <rcuDatabaseURL>

# The kubernetes secret containing the database credentials
rcuCredentialsSecret: <domain_name>-rcu-access

# FMW Infrastructure Domain Type. Legal values are JRF or RestrictedJRF
fmwDomainType: JRF
```

> **Note:**
>
> For `rcuSchemaPrefix`, use the same value that you used to run the `create-rcu-schema.sh` script for creating the RCU schema.

You should make a copy of the `create-domain-inputs.yaml` file, and then run the `create-domain.sh` by pointing it to this file and an output directory, as shown in the following command:

```
scripts/create-fmw-infrastructure-domain/domain-home-on-pv/create-domain.sh -
i create-domain-inputs.yaml -o output
```

The script will complete the following tasks:

- Create a directory for the generated Kubernetes YAML files for this domain if it does not already exist. The path name is `/<path to output-directory>/weblogic-domains/<domainUID>`. If the directory already exists, ensure that you remove its contents before using this script.
- Create a Kubernetes job that will start up a utility WebLogic Server container and run offline WLST scripts or WebLogic Deploy Tooling (WDT) scripts, to create the domain on the shared storage.

After the Kubernetes job is complete, use the `domain.yaml` file to create the Kubernetes resource using the `kubectl apply` command, as shown in the following example:

```
kubectl apply -f <outputdir>/weblogic-domains/<domain_name>/domain.yaml
```

Verify the Domain

To confirm that the domain has been created, use the following command:

```
$ kubectl describe domain <domain_name> -n <domain_name>-ns
```

To see the services associated with the domain, use the following command:

```
$ kubectl get services -n <domain_name>-ns
```

To verify the domain folder on the FSS mount, use the following command:

```
$ ls /u01/shared/weblogic-domains/<domain_name>/
```

To view the content of the JDBC folder, use the following command:

```
$ ls /u01/shared/weblogic-domains/<domain_name>/config/jdbc/
```

Rebase the Existing Base Image for the JRF Domain

You can use the Jenkins job `rebase domain image` to update a base image of a JRF domain with a different base image for Oracle WebLogic Server for OKE. This target base image can be an existing base image or a new base image created using the Jenkins job `create base image`. The `rebase domain image` job removes the domain definition from the existing domain and rebases the existing domain with the selected target base image.

Apply the Patched Images to the Running JRF Domain

For Domain in PV domains, the container image contains only the JDK and the WebLogic Server binaries, and its domain home that gets created is located in a persistent volume. For this domain home source type, you can create your own patched images using the following steps or you can obtain the patched images from Oracle Container Registry. See Obtain Images From the Oracle Container Registry.

1. Add the patch to the WebLogic Image Tool (WIT) cache.

   ```
   $ /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
   patchId=<patch_id> --path=<path>
   ```

2. Check if the patch is added correctly to the WIT cache.

   ```
   $ /u01/shared/tools/imagetool/bin/imagetool.sh cache listItems
   ```

3. Create the container image with all the patches.

   ```
   /u01/shared/tools/imagetool/bin/imagetool.sh update --fromImage
   <primordial_domain_image> --tag <tag_for_the_final_build_image> --patches
   <patch_ids>
   ```

4. Update the domain's configmap with the new base image.

   ```
   kubectl edit configmap <configmap_name> -n <domain_ns>
   ```

   This command will open the Vim editor where you can replace the value of `primordial_domain_image` with the new image.

To apply the patched image:

1. Edit the domain resource image reference with the new image name in `domain.yaml`. For example:

   ```
   output/weblogic-domains/<domin_name>/domain.yaml
   ```

2. Execute the `kubectl apply` command to create the domain resource YAML file.

   ```
   kubectl apply -n <domain_name>_ns -f domain.yaml
   ```

   The Operator automatically performs a rolling restart of the WebLogic domain to update the Oracle Home of the servers.

Delete the Generated Domain Home

Sometimes in production, but most likely in the testing environment, you may want to remove the domain home that is generated using the `create-domain.sh` script. This script creates the `delete-domain-job.yaml` file in the `/<path to output-directory>/weblogic-domains/<domainUID>` directory. For example:

```
outputdir/weblogic-domains/domainmb2/delete-domain-job.yaml
```

You can delete the domain by running the generated `delete-domain-job.yaml` file in the `/<path to output-directory>/weblogic-domains/<domainUID>` directory, as shown in the following example:

```
$ kubectl create -f delete-domain-job.yaml
```

4

Manage WebLogic Domains in Verrazzano

Learn how to manage a WebLogic domain in Verrazzano after creating it with Oracle WebLogic Server for OKE.

Topics:

About Managing a WebLogic Domain

Learn how to create and deploy custom domain images, and access the Jenkins and Verrazzano consoles.

In general, you configure, manage, and maintain an Oracle WebLogic Server for OKE domain just like an on-premise domain. For example, to deploy an application, see Roadmap for Deploying Applications in WebLogic Server (12.2.1.4)

About WebLogic Deploy Tooling

Learn about the WebLogic Deploying Tool (WDT) to manage a WebLogic domain in Oracle WebLogic Server for OKE.

WebLogic Deploy Tooling automates the domain creation and application deployment tasks using the metadata model that describes the domain and applications (with their dependent resources). WebLogic Deploy Tooling also provides single-purpose tools that perform domain lifecycle operations based on the content of the model.

WebLogic Deploy Tooling is used to create the container image and represent the WebLogic configuration. It makes use of a model Yaml file, an archive zip, and a properties file.

When you have multiple WDT files, the WebLogic Deploy Tooling merges them together and create a single WebLogic domain configuration represented by a single model Yaml file.

The model Yaml file that represents your WebLogic configuration is located in `/u01/shared/` location.

Following is an example of a model Yaml file that has the name of the application as `my-application` and target cluster as `my-WebLogic-domain-cluster`:

```
appDeployments:
    Application:
      my-application :
        SourcePath: wlsdeploy/applications/my-application
        Target: my-WebLogic-domain-cluster
        ModuleType: ear
```

Following is an example of a model Yaml file that has only WebLogic credentials:

```
domainInfo:
    AdminUserName: '@@SECRET:__weblogic-credentials__:username@@'
    AdminPassword: '@@SECRET:__weblogic-credentials__:password@@'
```

Following is an example of a model Yaml file that has RCU database connection information:

```
domainInfo:
    RCUDbInfo:
      rcu_prefix: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-access:rcu_prefix@@'
      rcu_schema_password: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-
access:rcu_schema_password@@'
      rcu_db_conn_string: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-
access:rcu_db_conn_string@@'
      rcu_db_user: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-access:rcu_db_user@@'
      rcu_admin_password: '@@SECRET:@@ENV:DOMAIN_UID@@-rcu-
access:rcu_admin_password@@'
```

To know more, see Metadata Model in *WebLogic Deploy Tooling*.

Project Components

Learn about the different tools and files that you can use to build Jenkins projects for your domain.

Oracle WebLogic Server for OKE creates an Administration load balancer (private) for your domain, which you use to access the Jenkins console and the Verrazzano consoles running on the Kubernetes cluster. Oracle WebLogic Server for OKE also creates an Application load balancer for your domain that you can use to access the applications. An NGINX ingress controller is used to route traffic from the private load balancer to Jenkins.

All Jenkins files are found on the shared file system.

- `/u01/shared/var/jenkins_home` - The Jenkins controller server configuration
- `/u01/shared/scripts/pipeline` - The resources used to run the sample Jenkins job, including scripts and metadata files

To access or modify these files, use the administration compute instance for your domain. This compute instance also includes the following software:

- `kubectl` - Deploy and manage pods in the Kubernetes cluster for this domain.
- `docker` - Download, modify, and update container images in Oracle Cloud Infrastructure Registry. Use the `login` command to connect to the registry.

- `oci` - View, create and update resources in Oracle Cloud Infrastructure.

About Verrazzano WebLogic Components and Application Configuration

Verrazzano supports application definition using Open Application Model (OAM). Verrazzano applications are composed of OAM components and OAM application configurations.

See Verrazzano.

A Verrazzano OAM Component is a Kubernetes Custom Resource that describes an application's general composition and environment requirements. A Verrazzano application configuration is a Kubernetes Custom Resource which provides environment specific customizations.

A containerized WebLogic workload is modeled as a component.

An application configuration assembles a set of component instances, their traits, and the application scopes in which they are placed, combined with configuration parameters and metadata.

You can deploy one or more yaml files that has Verrazzano components and applications with the Verrazzano application generated by Oracle WebLogic Server for OKE.

Following is an example of the structure of a component:

```
apiVersion: core.oam.dev/v1alpha2
kind: Component
metadata:
 name: todo-domain
spec:
 workload:
   apiVersion: oam.verrazzano.io/v1alpha1
   kind: WebLogicWorkload
   metadata:
     name: todo-workload
     labels:
       app: todo
   spec:
     deploymentTemplate:
       metadata:
         name: todo-deployment
       podSpec:
         containers:
           - name: todo-container
             image: "myrepo/myuser/hello-world"
             ports:
               - containerPort: port-number
                 name: http
```

Following is an example of the structure of an application configuration:

```
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
```

```
  name: todo-appconf
  namespace: todo-list
  annotations:
    version: v1.0.0
    description: "ToDo List example application"
spec:
  components:
    - componentName: todo-domain
      traits:
        - trait:
            apiVersion: oam.verrazzano.io/v1alpha1
            kind: MetricsTrait
            spec:
              scraper: verrazzano-system/vmi-system-prometheus-0
        - trait:
            apiVersion: oam.verrazzano.io/v1alpha1
            kind: IngressTrait
            spec:
              rules:
                - paths:
                    - path: "/todo"
                      pathType: Prefix
```

To know more, see Application deployment in *Verrazzano* documentation.

Access Resources

Learn how to access the administration console, Jenkins, and the Verrazzano consoles using the tools provided with Oracle WebLogic Server for OKE.

Topics:

Access the Administration Instance

Access the administration compute instance for a Oracle WebLogic Server for OKE.

From the administration compute instance, you can access the shared file system at `/u01/shared`. It also includes the following software:

- `kubectl` - Deploy and manage pods in the Kubernetes cluster for this stack.
- `docker` - Download, modify and update container images in Oracle Cloud Infrastructure Registry. Use the `login` command to connect to the registry.
- `oci` - View, create and update resources in Oracle Cloud Infrastructure.

This compute instance is on a private subnet and cannot be directly accessed from the public Internet. You can use the bastion instance, which is on a public subnet, and the proxy option of a secure shell (SSH) utility.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.
5. Click **Jobs**.
6. In the **Jobs** table, click the Apply job for the stack.
7. Click **Application Information**.
8. Identify and make a note of the following IP addresses:
   - `Bastion Instance Public IP` - The public IP address of the bastion compute instance
   - `Admin Instance Private IP` - The private IP address of the administration compute instance
9. From your computer, create an SSH connection to the administration instance's IP address, and also specify the bastion instance's IP address as a proxy.

   Connect as the `opc` user.

   Provide the path to the private key that corresponds to the public key that you specified when you created the stack.

   The SSH command format is:

   ```
   ssh -i <path_to_private_key> -o ProxyCommand="ssh -W %h:%p -i
   <path_to_private_key> opc@<bastion_public_ip>" opc@<admin_ip>
   ```

   For example:

   ```
   ssh -i ~/.ssh/mykey.openssh -o ProxyCommand="ssh -W %h:%p -i ~/.ssh/
   mykey.openssh opc@203.0.113.13" opc@198.51.100.1
   ```

   On a Windows platform, you can use Windows PowerShell to run the SSH command.
10. If prompted, enter the passphrase for the private key.

Access the Jenkins Console

Access the Jenkins build engine for a stack in which you configured Verrazzano with Oracle WebLogic Server for OKE.

Jenkins runs as a pod in the Kubernetes cluster and is hosted on the Verrazzano's private load balancer. This load balancer cannot be directly accessed from the public Internet. Jenkins is and can be accessed through socks proxy securely.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.

5. Click **Jobs**.
6. Click the Apply job for the stack.
7. Click **Outputs**.
8. Identify the public IP address of the bastion compute instance, `bastion_instance_public_ip`.
9. Click **Logs**.
10. Search for the attribute `jenkins_console_url`. Copy the URL.

    The URL format is:

    ```
    http://
    jenkins.<resource_name_prefix>.<internal_lb_ip>.<wildcard_DNS_provider>/
    jenkins
    ```

    If you specify the environment name during stack creation with Verrazzano, the URL format is:

    ```
    http://jenkins.<environment_name>.<internal_lb_ip>.<wildcard_DNS_provider>/
    jenkins
    ```

    Examples:

    - `http://jenkins.abc.<internal_lb_ip>.nip.io/jenkins`

      where, *abc* is the specified environment name for the Verrazzano installation and *nip.io* is the default wildcard DNS service.

    - `http://jenkins.abc.<internal_lb_ip>.sslip.io/jenkins`

      where, *abc* is the specified environment name for the Verrazzano installation and *sslip.io* is the wildcard DNS service that is configured during Verrazzano installation.

    To know more about the DNS services, see Customize DNS.

    > **Note:**
    >
    > If you provision a domain without a bastion instance, you must obtain the internal load balancer IP address to use in the Jenkins URL.
    >
    > To get the internal load balancer IP address, access the administration instance for your stack (see Access the Administration Instance) and run the following command:
    >
    > ```
    > kubectl get svc -A
    > ```
    >
    > The private load balancer is listed with the namespace `wlsoke-ingress-nginx` and name *okename*`-internal`.

11. From your computer, open an SSH tunnel to an unused port on the bastion compute instance as the `opc` user.

For example, you can use port `1088` for SOCKS proxy.

Specify the `-D` option to use dynamic port forwarding. Provide the path to the private key that corresponds to the public key that you specified when you created the domain.

The SSH command format is:

```
ssh -D <port_for_socks_proxy> -fCqN -i <path_to_private_key>
opc@<bastion_public_ip>
```

For example:

```
ssh -D 1088 -fCqN -i ~/.ssh/mykey.openssh opc@198.51.100.1
```

On a Windows platform, you can use Windows PowerShell to run the SSH command.

12. In your browser settings, set up the SOCKS (version 5) proxy configuration. Specify your local computer and the same SOCKS port that you used in your SSH command.
13. Browse to the Jenkins console URL.
14. If this is the first time using the Jenkins console, you are prompted to create a new admin user.

Access the Verrazzano Consoles

Access the Verrazzano consoles for a stack in which you configured Verrazzano with Oracle WebLogic Server for OKE.

Verrazzano installs several consoles. To access these consoles:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.
5. Click **Jobs**.
6. Click the Apply job for the stack.
7. Click **Outputs**.
8. Identify the public IP address of the bastion compute instance, `bastion_instance_public_ip`.
9. Click **Logs**.
10. Search for the attribute `consoleUrl`.

    The following are the console URL formats with the default wildcard DNS provider, *nip.io*:

> **Note:**
>
> If you do not specify the environment name during stack creation with Verrazzano, the resource name prefix is used as the environment name.

- Verrazzano - `https://verrazzano.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- ElasticSearch - `https://elasticsearch.vmi.system.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- Grafana - `https://grafana.vmi.system.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- KeyCloak - `https://keycloak.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- Kiali - `https://kiali.vmi.system.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- Kibana - `https://kibana.vmi.system.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- Prometheus - `https://prometheus.vmi.system.<resource_name_prefix>.<internal_lb_ip>.nip.io`
- Rancher - `https://rancher.<resource_name_prefix>.<internal_lb_ip>.nip.io`

where, `internal_lb_ip` is the internal load balancer IP address.

An example of Verrazzano console URL format with *abc* environment name and default wildcard DNS provider, *nip.io*:

```
https://verrazzano.<environment_name>.<internal_lb_ip>.nip.io
```

An example of ElasticSearch console URL format with *abc* environment name and *sslip.io* wildcard DNS provider configured during Verrazzano installation.

```
https://elasticsearch.vmi.system.abc.<internal_lb_ip>.sslip.io
```

> **Note:**
>
> If you provision a domain without a bastion instance, you must obtain the internal load balancer IP address to use in the Verrazzano console URLs.
>
> To get the internal load balancer IP address, access the administration instance for your stack (see Access the Administration Instance) and run the following command:
>
> ```
> kubectl get svc -A
> ```
>
> The private load balancer is listed with the namespace `wlsoke-ingress-nginx` and name *okename*`-internal`.

11. From your computer, open an SSH tunnel to an unused port on the bastion compute instance as the `opc` user.

    For example, you can use port `1088` for SOCKS proxy.

    Specify the `-D` option to use dynamic port forwarding. Provide the path to the private key that corresponds to the public key that you specified when you created the domain.

The SSH command format is:

```
ssh -D <port_for_socks_proxy> -fCqN -i <path_to_private_key>
opc@<bastion_public_ip>
```

For example:

```
ssh -D 1088 -fCqN -i ~/.ssh/mykey.openssh opc@198.51.100.1
```

On a Windows platform, you can use Windows PowerShell to run the SSH command.

12. In your browser settings, set up the SOCKS (version 5) proxy configuration. Specify your local computer and the same SOCKS port that you used in your SSH command.
13. Browse to the console URLs.

    To know the preconfigured users and to get the password for the consoles, see Get Console Credentials.

Access the WebLogic Console

Access the WebLogic Server Administration Console for a domain that you created with Oracle WebLogic Server for OKE with Verrazzano.

> **Note:**
>
> - Before you access the WebLogic console, you must have created a WebLogic domain. See Create a WebLogic Domain.
> - Do not use the WebLogic console to make any configuration changes. All configuration changes should be should done through jobs, this ensure that the changes are persistent.
> - Security check warnings are displayed at the top of the console. See About the Security Checkup Tool for the warnings and how to handle them.

You must have installed OCI CLI and `kubectl` in your local computer. If you do not have them installed, see Install CLI and Install kubectl.

> **Note:**
>
> The following instructions are applicable only if your local computer has access to the Kubernetes API endpoint.

1. From your computer, create an SSH connection to the administration instance's IP address, and also specify the bastion instance's IP address as a proxy.

   Connect as the `opc` user.

   Provide the path to the private key that corresponds to the public key that you specified when you created the stack.

The SSH command format is:

```
ssh -i <path_to_private_key> -o ProxyCommand="ssh -W %h:%p -i
<path_to_private_key> opc@<bastion_public_ip>" opc@<admin_ip>
```

For example:

```
ssh -i ~/.ssh/mykey.openssh -o ProxyCommand="ssh -W %h:%p -i ~/.ssh/
mykey.openssh opc@203.0.113.13" opc@198.51.100.1
```

On a Windows platform, you can use Windows PowerShell to run the SSH command.

2. Navigate to the `.kube/config` folder.
3. Open the config file and copy the contents of the config file.
4. In your local computer, under `$HOME/.kube` directory, create a `config` folder, and copy the contents of the config file.

> **Note:**
>
> If you do not have the directory to contain the kubeconfig file, then create the directory using the following command:
>
> ```
> $ mkdir -p $HOME/.kube
> ```

5. Sign in to the Oracle Cloud Infrastructure Console.
6. Click the navigation menu  , select **Developer Services**. Under **Containers**, click **Kubernetes Clusters (OKE)**.
7. Select the **Compartment** containing the cluster.
8. On the Cluster List page, click the name of the cluster you want to access using kubectl. The Cluster page shows details of the cluster.
9. Click **Access Cluster**.
10. On the **Access Your Cluster** dialog box, click **Local Access**.
11. Run the Oracle Cloud Infrastructure CLI command to set up the `kubeconfig` file and save it in a location accessible to kubectl.

    For example, on Linux, enter the following command (or copy and paste it from the Access Your Cluster dialog box) in a local terminal window:

    ```
    oci ce cluster create-kubeconfig --cluster-id <OCID of the current
    cluster> --file /.kube/config --region <region_name> --token-version 2.0.0
    --kube-endpoint PRIVATE_ENDPOINT
    ```

12. Set the value of the KUBECONFIG environment variable to point to the name and location of the kubeconfig file.

    ```
    export KUBECONFIG=$HOME/.kube/config
    ```

13. Set up port forwarding to the administration server that you want to connect.

    ```
    $ kubectl port-forward pods/<admin_server_pod_name> 7001:7001 -n
    <namespace>
    ```

    Example:

    ```
    $ kubectl port-forward pods/oci-domain-uid-ocidomain-adminserver 7001:7001
    -n <namespace>
    ```

14. Browse to the WebLogic Server Administration Console URL.

    ```
    http://localhost:7001/<domain_name>/console
    ```

15. Sign in using the administrator credentials for your domain.

Create a WebLogic Domain

Create a WebLogic domain for Oracle WebLogic Server for OKE stack with Verrazzano by using the tools provided with Oracle WebLogic Server for OKE.

Topics:

About Creating a Domain

Learn about the options you have when creating a domain.

- Domain Type

  A Non-JRF domain does not require an existing database.

Prerequisites to Create a Domain

Complete the prerequisites before you create a domain.

Required Tasks

Create Policies for the Dynamic Group

Create policies in Oracle Cloud Infrastructure so that the compute instances in Oracle WebLogic Server for OKE with Verrazzano can access your encryption key.

When you create a stack with Verrazzano, depending on your custom setup, compute instances in Oracle WebLogic Server for OKE need to access specific components in Oracle Cloud Infrastructure.

If you are creating a stack with Verrazzano that uses custom OCI DNS, then the following sample policy grants access to the OCI DNS:

```
Allow dynamic-group MyInstancesPrincipalGroup to inspect dns-zones in
compartment id DNS_Zone_CompartmentID
Allow dynamic-group MyInstancesPrincipalGroup to use dns-zones in compartment
id DNS_Zone_CompartmentID where target.dns-zone.id='<dns_zone_ocid>'
Allow dynamic-group MyInstancesPrincipalGroup to manage dns-records in
compartment id DNS_Zone_CompartmentID where target.dns-
zone.id='<dns_zone_ocid>'
```

If you are creating a stack with Verrazzano that uses OCI private DNS and a new VCN, then the following sample policy grants access to the required resources:

```
Allow dynamic-group MyInstancesPrincipalGroup to inspect vcns in compartment
id NetworkCompartmentID where target.vcn.id='<vcn_id>'
Allow dynamic-group MyInstancesPrincipalGroup to use dns-resolvers in
compartment id NetworkCompartmentID where target.dns-resolver.display-
name='<label_prefix-vcn>'
Allow dynamic-group MyInstancesPrincipalGroup to inspect dns-views in
compartment id DNS_Zone_CompartmentID
```

If you are creating a stack with Verrazzano that uses custom CA, then the following sample policy grants access to the Custom CA certificates:

```
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
tenancy where target.secret.id = '<custom_ca_cert_secret_ocid>'
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
tenancy where target.secret.id = '<custom_ca_signing_key_secret_ocid>'
```

If you are creating a stack with Verrazzano that uses OCIR repository, then the following sample policy grants access to the OCIR repository:

```
Allow dynamic-group MyInstancesPrincipalGroup to manage repos in compartment
id OCI_Identity_CompartmentID
```

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Writing Policies for Dynamic Groups

Validate Existing Network Setup

You can use helper scripts from the Oracle Cloud Infrastructure Cloud shell to certify the existing network setup (existing VCN and existing WebLogic Server subnet) in Oracle WebLogic Server for OKE. See Using Cloud Shell in Oracle Cloud Infrastructure documentation.

The helper scripts perform the following validations and functions:

- Validates if the service gateway or the NAT gateway is created for the administration instance private subnet and the worker nodes private subnets.
- Validates if internet gateway is created for public bastion, file shared system and load balancer subnets.
- Checks if port 22 in WebLogic Server Subnet is open for access to the CIDR of the bastion instance subnet or bastion host IP.
- Checks if the private subnet for the Oracle WebLogic Server compute instances using the service gateway route rule has **All *<Region>* Services In Oracle Services Network** as the destination.
- Checks if the existing subnet for the load balancer has a security list that enables inbound access to ports 80 and 443.
- Validates if all protocols are open in private subnet for Kubernetes worker node for the Worker CIDR range.
- Validates if all protocols are open in private subnet for Kubernetes worker node for the VCN CIDR range.
- Validates if the file shared system has a security list that enables outbound access to ports 111 and 2048 (both TCP and UDP).
- Validates if the database port is accessible from WebLogic Server subnets.

Using the Validation Script

You can run the helper scripts to perform validations for existing private subnets, existing public subnets, and existing VCN peered subnets.

You must run the commands on the validation script file to check the existing network setup. For example, in this case, let's run the commands on the validation script file named `validateoke.sh`. See Script File To Validate Network Setup to create the `validateoke.sh` file.

1. Set execute permission to the `validateoke.sh` file.

   ```
   chmod +x validateoke.sh
   ```

2. Run the following command prior to creating a domain:

   - Basic domain

     ```
     ./validateoke.sh -b <Bastion Subnet OCID> -a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File Shared System Subnet OCID> -l <Load Balancer Subnet OCID>
     ```

> **Note:**
>
> If you restricted the bastion compute instance to access port 22 in WebLogic subnet, you can validate using the `Bastion Host IP CIDR` rather than the entire bastion subnet CIDR.
>
> ```
> ./validateoke.sh -b <Bastion Subnet OCID> -i <Bastion Host IP CIDR> -
> a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File
> Shared System Subnet OCID> -l <Load Balancer Subnet OCID>
> ```

```
validateoke.sh

example_user@cloudshell:~ (us-phoenix-1)$ ./validateoke.sh -b <Bastion Subnet
OCID>
-a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File Shared
System Subnet OCID>
-l <Load Balancer Subnet OCID>
ERROR: SSH port 22 is not open for access by [0.0.0.0/0] in <Bastion Subnet
OCID>
WARNING: SSH port 22 is not open for access by Bastion Subnet CIDR
[10.0.0.0/24] in private Admin Host Subnet [<Administration Host Subnet OCID>]
ERROR: Missing Service or NAT gateway in the VCN of the private ADMIN_SUBNET
Host subnet ocid [<Administration Host Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the BASTION_SUBNET subnet
[<Bastion Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the LB_SUBNET subnet [<Load
Balancer Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the FSS_SUBNET_OCID subnet
[<File Shared System Subnet OCID>]
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 31474
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 10256
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 31804
WARNING: All Ports are not open for LB Subnet CIDR
WARNING: All Ports are not open for LB Subnet CIDR
WARNING: All Ports are not open for LB Subnet CIDR
ERROR: All Protocols are not open for WORKER's Subnet CIDR
ERROR: All Protocols are not open in WORKER's Subnet for VCN CIDR
ERROR: TCP -- 111 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2048 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2049 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2050 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 111 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2048 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2049 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2050 -- Port is not open in FSS Subnet for VCN CIDR
```

Create a Domain

After you create a stack, use the Jenkins job to create a domain for Oracle WebLogic Server for OKE.

When you create a domain with the **create domain** job, a new domain that has a basic configuration with no custom applications or libraries, called the primordial domain, is created.

This domain contains the base WebLogic Server image that has the WebLogic installer, JDK installer, and WebLogic patches for Oracle WebLogic Server for OKE.

To know about the primordial domain, see Mutate the Domain Layer in *WebLogic Kubernetes Operator* documentation.

To create a domain, you must create a domain as a component, register the component, and register the application.

Topics:

Create a Domain as a Component

Before you create a domain as component, ensure that all the prerequisites are completed. See Prerequisites to Create a Domain.

Perform the following tasks:

- Configure WebLogic Server
- Configure the Registry
- Configure the Domain
- Run the Pipeline Job

Configure WebLogic Server

Specify the parameters required to configure a WebLogic server on a container cluster.

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **create domain as component**.
3. Click **Build with Parameters**.
4. For **Domain_Name**, specify a WebLogic name.
5. For **WebLogic_Version**, select a version of Oracle WebLogic Server.
   The available versions are 12.2.1.4.0, 14.1.1.0.0 running on JDK 8, and 14.1.1.0.0 running on JDK 11.
6. *Optional*: Select the required base image from the **Base_Image** list.
   The images are displayed based on the Weblogic Server version. For example, if you select 12.2.1.4.0, 12.2.1.4 images are displayed, and if you select 14.1.1.0.0, 14.1.1.0.0_jdk8 and 14.1.1.0.0_jdk11 images are displayed.

> **Note:**
>
> For 14.1.1.0.0., make sure that the base image you select should have the same WebLogic Server and JDK version as in step 5.
>
> The custom base images and uploaded images are also listed in the **Base_Image** list. See Create a Custom Base Image.

7. Enter a user name for the WebLogic Server administrator.
8. Enter the password for the WebLogic Server administrator.
9. Select the number of running managed servers in the domain you want to create. You can specify up to nine manged servers.
   The number of running managed servers is also the number of WebLogic Server pods in the Kubernetes cluster. Each managed server runs in a separate pod in the Kubernetes cluster. Managed servers are members of a WebLogic Server cluster.

Configure the Registry

Specify the credentials to access container images in the Oracle Cloud Infrastructure Registry(OCIR).

> **Note:**
>
> If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

1. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.
   The registry user name format is `tenancy_namespace/<username>`. If your tenancy is federated with Oracle Identity Cloud Service, then the registry user name format is `tenancy_namespace/oracleidentitycloudservice/<username>`.

   You can choose either to include the `tenancy_namespace` or remove the `tenancy_namespace` in the user name format. For example, you can either use `tenancy_namespace/<username>` or `<username>`.

   > **Note:**
   >
   > If you choose to include `tenancy_namespace` in the user format, ensure that you use the correct namespace for your tenancy.

2. In the **Registry_Authentication_Token** field, enter the OCID of the secret for the auth token generated for the registry user.

For information about how to create a container registry, see Overview of Registry in the Oracle Cloud Infrastructure documentation.

Configure the Domain

In the Provision with JRF section, keep the default selection for **Domain_Type** as **Non_JRF**.

Run the Pipeline Job

Click **Build** to run the Pipeline job.

After the job is successful, you can access the WebLogic Console. See Access the WebLogic Console.

Register a Component

Use the Jenkins job to specify the Verrazzano WebLogic component to register for the WebLogic domain.

See About Verrazzano WebLogic Components and Application Configuration.

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **register component**.
3. Click **Build with Parameters**.
4. From **Component_Names**, select the Verrazzano component to register.
5. Click **Build** to run the Pipeline job.

Create an Application Configuration

Use this job to select a component, which allows to generate the basic application configuration with an *IngressTrait* and a *MetricsTrait*.

Perform the following tasks:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **create application configuration**.
3. Click **Build with Parameters**.
4. For **Component_Name**, select the Verrazzano component for which you want to create the application configuration.
5. Click **Build** to run the Pipeline job.

Register an Application

Use the Jenkins job to deploy the Verrazzano application to a Verrazzano system.

See About Verrazzano WebLogic Components and Application Configuration.

You can use one of the following sources to specify the location of the application configuration file:

- Application Name - The Verrazzano application to register.
- File Upload - Uploads the file from the local system.
- Object Storage - Uses the pre-authenticated URL on the Object Storage.
- Shared File System - Uses the path of the shared file storage.

  The NFS shared file system path is mounted on `/u01/shared` location on the administration host.

In the job you should select either an application or provide a custom application yaml.

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **register application**.
3. Click **Build with Parameters**.
4. Select either one of the following:
   - Select the application from the **Application_Name** list.
   - Select the source of the application configuration file from the **Application_Source** list.

   > **Note:**
   > - If both the application name and a custom application yaml is provided, the application name selection is ignored.
   > - For the **Application_Source** option, if the files are larger than one MB, then use the Object Storage or Shared File System option.

5. If you have select the **Application_Source** option:

   For **Application_Source_Location**, either browse to select the zip file, specify the http pre-authenticated URL, or specify the path of the configuration Yaml file on the shared file system.
6. Click **Build** to run the Pipeline job.

Update a WebLogic Domain

Learn how to update a domain for Oracle WebLogic Server for OKE stack with Verrazzano by using the tools provided with Oracle WebLogic Server for OKE.

Topics:

Create a Custom Base Image

Use the Jenkins job to create a custom base WebLogic image from WebLogic installer, JDK installer, and WebLogic patches for Oracle WebLogic Server for OKE.

You can use one of the following sources to specify the location of the JDK installer file, WebLogic installer file, and WebLogic patch file:

- Object Storage - Uses the pre-authenticated URL on the Object Storage.

  For JDK installer file, you must specify the location of a JDK that uses the Linux x64 compressed archive format (`.tar.gz`).
- Shared File System - Uses the path of the shared file storage.

  The NFS shared file system path is mounted on `/u01/shared` location on the administration host.

> **Important:**
>
> You can get your own preferred WebLogic base image and upload them to the OCIR. You must upload the images to the following location:
>
> `<region>/<tenancy>/<servicename>/wls-base-image/12214`

To create a custom base image:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **create base image**.
3. Click **Build with Parameters**.
4. Select the source of the JDK installer file from the **JDK_Installer** list.
5. For **JDK_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system.
6. From **FMW_Installer**, select the source of the Fusion Middleware installer file.
7. For **FMW_Installer_Location**, specify the http pre-authenticated URL or the path of the JAR file on the shared file system.
8. *Optional:* Select patches.
   a. Select the source of the Weblogic patch file from the **WLS_Opatches** list. **WLS_Opatches** refers to any patch that may be required to patch WLS or Coherence product binaries (including ADR patches, OPSS, OWSM, OPatches).
   b. If you selected a Weblogic patch file: For **Opatches_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system. For multiple patches, specify the location in separate lines.
9. *Optional:* Configure the Registry.

   If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).
   a. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.
   b. In the **Registry_Authentication_Token** field, enter the OCID for the secret for the auth token generated for the registry user.
10. Click **Build** to run the pipeline job.

    The custom base image is created and is available in the following location.

    `<region>/<tenancy>/<servicename>/wls-base-image/12214`

The created custom base images and uploaded images are listed when you create a domain, in the **Base_Image** list. See Create a Domain as a Component.

Update a Domain Configuration

Use the Jenkins job to update a domain that you created with Oracle WebLogic Server for OKE.

When you run the **update domain** job, the primordial domain created using the **create domain** job is used as the base domain, and updates are applied to this domain.

Therefore, if you run the **update domain** job for the first time and deploy an application-A in this job, the resulting primordial domain will have application-A deployed. Then, if you run the **update domain** job for the second time and deploy only application-B in the job, the resulting primordial domain will have only application-B deployed. That is, the state on top of the primordial domain is not stored within the image and this allows you to use the source control system to store WebLogic Deploy Tooling (WDT) model to persist any configuration changes on top of the primordial domain.

To know about the primordial domain, see Mutate the Domain Layer in *WebLogic Kubernetes Operator* documentation.

You can extend your model Yaml file with new definitions using the **update domain** job. For model Yaml file, see About WebLogic Deploy Tooling.

> **Tip:**
>
> To deploy and undeploy an application, see Tutorial.

To update the WebLogic domain you can provide a WDT model file, a WDT properties file, or an archive file. The archive file can contain applications, libraries, model file along with other artifacts. For the structure of the archive file, see archive file in *WebLogic Deploy Tooling* documentation.

When you are updating the domain, if you provide a model through the WDT model field and within the archive, the model specified in the field takes precedence.

You can use one of the following sources to specify the location of the archive zip file, domain model Yaml file, and variables properties file:

- File Upload - Uploads the file from the local system.
- Object Storage - Uses the pre-authenticated URL on the Object Storage.
- Shared File System - Uses the path of the shared file storage.

  The NFS shared file system path is mounted on `/u01/shared` location on the administration host.

To update the domain configuration:

- The model Yaml file must contain the required secrets like WebLogic Admin Password, runtime encryption secret, and Repository Schema Utility (RCU) schema user password.
- In case of Oracle Cloud Infrastructure Database (DB system), the model Yaml file must include the datasource secret and in case of Oracle Autonomous Database, the model Yaml file must include the datasource secret and wallet with the keystore passwords. See About Data Sources.

See RCU Connection Information in *WebLogic Deploy Tooling* documentation.

To update a domain and deploy applications, shared libraries, and resources to the domain:

> **Note:**
>
> Every time you update a domain, you have to provide a model containing all the resources (applications, libraries, and so on) you want your domain to have. If your domain currently have application-A installed, and you update your domain with a model with application-B, then the domain will have only application-B. If you want to have both applications, you have to use a model with both application-A and application-B.

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **update domain in component**.
3. Click **Build with Parameters**.
4. From **WebLogic_Component**, select the component that you want to update.
5. Select the source of the archive zip file from the **Archive_Source** list.

   > **Note:**
   >
   > In case of files larger than one MB, use the **Object Storage** or **Shared File System** option.

6. For **Archive_File_Location**, either browse to select the zip file, specify the http pre-authenticated URL, or specify the path of the zip file on the shared file system.

   To know about the structure of the archive file, see archive file in *WebLogic Deploy Tooling* documentation.
7. Select the source of the domain model Yaml file from the **Domain_Model_Source** list.

   If the archive zip file contains the domain model Yaml file, you can skip this step.
8. For **Model_File_Location**, either browse to select the Yaml file, specify the http pre-authenticated URL, or specify the path of the Yaml on the shared file system.

   See model YAML file in *WebLogic Deploy Tooling* documentation.
9. Select the source of the variable properties file from the **Variable_Source** list.
10. For **Variable_File_Location**, either browse to select the file, specify the http pre-authenticated URL, or specify the path of the properties file on the shared file system.
11. *Optional*: Configure the Registry.

    > **Note:**
    >
    > If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

    a. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.

b. In the **Registry_Authentication_Token** field, enter the OCID for the secret for the auth token generated for the registry user.

12. Click **Build** to run the Pipeline job.

You can use the WebLogic Server Administration Console to verify that the domain is updated with all the specified parameters. See Access the WebLogic Console.

After you update the domain, you must register the component and register the application. See Register a Component and Register an Application.

You need not register the application if the application pods are running on the node pool.

Rebase a Component Image

Use the Jenkins job to update the Verrazzano component with a different base image for Oracle WebLogic Server for OKE.

If you created a new base image to apply patches to the JDK and WebLogic binaries, or to apply new set of OS packages, you can update the existing base image in your Verrazzano component with the **rebase domain** job.

To update a base image (JDK and Fusion Middleware) of the Verrazzano component with a different base image:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **rebase component image**.
3. Click **Build with Parameters**.
4. From **Component_Name**, select the Verrazzano component that you want to update.
5. From **Base_Image**, select the image that you to apply to the selected component.
6. *Optional:* Configure the Registry.

   If you want to use another user credentials, other than the one specified when creating a stack, then specify the credentials that Oracle WebLogic Server for OKE must use to access container images in the Oracle Cloud Infrastructure Registry (OCIR).

   a. In the **Registry_Username** field, enter a user name that Kubernetes uses to access the image in the registry.

   b. In the **Registry_Authentication_Token** field, enter the OCID for the secret for the auth token generated for the registry user.

7. Click **Build** to run the Pipeline job.

This job removes the domain definition from the existing domain and rebases the Verrazzano component with the selected base image.

After you rebase the component image, you must register the component and register the application. See Register a Component and Register an Application.

You need not register the application if the application pods are running on the node pool.

Upgrade Verrazzano

You can upgrade to the latest available Verrazzano version for a Oracle WebLogic Server for OKE stack with Verrazzano.

You can upgrade the Verrazzano version using private registry (offline upgrade) or using the internet (online upgrade).

Upgrade Using Private Registry

To upgrade the Verrazzano version:

1. Download the latest version of the Verrazzano zip file from Oracle Software Delivery Cloud.

   > **Note:**
   >
   > For patched versions, go to Oracle Support and raise a service request for the latest version of the Verrazzano zip file.

   a. In your browser, go to Oracle Software Delivery Cloud and log in with your credentials.
   b. Click the drop-down next to the search bar and select **Download Package**.
   c. In the search bar, enter *Verrazzano Enterprise Container Platform* and click **Search**.
   d. Select the *DLP: Oracle Verrazzano Enterprise Edition <version>* link. This adds the link to your download queue.
   e. Select the **Continue** link.
   f. Accept the license agreement and click **Continue**.
   g. Download the file:
      - To download the zip file directly, select the file link in the list.
      - To download the zip file using Oracle Download Manager, click **Download** and run the Oracle Download Manager executable file.
2. Place the zip file in `/u01/shared` location on the administration host.
3. Run the following upgrade script from the administration host:

```
/u01/scripts/verrazzano/utils/upgrade-verrazzano.sh -v <latest_version> -p
-a <path_to_Verrazzano_archive(zip)_file>
```

Upgrade Using Internet

Ensure that a NAT gateway is created for the administration instance private subnet and the worker nodes private subnet.

To upgrade the Verrazzano version using internet, run the following upgrade script from the administration host:

```
/u01/scripts/verrazzano/utils/upgrade-verrazzano.sh -v <latest_version>
```

> **Note:**
>
> If you installed the current Verrazzano version in your Kubernetes cluster using private registry, before you upgrade Verrazzano, perform the following steps:
>
> 1. Get the name of the verrazzano-platform-operator helm chart.
>
>    ```
>    helm list -o json | jq -r '.[] | select(.chart |
>    contains("verrazzano-platform-operator")) | .name'
>    ```
>
> 2. Uninstall the verrazzano-platform-operator helm chart.
>
>    ```
>    helm uninstall <verrazzano-platform-operator>
>    ```

Troubleshoot a WebLogic Domain

Learn about the common issues when creating and managing a domain and then how to diagnose to solve them.

Topics

Terminate a WebLogic Domain

Use the Jenkins job to terminate a domain that you created for Oracle WebLogic Server for OKE with Verrazzano.

To terminate a domain, you must unregister the application, unregister the component, and then terminate the component.

Topics:

- Terminate a Component

Unregister an Application

Use the Jenkins job to stop the application pods that are running on the node pool.

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **unregister application**.
3. Click **Build with Parameters**.
4. From **Application_Names**, select the Verrazzano application to unregister.
5. Click **Build** to run the Pipeline job.

Unregister a Component

Use the Jenkins job to specify the Verrazzano component to unregister for the WebLogic domain.

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **unregister component**.
3. Click **Build with Parameters**.
4. From **Component_Names**, select the Verrazzano component to unregister.
5. Click **Build** to run the Pipeline job.

Terminate a Component

Use the Jenkins job to terminate the Verrazzano component and associated resources such as RCU, external load balancer, and the Kubernetes namespaces that we created for the domain.

To terminate a component:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **terminate component**.
3. Click **Build with Parameters**.
4. From **Component_Name**, select the component that you want to terminate.
5. Click **Build** to run the Pipeline job.

After the terminate job is complete, it displays a log that includes information about the actions that were performed by the terminate job.

5
Managing Resources

Learn how to stop, start, back up, scale, and perform other management operations on your domain.

For information about scaling, see Scaling in *WebLogic Kubernetes Operator* documentation.

Topics:

About Data Sources

Learn about creating and adding additional data sources after you create an Oracle WebLogic Server for OKE domain.

For mutable values such as database user names, passwords, and URLs, you must use model macros to reference arbitrary secrets from model Yaml files. All password fields in the model Yaml must use secret macros; passwords should not be directly included in the model Yaml files. So, to create the data source for the database, you must create data source secrets.

Before creating data sources, you must set up the database to create a schema user. See Prerequisites to Create a Data Source.

You can create data sources that enable you to connect to either an Oracle Autonomous Database or an Oracle Cloud Infrastructure Database (DB System) database. You can also create multi data sources and Active GridLink (AGL) data source when using Oracle Real Application Clusters (RAC) database.

Topics:

Prerequisites to Create a Data Source

Before you create a data source, you must set up the database to create a schema user.

Complete the following step to create a schema user:

1. Create a schema user for the datasource.

   For example, let's create `hello` schema user:

   ```
   create user hello identified by "<password>";

   GRANT CONNECT TO hello;
   GRANT RESOURCE TO hello;
   GRANT DBA TO hello;

   GRANT CREATE SESSION TO hello;
   GRANT UNLIMITED TABLESPACE TO hello;

   ALTER SESSION SET CURRENT_SCHEMA = hello;

   CREATE TABLE products
   ( product_id number(10) not null,
   product_name varchar2(50) not null,
   CONSTRAINT products_pk PRIMARY KEY (product_id)
   );

   insert into products values
   (1, 'lamp');

   commit;
   ```

2. Add the schema user that you created in step 1 to the model Yaml template.

   Example:

   ```
   Properties:
               user:
                 Value:<schema user name>
   ```

Create a Data Source for an ATP Database

Use the model Yaml templates to create a data source for an Oracle Autonomous Database.

Following is an example of model Yaml template for app deployment with ATP datasource properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    hellods:
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          JNDIName:
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-
secret:password@@'
          Properties:
            user:
              Value:
            javax.net.ssl.keyStore:
              Value:
            javax.net.ssl.keyStoreType:
              Value: JKS
            javax.net.ssl.keyStorePassword:
              Value: '@@SECRET:@@ENV:DOMAIN_UID@@-keystore-secret:password@@'
            javax.net.ssl.trustStore:
              Value:
            javax.net.ssl.trustStoreType:
              Value: JKS
            javax.net.ssl.trustStorePassword:
              Value: '@@SECRET:@@ENV:DOMAIN_UID@@-keystore-secret:password@@'
            oracle.net.ssl_version:
              Value: '1.2'
            oracle.net.ssl_server_dn_match:
              Value: true
            oracle.net.tns_admin:
              Value:
            oracle.jdbc.fanEnabled:
              Value: false
        JDBCConnectionPoolParams:
          InitialCapacity: 1
          MaxCapacity: 1
          TestTableName: SQL ISVALID
          TestConnectionsOnReserve: true
```

You must set up the database to create a schema user before you create the data source. See Prerequisites to Create a Data Source.

