Oracle® Cloud

Using Oracle WebLogic Server for OCI

F74163-02
August 2023

Oracle Cloud Using Oracle WebLogic Server for OCI,

F74163-02

Primary Author: Oracle Corporation

Contents

Preface

Documentation Accessibility x
Diversity and Inclusion x

1 Get Started

About Oracle WebLogic Server for OCI 1-1
About the Components of Oracle WebLogic Server for OCI 1-2
Oracle WebLogic Server 1-3
Marketplace 1-4
Resource Manager 1-4
Compute 1-5
Virtual Cloud Network 1-6
Load Balancer 1-6
Database 1-7
Vault 1-10
Identity 1-10
About Oracle WebLogic Server for OCI Versions and Retirement Policy 1-12
Retirement Policy 1-12
About Observability 1-13
Logging 1-13
Application Performance Monitoring 1-13
Application Performance Monitoring Dashboard 1-14
Autoscaling 1-16
Before You Begin with Oracle WebLogic Server for OCI 1-22
Understand Service Requirements 1-23
Create a Compartment 1-24
Create Compartment Policies 1-24
Create Root Policies 1-26
Create Dynamic Groups and Policies 1-26
Create a Dynamic Group 1-27
Create Policies for the Dynamic Group 1-27
Create Dynamic Groups and Policies for Observability 1-29

Create Additional Policies for Observability 1-31
Create an Encryption Key 1-33
Create Secrets for Passwords 1-33
Create an SSH Key 1-34
Create a Virtual Cloud Network 1-34
Create a Private Subnet for the Oracle WebLogic Server Nodes 1-35
Configure the Load Balancer 1-38
Create a Subnet for the Load Balancer 1-40
Create a Subnet for the Bastion Node 1-41
Create a Database 1-42
Create a Confidential Application 1-44
Create an Application Performance Monitoring Domain 1-45
Create an Auth Token 1-46
Validate Existing Network Setup 1-46
Using the Validation Script 1-47

2 Create a Stack

About Creating a Domain 2-1
Create a Basic Domain 2-3
Launch a Stack 2-4
Specify Stack Information 2-5
Configure WebLogic Instance Parameters 2-5
Configure Advanced Parameters for a Domain 2-7
Configure Network Parameters 2-7
Configure a WebLogic Console Port 2-8
Configure a Load Balancer 2-8
Configure File Storage 2-10
Create OCI Policies 2-11
Configure WebLogic Authentication 2-11
Configure Database Parameters 2-12
Configure a Data Source for an Application Database 2-12
Set Local VCN Peering for an Application Database 2-13
Configure Observability 2-14
Configure Autoscaling 2-15
Configure Tags 2-15
Create the Domain Stack 2-16
Use Your New Domain 2-16
Create a Basic Domain in a Private Subnet 2-17
Launch a Stack 2-18
Specify Stack Information 2-19

Configure WebLogic Instance Parameters 2-19

Configure Advanced Parameters for a Domain 2-20

Configure Network Parameters 2-21

Configure a Bastion 2-22

Configure a Load Balancer 2-22

Configure File Storage 2-24

Create OCI Policies 2-25

Configure WebLogic Authentication 2-25

Configure Database Parameters 2-26

Configure a Data Source for an Application Database 2-26

Set Local VCN Peering for an Application Database 2-27

Configure Observability 2-28

Configure Autoscaling 2-29

Configure Tags 2-29

Create the Domain Stack 2-30

Use Your New Domain 2-30

Create a JRF-Enabled Domain 2-31

Launch a Stack 2-32

Specify Stack Information 2-33

Configure WebLogic Instance Parameters 2-33

Configure Advanced Parameters for a Domain 2-35

Configure Network Parameters 2-35

Configure a WebLogic Console Port 2-37

Configure a Load Balancer 2-37

Configure File Storage 2-39

Create OCI Policies 2-40

Configure WebLogic Authentication 2-40

Configure Database Parameters 2-40

Set Local VCN Peering 2-42

Configure a Data Source for an Application Database 2-43

Set Local VCN Peering for an Application Database 2-45

Configure Observability 2-45

Configure Autoscaling 2-46

Configure Tags 2-47

Create the Domain Stack 2-47

Use Your New Domain 2-47

View the Cloud Resources for a Stack 2-48

About the Resources in a Stack 2-48

Compute Instances 2-49

Network Resources 2-49

Load Balancer 2-51

Identity Resources for Dynamic Group and Root Policy 2-51
Identity Resources for Oracle Identity Cloud Service 2-52

3 Manage a Domain

About Managing a Domain 3-1
About the Default Ports 3-2
About the Security Checkup Tool 3-3
Access the WebLogic Console 3-6
Access the WebLogic Console in a Public Subnet 3-6
Access the WebLogic Console in a Private Subnet 3-7
Access by Using the Bastion Instance 3-7
Access by Using the Bastion Service 3-8
Access the WebLogic Console Through a Load Balancer 3-9
Access the Fusion Middleware Control Console 3-11
Access the Fusion Middleware Control Console in a Public Subnet 3-11
Access the Fusion Middleware Control Console in a Private Subnet 3-12
Access the Sample Application 3-13
Access the Sample Application Through a Public Load Balancer 3-14
Access the Sample Application Through a Private Load Balancer 3-14
Access the Sample Application Without a Load Balancer 3-16
Access the Sample Application in a Private Subnet 3-17
Access the Sample Application Using Identity Cloud Service 3-18
Start and Stop a Domain 3-19
Add IDCS after Creating a Domain 3-20
Back Up and Restore a Domain 3-22
About Volume Backups 3-22
About Database Backups 3-23
Create a Backup 3-24
Restore a Backup 3-24
Restore to a Previous Domain 3-25
Recover from an Operating System Failure 3-26
Repair a Boot Volume 3-27
Add a Certificate to the Load Balancer 3-29
Secure a Domain Using Identity Cloud Service 3-30
Create WebLogic Administrator Groups 3-31
Update WebLogic Administrator Roles 3-31
Update Protected Application Resources 3-32
Update Application Deployment Descriptors 3-33
Secure Web Services Using Identity Cloud Service 3-34
Deploy a Sample Web Service Client Application 3-35

Get Information About Identity Cloud Service 3-40
Configure OAuth for the Web Services Provider 3-41
Update the Confidential Application for the Web Services Provider 3-44
Configure OAuth for the Web Services Client 3-45
Update the Confidential Application for the Web Services Client 3-47
Test the Sample Web Service Client 3-48
Integrate OPSS User and Group APIs with Identity Cloud Service 3-48
Upgrade the Oracle Identity Cloud Service App Gateway Version 3-49
Configure Session Persistence 3-55
Enable Session Affinity or Sticky Sessions 3-56
Update the Password Secret OCID and Policy 3-56
Manage Data Sources 3-57
Create a Data Source for an Oracle Autonomous Database 3-57
Download the Oracle Autonomous Database Wallet 3-58
Configure a Data Source for an Oracle Autonomous Database 3-60
Create a Data Source for a DB System Database 3-62
Verify the PDB Name 3-63
Configure a Data Source for a DB System Database 3-63
Connect to a Domain Using Oracle JDeveloper 3-64
Add an Ingress Rule 3-65
Create a Network Channel 3-66
Create a Connection to the Domain Through Oracle JDeveloper 3-67
Upgrade a Domain 3-67
Backup the Database 3-68
Create a 12.2.1.4 Instance 3-68
Stop the WebLogic Server Processes on the Source 12.2.1.3 Instance 3-68
Replace the Domain on the Target 12.2.1.4 Instance with Cloning 3-69
Set up VNC Server 3-70
Perform Readiness Check 3-72
Upgrade Infrastructure Schemas 3-73
Reconfigure the Domain 3-74
Upgrade the Domain 3-75
Restart Servers 3-76
Post Upgrade 3-76
Roll Back Upgrade 3-77

4 Scale a Stack

Add or Remove WebLogic Server Nodes 4-1
Add UCM WebLogic Server Node to a BYOL Stack 4-3
Manage Autoscaling Resources 4-4

Create Alarm Definitions 4-4
Update Alarm Definitions 4-5
Configure Function Application 4-6
Reenable Autoscaling for a Stack 4-7
Change the Shape of the Existing Compute Instances 4-8
Change Reserved Public IP Usage 4-9
Add a Load Balancer 4-9
Remove the Load Balancer 4-11
Add a File Storage 4-12
Remove the File Storage 4-13

5 Clone a Stack

Move a WebLogic Server Instance 5-2
Clone a JRF or non-JRF Instance using a Script 5-2
Copy a WebLogic Server Instance 5-5
Clone a non-JRF Instance 5-5
Scale Out the Cloned Instance 5-10

6 Delete a Stack

Delete a JRF Database Schema 6-1
Delete the Identity Cloud Service Resources 6-3
Delete Autoscaling Resources 6-4
Destroy Stack Resources 6-5
Delete the Stack 6-6
Delete the Database Security List 6-7
Delete the Identity Cloud Service Resources Manually 6-7

7 Troubleshoot

Check Known Issues 7-1
Clone Script Failed 7-1
Cleanup Resources of a Deleted Instance 7-2
Stack Creation Failed 7-17
Error in Mounting the Volume During Provisioning 7-22
Unable to Access the Domain 7-23
Load Balancer does not send Cookie X-Oracle-BMC-LBS-Route 7-25
Autoscaling Failed to Create Functions 7-25
Management Agents Are Not Deleted on Instance Termination 7-27
Enterprise Manager Console Is Not Loading 7-27
Scale Out Fails on the Administration Compute Instance 7-28

Enable OS Management to Install Patches 7-29
Security Checkup Tool Warnings 7-29
Running python3 Command Fails 7-31
Get Additional Help and Contact Support 7-32

8 Patches

About Patching Utility Tool 8-9
Patch Management Using Patching Utility 8-10
    View Patching Tool Version 8-10
    Configure Initial Setup 8-11
    List Patches 8-12
    List Pending Patches 8-14
    View Patch Details 8-15
    Download Patches 8-16
    Apply Patches 8-17
    Roll Back Patches 8-21
    Upgrade Patching Tool 8-24

A License Information

B Configure SSL for a Domain

Create an HTTPS Listener for the Load Balancer B-1
Add a Certificate to the Load Balancer B-3
Update the App Gateway for HTTPS B-4

C Script Files

Script File to Validate Network Setup C-1
Script File to Create Policies for Autoscaling Functions Post Provisioning C-16
Script File to Delete Resources C-24

D Migrate Terraform Scripts

E Create a Domain for Oracle WebLogic Server 12.2.1.3.0 Using Terraform

Preface

Using Oracle WebLogic Server for Oracle Cloud Infrastructure Oracle WebLogic Server for OCI explains how to create an Oracle WebLogic Server domain in Oracle Cloud Infrastructure Marketplace, and then administer the domain and deploy Java EE applications.

Topics:

- Documentation Accessibility
- Diversity and Inclusion

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 (Oracle WebLogic Server for OCI), and perform any prerequisite tasks.

Topics

- About Oracle WebLogic Server for OCI
- About the Components of Oracle WebLogic Server for OCI
- About Oracle WebLogic Server for OCI Versions and Retirement Policy
- About Observability
- Before You Begin with Oracle WebLogic Server for OCI

About Oracle WebLogic Server for OCI

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

Oracle WebLogic Server for OCI 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 and provision your domains along with any supporting cloud resources like compute instances, networks and load balancers.