When you create a data source for an ATP database, you need the ATP client credentials or wallet files. So, you must run the download script before you create the data source. See Download the ATP Wallet.

To create a data source for an ATP database:

1. Update the `Properties` section in the model Yaml template.

   For example, update `keyStore` and `trustStore` file locations from the downloaded ATP wallet, `tns_admin` to point to the directory of the ATP wallet, and the schema user you created in Prerequisites to Create a Data Source.

   ```
   Properties:
     user:
       Value: <schema user>
     javax.net.ssl.keyStore:
       Value: /u01/shared/atp_wallet/keystore.jks
     javax.net.ssl.trustStore:
       Value: /u01/shared/atp_wallet/truststore.jks
     oracle.net.tns_admin:
       Value: /u01/shared/atp_wallet
   ```

2. Create the data source secrets in the model Yaml file as follows:

   a. Create `<domainuid>-datasource-secret` with password and url using the `kubectl` command.

      Example:

      ```
      kubectl create secret generic <domainuid>-datasource-secret --from-
      literal=password=<password>
      --from-literal=url='jdbc:oracle:thin:@(description=(retry_count=20)
      (retry_delay=3)(address=(protocol=tcps)(port=1522)(host=adb.us-
      ashburn-1.oraclecloud.com))(connect_data=(service_name=<service_name>))
      (security=(ssl_server_dn_match=yes)))' -n <domainname>-ns
      ```

   b. Create the `<domainuid>-keystore-secret` with the password used to download the ATP wallet using the `kubectl` command.

      For example:

      ```
      kubectl create secret generic <domainuid>-keystore-secret --from-
      literal=password=<password> -n <domainname>-ns
      ```

      > **Note:**
      >
      > You must use single quotes ' ' to escape special characters such as $, \, *, =, and ! in your strings. Otherwise, your shell will interpret these characters and the create secret command will fail.

Download the ATP Wallet

The download script unpacks and copies the ATP wallet contents to a node.

1. Open an SSH connection to the stack's Administration Server node as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

2. Download the ATP wallet to the administration host using the following wallet script:

```
python3 /u01/scripts/utils/download_atp_wallet.shutils/oci_api_utils.py
<atp_database_ocid>
 <atp_wallet_password> <path_to_extract_wallet_files>
```

Example:

```
python3 /u01/scripts/utils/oci_api_utils.py download_atp_wallet_with_pwd
ocid1.autonomousdatabase.oc1.phx.abyhqljtzrr25tfpdyzbjq5udz3lz2hkb5txvtfejc
kwd25z6hjg6qbxm4ta <password>
/u01/shared/atp_wallet
```

Eight files are extracted to the subdirectory. The following is an example of the script response:

```
<Apr 29, 2021 09:51:35 PM GMT> <INFO> <oci_api_utils> <(host:ly2-
admin.okeworkdersregi.paasdevjcsphx.oraclevcn.com) - <WLSOKE-VM-
INFO-0100> : ATP Wallet downloaded>
Archive:  /tmp/atp_wallet.zip
  inflating: /u01/shared/atp_wallet/README
  inflating: /u01/shared/atp_wallet/cwallet.sso
  inflating: /u01/shared/atp_wallet/tnsnames.ora
  inflating: /u01/shared/atp_wallet/truststore.jks
  inflating: /u01/shared/atp_wallet/ojdbc.properties
  inflating: /u01/shared/atp_wallet/sqlnet.ora
  inflating: /u01/shared/atp_wallet/ewallet.p12
  inflating: /u01/shared/atp_wallet/keystore.jks
```

Create a Data Source for a DB System Database

Use the model Yaml templates to create a data source for an Oracle Cloud Infrastructure Database.

Following is an example of model Yaml template for app deployment with single-instance (SI) DB System data source properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    hellods:
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          JNDIName: jdbc/hellods
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
```

```
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-
secret:password@@'
          Properties:
            user:
              Value:
            oracle.net.CONNECT_TIMEOUT:
              Value: '120000'
            SendStreamAsBlob:
              Value: true
        JDBCConnectionPoolParams:
          InitialCapacity: 1
          MaxCapacity: 1
          TestTableName: SQL ISVALID
          TestConnectionsOnReserve: true
```

You must set up the database to create a schema user before you create the data source and update . See Prerequisites to Create a Data Source.

1. Update the schema user that you created in the `Properties` section in the model Yaml template.

   See Prerequisites to Create a Data Source.

   For example:

   ```
   Properties:
     user:
       Value: <schema user>
   ```

2. Create the data source secrets in the model Yaml file as follows:

   a. Obtain the `tns` connect string using the python script.

   ```
   python3 /u01/shared/scripts/pipeline/create_domain/scripts/
   precheck_utils.py "get_oci_db_connect_string" "<database_ocid>"
   ```

   b. Create `<domainuid>-datasource-secret` with password and url using the `kubectl` command.

   For example:

   ```
   kubectl create secret generic <domainuid>-datasource-secret --from-
   literal=password=<password>
   --from-literal=url='jdbc:oracle:thin:@//wrdb19-
   scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:1521/
   WRPDB.SUBNET2AD2PHX.PAASDEVJCSPHX.ORACLEVCN.COM'
   -n <domainname>-ns
   ```

Create a Multi Data Source for a RAC Database

Use the model Yaml templates to create a multi data source for an Oracle Real Application Cluster (RAC) database with WebLogic Enterprise Edition.

Following is an example of model Yaml template for app deployment with multi data source (RAC as infra DB with WebLogic Enterprise Edition) properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    'db1-hellods':
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          GlobalTransactionsProtocol: None
          JNDIName: [ 'jdbc/db1-hellods' ]
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-db1datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-db1datasource-
secret:password@@'
          Properties:
            user:
              Value:
        JDBCConnectionPoolParams:
          StatementCacheSize: 0
          TestTableName: SQL ISVALID
          InitialCapacity: 1
          MaxCapacity: 1
    'db2-hellods':
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          GlobalTransactionsProtocol: None
          JNDIName: [ 'jdbc/db2-hellods' ]
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-db2datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-db2datasource-
secret:password@@'
          Properties:
            user:
              Value:
        JDBCConnectionPoolParams:
          StatementCacheSize: 0
          TestTableName: SQL ISVALID
          InitialCapacity: 1
          MaxCapacity: 1
    'hellods':
       Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
       JdbcResource:
         DatasourceType: MDS
         JDBCDataSourceParams:
           AlgorithmType: 'Load-Balancing'
           DataSourceList: [ 'db1-hellods','db2-hellods' ]
           JNDIName: [ jdbc/hellods ]
```

You must set up the database to create a schema user before you create the data source. See Prerequisites to Create a Data Source.

1. Update the schema user that you created in the `Properties` section in the model Yaml template.

   See Prerequisites to Create a Data Source.

   For example:

   ```
   Properties:
     user:
       Value: <schema user>
   ```

2. Create the data source secrets in the model Yaml file as follows:

   a. Open an SSH connection to the domain's Administration Server node as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

   b. Go to `/u01/shared/helper-scripts` location and obtain the connect string for each node using the following commands:

   ```
   ./url.sh
   URL1 = jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
   (HOST=<host_name>)(PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=<service_name>)
   (INSTANCE_NAME=<instance1_name>)))
   URL2 = jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
   (HOST=<host_name>)(PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=<service_name>)
   (INSTANCE_NAME=<instance2_name>))
   ```

   c. Create `<domainuid>-db1datasource-secret/<domainuid>-db2datasource-secret` with password and url using the `kubectl` commands.

   ```
   kubectl create secret generic <domainuid>-db1datasource-secret --from-
   literal=password=<password>
   --from-
   literal=url='jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
   HOST=<host1_name>)(PORT=1521))CONNECT_DATA=(SERVICE_NAME=<service_name>)
   (INSTANCE_NAME=<instance_name>)))'
   -n <domainname>-ns

   kubectl create secret generic <domainuid>-db2datasource-secret --from-
   literal=password=<password>
   --from-
   literal=url='jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
   HOST=<host2_name)(PORT=1521))CONNECT_DATA=(SERVICE_NAME=<service_name>)
   (INSTANCE_NAME=<instance_name>)))'
   -n <domainname>-ns
   ```

Create an Active GridLink Data Source for a RAC Database

Use the model Yaml templates to create an Active GridLink (AGL) data source for an Oracle Real Application Cluster (RAC) database with WebLogic Suite Edition.

Following is an example of model Yaml template for app deployment with AGL datasource (RAC as infra DB with WebLogic Enterprise Edition) properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    'hellods':
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        DatasourceType: AGL
        JDBCConnectionPoolParams:
          StatementCacheSize: 0
          TestTableName: SQL ISVALID
        JDBCDataSourceParams:
          GlobalTransactionsProtocol: None
          JNDIName: [ jdbc/hellods ]
        JDBCDriverParams:
          DriverName: oracle.jdbc.replay.OracleDataSourceImpl
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-
secret:password@@'
          Properties:
            user:
              Value:
        JDBCOracleParams:
          FanEnabled: true
          ActiveGridlink: true
          OnsNodeList:
```

You must set up the database to create a schema user before you create the data source. See Prerequisites to Create a Data Source.

1. Update the `Properties` section in the model Yaml template to set `OnsNodeList`.
   a. Open an SSH connection to a node as the `opc` user.

      ```
      ssh -i <path_to_private_key> opc@<node_public_ip>
      ```

   b. Go to `/u01/shared/helper-scripts` location and invoke the following script:

      ```
      ./scan_address.sh
      <db_hostname>-scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:6200
      ```

2. Create the data source secrets in the model YAML file as follows:

   a. Obtain the `tns` connect string using the python script.

   ```
   python3 /u01/shared/scripts/pipeline/create_domain/scripts/
   precheck_utils.py "get_oci_db_connect_string" "<database_ocid>"
   ```

   b. Create `<domainuid>-datasource-secret` with password and url using the `kubectl` command.

   For example:

   ```
   kubectl create secret generic <domainuid>-datasource-secret --from-
   literal=password=<password>
   --from-literal=url='jdbc:oracle:thin:@//<db_hostname>-
   scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:1521/<service_name>'
   -n <domainname>-ns
   ```

Authenticate by using an External LDAP Server

This section describes the steps required to add authenticators for external systems, like OpenLDAP. This helps you to use the groups and users defined in the system for your applications deployed in the Oracle WebLogic Server for OKE domain. Also, it provides information about how to use an OpenLDAP server.

In Oracle WebLogic Server for OKE, you cannot add users and groups to the WebLogic embedded LDAP server for your applications. As the embedded LDAP has limited use when using the domain home in the image model, if you add users to the embedded LDAP through the Administration console, the users are not persisted in the image and disappear when you restart the admin server pod. Also, it is not recommended to change the domain from the administration console.

Prerequisites

Before you authenticate by using an external LDAP sever, ensure that you have completed the required prerequisites.

1. You must have created a domain with Oracle WebLogic Server for OKE instance. See Create a Stack .
2. You must have an OpenLDAP server, which is ready to use.
   The OpenLDAP server must be accessible from the Oracle WebLogic Server for OKE nodes, where your WebLogic domain is running and from the admin host. That is, OpenLDAP server must connect to the OpenLDAP host (either by name or IP address) and use the port where the LDAP server is listening.

   Default ports are `389` for LDAP and `636` for LDAPS (LDAP over SSL).

   If the LDAP server is on premises, then connect the VCN where the stack was created with your datacenter, by using either FastConnect or VPN connect. See Access to Your On-Premises Network.

Add a new OpenLDAP Authenticator to the Domain

Define the authenticators that you want to add to your WebLogic domain in a WDT model file, and then apply this model by using the `update-domain` pipeline job.

1. Create an `OpenLDAP_authenticator.yaml` file.
   The following is an example of a model file specifying an OpenLDAP authenticator and helps to connect to an OpenLDAP server by using LDAP protocol, that is with SSL disabled. This model is based on the models presented in Modeling Security Providers.

   > **Note:**
   >
   > You must create a secret for the administrator password, and you use this secret when you run the `update-domain` pipeline job.

   Example of an `OpenLDAP_authenticator.yaml` file:

```
topology:
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True
                            EnableGroupMembershipLookupHierarchyCaching:
True
                            Host: myldap.example.com
                            Port: 389
                            UserObjectClass: inetOrgPerson
                            GroupHierarchyCacheTTL: 600
                            SSLEnabled: False
                            UserNameAttribute: cn
                            Principal: 'cn=foo,ou=users,dc=example,dc=com'
                            CredentialEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-secret:password@@'
                            UserBaseDn: 'ou=users,dc=example,dc=com'
                            UserSearchScope: subtree
                            UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                            GroupBaseDN: 'ou=groups,dc=example,dc=com'
                            StaticGroupObjectClass: groupofnames
                            StaticGroupNameAttribute: cn
                            StaticMemberDNAttribute: member
                            StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                            UseRetrievedUserNameAsPrincipal: True
                            KeepAliveEnabled: True
                            GuidAttribute: uuid
                    DefaultAuthenticator:
                        DefaultAuthenticator:
                            ControlFlag: SUFFICIENT
```

```
DefaultIdentityAsserter:
    DefaultIdentityAsserter:
```

In order to keep `DefaultAuthenticator` and `DefaultIdentityAsserter` while changing or adding providers, they must be specified in the model with any non-default attributes, as in the example. The order of providers in the model will be the order the providers set in the WebLogic security configuration. See Modeling Security Providers.

2. Run the `update-domain` pipeline job to add the authenticators. See Update a Domain Configuration.

Enable SSL Support

To enable SSL support, you need to perform a few additional steps. Here you need to configure both a trust keystore and an identity keystore, although only trust keystore is required for one-way SSL connection to the LDAP server. You must also configure SSL with the host name verifier.

1. Obtain the root Certificate Authority (CA) certificate for the LDAP server.
2. Create a trust keystore by using the preceding certificate or if you already have an existing trust keystore, import the certificate to the trust keystore.
   Following is an example to create the keystore *myTrust.jks* with the root CA certificate `rootca.pem`, by using the `keytool` command:

   ```
   keytool -import -keystore ./myTrust.jks -trustcacerts -alias oidtrust -
   file rootca.pem -storepass TrustKeystorePwd -noprompt
   ```

3. Create an identity keystore, if you do not have an existing identity keystore.
   Following is an example to create the identity keystore:

   ```
   keytool -genkeypair -alias server_cert -keyalg RSA -sigalg SHA256withRSA -
   keysize 2048 -dname
   "CN=example.com,OU=Support,O=Example,L=Reading,ST=Berkshire,C=GB" -
   keystore ./myIdentity.jks
   ```

4. Copy the trust and identity keystores to the `u01` shared location, and specify the location of these files in the `model.yaml` file.
5. Create a `model.yaml` file, specifying the OpenLDP Authentication Provider with SSL enabled, the `DefaultAuthenticator` and `DefaultIdentityAsserter` information, and the custom keystores for admin server and the servers that are part of the dynamic cluster configured in the domain.

> **Note:**
>
> You must create a secret for the administrator password, and provide this secret in the `model.yaml` file; this secret is used when you run the `update-domain` job. For information see, Update a Domain Configuration.
>
> You must use ENV macros for server names as specified in the following `model.yaml` example files for non-JRF domain.

Following is a sample of the `model.yaml` file for a non-JRF domain:

In this `model.yaml` file, `ServerPrivateKeyAlias`, refers to the alias used when you created the `DemoIdentity` keystore and `ServerPrivateKeyPassPhraseEncrypted` refers to the password set for `ServerPrivateKeyAlias`.

```
topology:
    Server:
        '@@ENV:RESOURCE_PREFIX@@-adminserver':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
    ServerTemplate:
        '@@ENV:RESOURCE_PREFIX@@-cluster-template':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True
                            EnableGroupMembershipLookupHierarchyCaching:
True
                            Host: 'pg-openldap'
                            Port: 636
                            UserObjectClass: inetOrgPerson
                            GroupHierarchyCacheTTL: 600
                            SSLEnabled: True
                            UserNameAttribute: cn
                            Principal: 'cn=admin,dc=wlsoketest-ldap,dc=com'
                            CredentialEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-secret:password@@'
                            UserBaseDn: 'ou=people,dc=wlsoketest-
ldap,dc=com'
```

```
                                UserSearchScope: subtree
                                UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                                GroupBaseDN: 'ou=groups,dc=wlsoketest-
ldap,dc=com'
                                StaticGroupObjectClass: groupofnames
                                StaticGroupNameAttribute: cn
                                StaticMemberDNAttribute: member
                                StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                                UseRetrievedUserNameAsPrincipal: True
                                KeepAliveEnabled: True
                                GuidAttribute: entryuuid
                        DefaultAuthenticator:
                            DefaultAuthenticator:
                                ControlFlag: SUFFICIENT
                        DefaultIdentityAsserter:
                            DefaultIdentityAsserter:
```

6. Apply the model and archive to the running WebLogic Server domain.
   To run the `update-domain` CI/CD pipeline to update the running domain and add the Authentication Providers and custom keystores, complete the following steps:
   a. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
   b. On the Dashboard page, click **update domain**.
   c. From **Domain_Name**, select the domain that you want to update.
   d. Click **Build with Parameters**.
   e. Select **Shared File System** from the **Archive_Source** list.
   f. For **Archive_File_Location**, browse to select the archive zip file or specify the path of the zip file on the shared file system.
   g. Select **Shared File System** from the **Domain_Model_Source** list.
   h. For **Model_File_Location**, browse to select the YAML file or specify the path of the YAML on the shared file system.
   i. Select **None** from the **Variable_Source** list.
   j. Select the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).
      If you deselected this check box, you can rollback to the previous image later from the backup.

      The **Rollback_On_Failure** check box is selected by default.
   k. Click **Build** to run the Pipeline job.

Configure SSL with host name verifier

1. In the `model.yaml` file, for the non-JRF domain, add the SSL configuration with custom `HostnameVerifier` class in the admin sever and managed server.

   Following is a sample `model.yaml` file for a non-JRF domain:

```
topology:
  Server:
    '@@ENV:RESOURCE_PREFIX@@-adminserver':
```

```
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
  ServerTemplate:
    '@@ENV:RESOURCE_PREFIX@@-cluster-template':
      ListenPort: 8001
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        ListenPort: 8100
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
```

2. Apply the `model.yaml` to the running WebLogic Server domain. See step 6 in Enable SSL Support.

Verify the Authenticator

Verify that the authentication provider is created successfully and the expected LDAP provider users and groups are synced.

Create JMS Resources

Java Messaging Service (JMS) resources can be created using the provided templates.

JMS resources can be added to non-JRF domains. It is recommended to use JDBC persistent stores for JMS and Transaction stores to be in the database. So, we need to create a datasource and corresponding leasing table for JDBC store for the non-JRF domain.

Complete the following steps:

1. Create a table for database leasing:
2. Format of the tablename should be: `<datasourceschemausername>.<tablename>`
   Example:

```
create table mydbuser.mytable
(
SERVER VARCHAR2(255) NOT NULL,
INSTANCE VARCHAR2(255) NOT NULL,
DOMAINNAME VARCHAR2(255) NOT NULL,
CLUSTERNAME VARCHAR2(255) NOT NULL,
TIMEOUT DATE,
PRIMARY KEY (SERVER, DOMAINNAME, CLUSTERNAME)
);
```

3. The tablename has to be specified in the model template yaml for `AutoMigrationTableName`

The sample model template yaml file, `non_jrf_domain_jms_resources.yaml` is located at `/u01/shared/scripts/pipeline/templates`.

These are basic sample template model yaml files that can be used and has to be modified according to your requirement. These are not the final model template.

> **Note:**
>
> Irrespective of the number of managed servers, increase the initial and max capacity of `mydatasource` to a higher value.

For best practices about creating JMS resources, see Best Practices for JMS Beginners and Advanced Users in *Administering JMS Resources for Oracle WebLogic Server*.

Configure SSL Certificate for a Load Balancer

If you add an SSL certificate manually using the Oracle Cloud Infrastructure console to the load balancer created in Oracle WebLogic Server for OKE, when you update the domain using the **update domain** job, the certificate you configured in the console is reverted to the default certificate.

To retain the SSL certificate, you must configure the SSL certificate for the load balancer using the `update_lb_ssl_cert.sh` script located in `/u01/scripts/utils` in the administration instance. After configuring the load balancer, you can verify the SSL certificate configuration. See Verify SSL Certificate Configuration.

> **Note:**
>
> For stacks created prior to 22.3.1 (July, 2022), before you configure SSL certificate for the load balancer, you must create the `update_lb_ssl_cert.sh` script and copy the script in the administration instance. See Script File to Update SSL Certificate for Load Balancer.

Perform the following steps to configure SSL certificate for the load balancer:

1. Access the administration compute instance for your stack.
   See Access the Administration Instance.
2. Store the SSL certificate and private key in the administration instance.
   For example, you can store the SSL certificate and private key in the `/tmp/tls.crt` and `/tmp/tls.key` location in the administration instance.
3. Navigate to `/u01/scripts/utils` directory, and set execute permission to the `update_lb_ssl_cert.sh` file.

   ```
   chmod +x update_lb_ssl_cert.sh
   ```

4. Run the following command:

   ```
   ./update_lb_ssl_cert.sh -d <WebLogic_ Domain_Name> -s
   <Kubernetes_Secret_Name> -k <SSL_Certificate_Key_File> -c
   <SSL_Certificate_File>
   ```

Example:

```
./update_lb_ssl_cert.sh -d domain1 -s lbsslcert -k tls.key -c tls.cert
```

5. After the load balancer is updated with the certificates, delete the `tls.key` and `tls.cert`. For example, if your SSL certificate file is located in `/tmp` directory, run the following command to delete the certificate:

```
rm /tmp/tls.cert
```

Verify SSL Certificate Configuration

Perform the following steps to verify the SSL certificate configuration:

1. Run the following command to verify that the `tls` secret field was updated to the secret name you provided in the script:

```
kubectl describe svc -n wlsoke-ingress-nginx <domain_name>-lb-external |
grep tls-secret
```

Example:

```
kubectl describe svc -n wlsoke-ingress-nginx domain1-lb-external | grep
tls-secret
```

The command output looks like:

```
service.beta.kubernetes.io/oci-load-balancer-tls-secret: lbsslcert
```

2. Access the Oracle Cloud Infrastructure console.
3. From the navigation menu  , click **Networking** and then click **Load Balancers**.
4. Select the **Compartment** that contains your stack.
5. Click the name of the load balancer.
6. Under **Resources**, click **Listeners**. The list of listeners is displayed.
7. Click the **Actions** icon (⋮) associated with the listener set and click **Edit**.
8. Check if the **Certificate Name** is the same as the secret name that you provided in the script in step 4.
9. Run the **update domain** job and repeat steps 3 through 8 to verify if the SSL certification is retained for load balancer.
   See Defining Kubernetes Services of Type LoadBalancer in Oracle Cloud Infrastructure documentation.

Set the JVM Arguments Definition

To explicitly set the Java Virtual Machine (JVM) heap size in the WebLogic Server pods that are created, modify the domain and specify the JVM settings in the domain YAML file.

To set the JVM heap size:

1. Access the administration compute instance for your stack.

   See Access the Administration Instance.

2. Modify the domain using the `kubectl` command.

   ```
   kubectl edit domain <domain_name> -n <domain-namespace> -o yaml
   ```

   This command opens the Domain definition in an editor.

3. Specify the following configuration for the `USER_MEM_ARGS` variable:

   ```
   name: USER_MEM_ARGS
   value: '-Xms256M -Xmx512M -XX:+UseG1GC -Djava.security.egd=file:/dev/./
   urandom
   -Dweblogic.security.SSL.ignoreHostnameVerification=true '
   ```

Session Persistence Considerations

You can configure session persistence when deploying Java EE applications to a WebLogic cluster. You must configure session persistence by updating the `weblogic.xml` deployment descriptor's *`session-descriptor`* element, specifically the *`persistent-store-type`* element, whose default value is *`memory`*.

To edit the `weblogic.xml` deployment descriptor's *`session-descriptor`* element, see session-descriptor.

For applications deployed to a cluster, use a value suitable for clustered applications, for example, *`replicated_if_clustered`*. For information, see Using Sessions and Session Persistence.

For ADF applications, see additional considerations at High Availability Checklist for ADF Applications.

Enabling session affinity or sticky sessions at the ingress controller

To send all client requests of a session to the same Oracle WebLogic Server, you must edit the Kubernetes ingress *wls-cluster-ingress* and add session affinity annotations.

To ensure session affinity annotations work, we also need to specify a host in the ingress *wls-cluster-ingress*. You can edit the kubectl to set session affinity.

Run the following command to view the contents of your *wls-cluster-ingress* file:

```
kubectl get ingress -n <domainname>-ns wls-cluster-ingress -o yaml
```

Sample output:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx-applications
    meta.helm.sh/release-name: ingress-controller
    meta.helm.sh/release-namespace: default
```

```
    nginx.ingress.kubernetes.io/configuration-snippet: |
      more_clear_input_headers "WL-Proxy-Client-IP" "WL-Proxy-SSL";
      more_set_input_headers "X-Forwarded-Proto: https";
      more_set_input_headers "WL-Proxy-SSL: true";
  creationTimestamp: "2020-11-30T20:28:48Z"
  generation: 1
  labels:
    app.kubernetes.io/managed-by: Helm
  name: wls-cluster-ingress
  namespace: <domainname>-ns
  resourceVersion: "2414741"
  selfLink: /apis/extensions/v1beta1/namespaces/<domainname>-ns/ingresses/wls-
cluster-ingress
  uid: f5aa919c-7e93-4ca8-a4ca-ddf791c126dd
spec:
  rules:
  - http:
      paths:
      - backend:
          serviceName: mydomain-test-cluster-myoke-cluster
          servicePort: 8001
        path: /
status:
  loadBalancer:
    ingress:
    - ip: <public_ip>
```

Complete the following steps:

1. Run the following command to enable session affinity:

   ```
   kubectl edit ingress -n <domainname>-ns wls-cluster-ingress
   ```

2. Add the session affinity annotations and the host.

   > **Note:**
   >
   > The minimum set of annotations to add is: `nginx.ingress.kubernetes.io/affinity`

   You can add more annotations to modify the default behavior. For example, you can add `nginx.ingress.kubernetes.io/affinity-mode` and for maximum stickiness set its value to `persistent`, or `nginx.ingress.kubernetes.io/session-cookie-name` to change the default name of the cookie.

   For information about session affinity annotations, see Sticky sessions and Session Affinity.

3. Save the contents.

4. Run the following command to view the contents of your *wls-cluster-ingress* file:

   ```
   kubectl get ingress -n <domainname>-ns wls-cluster-ingress -o yaml
   ```

In the following sample output, the text with comments show the modifications of the ingress. In this case, we enabled session affinity and configured the expiration time for the session cookie:

```
[opc@myoke-admin templates]$ kubectl get ingress -n <domainname>-ns   wls-
cluster-ingress -o yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx-applications
    meta.helm.sh/release-name: ingress-controller
    meta.helm.sh/release-namespace: default
    nginx.ingress.kubernetes.io/affinity: cookie  # New annotation
    nginx.ingress.kubernetes.io/configuration-snippet: |
      more_clear_input_headers "WL-Proxy-Client-IP" "WL-Proxy-SSL";
      more_set_input_headers "X-Forwarded-Proto: https";
      more_set_input_headers "WL-Proxy-SSL: true";
    nginx.ingress.kubernetes.io/session-cookie-expires: "172800"   #New
annotation
    nginx.ingress.kubernetes.io/session-cookie-max-age: "172800"   #New
annotation
  creationTimestamp: "2020-11-30T20:28:48Z"
  generation: 2
  labels:
    app.kubernetes.io/managed-by: Helm
  name: wls-cluster-ingress
  namespace: <domainname>-ns
  resourceVersion: "2419743"
  selfLink: /apis/extensions/v1beta1/namespaces/<domainname>-ns/ingresses/
wls-cluster-ingress
  uid: f5aa919c-7e93-4ca8-a4ca-ddf791c126dd
spec:
  rules:
  - host: my-server   #New host
    http:
      paths:
      - backend:
          serviceName: mydomain-test-cluster-myoke-cluster
          servicePort: 8001
        path: /
status:
  loadBalancer:
    ingress:
    - ip: <public_ip>
```

Note:

If you set the host name in the ingress, you will not be able to access your applications by using the load balancer public IP. To access your applications use the host name you specified in the ingress. That is, you need to add the host name to your DNS servers, or manually map the host name to the public IP of the load balancer. For example, by editing the `/etc/hosts` file.

Monitor a WebLogic Domain

Learn how to monitor a domain by using the tools provided with Oracle WebLogic Server for OKE

Topics:

About the Security Checkup Tool

Oracle WebLogic Server Administration console includes a security checkup tool that displays security check warnings.

In case of Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied, the message `Security warnings detected. Click here to view the report and recommended remedies` is displayed at the top of the Oracle WebLogic Server Administration console. When you click the message, a list of security warnings are displayed as listed in the following table.

The warning messages listed in the table are examples.

Security Warnings

Warning Message	Resolution
`The configuration for key stores for this server are set to Demo Identity and Demo Trust. Trust Demo certificates are not supported in production mode domains.`	Configure the identity and trust keystores for each server and the name of the certificate in the identity keystore that the server uses for SSL communication. See Configure Keystore Attributes for Identity and Trust. **Note:** This warning is displayed for Oracle WebLogic Server for OKE instances created after October 20, 2021, or the instances on which the October PSUs are applied.
`Production mode is enabled but the file or directory <directory_name>/startWebLogic.sh is insecure since its permission is not a minimum of umask 027.`	Run the following command in the administration server as `oracle` user: `chmod 640 /u01/data/domains/<domain_name>/bin`
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Set the java properties for anonymous RMI T3 and IIOP requests during server start up. See Set the Java Properties.

> **Note:**
>
> For existing Oracle WebLogic Server for OKE instances (created before July 20, 2021), you see the SSL host name verification warnings. For details, see Security Checkup Tool Warnings.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OKE instances created after July 20, 2021, though the java properties to disable anonymous requests for preventing anonymous RMI access are configured, the warnings still appear. This is a known issue in Oracle WebLogic Server.

Set the Java Properties

To set the java properties for anonymous RMI T3 and IIOP requests:

1. Edit the `domain.yaml` located in `/u01/shared/weblogic-domains/<domain_name>/domain.yaml` for all instances of `serverPod` definitions as follows:

```
serverPod:
      env:
      - name: USER_MEM_ARGS
        #admin server memory is explicitly set to min of 256m and max of
512m and GC algo is G1GC
        value: "-Xms256m -Xmx512m -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
      - name: JAVA_OPTIONS
        value: "-Dweblogic.store.file.LockEnabled=false
        -Dweblogic.rjvm.allowUnknownHost=true
        -Dweblogic.security.remoteAnonymousRMIT3Enabled=false
        -Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

2. Apply the `domain.yaml` using the `kubectl` command:

```
kubectl -f <path_to_domain.yaml>
```

Configure Keystore Attributes for Identity and Trust

To configure the identity and trust keystore files and the name of the certificate in the identity keystore in the WebLogic Server Administration console:

1. Locate the **Change Center** and click **Lock & Edit** to lock the editable configuration hierarchy for the domain.
2. Under **Domain structure**, select **Environment** and then select **Servers**.
3. In the Servers table, select the server you want to configure.
4. On the **Configuration** tab, click **Keystores**, and then click **Change**.
5. Select *Custom Identity and Custom Trust*, and then click **Save**.
6. Under **Identity**, provide the following details:
   a. Enter the full path of your identity keystore.

      For example: `/u01/data/keystores/identity.jks`

b. For **Custom Identity Keystore Type**, enter *JKS*.

c. For **Custom Identity Keystore Passphrase**, enter your keystore password. Enter the same value for **Confirm Custom Identity Keystore Passphrase**.

7. Under **Trust**, provide the following details:

a. Enter the full path of your identity keystore.

For example, `/u01/data/keystores/trust.jks`

b. For **Custom Trust Keystore Type**, enter *JKS*.

c. For **Custom Trust Keystore Passphrase**, enter your keystore password. Enter the same value for **Confirm Custom Trust Keystore Passphrase**.

8. Click **Save**.

9. Click the **SSL** tab.

10. Under **Identity**, provide the following details:

a. For **Private Key Alias**, enter the name of the certificate (private key) in the identitykeystore, *server_cert*.

b. For **Private Key Passphrase**, enter the password for this certificate in the keystore. Enter the same value for **Confirm Private Key Passphrase**.

By default, the password for the certificate is the same as the identity keystore password.

11. Click **Save**.

After saving the changes, return to **Change Center** and click **Activate Changes**.

12. Repeat steps 3 to 9 to configure each server in the domain.

Component Health Check

Monitor the health of key components of the service in Oracle WebLogic Server for OKE.

The following topic describes how to check the health of such components:

- Check the Health of a Cluster
- Check the Health of a Load Balancer
- Check the Health of a WebLogic Domain

Check the Health of a Cluster

Learn how to view cluster metrics that help you to monitor the health, capacity, and performance of the instance's Kubernetes cluster manged by Oracle WebLogic Server for OKE.

Check the Metrics for Clusters

You can view metrics information for the clusters in your domain.

1. Sign in to the Oracle Cloud Infrastructure Console.

2. Click the navigation menu  and select **Developer Services**. Under the **Containers** group, click **Kubernetes Clusters**.

3. Select the **Compartment** containing the cluster for which you want to view metrics.

4. Click the name of the cluster for which you want to view metrics.
5. Under Resources on the left, click **Metrics**.

   The **Metrics** tab displays a chart for each metric for the cluster that is emitted by the metric namespace. For more information about the displayed metrics, see Available Metrics: oci_oke.

Check the Metrics for Node Pool Clusters

You can view metrics information of the node pools of the instance's Kubernetes cluster and also view the metrics information for each node.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  and select **Developer Services**. Under the **Containers** group, click **Kubernetes Clusters**.
3. Select the **Compartment** containing the cluster for which you want to view metrics.
4. Click the name of the cluster for which you want to view metrics.
5. Under Resources on the left, click **Node pools**.
6. On the **Node Pools** tab, click the name of a node pool for which you want to see detailed status.
7. Under Resources on the left, click **Metrics**.

   This displays more granular information about the health, capacity, and performance of the node pool.
8. Under Resources on the left, click **Nodes**.

   This displays the summary status of each worker node in the node pool
9. Click View Mertics beside the node to view more granular information about the health, capacity, and performance of that node.

   For more information about the displayed metrics, see Available Metrics: oci_oke.

Check the Health of a Load Balancer

Learn how to view the status of a load balancers associated with the instance's Kubernetes cluster managed by Oracle WebLogic Server for OKE.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.
2. Run the following command:

   ```
   kubectl get services --all-namespaces
   ```

   Sample output of services:

   ```
   NAMESPACE                 NAME
   TYPE           CLUSTER-IP      EXTERNAL-IP
   PORT(S)                    AGE
   default                   kubernetes
   ClusterIP      10.96.0.1       <none>
   443/TCP                    27h
   wlsoke-ingress-nginx      okename-internal
   ```

```
LoadBalancer   10.96.185.81    100.121.170.271
80:32144/TCP                27h
jenkins-ns                jenkins-service
ClusterIP      10.96.121.100   <none>            8080/
TCP,50000/TCP           27h
kube-system               kube-dns
ClusterIP      10.96.7.5       <none>            53/UDP,53/
TCP,9153/TCP       27h
kube-system               tiller-deploy
ClusterIP      10.96.76.135    <none>
44134/TCP                   27h
okename-domain-ns         mydomain-cluster-okename-cluster
ClusterIP      10.96.143.98    <none>
8001/TCP                    27h
okename-domain-ns         mydomain-okename-adminserver
ClusterIP      None            <none>            30012/
TCP,7001/TCP           27h
okename-domain-ns         mydomain-okename-managed-server1
ClusterIP      None            <none>
8001/TCP                    27h
okename-domain-ns         mydomain-okename-managed-server2
ClusterIP      None            <none>
8001/TCP                    27h
wlsoke-ingress-nginx      mydomain-lb-external
LoadBalancer   10.96.162.263   144.25.10.101     80:32148/
TCP,443:31808/TCP   27h
okename-operator-ns       internal-weblogic-operator-svc
ClusterIP      10.96.92.254    <none>
8082/TCP                    27h
```

Sample output of services when Verrazzano is enabled:

```
NAMESPACE        NAME                                          TYPE
CLUSTER-IP     EXTERNAL-IP      PORT(S)
AGE
cattle-system    rancher
ClusterIP     10.96.233.56    <none>          80/
TCP,443/TCP                              27h
cattle-system    rancher-webhook
ClusterIP     10.96.46.145    <none>
443/TCP                                     27h
cert-manager     cert-manager
ClusterIP     10.96.17.222    <none>
9402/TCP                                    27h
cert-manager     cert-manager-webhook
ClusterIP     10.96.237.157   <none>
443/TCP                                     27h
default          kubernetes
ClusterIP     10.96.0.1       <none>
443/TCP                                     27h
fleet-system     gitjob
ClusterIP     10.96.195.224   <none>
80/TCP                                      27h
ingress-nginx    ingress-controller-ingress-nginx-controller
LoadBalancer  10.96.220.49    10.0.5.242
```

```
443:32231/TCP                                27h
ingress-nginx    ingress-controller-ingress-nginx-controller-metrics
ClusterIP     10.96.60.166    <none>
10254/TCP                                    27h
ingress-nginx    ingress-controller-ingress-nginx-defaultbackend
ClusterIP     10.96.154.132   <none>
80/TCP                                       27h
istio-system     istio-egressgateway
ClusterIP     10.96.255.249   <none>          80/
TCP,443/TCP                              27h
istio-system     istio-ingressgateway
LoadBalancer  10.96.47.34     129.80.253.214  15021:32050/TCP,80:31611/
TCP,443:30082/TCP 27h
```

Make a note of the IP address of the services with type `LoadBalancer` and in the `EXTERNAL-IP` column.

3. Disconnect from the administration compute instance.
4. Sign in to the Oracle Cloud Infrastructure Console.
5. Click the navigation menu and select **Networking** > **Load Balancers**.
6. Select the **Compartment** that contains your stack.
7. Find the required load balancer by searching with the IP addresses that you noted in Step2.
8. Click the name of the load balance against the IP address you searched for.
9. Under Resources on the left, click **Metrics**.

   The **Metrics** tab displays a default set of charts for the selected load balancer.

Check the Health of a WebLogic Domain

Learn how to view the status of a WebLogic domain manged by Oracle WebLogic Server for OKE.

Each server that is part of the domain runs in a pod. You can run `kubectl` commands to check the status of the pods that are part of the domain.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.
2. Run the following commands:

```
kubectl get pods -n <service>-domain-
ns                                          # list all the pods in the
domain namespace
kubectl describe pod -n <service>-domain-ns <domain>-<service>-
adminserver          # get details of admin server.
kubectl describe pod -n <service>-domain-ns <domain>-<service>-managed-
server1      # get details of managed server 1.
```

Where, <*service*> is the name of the domain and <*domain*> is the domain name.

Output example for command `kubectl get pods -n <service>-domain-ns`:

```
NAME                                READY   STATUS    RESTARTS   AGE
mydomain-nameoke-adminserver        1/1     Running   0          1d4h
mydomain-nameoke-managed-server1    0/1     Running   0          1d4h
mydomain-nameoke-managed-server2    1/1     Running   0          1d4h
```

See the `READY` column to know the status of the respective server.

- `1/1`: servers are running and ready to accept request.
- `0/1`: pod is running, but is not ready to accept request.

Start and Stop Servers

Oracle WebLogic Server for OKE provides utilities to manage the servers in your domain.

With these utilities you can start and stop the admin server and the managed servers in your domain.

> **Note:**
>
> Do not use the WebLogic Server Administration Console to start or stop servers.

1. Identify the following IP address of the node in your domain:
   - The public IP address to the Administration Server node.
   - The public IP address of the bastion and the private IP address of the compute instance.
2. From your computer, run the `ssh` command to connect to the domain's Administration Server node as the `opc` user.

   ```
   ssh -i <path_to_private_key> -o ProxyCommand="ssh -W %h:%p -i
   <path_to_private_key> opc@<bastion_public_ip>" opc@<node_private_ip>
   ```

3. Run the following command:

   ```
   cd /u01/scripts/wls-domain-lifecycle
   ```

   The scripts to run the lifecycle operations on the WebLogic pods are displayed.
4. Run the following commands:

Command	Result
`sh startServer.sh -h` This help command can be used with all the scripts that are available in `wls-domain-lifecycle`.	Displays the help information that includes the command format and the parameters that can be used with the script.
`sh stopServer.sh -s <server name> -n <namespace> -d <domain uid>`	Stops the managed server
`sh startServer.sh -s <server name> -n <namespace> -d <domain uid>`	Starts the managed server

Command	Result
`sh stopCluster.sh -n <namespace> -d <domain uid> -c <cluster name>`	Stops all the managed servers running in your domain
`sh startCluster.sh -n <namespace> -d <domain uid> -c <cluster name>`	Starts all the managed servers running in your domain
`sh stopDomain.sh -n <namespace> -d <domain uid>`	Stops the admin server and the managed servers running in your domain
`sh startDomain.sh -n <namespace> -d <domain uid>`	Starts the admin server and the managed servers running in your domain

For additional information on the scripts, see the `readme` file in *WebLogic Kubernetes Operator* documentation.

There are fields on the Domain that specify which servers should be running, which servers should be stopped, and the desired initial state. You can also modify these fields on the Domain to start and stop servers. See Starting and stopping servers in *WebLogic Kubernetes Operator* documentation.

Scale a WebLogic Cluster

You can change the number of cluster pods or nodes in your Oracle WebLogic Server for OKE stack to increase performance or to reduce costs.

Add pods to scale out, or remove pods to scale in.

1. Access the administration compute instance for your stack.

   See Access the Administration Instance.

2. Use `kubectl` to modify the domain.

   ```
   kubectl edit cluster <domain_name>-cluster -n [namespace]
   ```

   Where, `<domain_name>` is the domain that you want to scale.

   This opens the cluster definition in an editor.

3. Edit the `replicas` value to the desired number of pods.

4. Save and commit the cluster definition.

   You will be notified of the change and the domain immediately scales the corresponding cluster by reconciling the number of running pods with the `replicas` value you specified.

For more information about scaling, see Scaling in *WebLogic Kubernetes Operator* documentation.

Scale the Node Pools

You can change the number of nodes in the node pool to optimize your resource usage.

Add nodes to a node pool to scale out, or remove nodes from a node pool to scale in.

To scale out the node pool:

1. Access the Oracle Cloud Infrastructure Console.

2. From the navigation menu, select **Developer Services**. Under **Containers & Artifacts**, click **Kubernetes Cluster (OKE)**.
3. From the **Compartment** dropdown, select the compartment where your cluster is located.
4. Click the name of the cluster you want to modify.
5. Under **Resources**, click **Node Pools**, and then click the node pool that you want to scale out.
6. On the Node Pool details page, click **Scale** and edit the number of the nodes in the node pool.
7. Click **Scale** to save the changes.

To scale in the node pool:

1. Access the Oracle Cloud Infrastructure Console.
2. From the navigation menu, select **Developer Services**. Under **Containers & Artifacts**, click **Kubernetes Cluster (OKE)**.
3. From the **Compartment** dropdown, select the compartment where your cluster is located.
4. From the list of clusters, click the name of the cluster you want to modify.
5. Under **Resources**, click **Node Pools**.
6. From the list of node pools, click the name of the node pool.
7. Under **Resources**, click **Nodes**.
8. Click the arrow on the right, and under **Actions**, click **Delete Node**. In the Confirm page, do the following:
   a. Click **Show advanced options**.
   b. Under **Cordon and drain**, before terminating the node, specify the time to allow to cordon and drain the node.
   c. Select the **Force terminate after grace period** check box, if you want to terminate the node at the end of the eviction grace period.
   If this check box is selected, the nodes are terminated even if they have not been successfully cordoned and drained.
9. Repeat steps 2 to 8 for each node that you want to remove.

Update the Repository Schema Utility Password using Secrets

If you modified the Repository Schema Utility (RCU) password, then you must update the schema password in the domain.

During initial provisioning, we create a secret named `<resource_prefix>-rcu-access`, which contains all the RCU related information, like `db_connect_String`, schema prefix, and schema password.

Complete the following steps to update the schema password in the domain:

1. Shutdown the domain.

   Run the following command:

   ```
   kubectl edit domain -n <domain_ns> -o yaml
   ```

Sample output:

```
kind: Domain
  metadata:
    name: domain1
  spec:
    serverStartPolicy: "NEVER"
```

Change the `serverStartPolicy` value, from `IF_NEEDED` to `NEVER`. See Starting and stopping servers.

2. If you have not changed the RCU schema password on the database, then complete this step.

   a. In the administration server, run the `rcu_secret.sh` script, which is located at `/u01/scripts/pipeline/helper-scripts`. This displays the existing `schemaPrefix` information.

   b. Connect to the database as `sysdba` user using `sqlplus`.

   ```
   connect sys/<password>@//dbhost.subnet1.vcn1.oraclevcn.com:1521/
   pdbName.subnet1.vcn1.oraclevcn.com as sysdba
   ```

   Example:

   ```
   connect sys/<password>@//sidb19-
   scan.admin.existingnetwork.oraclevcn.com:1521/
   sipdb.admin.existingnetwork.oraclevcn.com as sysdba
   ```

   c. Set the login attempts for the `DEFAULT` profile to `UNLIMITED` and then check the limit for the `DEFAULT` profile.

   ```
   ALTER PROFILE DEFAULT LIMIT FAILED_LOGIN_ATTEMPTS UNLIMITED;
   select limit from dba_profiles where profile='DEFAULT' /
   and resource_name='FAILED_LOGIN_ATTEMPTS';
   LIMIT
   ----------------------------------------------------------------------
   UNLIMITED
   ```

   d. List all the dba users where username like `<schema_prefix>__%`.

   ```
   select username from dba_users where username like '<schema_prefix>__%';



   USERNAME
   ------------------------------
   <schema_prefix>_STB
   <schema_prefix>_IAU_APPEND
   <schema_prefix>_OPSS
   <schema_prefix>_WLS
   <schema_prefix>_IAU
   <schema_prefix>_WLS_RUNTIME
   <schema_prefix>_IAU_VIEWER
   ```

```
<schema_prefix>_UMS
<schema_prefix>_MDS
```

For example:

```
select username from dba_users where username like 'SP1601029287_%';
```

All user names are prefixed with `SP1601029287` as in `SP1601029287_STB`.

e. Change the password for the following users:

```
alter user <schema_prefix>_STB identified by <new_password>;
alter user <schema_prefix>_IAU_APPEND identified by <new_password>;
alter user <schema_prefix>_OPSS identified by <new_password>;
alter user <schema_prefix>_WLS identified by <new_password>;
alter user <schema_prefix>_IAU identified by <new_password>;
alter user <schema_prefix>_WLS_RUNTIME identified by <new_password>;
alter user <schema_prefix>_IAU_VIEWER identified by <new_password>;
alter user <schema_prefix>_UMS identified by <new_password>;
alter user <schema_prefix>_MDS identified by <new_password>;
```

For example:

```
alter user SP1601029287_STB identified by <new_password>;
```

f. Connect to the database for the MDS user.

```
connect <schema_prefix>_MDS/<password>//dbhost.example.com:1521
Connected
```

g. List the table names in the database for the current user.

```
select table_name from user_tables;
```

h. Exit SQL.

```
exit
```

3. Delete the existing kubernetes secret: `<resource_prefix>-rcu-access`
4. Run the following command to obtain the name of the secret:

```
kubectl get secrets -n <domain_namespace> |grep rcu-access
```

For example, run `kubectl get secrets -n domain10-ns |grep rcu-access`. The name of the secret obtained is `<resource_prefix>-rcu-access>`

5. Recreate the secret with the same name. The name that you obtained in the previous step.

The `<resource_prefix>-rcu-access` secret has certain fields other than the schema password, which also needs to be specified based on the database type. When you run the `rcu_secret.sh` script, located at `/u01/scripts/pipeline/helper-scripts/`, it outputs all the other required fields in addition to the schema password for recreating the secret.

For ATP database:

```
[opc@wrjrf1-admin helper-scripts]$./rcu_secret.sh
rcu_db_name = <atp_db_name_low>
rcu_prefix = <prefix>
rcu_wallet_password = <password>
[opc@wrjrf1-admin helper-scripts]$kubectl create secret generic -n
<domain_ns> '<resource_prefix>-rcu-access' --from-
literal=rcu_db_name=<atp_db_name_low> --from-literal=rcu_prefix=<prefix> --
from-literal=rcu_wallet_password=<password> --from-
literal=rcu_schema_password=<new_password>
```

For OCI, SI, or RAC database:

```
[opc@wrjrf1-admin helper-scripts]$./rcu_secret.sh
rcu_admin_password = <admin_password>
rcu_db_conn_string = <connect_String>
rcu_db_user = sys
rcu_prefix = <prefix>
[opc@wrjrf1-admin helper-scripts]$kubectl create secret generic -n
<domain_ns> '<resource_prefix>-rcu-access' --from-
literal=rcu_admin_password=<admin_password> --from-
literal=rcu_db_conn_string=name-
scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:1521/
db0409_pdb1.subnet2ad2phx.paasdevjcsphx.oraclevcn.com --from-
literal=rcu_db_user=sys --from-literal=rcu_prefix=<prefix> --from-
literal=rcu_schema_password=<new_password>
```

6. Change the `serverStartPolicy` value, from `NEVER` to `IF_NEEDED`, and then increment the `restartVersion`.

```
kind: Domain
  metadata:
    name: domain1
  spec:
    serverStartPolicy: "IF_NEEDED"
    restartVersion: "3"
```

7. Wait for the domain to start for a rolling restart. Then, verify that the datasource `mds-owsm` tests okay in the WebLogic Server administration console.

Update the Oracle Cloud Infrastructure Registry Auth Token Credentials

If you update the registry user name and Oracle Cloud Infrastructure Registry (OCIR) auth token credentials for your Oracle WebLogic Server for OKE stack, you must remove the existing OCIR secrets, recreate the secrets, and update the registry user name.

1. Create a `config.json` file with the updated OCIR token password.

   ```
   {"auths": {"phx.ocir.io": {"Username": "tenancy_object_storage_namespace/
   ociruser", "Password": "XXX"}}}
   ```

   Where `phx.ocir.io` is the container registry and `Password` is the password in clear text used for docker login to container registry.

   You can get the user name using the following command:

   ```
   curl -H "Authorization:Bearer Oracle" -L http://169.254.169.254/opc/v2/
   instance/metadata | grep ocir_user
   ```

2. List the secrets created with previous auth token.

   ```
   kubectl get secrets -A | grep ocirsecrets
   ```

3. Remove all OCIR secrets obtained in step 2.

   Example commands to remove the following OCIR secrets:

   ```
   kubectl delete secret ocirsecrets -n jenkins-ns
   kubectl delete secret ocirsecrets -n <domain_name>-ns
   kubectl delete secret ocirsecrets -n <service_prefix_name>-operator-ns
   kubectl delete secret ocirsecrets -n wlsoke-ingress-nginx
   ```

   > **Note:**
   >
   > If you have created multiple domains, you must delete the secrets for all domains.

4. Recreate the OCIR secrets you deleted in step 3 for each of the namespaces.

   Example commands to recreate OCIR secrets:

   ```
   kubectl create secret generic ocirsecrets -n jenkins-ns --from-
   file=.dockerconfigjson=config.json --type=kubernetes.io/dockerconfigjson
   kubectl create secret generic ocirsecrets -n <domain_name>-ns --from-
   file=.dockerconfigjson=config.json --type=kubernetes.io/dockerconfigjson
   kubectl create secret generic ocirsecrets -n <service_prefix_name>-
   operator-ns --from-file=.dockerconfigjson=config.json --type=kubernetes.io/
   dockerconfigjson
   kubectl create secret generic ocirsecrets -n wlsoke-ingress-nginx --from-
   file=.dockerconfigjson=config.json --type=kubernetes.io/dockerconfigjson
   ```

   > **Note:**
   >
   > If you have created multiple domains, you must recreate the secrets for all domains.

5. In `wlsoke-metadata-configmap`, update ocir_user.

```
kubectl edit configmap wlsoke-metadata-configmap -n jenkins-ns
kubectl delete secret ocirtokensecret -n jenkins-ns
kubectl create secret generic ocirtokensecret -n jenkins-ns --from-
literal=username="tenancy_object_storage_namespace/ociruser" --from-
literal=password="XXX"
```

Where, `Password` is the password in clear text used for docker login to container registry.

Upgrade the Kubernetes Version

Learn how to upgrade the Kubernetes version of the cluster and the node pools in your Oracle WebLogic Server for OKE stack.

> **Note:**
>
> You can upgrade the cluster and node pool only for Oracle WebLogic Server for OKE instances created from June 2022 (release 22.2.3).

For an Oracle WebLogic Server for OKE stack created with an older Kubernetes version, Oracle recommends you upgrade the Kubernetes cluster and the node pool to new Kubernetes version that is supported by Oracle Container Engine for Kubernetes. The current Kubernetes version supported for Oracle WebLogic Server for OKE is 1.28.2. To know the Kubernetes versions, see Currently Supported Kubernetes Versions in the Oracle Cloud Infrastructure documentation.

If your Kubernetes cluster and the node pool is three minor versions behind the current supported version, you must upgrade the cluster to at least one minor version higher than the existing cluster version. However, you can either continue to use the node pool running the older version of Kubernetes or upgrade the node pool version also to the Kubernetes cluster version.

> **Note:**
>
> The older node pool version must be compatible with the Kubernetes cluster version.

> **Note:**
>
> If your Kubernetes cluster and node pool version is 1.21 or lower, after you upgrade the cluster and node pool to version 1.28, you must upgrade the nginx-ingress image to 1.3.1 version and then reinstall the ingress-controllers. See Upgrade the NGINX Image Version.

Topics:

- Determine the Version of the Cluster and Node Pools
- Upgrade Cluster and Node Pool Using Script

- Upgrade the NGINX Image Version

Determine the Version of the Cluster and Node Pools

You must identify versions of the Kubernetes cluster and the node pools before performing the upgrade.

1. Access the Oracle Cloud Infrastructure Console.
2. From the navigation menu, select **Developer Services**. Under **Containers & Artifacts**, click **Kubernetes Cluster (OKE)**.
3. From the **Compartment** dropdown, select the compartment where your cluster is located.
4. From the list of clusters, click the name of the cluster you want to modify.
5. In the **Cluster Details** tab, the **Kubernetes version** of the cluster is displayed and information, if upgrade is available.
6. Under **Resources**, click **Node pools**.
7. From the list of node pools, click the name of the node pool.
8. In the **Node pool details** tab, the Kubernetes version of the node pool is displayed and information, if upgrade is available.
9. Navigate to the Cluster details page, and repeat step 6 through step 8 for each node pool in the cluster.

Upgrade Cluster and Node Pool Using Script

You can use the `upgrade_cluster.py` script to upgrade the Kubernetes version of the cluster and the node pools in your Oracle WebLogic Server for OKE stack.

The script upgrades the cluster and the node pool to the specified target version.

After you upgrade the cluster to 1.24, it is recommended to upgrade the `kubectl` version to 1.23.

Prerequisites:

Complete the following steps before you run the upgrade script:

1. Access the administration instance. See Access the Administration Instance.
2. Install OCI Python SDK on the administration instance.

> **Note:**
>
> To install OCI Python SDK, a NAT gateway must be configured for the administration instance private subnet.

```
sudo bash
python3 -m pip install oci==2.90
```

If the `pip` command fails, then use the `yum` command to install OCI Python SDK.

```
yum install python36-oci-sdk
```

3. In the OCI Console, delete the nodes for each node pool in the cluster.
   a. Access the Oracle Cloud Infrastructure Console.
   b. From the navigation menu, select **Developer Services**. Under **Containers & Artifacts**, click **Kubernetes Cluster (OKE)**.
   c. From the **Compartment** dropdown, select the compartment where your cluster is located.
   d. From the list of clusters, click the name of the cluster you want to modify.
   e. Under **Resources**, click **Node Pools**.
   f. From the list of node pools, click the name of the node pool.
   g. Under **Resources**, click **Nodes**.
   h. Make a note of the node count for the node pool.
   You have to create the same number of nodes that you have noted, after upgrading the cluster and node pool.
   i. Click the arrow on the right, and under **Actions**, click **Delete Node**. In the Confirm page, click **Delete**.

   See Delete a worker node in the Oracle Cloud Infrastructure documentation.

Upgrade the cluster and node pool using the script

You can use the script to upgrade the cluster, the node pool, or both cluster and node pool.

1. Access the administration instance. See Access the Administration Instance.
2. Go to `/u01/scripts/utils` location and create a file, `upgrade_cluster.py`.
3. Copy and paste the script specified in Script file for Upgrade Cluster and Node Pool to the `upgrade_cluster.py` file.
4. Run the following commands:

> **Note:**
>
> The current Kubernetes version supported for Oracle WebLogic Server for OKE is 1.28.x. To know the Kubernetes versions, see Currently Supported Kubernetes Versions in the Oracle Cloud Infrastructure documentation.
>
> When you are running the script to upgrade the cluster and node pool, you will see a downtime for the domain pods and Jenkins.

- To upgrade the cluster and node pool:

```
python3 /u01/scripts/utils/upgrade_cluster.py <target_k8s_version>
```

- To upgrade only the cluster:

```
python3 /u01/scripts/utils/upgrade_cluster.py <target_k8s_version>
cluster
```

- To upgrade only the node pool:

  ```
  python3 /u01/scripts/utils/upgrade_cluster.py <target_k8s_version>
  nodepool
  ```

5. In the OCI Console, add the nodes for each node pool in the cluster. See step 3h to know the number of nodes to be added.
   a. Access the Oracle Cloud Infrastructure Console.
   b. From the navigation menu, select **Developer Services**. Under **Containers & Artifacts**, click **Kubernetes Cluster (OKE)**.
   c. From the **Compartment** dropdown, select the compartment where your cluster is located.
   d. From the list of clusters, click the name of the cluster you want to modify.
   e. Under **Resources**, click **Node Pools**, and then click the node pool that you want to scale out.
   f. On the Node Pool details page, click **Scale** and edit the number of the nodes in the node pool.
   g. Click **Scale** to save the changes.
6. Wait for the Node State of the nodes in the node pool to change to *Active*.
7. Verify if the domain pods are up and running for each namespace.

   ```
   kubectl get po -n <domain_name>-ns
   ```

   If you are reusing the namespaces, use the following command to verify if the domain pods are up and running:

   ```
   kubectl get po -n <namespace>
   ```

8. Verify if Jenkins is up and running by accessing the Jenkins console. See Access the Jenkins Console.

Upgrade the NGINX Image Version

Upgrade the nginx-ingress image to 1.3.1 version and reinstall the ingress-controllers.

1. From your computer, run the SSH command to connect to the domain's Administration Server node as the opc user.

   ```
   ssh -i <path_to_private_key> opc@<node_IP_address>
   ```

2. List the existing domains ingress and jenkins ingress, and delete the existing ingress.

   ```
   #List all the existing domains ingress and jenkins ingress
   kubectl get ing -A
   #This will list the existing domains ingress and jenkins ingress

   #Delete all the existing domain and jenkins ingress
   kubectl delete ing wls-admin-ingress -n nonjrf-ns
   kubectl delete ing wls-cluster-ingress -n nonjrf-ns
   ```

```
kubectl delete ing wls-console-help-ingress -n nonjrf-ns
kubectl delete ing jenkins-dashboard-ingress -n jenkins-ns
```

3. List the existing deployments, and delete the old internal and external deployments.

```
#List all the existing deployments in wlsoke-ingress-nginx namespace
kubectl get deploy -n wlsoke-ingress-nginx
Example output:
wlsoke-ingress-nginx    nginx-ingress-controller
0/1      1      0    2h
wlsoke-ingress-nginx    nginx-ingress-controller-nonjrf-external
0/1      1      0    2h

#Delete all the existing deployments for wlsoke-ingress-nginx namespace
kubectl delete deployment.apps/nginx-ingress-controller -n wlsoke-ingress-
nginx
kubectl delete deployment.apps/nginx-ingress-controller-nonjrf-external -n
wlsoke-ingress-nginx
```

4. List the existing services, delete the services for load balancer and Jenkins.

```
#List all the existing services in wlsoke-ingress-nginx namespace
kubectl get svc -n  wlsoke-ingress-nginx
Example output:
wlsoke-ingress-nginx   nonjrf-lb-external   LoadBalancer   10.96.26.2
152.0.0.1   443:30118/TCP   2h
wlsoke-ingress-nginx   wlsokeprefix-internal      LoadBalancer
10.96.20.239  10.0.0.1      80:32686/TCP    2h

#Delete all the existing services
kubectl delete svc nonjrf-lb-external -n wlsoke-ingress-nginx
kubectl delete svc wlsokeprefix-internal -n wlsoke-ingress-nginx

#Delete the jenkins service
kubectl delete svc jenkins-service -n jenkins-ns
```

5. Uninstall ingress controller helm chart and Jenkins charts.

```
helm uninstall ingress-controller
helm uninstall jenkins-oke
```

6. Copy files to the `tmp` directory

```
cp /u01/provisioning-data/*.yaml /tmp
```

7. Replace the files, `_nginx-ingress.tpl` and `_nginx-role.tpl` in `/u01/scripts/ingress-controller`.

   To view the contents of the files, `_nginx-ingress.tpl` and `_nginx-role.tpl` located in `/u01/scripts/ingress-controller`, see NGINX Ingress Template File and NGINX Role Template File.

8. Replace the file `_nginx-ingress.tpl` in `/u01/shared/scripts/pipeline/create_domain/ingress-controller`.

To view the contents of the file `_nginx-ingress.tpl` located in `/u01/shared/scripts/pipeline/create_domain/ingress-controller`, see NGINX Ingress Template File.

9. Replace the file `jenkins-role.tpl` in `/u01/scripts/jenkins/charts/templates`.

   To view the contents of the file, `jenkins-role.tpl` located in `/u01/scripts/jenkins/charts/templates`, see Jenkins Role Template File.

10. Pull the ingress controller image 1.3.1.

```
#https://newreleases.io/project/github/kubernetes/ingress-nginx/release/
controller-v1.3.1
docker pull registry.k8s.io/ingress-nginx/
controller:v1.3.1@sha256:54f7fe2c6c5a9db9a0ebf1131797109bb7a4d91f56b9b362bd
e2abd237dd1974
```

11. Tag the image.

```
#Tag the image corresponding to your region and service name
docker tag <image_identifier> <region_key>.ocir.io/<tenancy_namespace>/
<repository_name>:<tag>
```

Example:

```
docker tag 8e6k1kebb869 phx.ocir.io/mytenancy/wlsokeprefix/infra/nginx-
ingress-controller:1.3.1
```

12. Push the tagged image.

```
docker push <region_key>.ocir.io/<tenancy_namespace>/
<repository_name>:<tag>
```

Example:

```
docker push phx.ocir.io/mytenancy/wlsokeprefix/infra/nginx-ingress-
controller:1.3.1
```

13. Update the value for `ocir_ingress_image_tag` in `/tmp/ingress-controller-input-values.yaml` to point to the new image.

    Example of an `ingress-controller-input-values.yaml` file in `tmp` directory:

```
{
   "jenkins_service": "jenkins-service",
   "jenkins_service_port": "8080",
   "ocir_ingress_image_tag":"iad.ocir.io/mytenancy/wlsokeprefix/infra/
nginx-ingress-controller:1.3.1",
   "ingress_namespace":"wlsoke-ingress-nginx",
   "ingress_ocir_secret_name":"ocirsecrets",
   "ingress_lb_service_name":"wlsokeprefix-internal",
   "ingress_lb_shape":"flexible",
   "ingress_lb_shape_min":"10",
   "ingress_lb_shape_max":"100",
   "ingress_enable_http_port":true,
   "ingress_enable_https_port":false,
   "ingress_http_port":"80",
```

```
   "ingress_https_port":"443",
   "jenkins_namespace":"jenkins-ns",
   "cert_secret_name": "oke-ssl-secret"
 }
```

14. Install ingress controller again using the following command:

```
helm upgrade --install ingress-controller /u01/scripts/ingress-controller
--values /tmp/ingress-controller-input-values.yaml -n default --wait
```

15. Install the Jenkins charts.

```
#get the container image for jenkins controller
docker images |grep jenkins-controller
#Example output:
phx.ocir.io/mytenancy/wlsokeprefix/infra/cisystem-jenkins-controller
1.0.5_2.235.11    369269139728    10 months ago    2.16GB

helm upgrade --install jenkins-oke /u01/scripts/jenkins/charts/ --set
image.repository=phx.ocir.io/mytenancy/wlsokeprefix/infra/cisystem-jenkins-
controller:1.0.5_2.235.11
--values /tmp/jenkins-inputs.yaml -n default --wait
```

16. Add the value for `ocir_ingress_image_tag` in `/u01/shared/weblogic-domains/domain1/ingress-controller-inputs.yaml` to point to the new image.

    Example of an `ingress-controller-input-values.yaml` file in `/u01/shared/weblogic-domains/<domain_name>` directory:

```
{
   "domain_name": "domain1",
   "admin_service": "domain1-domain1-adminserver",
   "admin_service_port": "8765",
   "cluster_service": "domain1-cluster-domain1-cluster",
   "cluster_service_port": "9765",
   "ingress_namespace":"wlsoke-ingress-nginx",
   "ingress_ocir_secret_name":"ocirsecrets",
   "wls_domain_namespace":"domain1-ns",
   "cert_secret_name": "oke-ssl-secret",
   "lb_name": "domain1-lb-external",
   "lb_namespace": "domain1-ns",
   "lb_shape": "flexible",
   "lb_shape_min": "10",
   "lb_shape_max": "40",
   "service_ssl_port": "443",
   "is_idcs_selected": false,
   "is_private_lb": false,
   "reserved_public_ip": "",
   "ocir_ingress_image_tag": "iad.ocir.io/mytenancy/wlsokeprefix/infra/
nginx-ingress-controller:1.3.1"
 }
```

17. Install ingress controller for all the existing domains.

```
#Run the following command for each existing domain by replacing the
domain name in location /u01/shared/weblogic-domains/<domain_name>/ingress-
```

```
controller-inputs.yaml
helm upgrade --install ingress-controller /u01/shared/scripts/pipeline/
create_domain/ingress-controller
--values /u01/shared/weblogic-domains/domain1/ingress-controller-
inputs.yaml -n default --wait
```

18. Edit the configmap to change the kubernetes version and new ingress controller image value.

```
#edit the cm wlsoke-metadata-configmap in jenkins namespace for the
attributes, kubernetes_version and
ocir_ingress_controller_repokubernetes_version
kubectl edit cm wlsoke-metadata-configmap -n jenkins-ns -o yaml
kubernetes_version: v1.24
ocir_ingress_controller_repo: phx.ocir.io/mytenancy/wlsokeprefix/infra/
nginx-ingress-controller:1.3.1
#save the configmap
```

Template Files

This section lists all the template files required when you upgrade the Kubernetes version of your cluster to 1.24 and your node pools to 1.24.

Topics:

- NGINX Ingress Template File located in `/u01/scripts/ingress-controller` directory
- NGINX Ingress Template File located in `/u01/shared/scripts/pipeline/create_domain/ingress-controller` directory
- NGINX Role Template File
- Jenkins Role Template File

NGINX Ingress Template File

The contents of the NGINX ingress template file, `_nginx-ingress.tpl` located in `/u01/scripts/ingress-controller` directory is provided below.

```
#
# Copyright (c) 2020, 2022, Oracle and/or its affiliates. All rights reserved.
#

{{- define "nginx.ingress" }}
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jenkins-dashboard-ingress
  namespace: {{ .jenkins_namespace }}
  annotations:
    helm.sh/resource-policy: keep
    kubernetes.io/ingress.class: "nginx"
spec:
#  tls:
#    - secretName: {{ .cert_secret_name }}
  rules:
```

```
  - http:
      paths:
      - path: /jenkins
        pathType: Prefix
        backend:
          service:
            name: {{ .jenkins_service }}
            port:
              number: {{ .jenkins_service_port }}
{{- end }}
```

NGINX Ingress Template File

The contents of the NGINX ingress template file, `_nginx-ingress.tpl` located in `/u01/shared/scripts/pipeline/create_domain/ingress-controller` directory is provided below.

```
#
# Copyright (c) 2021, 2022, Oracle and/or its affiliates. All rights reserved.
#

{{- define "nginx.ingress" }}
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wls-admin-ingress
  namespace: {{ .wls_domain_namespace }}
  annotations:
    helm.sh/resource-policy: keep
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - http:
      paths:
      - path: {{ "/" }}{{ .domain_name }}{{ "/console" }}
        pathType: Prefix
        backend:
          service:
            name: {{ .admin_service }}
            port:
              number: {{ .admin_service_port }}
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wls-console-help-ingress
  namespace: {{ .wls_domain_namespace }}
  annotations:
    helm.sh/resource-policy: keep
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - http:
      paths:
      - path: {{ "/" }}{{ .domain_name }}{{ "/consolehelp" }}
```

```
        pathType: Prefix
        backend:
          service:
            name: {{ .admin_service }}
            port:
              number: {{ .admin_service_port }}
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wls-cluster-ingress
  namespace: {{ .lb_namespace }}
  annotations:
    helm.sh/resource-policy: keep
    kubernetes.io/ingress.class: "{{ .domain_name }}-nginx-applications"
    nginx.ingress.kubernetes.io/configuration-snippet: |
      more_clear_input_headers "WL-Proxy-Client-IP" "WL-Proxy-SSL";
      more_set_input_headers "X-Forwarded-Proto: https";
      more_set_input_headers "WL-Proxy-SSL: true";
      more_set_input_headers "is_ssl:ssl";
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/session-cookie-name: "JSESSIONID"
spec:
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          {{ if .is_idcs_selected }}
            service:
              name: {{ .domain_name }}-idcsappgateway-service
              port:
                number: 80
          {{ else }}
            service:
              name: {{ .cluster_service }}
              port:
                number: {{ .cluster_service_port }}
          {{ end }}
{{- end }}
```

NGINX Role Template File

The contents of the NGINX role template file, `_nginx-role.tpl` is provided below.

```
#
# Copyright (c) 2020, 2022, Oracle and/or its affiliates. All rights reserved.
#

{{- define "nginx.role" }}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
```

```
  name: nginx-ingress-clusterrole
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
  annotations:
    helm.sh/resource-policy: keep
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
      - endpoints
      - nodes
      - pods
      - secrets
    verbs:
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - services
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
      - patch
  - apiGroups:
      - "extensions"
      - "networking.k8s.io"
    resources:
      - ingresses
      - ingressclasses
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - "extensions"
      - "networking.k8s.io"
    resources:
      - ingresses/status
    verbs:
      - update
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: nginx-ingress-role
  namespace: {{ .ingress_namespace }}
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
  annotations:
    helm.sh/resource-policy: keep
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
      - pods
      - secrets
      - namespaces
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - configmaps
    resourceNames:
      # Defaults to "<election-id>-<ingress-class>"
      # Here: "<ingress-controller-leader>-<nginx>"
      # This has to be adapted if you change either parameter
      # when launching the nginx-ingress-controller.
      - "ingress-controller-leader-nginx"
    verbs:
      - get
      - update
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - create
      - update
      - delete
  - apiGroups:
      - ""
    resources:
      - endpoints
    verbs:
      - get
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: nginx-ingress-role-nisa-binding
  namespace: {{ .ingress_namespace }}
  labels:
    app.kubernetes.io/name: ingress-nginx
```

```
    app.kubernetes.io/part-of: ingress-nginx
  annotations:
    helm.sh/resource-policy: keep
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: nginx-ingress-role
subjects:
  - kind: ServiceAccount
    name: nginx-ingress-serviceaccount
    namespace: {{ .ingress_namespace }}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: nginx-ingress-clusterrole-nisa-binding
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
  annotations:
    helm.sh/resource-policy: keep
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: nginx-ingress-clusterrole
subjects:
  - kind: ServiceAccount
    name: nginx-ingress-serviceaccount
    namespace: {{ .ingress_namespace }}
  - kind: ServiceAccount
    name: provisioning-sa
    namespace: {{ .ingress_namespace }}
{{- end }}
```

Jenkins Role Template File

The contents of the Jenkins role template file, `jenkins-role.tpl` is provided below.