You can track and monitor the progress of an Oracle WebLogic Server for OCI stack in Resource Manager. As your workload requirements change, you can use Resource Manager to scale the domain by adding or removing managed servers. 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 domain, you can administer the domain and deploy Java EE applications to it just like on-premises domains. Use standard Oracle WebLogic Server tools like the administration console and WebLogic Scripting Tool (WLST). You can also administer the operating system on the compute instances using a secure shell (SSH) client and standard Linux tools.

Oracle WebLogic Server for OCI can also create a domain that includes the Java Required Files (JRF) components. A JRF-enabled domain:

- Supports the Oracle Application Development Framework (ADF)
- Can be administered and monitored using the Oracle Fusion Middleware Control console
- Connects to an existing database in Oracle Cloud Infrastructure

After creating a JRF-enabled domain with Oracle WebLogic Server for OCI, you can install additional Oracle Fusion Middleware products onto the domain if you have existing on-premise licenses for these products.

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

- Oracle WebLogic Server Standard Edition
- Oracle WebLogic Server Enterprise Edition
- Oracle WebLogic Suite

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

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

> **Note:**
>
> From release 21.4.3 (December 9, 2021) onwards, you cannot provision a domain in Oracle Oracle WebLogic Server for OCI for Oracle WebLogic server versions 11g (10.3.6.0) and 12c (12.2.1.3) from the Marketplace.
> However, you can provision a domain for Oracle WebLogic server version 12c (12.2.1.3) using the terraform scripts. See Create a Domain for Oracle WebLogic Server 12.2.1.3.0 Using Terraform.
> The WebLogic binaries for 12.2.1.3 are also available in the public images.

Oracle WebLogic Server for OCI supports these billing options:

- Universal Credits (also called UCM) - You are billed for the cost of the Oracle WebLogic Server Enterprise Edition or Oracle WebLogic Suite license (based on OCPUs per hour) in addition to the cost of the compute resources.
- Bring Your Own License (BYOL) - You reuse your existing on-premises Oracle WebLogic Server Standard Edition, Oracle WebLogic Server Enterprise Edition, or Oracle WebLogic Suite licenses in Oracle Cloud. You are billed only for the cost of the compute resources.

Oracle WebLogic Server Standard Edition is available only as BYOL.

Oracle WebLogic Server for OCI requires Oracle Cloud Infrastructure Vault (formerly known as Key Management) in order to securely store the passwords for your domains. See Oracle Cloud Infrastructure Vault FAQ.

About the Components of Oracle WebLogic Server for OCI

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

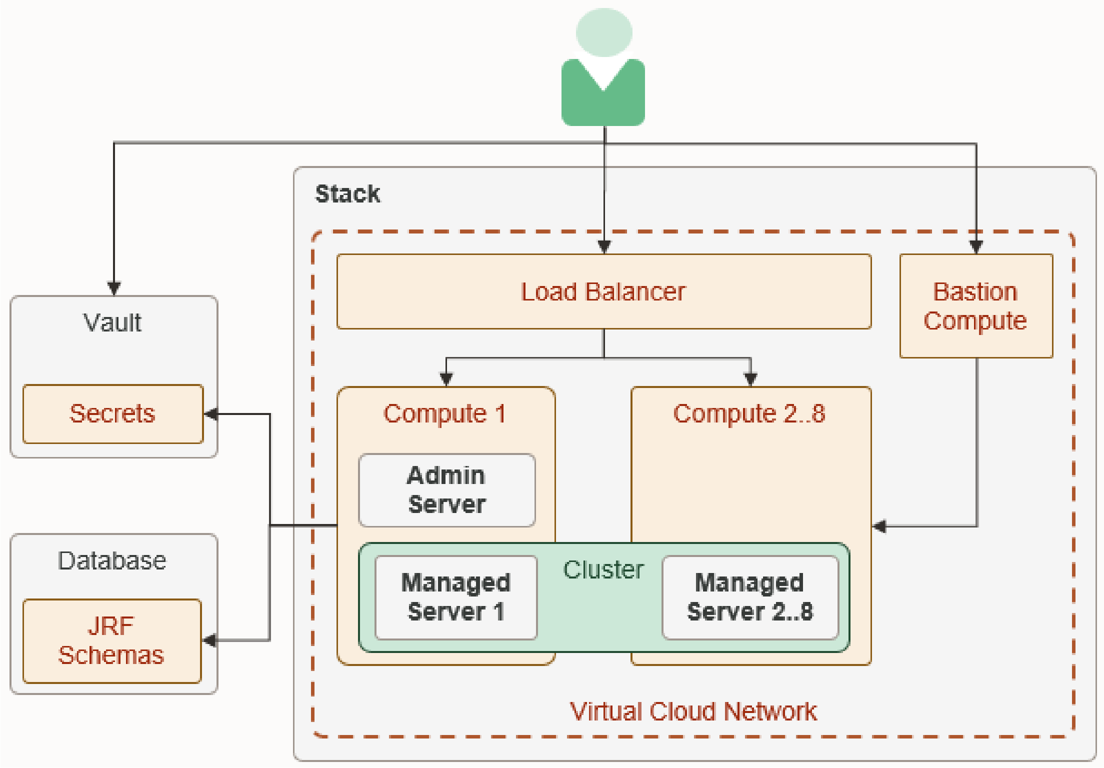
Topics

- Oracle WebLogic Server
- Marketplace
- Resource Manager
- Compute
- Virtual Cloud Network
- Load Balancer
- Database

- Vault
- Identity

The following diagram illustrates the components of a typical Oracle WebLogic Server for OCI deployment.

Figure 1-1 Components of a Typical Deployment



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 OCI supports these Oracle WebLogic Server editions:

- Oracle WebLogic Server Standard Edition
- Oracle WebLogic Server Enterprise Edition
  - Includes all features and benefits of Oracle WebLogic Server Standard 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 Oracle Coherence for increased performance and scalability
  - Includes Active Gridlink for RAC for advanced database connectivity

Oracle WebLogic Server for OCI does not provision a cluster in domains running WebLogic Server Standard Edition.

Oracle Cloud Infrastructure OS Management service is enabled on Oracle WebLogic Server for OCI instance. See Getting Started with OS Management.

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

Oracle WebLogic Server for OCI can create these domain configurations:

- A basic domain that does not require a database (Oracle WebLogic 12c only).
- A domain that includes the Java Required Files (JRF) components and also requires a database. A JRF-enabled domain:
  - Supports the Oracle Application Development Framework (ADF)
  - Can be administered and monitored using the Oracle Fusion Middleware Control console, as well as the standard Oracle WebLogic Server tools

Marketplace

Oracle WebLogic Server for OCI 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 OCI 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 domain and supporting cloud resources.

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

See Overview of Marketplace in the Oracle Cloud Infrastructure documentation.

Resource Manager

Oracle WebLogic Server for OCI uses Resource Manager in Oracle Cloud Infrastructure to provision the cloud instances and networks that support your Oracle WebLogic Server domain.

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 OCI is comprised of these components:

- A compute instance running the administration server and the first managed server
- A compute instance for each additional managed server in the domain
- A bastion compute instance that provides administrative access to a domain on a private subnet

- A virtual cloud network (VCN), including subnets, route tables, and security lists (optional)
- A load balancer (optional)

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

Compute

The servers that make up an Oracle WebLogic Server domain run on one or more Oracle Cloud Infrastructure Compute instances.

Oracle WebLogic Server for OCI creates Oracle Linux compute instances, and automatically installs the Oracle WebLogic Server software and creates the domain configuration on these instances.

> **Note:**
>
> The Oracle Linux version of the compute instances is 7.9.

You select the Oracle WebLogic Server edition and version you want to provision. If you plan to run Oracle WebLogic Server Standard Edition and require more than 4 nodes, then create a domain that runs Oracle WebLogic Server 12c Standard Edition. For all other editions and versions, the maximum is 8 nodes, which can be scaled out to 30 when you edit the domain.

You assign a shape to a domain, which determines the number of CPUs and the amount of memory allocated to each compute instance in the domain. If you create a domain in a private subnet, you can assign a different shape to the bastion compute instance. Oracle Cloud Infrastructure offers a variety of bare metal (BM) and virtual machine (VM) shapes. However, Oracle WebLogic Server for OCI only supports the following shapes:

- Standard: `VM.Standard2.x`, `VM.Standard.E2.x`, `BM.Standard2.x`, `BM.Standard.E2.x`, `BM.Standard3.64`
- Flexible: `VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`
- Optimized: `BM.Optimized3`, `VM.Optimized3.Flex`

> **NOT_SUPPORTED:**
>
> Some shapes might not be available in all regions.

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.

All of the compute instances for a domain are created in a single availability domain (AD). An availability domain represents a data center within an Oracle Cloud Infrastructure region. Each availability domain contains three fault domains. Oracle WebLogic Server for OCI automatically distributes the compute instances across these availability domains for high availability. If a single AD is available, the VMs are spread across fault domains.

> **Note:**
>
> In a regional subnet, if you use shapes with service limits that are set for an availability domain, then for high availability the fault domains are used.

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

Virtual Cloud Network

Oracle WebLogic Server for OCI assigns compute instances and load balancers 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 subnet is a subdivision of a VCN that consists of a contiguous range of IP addresses that do not overlap with other subnets in the VCN. A VCN includes one or more subnets, route tables, security lists, gateways, and DHCP options.

Oracle WebLogic Server for OCI can automatically create a VCN and subnets for a new Oracle WebLogic Server 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.

By default subnets are public. Any compute instances assigned to a private subnet can not be directly accessed from outside of Oracle Cloud. To enable the administration of compute instances in a private subnet, Oracle WebLogic Server for OCI can create a separate public subnet and bastion compute instance. Oracle WebLogic Server for OCI can also create a service gateway in a VCN so that compute instances can access other cloud services like Key Management and Oracle Autonomous Database, without using the public Internet.

If you already have an existing bastion to provide public access to the compute instances, or if you already have a VPN connection to your on-premise network, then you can delete the bastion instance created by Oracle WebLogic Server for OCI.

> **Note:**
>
> Configuring a bastion is optional.
>
> If you do not configure a bastion, no status is returned for provisioning. You must check the status of provisioning by connecting to each compute instance and confirm that the `/u01/provStartMarker` file exists with details found in the file `/u01/logs/provisioning.log` file. See Configure a Bastion.

See Overview of Networking in the Oracle Cloud Infrastructure documentation.

Load Balancer

Oracle Cloud Infrastructure Load Balancing routes requests it receives from clients to the managed servers in your Oracle WebLogic Server domain.

When you create a domain, Oracle WebLogic Server for OCI can automatically create a load balancer in Oracle Cloud Infrastructure and configure it to distribute traffic across the servers in your domain. Using a load balancer is recommended if your cluster size is greater than one.

By default, the load balancer is public. You can also provision a public load balancer with a reserved public IP. If you create a domain in a private subnet, then you can provision a public or private load balancer.

A private load balancer does not have a public IP address and cannot be accessed from outside of Oracle Cloud, unless you have configured a virtual private network (VPN) between your VCN and your on-premise data center.

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

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

Session persistence is a method to direct all requests originating from a single logical client to a single backend server. By default, session persistence is enabled on the load balancer with the `Enable Load balancer cookie persistence` option, but you can update the load balancer after creating a domain.

See these topics in the Oracle Cloud Infrastructure documentation:

- Overview of Load Balancing
- VPN Connect
- Session Persistence

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.
> - Oracle Application Express (APEX) is not supported.

Choose one of these database options:

- Oracle Autonomous Database
  - Both dedicated and shared infrastructure autonomous database options are supported.

  - See Overview of the Autonomous Database in the Oracle Cloud Infrastructure documentation.

  > **Note:**
  >
  > From 22.1.2 release (February 24, 2022) onwards, Free-Tier autonomous database is supported.

- Oracle Cloud Infrastructure Database
  - Bare metal, virtual machine (VM), and Exadata DB systems are supported.
  - For a 1-node VM DB system, you can use the fast provisioning option to create the database. Oracle WebLogic Server for OCI supports using Logical Volume Manager as the storage management software for a 1-node VM DB system.
  - For Oracle WebLogic Server 12c, you can also specify a database connection string. This database connection string can be used only with existing VCN. To know the database connection string details, see Database Connect String for Database Version and Type in Configure Database Parameters.
  - See Overview of the Database Service in the Oracle Cloud Infrastructure documentation.

Oracle WebLogic Server for OCI 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)

> **Note:**
>
> From release 21.4.3 (December 9, 2021) onwards, you cannot provision a domain in Oracle Oracle WebLogic Server for OCI for Oracle WebLogic server versions 11g (10.3.6.0) and 12c (12.2.1.3) from the Marketplace. However, you can provision a domain for Oracle WebLogic server version 12c (12.2.1.3) using the terraform scripts See Create a Domain for Oracle WebLogic Server 12.2.1.3.0 Using Terraform. The WebLogic binaries for 12.2.1.3 are also available in the public images.

When you create a domain and associate it with an existing database, Oracle WebLogic Server for OCI 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

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

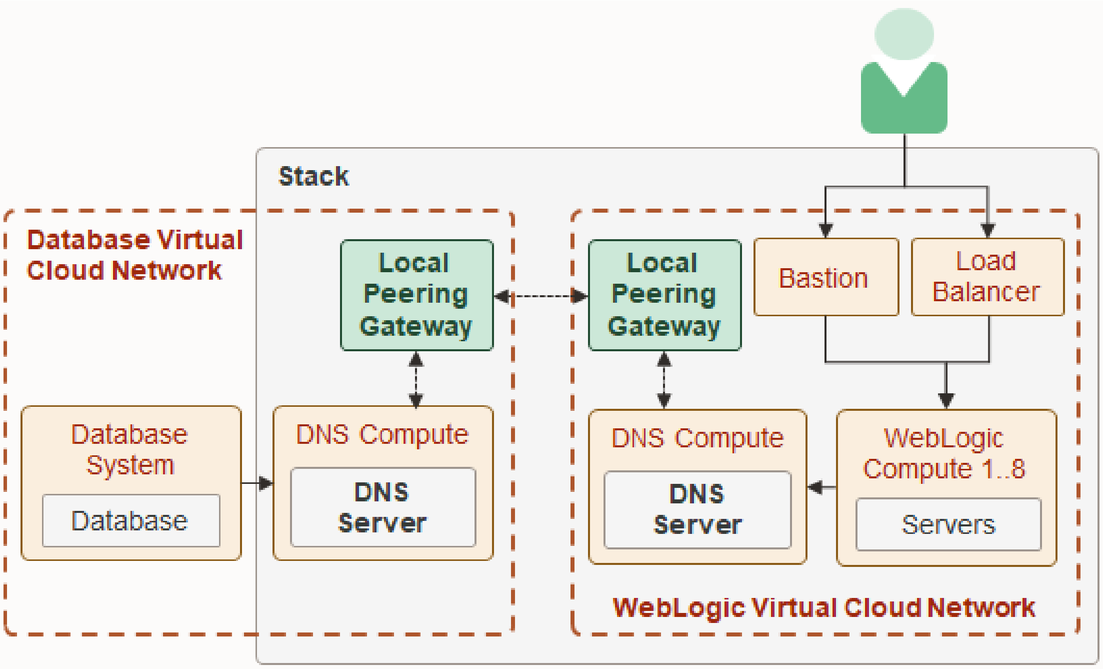
See Understanding JDBC Resources in WebLogic Server in *Administering JDBC Data Sources for Oracle WebLogic Server*.

> **Note:**
>
> If you use database connect string, then Oracle WebLogic Server for OCI 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 you use a private subnet for WebLogic Server and Oracle Autonomous Database, Oracle WebLogic Server for OCI uses a service gateway in the VCN to access the database. The service gateway provides network access to cloud service without using the public Internet. See Access to Oracle Services: Service Gateway in the Oracle Cloud Infrastructure documentation.

If your Oracle Cloud Infrastructure Database is on a different VCN than the VCN you want to use for WebLogic Server, then Oracle WebLogic Server for OCI creates a Local Peering Gateway in each VCN so that they are able to communicate. Oracle WebLogic Server for OCI also creates a separate public subnet and compute instance in each VCN to forward Domain Name Service (DNS) traffic across the VCNs.

This configuration is called local VCN peering, and it is illustrated by the following diagram.



If you use Oracle Cloud Infrastructure Database with connect string, you must peer the VCNs manually before creating the stack if your database VCN is on a different VCN than the WebLogic Server VCN.

If you want multiple JRF-enabled domains to use the same Oracle Cloud Infrastructure Database, then you cannot use local VCN peering. For this case, you must create the domains and the database in the same VCN.

If you choose to create a virtual cloud network for an Oracle WebLogic Server domain, use Oracle WebLogic Server for OCI to create a Local Peering Gateway, else create a network with VCN peering and then use this existing network to provision the domain.

Local VCN peering cannot be used to connect WebLogic Server to an Oracle Cloud Infrastructure Database in a different region. See Local VCN Peering in the Oracle Cloud Infrastructure documentation.

Vault

Oracle Cloud Infrastructure Vault (formerly known as Key Management) enables you to manage sensitive information using vaults, keys, and secrets when creating an Oracle WebLogic Server domain.

A vault is a container for encryption keys and secrets. 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.

Secrets store credentials such as required passwords for a new domain. You use an encryption key in a vault to encrypt and import secret contents to the vault. Secret contents are based64-encoded. Oracle WebLogic Server for OCI uses the same key to retrieve and decrypt secrets when creating the domain.

Parameters for a new domain include:

- The secret for the password for the default Oracle WebLogic Server administrator
- The secret for the administrator password for an existing database, if you are creating a domain that includes the Java Required Files (JRF) components
- The secret for the client secret for an existing confidential application, if you are creating a domain that uses Oracle Identity Cloud Service for authentication

By default, Oracle WebLogic Server for OCI creates a dynamic group and root policy to allow compute instances to access your keys and secrets.

See:

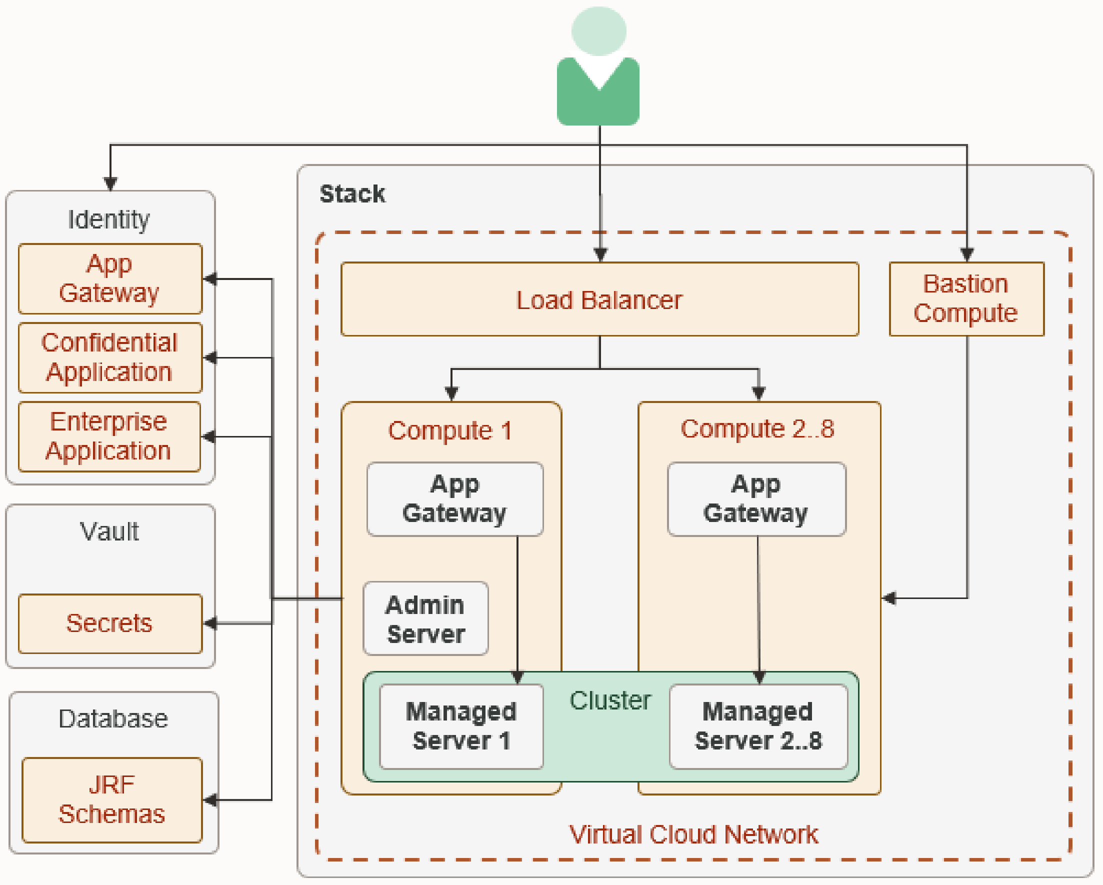
- Overview of Vault in the Oracle Cloud Infrastructure documentation
- Oracle Cloud Infrastructure Vault FAQ

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 OCI can configure a domain running WebLogic Server 12c to use Oracle Identity Cloud Service for authentication. The following diagram illustrates this configuration.



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

Oracle WebLogic Server for OCI configures an App Gateway in Oracle Identity Cloud Service. It also provisions each compute instance in the domain with the App Gateway software appliance. 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 OCI 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.

If you enable integration with Oracle Identity Cloud Service for a domain, then you must also enable a load balancer for the domain.

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

About Oracle WebLogic Server for OCI Versions and Retirement Policy

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

A version number for Oracle WebLogic Server for OCI contains six decimal places in the form:

```
WLS n.n.n.n.yymmdd.n
```

For example:

```
WLS 12.2.1.4.190716.01
WLS 14.1.1.0.190716.01
```

The first 4 decimal places together describe the base version of a WebLogic Server release, such as:

`12.2.1.4` for Oracle WebLogic Server 12c Release 3

`14.1.1.0` for Oracle WebLogic Server 14g Release 1

The 5th decimal place is a quarterly patch set release date. For example, `190716` is the July 2019 patch set.