```
#
# Copyright (c) 2020, 2022, Oracle and/or its affiliates. All rights reserved.
#
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: jenkins-clusterrole
rules:
  - apiGroups:
      - ""
    resources:
      - endpoints
      - nodes
      - pods
      - services
      - secrets
      - namespaces
```

```
    - deployments
    - ingresses
    - persistentvolumes
    - persistentvolumeclaims
    - serviceaccounts
    - configmaps
    - events
    - pods
    - pods/log
    - pods/exec
  verbs:
    - list
    - watch
    - get
    - create
    - delete
    - update
    - patch
    - deletecollection
- apiGroups:
    - apps
  resources:
    - endpoints
    - nodes
    - pods
    - services
    - secrets
    - deployments
    - ingresses
    - replicasets
  verbs:
    - list
    - watch
    - get
    - create
    - delete
    - update
    - patch
- apiGroups:
    - "weblogic.oracle"
  resources:
    - domains
  verbs:
    - get
    - create
    - list
    - watch
    - delete
    - patch
- apiGroups:
    - extensions
  resources:
    - ingresses
  verbs:
    - get
    - create
```

```
    - list
    - watch
    - delete
    - patch
- apiGroups:
    - rbac.authorization.k8s.io
  resources:
    - rolebindings
    - clusterroles
    - clusterrolebindings
    - roles
  verbs:
    - get
    - list
    - watch
    - update
    - create
    - patch
    - delete
- apiGroups:
    - batch
  resources:
    - jobs
  verbs:
    - list
    - watch
    - get
    - create
    - delete
    - update
    - patch
    - deletecollection
- apiGroups:
    - core.oam.dev
  resources:
    - components
    - applicationconfigurations
    - nodes
    - pods
    - services
    - secrets
    - namespaces
    - deployments
    - ingresses
    - persistentvolumes
    - persistentvolumeclaims
    - configmaps
    - events
    - pods
    - pods/log
    - pods/exec
  verbs:
    - list
    - watch
    - get
    - create
```

```
      - delete
      - update
      - patch
      - deletecollection
  - apiGroups:
      - networking.istio.io
    resources:
      - gateways
    verbs:
      - list
      - watch
      - get
      - update
      - delete
      - patch
      - create
  - apiGroups:
      - networking.k8s.io
    resources:
      - ingresses
    verbs:
      - get
      - create
      - delete
      - update
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: jenkins-role
  namespace: {{ .Values.service.namespace }}
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
      - pods
      - pods/log
      - pods/exec
      - secrets
    verbs:
      - create
      - update
      - patch
      - list
      - watch
      - get
      - delete
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: jenkins-rolebinding
  namespace: {{ .Values.service.namespace }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
  kind: Role
  name: jenkins-role
subjects:
  - kind: ServiceAccount
    name: jenkins-serviceaccount
    namespace: {{ .Values.service.namespace }}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-rolebinding
  namespace: {{ .Values.service.namespace }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: jenkins-role
subjects:
  - kind: ServiceAccount
    name: default
    namespace: {{ .Values.service.namespace }}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: default-clusterrole-binding
  namespace: {{ .Values.service.namespace }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: jenkins-clusterrole
subjects:
  - kind: ServiceAccount
    name: default
    namespace: {{ .Values.service.namespace }}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: jenkins-clusterrole-binding
  namespace: {{ .Values.service.namespace }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: jenkins-clusterrole
subjects:
  - kind: ServiceAccount
    name: jenkins-serviceaccount
    namespace: {{ .Values.service.namespace }}
```

Upgrade the WebLogic Kubernetes Operator

Learn how to upgrade the WebLogic Kubernetes operator version for an Oracle WebLogic Server for OKE domain, if your domain uses an older operator version, or if your Kubernetes cluster version is not compatible with the WebLogic Server Kubernetes Operator version.

You can use Cloud Shell or the administration host to perform the upgrade. However, some of the upgrade steps must be performed on the administration host only.

If the subnet for the administration host does not have a NAT gateway configured, you must use Cloud Shell to perform the upgrade.

Before you perform the upgrade:

- Access the Oracle Cloud Infrastructure console and open Cloud Shell.
- Connect to the Administration Server node as the `opc` user.
  The SSH command format is:

  ```
  ssh -i path_to_private_key opc@admin_ip
  ```

  To SSH using Windows, see *To connect to a Linux instance from a Windows system using PuTTY* in Connecting to Your Linux Instance Using SSH in Oracle Cloud Infrastructure documentation.

Topics:

Upgrade the WebLogic Kubernetes Operator to 3.4.4

Perform the following steps to upgrade the operator on the administration host:

1. Download the operator image.

   ```
   docker pull ghcr.io/oracle/weblogic-kubernetes-operator:3.4.4
   ```

   > **Note:**
   >
   > If the subnet for the admin host does not have a NAT gateway configured, you must use Cloud Shell to download the operator image.

2. Run the following command, and from the command output, make a note of the docker image ID and repository name that is tagged to *3.4.4* version:

   ```
   docker images | grep weblogic-kubernetes-operator
   ```

   Sample output:

   ```
   REPOSITORY
   TAG           IMAGE ID       CREATED      SIZE
   ```

```
ghcr.io./oracle/weblogic-kubernetes-operator
3.4.2         g568900tt73    1 week ago   222MB
phx.ocir.io/<tenancy_name>/<stack-name>weblogic-kubernetes-operator
3.4.4         568900gtt73    3 days ago   231MB
```

In case of Cloud Shell, use the following command:

```
docker images | grep weblogic-kubernetes-operator | grep 3.4.4
```

3. Tag the downloaded docker image.

```
docker tag <image_ID> <region_name>.ocir.io/<tenancy_name>/
<resource_prefix>/infra/weblogic-kubernetes-operator:3.4.4
```

4. Push the image into the OCIR repository.
   a. Log in to the OCIR repository.

      ```
      source /u01/scripts/utils/provisioning_functions.sh ocir_login
      ```

      In case of Cloud Shell, use the following command and provide the username and password when prompted:

      ```
      docker login <region_name>.ocir.io
      ```

      Example command to log in to the OCIR repository for phoenix region is:

      ```
      docker login phx.ocir.io
      ```

   b. Push the docker image.

      ```
      docker push <region_name>.ocir.io/<tenancy_name>/<resource_prefix>/
      infra/weblogic-kubernetes-operator:3.4.4
      ```

5. Copy the operator values yaml to the `/tmp` directory.

```
cp /u01/provisioning-data/wls-operator-values.yaml /tmp
```

6. Open the file `/tmp/wls-operator-values.yaml` in vi editor and change the image to point to *3.4.4* version.

   For example, if you are upgrading the operator version from *3.4.2* to *3.4.4*, change

```
image: <region_name>.ocir.io/<tenancy_name>/<resource_prefix>/infra/
weblogic-kubernetes-operator:3.4.2
```

   to

```
image: <region_name>.ocir.io/<tenancy_name>/<resource_prefix>/infra/
weblogic-kubernetes-operator:3.4.4
```

7. Install the operator *3.4.4* version using the `helm upgrade` command.

```
helm upgrade <resource_prefix>-weblogic-operator \
/u01/shared/scripts/wls-operator-charts/ \
--reuse-values \
--set image=<region_name>.ocir.io/<tenancy_name>/<resource_prefix>/infra/
weblogic-kubernetes-operator:3.4.4 \
--namespace <resource_prefix>-operator-ns \
--values /tmp/wls-operator-values.yaml
--wait
```

8. Update the configmap file.
   a. Open the configmap file in vi editor.

```
kubectl edit configmap wlsoke-property-configmap -n jenkins-ns
```

   b. Search `weblogic-kubernetes operator` image and change the value to point to 3.4.4 version.

9. After upgrade, verify the operator version is updated to *3.4.4* version, and the existing WebLogic domain PODs are running and using *3.4.4* version.

```
#Verify the operator version
docker images | grep weblogic-kubernetes-operator
#Verify existing Pods are running and using the 3.4.4 version
kubectl get po -n <domain name-space>
kubectl describe po <server-pod-name> -n <domain_name_space>
```

Upgrade the WebLogic Kubernetes Operator to 4.0.5

If you upgrade the cluster and node pools to *1.25.x* using the Oracle Cloud Infrastructure console, you must upgrade the WebLogic Server Kubernetes Operator version to *4.0.5*. else domain-related operations fail.

Perform the following steps to upgrade the operator on the administration host:

1. List all namespaces.

```
helm list -A
```

2. Uninstall the operator.

```
helm uninstall <resource_prefix>-weblogic-operator -n <resource_prefix>-
operator-ns
```

3. Download the operator image.

```
docker pull ghcr.io/oracle/weblogic-kubernetes-operator:4.0.5
```

> **Note:**
>
> If the subnet for the admin host does not have a NAT gateway configured, you must use Cloud Shell to download the operator image.

4. Tag the downloaded docker image to `<region_name>.ocir.io/<tenanacy_name>/<resource_prefix>/infra/weblogic-kubernetes-operator:4.0.5`

   ```
   docker tag ghcr.io/oracle/weblogic-kubernetes-operator:4.0.5
   <region_name>.ocir.io/<tenanacy_name>/<resource_prefix>/infra/weblogic-
   kubernetes-operator:4.0.5
   ```

5. Push the image into the OCIR repository.

   a. Log in to the OCIR repository.

      ```
      source /u01/scripts/utils/provisioning_functions.sh ocir_login
      ```

      In case of Cloud Shell, use the following command and provide the username and password when prompted:

      ```
      docker login <region_name>.ocir.io
      ```

      Example command to log in to the OCIR repository for phoenix region is:

      ```
      docker login phx.ocir.io
      ```

   b. Push the docker image.

      ```
      docker push <region_name>.ocir.io/<tenancy_name>/<resource_prefix>/
      infra/weblogic-kubernetes-operator:4.0.5
      ```

6. Copy the operator values yaml to the `/tmp` directory.

   ```
   cp /u01/provisioning-data/wls-operator-values.yaml /tmp
   ```

7. Open the file `/tmp/wls-operator-values.yaml` in vi editor and change the image to point to *4.0.5* version.

   For example, if you are upgrading the operator version from *3.4.4* to *4.0.5*, change

   ```
   image: <region_name>.ocir.io/<tenancy_name>/<resource_prefix>/infra/
   weblogic-kubernetes-operator:3.4.4
   ```

   to

   ```
   image: <region_name>.ocir.io/<tenancy_name>/<resource_prefix>/infra/
   weblogic-kubernetes-operator:4.0.5
   ```

8. Navigate to `/u01/shared/scripts` folder, create a backup of the existing operator, and download the operator *4.0.5*.

```
cd /u01/shared/scripts
mv wls-operator-charts wls-operator-charts-bkp
mkdir wls-operator-charts
wget https://github.com/oracle/weblogic-kubernetes-operator/archive/refs/
tags/v4.0.5.zip
```

9. Extract the contents of the zip file.

```
unzip v4.0.5.zip
```

   The contents are extracted to `weblogic-kubernetes-operator-4.0.5`.

10. Copy the operator to the `/u01/shared/scripts/` location.

```
cp /u01/shared/scripts/weblogic-kubernetes-operator-4.0.5/kubernetes/
charts/weblogic-operator /u01/shared/scripts/
```

11. Set the following properties:

```
scriptsDir="/u01/scripts"
domain_operator_namespace=<resource_prefix>-operator-ns
weblogic_operator_name=$(python3 ${scriptsDir}/metadata/databag.py
weblogic_operator_name)
weblogic_operator_name=$(echo "$weblogic_operator_name"| awk '{print
tolower($0)}')
wls_operator_charts=/u01/shared/scripts/weblogic-operator/
wls_operator_values_file="/tmp/wls-operator-values.yaml"
```

12. Install the operator *4.0.5* using the following command:.

```
helm install $weblogic_operator_name $wls_operator_charts --
namespace $domain_operator_namespace --values $wls_operator_values_file
```

13. After upgrade, verify the operator version is updated to *4.0.5* version, and the existing WebLogic domain PODs are running and using *4.0.5* version.

```
#Verify the operator version
docker images | grep weblogic-kubernetes-operator
#Verify existing Pods are running and using the 4.0.5 version
kubectl get po -n <domain name-space>
kubectl describe po <server-pod-name> -n <domain_name_space>
```

14. Replace the following files on the administration host:

> **WARNING:**
>
> If you do not replace the files, the **update domain** job fails.

- `log_messages.json` located in `/u01/shared/scripts/pipeline/clogging` with log_messages.json.

- `domain_builder_utils.py` and `pipeline_utils.sh` in `/u01/shared/scripts/pipeline/common` with domain_builder_utils.py and pipeline_utils.sh.

Log File and Script Files

This section lists the log file and the script files that need to be replaced on the administration host after you upgrade the WebLogic Server Kubernetes Operator to *4.0.5*.

Topics:

log_messages.json

Copy the following content in `log_messages.json` located in `/u01/shared/scripts/pipeline/clogging` directory.

```
{
  "WLSOKE-VM-CRITICAL": {

  },
  "WLSOKE-VM-ERROR": {
    "0001": "Error - executing check_versions.sh: [%s]",
    "0002": "TF scripts version does not match the scripts on the vm",
    "0003": "Unable to login to custom OCIR [%s]",
    "0004": "Docker unable to download [%s] image from custom repository.
Exit Code= [%s]",
    "0005": "Error in docker initialization. Exit code[%s].",
    "0006": "Cluster info file is not found.",
    "0007": "OKE Cluster info file is missing.",
    "0008": "OKE Cluster worker nodes are not yet available.",
    "0009": "Error executing status check for OKE worker nodes.",
    "0010": "Error creating namespace [%s]. Error [%s]",
    "0011": "Error creating weblogic operator service account. Exit code[%s]",
    "0012": "Error decrypting the value[%s]",
    "0013": "Error installing weblogic operator. Exit code[%s]",
    "0014": "Weblogic Operator values yaml is not available. Unable to
install operator.",
    "0015": "Error mounting FSS on admin host at mount point [%s]. [exit code
= %s]",
    "0016": "Error creating ocir secrets [ %s ] in oke [namespace=%s]. Error
[%s]",
    "0017": "Pod [%s] condition [%s]",
    "0018": "Error installing public LB for weblogic cluster. [ exit_code
=%s ]",
    "0019": "Usage: /u01/scripts/bootstrap/install_wls_operator.sh
<operator_ns> <wls_domain_ns>",
    "0020": "Error executing %s. Exit code [%s]",
    "0021": "Error create weblogic credential secrets. Exit code [%s]",
    "0022": "Usage: /u01/scripts/domain/create_domain_image.sh <wls_domain>
<wls_domain_ns>",
    "0023": "Error uploading image [%s] to the OCIR repo. Exit code [%s]",
    "0024": "Error in kubectl apply for domain: %s",
```

```
    "0025": "Error: Failed to retrieve secret using secret ocid: %s ",
    "0026": "Error: Weblogic domain is not ready after waiting for [%s]
seconds.",
    "0027": "Error executing command [%s] exit code [%s] and output [%s]",
    "0028": "Exception executing command [%s] is [%s]",
    "0029": "Error creating domain in home image. exit code[%s]",
    "0030": "Error writing load balancer details in file[%s]. [%s]",
    "0031": "Failed to provision domain [%s].[Exception :%s]",
    "0032": "Failed to execute command [%s]. exit_cod[%s]",
    "0033": "Process handled not saved for asynchronous polling",
    "0034": "Timed out waiting for the process [%s] to complete.",
    "0035": "Error: Credential file path not set.",
    "0036": "Error: timed out waiting for the credential file.",
    "0037": "Error: executing provisioner scripts. [exit_code=%s]",
    "0038": "Error: unable to push the domain container image to OCIR. exit
code [%s]",
    "0039": "Failed to save to verified markers file",
    "0040": "Failed to read verified marker file.",
    "0041": "Error running wait-for-all-markers check. [%s]",
    "0042": "Error creating domain in home image after retry. exit code[%s]",
    "0043": "Provisioning failed marker found. Aborting provisioning check.",
    "0044": "Failed to get attribute [%s]: [%s]",
    "0045": "Execute error: [cmd=%s] [exit_code=%s] [error=%s] [output=%s]",
    "0046": "Error deploying persistent volume claim for OKE cluster.",
    "0047": "Error: Persistent volume claim is not bound.[bound = %s]",
    "0048": "Error: Failed to change permission on persistent volume claim.",
    "0049": "Error mounting shared FSS on admin host.[exit code = %s]",
    "0050": "Error loading all attributes: [%s]",
    "0051": "Usage: /u01/scripts/bootstrap/install_jenkins.sh
<path_to_inputs_file>",
    "0052": "Error installing jenkins charts. Exit code[%s]",
    "0053": "Error installing fss charts for jenkins. Exit code[%s]",
    "0054": "Error deploying persistent volume claim for OKE cluster in
jenkins pod.",
    "0055": "Error: Persistent volume claim is not bound in jenkins pod.
[bound = %s]",
    "0056": "Error: Docker images metadata file does not exists.",
    "0057": "Exception: [ %s ]",
    "0058": "Error installing ingress controller with Helm. Exit code [%s]",
    "0059": "Error generating metadata file exception=[%s]",
    "0060": "Error writing provisioning metadata file exception=[%s]",
    "0061": "Error updating the dynamic group for admin.",
    "0063": "Error updating kube config at [%s] with exit code [%s].",
    "0064": "Error updating jenkins configuration. [%s]",
    "0065": "Error Response code for adding request header to load balancer:
[%s]",
    "0066": "Error Response code for getting load balancer details: [%s]",
    "0067": "Error : [%s]. Exception [%s]",
    "0070": "Usage: /u01/scripts/bootstrap/install_ingress_controller.sh
<path_to_inputs_file>",
    "0071": "Error: Failed to install WebLogic Deploy Tool - [%s] Admin Mount
Path not available.",
    "0072": "Error: Failed to install WebLogic Deploy Tool - [%s] WebLogic
Deploy Tool zip not found.",
    "0073": "Error: Failed to install WebLogic Image Tool - [%s] WebLogic
Image Tool zip not found.",
```

```
    "0074": "Error: Failed vulnerability scan of image: [%s]",
    "0075": "Error: [%s]. Exit code [%s]",
    "0076": "Error loading all attributes with v2 endpoint: [%s]",
    "0077": "Unable to create kubeconfig file for cluster: [%s]",
    "0078": "Unable to create kubeconfig file for cluster",
    "0079": "Unable to download atp wallet: [%s]",
    "0080": "ATP download response code: [%s]",
    "0081": "Error while creating IDCS applications in host %s for tenant %s:
[%s]",
    "0082": "Creation of IDCS applications and app gateway failed. Please
check the logs",
    "0083": "Usage: /u01/scripts/bootstrap/install_idcs_services.sh
<path_to_inputs_file>",
    "0084": "Error creating ConfigMap weblogic_conf [%s]",
    "0085": "Error creating ConfigMap for cwallet.sso [%s]",
    "0086": "File [%s] not found after generation.",
    "0087": "Failed to create IDCS Deployment. Please check the logs.",
    "0088": "Status code [%s] while getting cluster info",
    "0089": "Error getting cluster info [%s]",
    "0090": "OKE cluster belongs to different compartment/vcn [%s / %s] than
stack compartment/vcn [%s / %s]",
    "0091": "Error updating the dynamic group for os management.",
    "0092": "Failed to patch domain",
    "0093": "Introspector job is in failed state in weblogic domain namespace
[%s]. [exit_code :  %s]. Please check the introspector logs at location /u01/
shared/logs.",
    "0094": "Failed to create repository [%s] in compartment [%s]. Reason:
[%s]",
    "0095": "Error installing patching tool. [%s]",
    "0096": "Error pushing Verrazzano images to OCIR repo. Exit code: [%s]",
    "0097": "Failed to create Docker registry secret for Verrazzano container
images. Exit code: [%s]",
    "0098": "Failed to install Verrazzano operator. Exit code: [%s]",
    "0099": "Failed to install Verrazzano. Exit code: [%s]",
    "0100": "Failed to install ingress for Jenkins. Exit code [%s]",
    "0101": "Failed to make the repository [%s] public. Please make the repo
public manually.",
    "0102": "Failed to get node pools [%s]",
    "0103": "Failed to get node pool ID [%s]",
    "0104": "Failed to get node pool details [%s]",
    "0105": "Unknown DNS type for Jenkins ingress: [%s]",
    "0106": "Failed to create secret [%s] in namespace [%s] for OCI DNS in
Verrazzano. Exit code: [%s]. Output: [%s]",
    "0107": "Could not retrieve OCID of DNS resolver for VCN [%s]",
    "0108": "Could not retrieve OCID of DNS view for DNS zone [%s]",
    "0109": "Failed to update DNS resolver [%s] with DNS view [%s].
Exception: [%s]",
    "0110": "Failed to create secret [%s] in namespace [%s] for Custom CA in
Verrazzano. Exit code: [%s]. Output: [%s]",
    "0111": "Failed to configure prerequisites for certificates in
Verrazzano. Exit code: [%s]. Output: [%s]",
    "0112": "Failed to configure prerequisites for DNS in Verrazzano. Exit
code: [%s]. Output: [%s]",
    "0113": "Failed to configure prerequisites for Verrazzano. Exit code:
[%s]. Output: [%s]"
  },
```

```
  "WLSOKE-VM-WARNING": {
    "0001": "Failed to delete RCU schemas for prefix = [%s]",
    "0002": "Retrying create domain as earlier attempt failed.",
    "0003": "Warning while trying to format message %s with provided
arguments %s. [%s]",
    "0004": "Warning found failure marker created. Exiting provisioning
flow.",
    "0005": "Retrying create namespace as earlier attempt failed.",
    "0006": "Error running provisioning status check. Please check the
provisioning logs for status of provisioning",
    "0007": "Warning missing life cycle management scripts",
    "0008": "The node [%s] does not have IP addresses",
    "0009": "Verrazzano install did not finish in 30 minutes. Login to the
admin host and check if the install has succeeded."
  },
  "WLSOKE-VM-INFO": {
    "0001": "Executing check_versions script",
    "0002": "VM scripts version: [%s]",
    "0003": "TF scripts version: [%s]",
    "0004": "Executed check_versions script with exit code [%s]",
    "0005": "Executing docker_init script",
    "0006": "Docker login into OCIR [%s]",
    "0007": "Docker downloading image [%s] from ocir",
    "0008": "OKE Cluster information [%s]",
    "0009": "Found [%s] node pools in OKE Cluster",
    "0010": "OKE Node pools statuses : [%s]",
    "0011": "Waiting for the workers nodes to be Active. Retrying...",
    "0012": "Installing weblogic operator",
    "0013": "Created operator namespace [%s]",
    "0014": "Creating operator service account",
    "0015": "Successfully created operator service account [%s]",
    "0016": "Creating RBAC policy for service account.",
    "0017": "Checking pod [ %s ] type [%s] status[ %s ]",
    "0018": "Creating weblogic operator values yaml file",
    "0019": "Successfully created weblogic operator values yaml file",
    "0020": "Installing weblogic operator in namespace [%s]",
    "0021": "Successfully installed weblogic operator.",
    "0022": "Writing operator parameter file",
    "0023": "Operator Parameters: %s",
    "0024": "Domain attributes: %s",
    "0025": "Creating domain yaml file: %s",
    "0026": "Creating namespace: %s",
    "0027": "Creating ocir secrets in oke: %s",
    "0028": "Executing create weblogic domain script...",
    "0029": "Create weblogic credential secrets...",
    "0030": "Create weblogic domain inputs yaml file...",
    "0031": "Uploading container image [%s]",
    "0032": "Docker initialised for provisioning",
    "0033": "Retrieving secret content for OCID %s",
    "0034": "Executing create domain scripts",
    "0035": "Applying domain yaml to OKE cluster",
    "0036": "Waiting pods in domain [%s] to be running",
    "0037": "Successfully created weblogic OKE cluster",
    "0038": "Applying Load balancer yaml file",
    "0039": "Waiting for load balancer service details",
    "0040": "Successfully created load balancer",
```

```
    "0041": "Successfully created model in image",
    "0042": "Successfully executed command [%s]",
    "0043": "Successfully applied domain yaml[%s]",
    "0044": "Executing provisioner scripts",
    "0045": "Successfully completed provisiong [ %s ]",
    "0047": "Returning status code for marker verification [%s]",
    "0048": "Successfully verified all the markers. WebLogic for OKE
provisioning is successful.",
    "0049": "Successfully created the namespace [%s].",
    "0050": "Mounting share FSS on the admin host.",
    "0051": "Successfully mounted share FSS on the admin host",
    "0052": "Executing the mount fss script on admin.",
    "0053": "Successfully executed the mount fss script on admin.",
    "0054": "Executing helm install for fss on OKE cluster.",
    "0055": "Updating the dynamic group for the Admin instance",
    "0056": "Installing jenkins %s",
    "0057": "Successfully installed jenkins in namespace [ %s ]",
    "0058": "Installing ingress controller charts for jenkins [ %s ]",
    "0059": "Successfully installed ingress controller",
    "0060": "Successfully downloaded image [%s] from ocir",
    "0061": "Successfully written file to FSS",
    "0062": "Successfully updated the dynamic group for the Admin",
    "0064": "Executing docker pull on oke nodes",
    "0065": "Executing docker pull on node [ %s ]",
    "0066": "Executed docker pull on node [ %s ]",
    "0067": "Successfully executed docker pull on all nodes",
    "0068": "Running post provisioning clean up scripts",
    "0069": "Executing clean up scripts",
    "0070": "Unzipping lcm and pipeline scripts to FSS.",
    "0071": "Executed clean up scripts",
    "0072": "Oke node init status [exit_code= %s], [output=%s]",
    "0073": "Executing script command [ %s ]",
    "0074": "Executing ssh key update on oke node[%s]",
    "0075": "Successfully executed ssh key update on oke node[%s]",
    "0076": "Creating provisioning service account [%s]",
    "0077": "Creating clusterrolebinding with cluster administration
permissions",
    "0078": "Getting provisioning service account token name",
    "0079": "Updating kubeconfig with provisioning service account token
value",
    "0080": "Setting the provisioning service account in the kubeconfig file
for the current context",
    "0081": "Successfully updated kubeconfig with provisioning service
account token.",
    "0082": "Updating jenkins configuration.",
    "0083": "Found [%s] out [%s] pods in namespace [%s].",
    "0084": "Waiting for command [ %s ] to finish",
    "0087": "Creating ssl certificate secret in weblogic domain and jenkins
namespaces",
    "0088": "Successfully created certificate secret [ %s:%s ] in namespace
[ %s ]",
    "0089": "Successfully added header for Weblogic SSL termination [%s]",
    "0090": "Configuring [%s] load balancer [%s]",
    "0091": "Updating Jenkins job files in shared directory [%s]",
    "0092": "Successfully configured [%s] load balancer for SSL [%s]",
    "0093": "Creating File [%s]",
```

```
    "0095": "Successfully updated the Jenkins job files in shared directory
[%s]",
    "0096": "Successfully installed WebLogic Deploy Tool [%s]",
    "0097": "Successfully installed WebLogic Image Tool [%s]",
    "0098": "Creating configmap [%s]",
    "0099": "Successfully created kubeconfig file for cluster",
    "0100": "ATP Wallet downloaded",
    "0101": "Creating confidential IDCS application %s in host %s for tenant
%s",
    "0102": "Writing confidential IDCS application details to %s...",
    "0103": "Created confidential IDCS application [%s] in host [%s] for
tenant [%s]",
    "0104": "Creating enterprise IDCS application %s in host %s for tenant
%s",
    "0105": "Writing enterprise IDCS application details to %s...",
    "0106": "Created enterprise IDCS application [%s] in host [%s] for tenant
[%s]",
    "0107": "Creating IDCS Application Gateway %s in host %s for tenant %s",
    "0108": "Writing IDCS Application Gateway details to %s...",
    "0109": "Creating IDCS Application Gateway server %s in host %s for
tenant %s",
    "0110": "Creating IDCS Application Gateway mapping with description %s",
    "0111": "Activating IDCS Application Gateway with id %s",
    "0112": "Created IDCS Application Gateway [%s] in host [%s] for tenant
[%s]",
    "0113": "Creating IDCS applications and app gateway",
    "0114": "Deactivating App Gateway [%s] with id [%s]",
    "0115": "Deleting App Gateway [%s] with id [%s]",
    "0116": "The IDCS Application Gateway %s already exists. Deleting it...",
    "0117": "Deactivating IDCS Application Gateway %s.",
    "0118": "Deleting IDCS Application Gateway %s.",
    "0119": "The IDCS Application Gateway server %s already exists. Deleting
it...",
    "0120": "Deleting IDCS Application Gateway server %s.",
    "0121": "The IDCS application %s already exists. Deleting it...",
    "0122": "Deactivating IDCS application %s.",
    "0123": "Deleting IDCS application %s.",
    "0124": "Obtaining information for IDCS app role [%s]",
    "0125": "Adding app role [%s] to application with app id [%s]",
    "0126": "Creating IDCS app gateway config files",
    "0127": "Skipping IDCS Applications creation as IDCS is not selected",
    "0128": "Installing IDCS Service charts for [ %s ]",
    "0129": "Successfully installed IDCS Services",
    "0130": "Starting cwallet generation ...",
    "0131": "Removing client id and secret from [%s]",
    "0132": "The cwallet file generated in [%s]",
    "0133": "OKE cluster passed compartment and vcn validation",
    "0134": "Successfully updated the dynamic group for the os management",
    "0135": "Successfully patched domain",
    "0136": "Successfully created domain yaml",
    "0137": "Successfully updated domain yaml",
    "0138": "Successfully completed status check for Introspector job in
weblogic domain namespace [%s]. [return_code :  %s]",
    "0139": "Successfully created OCIR repo %s in compartment %s",
    "0140": "Successfully installed patching tool [%s]",
    "0141": "Loading Verrazzano images into OCIR",
```

```
    "0142": "Finished loading Verrazzano images into ocir",
    "0143": "Created docker registry secret to allow access to required
images",
    "0144": "Successfully installed Verrazzano operator",
    "0145": "Successfully installed Verrazzano",
    "0146": "Successfully installed Jenkins ingress on Verrazzano ingress
controller",
    "0147": "Making image [%s] public",
    "0148": "Submitted Verrazzano installation task",
    "0149": "Node pool created successfully. Node pool id [%s]",
    "0150": "Using DNS type [%s] for Verrazzano",
    "0151": "Creating secret [%s] in namespace [%s] for OCI DNS in
Verrazzano",
    "0152": "Updated DNS resolver [%s] with DNS view [%s]",
    "0153": "Creating the secret for provisioning service account"
  },

  "WLSOKE-VM-DEBUG": {
    "0001": "OKE nodes : [%s]",
    "0002": "Successfully executed command [%s]",
    "0003": "Executing %s: [%s]",
    "0004": "Pod:%s is not ready. Waiting for %s before retrying",
    "0005": "Pods in namespace [%s] is not ready. Waiting for %s before
retrying",
    "0006": "Waiting for service [%s] external IP. Waiting for %s before
retrying",
    "0007": "Found following markers created during provisioning: %s",
    "0008": "Saved verified markers to status file [markers=%s]",
    "0009": "Provisioning status check found pending markers[%s].",
    "0010": "Verified markers list: %s",
    "0011": "Error getting metadata attribute [%s]: [%s]",
    "0012": "Failed to get metadata from v2 endpoint [attribute=%s] [%s] ",
    "0013": "Error getting attribute [%s]: [%s]",
    "0014": "Failed to get attributes from v2 endpoint [attribute=%s] [%s] ",
    "0015": "Wallet file configmap in namespace [%s] is not available.
Waiting for [%s] before retrying"
  }
}
```

domain_builder_utils.py

Copy the following content in `domain_builder_utils.py` located in `/u01/shared/scripts/pipeline/common` directory.

```
"""
Copyright (c) 2020, 2021, Oracle Corporation and/or its affiliates.
Licensed under the Universal Permissive License v 1.0 as shown at https://
oss.oracle.com/licenses/upl.
"""
import yaml
import sys
import json
import os

def usage(args):
```

```
    """
    Prints usage.

    :param args:
    :return:
    """
    print('Args passed: ' + args)
    print("""
    Usage: python3 domain_builder_utils.py <operation>
    where,
        operation = create-test-domain-yaml
        args = <running_domain_yaml> <test_domain_yaml_file> <new domain
image>

        operation = check-pods-ready
        args = sys.stdin

        operation = get-replica-count
        args = <domain_yaml_file>


    """)
    sys.exit(1)


def get_replica_count(domain_yaml_file):
    """
    Get replica count from domain yaml
    :param domain_yaml_file:
    :return:
    """
    replica_count = 0
    try:
        with open(domain_yaml_file) as f:
            domain_yaml = yaml.full_load(f)
            replica_count = domain_yaml["status"]["clusters"][0]["replicas"]
    except Exception as ex:
        print("Error in parsing json file [%s]: %s" % (domain_yaml_file,
str(ex)))

    print(str(replica_count))

def create_running_domain_yaml(running_domain_yaml_file, new_domain_img,
secrets):
    """
    Create test domain YAML file.

    :param running_domain_yaml_file:        YAML for currently running domain
    :param new_domain_img:                  New image to be updated
    :param secrets:                         List of user defined secrets
    :return:
    """

    existing_secrets = []

    with open(running_domain_yaml_file) as f:
```

```
        running_domain_yaml = yaml.full_load(f)

    running_domain_yaml["spec"]["image"] = new_domain_img
    existing_secrets.extend([running_domain_yaml["spec"]["configuration"]
["model"]["runtimeEncryptionSecret"], running_domain_yaml["spec"]
["webLogicCredentialsSecret"]["name"]])
    if running_domain_yaml["spec"]["configuration"]["model"]["domainType"] ==
"JRF":
        existing_secrets.extend([running_domain_yaml["spec"]["configuration"]
["opss"]["walletPasswordSecret"], running_domain_yaml["spec"]["configuration"]
["opss"]["walletFileSecret"]])

    new_secrets = [name for name in secrets if name not in existing_secrets]
    if new_secrets:
        running_domain_yaml["spec"]["configuration"]["secrets"] = new_secrets

    with open(running_domain_yaml_file, 'w') as f:
            yaml.dump(running_domain_yaml, f)

    print("Successfully created running domain yaml [%s]" %
running_domain_yaml_file)

def get_model_secrets(running_domain_yaml_file):
    """
    Get the list of secrets present in the running domain yaml
    :param running_domain_yaml_file:    Running domain yaml file
    :return:
    """

    existing_secrets = []

    with open(running_domain_yaml_file) as f:
        running_domain_yaml = yaml.full_load(f)

    existing_secrets.extend([running_domain_yaml["spec"]["configuration"]
["model"]["runtimeEncryptionSecret"], running_domain_yaml["spec"]
["webLogicCredentialsSecret"]["name"]])
    if running_domain_yaml["spec"]["configuration"]["model"]["domainType"] ==
"JRF":
        existing_secrets.extend([running_domain_yaml["spec"]["configuration"]
["opss"]["walletPasswordSecret"], running_domain_yaml["spec"]["configuration"]
["opss"]["walletFileSecret"]])

    if "secrets" in running_domain_yaml["spec"]["configuration"]:
        existing_secrets.extend(running_domain_yaml["spec"]["configuration"]
["secrets"])

    secrets = ''
    for name in existing_secrets:
        secrets += str(name) + ' '
    print(secrets)


def check_pods_ready(file):
    """
    Check if pods are ready and print the count of pods that are in ready
```

```
state
    :param file:    stdin file descriptor
    :return:
    """
    count = 0
    try:
        a = json.load(file)

        for i in a['items']:
            for j in i['status']['conditions']:
                if j['status'] == "True" and j['type'] == "Ready" and
i['status']['phase'] == 'Running':
                    # print(i['metadata']['name'])
                    count = count + 1
    except:
        print("The data from stdin doesn't appear to be valid json. Fix
this!")
        sys.exit(1)
    print(count)


def get_ocir_user(ocir_url, file):
    """
    Get OCIR user from the input ocirsecrets auths json.

    :param ocir_url: e.g. phx.ocir.io
    :param file: stdin from kubectl command to read ocirsecrets json.
    :return:
    """
    try:
        a = json.load(file)
        if 'Username' in a['auths'][ocir_url]:
            print(a['auths'][ocir_url]['Username'])
        else:
            print(a['auths'][ocir_url]['username'])
    except:
        print("The data from stdin doesn't appear to be valid json. Fix
this!")
        sys.exit(1)

def get_ocir_auth_token(ocir_url, file):
    """
    Get OCIR auth token from the input ocirsecrets auths json.

    :param ocir_url: e.g. phx.ocir.io
    :param file: stdin from kubectl command to read ocirsecrets json.
    :return:
    """
    try:
        a = json.load(file)
        if 'Password' in a['auths'][ocir_url]:
            print(a['auths'][ocir_url]['Password'])
        else:
            print(a['auths'][ocir_url]['password'])
    except:
        print("The data from stdin doesn't appear to be valid json. Fix
```

```
this!")
        sys.exit(1)

def get_metadata_attribute(attr):
    """
    Get Metadata attribute.
    Assumes that the metadata attributes are loaded from configmap and
exposed in the pod as environment variables.

    :param attr:    Attribute to look for.
    :return:
    """
    return os.environ[attr]

def main():
    if len(sys.argv) < 2:
        usage(sys.argv)
    try:
        operation = sys.argv[1]

        if operation == 'create-running-domain-yaml':
            if len(sys.argv) < 5:
                usage(sys.argv)
            running_domain_yaml_file = sys.argv[2]
            new_domain_img = sys.argv[3]
            secrets = sys.argv[4:]

            create_running_domain_yaml(running_domain_yaml_file,
new_domain_img, secrets)
        elif operation == 'check-pods-ready':
            check_pods_ready(sys.stdin)
        elif operation == 'get-ocir-user':
            ocir_url = sys.argv[2]
            get_ocir_user(ocir_url, sys.stdin)
        elif operation == 'get-ocir-auth-token':
            ocir_url = sys.argv[2]
            get_ocir_auth_token(ocir_url, sys.stdin)
        elif operation == 'get-replica-count':
            if len(sys.argv) < 3:
                usage(sys.argv)
            domain_yaml_file = sys.argv[2]
            get_replica_count(domain_yaml_file)
        elif operation == 'get-secrets-list':
            running_domain_yaml_file = sys.argv[2]
            get_model_secrets(running_domain_yaml_file)

    except Exception as ex:
        print("Error: " + str(ex))
        sys.exit(1)


if __name__ == "__main__":
    main()
```

pipeline_utils.sh

Copy the following content in `pipeline_utils.sh` located in `/u01/shared/scripts/pipeline/common` directory.

```
#!/usr/bin/env bash
# Copyright (c) 2020, 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.


# This script defines functions that are called across different pipeline
stages.

script="${BASH_SOURCE[0]}"
scriptDir="$( cd "$( dirname "${script}" )" && pwd )"
fileName=$(basename $BASH_SOURCE)

source ${scriptDir}/pipeline_constants.sh

# Function: Get metadata attribute.
# metadata_file: path to metadata file
# attribute: metadata attribute
get_metadata_attribute() {
    val=$(printenv $1)
    if [[ -z $val ]]; then
       val=$(kubectl get cm wlsoke-metadata-configmap -n jenkins-ns -o
jsonpath="{.data.$1}")
    fi
    echo $val
}

# Function: Get OCIR Username
# ocir_url: OCIR url e.g. phx.ocir.io
# domain_ns: domain namespace
# ocirsecret_name: Name of imagePullSecrets[0] defined in domain.yaml
get_ocir_user() {
    local ocir_url=$(get_metadata_attribute  'ocir_url')
    local ocirsecret_name=$1
    local domain_ns=$(get_metadata_attribute  'wls_domain_namespace')

    local auths_json=$(kubectl get secret ${ocirsecret_name} -n ${domain_ns} -
o jsonpath="{.data.\.dockerconfigjson}" | base64 -d)

    result=$(echo ${auths_json} | python3 ${scriptDir}/
domain_builder_utils.py 'get-ocir-user' ${ocir_url})

    echo ${result}
}

# Function: Get OCIR Auth Token
# ocir_url: OCIR url e.g. phx.ocir.io
# domain_ns: domain namespace
# ocirsecret_name: Name of imagePullSecrets[0] defined in domain.yaml
get_ocir_auth_token() {
```

```
    local ocir_url=$(get_metadata_attribute  'ocir_url')
    local ocirsecret_name=$1
    local domain_ns=$(get_metadata_attribute  'wls_domain_namespace')

    local auths_json=$(kubectl get secret ${ocirsecret_name} -n ${domain_ns} -
o jsonpath="{.data.\.dockerconfigjson}" | base64 -d)

    result=$(echo ${auths_json} | python3 ${scriptDir}/
domain_builder_utils.py 'get-ocir-auth-token' ${ocir_url})

    echo ${result}
}

# Function: Generate updated domain image tag for WLS-OKE.
# metadata_file: metadata file
# timestamp: build timestamp to be used for image tagging
generate_domain_img_tag() {
    local timestamp=$1
    local tag=""
    # Generate a tag for new domain image
    if [[ -n ${DOMAIN_NAME} ]]; then
      #BASE_IMAGE: <ocir-url>/<tenancy>/okestack/wls-base-image/
14110:14.1.1.0.230117-230117
      local ocir_domain_image_repo=$(get_domain_property "${DOMAIN_NAME}"
"BASE_IMAGE")
      # repo_path=<ocir-url>/<tenancy>/mytest/wls-base-image/12214
      local repo_main_path=$(echo ${ocir_domain_image_repo} | cut -d ":" -f 1)
      # repo_path=<ocir-url>/mytenancy/myserv/wls-base-image
      repo_path=$(dirname ${repo_main_path})
      # wls_base_version = 12.2.1.4.191220-200203
      local wls_base_version=$2
      if [[ -z $wls_base_version ]]; then
        wls_base_version=$( echo ${ocir_domain_image_repo} | cut -d ":" -f 2)
      fi
      # tag = <ocirurl>/mytenancy/myserv/wls-base-image/mydomain/
12214:12.2.1.4.191220-200203-<timestamp>
      wls_version=$(echo $(echo ${ocir_domain_image_repo} | cut -d ":" -f 1)
| cut -d "/" -f5-)
      tag="$repo_path/$DOMAIN_NAME/${wls_version}:${wls_base_version}-$
{timestamp}"
    else
      wls_base_version=$2
      #iad.ocir.io/idivyfxzwa6h/mystack1411/wls-base-image
      local ocir_domain_image_repo=$(get_metadata_attribute
'ocir_domain_repo')
      # Create Domain Base Image job. This does not act on a given domain.
      wls_version=$(echo "${wls_base_version}" | sed 's/\.//g')
      #wls_version=14.1.1.0.0, need to exclude the last zero for the base
image wls version
      tag=$ocir_domain_image_repo"/"${wls_version::5}":"${wls_base_version}-$
{timestamp}
    fi
    echo ${tag}
}

# Function: Generate updated domain image tag for Verrazzano.
```

```
# metadata_file: metadata file
# timestamp: build timestamp to be used for image tagging
vz_generate_domain_img_tag() {
    local timestamp=$1
    if [[ $# == 2 ]]; then
       DOMAIN_NAME=$2
    fi
    # Generate a tag for new domain image for Verrazzano
    #<ocir-url>/<tenancy>/mytest/wls-base-image:12.2.1.4.210420-210502
    local ocir_domain_image_repo=$(get_domain_property "${DOMAIN_NAME}"
"BASE_IMAGE")
    # repo_path=<ocir-url>/<tenancy>/mytest/wls-base-image/12214
    local repo_main_path=$(echo ${ocir_domain_image_repo} | cut -d ":" -f 1)
    # repo_path=<ocir-url>/mytenancy/myserv/wls-base-image
    repo_path=$(dirname ${repo_main_path})
    # wls_base_version = 12.2.1.4.191220-200203
    wls_base_version=$( echo ${ocir_domain_image_repo} | cut -d ":" -f 2)
    local tag=""
    if [[ -n ${DOMAIN_NAME} ]]
    then
      # tag = <ocirurl>/mytenancy/myserv/wls-base-image/mydomain/
12214:12.2.1.4.191220-200203-<timestamp>
      wls_version=$(echo $(echo ${ocir_domain_image_repo} | cut -d ":" -f 1)
| cut -d "/" -f5-)
      tag="$repo_path/$DOMAIN_NAME/$wls_version:${DOMAIN_NAME}-${timestamp}"
    else
      # Create Domain Base Image job. This does not act on a given domain.
      # tag = <ocir-url>/mytenancy/myserv/mydomain/wls-domain-
base:12.2.1.4.191220-200203-<timestamp>
      tag=$repo_main_path":"${wls_base_version}-${timestamp}
    fi
    echo ${tag}
}

# Function: Update running domain yaml file.
# metadata_file:
# running_domain_yaml: path to the running domain.yaml file to be updated
# build_timestamp: timestamp used for the new configmap
update_running_domain_yaml() {
    local running_domain_yaml=$1
    local build_timestamp=$2
    local secrets=()
    local i=0

    for name in `kubectl get secrets -n ${DOMAIN_NS} |grep Opaque|awk
'{print $1}'` ;
    do
      secrets[$i]=$name
      i=`expr $i + 1`
    done

    python3 ${scriptDir}/domain_builder_utils.py 'create-running-domain-
yaml' ${running_domain_yaml} ${build_timestamp} "${secrets[@]}"
}

#Function: Check introspector pod status
```

```
#domain_ns: domain namespace
#domain_uid: domain UID
#max_wait_time: timeout duration for pods in the domain to come to ready state
check_introspector_status(){
    set -x
    local domain_ns=$1
    local domain_uid=$2
    local max_wait_time=$3
    local interval=30
    local status=2

    local consistent_result_count=0
    local max_consistency_count=6

    local count=0
    let max_retry=${max_wait_time}/${interval}
    mkdir -p /tmp/intro
    cd /tmp/intro

    $log Info "0116" $fileName

    # Checking the domain status and admin server image is updated with the
latest image.
    sleep ${interval}s
    status_type=$(kubectl get domain ${domain_uid} -n ${domain_ns} -o
jsonpath="{..status.conditions[0].type}")
    get_admin_server_image=$(kubectl get po ${domain_uid}-${domain_uid}-
adminserver -n ${domain_ns} -o jsonpath="{.spec.containers[0].image}")
    message=$(kubectl get domain ${domain_uid} -n ${domain_ns} -o
jsonpath="{..status.conditions[0].message}")

    #Checking and throwing error either status message contains Failed/failed
string or status type is Failed.
    if [[ ${message} =~ "ailed" || "${status_type}" == "Failed" ]]; then
       $log Error "0092" $fileName
       ((status=-1))
    else
      # Waiting for Admin pod to come up with the new image.
      while [[ "${NEW_DOMAIN_IMAGE}" != "$get_admin_server_image"  && $
{consistent_result_count} -ne ${max_consistency_count}  ]] ; do
         ((consistent_result_count++))
         $log Info "0117" $fileName
         sleep ${interval}s
         status_type=$(kubectl get domain ${domain_uid} -n ${domain_ns} -o
jsonpath="{..status.conditions[0].type}")
         get_admin_server_image=$(kubectl get po ${domain_uid}-${domain_uid}-
adminserver -n ${domain_ns} -o jsonpath="{.spec.containers[0].image}")
         message=$(kubectl get domain ${domain_uid} -n ${domain_ns} -o
jsonpath="{..status.conditions[0].message}")

         if [[ -n ${message} ]] && [[ "${status_type}" == "Failed" ]]; then
            $log Error "0092" $fileName
            $log Error "0093" $fileName "${message}"
            ((status=-1))
            break
         elif [[ "${status_type}" == "Available" ]] && [[ -z ${message} ]];
```

```
then
            continue
          fi
      done

      #Waiting for all pods to be updated with the new image.
      if [[ $status -eq 2 ]]; then
        while [[ $count -lt ${max_retry} ]] ; do
          $log Info "0119" $fileName
          sleep ${interval}s
          #Getting the latest status of the serves.
          get_admin_server_image=$(kubectl get po ${domain_uid}-${domain_uid}-
adminserver -n ${domain_ns} -o jsonpath="{.spec.containers[0].image}")
          admin_pod_status=$(kubectl get po ${domain_uid}-${domain_uid}-
adminserver -n ${domain_ns} -o jsonpath="{..status.phase}")
          get_managed_server_image=$(kubectl get po ${domain_uid}-$
{domain_uid}-managed-server1 -n ${domain_ns} -o
jsonpath="{.spec.containers[0].image}")
          managed_server_pod_status=$(kubectl get po ${domain_uid}-$
{domain_uid}-managed-server1 -n ${domain_ns} -o jsonpath="{..status.phase}")

          # Checking Admin and Managed servers are updated with the new
domain image and both are running or not?
          if [[ "${NEW_DOMAIN_IMAGE}" == "$get_admin_server_image"  &&
"$admin_pod_status" == "Running" ]]; then
           if [[ "${NEW_DOMAIN_IMAGE}" == "$get_managed_server_image" &&
"$managed_server_pod_status" == "Running" ]]; then
            $log Info "0118" $fileName
            status=0
            break;
           fi
          else
            ((count++))
            continue;
          fi
        done
      fi
    fi

    if [[ ${status} -eq 0 ]]
    then
        return 0
    else
        return 1
    fi
}


# Function: Check pod status is ready
check_pods_ready() {
    result=$(python3 ${scriptDir}/domain_builder_utils.py 'check-pods-ready')
    echo ${result}
}

# Function: Wait for pods to be ready
# domain_yaml: domain yaml
```

```
# max_wait_time: timeout duration for pods in the domain to come to ready
state
# domain_ns: domain namespace
# num_pods_to_run: how many pods are there in the domain
wait_for_pods() {
    set +x
    local domain_yaml=$1
    local max_wait_time=$2
    local domain_ns=$3
    local num_pods_to_run=$4

    local count=0
    local interval=30
    let max_retry=${max_wait_time}/${interval}

    # Ensure the check_pods_ready count remains consistent over 4*30secs =
2mins
    # This is needed for the case when we are doing rolling restart of the
domain (default case)
    #
    local consistent_result_count=0
    local max_consistency_count=4

    echo "Waiting for domain server pods [$num_pods_to_run] to be ready
(max_retries: $max_retry at interval: $interval seconds) ..."
    local START=$(date +%s)

    count_pods_ready=$(kubectl get pods -n ${domain_ns} -o json |
check_pods_ready)
    while [[ ( ${count_pods_ready} -ne ${num_pods_to_run} || $
{consistent_result_count} -ne ${max_consistency_count} ) && $count -lt $
{max_retry} ]] ; do
        sleep ${interval}s
        count_pods_ready=$(kubectl get pods -n ${domain_ns} -o json |
check_pods_ready)

        # Check if all pods are ready then the result remains consistent for
sometime
        while [[ ${consistent_result_count} -ne ${max_consistency_count} ]]
&& [[ ${count_pods_ready} -eq ${num_pods_to_run} ]] ; do
            let consistent_result_count=consistent_result_count+1

            echo "Consistent result count: ${consistent_result_count}"
            echo "[$count_pods_ready of $num_pods_to_run] are ready"

            sleep ${interval}s
            count_pods_ready=$(kubectl get pods -n ${domain_ns} -o json |
check_pods_ready)
        done
        if [[ ${consistent_result_count} -eq ${max_consistency_count} ]] &&
[[ ${count_pods_ready} -eq ${num_pods_to_run} ]] ; then
            break
        else
            let consistent_result_count=0
        fi
        let count=count+1
```

```
    done

    echo "Exiting wait_for_pods: [$count_pods_ready of $num_pods_to_run] are
ready"
    echo "consistent_result_count: [$consistent_result_count] of
[$max_consistency_count]"
    echo "retries: [$count] of [$max_retry]"

    local END=$(date +%s)
    echo "Domain startup took:"
    echo $((END-START)) | awk '{print int($1/60)"m:"int($1%60)"s"}'

    if [[ ${count_pods_ready} -eq ${num_pods_to_run} ]]
    then
        return 0
    else
        return 1
    fi
    set -x
}

# Function: Login to OCIR.
# metadata_file: provisioning metadata file
# ocir_user: OCIR username
# ocir_auth_token: OCIR user auth token
ocir_login() {
    set +x
    local ocir_url=$(get_metadata_attribute 'ocir_url')
    local ocir_user=$1
    local ocir_auth_token=$2

    $log Info "0120" $fileName "$ocir_url" "$ocir_user"
    echo ${ocir_auth_token} | docker login ${ocir_url} --username $
{ocir_user} --password-stdin
    exit_code=$?

    if [[ $exit_code -ne 0 ]]; then
        $log Error "0094" $fileName "$ocir_url"
        exit 1
    fi
}

#Function: Select the OS Linux Base Image
#linux_version: Linux version used for base image
os_base_image() {
   set +x
   local version=`echo ${1} | cut -d '-' -f2`
   local result

   images=(`docker images| tail -n +2`)

   for ((x=0;x<${#images[@]};x++)); do
      if [[ ${images[$x]} =~ "oraclelinux" ]]; then
         name="${images[$x]}"
         ((x++))
         if [[ ${images[$x]} =~ $version ]]; then
```

```
            result=$name:"${images[$x]}"
            break
          fi

      fi
   done

   echo ${result}
}

#
# Uploads image to ocir repo
# param: ocir_image_tag
# param: exit code to return for error
function ocir_image_upload() {
    set -x
    ocir_image_tag=$1
    return_exit_code=$2

    if [[ ${return_exit_code} == "" ]]; then
        return_exit_code=1
    fi

    python3 /u01/shared/scripts/pipeline/clogging/shellLogging.py Info
"0016" $fileName $ocir_image_tag

    #Pushing the docker image to compartment level instead of root compartment
    repo_path=$(echo $(echo "${ocir_image_tag}" | cut -d ":" -f 1) | cut -d
"/" -f3-)
    #Creating the repo at compartment level to push the pipeline images
    compartment_id=$(kubectl get cm wlsoke-metadata-configmap -n jenkins-ns -
o jsonpath="{.data.\oke_cluster_compartment_id}")
    python3 ${pipeline_common}/create_repo.py "$compartment_id" "$repo_path"
    exit_code=$?

    if [[ $exit_code -ne 0 ]]; then
      $log Error "0150" $fileName "$docker_repo_path"
"$compartment_id" $exit_code
      exit 2
    fi

    cmd_output=$(docker push $ocir_image_tag 2>&1)
    exit_code=$?
    echo "${cmd_output}"

    if [[ ${exit_code} -ne 0 ]]; then
        docker images
        python3 /u01/shared/scripts/pipeline/clogging/shellLogging.py Error
"0021" $fileName $ocir_image_tag $exit_code
        exit $return_exit_code
    fi

}

# Function: Validate the Domain is running and server PODs are in RUNNING
state.
```

```
validate_running_domain() {
    set -x
    local running_domain_yaml=/tmp/running-domain-${BUILD_TIMESTAMP}.yaml
    local is_idcs_selected=$(get_domain_property $DOMAIN_NAME
IS_IDCS_SELECTED)

    # Get running domain yaml
    kubectl get domain ${DOMAIN_NAME} -n ${DOMAIN_NS} -o yaml > $
{running_domain_yaml}
    exit_code=$?
    if [[ $exit_code -ne 0 ]]; then
        $log Error "0089" $fileName "${DOMAIN_NAME}" "${DOMAIN_NS}"
        exit_with_cleanup 7
    fi

    # Get replica count in domain yaml
    local replica_count=$(python3 /u01/shared/scripts/pipeline/common/
domain_builder_utils.py 'get-replica-count' ${running_domain_yaml})
    if [[ ${is_idcs_selected} == "YES" ]]; then
      let num_pods_to_run=replica_count+2
    else
      let num_pods_to_run=replica_count+1
    fi

    # Max wait time 120 mins for pods to be ready
    let max_wait_time=120*60

    wait_for_pods ${running_domain_yaml} ${max_wait_time} ${DOMAIN_NS} $
{num_pods_to_run}

    if [[ $? -ne 0 ]]; then
       $log Error "0090" $fileName
        exit_with_cleanup 7
    else
        $log Info "0111" $fileName
    fi
    set +x
}

# Function: Deploy updated domain image to the running domain.
deploy_domain_img() {
    set -x
    local tag=${NEW_DOMAIN_IMAGE}

    # Apply the domain image to running domain if publish is selected
    $log Info "0108" $fileName "$tag"
    local running_domain_yaml=/tmp/running-domain-${BUILD_TIMESTAMP}.yaml

    # Get running domain yaml
    kubectl get domain ${DOMAIN_UID} -n ${DOMAIN_NS} -o yaml > $
{running_domain_yaml}
    exit_code=$?
    if [[ $exit_code -ne 0 ]]; then
        $log Error "0085" $fileName "${DOMAIN_UID}" "${DOMAIN_NS}"
        exit_with_cleanup 3
    fi
```

```
    # Back it up
    mkdir -p /u01/shared/weblogic-domains/${DOMAIN_UID}/backups/$
{BUILD_TIMESTAMP}
    cp ${running_domain_yaml} /u01/shared/weblogic-domains/${DOMAIN_UID}/
backups/${BUILD_TIMESTAMP}/prev-domain.yaml

    # Update domain.yaml with the new image and secrets.
    update_running_domain_yaml ${running_domain_yaml} ${tag}

    # Apply the changes
    kubectl apply -f ${running_domain_yaml}
    exit_code=$?

    if [[ $exit_code -ne 0 ]]
    then
        exit_with_cleanup 4
    fi

    # Replace the domain.yaml file in shared filesystem
    cp ${running_domain_yaml} /u01/shared/weblogic-domains/${DOMAIN_UID}/
domain.yaml

    # Back up the new domain.yaml in backup directory
    cp ${running_domain_yaml} /u01/shared/weblogic-domains/${DOMAIN_UID}/
backups/${BUILD_TIMESTAMP}/domain.yaml

    set +x
}

# Function: Validate if the introspector has completed successfully
validate_introspector() {
    set -x
    max_wait_time=5*60

    local is_apply_jrf=$(get_domain_property $DOMAIN_NAME IS_APPLY_JRF)
     if [[ ${is_apply_jrf} == "true" ]]
    then
      max_wait_time=15*60
    fi

    check_introspector_status ${DOMAIN_NS} ${DOMAIN_NAME} ${max_wait_time}
    if [[ $? -ne 0 ]]; then
        $log Error "0088" $fileName "${DOMAIN_NAME}"
        cp /u01/shared/weblogic-domains/${DOMAIN_NAME}/backups/$
{BUILD_TIMESTAMP}/prev-domain.yaml /u01/shared/weblogic-domains/$
{DOMAIN_NAME}/domain.yaml
        exit_with_cleanup 6
    else
        $log Info "0110" $fileName
    fi

    set +x
}

# Function: Rollback domain
```

```
rollback_domain() {
    set -x
    local running_domain_yaml=/tmp/running-domain-${BUILD_TIMESTAMP}.yaml
    local prev_domain_yaml=/u01/shared/weblogic-domains/${DOMAIN_NAME}/
backups/${BUILD_TIMESTAMP}/prev-domain.yaml

    if [[ -f ${prev_domain_yaml} ]]; then
        #Get running domain yaml
        kubectl get domain ${DOMAIN_NAME} -n ${DOMAIN_NS} -o yaml > $
{running_domain_yaml}
        exit_code=$?
        if [[ $exit_code -eq 0 ]]; then
            old_mii_image=`echo $(sed -n '/image:/p' ${prev_domain_yaml} |
cut -d':' -f2-) | sed 's/ *$//g'`
            cp ${running_domain_yaml} ${prev_domain_yaml}
            # Update domain.yaml with the old mii image
            sed -i -e "s|\(image: \).*|\1 \"${old_mii_image}\"|g" $
{prev_domain_yaml}
            $log Info "0112" $fileName "${prev_domain_yaml}" "$
{old_mii_image}"
            cat ${prev_domain_yaml}
            # Apply the domain yaml with old domain image
            kubectl apply -f ${prev_domain_yaml}
        else
            #In case there is no domain (which should not happen) we will try
to apply the file that was backup when the new domain image was applied
            $log Error "0091" $fileName "${DOMAIN_UID}" "${DOMAIN_NS}" "$
{prev_domain_yaml}"
              # Apply the domain yaml backed up when the image was applied.
            kubectl apply --force -f ${prev_domain_yaml}
        fi
        validate_running_domain
        # Replace current domain.yaml with most current domain
        kubectl get domain ${DOMAIN_UID} -n ${DOMAIN_NS} -o yaml > /u01/
shared/weblogic-domains/${DOMAIN_UID}/domain.yaml
    else
        $log Info "0113" $fileName
    fi
    # Cleanup the earlier image from OCIR repo. TODO: automate this step
    $log Info "0114" $fileName "${NEW_DOMAIN_IMAGE}"
    set +x
}


# Function: Rollback to specified domain image.
# Param: rollback_to_image - domain image to rollback to.
rollback_domain_to_image() {
  set -x
  local rollback_to_image=$1

  timestamp=$(date +"%y-%m-%d_%H-%M-%S")
  local running_domain_yaml=/tmp/running-domain-${timestamp}.yaml

  # Get running domain yaml
  kubectl get domain ${DOMAIN_UID} -n ${DOMAIN_NS} -o yaml > $
{running_domain_yaml}
```

```
  exit_code=$?
  if [[ $exit_code -ne 0 ]]; then
      $log Error "0089" $fileName "${DOMAIN_UID}" "${DOMAIN_NS}"
      exit_with_cleanup 2
  fi

  # Update domain.yaml with the old domain image
  sed -i -e "s|\(image: \).*|\1 \"${rollback_to_image}\"|g" $
{running_domain_yaml}
  $log Info "0112" $fileName "${running_domain_yaml}" "${rollback_to_image}"
  cat ${running_domain_yaml}

  # Apply the domain yaml with old domain image
  kubectl apply -f ${running_domain_yaml}
  exit_code=$?
  if [[ $exit_code -ne 0 ]]
  then
    exit_with_cleanup 3
  fi

  validate_running_domain

  # Remove the temp running domain yaml
  rm -f ${running_domain_yaml}
  set +x
}

# Copy this function here because this might be called from auto-patching
flow where pipeline_common.sh is not sourced.
exit_with_cleanup() {
    set -x
    local exit_code=$1

    # Get out of the build context directory so we can delete it
    cd "$(pwd)"
    # Remove the temp domain image build directory
    rm -rf /tmp/deploy-apps
    # Temp domain yaml files created
    rm -f /tmp/test-domain-${BUILD_TIMESTAMP}.yaml
    rm -f /tmp/running-domain-${BUILD_TIMESTAMP}.yaml
    # Remove patching files created
    rm -f /tmp/apply_opatches.log
    rm -f /tmp/finalbuild.txt
    rm -f /tmp/oraInst.loc
    rm -rf /tmp/opatch_updated_tag.txt
    # Scan image cleanup
#    docker stop ${BUILD_TIMESTAMP}-clair
#    docker stop ${BUILD_TIMESTAMP}-db
#    rm -f /tmp/clair-${BUILD_TIMESTAMP}.log
    set +x
    exit ${exit_code}
}
```

Upgrade the Tools in Oracle WebLogic Server for OKE

Learn how to upgrade the tools used in Oracle WebLogic Server for OKE.

Topic:

Upgrade WebLogic Deploy Tooling

To upgrade WebLogic Deploy Tooling (WDT), download the latest version of the WebLogic Deploy Tooling (WDT) from GitHub.

1. Access the administration compute instance for your stack. See Access the Administration Instance.
2. Go to `/u01/shared/tools/` location.
3. Create a backup of existing WDT file.

   ```
   mv weblogic-deploy weblogic-deploy_backup
   mv weblogic-deploy.zip weblogic-deploy.zip_backup
   ```

4. Browse to the URL, https://github.com/oracle/weblogic-deploy-tooling/releases/latest.
5. Download the `weblogic-deploy.zip` file to the `/u01/shared/tools/` location on the administration host.

   ```
   wget <location of weblogic-deploy.zip file>
   ```

   Example:

   ```
   wget https://github.com/oracle/weblogic-deploy-tooling/releases/download/
   release-2.3.2/weblogic-deploy.zip
   ```

   If the download fails due to connection time out issue, configure the proxy settings using the following commands:

   ```
   export http_proxy=<proxy_server_ip_address>
   export https_proxy=$http_proxy
   ```

6. Extract the contents of the WDT zip file.

   ```
   unzip weblogic-deploy.zip
   ```

   You can now create a domain and update an domain using the upgraded WDT file.

Back Up and Restore a Model in Image Domain

Oracle WebLogic Server for OKE provides the ability to backup and restore your domain.

Back up a domain:

- By default, every time you run a job, a back up of the domain is created automatically.
- There is no job available to periodically back up your domain.
- You can also manually take a backup by copying the current `yaml` file to your preferred location. A `yaml` file has information about the current setup of the domain.

The backups are location at: `/u01/shared/weblogic-domains/<domain_name>/backups`

A sample backup:

```
[opc@oracle-admin oracledomain]$ pwd
/u01/shared/weblogic-domains/oracledomain
[opc@oracle-admin oracledomain]$ ll -tR
.:
total 25
-rwxr-xr-x. 1 opc opc 3728 Sep 24 16:14 domain.yaml
drwxr-xr-x. 3 opc opc 1 Sep 24 16:14 backups
-rw-rw-r--. 1 opc opc 4642 Sep 23 17:16 provisioning_metadata.json
-rw-rw-r--. 1 opc opc 1495 Sep 23 16:55 create-domain-inputs.yaml

./backups:
total 1
drwxr-xr-x. 2 opc opc 2 Sep 24 16:14 20-09-24_15-57-44

./backups/20-09-24_15-57-44:
total 16
-rw-r--r--. 1 opc opc 3728 Sep 24 16:14 domain.yaml
-rw-r--r--. 1 opc opc 3725 Sep 24 16:14 prev-domain.yaml
```

Where:

- `prev-domain.yaml`, is the previous domain that was running before the current job was completed.
- `domain.yaml`, is the existing domain after the current job was completed.

Restore a domain:

1. Access the administration compute instance for your stack.
   See Access the Administration Instance.
2. Go to the backup location, where the domain `yaml` that you want to apply is located.
3. Open the domain `yaml` and make a note of the image id.
   For example:

   ```
   image: "phx.ocir.io/ax8cfrmecktw/oracle/oracle_domain/wls-domain-
   base:12.2.1.4.200714-200819-20-09-23_14-51-16"
   ```

4. Run the following commands to set the required environment variables:

   ```
   export wls_domain_namespace=<domain namespace>
   export wls_domain_uid=<domain UID>
   export ocir_url=<region>.ocir.io
   ```

5. Run the following command:

   ```
   /u01/shared/scripts/pipeline/common/pipeline_common.sh -i <image_id>
   ```

Where, <*image_id*> is the image id you noted in step 2.

For example:

```
/u01/shared/scripts/pipeline/common/pipeline_common.sh -i phx.ocir.io/
ax8cfrmecktw/oracle/oracle_domain/wls-domain-
base:12.2.1.4.200714-200819-20-09-23_14-51-16
```

Back Up and Restore a Domain on PV

Oracle recommends that you back up the domain home directory (and the JRF database for the JRF domain) after you have created the initial domain. You can then continue to take periodic backups, ensuring that all the latest changes are backed up.

You may update the domain on PV domain configuration after its initial deployment. For example, you may want to add new applications. However, after the update, the original WDT model files used to create the initial domain may not match the current state of the domain.

Oracle recommends that you periodically back up the domain home with the latest changes. The domain can be restored from the backup copy if you want to revert the changes made to the domain home directory. If the domain home is not properly backed up, there is a possibility of losing existing data if the domain home becomes corrupt or gets deleted.

Topics:

Back Up the Domain

The back up procedure comprises the following steps:

Back Up the Domain Home Directory

You can use the following script to back up the domain home to the `/u01/shared` backup location. This script is available in the `/u01/scripts/utils/backup.sh` on the administration instance.

```
#!/bin/bash -x
if [ -z "$1" ]; then
  echo "Usage: backup.sh <domain_name>"
  exit 1
fi

domain_name=$1
timestamp=`date '+%Y-%m-%dT%T.%3N'`
domain_dir=/u01/shared/data/domains/$domain_name
domain_uid_dir=${domain_dir}-uid

backup_dir=/u01/shared/data/domains.backup/$domain_name/$timestamp/
mkdir -p $backup_dir
echo backup_dir: $backup_dir
```

```
echo "backing up $domain_dir"
cp -r $domain_dir $backup_dir

echo "backing up $domain_uid_dir"
cp -r $domain_uid_dir $backup_dir

echo "backup created: $timestamp"
```

Execute the script using the following command:

```
bash /u01/scripts/utils/backup.sh <domain_name>
```

This is an example of the output:

```
$ bash backup.sh pvdomain
backup_dir: /u01/shared/data/domains.backup/pvdomain/2023-12-04T18:53:24.381/
backing up /u01/shared/data/domains/pvdomain
backing up /u01/shared/data/domains/pvdomain-uid
backup created: 2023-12-04T18:53:24.381
```

The last line of the backup directory path is the backup timestamp (in the above example, it is ***2023-12-04T18:53:24.381***). The timestamp string is used to restore the backup. See Restore the Domain From the Backup.

Back Up the JRF Domain

A JRF domain has a one-to-one relationship with the RCU schema. After you create a domain using a particular RCU schema, you should back up the JRF database with the RCU schema and the associated wallet. For more information, see JRF Domains.

The back up procedure comprises the following steps:

- Back Up the JRF Schema Database
- Back Up the OPSS Wallet

Back Up the JRF Schema Database

After you create the JRF schema, create a backup of the JRF database. You can use the Oracle Database Backup Cloud Service or any database restore and back up method you prefer. For more information about Oracle Database Backup Cloud Service, see Getting Started with Oracle Database Backup Cloud Service.

Back Up the OPSS Wallet

After you create domain, the WebLogic Kubernetes Operator automatically exports the OPSS wallet and stores it in an introspector `ConfigMap`. The name of the `ConfigMap` follows this pattern: <*domain uid*>`-weblogic-domain-introspect-cm` with the key `ewallet.p12`. Save the OPSS wallet file in a safe, backed-up location immediately after the initial JRF domain gets created. This will allow the secret to be available when you want to recover the domain.

Execute the following commands on the administration instance:

```
mkdir -p /u01/shared/data/domains.backup/<domain>/
```

```
bash opss-wallet.sh -n <domain>-ns -d <domain>  -s -wf /u01/shared/data/
domains.backup/<domain>/ewallet.p12
```

For example:

```
$ mkdir -p /u01/shared/data/domains.backup/pvdomain/
```

```
$ bash /u01/scripts/wls-domain-lifecycle/opss-wallet.sh -n pvdomain-ns -d
pvdomain -s -wf /u01/shared/data/domains.backup/pvdomain/ewallet.p12
@@ Info: Running 'opss-wallet.sh'.
@@ Info: Saving wallet from from configmap 'pvjrf2-weblogic-domain-introspect-
cm' in namespace 'pvjrf2-ns' to file '/u01/shared/data/domains.backup/
pvdomain/ewallet.p12'.
```

Restore the Domain

The restore procedure comprises the following steps:

- Restore the Domain From the Backup
- Restart the Domain After the Restore

Restore the Domain From the Backup

To revert the domain updates, or to recover from the lost domain home directory, restore the domain from a backup copy of the domain home directory.

You can use the following script to restore the domain home from the backup location on `/u01/shared`. The script is available in the `/u01/scripts/utils/restore.sh` on the administration instance.

```
#!/bin/bash -x
if [[ -z $1 || -z $2 ]]; then
  echo "Usage: restore.sh <domain_name> <timestamp>"
  exit 1
fi

domain_name=$1
timestamp=$2
domain_dir=/u01/shared/data/domains/$domain_name
domain_uid_dir=${domain_dir}-uid

backup_dir=/u01/shared/data/domains.backup/$domain_name/$timestamp
backup_domain_dir=${backup_dir}/$domain_name
backup_domain_uid_dir=${backup_dir}/${domain_name}-uid

if ! [ -d ${backup_domain_dir} ]; then
   echo ${backup_domain_dir} not found !
   exit 2
```

```
fi

if ! [ -d ${backup_domain_uid_dir} ]; then
   echo ${backup_domain_uid_dir} not found !
   exit 3
fi

echo "Contents of the domain $domain_name at $domain_dir will be replaced by
the backup contents at $backup_domain_dir"
echo "Contents of the domain $domain_name at $domain_uid_dir will be replaced
by the backup contents at $backup_domain_uid_dir"
read -r -p "Continue? [y/N] " response
case "$response" in
    [yY][eE][sS]|[yY])
        :
        ;;
    *)
        echo "Aborting. No changes were done"
        exit 4
        ;;
esac

echo "restoring $domain_dir"
cp -r $backup_domain_dir $domain_dir

echo "restoring $domain_uid_dir"
cp -r $backup_domain_uid_dir $domain_uid_dir

echo "Domain Restored: $timestamp"
```

For example, to restore from the earlier backup that you created (see Back Up the Domain Home Directory), execute the script using the following command:

```
bash /u01/scripts/utils/restore.sh <domain_name> <backup_timestamp>
```

The output will be as follows:

```
$ bash restore.sh pvdomain 2023-12-04T18:53:24.381

Contents of the domain pvdomain at /u01/shared/data/domains/pvdomain will be
replaced by the backup contents at /u01/shared/data/domains.backup/pvdomain/
2023-12-04T18:53:24.381/pvdomain
Contents of the domain pvdomain at /u01/shared/data/domains/pvdomain-uid will
be replaced by the backup contents at /u01/shared/data/domains.backup/
pvdomain/2023-12-04T18:53:24.381/pvdomain-uid
Continue? [y/N] y
restoring /u01/shared/data/domains/pvdomain
restoring /u01/shared/data/domains/pvdomain-uid
Domain Restored: 2023-12-04T18:53:24.381
```

Restart the Domain After the Restore

After restoring the backup, you should restart the domain in a rolling manner.

1. Execute the following command on the administration instance:

```
bash /u01/scripts/wls-domain-lifecycle/rollDomain.sh -n <domain-namespace>
-d <domain>
```

2. Wait for all the server pods of the domain to terminate, and then restart the domain.

   For example:

```
[opc@wlsoke-admin ~]$ bash /u01/scripts/wls-domain-lifecycle/rollDomain.sh
-n pvdomain-ns -d pvdomain
[2023-12-04T22:53:32.889498983Z][INFO] Patching restartVersion for domain
'pvdomain' to '2'.
domain.weblogic.oracle/pvdomain patched
[2023-12-04T22:53:34.763168023Z][INFO] Successfully patched restartVersion
for domain 'pvdomain'!


k get pods -n pvdomain-ns -w
```

   This is an example of the output:

```
NAME                                READY   STATUS    RESTARTS   AGE
pvdomain-pvdomain-adminserver       1/1     Running   0          4h17m
pvdomain-pvdomain-managed-server1   1/1     Running   0          4h13m
pvdomain-pvdomain-managed-server2   1/1     Running   0          4h10m
```

Back Up the File Storage

Just as you back up and restore a domain, it is also important to back up the entire shared file system, which contains other domains, Jenkins pipeline configurations, pipeline scripts, supporting tools, Jenkins job logs, and so on. You can then use this back up to restore the sytem if the original data gets corrupt or is destroyed.

Ensure that you regularly back up the shared file storage using Snapshot any other preferred mechanism. For information about using Snapshot, see Creating a Snapshot.

6
Delete a Stack

You can delete the stack your created for Oracle WebLogic Server for OKE.

> **Note:**
>
> Before you delete a stack, ensure that you have deleted all the resources of your domain. See Terminate a WebLogic Domain in *Manage WebLogic Domains*.
> If you enabled Verrazzano for the stack, see Terminate a WebLogic Domain in *Manage WebLogic Domains in Verrazzano*.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.
2. Run the following command to delete the resources:

   ```
   /u01/shared/scripts/lcm/delete_resources.sh -p <OCIR Auth Token> -l
   ```

   This deletes the OCIR repos created during provisioning and the OCI Load Balancer associated with the internal and external ingress services.
3. Sign in to the Oracle Cloud Infrastructure Console.
4. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
5. Click the stack you want to delete.
6. Click **Terraform Actions**, and then select **Destroy**.
7. When prompted for confirmation, click **Destroy**.
8. Periodically monitor the progress of the Destroy job until it is finished.

   If an email address is associated with your user profile, you will receive an email notification.

   Ensure that all the resources of the stack are deleted successfully.
9. Click **Delete Stack**.

7

Troubleshoot and Known Issues

Identify common problems in Oracle WebLogic Server for OKE and learn how to diagnose and solve them.

Topics

Troubleshoot a Stack

Identify common problems in a Oracle WebLogic Server for OKE stack and learn how to diagnose to solve them.

Topics

Stack Creation Failed

Troubleshoot a failed Oracle WebLogic Server domain that you created using Oracle WebLogic Server for OKE.

Failed to install WebLogic Operator

Stack provisioning might fail when you create a domain with Oracle WebLogic Server for OKE in a new subnet for an existing VCN due to error in installation of WebLogic Server Kubernetes Operator.

Example message:

```
module.provisioner.null_resource.check_provisioning_status_1  (remote-exec):
<Aug 27, 2020 07:01:31 PM GMT> <INFO>  <install_wls_operator.sh>
<(host:sample-admin.admin1.existingnetwork.oraclevcn.com) -  <WLSOKE-VM-
INFO-0020> :
Installing weblogic operator in namespace [wrjrf8-operator-ns]>
module.provisioner.null_resource.check_provisioning_status_1  (remote-exec):
<Aug 27, 2020
07:02:12 PM GMT> <ERROR>  <install_wls_operator.sh>
<(host:sample-admin.admin1.existingnetwork.oraclevcn.com) -  <WLSOKE-VM-
ERROR-0013> : Error
installing weblogic operator. Exit  code[1]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to create service account

Stack provisioning might fail with `HTTP 409` conflict error if the service account creation fails.

Example message:

```
module.provisioner.null_resource.check_provisioning_status_1 (remote-exec):
HTTP response body: {"kind":"Status","apiVersion":"v1","metadata":
{},"status":"Failure","message":
"Operation cannot be fulfilled on serviceaccounts \"default\": the object has
been modified;
please apply your changes to the latest version and try
again","reason":"Conflict","details":
{"name":"default","kind":"serviceaccounts"}

,"code":409}
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to login to OCIR

Stack provisioning might fail if the docker login to OCI registry is not succcesful.

Example message:

```
[phx.ocir.io]>module.provisioner.null_resource.check_provisioning_status_1
(remote-exec):
<Sep 22, 2020 02:33:46 PM GMT> <ERROR> <docker_init.sh> <(host:sample-
admin.admin.existingnetwork.oraclevcn.com)
- <WLSOKE-VM-ERROR-0003> : Unable to login to custom OCIR
[phx.ocir.io]>module.provisioner.null_resource.check_provisioning_status_1
(remote-exec):
]>module.provisioner.null_resource.check_provisioning_status_1 (remote-exec):
<Sep 22, 2020 02:33:46 PM GMT> <ERROR> <docker_init.py> <(host:sample-
admin.admin.existingnetwork.oraclevcn.com)
- <WLSOKE-VM-ERROR-0020> : Error executing sh /u01/scripts/bootstrap/
docker_init.sh. Exit code [1]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to verify OKE cluster node status

Stack provisioning fails if the OKE cluster worker nodes are inactive when you create the WebLogic domain with Oracle WebLogic Server for OKE.