For patch set update (PSU) naming purposes, releases in a quarterly patch set for Oracle WebLogic Server for OCI are named uniquely by the 6th decimal place. The first release is `01`. The number is incremented by one for every patch or every update of that patch set, including one-off patches and updates to the VM image, the WebLogic scripts placed on the VM images, the Terraform scripts, and the Resource Manager schema.

An Oracle WebLogic Server for OCI PSU is created from the Critical Patch Updates (CPUs) of several products, namely, Oracle WebLogic Server, Oracle JDeveloper, Oracle Java Development Kit, Oracle Platform Security Services and Oracle Web Services Manager. While the Oracle WebLogic Server for OCI quarterly patch set release date (`yymmdd`) is mainly derived from a WebLogic Server PSU, the latest PSUs of all those products are included. See Patches.

Retirement Policy

Oracle WebLogic Server for OCI PSUs do not retire based on any fixed time range. Instead, for each WebLogic Server release (12c or 14c), only the last two patch sets (identified by `yymmdd`) are retained in a quarter.

> **Note:**
>
> The Oracle WebLogic Server for OCI 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 WebLogic scripts on the VM.

For example, suppose at the end of December 2019 there are two July PSUs and two October PSUs for WebLogic Server release 12.2.1.4:

```
12.2.1.4.190716.01
12.2.1.4.190716.02
12.2.1.4.191016.01
12.2.1.4.191016.02
```

When the 2020 January PSU is released (say, `12.2.1.4.200116.01`), the July PSUs are retired and only the PSUs for October 2019 and January 2020 are retained:

```
12.2.1.4.191016.01
12.2.1.4.191016.02
12.2.1.4.200116.01
```

Note the following about all retained Oracle WebLogic Server for OCI versions:

- 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.

See Known Issues in Oracle WebLogic Server for Oracle Cloud Infrastructure for known problems in a retained version and how to work around them.

About Observability

Learn about the observability services in Oracle WebLogic Server for OCI.

Topics

- Logging
- Application Performance Monitoring
- Application Performance Monitoring Dashboard
- Autoscaling

Logging

Oracle WebLogic Server for OCI uses the Logging Service to view the server logs such as errors or warnings in the Oracle Cloud Infrastructure console.

The logging resources, Custom Logs, Log Groups, and Unified Agent Configuration, are created during stack provisioning. See Logging Concepts in the Oracle Cloud Infrastructure documentation.

Application Performance Monitoring

Application Performance Monitoring service monitors the performance of administration and managed servers in Oracle WebLogic Server for OCI domain and displays the server metrics in the Application Performance Monitoring dashboard. It also helps to understand workloads and alerts the user for any issues in the logs.

See About Application Performance Monitoring in the Oracle Cloud Infrastructure documentation.

Application Performance Monitoring Dashboard

You can use the Oracle-defined Application Performance Monitoring (APM) dashboard for Oracle WebLogic Server for OCI to view the WebLogic metrics that are exported using APM Java Agent. This Oracle-defined Application Performance Monitoring (APM) dashboard is called **WebLogic Domains**.

You must access the Dashboards page to view the **WebLogic Domains** dashboard. To access the dashboard, see Access Dasboards.

On the Dashboards page, when you click **WebLogic Domains**, you can view the dashboard and the following filter options:

- Compartment - The compartment where the APM domain resides.
- APM Domain - The APM domain where your metrics is located.
- WebLogic Domain - The name of the WebLogic domain for which you can view the metrics. This option lists all the domains of the selected APM domain. The servers that belongs to the selected WebLogic domain are displayed in the **WebLogic Server** filter.

  > **Note:**
  >
  > By default, Oracle WebLogic Server for OCI supports only one domain. However, you can create additional domains.

- WebLogic Server - The name of the server for which you can view the metrics. This option lists only the servers that belong to the selected WebLogic domain. But, if **All servers** option is selected, the widgets shows metrics for all the servers.

The **WebLogic Domains** dashboard page includes default widgets that can be used to monitor metrics such as CPU load, heap percentage, and stuck threads.

But, if you want to add widgets to the **WebLogic Domains** dashboard, you must first create a custom dashboard by creating a copy of the **WebLogic Domains** dashboard, and then add widgets to your newly created custom dashboard. See Customize an Oracle-defined Dashboard.

The following table lists the widgets that are available by default in the **WebLogic Domains** dashboard.

Table 1-1 Widgets Displayed by Default in the WebLogic Domains Dashboard

Widget Name	Description	Type
WebLogic CPU load (System and Process)	Displays the process and system CPU load (one line per metric) for the selected server. If multiple servers are selected, the widget displays the mean of each metric of all selected servers.	Line chart, aggregated
WebLogic Free Heap percentage	Displays the free heap percentage for the selected server. If multiple servers are selected, the widget displays the mean of the load of all selected servers.	Line chart, aggregated

Table 1-1 (Cont.) Widgets Displayed by Default in the WebLogic Domains Dashboard

Widget Name	Description	Type
WebLogic Free Heap percentage (by Server)	Displays the free heap percentage for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic Process CPU load (by Server)	Displays the process CPU for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic Stuck threads (by Server)	Displays the stuck threads for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic Queue Length (by Server)	Displays the queue length for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server

The following table lists the widgets that you can add to your newly created custom dashboard.

Table 1-2 Widgets for Your Custom Dashboard

Widget Name	Description	Type
WebLogic GC Young Time (by Server)	Displays the garbage collection time for the young generation for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic GC Old Time (by Server)	Displays the garbage collection time for the old generation for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic Servers	Displays a table with the following information for the selected servers: • Heap Free Percentage Average • Last WebLogic Server State • Process CPU Load Average • System CPU Load Average	Table
WebLogic Execute Thread Total Count (by Server)	Displays the `WeblogicThreadPoolExecuteThreadTotalCount` for the selected server. If multiple servers are selected, the widget displays one line for each selected server	Line chart, grouped by server
WebLogic Heap Committed (by Server)	Displays the amount of heap memory committed for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic Heap Used (by Server)	Displays the amount of heap memory used for the selected server. If multiple servers are selected, the widget displays one line for each selected server.	Line chart, grouped by server
WebLogic Threadpool	Displays the number of stuck threads, execute thread total count and queue length (one line per metric) for the selected server. If multiple servers are selected, the widget displays the mean of each metric for all selected servers.	Line chart, aggregated

Table 1-2 (Cont.) Widgets for Your Custom Dashboard

Widget Name	Description	Type
WebLogic GC Time	Displays the garbage collection total time for young and old generation (one line per metric) for the selected server. If multiple servers are selected, the widget displays the mean of each metric for all selected servers.	Line chart, aggregated
WebLogic GC Count	Displays the garbage collection total count for young and old generation (one line per metric) for the selected server. If multiple servers are selected, the widget displays the mean of each metric for all selected servers.	Line chart, aggregated
WebLogic Heap Usage	Displays the amount of heap used memory and heap committed memory (one line per metric) for the selected server. If multiple servers are selected, the widget displays the mean of each metric for all selected servers.	Line chart, aggregated
WebLogic Non Heap Usage	Displays the amount of non-heap used memory and non-heap committed memory (one line per metric) for the selected server. If multiple servers are selected, the widget displays the mean of each metric for all selected servers	Line chart, aggregated
WebLogic Last Known Status	Displays the last known status of the selected server (e.g. Running, admin).	SingleValue
WebLogic Open Sockets	Displays the last known value of open sockets of the selected server.	SingleValue
WebLogic Active Threads	Displays the last known value for active threads of the selected server, based on the formula: `"( WeblogicThreadPoolExecuteThreadTotalCountLast.value - (WeblogicThreadPoolStandbyCountLast.value + WeblogicThreadPoolExecuteThreadIdleCountLast.value + WeblogicThreadPoolStuckCountLast.value))"`	SingleValue

Autoscaling

Oracle WebLogic Server for OCI uses autoscaling to automatically manage the size and lifecycle state of your WebLogic compute instances.

Before you enable autoscaling:

- You must create an Application Performance Monitoring domain or use an existing APM domain. See Create an APM Domain in the Oracle Cloud Infrastructure documentation.

  Note:

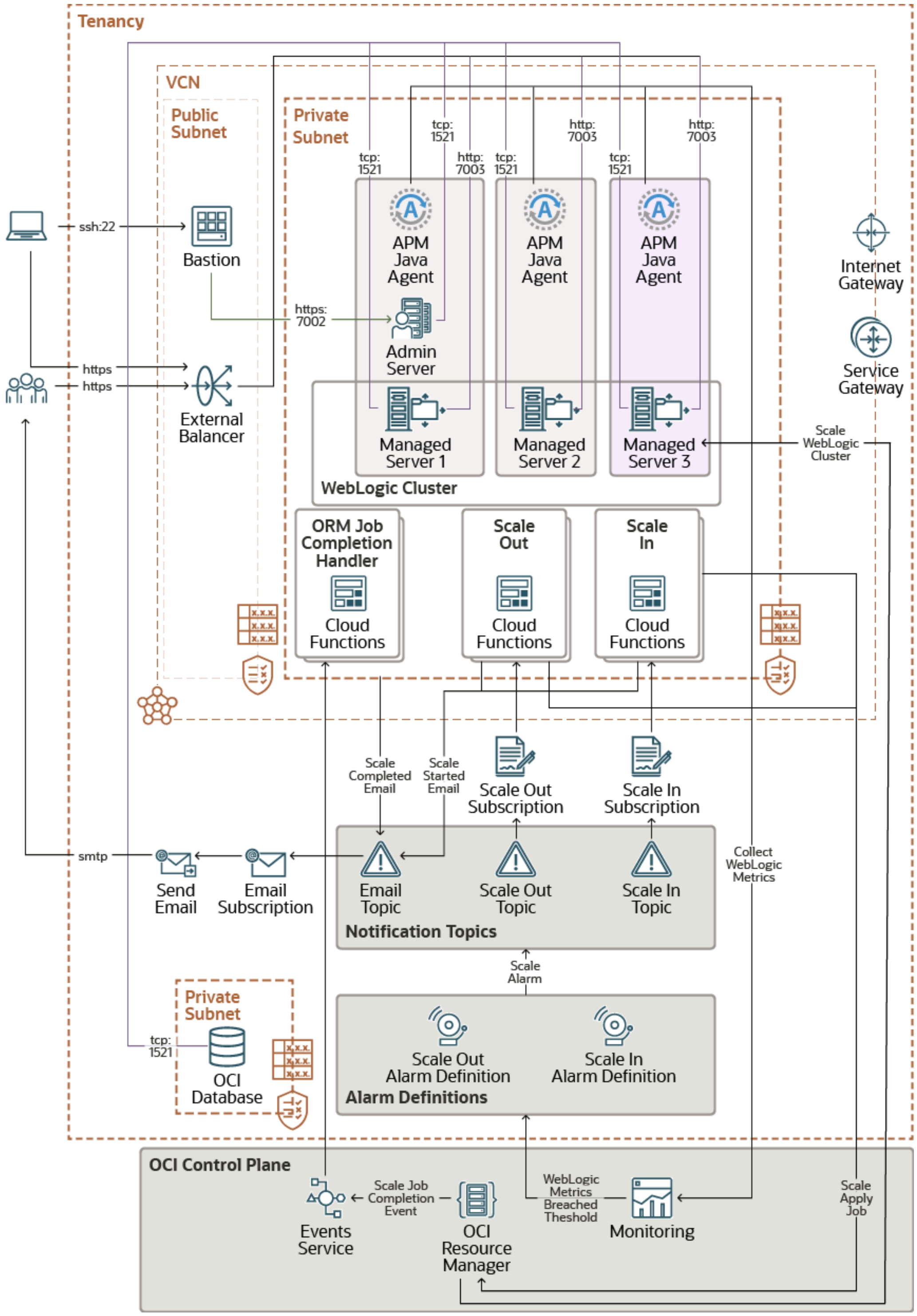
  Application Performance Monitoring always free domain is not supported for autoscaling.