Example message:

```
<INFO> <oke_worker_status.py>
<(host:sample-admin.nokeadmin.okevcn.oraclevcn.com) - <WLSOKE-VM-INFO-0011> :
Waiting
for the workers nodes to be Active. Retrying...><Dec 17, 2020 04:47:56 PM
GMT> <ERROR>
<markers.py> <(host:sample-admin.okeadmin.okevcn.oraclevcn.com) - <Dec 17,
2020
```

```
16:47:56> - <WLS-OKE-ERROR-003> - Failed to verify oke cluster nodes status.
[Exit code : Status
check timed out]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Nodepools are not Recreated with the Latest Kubernetes Version

Issue: If you upgrade an existing Kubernetes cluster and scale out a nodepool, the new nodes are created with Kubernetes version 1.20 or later.

> **Note:**
>
> This topic is applicable for instances provisioned prior to release 22.1.2.

Workaround:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **create domain**.
3. Click on the pipeline for groovy pipeline definition.
4. Search for `agent-label-jenkins` and replace it with `agent-label`.
5. On the Dashboard page, click **create wls nodepool**, and then complete the step 3 and step 4.
6. On the Dashboard page, click **create base image**, and then complete the step 3 and step 4.
7. Sign in to the Oracle Cloud Infrastructure Console.
8. Go to the recreated nodepools and make a note of the IP address of the nodes which are on Kubernetes version 1.20 or later.
9. Log in to each node as an `opc` user.
10. Run the following command on each node:

```
sudo yum install docker-engine-19.03.11.ol-4.el7
systemctl start docker
```

Load Balancer Creation Failed

After creating a stack, you might encounter an issue where the internal Load Balancer (LB) is missing.

When you run the following command, the internal IP for the LB would is displayed as `<pending>`:

```
kubectl get svc -n <domain-name>-internal
```

Following are the reasons the load balancer creation fails:

1. Lack of quota for the selected LB shapes.

2. Lack of available private IPs in the VCN or subnets selected during provisioning.

Complete the following steps:

1. Check the Status of the Load Balancers
2. Reinstall the Load Balancer

Check the Status of the Load Balancers

You can view the status of the load balancers by checking the load balancer services and the provisioning logs.

Load Balancer Services:

To check the load balancer services, run the following command:

```
kubectl get svc -n wlsoke-ingress-nginx
```

If the output lists any of the load balancer services as `<pending>`, under the `EXTERNAL-IP` column, then the load balancers are not created.

Sample output:

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
okename-internal    LoadBalancer   10.96.185.81    <pending>
443:30618/TCP    11m
```

Provisioning logs:

If the internal load balancer is *not* created successfully, the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error message:

```
<WLSOKE-VM-INFO-0058> : Installing ingress controller charts for jenkins
[ ingress-controller ]>
<WLSOKE-VM-ERROR-0058> : Error installing ingress controller with Helm. Exit
code [1]>
```

And, in the `/u01/logs/provisioning_cmd.out` file, you would see the following error message:

```
<install_ingress_controller.sh>  -  Error: timed out waiting for the condition
```

Reinstall the Load Balancer

After identifying and fixing the cause of the failure, like increased quota for the selected LB shape, you can reinstall the private load balancer in the stack.

1. Run the following command to bounce the Jenkins service:

   ```
   kubectl delete deployment.apps/nginx-ingress-controller -n wlsoke-ingress-
   nginx
   ```

2. Run the following command to delete the load balancer that has an issue:

   ```
   kubectl delete service/<service-prefix>-internal -n wlsoke-ingress-nginx
   ```

3. Run the following command to remove the existing helm release:

   ```
   helm uninstall ingress-controller
   ```

4. Copy the YAML file to the temporary folder:

   ```
   cp /u01/provisioning-data/*.yaml /tmp
   ```

5. Run the following command to install the load balancer:

   ```
   /u01/scripts/bootstrap/install_ingress_controller.sh /tmp/ingress-
   controller-input-values.yaml
   ```

6. Run the following command to verify if load balancer services are created and have an IP addresses:

   ```
   kubectl get svc -n wlsoke-ingress-nginx
   ```

   Sample output:

   ```
   NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
   PORT(S)          AGE
   domain_name-internal   LoadBalancer   10.0.0.1   100.0.0.1
   80:30605/TCP    12m
   ```

Troubleshoot a WebLogic Domain

Learn about the common issues when creating and managing a domain and then how to diagnose to solve them.

Topics

- Unrecognized Arguments When Using the Patching Utility Tool
- Security Checkup Tool Warnings
- Revert the Jenkins Installation to the Original State

Patching Job Fails

When you perform an **apply patch**, **create base image**, or **automatic patching**, the job fails.

> **Note:**
>
> This is applicable only if you use the WebLogic Server version 21.3.3 (September 2021).

To solve this issue, you must update the `pipeline_common.sh` and `apply_latest_psu.sh` scripts:

1. Go to the `/u01/shared/scripts/pipeline/common` location and open the `pipeline_common.sh` file.
2. Remove the lines `347` to `348` and `526` to `530`.
3. Add the following code at line `347`.

```
if [[ "$PATCH_NEW" != *"28186730"* ]]
        then
            patch_array[counter++]=${PATCH_NEW}_${wls_version}".0"
            /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${PATCH_NEW}_${wls_version}".0" --path ${OPATCH_PATCH}
        else
            unzip -j -p ${OPATCH_PATCH} */version.txt > /tmp/version.txt
            version=$(cat /tmp/version.txt)
            /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${PATCH_NEW}_${version} --path ${OPATCH_PATCH}
            SKIP_OPATCH_UPDATE=false
        fi
```

4. Add the following variable at line `314`:

```
local SKIP_OPATCH_UPDATE=true
```

5. Add the following code at line `547`.

```
if [[ "$PATCH_NEW" != *"28186730"* ]]
            then
                patch_array[counter++]=${PATCH_NEW}_${wls_version}".0"
                /u01/shared/tools/imagetool/bin/imagetool.sh cache
addPatch --patchId ${PATCH_NEW}_${wls_version}".0" --path ${OPATCH_PATCH}
            else
                unzip -j -p ${OPATCH_PATCH} */version.txt > /tmp/
version.txt
                version=$(cat /tmp/version.txt)
                /u01/shared/tools/imagetool/bin/imagetool.sh cache
```

```
addPatch --patchId ${PATCH_NEW}_${version} --path ${OPATCH_PATCH}
            fi
```

6. Save and close the `pipeline_common.sh` file.
7. Go to the `/u01/shared/scripts/pipeline/auto-patch/scripts` location and open the `apply_latest_psu.sh` file.
8. Remove the lines `69` to `73`.
9. Add the following code at line `69`.

```
if [[ "$patch_id" != *"28186730"* ]]
      then
         patch_array[counter++]=${patch_id}_${wls_version}".0"
         /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${patch_id}_${wls_version}".0" --path ${patch_dir}/${patch_zip}
      else
        unzip -j -p ${patch_dir}/${patch_zip} */version.txt > /tmp/
version.txt
        version=$(cat /tmp/version.txt)
        /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${patch_id}_${version} --path ${patch_dir}/${patch_zip}
      fi
```

10. Save and close the `apply_latest_psu.sh` file.

Provisioning Fails at a Specific Stage

When you create a domain, the provisioning might fail at the specific stage. After you fix the issue, you must continue to create the domain from the previously failed stage only.

To restart provisioning from the previously failed stage:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **create domain**.
3. Click **Status**.
4. From the **Stage View** page, click on the job number that failed.
5. Click **Restart from Stage**.
6. From **Stage Name**, select the stage that first failed.
7. Click **Run** to run the job from the selected stage.

   After the job is successful, you can access the WebLogic Console. See Access the WebLogic Console.

Unable to View Jenkins UI Input Parameters

You need to approve groovy scripts to view all the parameters in a list.

Issue: At times, the Jenkins UI input parameters in a list are not rendered as you need to approve the scripts.

Workaround:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.

2. Go to **Dashboard** > **Manage Jenkins**.
3. Under Security, click **In-process Script Approval**.
4. Click **Approve** against all the groovy scripts.
   All the parameters are now listed in the pipeline jobs.

Cleanup Resources Manually for a Failed Domain

When the domain creation for a WebLogic domain with domain name, *domain_1*, fails and you create another WebLogic domain with the same name, *domain_1*, this domain creation also fails.

As the resources created for the domain, `domain_1` cannot be deleted using the **terminate domain** job, you must clean up the resources manually for `domain_1` using the following command:

```
/u01/shared/scripts/pipeline/helper-scripts/domain_resource_cleanup.sh
<domain_name>
```

For example, to clean up the resources for *domain_1*, run the following command:

```
/u01/shared/scripts/pipeline/helper-scripts/domain_resource_cleanup.sh domain_1
```

You can now create the WebLogic domain with the domain name, *domain_1*.

Terminate Domain Job Is Stuck at Finish_cleanup Phase

Troubleshoot the problem to terminate a domain created for Oracle WebLogic Server for OKE.

> **Note:**
>
> This topic is applicable for Oracle WebLogic Server for OKE domains created in release 22.2.2 (May, 2022).

Issue:

The terminate domain job gets stuck at the **Finish_cleanup** phase waiting for the pods to come up, and does not terminate the domain.

Workaround:

You must update the Jenkins template details before you terminate a domain.

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **Manage Jenkins**.
3. Under **System Configuration**, click **Manage Nodes and Clouds**, and then click **Configure Clouds**.
4. On the Configure Clouds page, click **Pod Templates**.
5. For *pod-template-jenkins*, click **Pod Template details**.
6. Locate the **Node Selector** field and update the node label to *usage-jenkins=jenkins*.

Introspection Failed when Running Pipeline Jobs

In some instances, the Kubernetes job (`DOMAIN_UID-introspector`) created for the introspection fails. When the initial introspection fails, the operator does not start any WebLogic Server instances. If there are already WebLogic Server instances running, then a failed introspection leaves the existing WebLogic Server instances running without making any changes to the operational state of the domain. The introspection is periodically retried and then eventually timeout with the Domain status indicating the processing failed. To recover from a failed state, you need correct the underlying problem and update the `introspectVersion` or `restartVersion`.

Check the introspector job

If your introspector job failed, then examine the `kubectl describe` of the job and its pod. Also, examine its log, located at `/u01/shared/weblogic-domains/<domain-name>/logs/introspector_script.out`.

For example, assuming your domain UID is `sample-domain1` and your domain namespace is `sample-domain1-ns`, following is a failed introspector job pod among the domain's pods:

```
$ kubectl -n sample-domain1-ns get pods -l weblogic.domainUID=sample-domain1

NAME                                        READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server                 1/1     Running   0          19h
sample-domain1-introspector-v2l7k           0/1     Error     0          75m
sample-domain1-managed-server1              1/1     Running   0          19h
sample-domain1-managed-server2              1/1     Running   0          19h
```

Let us look at the job's describe:

```
$ kubectl -n sample-domain1-ns describe job/sample-domain1-introspector
```

Now, let us look at the job's pod describe, in particular look at its events:

```
$ kubectl -n sample-domain1-ns describe pod/sample-domain1-introspector-v2l7k
```

Finally, let us look at the job's pod's log:

```
$ kubectl -n sample-domain1-ns logs job/sample-domain1-introspector
```

Alternative log command (will have same output as previous):

```
$ kubectl -n sample-domain1-ns logs pod/sample-domain1-introspector-v2l7k
```

A common reason for the introspector job to fail is because of an error in a model file. Following is a sample log output from an introspector job that displays such a failure:

```
...
SEVERE Messages:
    1. WLSDPLY-05007: Model file /u01/wdt/models/model1.yaml,/weblogic-
operator/wdt-config-map/..2020_03_19_15_43_05.993607882/datasource.yaml
contains an
unrecognized section: TYPOresources. The recognized sections are domainInfo,
topology, resources, appDeployments, kubernetes
```

Initiating Rolling Restart

If a model file error references a model file in your `spec.configuration.model.configMap` file, then you can correct the error by redeploying the ConfigMap with a corrected model file and then initiating a domain restart or roll. Similarly, if a model file error references a model file in your model image, then you can correct the error by deploying a corrected image, modifying your Domain YAML file to reference the new image under `spec.image`, and then initiating a domain restart or roll.

To continue to use the pipeline jobs to update the running domain, we need to ensure that the introspector is in `Success` status, which can be achieved by rolling the domain to the previous successful image.

To rollback to the previous previous successful image, run the following command:

```
/u01/shared/scripts/pipeline/common/pipeline_common.sh -i <image_name>
```

Where, *<image_name>* is the image ID in the `prev-domain.yaml` file, located in the backup directory at `/u01/shared/weblogic-domains/<domain_name>/backups`.

> **Note:**
>
> `prev-domain.yaml` is the previous domain that was running before the current job completed.

As the introspector was in the failure status, the domain pods did not restart and would be in the previous image. Once the above function is invoked, introspector succeeds and the pipeline jobs can be reused.

If the error is due to `configmap`, initiate the rolling restart by completing the following steps:

1. Rectify the error in the yaml file.
2. Increment the value of `spec.restartVersion`.
   a. Perform a `kubectl edit domain -n <domain_ns> -o yaml`.
      This opens the yaml file in the VI editor.
   b. Under spec, search for the `restartVersion` flag and increment the value.
3. Save the yaml file.

Run the following command to verify the fix:

```
kubectl get pods -A
```

The age for the pod must not correspond to the time when the update-domain job completed.

Sample output:

```
NAME                                        READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server                 1/1     Running   0          19h
sample-domain1-managed-server1              1/1     Running   0          19h
sample-domain1-managed-server2              1/1     Running   0          19h
```

New Data Source Incorrectly Deployed

This section covers the known issue when you create data sources in your Oracle WebLogic Server for OKE domain.

If the user adds a new data source and deploys the data source to a cluster only, by default, the data source is deployed to both the managed server and the administration server in the cluster.

WebLogic Server Pod Fails to Start

By default, the WebLogic stores are mount to the shared file system, which use Network File System (NFS) version 3 and is disabled. Therefore, the file locks on the different WebLogic stores and may not release if the VM of any node pool in the WebLogic Node pool is abruptly shut down. This is encountered in different scenarios, like, when a VM is stopped, restarted, or terminated, and there are WebLogic pods assigned to the worker node that is being terminated.

Issue: The WebLogic Server Pod (Admin Server or any manged server) fails to start and displays the following error in the WebLogic logs:

```
[Store:280105]The persistent file store "_WLS_myinstance-admin-server" cannot
open file _WLS_<instanceName>-<ServerName>000000.DAT.
```

Workaround:

To solve this issue, complete the following steps:

> **Note:**
>
> Even if you are using an earlier version of WebLogic Server you need to complete these steps.

1. Apply patch 32471832 by using the **apply patch** job, which is available in July 2021 PSUs.
2. For administration and managed server pods in the cluster, update the `domain.yaml` file by adding the `Dweblogic.store.file.LockEnabled=false` parameter.
   Following is an example, where the `Dweblogic.store.file.LockEnabled=false` parameter is added:

```
serverPod:
  env:
  - name: USER_MEM_ARGS
    #Default to G1GC algo
    value: "-XX:+UseContainerSupport -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
  - name: JAVA_OPTIONS
    value: "-Dweblogic.store.file.LockEnabled=false -
Dweblogic.rjvm.allowUnknownHost=true -
Dweblogic.security.SSL.ignoreHostnameVerification=true -
Dweblogic.security.remoteAnonymousRMIT3Enabled=false -
Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

3. Run the following command to apply `domain.yaml`.

```
kubectl -f <domain.yaml-file-path>
```

> **Note:**
>
> If you have created Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied, a few Security warnings are displayed. See About the Security Checkup Tool.

Unable to Access the Console or the Application

Troubleshoot problems accessing the console or the application after the Oracle WebLogic Server for OKE domain is successfully created.

Error accessing the console or the application

If you receive 502 bad gateway error when accessing the Jenkins console and WebLogic Server console, or the application using load balancer, use the `kubectl` command to get the node ports that are used by the system and ensure that these node ports are open for access via the load balancer subnet.

For example:

```
kubectl describe service --all-namespaces | grep -i nodeport
NodePort: http 32062/TCP
NodePort: https 30305/TCP
```

To check port access:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Virtual Cloud Networks**.
3. Select the compartment in which you created the domain.
4. Select the virtual cloud network in which the domain was created.
5. Select the subnet where the WebLogic Server compute instance is provisioned.
6. Select the security list assigned to this subnet.
7. For an Oracle WebLogic Server for OKE cluster using a private and public subnet, make sure the following ingress rules exist:

```
Source: <LB Subnet CIDR>
IP Protocol: TCP
Source Port Range: All
Destination Port Range: 32062


Source: <LB Subnet CIDR>
IP Protocol: TCP
Source Port Range: All
Destination Port Range: 30305
```

For a domain on a private and public subnet, set the `Source` to the CIDR of the load balancer subnet.

Load Balancer Creation Failed

After creating a domain, you might encounter an issue where the external Load Balancer (LB) is missing.

Issue: When you run the following command, the external IP for the LB would is displayed as `<pending>`:

```
kubectl get svc -n <domain-name>-lb-external
```

Sample output:

```
NAMESPACE                  NAME                        TYPE           CLUSTER-
IP      EXTERNAL-IP     PORT(S)                       AGE
wlsoke-ingress-nginx       mydomain-lb-external      LoadBalancer   10.0.0.1
<pending>       80:32148/TCP,443:31808/TCP   27h
```

The load balancer creation fails because there is a lack of available private IPs in the VCN or subnets selected during provisioning.

Workaround: Clean up any unwanted resources to release the IPs.

Jenkins Installation Fails

When you create a Oracle WebLogic Server for OKE instance, Jenkins is installed by installing a Helm release called *jenkins-oke*. During provisioning, Jenkins installation may fail, but provisioning is not stopped, because Jenkins can be installed after provisioning. This section explains how to install Jenkins manually, if Jenkins installation has failed during provisioning.

Check if Jenkins Install Failed During Provisioning

You can know if the Jenkins install failed by trying to access the Jenkins console, checking the provisioning logs, and checking the Kubernetes resources (pods, services, and so on) under the `jenkins-ns` namespace.

Access the Jenkins console:

Access the Jenkins console, as described in Access the Jenkins Console.

If you are not able to access the console, then continue to the next section to check the logs.

Provisioning logs:

If Jenkins is *not* installed successfully, then the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error:

```
<WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<WLSOKE-VM-ERROR-0052> : Error installing jenkins charts. Exit code[1]>
```

And, you would see the details of the failure in the `/u01/logs/provisioning_cmd.out` file.

Kubernetes resources:

To check the Kubernetes resources in the `jenkins-ns` namespace, run the following command:

```
kubectl get all -n jenkins-ns
```

Following is a sample output, where Jenkins was installed correctly:

```
NAME                                      READY   STATUS    RESTARTS   AGE
pod/jenkins-deployment-5bb8596b9-abcd   1/1     Running   0          26m

NAME                       TYPE        CLUSTER-IP    EXTERNAL-IP
PORT(S)              AGE
service/jenkins-service   ClusterIP   10.0.0.1   <none>        8080/
TCP,50000/TCP   26m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/jenkins-deployment   1/1     1            1           26m

NAME                                           DESIRED   CURRENT   READY
AGE
replicaset.apps/jenkins-deployment-5bb55586b9   1         1         1
26m
```

Install Jenkins Manually

After identifying and fixing the cause of the failure, install Jenkins in your instance.

1. Check if the `provisioning_metadata.properties` file exists, at the `/u01/shared/weblogic-domains/<domain>` directory.

   Does the `provisioning_metadata.properties` file exist?

   - **Yes**: Continue with the next step.
   - **No**: Run the following command:

     ```
     python /u01/scripts/metadata/provisioning_metadata.py
     ```

     Continue with the next step.

2. Run the following command to remove the existing helm release:

   ```
   helm uninstall jenkins-oke
   ```

3. Run the following command to install Jenkins:

   ```
   /u01/scripts/bootstrap/install_jenkins.sh  /u01/provisioning-data/jenkins-inputs.yaml
   ```

   Where, `jenkins-inputs.yaml` file contains the required variables.

Sample Output:

```
<Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.py>
<(host:host_name) - updated /u01/provisioning-data/jenkins-inputs.yaml>
<Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0098> : Creating configmap [wlsoke-
metadata-configmap]>
<Nov 23, 2020 05:10:09 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<Nov 23, 2020 05:10:22 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0057> : Successfully installed jenkins
in namespace [ jenkins-ns ]>
```

You have successfully installed the Jenkins console. Try accessing the Jenkins console, as described in Access the Jenkins Console.

T3 RMI Communication Between Domains Fails

You might encounter a T3 communication error between domains in Oracle WebLogic Server for OKE.

Issue:

When you try to establish an RMI communication between two domains, *Domain A* and *Domain B*, in different namespaces within the same cluster, using the T3 protocol, the connection fails.

Workaround:

You must set up the WebLogic custom channel on *Domain B*. To configure the WebLogic custom channel, see Configuring a WebLogic custom channel in WebLogic Kubernetes Operator documentation.

Before you set up the WebLogic custom channel, perform the following steps:

1. Run the following command to obtain the cluster service names:

   ```
   kubectl get svc -n <domain_namespace>
   ```

   Sample output:

   ```
   NAMESPACE            NAME                                TYPE
   CLUSTER-IP    EXTERNAL-IP   PORT (S)            AGE
   domainB-ns           domainB-cluster-domainB-cluster   ClusterIP
   10.96.37.63    <none>       7999/TCP,8001/TCP   3d1h
   domainB-ns           domainB-cluster-domainB-cluster   ClusterIP
   10.96.37.63    <none>       7999/TCP,8001/TCP   3d1h
   domainB-ns           domainB-domainB-adminserver       ClusterIP
   None           <none>       30012/TCP,7001/TCP  10d
   domainB-ns           domainB-domainB-managed-server1   ClusterIP
   None           <none>       7999/TCP,8001/TCP   3d1h
   domainB-ns           domainB-domainB-managed-server2   ClusterIP
   None           <none>       7999/TCP,8001/TCP   3d1h
   ```

2. Use the cluster service name and domain namespace from step 1 to obtain the cluster address.

The cluster address format is:

```
t3://<name of the cluster service>.<domain namespace>:ListenPort
```

For example:

```
t3://domainB-ns domainB-cluster-domainB-cluster.domainB-ns:7999
```

3. Run the **update-domain** job on *Domain B* using the following model yaml file. See Update a Domain Configuration.
In the model Yaml file, under the `NetworkAccessPoint` section, specify the cluster address from step 2.

   Example of a model Yaml file.

```
topology:
   Cluster:
      '@@ENV:OKE_DOMAIN_NAME@@-cluster':
         WeblogicPluginEnabled: true
         DynamicServers:
            ServerNamePrefix: '@@ENV:OKE_DOMAIN_NAME@@-managed-server'
            MaxDynamicClusterSize: 9
            CalculatedListenPorts: false
            ServerTemplate: '@@ENV:OKE_DOMAIN_NAME@@-cluster-template'
            DynamicClusterSize: 9
   Server:
      '@@ENV:OKE_DOMAIN_NAME@@-adminserver':
         RestartDelaySeconds: 10
         GracefulShutdownTimeout: 120
         RestartMax: 20
         NetworkAccessPoint:
            T3Channel:
              PublicPort: 30012
              ListenPort: 30012
         SSL:
            OutboundCertificateValidation: BuiltinSSLValidationOnly
            HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
            InboundCertificateValidation: BuiltinSSLValidationOnly
         WebServer:
            HttpsKeepAliveSecs: 310
            KeepAliveSecs: 310
         ListenPort: 7001
   ServerTemplate:
      '@@ENV:OKE_DOMAIN_NAME@@-cluster-template':
         ListenPort: 8001
         Cluster: '@@ENV:OKE_DOMAIN_NAME@@-cluster'
         SSL:
            ListenPort: 8100
            OutboundCertificateValidation: BuiltinSSLValidationOnly
            HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
            InboundCertificateValidation: BuiltinSSLValidationOnly
         WebServer:
            HttpsKeepAliveSecs: 310
```

```
            KeepAliveSecs: 310
         NetworkAccessPoint:
            MyT3Channel:
              Protocol: 't3'
              ListenPort: 7999
              PublicAddress: '@@ENV:DOMAIN_UID@@-@@ENV:DOMAIN_UID@@-
managed-server${id}.@@ENV:NAMESPACE@@'
              HttpEnabledForThisProtocol: true
              TunnelingEnabled: true
              OutboundEnabled: false
              Enabled: true
              ClusterAddress: t3://<name of the cluster service>.<domain
namespace>:ListenPort
              TwoWaySSLEnabled: false
              ClientCertificateEnforced: false
```

Unrecognized Arguments When Using the Patching Utility Tool

When you run the patching utility tool with some of the documented arguments, you see the `unrecognized arguments` message.

Issue:

Run the patching utility tool to list latest patches and download latest patches using `patch-utils` with the following arguments:

```
#List patches
patch-utils list -L
#Download latest patches
patch-utils download -L -p /tmp/<Location to download>
```

The following message is displayed:

```
usage: patch-utils <action> [options]
patch-utils: error: unrecognized arguments:
```

The listed arguments correspond to latest features added to the patching utility tool for Oracle WebLogic Server for OKEinstances created after December 14, 2022 (22.4.3). So, if you are using Oracle WebLogic Server for OKE instances created before release December 14, 2022, you see the `unrecognized arguments` message.

Workaround:

Run `patch-utils upgrade` to upgrade the patching tool, if you are using the latest features of the patching utility tool for your existing instances (created before release December 14, 2022). See Upgrade Patching Tool.

Security Checkup Tool Warnings

Learn about the security check warnings that are displayed in the Oracle WebLogic Server Administration console and how to troubleshoot them.

At the top of the WebLogic Server Administration console, the message `Security warnings detected. Click here to view the report and recommended remedies` is displayed for

Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied.

When you click the message, a list of security warnings are displayed as listed in the following table.

The warning messages listed in the table are examples.

Security Warnings

Warning Message	Resolution
`The configuration for key stores for this server are set to Demo Identity and Demo Trust. Trust Demo certificates are not supported in production mode domains.`	Configure the identity and trust keystores for each server and the name of the certificate in the identity keystore that the server uses for SSL communication. See Configure Keystore Attributes for Identity and Trust. **Note:** This warning is displayed for Oracle WebLogic Server for OKE instances created after October 20, 2021, or the instances on which the October PSUs are applied.
`SSL hostname verification is disabled by the SSL configuration.`	Review your applications before you make any changes to address these SSL host name security warnings. For applications that connect to SSL endpoints with a host name in the certificate, which does not match the local machine's host name, the connection fails if you configure the BEA host name verifier in Oracle WebLogic Server. For applications that connect to Oracle provided endpoints such as Oracle Identity Cloud Service (for example,`*.identity.oraclecloud.com`), the connection fails if you did not configure the wildcard host name verifier or a custom host name verifier that accepts wildcard host names. If you are not sure of the SSL configuration settings you should configure to address the warning, Oracle recommends that you configure the wildcard host name verifier. You see the SSL host name verification warnings in case of existing Oracle WebLogic Server for OKE instances (created before July 20, 2021). To address this warning, you must configure SSL with host name verifier. See Configure SSL with host name verifier.
`Production mode is enabled but the file or directory <directory_name>/ startWebLogic.sh is insecure since its permission is not a minimum of umask 027`	Run the following command in the administration server as `oracle` user: `chmod 640 /u01/data/domains/<domain_name>/bin`
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Set the java properties for anonymous RMI T3 and IIOP requests during server start up. See Set the Java Properties.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OKE instances created after July 20, 2021, though the java properties to disable anonymous requests for preventing anonymous RMI access are configured, the warnings still appear. This is a known issue in Oracle WebLogic Server.

Set the Java Properties

To set the java properties for anonymous RMI T3 and IIOP requests:

1. Edit the `domain.yaml` located in `/u01/shared/weblogic-domains/<domain_name>/domain.yaml` for all instances of `serverPod` definitions as follows:

```
serverPod:
      env:
      - name: USER_MEM_ARGS
        #admin server memory is explicitly set to min of 256m and max of
512m and GC algo is G1GC
        value: "-Xms256m -Xmx512m -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
      - name: JAVA_OPTIONS
        value: "-Dweblogic.store.file.LockEnabled=false
        -Dweblogic.rjvm.allowUnknownHost=true
        -Dweblogic.security.remoteAnonymousRMIT3Enabled=false
        -Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

2. Apply the `domain.yaml` using the `kubectl` command:

```
kubectl -f <path_to_domain.yaml>
```

Configure Keystore Attributes for Identity and Trust

To configure the identity and trust keystore files and the name of the certificate in the identity keystore in the WebLogic Server Administration console:

1. Locate the **Change Center** and click **Lock & Edit** to lock the editable configuration hierarchy for the domain.
2. Under **Domain structure**, select **Environment** and then select **Servers**.
3. In the Servers table, select the server you want to configure.
4. On the **Configuration** tab, click **Keystores**, and then click **Change**.
5. Select *Custom Identity and Custom Trust*, and then click **Save**.
6. Under **Identity**, provide the following details:
   a. Enter the full path of your identity keystore.

      For example: `/u01/data/keystores/identity.jks`

   b. For **Custom Identity Keystore Type**, enter *JKS*.

   c. For **Custom Identity Keystore Passphrase**, enter your keystore password. Enter the same value for **Confirm Custom Identity Keystore Passphrase**.
7. Under **Trust**, provide the following details:
   a. Enter the full path of your identity keystore.

      For example, `/u01/data/keystores/trust.jks`

   b. For **Custom Trust Keystore Type**, enter *JKS*.

c. For **Custom Trust Keystore Passphrase**, enter your keystore password. Enter the same value for **Confirm Custom Trust Keystore Passphrase**.

8. Click **Save**.
9. Click the **SSL** tab.
10. Under **Identity**, provide the following details:
    a. For **Private Key Alias**, enter the name of the certificate (private key) in the identitykeystore, *server_cert*.
    b. For **Private Key Passphrase**, enter the password for this certificate in the keystore. Enter the same value for **Confirm Private Key Passphrase**.

       By default, the password for the certificate is the same as the identity keystore password.
11. Click **Save**.

    After saving the changes, return to **Change Center** and click **Activate Changes**.
12. Repeat steps 3 to 9 to configure each server in the domain.

Revert the Jenkins Installation to the Original State

When you create a Oracle WebLogic Server for OKE instance, it also provisions the Jenkins primary server on a pod in the Kubernetes cluster by installing a Helm release called *jenkins-oke*. Oracle WebLogic Server for OKE uses the Jenkins pipeline to update the domain to deploy applications, libraries, and resources; apply JDK and WebLogic Server patches; and update an existing image. If you encounter any errors in these tasks, you can revert the Jenkins installation to its original state.

> **Note:**
>
> Oracle recommends you to not update the Jenkins plug-ins using the Jenkins console (at `http://internal_lb_ip/jenkins/pluginManager/`) because some plug-in updates may break the current Oracle WebLogic Server for OKE pipeline jobs due to incompatibilities.

Here are the steps to revert the Jenkins installation to the original state. These steps revert any plug-in updates while preserving the pipeline job logs.

1. Move all the Jenkins files and directories from the Jenkins Home directory to another directory (with a new name) as shown in the following example:

   ```
   mv /u01/shared/var/jenkins_home /u01/shared/var/jenkins_home.old
   ```

2. Create a new Jenkins Home directory.

   ```
   mkdir /u01/shared/var/jenkins_home
   ```

3. Copy the `jobs` directory that you moved in Step 1 (to `jenkins_home.old`) to the newly created Jenkins Home directory (`jenkins_home`) you created in Step 2.

   ```
   cp -R /u01/shared/var/jenkins_home.old/jobs /u01/shared/var/jenkins_home/
   ```

4. Reinstall Jenkins manually. For instructions, see Install Jenkins Manually.

5. Sign in to the Jenkins console for your stack and create the first Jenkins administration user again. See Access the Jenkins Console.
6. If everything appears fine, you may delete the `/u01/shared/var/jenkins_home.old` directory.

Troubleshoot a WebLogic Domain in Verrazzano

Learn about the common issues when creating and managing a domain in Verrazzano and then how to diagnose to solve them.

The following topics are also applicable for Oracle WebLogic Server for OKE domains with Verrrazzano.

Topics

Patching Job Fails

When you perform an **apply patch**, **create base image**, or **automatic patching**, the job fails.

> **Note:**
>
> This is applicable only if you use the WebLogic Server version 21.3.3 (September 2021).

To solve this issue, you must update the `pipeline_common.sh` and `apply_latest_psu.sh` scripts:

1. Go to the `/u01/shared/scripts/pipeline/common` location and open the `pipeline_common.sh` file.
2. Remove the lines `347` to `348` and `526` to `530`.

3. Add the following code at line `347`.

```
if [[ "$PATCH_NEW" != *"28186730"* ]]
        then
            patch_array[counter++]=${PATCH_NEW}_${wls_version}".0"
            /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${PATCH_NEW}_${wls_version}".0" --path ${OPATCH_PATCH}
        else
            unzip -j -p ${OPATCH_PATCH} */version.txt > /tmp/version.txt
            version=$(cat /tmp/version.txt)
            /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${PATCH_NEW}_${version} --path ${OPATCH_PATCH}
            SKIP_OPATCH_UPDATE=false
        fi
```

4. Add the following variable at line `314`:

```
local SKIP_OPATCH_UPDATE=true
```

5. Add the following code at line `547`.

```
if [[ "$PATCH_NEW" != *"28186730"* ]]
            then
                patch_array[counter++]=${PATCH_NEW}_${wls_version}".0"
                /u01/shared/tools/imagetool/bin/imagetool.sh cache
addPatch --patchId ${PATCH_NEW}_${wls_version}".0" --path ${OPATCH_PATCH}
            else
                unzip -j -p ${OPATCH_PATCH} */version.txt > /tmp/
version.txt
                version=$(cat /tmp/version.txt)
                /u01/shared/tools/imagetool/bin/imagetool.sh cache
addPatch --patchId ${PATCH_NEW}_${version} --path ${OPATCH_PATCH}
            fi
```

6. Save and close the `pipeline_common.sh` file.
7. Go to the `/u01/shared/scripts/pipeline/auto-patch/scripts` location and open the `apply_latest_psu.sh` file.
8. Remove the lines `69` to `73`.
9. Add the following code at line `69`.

```
if [[ "$patch_id" != *"28186730"* ]]
      then
         patch_array[counter++]=${patch_id}_${wls_version}".0"
         /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${patch_id}_${wls_version}".0" --path ${patch_dir}/${patch_zip}
      else
        unzip -j -p ${patch_dir}/${patch_zip} */version.txt > /tmp/
version.txt
        version=$(cat /tmp/version.txt)
        /u01/shared/tools/imagetool/bin/imagetool.sh cache addPatch --
patchId ${patch_id}_${version} --path ${patch_dir}/${patch_zip}
      fi
```

10. Save and close the `apply_latest_psu.sh` file.

Provisioning Fails at a Specific Stage

When you create a domain, the provisioning might fail at the specific stage. After you fix the issue, you must continue to create the domain from the previously failed stage only.

To restart provisioning from the previously failed stage:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. On the Dashboard page, click **create domain**.
3. Click **Status**.
4. From the **Stage View** page, click on the job number that failed.
5. Click **Restart from Stage**.
6. From **Stage Name**, select the stage that first failed.
7. Click **Run** to run the job from the selected stage.

   After the job is successful, you can access the WebLogic Console. See Access the WebLogic Console.

Unable to View Jenkins UI Input Parameters

You need to approve groovy scripts to view all the parameters in a list.

Issue: At times, the Jenkins UI input parameters in a list are not rendered as you need to approve the scripts.

Workaround:

1. Sign in to the Jenkins console for your stack. See Access the Jenkins Console.
2. Go to **Dashboard** > **Manage Jenkins**.
3. Under Security, click **In-process Script Approval**.
4. Click **Approve** against all the groovy scripts.
   All the parameters are now listed in the pipeline jobs.

Cleanup Resources Manually for a Failed Domain

When the domain creation for a WebLogic domain with domain name, *domain_1*, fails and you create another WebLogic domain with the same name, *domain_1*, this domain creation also fails.

As the resources created for the domain, `domain_1` cannot be deleted using the **terminate domain** job, you must clean up the resources manually for `domain_1` using the following command:

```
/u01/shared/scripts/pipeline/helper-scripts/domain_resource_cleanup.sh
<domain_name>
```

For example, to clean up the resources for *domain_1*, run the following command:

```
/u01/shared/scripts/pipeline/helper-scripts/domain_resource_cleanup.sh domain_1
```

You can now create the WebLogic domain with the domain name, *domain_1*.

Verrazzano Installation Failed

Issue:

When you create a stack with Verrazzano that has shape lower than `VM.Standard2.4` and less than three nodes for the node pool, the Verrazzano installation fails.

Workaround:

Run the reinstall script to clean up the existing node pool and recreate a new node pool with the provided shape and nodes, and reinstall Verrazzano.

The reinstall script is located at: `/u01/scripts/utils`

Run the following command:

```
sh verrazzano_reinstall_util.sh <node_pool_shape> <node_pool_size>
```

For example:

```
sh verrazzano_reinstall_util.sh VM.Standard2.4 3
```

Unable to Access the Verrazzano Console

Troubleshoot the problem accessing the Verrazzano console for an Oracle WebLogic Server for OKE stack with Verrazzano configuration.

Issue:

For an Oracle WebLogic Server for OKE stack with Verrazzano configuration that uses LetsEncrypt certificates, you receive a connection failure message when accessing the Verrazzano console.

Workaround:

Verify the pods running in the cluster and then restart all the authproxy pods.

```
kubectl get pods -A | grep auth
kubectl delete pod verrazzano-authproxy_<pod_name> -n verrazzano-system
```

Introspection Failed when Running Pipeline Jobs

In some instances, the Kubernetes job (`DOMAIN_UID-introspector`) created for the introspection fails. When the initial introspection fails, the operator does not start any WebLogic Server instances. If there are already WebLogic Server instances running, then a failed introspection leaves the existing WebLogic Server instances running without making any changes to the operational state of the domain. The introspection is periodically retried and then eventually timeout with the Domain status indicating the processing failed. To recover from a failed state, you need correct the underlying problem and update the `introspectVersion` or `restartVersion`.

Check the introspector job

If your introspector job failed, then examine the `kubectl describe` of the job and its pod. Also, examine its log, located at `/u01/shared/weblogic-domains/<domain-name>/logs/introspector_script.out`.

For example, assuming your domain UID is `sample-domain1` and your domain namespace is `sample-domain1-ns`, following is a failed introspector job pod among the domain's pods:

```
$ kubectl -n sample-domain1-ns get pods -l weblogic.domainUID=sample-domain1

NAME                                        READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server                 1/1     Running   0          19h
sample-domain1-introspector-v217k           0/1     Error     0          75m
sample-domain1-managed-server1              1/1     Running   0          19h
sample-domain1-managed-server2              1/1     Running   0          19h
```

Let us look at the job's describe:

```
$ kubectl -n sample-domain1-ns describe job/sample-domain1-introspector
```

Now, let us look at the job's pod describe, in particular look at its events:

```
$ kubectl -n sample-domain1-ns describe pod/sample-domain1-introspector-v217k
```

Finally, let us look at the job's pod's log:

```
$ kubectl -n sample-domain1-ns logs job/sample-domain1-introspector
```

Alternative log command (will have same output as previous):

```
$ kubectl -n sample-domain1-ns logs pod/sample-domain1-introspector-v217k
```

A common reason for the introspector job to fail is because of an error in a model file. Following is a sample log output from an introspector job that displays such a failure:

```
...
SEVERE Messages:
    1. WLSDPLY-05007: Model file /u01/wdt/models/model1.yaml,/weblogic-
operator/wdt-config-map/..2020_03_19_15_43_05.993607882/datasource.yaml
contains an
unrecognized section: TYPOresources. The recognized sections are domainInfo,
topology, resources, appDeployments, kubernetes
```

Initiating Rolling Restart

If a model file error references a model file in your `spec.configuration.model.configMap` file, then you can correct the error by redeploying the ConfigMap with a corrected model file and then initiating a domain restart or roll. Similarly, if a model file error references a model file in your model image, then you can correct the error by deploying a corrected image, modifying your Domain YAML file to reference the new image under `spec.image`, and then initiating a domain restart or roll.

To continue to use the pipeline jobs to update the running domain, we need to ensure that the introspector is in `Success` status, which can be achieved by rolling the domain to the previous successful image.

To rollback to the previous previous successful image, run the following command:

```
/u01/shared/scripts/pipeline/common/pipeline_common.sh -i <image_name>
```

Where, `<image_name>` is the image ID in the `prev-domain.yaml` file, located in the backup directory at `/u01/shared/weblogic-domains/<domain_name>/backups`.

> **Note:**
>
> `prev-domain.yaml` is the previous domain that was running before the current job completed.

As the introspector was in the failure status, the domain pods did not restart and would be in the previous image. Once the above function is invoked, introspector succeeds and the pipeline jobs can be reused.

If the error is due to `configmap`, initiate the rolling restart by completing the following steps:

1. Rectify the error in the yaml file.
2. Increment the value of `spec.restartVersion`.
   a. Perform a `kubectl edit domain -n <domain_ns> -o yaml`.
      This opens the yaml file in the VI editor.
   b. Under spec, search for the `restartVersion` flag and increment the value.
3. Save the yaml file.

Run the following command to verify the fix:

```
kubectl get pods -A
```

The age for the pod must not correspond to the time when the update-domain job completed.

Sample output:

```
NAME                                     READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server              1/1     Running   0          19h
sample-domain1-managed-server1           1/1     Running   0          19h
sample-domain1-managed-server2           1/1     Running   0          19h
```

New Data Source Incorrectly Deployed

This section covers the known issue when you create data sources in your Oracle WebLogic Server for OKE domain.

If the user adds a new data source and deploys the data source to a cluster only, by default, the data source is deployed to both the managed server and the administration server in the cluster.

WebLogic Server Pod Fails to Start

By default, the WebLogic stores are mount to the shared file system, which use Network File System (NFS) version 3 and is disabled. Therefore, the file locks on the different WebLogic stores and may not release if the VM of any node pool in the WebLogic Node pool is abruptly shut down. This is encountered in different scenarios, like, when a VM is stopped, restarted, or

terminated, and there are WebLogic pods assigned to the worker node that is being terminated.

Issue: The WebLogic Server Pod (Admin Server or any manged server) fails to start and displays the following error in the WebLogic logs:

```
[Store:280105]The persistent file store "_WLS_myinstance-admin-server" cannot
open file _WLS_<instanceName>-<ServerName>000000.DAT.
```

Workaround:

To solve this issue, complete the following steps:

> **Note:**
>
> Even if you are using an earlier version of WebLogic Server you need to complete these steps.

1. Apply patch 32471832 by using the **apply patch** job, which is available in July 2021 PSUs.
2. For administration and managed server pods in the cluster, update the `domain.yaml` file by adding the `Dweblogic.store.file.LockEnabled=false` parameter.
   Following is an example, where the `Dweblogic.store.file.LockEnabled=false` parameter is added:

```
serverPod:
  env:
  - name: USER_MEM_ARGS
    #Default to G1GC algo
    value: "-XX:+UseContainerSupport -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
  - name: JAVA_OPTIONS
    value: "-Dweblogic.store.file.LockEnabled=false -
Dweblogic.rjvm.allowUnknownHost=true -
Dweblogic.security.SSL.ignoreHostnameVerification=true -
Dweblogic.security.remoteAnonymousRMIT3Enabled=false -
Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

3. Run the following command to apply `domain.yaml`.

```
kubectl -f <domain.yaml-file-path>
```

> **Note:**
>
> If you have created Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied, a few Security warnings are displayed. See About the Security Checkup Tool.

Load Balancer Creation Failed

After creating a domain, you might encounter an issue where the external Load Balancer (LB) is missing.

Issue: When you run the following command, the external IP for the LB would is displayed as `<pending>`:

```
kubectl get svc -n <domain-name>-lb-external
```

Sample output:

```
NAMESPACE                  NAME                        TYPE           CLUSTER-
IP      EXTERNAL-IP     PORT(S)                       AGE
wlsoke-ingress-nginx      mydomain-lb-external      LoadBalancer   10.0.0.1
<pending>       80:32148/TCP,443:31808/TCP   27h
```

The load balancer creation fails because there is a lack of available private IPs in the VCN or subnets selected during provisioning.

Workaround: Clean up any unwanted resources to release the IPs.

Jenkins Installation Fails

When you create a Oracle WebLogic Server for OKE instance, Jenkins is installed by installing a Helm release called *jenkins-oke*. During provisioning, Jenkins installation may fail, but provisioning is not stopped, because Jenkins can be installed after provisioning. This section explains how to install Jenkins manually, if Jenkins installation has failed during provisioning.

Check if Jenkins Install Failed during Provisioning

You can know if the Jenkins install failed by trying to access the Jenkins console, checking the provisioning logs, and checking the Kubernetes resources (pods, services, and so on) under the `jenkins-ns` namespace.

Access the Jenkins console:

Try accessing the Jenkins console, as described in Access the Jenkins Console.

If you are not able to access the console, then continue to the next section to check the logs.

Provisioning logs:

If Jenkins is *not* installed successfully, then the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error:

```
<WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<WLSOKE-VM-ERROR-0052> : Error installing jenkins charts. Exit code[1]>
```

And, you would see the details of the failure in the `/u01/logs/provisioning_cmd.out` file.

Kubernetes resources:

To check the Kubernetes resources in the `jenkins-ns` namespace, run the following command:

```
kubectl get all -n jenkins-ns
```

Following is a sample output, where Jenkins was installed correctly:

```
NAME                                      READY   STATUS    RESTARTS   AGE
pod/jenkins-deployment-5bb8596b9-abcd   1/1     Running   0          26m

NAME                       TYPE        CLUSTER-IP    EXTERNAL-IP
PORT(S)             AGE
service/jenkins-service   ClusterIP   10.0.0.1   <none>        8080/
TCP,50000/TCP   26m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/jenkins-deployment   1/1     1            1           26m

NAME                                            DESIRED   CURRENT   READY
AGE
replicaset.apps/jenkins-deployment-5bb55586b9   1         1         1
26m
```

Install Jenkins Manually

After identifying and fixing the cause of the failure, install Jenkins in your instance.

1. Check if the `provisioning_metadata.properties` file exists, at the `/u01/shared/weblogic-domains/<domain>` directory.
   Does the `provisioning_metadata.properties` file exist?
   - **Yes**: Continue with the next step.
   - **No**: Run the following command:

     ```
     python /u01/scripts/metadata/provisioning_metadata.py
     ```

     Continue with the next step.
2. Run the following command to remove the existing helm release:

   ```
   helm uninstall jenkins-oke
   ```

3. Run the following command to install Jenkins:

   ```
   /u01/scripts/bootstrap/install_jenkins.sh  /u01/provisioning-data/jenkins-inputs.yaml
   ```

   Where, `jenkins-inputs.yaml` file contains the required variables.

   Sample Output:

   ```
   <Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.py>
   <(host:host_name) - updated /u01/provisioning-data/jenkins-inputs.yaml>
   <Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.sh>
   ```

```
<(host:host_name) - <WLSOKE-VM-INFO-0098> : Creating configmap [wlsoke-
metadata-configmap]>
<Nov 23, 2020 05:10:09 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<Nov 23, 2020 05:10:22 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0057> : Successfully installed jenkins
in namespace [ jenkins-ns ]>
```

You have successfully installed the Jenkins console. Try accessing the Jenkins console, as described in Access the Jenkins Console.

T3 RMI Communication Between Domains Fails

You might encounter a T3 communication error between domains in Oracle WebLogic Server for OKE.

Issue:

When you try to establish an RMI communication between two domains, *Domain A* and *Domain B*, in different namespaces within the same cluster, using the T3 protocol, the connection fails.

Workaround:

You must set up the WebLogic custom channel on *Domain B*. To configure the WebLogic custom channel, see Configuring a WebLogic custom channel in WebLogic Kubernetes Operator documentation.

Before you set up the WebLogic custom channel, perform the following steps:

1. Run the following command to obtain the cluster service names:

   ```
   kubectl get svc -n <domain_namespace>
   ```

   Sample output:

   ```
   NAMESPACE            NAME                                TYPE
   CLUSTER-IP    EXTERNAL-IP   PORT (S)            AGE
   domainB-ns           domainB-cluster-domainB-cluster   ClusterIP
   10.96.37.63    <none>       7999/TCP,8001/TCP   3d1h
   domainB-ns           domainB-cluster-domainB-cluster   ClusterIP
   10.96.37.63    <none>       7999/TCP,8001/TCP   3d1h
   domainB-ns           domainB-domainB-adminserver       ClusterIP
   None           <none>       30012/TCP,7001/TCP  10d
   domainB-ns           domainB-domainB-managed-server1   ClusterIP
   None           <none>       7999/TCP,8001/TCP   3d1h
   domainB-ns           domainB-domainB-managed-server2   ClusterIP
   None           <none>       7999/TCP,8001/TCP   3d1h
   ```

2. Use the cluster service name and domain namespace from step 1 to obtain the cluster address.
   The cluster address format is:

   ```
   t3://<name of the cluster service>.<domain namespace>:ListenPort
   ```

For example:

```
t3://domainB-ns domainB-cluster-domainB-cluster.domainB-ns:7999
```

3. Run the **update-domain** job on *Domain B* using the following model yaml file. See Update a Domain Configuration.
   In the model Yaml file, under the `NetworkAccessPoint` section, specify the cluster address from step 2.

   Example of a model Yaml file.

```
topology:
   Cluster:
      '@@ENV:OKE_DOMAIN_NAME@@-cluster':
         WeblogicPluginEnabled: true
         DynamicServers:
            ServerNamePrefix: '@@ENV:OKE_DOMAIN_NAME@@-managed-server'
            MaxDynamicClusterSize: 9
            CalculatedListenPorts: false
            ServerTemplate: '@@ENV:OKE_DOMAIN_NAME@@-cluster-template'
            DynamicClusterSize: 9
   Server:
      '@@ENV:OKE_DOMAIN_NAME@@-adminserver':
         RestartDelaySeconds: 10
         GracefulShutdownTimeout: 120
         RestartMax: 20
         NetworkAccessPoint:
            T3Channel:
              PublicPort: 30012
              ListenPort: 30012
         SSL:
            OutboundCertificateValidation: BuiltinSSLValidationOnly
            HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
            InboundCertificateValidation: BuiltinSSLValidationOnly
         WebServer:
            HttpsKeepAliveSecs: 310
            KeepAliveSecs: 310
         ListenPort: 7001
   ServerTemplate:
      '@@ENV:OKE_DOMAIN_NAME@@-cluster-template':
         ListenPort: 8001
         Cluster: '@@ENV:OKE_DOMAIN_NAME@@-cluster'
         SSL:
            ListenPort: 8100
            OutboundCertificateValidation: BuiltinSSLValidationOnly
            HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
            InboundCertificateValidation: BuiltinSSLValidationOnly
         WebServer:
            HttpsKeepAliveSecs: 310
            KeepAliveSecs: 310
         NetworkAccessPoint:
            MyT3Channel:
              Protocol: 't3'
              ListenPort: 7999
```

```
                PublicAddress: '@@ENV:DOMAIN_UID@@-@@ENV:DOMAIN_UID@@-
managed-server${id}.@@ENV:NAMESPACE@@'
                HttpEnabledForThisProtocol: true
                TunnelingEnabled: true
                OutboundEnabled: false
                Enabled: true
                ClusterAddress: t3://<name of the cluster service>.<domain
namespace>:ListenPort
                TwoWaySSLEnabled: false
                ClientCertificateEnforced: false
```

Unrecognized Arguments When Using the Patching Utility Tool

When you run the patching utility tool with some of the documented arguments, you see the `unrecognized arguments` message.

Issue:

Run the patching utility tool to list latest patches and download latest patches using `patch-utils` with the following arguments:

```
#List patches
patch-utils list -L
#Download latest patches
patch-utils download -L -p /tmp/<Location to download>
```

The following message is displayed:

```
usage: patch-utils <action> [options]
patch-utils: error: unrecognized arguments:
```

The listed arguments correspond to latest features added to the patching utility tool for Oracle WebLogic Server for OKEinstances created after December 14, 2022 (22.4.3). So, if you are using Oracle WebLogic Server for OKE instances created before release December 14, 2022, you see the `unrecognized arguments` message.

Workaround:

Run `patch-utils upgrade` to upgrade the patching tool, if you are using the latest features of the patching utility tool for your existing instances (created before release December 14, 2022). See Upgrade Patching Tool.

Security Checkup Tool Warnings

Learn about the security check warnings that are displayed in the Oracle WebLogic Server Administration console and how to troubleshoot them.

At the top of the WebLogic Server Administration console, the message `Security warnings detected. Click here to view the report and recommended remedies` is displayed for Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied.

When you click the message, a list of security warnings are displayed as listed in the following table.

The warning messages listed in the table are examples.

Security Warnings

Warning Message	Resolution
`The configuration for key stores for this server are set to Demo Identity and Demo Trust. Trust Demo certificates are not supported in production mode domains.`	Configure the identity and trust keystores for each server and the name of the certificate in the identity keystore that the server uses for SSL communication. See Configure Keystore Attributes for Identity and Trust. **Note:** This warning is displayed for Oracle WebLogic Server for OKE instances created after October 20, 2021, or the instances on which the October PSUs are applied.
`SSL hostname verification is disabled by the SSL configuration.`	Review your applications before you make any changes to address these SSL host name security warnings. For applications that connect to SSL endpoints with a host name in the certificate, which does not match the local machine's host name, the connection fails if you configure the BEA host name verifier in Oracle WebLogic Server. For applications that connect to Oracle provided endpoints such as Oracle Identity Cloud Service (for example,`*.identity.oraclecloud.com`), the connection fails if you did not configure the wildcard host name verifier or a custom host name verifier that accepts wildcard host names. If you are not sure of the SSL configuration settings you should configure to address the warning, Oracle recommends that you configure the wildcard host name verifier. You see the SSL host name verification warnings in case of existing Oracle WebLogic Server for OKE instances (created before July 20, 2021). To address this warning, you must configure SSL with host name verifier. See Configure SSL with host name verifier.
`Production mode is enabled but the file or directory <directory_name>/ startWebLogic.sh is insecure since its permission is not a minimum of umask 027`	Run the following command in the administration server as `oracle` user: `chmod 640 /u01/data/domains/<domain_name>/bin`
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Set the java properties for anonymous RMI T3 and IIOP requests during server start up. See Set the Java Properties.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OKE instances created after July 20, 2021, though the java properties to disable anonymous requests for preventing anonymous RMI access are configured, the warnings still appear. This is a known issue in Oracle WebLogic Server.

Set the Java Properties

To set the java properties for anonymous RMI T3 and IIOP requests:

1. Edit the `domain.yaml` located in `/u01/shared/weblogic-domains/<domain_name>/domain.yaml` for all instances of `serverPod` definitions as follows:

```
serverPod:
      env:
      - name: USER_MEM_ARGS
        #admin server memory is explicitly set to min of 256m and max of
512m and GC algo is G1GC
        value: "-Xms256m -Xmx512m -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
      - name: JAVA_OPTIONS
        value: "-Dweblogic.store.file.LockEnabled=false
        -Dweblogic.rjvm.allowUnknownHost=true
        -Dweblogic.security.remoteAnonymousRMIT3Enabled=false
        -Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

2. Apply the `domain.yaml` using the `kubectl` command:

```
kubectl -f <path_to_domain.yaml>
```

Configure Keystore Attributes for Identity and Trust

To configure the identity and trust keystore files and the name of the certificate in the identity keystore in the WebLogic Server Administration console:

1. Locate the **Change Center** and click **Lock & Edit** to lock the editable configuration hierarchy for the domain.
2. Under **Domain structure**, select **Environment** and then select **Servers**.
3. In the Servers table, select the server you want to configure.
4. On the **Configuration** tab, click **Keystores**, and then click **Change**.
5. Select *Custom Identity and Custom Trust*, and then click **Save**.
6. Under **Identity**, provide the following details:
   a. Enter the full path of your identity keystore.

      For example: `/u01/data/keystores/identity.jks`

   b. For **Custom Identity Keystore Type**, enter *JKS*.
   c. For **Custom Identity Keystore Passphrase**, enter your keystore password. Enter the same value for **Confirm Custom Identity Keystore Passphrase**.
7. Under **Trust**, provide the following details:
   a. Enter the full path of your identity keystore.

      For example, `/u01/data/keystores/trust.jks`

   b. For **Custom Trust Keystore Type**, enter *JKS*.
   c. For **Custom Trust Keystore Passphrase**, enter your keystore password. Enter the same value for **Confirm Custom Trust Keystore Passphrase**.
8. Click **Save**.
9. Click the **SSL** tab.
10. Under **Identity**, provide the following details:

a. For **Private Key Alias**, enter the name of the certificate (private key) in the identitykeystore, *server_cert*.

b. For **Private Key Passphrase**, enter the password for this certificate in the keystore. Enter the same value for **Confirm Private Key Passphrase**.

   By default, the password for the certificate is the same as the identity keystore password.

11. Click **Save**.

    After saving the changes, return to **Change Center** and click **Activate Changes**.

12. Repeat steps 3 to 9 to configure each server in the domain.

Get Additional Help and Contact Support

Use online help, email, customer support, and other tools if you have questions or problems with Oracle WebLogic Server for OKE.

For customer support, you can create support tickets using the Oracle Cloud Infrastructure (OCI) console or My Oracle Support.

Create Support Ticket Using OCI Console

Use the Support Center in Oracle Cloud Infrastructure console to create a support ticket for your technical issues for Oracle WebLogic Server for OKE service in the Marketplace.

> **Note:**
>
> Make sure to provision your support account before you create a support request. See Configuring Your Oracle Support Account in Oracle Cloud Infrastructure documentation.

To create a support ticket:

1. Sign in to the Oracle Cloud Infrastructure console.
2. Click the navigation menu, and select **Governance & Administration**. Under **Support**, click **Support Center**.
3. Click **Create Support Request**.
   The **Technical Support** tab on the Support Options page is displayed.
4. For **Issue Summary**, enter the a title that summarizes your issue.
5. For **Describe Your Issue**, enter a brief description of your issue.
6. Select the severity level of the issue based on the impact of service.
7. Select **Marketplace** from the **Select Service** list.
8. Select **Oracle WebLogic Server for OKE** from the **Select Category** list.
9. Select the type of issue you are experiencing.
10. Click **Create Support Request**.

After you submit the request, My Oracle Support sends a confirmation email to the address provided in the primary contact details. A follow-up email is sent if additional information is required.

Optionally, you can create a support ticket using the Help menu and the Support button in Oracle Cloud Infrastructure console. See Support Ticket Management in Oracle Cloud Infrastructure documentation.

However, when you a create support ticket using these options, the support ticket may not be assigned to a specific service or component for resources like compute instances, networks and load balancers. So, it is recommended to use Support Center in Oracle Cloud Infrastructure console to create support tickets.

Create Support Ticket Using My Oracle Support

Use the Service Request in My Oracle Support to create a support ticket for your technical issues for Oracle WebLogic Server for OKE service in the Marketplace.

> **Note:**
>
> Make sure you have a Support Identifier which verifies your eligibility for Support services, and an account at My Oracle Support.

To create a support ticket:

1. Sign in to My Oracle Support.
2. On the **Service Requests** tab, click **Create Technical SR**.
3. Enter the **Problem Summary** and the **Problem Description**.
4. Under **Where is the Problem**, click **Cloud**.
5. Select **Oracle WebLogic Server for OCI Container Engine** from the **Service Type** list.
6. Select the tenancy from the **Service** list.
7. Select a **Problem Type** and provide the **Support Identifier** details.
8. Click **Next** until you have provided all the mandatory information.
9. Click **Submit**.
   Your service request is created.

For general help with Oracle Cloud Marketplace, see How Do I Get Support in Oracle Cloud Infrastructure documentation.

8
Patches

Each Oracle WebLogic Server for OKE release includes patches from several products, namely, Oracle WebLogic Server, Oracle JDeveloper, Oracle Java Development Kit, Oracle Platform Security Services, and Oracle Web Services Manager.

> **Note:**
>
> If you are using Oracle WebLogic Server for OKE (**Release 21.3.2 or earlier**), see Using Oracle WebLogic Server for OKE (Release 21.3.2 or earlier).

You can use the `Automatic Patching` Jenkins job to automatically schedule patching for the required domains. See Automatic Patching.

The following table shows the patches in the Oracle WebLogic Server for OKE releases that use *multi-domain* source type. Use your Oracle Support account to locate and download the patch you want to apply.

> **Tip:**
>
> For a list of new features and enhancements that were added recently to improve your Oracle WebLogic Server for OKE experience, see What's New for Oracle WebLogic Server for OKE.

Oracle WebLogic Server for OKE Version	Patches	Patch List
24.2.2-oke_v1.28.2-1	**14.1.1.0.0** • 36410357 - Coherence 14.1.1.0 Cumulative Patch 17 (14.1.1.0.17) • 36454290 - WLS patch set update 14.1.1.0.240328 opatch version: • 28186730 - Opatch 13.9.4.2.15 for EM 13.5 and FMW/WLS 12.2.1.4.0 and 14.1.1.0.0	April 2024 PSUs
24.2.2-oke_v1.28.2-1	**12.2.1.4.0** • 36348444 - ADF BUNDLE PATCH 12.2.1.4.240228 • 36410345 - Coherence 12.2.1.4 Cumulative Patch 21 (12.2.1.4.21) • 36402397 - OWSM BUNDLE PATCH 12.2.1.4.240313 • 36468190 - FMW Thirdparty Bundle Patch 12.2.1.4.240401 • 36440005 - WLS patch set update 12.2.1.4.240325 • 36316422 - OPSS Bundle Patch 12.2.1.4.240220 opatch version: • 28186730 - Opatch 13.9.4.2.15 for EM 13.5 and FMW/WLS 12.2.1.4.0 and 14.1.1.0.0	April 2024 PSUs

Oracle WebLogic Server for OKE Version	Patches	Patch List
24.1.1-oke_v1.26.7-1	**14.1.1.0.0** • 36068072 - Coherence 14.1.1.0 Cumulative Patch 16 (14.1.1.0.16) • 36124787 - WLS Patch Set Update 14.1.1.0.231220 • 35965633 - ADR for Weblogic Server 14.1.1.0 Size Optimized for Jan 2024 opatch version: • 28186730 - Opatch 13.9.4.2.14 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	January 2024 PSUs
24.1.1-oke_v1.26.7-1	**12.2.1.4.0** • 36068046 - Coherence 12.2.1.4 Cumulative Patch 20 (12.2.1.4.20) • 36155700 - WLS Patch Set Update 12.2.1.4.240104 • 36074941 - ADF Bundle Patch 12.2.1.4.231205 • 36086980 - FMW Thirdparty Bundle Patch 12.2.1.4.231207 • 35965629 - ADR for Weblogic Server 12.2.1.4.0 Size Optimized for Jan 2024 • 35868571 - OWSM Bundle Patch 12.2.1.4.231003 opatch version: • 28186730 - Opatch 13.9.4.2.14 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	January 2024 PSUs
23.4.1-oke_v1.26.7-1	**14.1.1.0.0** opatch: • 35778872 - Coherence 14.1.1.0 Cumulative Patch 15 (14.1.1.0.15) • 35904051 - WLS Patch Set Update 14.1.1.0.231012 • 35476075 - ADR for Weblogic Server 14.1.1.0 CPU OCT 2023 opatch version: • 28186730 - Opatch 13.9.4.2.14 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	October 2023 PSUs
23.4.1-oke_v1.26.7-1	**12.2.1.4.0** opatch: • 35778804 - Coherence 12.2.1.4 Cumulative Patch 19 (12.2.1.4.19) • 35893811 - WLS Patch Set Update 12.2.1.4.231010 • 35735469 - ADF Bundle Patch 12.2.1.4.230823 • 35882299 - FMW Thirdparty Bundle Patch 12.2.1.4.231006 • 35476067 - ADR for Weblogic Server 12.2.1.4.0 CPU OCT 2023 • 34302154 - oracle.security.restsec.jwt.JwtToken has reference to jackson 1.x opatch version: • 28186730 - Opatch 13.9.4.2.14 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	October 2023 PSUs
23.3.2-oke_v1.26.2-1	**14.1.1.0.0** opatch: • 35505236 - Coherence 14.1.1.0 Cumulative Patch 14 (14.1.1.0.14) • 35560771 - WLS Patch Set Update 14.1.1.0.230703 opatch version: • 28186730 - Opatch 13.9.4.2.13 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	July 2023 PSUs

Oracle WebLogic Server for OKE Version	Patches	Patch List
23.3.2-oke_v1.26.2-1	**12.2.1.4.0** opatch: • 35505207 - Coherence 12.2.1.4 Cumulative Patch 18 (12.2.1.4.18) • 35557681 - WLS Patch Set Update 12.2.1.4.230702 • 35503128 - ADF Bundle Patch 12.2.1.4.230615 • 35547646 - FMW Thirdparty Bundle Patch 12.2.1.4.230628 • 32471832 - System prop to disable all file store locks. opatch version: • 28186730 - Opatch 13.9.4.2.13 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	July 2023 PSUs
23.3.1-oke_v1.26.2-1	**14.1.1.0.0** opatch: • 35505236 - Coherence 14.1.1.0 Cumulative Patch 14 (14.1.1.0.14) • 35560771 - WLS Patch Set Update 14.1.1.0.230703 opatch version: • 28186730 - Opatch 13.9.4.2.13 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	July 2023 PSUs
23.3.1-oke_v1.26.2-1	**12.2.1.4.0** opatch: • 35505207 - Coherence 12.2.1.4 Cumulative Patch 18 (12.2.1.4.18) • 35557681 - WLS Patch Set Update 12.2.1.4.230702 • 35503128 - ADF Bundle Patch 12.2.1.4.230615 • 35547646 - FMW Thirdparty Bundle Patch 12.2.1.4.230628 opatch version: • 28186730 - Opatch 13.9.4.2.13 for EM 13.4, 13.5 and FMW/WLS 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0	July 2023 PSUs
23.2.2-oke_v1.25.4-1	**14.1.1.0.0** opatch: • 35227385 - WLS patch set update 14.1.1.0.230328 • 35122412 - Coherence 14.1.1.0 Cumulative Patch 13 (14.1.1.0.13) opatch version: • 28186730 - Opatch 13.9.4.2.12 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.4.0, and 14.1.1.0.0	April 2023 PSUs

Oracle WebLogic Server for OKE Version	Patches	Patch List
23.2.2-oke_v1.25.4-1	**12.2.1.4.0** opatch: • 35226999 - WLS patch set update 12.2.1.4.230328 • 35122398 - Coherence 12.2.1.4 Cumulative Patch 17 (12.2.1.4.17) • 35159582 - OWSM Bundle Patch 12.2.1.4.230308 • 35162846 - FMW Third party Bundle Patch 12.2.1.4.230309 • 35148842 - ADF BUNDLE PATCH 12.2.1.4.230306 • 33093748 - FMW Platform 12.2.1.4.0 SPU for APR CPU 2021 • 30385564 - Oracle XML Developers Kit patch • 33950717 - OPSS Bundle Patch 12.2.1.4.220311 • 31544353 - ADR for WebLogic Server 12.2.1.4.0 July CPU 2020 • 33903365 - FMW consoles fail after applying JDK 1.80.331 (April Java CPU) or later • 34542329 - Merge request on top of 12.2.1.4.0 for bugs 34280277, 26354548, 26629487, 29762601 opatch version: • 28186730 - Opatch 13.9.4.2.12 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.4.0, and 14.1.1.0.0	April 2023 PSUs
23.2.1-oke_v1.24.1-2	**14.1.1.0.0** opatch: • 35227385 - WLS patch set update 14.1.1.0.230328 • 35122412 - Coherence 14.1.1.0 Cumulative Patch 13 (14.1.1.0.13) opatch version: • 28186730 - Opatch 13.9.4.2.12 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.4.0, and 14.1.1.0.0	April 2023 PSUs
23.2.1-oke_v1.24.1-2	**12.2.1.4.0** opatch: • 35226999 - WLS patch set update 12.2.1.4.230328 • 35122398 - Coherence 12.2.1.4 Cumulative Patch 17 (12.2.1.4.17) • 35159582 - OWSM Bundle Patch 12.2.1.4.230308 • 35162846 - FMW Third party Bundle Patch 12.2.1.4.230309 • 35148842 - ADF BUNDLE PATCH 12.2.1.4.230306 • 33093748 - FMW Platform 12.2.1.4.0 SPU for APR CPU 2021 • 30385564 - Oracle XML Developers Kit patch • 33950717 - OPSS Bundle Patch 12.2.1.4.220311 • 31544353 - ADR for WebLogic Server 12.2.1.4.0 July CPU 2020 • 33903365 - FMW consoles fail after applying JDK 1.80.331 (April Java CPU) or later • 34542329 - Merge request on top of 12.2.1.4.0 for bugs 34280277, 26354548, 26629487, 29762601 opatch version: • 28186730 - Opatch 13.9.4.2.12 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.4.0, and 14.1.1.0.0	April 2023 PSUs
23.1.2-oke_v1.24.1-1	**14.1.1.0.0** opatch: • 34890864 - WLS patch set update 14.1.1.0.221213 • 34845949 - Coherence 14.1.1.0 Cumulative Patch 12 (14.1.1.0.12) opatch version: • 28186730 - Opatch 13.9.4.2.11 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.4.0, and 14.1.1.0.0	January 2023 PSUs

Oracle WebLogic Server for OKE Version	Patches	Patch List
23.1.2-oke_v1.24.1-1	**12.2.1.4.0** opatch: • 34883826 - WLS patch set update 12.2.1.4.221210 • 34845927 - Coherence 12.2.1.4 Cumulative Patch 16 (12.2.1.4.16) • 34839859 - OWSM Bundle Patch 12.2.1.4.221128 • 34879707 - FMW Third party Bundle Patch 12.2.1.4.221209 • 34944256 - ADF Bundle Patch 12.2.1.4.230103 • 33093748 - FMW Platform 12.2.1.4.0 SPU for APR CPU 2021 • 30385564 - Oracle XML Developers Kit patch • 33950717 - OPSS Bundle Patch 12.2.1.4.220311 • 31544353 - ADR for WebLogic Server 12.2.1.4.0 July CPU 2020 • 33903365 - FMW consoles fail after applying JDK 1.80.331 (April Java CPU) or later • 34542329 - Merge request on top of 12.2.1.4.0 for bugs 34280277, 26354548, 26629487, 29762601 opatch version: • 28186730 - Opatch 13.9.4.2.11 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.4.0, and 14.1.1.0.0	January 2023 PSUs

About Patching Utility Tool

Oracle WebLogic Server for OKE provides the patching utility tool to download the patches for the WebLogic Server instances. This utility can be used if you do not have access to the support portal to download the required patches.

You can use this patching utility tool on the Administration host and the bastion instance.

Patch Management Using Patching Utility

Use the patching utility tool in Oracle WebLogic Server for OKE to list and download the patches, and view the patching tool version and upgrade the patching tool.

To apply the patches, see Apply a WebLogic Server Patch.

You can perform the following tasks using the patching utility tool:

> **Note:**
>
> If you want to use some of the new features, which were added to the patching utility tool in December 14, 2022, for your existing Oracle WebLogic Server for OKE instances (created before December 14, 2022), then ensure that you upgrade the patching tool. See Upgrade Patching Tool.

- View Patching Tool Version
- Configure Initial Setup
- List Patches
- View Patch Details

- Download Patches
- Upgrade Patching Tool

View Patching Tool Version

Use the patching tool to view the build version along with the Oracle license and copyright information.

1. Connect to the Administration Server node as the `opc` user.

   The SSH command format is:

   ```
   ssh -i path_to_private_key opc@admin_ip
   ```

2. Print the build version.

   ```
   patch-utils -v
   ```

   Sample output:

   ```
   Weblogic Cloud Patch-Utils <Patch version number>)
   Copyright (c) 2020, Oracle Corporation and/or its affiliates.Licensed
   under the Universal Permissive License v 1.0 as shown at
   https://oss.oracle.com/licenses/upl.
   ```

Configure Initial Setup

Use the patching tool to configure the region from where to download the patches and create the configuration file in the specified Middleware Home.

Currently, the regions: *us-phoenix-1*, *us-ashburn-1*, *eu-frankfurt-1*, *ap-mumbai-1*, *ap-tokyo-1*, and *sa-saopaulo-1* are supported.

> **Note:**
>
> You must set up the configuration before you run the patching tool on any provisioned VM.

1. Connect to the Administration Server node as the `opc` user.

   The SSH command format is:

   ```
   ssh -i path_to_private_key opc@admin_ip
   ```

2. Set up the configuration.

   ```
   patch-utils setup
   ```

   Sample output:

   ```
   Enter middleware home (default: /u01/app/oracle/middleware):
   Choose oci region for patch download
   ```

```
['us-ashburn-1', 'eu-frankfurt-1', 'ap-mumbai-1', 'ap-tokyo-1', 'us-
phoenix-1', 'sa-saopaulo-1']: us-phoenix-1
Created config file [/home/opc/.patchutils/config]
```

List Patches

Use the patching tool to list all the available patches in the patch catalog. You can also list current patches and latest patches that are available in the patching tool repository.

> **Note:**
>
> You must set up the configuration file before running the `patch-utils list` command. See Configure Initial Setup.

1. Connect to the Administration Server node as the `opc` user.

   The SSH command format is:

   ```
   ssh -i path_to_private_key opc@admin_ip
   ```

2. Run the following commands to list patches:

   - List all patches in the patch catalog for the applicable WebLogic Server version.

     ```
     patch-utils list
     ```

     Sample output:

     ```
     <Patch number> ADF Bundle Patch for Bug: <Bug number>, WLS version:
     <WLS version number>
     <Patch number> OPSS Patch Bundle Patch for Bug:<Bug number>, WLS
     version: <WLS version number>
     <Patch number> PATCH <Patch number>- OPATCH <OPatch version number>FOR
     FMW/WLS <WLS version number>AND <WLS version number>
     <Patch number> Oracle Coherence Patch Bundle Patch for Bug:<Bug
     number>, WLS version: <WLS version number>
     <Patch number>Weblogic Service Patch Bundle Patch for Bug:<Bug number>,
     WLS version: <WLS version number>
     ```

   - List all the current patches based on OPatch utility for 12c.

     ```
     patch-utils list -a
     ```

     Sample output:

     ```
     Listing current patches
     Oracle Interim Patch Installer version <Patch version number>)
     Copyright (c) 2020, Oracle Corporation. All rights reserved
     Oracle Home : /u01/app/oracle/middleware
     Central Inventory : /u01/app/oraInventory from : /u01/app/oracle/
     middleware/oraInst.loc
     OPatch version : <OPatch Version number>
     ```

```
OUI version : <OUI Version number>
Log file location : /u01/app/oracle/middleware/cfgtoollogs/opatch/
<opatchtimestamp>.log
OPatch detects the Middleware Home as "/u01/app/oracle/middleware"
Lsinventory Output file location : /u01/app/oracle/middleware/
cfgtoollogs/opatch/lsinv/<lsinventoryopatchtimestamp>.txt
Local Machine Information:
Hostname: testwls-wls-0.wlssubnet.subnet1.oraclevcn.com
ARU platform id: <ID number>
ARU platform description:: Linux x86-64
Interim patches (1):
Patch <WebLogic 12c version number>: applied on <day month date time>
Unique Patch ID: <Patch ID number>
Patch description: "Bundle patch for Oracle Coherence Version <WebLogic
12c version number>"
Created on <date month year time>
Bugs fixed:<Bug number>
OPatch succeeded.
```

- List the latest patches and other component patches for the relevant WebLogic Server version, from the available patches in catalog.
  In case of multiple Middleware Homes for WebLogic Server compute instances,, you can use the `patch-utils setup` command to change the Middleware Home.

  ```
  patch-utils list -L
  ```

  Sample output on the WebLogic Server compute instance:

  ```
  Patch Id           Description
  ----------
  ------------------------------------------------------------------------
  ---
  <Patch number>    FMW Thirdparty Bundle Patch 12.2.1.4.220915
  <Patch number>    Opatch 13.9.4.2.11 for EM 13.4, 13.5 and FMW/WLS
  12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0
  <Patch number>    WLS Patch Set Update 12.2.1.4.220929
  <Patch number>    Merge Request on Top of 12.2.1.4.0 for Bugs <Bug
  number> <Bug number> <Bug number>  <Bug number>
  <Patch number>    Coherence 12.2.1.4 Cumulative Patch 15 (12.2.1.4.15)
  ```

  Sample output on the administration instance:

  ```
  Choose wls type ['WLS', 'FusionMiddleware', 'Coherence', 'Forms',
  'Database'] (default: ALL):

  Patch Id           Description
  ----------
  ------------------------------------------------------------------------
  ---
  <Patch number>    FMW Thirdparty Bundle Patch 12.2.1.4.220915
  <Patch number>    Opatch 13.9.4.2.11 for EM 13.4, 13.5 and FMW/WLS
  12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0
  <Patch number>    WLS Patch Set Update 12.2.1.4.220929
  <Patch number>    Merge Request on Top of 12.2.1.4.0 for Bugs <Bug
  ```

```
number> <Bug number> <Bug number> <Bug number>
<Patch number>    Coherence 12.2.1.4 Cumulative Patch 15 (12.2.1.4.15)
```

View Patch Details

Use the patching tool to view information of the specified patch.

The WebLogic Server patches include the `readme` file that provides the patch details and other useful information about patching.

1. Connect to the Administration Server node as the `opc` user.

   The SSH command format is:

   ```
   ssh -i path_to_private_key opc@admin_ip
   ```

2. View the information for the selected patch.

   ```
   patch-utils info -i <Patch ID>
   ```

   By default, the first ten lines of the `readme.txt` file is displayed.

   Sample output:

   ```
   Patch Set Update (PSU) for Bug: <Bug number>
   Date: Fri Feb 28 17:33:37 2020
   Platform Patch for : Generic
   Product Patched : ORACLE WEBLOGIC SERVER
   Product Version : <WLS version number>
   This document describes how to install patch for bug # 31985811.It
   includes the following sections:
   Section 1: Known Issues
   .......
   more
   ....
   ```

   You can define the number of lines to be displayed using the `-l` parameter.

   For example, to print 25 lines, run the following command:

   ```
   patch-utils info -i <Patch ID> -n 25
   ```

Download Patches

Use the patching tool to download the patches to the specified location.

You can download the patches if NAT gateway is configured. However, if you provision an instance in a private subnet without a bastion (without NAT gateway), you must create a temporary bastion instance in the Oracle Cloud Infrastructure console, and then use the patching tool to download the patches on the bastion host. The patches are encrypted and can only be applied on the WebLogic Server VMs using the patching utility tool.

1. Connect to the Administration Server node as the `opc` user.

The SSH command format is:

```
ssh -i path_to_private_key opc@admin_ip
```

2. Run the following commands to download patches:
   - Download latest patches.

     ```
     patch-utils download -L -p /tmp/<Location to download>
     ```

     Sample output:

     ```
     Successfully downloaded following patches.
     Please copy them to weblogic hosts and apply them locally.['<Patch ID_Generic.zip']
     ```

   - Download patches using patch ID.
     Sample output:

     ```
     patch-utils download -l <Patch ID> -p /tmp/<Location to download>
     ```

     > **Note:**
     >
     > To download multiple patches, specify the patch IDs as comma separated values. Make sure to download the patches to an accessible location.

     Sample output:

     ```
     Successfully downloaded following patches.
     Please copy them to weblogic hosts and apply them locally.['<Patch ID_Generic.zip']
     ```

Upgrade Patching Tool

Use the patching tool to upgrade the patching tool utility to the latest version.