- Create a new auth token for a user with access to Oracle Cloud Infrastructure Registry, or use an existing auth token. See Managing User Credentials in the Oracle Cloud Infrastructure documentation.

- Create policies for an Oracle Cloud Infrastructure user who is not an administrator. See Create Dynamic Groups and Policies for Observability.

The following diagram illustrates the network configuration for autoscaling in Oracle WebLogic Server for OCI deployment.

Figure 1-2 Network Configuration

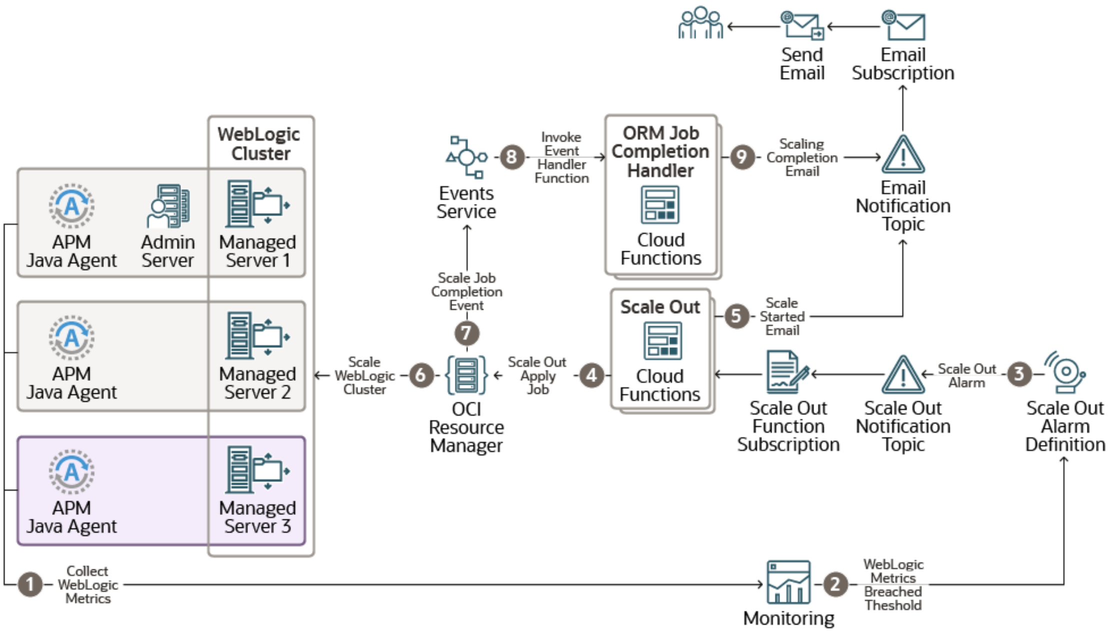


During stack creation for a domain, Oracle WebLogic Server for OCI allows you to configure metric-based autoscaling based on WebLogic Monitoring metrics.

> **Note:**
>
> You can configure autoscaling for Oracle WebLogic Server Enterprise Edition and Oracle WebLogic Suite only.

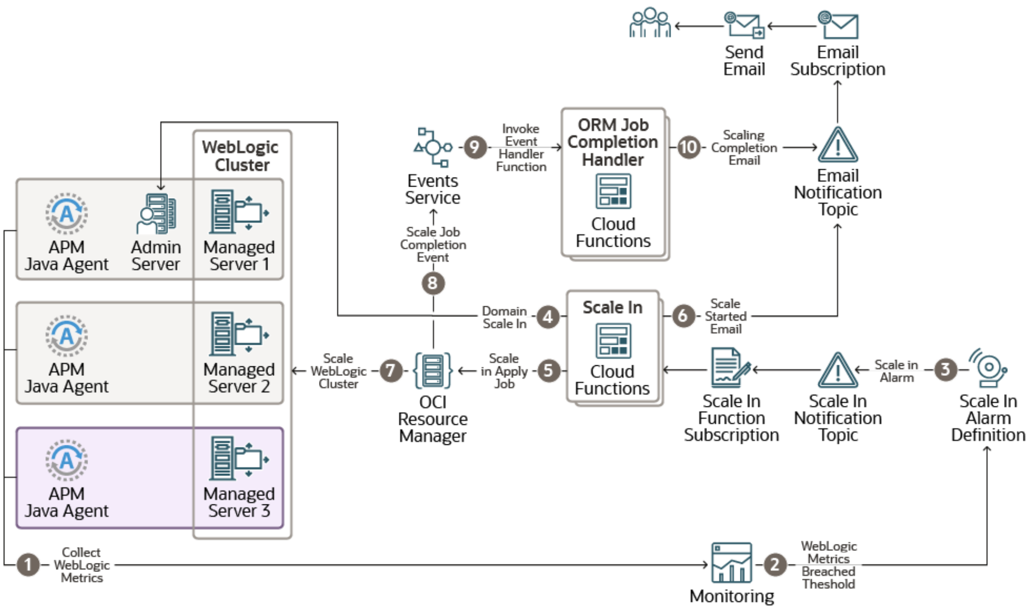
The following diagram illustrates Scale Out flow in Oracle WebLogic Server for OCI deployment.

Figure 1-3   Scale Out Flow



The following diagram illustrates Scale In flow in Oracle WebLogic Server for OCI deployment.

Figure 1-4 Scale In Flow



The resources created for autoscaling are:

- Application Performance Monitoring Agent - Application Performance Monitoring agent is used to collect WebLogic monitoring metrics and enables probes for tracing.
- WebLogic Monitoring metrics - WebLogic Monitoring metrics are collected by the Application Performance Monitoring Agent.
- Alarms Definition - Alarms Definitions are rules defined for metric-based autoscaling. They include the alarm query that evaluates the alarm using Monitoring Query Language (MQL) expression and the notification destination to send messages when the alarm is in the firing state, in addition to other alarm properties. See Managing Alarms in the Oracle Cloud Infrastructure documentation.
  The following Alarm Definitions are created during provisioning:
  - Scale Out Alarm Definition
  - Scale In Alarm Definition

  If the **OCI Policies** check box is selected during provisioning, alarms are enabled.

  If the **OCI Policies** check box is not selected during provisioning, alarms are disabled. You must create dynamic group and policies, and then enable alarms from the Oracle Cloud Infrastructure console post provisioning.

  To create dynamic group and policies, see Create Dynamic Groups and Policies for Observability and to enable alarms, see To enable an alarm in Oracle Cloud Infrastructure documentation.
- Notification topic - A topic is a communication channel for sending messages to its subscriptions. See Managing Topics and Subscriptions in the Oracle Cloud Infrastructure documentation.
  The following notification topics are created during provisioning:

  - Scale Out Topic
  - Scale In Topic
  - Email Topic
- Notification subscriptions - The following notification subscriptions are created during provisioning:
  - Scale Out Function Subscription for Scale Out Topic
  - Scale In Function Subscription for Scale In Topic
  - Email Subscription for Email Topic

  See Managing Topics and Subscriptions in the Oracle Cloud Infrastructure documentation.
- Function Application - Function Application is logical group of related functions, under which Scale In Function, Scale Out Function, and Resource Manager Job Completion Handler Function are grouped. See Overview of Functions in the Oracle Cloud Infrastructure documentation.
- Oracle Cloud Infrastructure Registry (OCIR) - OCIR is created using Terraform in root compartment and stores the docker image of the Functions. See Overview of Container Registry in the Oracle Cloud Infrastructure documentation.
  The following repositories are created during provisioning:
  - `<`*`service_prefix_name`*`>_autoscaling_function_repo/scaleout`
  - `<`*`service_prefix_name`*`>_autoscaling_function_repo/scalein`
  - `<`*`service_prefix_name`*`>_autoscaling_function_repo/orm_job_completion_handler`
- Functions - Functions trigger scale in and scale out operations when an alarm event is created upon reaching a threshold for the monitored metrics. See Overview of Functions in the Oracle Cloud Infrastructure documentation.
  The following functions are created during provisioning:
  - Scale Out Function
  - Scale In Function
  - Resource Manager Job Completion Handler Function
- Event Rule - The Event Rule defines event matching rule for Resource Manager `Job Create - End` event in the stack compartment. See Overview of Events in the Oracle Cloud Infrastructure documentation.
- Log Group and Logs - The log group contains one or more logs. The log resources created during provisioning are:
  - Function Application Log - This contains logs from all functions within the function application.
  - Event Rule Invoke Log - This contain logs for Resource Manager Job Completion event rule invocations.

  See Logging Overview in the Oracle Cloud Infrastructure documentation.

In Oracle WebLogic Server for OCI, you can apply metric-based autoscaling to WebLogic server instances. You can select a performance metric and set thresholds that this performance metric must reach to trigger an autoscaling event. The performance metrics in metric-based autoscaling are:

- CPU Load - The current CPU load of the JVM process.
- Used Heap Percent - The percentage of JVM heap that is used.
- Queue Length - The number of pending requests in the priority queue.
- Stuck Threads - The number of stuck threads in the thread pool.

See Metrics in the Oracle Cloud Infrastructure documentation.

After you select a performance metric and define the alarm minimum and maximum thresholds for the selected metric, two alarm definitions are triggered, Scale Out Alarm Definition and Scale In Alarm Definition, and a notification is sent to the Scale Out Topic or the Scale In Topic. Then, the Scale Out Function or the Scale In Function is invoked (based on the topic that receives the notification), and this function scales out or scales in the instances. Messages are sent out as emails to the notification address when the instance is scaled out or scaled in, and also after the scaling is complete with the job status.

See Notifications in the Oracle Cloud Infrastructure documentation.

The limitations with Autoscaling are:

- Autoscaling is not supported with Terraform CLI-based Oracle WebLogic Server for OCI provisioning.
- Autoscaling is not supported for cloning in this release.
- Autoscaling can be selected during provisioning only and cannot be enabled or disabled on reapply.
- Autoscaling is disabled if you select **Do Not Update Domain Configuration for Scale Out** during provisioning as for autoscaling, the domain update must be performed during provisioning.
- Autoscaling does not work as expected if bastion is not configured when WebLogic instances are in private subnet.
  For example, during provisioning, if you configure the bastion and you opt to manually update the domain configuration by not selecting **Do Not Update Domain Configuration for Scale Out**, even if the domain update fails, scale out stack apply job succeeds as it does not detect the domain update failure. So the stacks are available for autoscaling, and successive alarms trigger a scale out and thus end up scaling out to maximum node count of 30 nodes.