1. Connect to the Administration Server node as the `opc` user.

   The SSH command format is:

   ```
   ssh -i path_to_private_key opc@admin_ip
   ```

2. Upgrade `patch-utils` to the latest version.

   ```
   patch-utils upgrade
   ```

Note:

This command is used to upgrade VMs if the NAT Gateway is enabled on the WebLogic Server subnet.
Sample output:

```
Successfully updated patch-utils to [<Patch Utils version
number>]. Please rerun patch-utils.
```

A

Oracle Cloud Identifiers and Listings

Learn about the list of Oracle WebLogic Server for OKE images available on the Partner Image Catalog that contains the entitlement to use the different versions of Oracle WebLogic software for Oracle WebLogic Server for OKE UCM application. These images are priced at the same rate as the Oracle WebLogic Server Enterprise Edition for OKE and Oracle WebLogic Suite for OKE stacks.

See Oracle WebLogic Server Enterprise Edition for OKE and Oracle WebLogic Suite for OKE.

> **Note:**
>
> To view the pricing details, click the **Get App** link.

The following table shows the listings and the OCIDs for the different Oracle WebLogic Server editions of Oracle WebLogic Server for OKE.

WebLogic Server Edition	Image Name	Image ID	Listing ID	Resource Version ID
Oracle WebLogic Server Enterprise Edition	wlsoke-custom-np-image-ee-UCM-20.4.1-200917030044	ocid1.image.oc1..aaaaaaaaibbsg23uasf77j4kdldnjnbmgkfjxd5gqywabs3hwx2jw45pj24q	ocid1.appcataloglisting.oc1..aaaaaaaabw6dti6ejlfe4h5vcdtuemmzcbxc6myje2t4au6fox5excyiy2ma	20.4.1-200917030044-092120202314
Oracle WebLogic Suite Edition	wlsoke-custom-np-image-suite-UCM-20.4.1-200917030044	ocid1.image.oc1..aaaaaaaaznbtycmdn7747itt3qmipjvnui4xnnjgiztteszgghnjepjrbknq	ocid1.appcataloglisting.oc1..aaaaaaaaln2a5njbk3mtcqmokjrptv62cqeoqrm4ntyjojko5lqypqbgucua	20.4.1-200917030044-092120202313

B

License Information

Learn about the licensed third-party technology associated with Oracle WebLogic Server for OKE.

Open Source or Other Separately Licensed Software

Required notices for open source or other separately licensed software products or components distributed in Oracle WebLogic Server for OKE are identified in the following table along with the applicable licensing information. Additional notices and/or licenses may be found in the included documentation or `readme` files of the individual third party software.

Provider	Component(s)	Licensing Information
Docker Inc.	Docker 19.03.11.ol-4.el7	Apache License Version 2
Jenkins CI	Jenkins 2.235.5	MIT License
Jenkins CI	Jenkins active choices-plugin 2.3	MIT License
Jenkins CI	Jenkins Mask Passwords Plug-in 2.13	MIT License
Kubernetes	kubectl 1.17.9	Apache License Version 2
NginX	NGINX Ingress Controller 0.43.0	Apache License Version 2
Python Software Foundation	Python 3.6.8-13.0.1.el7	Python License 2.0
The Helm Authors	Helm 3.2.4	Apache License Version 2
The Kubernetes Authors	Kubernetes Python Client 11.0.0	Apache License Version 2
The pip developers	pip 20.1.1	MIT License
Yichun Zhang	OpenResty 1.15.8.2	BSD License

C

Script Files

This section list all the required script files required for Oracle WebLogic Server for OKE.

Topics:

Script File To Validate Network Setup

You must create a script file to validate if the existing WebLogic Server subnet and the database subnets meet the prerequisites to provision the WebLogic instance in Oracle WebLogic Server for OKE. You can copy the following scripts in Cloud Shell to perform the validation. For example, copy the scripts and save the file as `validateoke.sh`.

```
# Script to validate existing public, private and database subnets meet the
prerequisites
# for provisioning and proper functioning of Oracle WebLogic Server for OKE.
#
version="1.0.0"

# Set Flags
# -----------------------------------
# Flags which can be overridden by user input.
# Default values are below
# -----------------------------------
DB_PORT=1521
SSH_PORT=22
BASTION_SUBNET_OCID=""
ADMIN_SUBNET_OCID=""
WORKER_SUBNET_OCID=""
FSS_SUBNET_OCID=""
LB_SUBNET_OCID=""

DB_SUBNET_OCID=""
BASTION_HOST_IP_CIDR=""

debug=false
args=()

function ip_to_int() {
  local ip_addr="${1}"
  local ip_1 ip_2 ip_3 ip_4

  ip_1=$(echo "${ip_addr}" | cut -d'.' -f1)
  ip_2=$(echo "${ip_addr}" | cut -d'.' -f2)
  ip_3=$(echo "${ip_addr}" | cut -d'.' -f3)
```

```
  ip_4=$(echo "${ip_addr}" | cut -d'.' -f4)

  echo $(( ip_1 * 256**3 + ip_2 * 256**2 + ip_3 * 256 + ip_4 ))
}

#####################################################
# Determine whether IP address is in the specified subnet.
#
# Args:
#   cidr_subnet: Subnet, in CIDR notation.
#   ip_addr: IP address to check.
#
# Returns:
#   0|1
#####################################################
function in_cidr_range() {
  local cidr_subnet="${1}"
  local ip_addr="${2}"
  local subnet_ip cidr_mask netmask ip_addr_subnet subnet rval

  subnet_ip=$(echo "${cidr_subnet}" | cut -d'/' -f1)
  cidr_mask=$(echo "${cidr_subnet}" | cut -d'/' -f2)

  netmask=$(( 0xFFFFFFFF << $(( 32 - ${cidr_mask} )) ))

  # Apply netmask to both the subnet IP and the given IP address
  ip_addr_subnet=$(( netmask & $(ip_to_int ${ip_addr}) ))
  subnet=$(( netmask & $(ip_to_int ${subnet_ip}) ))

  # Subnet IPs will match if given IP address is in CIDR subnet
  [ "${ip_addr_subnet}" == "${subnet}" ] && rval=0 || rval=1

  return $rval
}

#####################################################
# Validates if one of service or nat gateways exist in the specified private
subnet.
#
# Returns:
#   0|1
#####################################################
function validate_service_or_nat_gw_exist() {
  local subnet_ocid=$1
  local vcn_ocid=""
  local vcn_compartment_ocid=""
  is_private_subnet=$(oci network subnet get --subnet-id "${subnet_ocid}" |
jq -r '.data["prohibit-public-ip-on-vnic"]')

  if [[ $is_private_subnet = true ]]
  then
    vcn_ocid=$(oci network subnet get --subnet-id "${subnet_ocid}" | jq -r
'.data["vcn-id"]')
    vcn_compartment_ocid=$(oci network vcn get --vcn-id "${vcn_ocid}" | jq -r
'.data["compartment-id"]')
    # Check if NAT gateway exists in the VCN
```

```
    res=$(oci network nat-gateway list --compartment-id $
{vcn_compartment_ocid} --vcn-id ${vcn_ocid})
    nat_gw_found=$(if [[ -n $res ]]; then echo 0; else echo 1; fi)

    # Check if Service gateway exists in the VCN
    res=$(oci network service-gateway list --compartment-id $
{vcn_compartment_ocid} --vcn-id ${vcn_ocid})
    svc_gw_found=$(if [[ -n $res ]]; then echo 0; else echo 1; fi)

    # One of NAT or Service Gateway must exist
    if [[ $nat_gw_found -ne 0 ]] && [[ $svc_gw_found -ne 0 ]]
    then
      echo 1
      return
    fi

    # Admin subnet should be using either NAT or service gateway or both in
its routetable
    rt_ocid=$(oci network subnet get --subnet-id ${subnet_ocid} | jq -r
'.data["route-table-id"]')
    rt_rules=$(oci network route-table get --rt-id ${rt_ocid} | jq -r
'.data["route-rules"]')
    rt_rules_count=$(echo $rt_rules | jq '.|length')

    nat=""
    svc=""
    nat_gw_id=""
    svc_gw_id=""

    for ((i = 0 ; i < $rt_rules_count ; i++))
    do
      network_entity_ocid=$(echo $rt_rules | jq -r --arg i "$i" '.[$i|
tonumber]["network-entity-id"]')
      nat_id=$(echo $network_entity_ocid | grep natgateway)
      if [[ -n $nat_id ]]; then nat_gw_id=$nat_id; fi

      svc_id=$(echo $network_entity_ocid | grep servicegateway)
      if [[ -n $svc_id ]]; then svc_gw_id=$svc_id; fi
    done

    if [[ (-z $nat_gw_id  && -z $svc_gw_id) ]]; then
      echo 2
      return
    fi

    # If WLS subnet route table has a rule to use service gateway then it
should be using
    # all-<region-code>-services-in-oracle-services-network destination
    echo ""
    if [[ -n $svc_gw_id ]]
    then
      is_all_services_name=$(oci network service-gateway get --service-
gateway-id $svc_gw_id | jq -r '.data.services[0]["service-name"]' | grep -i
"all.*services in oracle services network")
      if [[ -z $is_all_services_name ]]
      then
```

```
        echo 3
        return
      fi
      for ((i = 0 ; i < $rt_rules_count ; i++))
      do
        network_entity_ocid=$(echo $rt_rules | jq -r --arg i "$i" '.[$i|
tonumber]["network-entity-id"]')
        res=$(echo $network_entity_ocid | grep servicegateway)
        if [[ -n $res ]]
        then
          all_services_destination=$(echo $rt_rules | jq -r --arg i "$i" '.
[$i|tonumber].destination'  | grep -i "all-.*-services-in-oracle-services-
network")
          if [[ -z $all_services_destination ]]
          then
            echo 4
            return
          fi
        fi
      done
    fi
  fi
  echo 0
}

#####################################################
# Validates if the internet gateway exists in the VCN of Admin subnet.
# Without Internet gateway in Admin Subnet VCN, SSH access from ORM will not
work.
# When using terraform CLI from within private network, internet gateway is
not required.
# Hence this check will give a warning and not an error.
#
# Returns:
#   0|1
#####################################################
function validate_internet_gw_exist() {
  local subnet_ocid=$1
  local vcn_ocid=""
  local vcn_compartment_ocid=""

  vcn_ocid=$(oci network subnet get --subnet-id ${subnet_ocid} | jq -r
'.data["vcn-id"]')
  vcn_compartment_ocid=$(oci network vcn get --vcn-id ${vcn_ocid} | jq -r
'.data["compartment-id"]')
  # Check if Service gateway exists in the VCN
  res=$(oci network internet-gateway list --compartment-id $
{vcn_compartment_ocid} --vcn-id ${vcn_ocid})
  if [[ -n $res ]]; then
    echo 0
  else
    echo 1
  fi
}

#####################################################
```

```
# Checks if specified port is open to specified source CIDR in the specified
seclist's ingress rules.
#
# Args:
#     seclist_ocid: Security list OCID for the security list to check ingress
rules for.
#     port: destination port to check
#     source: Source CIDR (either block/range of IPs or single IP (with /32
suffix)
#
# Returns:
#   0|1
#####################################################
function check_tcp_port_open_in_seclist() {
  local seclist_ocid=$1
  local port=$2
  local source=$3
  local port_is_open=false
  local tcp_protocol="6"

  ingress_rules=$(oci network security-list get --security-list-
id $seclist_ocid | jq -r '.data["ingress-security-rules"]')
  ingress_rules_count=$(echo $ingress_rules | jq '.|length')

  for ((i = 0 ; i < $ingress_rules_count ; i++))
  do
    ingress_protocol=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].protocol')
    ingress_source=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].source')
    tcp_options=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]
["tcp-options"]')
    port_min=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["tcp-
options"]["destination-port-range"].min')
    port_max=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["tcp-
options"]["destination-port-range"].max')

    source_in_cidr_range=1
    if [[ $source = "0.0.0.0/0" ]]
    then
      if [[ $ingress_source = $source ]]
      then
        source_in_cidr_range=0
      else
        source_in_cidr_range=1
      fi
    else
      source_in_cidr_range=$(in_cidr_range $ingress_source $source ; echo $?)
    fi

    if [[ ($ingress_protocol = "all" || $ingress_protocol = $tcp_protocol )
&& ( $tcp_options = "null" || ( $port -ge $port_min && $port -
le $port_max ) ) && $source_in_cidr_range -eq 0 ]]
    then
       port_is_open=true
       echo 0
```

```
        return
      fi
    done
    echo 1
}

#####################################################
# Validates if the specified TCP port is open for the WLS subnet CIDR.
#
# Args:
#     port:         Destination port
#     source_cidr:  Source CIDR
#
# Returns:
#   0|1
#####################################################
function validate_subnet_port_access() {
  local port_found_open=1
  local subnet=$1
  local port=$2
  local source_cidr=$3
  local protocol=$4 # Default protocol is TCP, if it is UDP then need to pass
this param
  sec_lists=$(oci network subnet get --subnet-id ${subnet} | jq -c
'.data["security-list-ids"]')
  # Convert to bash array
  declare -A seclists_array

  while IFS="=" read -r key value
  do
      seclists_array[$key]="$value"
  done < <(jq -r 'to_entries|map("\(.key)=\(.value|tostring)")|.[]' <<<
"$sec_lists")
  # Check the ingress rules for specified destination port is open for access
by source CIDR
  for seclist_ocid in "${seclists_array[@]}"
  do
    if [[ $port_found_open -ne 0 ]]; then
      if [[ -z $protocol ]]; then # default is TCP
        port_found_open=$(check_tcp_port_open_in_seclist $seclist_ocid "$
{port}" "$source_cidr")
      else # protocol param is non empty then udp
        port_found_open=$(check_udp_port_open_in_seclist $seclist_ocid "$
{port}" "$source_cidr")
      fi
    fi
  done
  echo $port_found_open
}

#####################################################
# Checks if specified UDP port is open to specified source CIDR in the
specified seclist's ingress rules.
#
# Args:
#     seclist_ocid: Security list OCID for the security list to check ingress
```

```
rules for.
#     port: destination port to check
#     source: Source CIDR (either block/range of IPs or single IP (with /32
suffix)
#
# Returns:
#   0|1
#####################################################
function check_udp_port_open_in_seclist() {
  local seclist_ocid=$1
  local port=$2
  local source=$3
  local port_is_open=false
  local udp_protocol="17"

  ingress_rules=$(oci network security-list get --security-list-
id $seclist_ocid | jq -r '.data["ingress-security-rules"]')
  ingress_rules_count=$(echo $ingress_rules | jq '.|length')

  for ((i = 0 ; i < $ingress_rules_count ; i++))
  do
    ingress_protocol=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].protocol')
    ingress_source=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].source')
    udp_options=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]
["udp-options"]')
    port_min=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["udp-
options"]["destination-port-range"].min')
    port_max=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["udp-
options"]["destination-port-range"].max')

    source_in_cidr_range=1
    if [[ $source = "0.0.0.0/0" ]]
    then
      if [[ $ingress_source = $source ]]
      then
        source_in_cidr_range=0
      else
        source_in_cidr_range=1
      fi
    else
      source_in_cidr_range=$(in_cidr_range $ingress_source $source ; echo $?)
    fi

    if [[ ($ingress_protocol = "all" || $ingress_protocol = $udp_protocol )
&& ( $udp_options = "null" || ( $port -ge $port_min && $port -
le $port_max ) ) && $source_in_cidr_range -eq 0 ]]
    then
       port_is_open=true
       echo 0
       return
    fi
  done
  echo 1
}
```

```
#####################################################
# Validates if CIDR is a valid single host IP (must end with /32 suffix).
#
# Args:
#     ip_cidr: Single host IPv4 Address in CIDR format
#
# Returns:
#   0|1
#####################################################
function is_valid_ip_cidr() {
  local ip_cidr=$1

  is_valid=$(echo ${ip_cidr} | grep -E '^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4]
[0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/
(32))$')
  if [[ -n $is_valid ]]; then
    echo 0
  else
    echo 1
  fi
}

############## Begin Options and Usage ###################

# Print usage
usage() {
  echo -n "$0 [OPTIONS]...

 This script is used to validate existing subnets for OKE - Bastion, Admin,
Worker, FSS, LB subnets (and optionally database subnets) are setup correctly.
 ${bold}Options:${reset}
  -b, --bastionsubnet Bastion Subnet OCID (Required)
  -a, --adminsubnet   Admin Subnet OCID (Required)
  -w  --workersubnet  Workers Subnet OCID (Required)
  -f  --fsssubnet     FSS Subnet OCID (Required)
  -l  --lbsubnet      LB Subnet OCID (Required)
  -d, --dbsubnet      DB Subnet OCID
  -i, --bastionipcidr Bastion Host IP CIDR (should be suffixed with /32)
      --debug         Runs script in BASH debug mode (set -x)
  -h, --help          Display this help and exit
      --version       Output version information and exit
  "
}

# Iterate over options breaking -ab into -a -b when needed and --foo=bar into
# --foo bar
optstring=h
unset options
while (($#)); do
  case $1 in
    # If option is of type -ab
    -[!-]?*)
      # Loop over each character starting with the second
      for ((i=1; i < ${#1}; i++)); do
```

```
        c=${1:i:1}

        # Add current char to options
        options+=("-$c")

        # If option takes a required argument, and it's not the last char make
        # the rest of the string its argument
        if [[ $optstring = *"$c:"* && ${1:i+1} ]]; then
          options+=("${1:i+1}")
          break
        fi
      done
      ;;

    # If option is of type --foo=bar
    --?*=*) options+=("${1%%=*}" "${1#*=}") ;;
    # add --endopts for --
    --) options+=(--endopts) ;;
    # Otherwise, nothing special
    *) options+=("$1") ;;
  esac
  shift
done
set -- "${options[@]}"
unset options

# Print help if no arguments were passed.
[[ $# -eq 0 ]] && set -- "--help"

# Read the options and set stuff
while [[ $1 = -?* ]]; do
  case $1 in
    -h|--help) usage >&2; exit 0 ;;
    --version) echo "$(basename $0) ${version}"; exit 0 ;;
    -b|--bastionsubnet) shift; BASTION_SUBNET_OCID=${1} ;;
    -a|--adminsubnet) shift; ADMIN_SUBNET_OCID=${1} ;;
    -w|--workersubnet) shift; WORKER_SUBNET_OCID=${1} ;;
    -f|--fsssubnet) shift; FSS_SUBNET_OCID=${1} ;;
    -l|--lbsubnet) shift; LB_SUBNET_OCID=${1} ;;
    -d|--dbsubnet) shift; DB_SUBNET_OCID=${1} ;;
    -i|--bastionipcidr) shift; BASTION_HOST_IP_CIDR=${1} ;;
    --debug) debug=true;;
    --endopts) shift; break ;;
    *) "invalid option: '$1'." ; usage >&2; exit 1 ;;
  esac
  shift
done

# Store the remaining part as arguments.
args+=("$@")

############## End Options and Usage ###################

# ############# ############# #############
# ##       MAIN SCRIPT BODY             ##
# ##                                    ##
```

```
# ##                                      ##
# ############ ############ ############

# Set IFS to preferred implementation
IFS=$'\n\t'

# Exit on error. Append '||true' when you run the script if you expect an
error.
set -o errexit

# Run in debug mode, if set
if ${debug}; then set -x ; fi

# Bash will remember & return the highest exitcode in a chain of pipes.
# This way you can catch the error in case mysqldump fails in `mysqldump |
gzip`, for example.
set -o pipefail

# Validate all required params are present
if [[ -z ${BASTION_SUBNET_OCID} || -z ${ADMIN_SUBNET_OCID} || -z $
{WORKER_SUBNET_OCID} || -z ${FSS_SUBNET_OCID} || -z ${LB_SUBNET_OCID} ]]
then
  echo "One or more required params are not specified. Please provide either
bastion and Admin subnet OCIDs"
  usage >&2
  exit
fi

vcn_ocid=$(oci network subnet get --subnet-id "${BASTION_SUBNET_OCID}" | jq -
r '.data["vcn-id"]')
vcn_cidr=$(oci network vcn get --vcn-id "${vcn_ocid}" | jq -r '.data["cidr-
block"]')

# Check if SSH port - 22 is open for access by Bastion Subnet
if [[ -n ${BASTION_SUBNET_OCID} || -n ${BASTION_HOST_IP_CIDR} ]]
then
  all_ips="0.0.0.0/0"
  res=$(validate_subnet_port_access "${BASTION_SUBNET_OCID}" "${SSH_PORT}" "$
{all_ips}")

  if [[ ${res} -ne 0 ]]
    then
      echo "ERROR: SSH port ${SSH_PORT} is not open for access by [$all_ips]
in -- ${BASTION_SUBNET_OCID}"
  fi

  # Check if bastion host IP is valid CIDR
  bastion_cidr_block=""
  if [[ -n ${BASTION_HOST_IP_CIDR} ]]
  then
    is_valid_cidr=$(is_valid_ip_cidr "${BASTION_HOST_IP_CIDR}")
    if [[ $is_valid_cidr -ne 0 ]]
    then
      echo "Bastion host IP CIDR is not valid: [${BASTION_HOST_IP_CIDR}]"
      usage >&2
      exit
```

```
    fi
    bastion_cidr_block=${BASTION_HOST_IP_CIDR}
  else
    bastion_cidr_block=$(oci network subnet get --subnet-id "$
{BASTION_SUBNET_OCID}" | jq -r '.data["cidr-block"]')
  fi

  # Check if bastion CIDR has access to SSH port on ADMIN subnet
  res=$(validate_subnet_port_access "${ADMIN_SUBNET_OCID}" "${SSH_PORT}" "$
{bastion_cidr_block}")

  if [[ $res -ne 0 ]]
  then
    echo "WARNING: SSH port ${SSH_PORT} is not open for access by Bastion
Subnet CIDR [$bastion_cidr_block] in private Admin Subnet
[$ADMIN_SUBNET_OCID]"
  fi
fi

# Check if service or NAT gateway exists in ADMIN & WORKER subnet's VCN.
if [[ -n ${ADMIN_SUBNET_OCID} && -n ${WORKER_SUBNET_OCID} ]]
then
  subnet_names=('ADMIN_SUBNET' 'WORKER_SUBNET')
  i=0
  for subnet_ocid in ${ADMIN_SUBNET_OCID} ${WORKER_SUBNET_OCID}; do
    res=$(validate_service_or_nat_gw_exist "${subnet_ocid}")
    if [[ $res -eq 1 ]]
    then
      echo "ERROR: Missing Service or NAT gateway in the VCN of the private $
{subnet_names[i]} subnet ocid [$subnet_ocid]"
    elif [[ $res -eq 2 ]]
    then
      echo "ERROR: Private ${subnet_names[i]} subnet [$subnet_ocid] does not
use NAT or Service gateway"
    elif [[ $res -eq 3 ]]
    then
      echo "ERROR: Service Gateway in VCN of private ${subnet_names[i]}
subnet [$subnet_ocid] does not allow access to all services in Oracle
services network"
    elif [[ $res -eq 4 ]]
    then
      echo "ERROR: Route Rule of private ${subnet_names[i]} subnet
[$subnet_ocid] does not use 'ALL Services in Oracle services network'
destination"
    fi
  done
fi

# Check if internet gateway exists in BASTION & LB & FSS subnet's VCN.
subnet_names=('BASTION_SUBNET' 'LB_SUBNET' 'FSS_SUBNET_OCID')
i=0
for subnet_ocid in ${BASTION_SUBNET_OCID} ${LB_SUBNET_OCID} $
{FSS_SUBNET_OCID}; do
  res=$(validate_internet_gw_exist "${subnet_ocid}")

  if [[ $res -ne 0 ]]
```

```
  then
    echo "WARNING: Missing internet gateway in the VCN of the $
{subnet_names[i]} subnet [$subnet_ocid]"
  fi
  i=$((i+1))
done

# Check if LB Subnet ports are open  0.0.0.0/0 all, 443, 80
all_ips="0.0.0.0/0"
for port in 'all' '443' '80'; do
  res=$(validate_subnet_port_access "${LB_SUBNET_OCID}" "${port}" "$
{all_ips}")

  if [[ $res -ne 0 ]]
  then
    echo "WARNING: Port [$port] is not open for 0.0.0.0/0 in LB Subnet CIDR [$
{LB_SUBNET_OCID}]"
  fi
done

# Check if Worker Subnet all protocols are open for workers subnet
worker_subnet_cidr=$(oci network subnet get --subnet-id "$
{WORKER_SUBNET_OCID}" | jq -r '.data["cidr-block"]')
res=$(validate_subnet_port_access "${WORKER_SUBNET_OCID}" "all" $
{worker_subnet_cidr})

if [[ $res -ne 0 ]]
then
  echo "ERROR: All Protocols are not open for WORKER's Subnet CIDR [$
{worker_subnet_cidr}]"
fi

# FSS subnet verification - Checking All TCP Ports are open in FSS SUBNET
OCID for VCN CIDR
for port in '111' '2048' '2049' '2050'; do
  res=$(validate_subnet_port_access "${FSS_SUBNET_OCID}" "${port}" "$
{vcn_cidr}")
  if [[ $res -ne 0 ]]
  then
    echo "ERROR: TCP Port [${port}] is not open in FSS Subnet for VCN CIDR"
  fi
done

# FSS subnet verification - UDP - '111' '2048' in FSS SUBNET OCID for VCN
CIDR"
for port in '111' '2048'; do
  res=$(validate_subnet_port_access "${FSS_SUBNET_OCID}" "${port}" "$
{vcn_cidr}" "UDP")
  if [[ $res -ne 0 ]]
  then
    echo "ERROR: UDP Port [${port}] is not open in FSS Subnet for VCN CIDR"
  fi
done

# Check if DB port is open for access by Worker's subnet CIDR in DB subnet
(only if DB subnet is provided)
```

```
if [[ -n ${DB_SUBNET_OCID} ]]
then
  res=$(validate_subnet_port_access ${DB_SUBNET_OCID} ${DB_PORT} ${vcn_cidr})
  res1=$(validate_subnet_port_access ${DB_SUBNET_OCID} ${DB_PORT} $
{worker_subnet_cidr})

  if [[ (${res} -ne 0) || (${res1} -ne 0) ]]
  then
    echo "ERROR: DB port ${DB_PORT} is not open for access by VCN CIDR
[$vcn_cidr] or Worker Subnet CIDR [$worker_subnet_cidr] in DB Subnet
[$DB_SUBNET_OCID]"
  fi
fi
```

Script File to Update SSL Certificate for Load Balancer

You must create a script file, `update_lb_ssl_cert.sh` to update the OCI load balancer SSL certificate, in the administration instance.

```
#!/bin/bash

# Copyright (c) 2022, Oracle and/or its affiliates.  All rights reserved.
# This software is dual-licensed to you under the Universal Permissive
License (UPL) 1.0 as shown at https://oss.oracle.com/licenses/upl or Apache
License 2.0 as shown at http://www.apache.org/licenses/LICENSE-2.0. You may
choose either license.

# This script provides a way to update the OCI load balancer ssl certificate.
#
# The script will:
#       * Create a TLS secret in kubernetes
#       * Run Helm upgrade for ingress controller charts with new certificate
#       * Runs a check on the svc. Please check the annotations for the new
secret name.
#
#   Please refer to https://docs.oracle.com/en-us/iaas/Content/ContEng/Tasks/
contengcreatingloadbalancer.htm for more information


usage()
{
cat <<EOF
Usage: $0 [OPTION]
[Mandatory]
  -d WebLogic Domain Name
  -s Kubernetes secret name
  -k SSL Certificate Key file (e.g. tls.key)
  -c SSL Certificate file (e.g. tls.cert)
EOF
}

if [ "$#" -eq 0 ]; then
  usage
  exit 1
```

```
fi


while getopts ":d:s:k:c:h" opt; do
  case $opt in
    d) DOMAIN_NAME=$OPTARG >&2 ;;
    s) SSL_CERT_SECRET=$OPTARG >&2 ;;
    k) SSL_KEY_FILE=$OPTARG >&2 ;;
    c) SSL_CERT_FILE=$OPTARG >&2 ;;
    h) usage; exit 0 ;;
    \?) echo "Invalid option: -$OPTARG" >&2; usage; exit 1 ;;
    :)  echo "Option -$OPTARG requires an argument." >&2; usage; exit 1 ;;
  esac
done

echo $DOMAIN_NAME $SSL_CERT_SECRET $SSL_KEY_FILE $SSL_CERT_FILE

if [[ $DOMAIN_NAME == "" || $SSL_CERT_SECRET == "" || $SSL_KEY_FILE == ""
|| $SSL_CERT_FILE == "" ]];then
    usage; exit 0;
fi

[[ ! -f $SSL_KEY_FILE ]] && echo "Error:Cannot find $SSL_KEY_FILE." && exit 1

[[ ! -f $SSL_CERT_FILE ]] && echo "Error:Cannot find  $SSL_CERT_FILE." &&
exit 1

PROPERTIES_FILE="/u01/shared/provisioning_metadata.properties"
RELEASE_NAME="ingress-controller"
INGRESS_CHARTS=/u01/shared/scripts/pipeline/create_domain/ingress-controller
DEFAULT_VALUES=/u01/shared/weblogic-domains/$DOMAIN_NAME/ingress-controller-
inputs.yaml

[[ ! -f $PROPERTIES_FILE ]] && echo "Error:Missing $PROPERTIES_FILE file." &&
exit 1;

[[ ! -f $DEFAULT_VALUES ]] &&  echo "Error:Missing helm chart values file
[$DEFAULT_VALUES]" && exit 1;

INGRESS_NAMESPACE=$(cat $PROPERTIES_FILE| grep ingress_namespace | cut -d'=' -
f2)
OCIR_INGRESS_CONTROLLER_REPO=$(cat $PROPERTIES_FILE| grep
ocir_ingress_controller_repo | cut -d'=' -f2)


#1. Use the following command to create a TLS secret in Kubernetes, whose key
and certificate values are set by --key and --cert, respectively.
kubectl create secret tls $SSL_CERT_SECRET --key $SSL_KEY_FILE --
cert $SSL_CERT_FILE -n $INGRESS_NAMESPACE
kubectl create secret tls $SSL_CERT_SECRET --key $SSL_KEY_FILE --
cert $SSL_CERT_FILE -n $DOMAIN_NAME-ns

#2. Update helm charts with new SSL secret value
cmd_output=$(helm upgrade --install $RELEASE_NAME $INGRESS_CHARTS --
values $DEFAULT_VALUES --set cert_secret_name=$SSL_CERT_SECRET --set
ocir_ingress_image_tag=$OCIR_INGRESS_CONTROLLER_REPO --wait 2>&1)
```

```
exit_code=$?
echo "${cmd_output}"

#3. Verify lb service is updated
kubectl describe svc "${DOMAIN_NAME}-lb-external" -n "${INGRESS_NAMESPACE}"
```

Script File To Upgrade Cluster and Node Pool

Use the script file in Oracle WebLogic Server for OKE to upgrade cluster and node pools. You can upgrade either the cluster or the node pool, or both.

> **Note:**
>
> Ensure that you stop the domain before upgrading the nodepool, which contains the domain pods running inside it. Run the following command to stop the domain:
>
> `/u01/scripts/wls-domain-lifecycle/stopDomain.sh -d <domain-name> -n <domain-name>-ns`
>
> After you upgrade the nodepool, start the domain. Run the following command to start the domain:
>
> `/u01/scripts/wls-domain-lifecycle/startDomain.sh -d <domain-name> -n <domain-name>-ns`

Copy the following script in a file named, `upgrade_cluster.py`, and then run the script on the administration instance. See Upgrade Cluster and Node Pool Using Script .

```
#
# Copyright (c) 2023, Oracle Corporation and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at https://
oss.oracle.com/licenses/upl.
#

import oci
import sys
import re
sys.path.append('/u01/scripts')
from metadata import databag

'''
A python class for upgrading kubernetes cluster and the node pools along with
the nodes to a given kubernetes version.

Note:
 - User is responsible to provide correct target Kubernetes version to
upgrade.

Prerequisites:
 - Requires python OCI SDK 2.90 or above. To install python OCI SDK, run the
```

```
following command as 'root' user in the
   WebLogic for OKE admin host
   python3 -m pip install oci==2.90

Description:
 - Recursively upgrades the kubernetes cluster till the target version (to be
provided by user) is reached
   (stops at the previous version if the target version is not available;
does not rollback to original version)
 - For each node pool ...
    -- Get the nodes and delete them
    -- Upgrade the node pool to target kubernetes version
    -- Create the nodes (the same number) in the node pool
 This script can be used to only upgrade the cluster or only upgrade the
nodepool. Caution should be exercised when
 providing the correct kubernetes target version. Current targets supported
are 1.24.x, 1.25.x and 1.26.x
'''

class UpgradeCluster():

    def __init__(self, target_k8s_version):
        self.k8s_version = target_k8s_version
        self.cluster_id = databag.get_oke_cluster_id()
        principal = oci.auth.signers.InstancePrincipalsSecurityTokenSigner()
        self.ce_client =
oci.container_engine.ContainerEngineClient(config={},signer=principal)
        self.ce_client_ops =
oci.container_engine.ContainerEngineClientCompositeOperations(self.ce_client)
        self.upgrade_complete = False

    def upgrade_cluster(self):
        if self.upgrade_complete:
            print ("OKE cluster upgrade completed.")
            return

        print ("Getting cluster information ...")
        response = self.ce_client.get_cluster(self.cluster_id)

        if response.status == 200:
            cluster = response.data
            available_k8s_upgrades = cluster.available_kubernetes_upgrades

            if len(available_k8s_upgrades) == 0:
                print ("The kubernetes cluster is already at the highest
available version")
                self.upgrade_complete = True
                return

            upgrade_version = available_k8s_upgrades[0]
            print ("Upgrading cluster to version %s" % upgrade_version)
            if not self.check_versions(upgrade_version):
                print ("The version %s is not available for upgrade for this
cluster." % self.k8s_version)
                sys.exit(1)
```

```
            # Upgrade the cluster
            update_cluster_details =
oci.container_engine.models.UpdateClusterDetails(
                kubernetes_version = upgrade_version)
            update_cluster_response =
self.ce_client_ops.update_cluster_and_wait_for_state (
                self.cluster_id, update_cluster_details,

wait_for_states=[oci.container_engine.models.WorkRequest.STATUS_SUCCEEDED,

oci.container_engine.models.WorkRequest.STATUS_FAILED])

            if update_cluster_response.status ==
oci.container_engine.models.WorkRequest.STATUS_FAILED:
                print ("Failed to upgrade cluster. %s" %
update_cluster_response.data)
                self.upgrade_complete = True
                sys.exit(1)

            print ("Upgraded cluster to version %s" % upgrade_version)
            if upgrade_version == self.k8s_version:
                self.upgrade_complete = True

            self.upgrade_cluster()
        else:
            print ("Failed to get the kubernetes cluster details. Response
[%s] " % response.status)
            sys.exit(1)

    def check_versions(self, available_version):
        # The version will be of the format v<major>.<minor>.<patch>. We
should check if <major>.<minor> matches the
        # target version (or less than target in which case we will upgrade
and get into next upgrade iteration).
        # If <major>.<minor> match with the target version, then compare the
available and target version as strings.
        available_version_split_list = available_version.split('.')
        target_version_split_list = self.k8s_version.split('.')

        # the major version should be less than or equal to the target
        major_version_available_str = available_version_split_list[0]
        major_version_desired_str = target_version_split_list[0]
        if  major_version_available_str != major_version_desired_str:
            major_version_available_int = major_version_available_str[1:]
            major_version_desired_int = major_version_desired_str[1:]

            # available > desired. Skip the upgrade
            if major_version_available_int > major_version_desired_int:
                return False

        # Check the minor versions
        # If the available version is less than target, upgrade to the lower
version. Cluster upgrade is one version
        # at a time.
        if available_version_split_list[1] < target_version_split_list[1]:
            return True
```

```
        # Attempt to upgrade to a lower cluster version. Reject.
        elif available_version_split_list[1] > target_version_split_list[1]:
            return False
        # Minor versions match. Check the patch version
        elif available_version_split_list[1] == target_version_split_list[1]:
            if available_version_split_list[2] !=
target_version_split_list[2]:
                return False
            else:
                return True

    # Upgrade all the node pools  to the target k8s version. This should be
done after the cluster upgrade.
    def upgrade_nodepools(self):
        print ("Upgrading node pools ...")
        # Get the list of nodepools
        comp_id =
self.ce_client.get_cluster(self.cluster_id).data.compartment_id
        node_pools_list = self.ce_client.list_node_pools(comp_id).data

        for node_pool_summary in node_pools_list:
            # check if the node pool kubernetes version is target version, if
yes, skip update
            if node_pool_summary.kubernetes_version == self.k8s_version or
node_pool_summary.cluster_id != self.cluster_id:
                continue

            # Delete the nodes in the nodepool. Update Node Pool will create
new nodes with the updated k8s version
            node_pool =
self.ce_client.get_node_pool(node_pool_summary.id).data
            for node in node_pool.nodes:
                print ("Deleting node %s from node pool %s" % (node.name,
node_pool.name))
                if node.lifecycle_state ==
oci.container_engine.models.Node.LIFECYCLE_STATE_ACTIVE:
                    response =
self.ce_client_ops.delete_node_and_wait_for_state (
                        node_pool.id, node.id,

wait_for_states=[oci.container_engine.models.WorkRequest.STATUS_SUCCEEDED,

oci.container_engine.models.WorkRequest.STATUS_FAILED])

                    if response.status ==
oci.container_engine.models.WorkRequest.STATUS_FAILED:
                        print ("Failed to delete nodes in the node pool %s.
Continuing with the upgrade..." % node_pool.name)

            # Make NodeConfigDetails object
            node_config_details =
oci.container_engine.models.UpdateNodePoolNodeConfigDetails(
                size = node_pool_summary.node_config_details.size
            )
            # Make NodeSourceDetails object
            node_source_details =
```

```
oci.container_engine.models.NodeSourceViaImageDetails(
                source_type =
oci.container_engine.models.NodeSourceDetails.SOURCE_TYPE_IMAGE,
                image_id = node_pool_summary.node_image_id
            )
            # Make the UpdateNodePoolDetails
            update_nodepool_details =
oci.container_engine.models.UpdateNodePoolDetails(
                kubernetes_version = self.k8s_version,
                node_config_details = node_config_details,
                node_source_details = node_source_details
            )
            print ("Upgrading kuberenetes for node pool %s " % node_pool.name)
            response = self.ce_client_ops.update_node_pool_and_wait_for_state
(
                node_pool_summary.id, update_nodepool_details,

wait_for_states=[oci.container_engine.models.WorkRequest.STATUS_SUCCEEDED,

oci.container_engine.models.WorkRequest.STATUS_FAILED])

            if response.status ==
oci.container_engine.models.WorkRequest.STATUS_FAILED:
                print ("Failed to upgrade node pool %s" % node_pool.name)
                sys.exit(1)

        print ("Successfully upgraded node pools  in cluster %s" %
self.ce_client.get_cluster(self.cluster_id).data.name)
        return


if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python3 upgrade_cluster.py <target_k8s_version>
[<cluster, nodepool>]")
        sys.exit(1)

    k8s_version = sys.argv[1]
    if not re.search("^(v1.)2[4-6]\.{1}[0-9]{1,5}$", k8s_version):
        print ("Invalid/Unsupported kubernetes version provided for upgrade.
Supported versions are "
               "[v1.24.1, v1.25.4, v1.26.2].")
        sys.exit(1)

    upgrade = UpgradeCluster(k8s_version)

    # If option 'cluster' is provided, just upgrade the cluster, if
'nodepool' is provided, upgrade only the nodepool
    if len(sys.argv) > 2:
        component = sys.argv[2]
        if component.lower() == 'cluster':
            upgrade.upgrade_cluster()
        elif component.lower() == 'nodepool':
            upgrade.upgrade_nodepools()
        else:
            print ("unrecognized parameter %s. Provide one of [cluster,
```

```
nodepool]." % component)
            sys.exit(1)

    # If second argument is not provided, upgrade both cluster and nodepool.
    upgrade.upgrade_cluster()
    upgrade.upgrade_nodepools()
```