Before You Begin with Oracle WebLogic Server for OCI

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

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

> **Note:**
>
> Before you provision an instance, 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 a Dynamic Group
- Create Policies for the Dynamic Group
- Create Dynamic Groups and Policies for Observability
- Create Root Policies
- Create Compartment Policies
- Create Additional Policies for Observability
- Create an Encryption Key
- Create Secrets for Passwords
- Create an SSH Key

Optional Tasks

- Create a Virtual Cloud Network
- Create a Private Subnet for the Oracle WebLogic Server Nodes
- Configure the Load Balancer
- Create a Subnet for the Load Balancer
- Create a Subnet for the Bastion Node
- Create a Database
- Create a Confidential Application
- Create an Application Performance Monitoring Domain
- Create an Auth Token
- Validate Existing Network Setup

Understand Service Requirements

You require access to several services in order to use Oracle WebLogic Server for OCI.

- Identity and Access Management (IAM)
- Compute, Network, Block Storage
- Vault, Key, Secret
- Resource Manager
- Load Balancing (optional)
- Database (optional)
- Tagging (optional)

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

In Oracle Cloud Infrastructure Vault (formerly known as Key Management), 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. Each type of vault has a separate service limit in your Oracle Cloud Infrastructure tenancy. The limit for secrets spans all vaults.

See:

- Service Limits in the Oracle Cloud Infrastructure documentation
- Oracle Cloud Infrastructure Vault FAQ

Create a Compartment

Create compartments for your Oracle WebLogic Server for OCI resources, or use existing compartments.

This task is typically performed by an administrator.

When you create a domain with Oracle WebLogic Server for OCI, by default the compute instances, networks, and load balancer are all created within a single compartment. You can, however, choose to use two compartments, one compartment just for the compute instances (WebLogic Server and bastion nodes), and another compartment for all the network resources that are created for the domain (including load balancer, 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 domain using Oracle WebLogic Server for OCI.

Access to Oracle Cloud Infrastructure resources in a compartment is controlled through policies. This task is typically performed by the Oracle Cloud Infrastructure administrator.

The Oracle Cloud Infrastructure user who is not an administrator must have access to Marketplace applications, as well as management access to Resource Manager stacks and jobs, compute instances, and block storage volumes. If you want Oracle WebLogic Server for OCI to create resources for a domain like networks and load balancers, management access for these resources must also be provided.

A sample policy is shown below:

```
Allow group MyGroup to use app-catalog-listing in compartment
MyCompartment
Allow group MyGroup to manage instance-family in compartment
MyCompartment
Allow group MyGroup to manage virtual-network-family in compartment
MyCompartment
Allow group MyGroup to manage volume-family in compartment
MyCompartment
Allow group MyGroup to manage load-balancers in compartment
```

```
MyCompartment
Allow group MyGroup to manage orm-family in compartment MyCompartment
Allow group MyGroup to read metrics in compartment MyCompartment
Allow group MyGroup to inspect limits in tenancy
```

If you need to allow a user who is not an administrator to be able to create secrets with the passwords required during provisioning, ensure 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, ensure 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 you set up the appropriate policy for the network compartment. For example:

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

If you intend to create a domain that includes the Java Required Files (JRF) components, then you must be able to list databases in the compartment that contains your database, and to create network resources in the same 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
Allow group MyGroup to manage virtual-network-family in compartment
MyDBCompartment
```

If you need to allow a user who is not an administrator to be able to create file storage for instances in the same compartment as the WebLogic VCN compartment, ensure you grant manage access. For example:

```
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
```

If you need to allow a user who is not an administrator to be able to create file storage for instances in a separate compartment and mount target resources, ensure you grant manage access. 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
```

See Common Policies in the Oracle Cloud Infrastructure documentation.

Create Root Policies

You must be an Oracle Cloud Infrastructure administrator, or be granted specific root-level permissions, in order to create domains using Oracle WebLogic Server for OCI.

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

- View tenancies
- Use vault keys and secrets
- Inspect tag namespaces and apply defined tags from those namespaces to cloud resources
- Create dynamic groups
- Create root-level policies

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 manage tag-namespaces in tenancy
```

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Managing Tags and Tag Namespaces

Create Dynamic Groups and Policies

When you create a domain, by default the **OCI Policies** check box is selected and Oracle WebLogic Server for OCI 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

If you are not an administrator, ask them to create a dynamic group in Oracle Cloud Infrastructure whose members are the compute instances that Oracle WebLogic Server for OCI will create for a domain.

During stack creation for a domain, Oracle WebLogic Server for OCI 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 domain 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. Locate the compartment you plan to use for the compute instances. Then in the **OCID** column, copy the compartment's OCID.

   An OCID value looks similar to this:
   `ocid1.compartment.oc1..alongstringoflettersandnumbers123`
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 named compartment as members of this group.

   ```
   instance.compartment.id = 'WLS_Compartment_OCID'
   ```

   Provide the OCID for the compartment you copied in step 3.

   > **Note:**
   >
   > If domains can be created in more than one compartment, click **Rule Builder** to select **Any of the following** and then use separate lines to add each compartment OCID.
8. Click **Create Dynamic Group**.

See Managing Dynamic Groups in the Oracle Cloud Infrastructure documentation.

Create Policies for the Dynamic Group

If you are not an administrator, ask them to create policies that permit your dynamic group to access relevant services when you create a domain using Oracle WebLogic Server for OCI.

If you are an administrator, by default Oracle WebLogic Server for OCI creates these policies when you create a domain.

When you create a domain, compute instances in Oracle WebLogic Server for OCI need to access Oracle Cloud Infrastructure Vault secrets. If a load balancer is enabled, access to network resources is required. If you're creating a domain that includes the Java Required Files (JRF) components, depending on the database strategy you select, the instances will also need access to the Oracle Autonomous Database wallet, or the Oracle Cloud Infrastructure Database (DB System) and network resources.

The following sample policy grants the relevant vault, key, and secret permissions to a dynamic group:

```
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles
in tenancy where target.secret.id = '<OCID_of_wls_password_secret>'
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles
in tenancy where target.secret.id = '<OCID_of_db_password_secret>'
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles
in tenancy where target.secret.id = '<OCID_idcs_password_secret>'
```

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

```
Allow dynamic-group MyInstancesPrincipalGroup to manage instance-
family in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to manage volume-
attachments in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to inspect volumes in
compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to use volumes in
compartment MyCompartment where any { request.operation =
'AttachVolume', request.operation = 'DetachVolume'}
Allow dynamic-group MyInstancesPrincipalGroup to manage virtual-
network-family in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to manage load-balancers
in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to use autonomous-
transaction-processing-family in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to inspect database-
family in compartment MyCompartment
```

The following sample policy grants the OS Management service:

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

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

```
Allow dynamic-group MyInstancesPrincipalGroup to use logging-family in
compartment MyCompartment
```

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Writing Policies for Dynamic Groups

Create Dynamic Groups and Policies for Observability

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

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:

When **OCI Policies** check box is not selected when creating a domain, you can use the following script to create function's dynamic group and policies after creating the domain:

1. Create the script.
   a. Create a file with name: `create_autoscaling_policies.py`
   b. Go to Script File to Create Policies for Autoscaling Functions Post Provisioning.
   c. Copy and paste the script to the `create_autoscaling_policies.py` file.
   d. Save the file.
2. Run the following command from cloud shell:

   > **Note:**
   >
   > Ensure you have tenancy administrator privilege.

   ```
   python3 create_autoscaling_policies.py <OCID_of_the_stack>
   ```

   To run the script in a non-home region, run the following command:

   ```
   python3 create_autoscaling_policies.py <OCID_of_the_stack> --region <non-home_region>
   ```

   > **Note:**
   >
   > The script uses the stack's OCID and creates function's dynamic group and policy with permissions for the function's dynamic group. It verifies if the stack has `create_policies` set to `false`, in which case it returns an error that policies would be automatically created via stack. However, you can use the `--force` option to override and create dynamic group and policies when `create_policies == true`.

The following dynamic group and policies are created by the `create_autoscaling_policies.py` script. If you plan to not use the script, you can manually create the dynamic group and policies.

- Dynamic Group for OCI Function
- Dynamic Group Policies for OCI Function

Dynamic Group for OCI Function

When you create a domain with metric-based autoscaling a dynamic group is created for OCI functions service. This is required for the OCI function to be able to create an `Apply` job and to update the stack variables.

Group identifying all functions are in the specified compartment. All the OCI functions for the stack are tagged with the following parameters:

- `tagnamespace` is the default tag namespace of the Oracle WebLogic Server for OCI service, which is created in terraform when you create a domain.
- `tagkey` is the `wlsociserviceprefix`.
- `tagvalue` is the service prefix of Oracle WebLogic Server for OCI stack.

```
ALL {resource.type = 'fnfunc', resource.compartment.id = '<stack
compartment ocid>', tag.<tagnamespace>.<tagkey>.value='<tagvalue>'}
```

This dynamic group groups all OCI functions in the stack compartment that are assigned the `tag.<WLS-OCI service default tag namespace>.wlsociserviceprefix` tagkey with value of `service_name` of your Oracle WebLogic Server for OCI stack.

Dynamic Group Policies for OCI Function

```
Allow dynamic-group FunctionDynamicGroup to inspect tenancies in
tenancy
Allow dynamic-group FunctionDynamicGroup to inspect limits in tenancy
Allow dynamic-group FunctionDynamicGroup to manage volume-family in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage instance-family in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to use app-catalog-listing in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage virtual-network-
family in compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage load-balancers in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage orm-family in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to read metrics in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage repos in tenancy
Allow dynamic-group FunctionDynamicGroup to manage functions-family in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to use ons-topics in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to use instance-agent-command-
family in compartment id MyCompartmentOCID
```

```
Allow dynamic-group FunctionDynamicGroup to manage alarms in compartment id
MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage log-groups in compartment
id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage unified-configuration in
compartment id MyCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to inspect dynamic-groups in tenancy
Allow dynamic-group FunctionDynamicGroup to manage policies in tenancy
Allow dynamic-group FunctionDynamicGroup to use tag-namespaces in tenancy
Allow dynamic-group FunctionDynamicGroup to use apm-domain in compartment id
MyAPMDomainCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to inspect autonomous-transaction-
processing-family in compartment id MyATPDBCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to inspect database-family in
compartment id MyOCIDBCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to inspect autonomous-transaction-
processing-family in compartment id MyATPAppDBCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to inspect database-family in
compartment id MyOCIAppDBCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage mount-targets in
compartment id MyFSSCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage file-systems in
compartment id MyFSSCompartmentOCID
Allow dynamic-group FunctionDynamicGroup to manage export-sets in
compartment id MyFSSCompartmentOCID
```

Create Additional Policies for Observability

If you are not an Oracle Cloud Infrastructure administrator, you must be given management access to access relevant services when you create a domain using Oracle WebLogic Server for OCI.

If you are an administrator, by default Oracle WebLogic Server for OCI creates these policies when you create a domain.

Dynamic Group Policies

The following sample policy grants the relevant autoscaling permissions to create functions and deploy an Oracle WebLogic Server for OCI administrator instance:

```
Allow dynamic-group MyInstancesPrincipalGroup to manage functions-family in
compartment id MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to use repos in tenancy
Allow dynamic-group MyInstancesPrincipalGroup to manage ons-subscriptions in
compartment id MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to use tag-namespaces in
tenancy
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
tenancy where target.secret.id = '<OCIRAuthTokenSecretID>'
Allow dynamic-group MyInstancesPrincipalGroup to read orm-stacks in
compartment id MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to manage instance-agent-
command-execution-family in compartment id MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to manage cloudevents-rules in
compartment id MyCompartment
```

```
Allow dynamic-group MyInstancesPrincipalGroup to inspect compartments
in tenancy
Allow dynamic-group MyInstancesPrincipalGroup to use ons-topics in
compartment id MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to manage log-groups in
compartment id MyCompartment
```

Compartment Policies

Access to Oracle Cloud Infrastructure resources in a compartment is controlled through policies. This task is typically performed by the Oracle Cloud Infrastructure administrator.

If you need to allow a user who is not administrator to be able to enable exporting logs to OCI Logging Service, ensure you grant the required access. For example:

```
Allow group MyGroup to manage log-groups in compartment MyCompartment
Allow group MyGroup to use log-content in compartment MyCompartment
Allow group MyGroup to manage unified-configuration in compartment
MyCompartment
```

If you need to allow a user who is not an administrator to be able to use autoscaling, ensure you grant the required access. For Example:

```
Allow group MyGroup to manage functions-family in compartment id
MyCompartmentOCID
Allow group MyGroup to manage repos in tenancy
Allow group MyGroup to manage ons-subscriptions in compartment id
MyCompartmentOCID
Allow group MyGroup to manage alarms in compartment id
MyCompartmentOCID
Allow group MyGroup to read metrics in compartment id
MyCompartmentOCID where
target.metrics.namespace='oracle_apm_monitoring'
Allow group MyGroup to manage ons-topics in compartment id
MyCompartmentOCID
Allow group MyGroup to manage log-groups in compartment id
MyCompartmentOCID
Allow group MyGroup to use log-content in compartment id
MyCompartmentOCID
Allow group MyGroup to manage unified-configuration in compartment id
MyCompartmentOCID
Allow group MyGroup to read objectstorage-namespaces in tenancy
Allow group MyGroup to use apm-domains in compartment id
MyCompartmentOCID
Allow group MyGroup to manage cloudevents-rules in compartment id
MyCompartmentOCID
Allow group MyGroup to use cloud-shell in tenancy
```

Create an Encryption Key

An encryption key allows you to encrypt the contents of secrets required for Oracle WebLogic Server for OCI.

Oracle WebLogic Server for OCI can use one or more keys in Oracle Cloud Infrastructure Vault to decrypt the secrets for a single domain.

Use Oracle Cloud Infrastructure Vault to create a vault and encryption key, or use an existing vault and key. Oracle WebLogic Server for OCI supports keys in standard vaults and virtual private vaults. See Managing Keys in the Oracle Cloud Infrastructure documentation.

Create Secrets for Passwords

Use secrets in Oracle Cloud Infrastructure Vault to store the passwords that you need to create a domain with Oracle WebLogic Server for OCI.

You must provide secrets for these passwords:

- Administrator password for the new domain
- Administrator password for an existing database, if you are creating a domain that includes the Java Required Files (JRF) components
- Client secret for an existing confidential application, if you are creating a domain that uses Oracle Identity Cloud Service for authentication

You must use Oracle Cloud Infrastructure Vault to create your secrets. When you create a domain using Oracle WebLogic Server for OCI, you'll be asked to provide the OCID values of the secrets.

To create a secret and copy the OCID:

1. Access the Oracle Cloud Infrastructure console.
2. Navigate to the vault that contains your encryption key.
3. Click **Secrets**, and then click **Create Secret**.
4. Enter a name to identify the secret.
5. Select your encryption key.

   The key is used to encrypt the secret contents while they're imported to the vault.
6. In **Secret Contents**, enter the password you want to store in this secret.

   Ensure the password meets the criteria for which it will be used (for example, WebLogic Server administrator password).
   Passwords entered in plain-text are base64-encoded before they are sent to Oracle WebLogic Server for OCI.
7. Click **Create Secret**.
8. When the secret is created, click the name.
9. Copy the **OCID** for the secret.
10. Repeat steps 2 through 9 to create the remaining secrets you need.

See Managing Secrets in the Oracle Cloud Infrastructure documentation.

Create an SSH Key

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

A key pair consists of a public key and a corresponding private key. When you create a domain using Oracle WebLogic Server for OCI, 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 OCI can create a Virtual Cloud Network (VCN) in Oracle Cloud Infrastructure for a new Oracle WebLogic Server domain, or you can create your own VCN before creating a domain.

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

By default subnets are public. Any compute instances assigned to a private subnet cannot be directly accessed from outside of Oracle Cloud.

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

- The VCN must use DNS for hostnames.
- If you plan to use a public subnet for the bastion or load balancer, then the VCN must include an Internet Gateway.
- If you plan to create a public subnet in the VCN before creating a domain, then the VCN must include a route table that directs traffic to the Internet Gateway.
- When you create DHCP options, it is recommended to select the DNS type as **Internet and VCN Resolver**. If your network configuration demands you to select **Custom Resolver**, then add Oracle DNS Server IP `169.254.169.254` as the last DNS server entry.

If you plan to use a private subnet for the Oracle WebLogic Server compute instances, then the VCN must meet these additional requirements:

- The VCN for the bastion must have a route table that directs traffic to the Internet Gateway.
- The VCN must include a service gateway or a Network Address Translation (NAT) gateway, to provide access to other cloud services. For a service gateway, select the option **All <*Region*> Services In Oracle Services Network**.
- If you want to create the private subnet before creating a domain, then the VCN must also include a route table that directs traffic to the service gateway or the

NAT gateway. For a service gateway route rule, select the option **All *<Region>* Services In Oracle Services Network**.

- If you use a private subnet for a WebLogic domain, which requires access to the internet, then the VCN must also include a route table that directs traffic to access the NAT gateway.

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

If you use an existing VCN and existing subnets inOracle WebLogic Server for OCI, you can certify the existing network setup using helper scripts. See Validate Existing Network 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 Private Subnet for the Oracle WebLogic Server Nodes

Oracle WebLogic Server for OCI can create a subnet in Oracle Cloud Infrastructure for a new Oracle WebLogic Server domain, or you can create your own subnet before creating a domain.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a domain with Oracle WebLogic Server for OCI, the Oracle WebLogic Server compute instances are assigned to a subnet.

By default subnets span an entire region in Oracle Cloud Infrastructure.

Oracle WebLogic Server for OCI supports both regional and AD-scoped subnets.

If you assign a private subnet to the domain, the nodes cannot be directly accessed from outside of Oracle Cloud. Oracle WebLogic Server for OCI can create a bastion node on a public subnet, which you can use to administer the nodes that comprise your domain.

If you assign a private subnet that needs access to the public internet, you must include a route table that directs traffic to the NAT gateway.

If you want to use an existing subnet for the Oracle WebLogic Server nodes when creating a domain, 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) and to the administration server ports (by default, 7001 and 7002).
- The subnet must have a security list that enables inbound access to the managed server ports (by default, 7003 and 7004). If you are using a load balancer, the security list's source should be restricted to the subnets that you plan to use for the load balancer.
- If you are creating a domain that uses Oracle Identity Cloud Service to authenticate application users, the subnet must have a security list that enables inbound access to the listen port for the App Gateway in Oracle Identity Cloud Service (by default, 9999).
- If you are creating a domain with the Java Required Files (JRF) components, the subnet must have a security list that enables outbound access to the database port (1521 by default) on the database subnet.

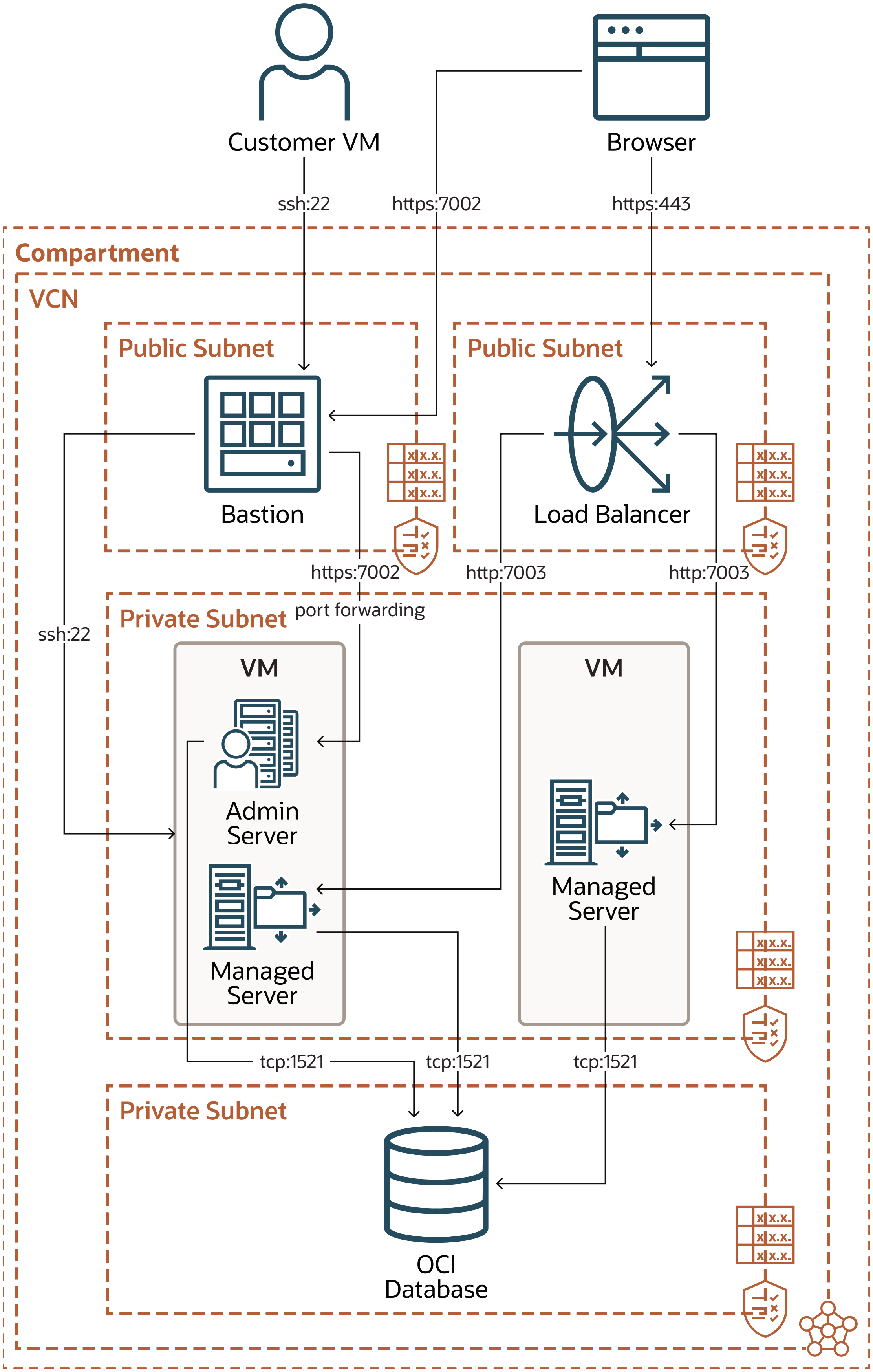
- If you are creating a domain with the JRF components, and the database is on a different VCN, then the subnet must have a security list that enables outbound access to port 53 (both TCP and UDP) on the subnet that you plan to create for DNS. The subnet must also be associated with the default DHCP option for the VCN (`Default DHCP Options for <vcn_name>`). The subnet cannot use a custom DNS resolver.

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

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

The following diagram illustrations the default destination ports.

Figure 1-5 Default Destination Ports



The following table summarizes the security list requirements for an existing subnet.

> **Note:**
>
> If you change the default ports when creating a domain, then open the related ports accordingly.

Rule Type	Source CIDR and Protocol	Default Destination Ports	Description
Stateful Ingress	Bastion network, TCP	22	SSH access
Stateful Ingress	Bastion network, TCP	7001, 7002	Admin server ports
Stateful Ingress	Your admin network, or your load balancer subnet, TCP	7003, 7004	Managed server ports
Stateful Ingress	Your load balancer subnet, TCP	9999 or custom redirect port	App Gateway in Oracle Identity Cloud Service
Stateful Ingress	Your database network, TCP	1521 or custom database port	Database for JRF-enabled domain
Stateful Ingress	DNS subnet, TCP	53	JRF-enabled domain and database is on a different VCN
Stateful Ingress	DNS subnet, UDP	53	JRF-enabled domain and database is on a different VCN
Stateful Ingress	Weblogic subnet, CIDR	9071	Used for provisioning and scaling
Stateful Ingress	Weblogic subnet, CIDR	5556	Used for accessing node manager
Stateful Ingress	Weblogic subnet, CIDR	22	Admin server ports

Network security groups are an alternative to security lists. After creating a domain 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).

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Configure the Load Balancer

Oracle WebLogic Server for OCI can create a load balancer in Oracle Cloud Infrastructure that is used to distribute traffic across the servers in your domain, or you can use your own load balancer for the domain.

If you use an existing load balancer, then you must configure the load balancer as follows:

1. Create a backend set
2. Create a listener
3. Create a routing policy

Create a backend set

Specify the load balancing and the health check policy for the backend set.

> **Note:**
>
> When you create a backend set, do not add backends for the backend set.

1. In the Oracle Cloud Infrastructure Console, from the navigation menu ☰, click **Networking**, and then click **Load Balancers**.
2. Click the name of the load balancer.
3. Under **Resources**, click **Backend Sets**.
4. Click **Create Backend Set**.
5. Enter a name for the backend set.
6. From **Traffic Distribution Policy**, select *Weighted Round Robin*.
7. From **Session Persistence**, select **Enable load balancer cookie persistence**.
8. Specify the following properties for **Health Check**:
   - Protocol: *HTTP*
   - Port: *7003*
     The default port is *7003*. You must specify the port that matches with managed server port of the backend set.
   - Interval in MS: *30000*
   - Timeout in MS: *3000*
   - Number of retries: *3*
   - Status code: *404*
   - URL Path (URI): */*
   - Response body regex: *.**
9. Click **Create Backend Set**.

Create a listener

Specify the listener properties for the load balancer.

1. In the Oracle Cloud Infrastructure Console, from the navigation menu ☰, click **Networking**, and then click **Load Balancers**.
2. Click the name of the load balancer.
3. Under **Resources**, click **Listeners**.
4. Click **Create Listener**.
5. Enter a Name for the listener.
6. Select the *HTTPS* Protocol.
7. For **Port**, enter *443*.

8. For **Certificate Resource**, select *Load Balancer Managed Certificate*, if not already selected, and then select the **Certificate Name** that you imported for the certificate resource.
9. Select the backend set that you created for the load balancer. See Create a backend set.
10. Click **Create Listener**.

Add the routing policy

Specify the listener properties for the load balancer.

1. In the Oracle Cloud Infrastructure Console, from the navigation menu ☰, click **Networking**, and then click **Load Balancers**.
2. Click the name of the load balancer.
3. Under **Resources**, click **Routing policies**.
4. Click **Routing Policy**.
5. Enter a name for the routing policy.
6. Specify the following conditions for the rule:
    - When the following conditions are met…: *If All Match*
    - Condition Type: *Path*
    - Operator: *Is*
    - URL String: */myPath*
7. Select the backend set that you created for the load balancer. See Create a backend set.
8. Click **Next**.
9. In the **Change Order** column, corresponding to the rule, click the down arrow to see a summary of the conditions and actions set in a rule.
10. Click **Reorder** to move a rule up or down in the policy order.
11. When the routing policy rules are created and in the right order, click **Create Routing Policy**.

See the folowing topics in Oracle Cloud Infrastructure documentation:

- Creating a Load Balancer
- Request Routing for Load Balancers

Create a Subnet for the Load Balancer

Oracle WebLogic Server for OCI can create subnets in Oracle Cloud Infrastructure for the load balancer that is used to access an Oracle WebLogic Server domain, or you can create your own subnets before creating a domain.

A subnet is a component of a Virtual Cloud Network (VCN). When you create a domain with a load balancer using Oracle WebLogic Server for OCI, the load balancer is assigned a subnet.

By default subnets span an entire region in Oracle Cloud Infrastructure.

To ensure high availability for a load balancer, you must assign it either one regional subnet, or two AD-scoped subnets.

If you want to use an existing subnet for the load balancer, 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 ports 80 and 443.
- The subnet must have a security list that enables outbound access to the managed server ports (by default, 7003 and 7004) on the subnet that you plan to use for Oracle WebLogic Server.

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

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

Network security groups are an alternative to security lists. After creating a domain 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).

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the Bastion Node

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

A subnet is a component of a Virtual Cloud Network (VCN). When you create a domain in Oracle WebLogic Server for OCI, you can assign the Oracle WebLogic Server compute instances to a public subnet or a private subnet. If you assign a private subnet, then the compute instances can not be directly accessed from outside of Oracle Cloud. Oracle WebLogic Server for OCI can create a bastion compute instance on a public subnet, and from this bastion you can administer the Oracle WebLogic Server compute instances.

> **Note:**
>
> Configuring a bastion is optional.
>
> If you do not configure a bastion, no status is returned for provisioning. You must check the status of provisioning by connecting to each compute instance and confirm that the `/u01/provStartMarker` file exists with details found in the file `/u01/logs/provisioning.log` file. See Configure a Bastion.

By default subnets span an entire region in Oracle Cloud Infrastructure.

Oracle WebLogic Server for OCI supports both regional and AD-scoped subnets.

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

- The subnet must use DNS for hostnames.
- The subnet must be public.
- The subnet's route table must have an Internet Gateway rule
- 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 Oracle WebLogic Server.

If you use an existing subnet, the subnet is created in the subnet compartment that is different than the VCN compartment.

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

Network security groups are an alternative to security lists. After creating a domain 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).

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Database

Before creating an Oracle WebLogic Server domain that includes the Java Required Files (JRF) components, you must create a 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.

A JRF-enabled domain supports the Oracle Application Development Framework (ADF). When you create a domain with Oracle WebLogic Server for OCI and associate it with an existing database, Oracle WebLogic Server for OCI 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 OCI also provides a tool to delete the JRF schemas for a specific domain from the database.

Choose one of these database options:

- Oracle Autonomous Database
  - Create an autonomous database using either the dedicated or shared infrastructure option. You can also create a Free-Tier autonomous database.
  - See Creating an Autonomous Database in the Oracle Cloud Infrastructure documentation.

> **Note:**
>
> Oracle Autonomous Database provides a private endpoint configuration which can be created in subnet within a VCN, by using listen port `1522`. However, the configuration is supported only for existing VCNs or existing VCNs with subnets topology. For Oracle Autonomous Database, VCN peering is not suported out-of-box with Terraform scripts. If you want to use private endpoint for Oracle Autonomous Database or Oracle Autonomous Database is on a different VCN, then manually peer the VCNs before provisioning.

- Oracle Cloud Infrastructure Database
  - Create bare metal, virtual machine (VM), and Exadata DB systems. For a 1-node VM DB system, note that you can use the fast provisioning option to create the database. Oracle WebLogic Server for OCI supports using Logical Volume Manager as the storage management software for a 1-node VM DB system.
  - For Oracle WebLogic Server 12.2.1.4.0, you can also specify a database connection string. This database connection string can be used only with existing VCN. To know the database connection string details, see Database Connect String for Database Version and Type in Configure Database Parameters.
  - 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 - by default, Oracle WebLogic Server for OCI creates a security list on the database's VCN that allows the WebLogic Server subnet to access the database. Alternatively, you can manually update the security lists in the database's VCN, or update the network security group that is assigned to the database. If you use database connection string, security list is not created to access the database. You must ensure that the ports are open to access the database.

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

- Administrator credentials
- Pluggable database (PDB) name (only for Oracle Cloud Infrastructure Database running Oracle Database 12c or later)

If your Oracle Cloud Infrastructure Database and domain are in different VCNs, then Oracle WebLogic Server for OCI configures local peering between the two VCNs. To support VCN peering, you must meet the following additional requirements:

- Multiple domains cannot use the same database.
- The CIDRs for the VCNs must not overlap. For example, you cannot create a domain in VCN `10.0.0.0/16` that uses a database in VCN `10.0.0.1/24`.
- The database subnet must have a security list that enables outbound access to port 53 (both TCP and UDP) on the subnet that you plan to create for DNS.
- The database subnet must be associated with the default DHCP option for the VCN (`Default DHCP Options for <vcn_name>`). The subnet cannot use a custom DNS resolver.

If you use Oracle Cloud Infrastructure Database with connect string, you must peer the VCNs manually before creating the stack if your database VCN is on a different VCN than the WebLogic Server VCN.

The following table summarizes the security list requirements for an existing subnet that will use local VCN peering to communicate with the domain.

Rule Type	CIDR and Protocol	Destination Ports	Description
Stateful Ingress	WebLogic Server subnet, TCP	1521 or custom database port	Database access
Stateful Ingress	DNS subnet, TCP	53	Access to custom DNS resolver
Stateful Ingress	DNS subnet, UDP	53	Access to custom DNS resolver

Oracle WebLogic Server for OCI 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)

> **Note:**
>
> From release 21.4.3 (December 9, 2021) onwards, you cannot provision a domain in Oracle Oracle WebLogic Server for OCI for Oracle WebLogic server versions 11g (10.3.6.0) and 12c (12.2.1.3) from the Marketplace. However, you can provision a domain for Oracle WebLogic server version 12c (12.2.1.3) using the terraform scripts See Create a Domain for Oracle WebLogic Server 12.2.1.3.0 Using Terraform. The WebLogic binaries for 12.2.1.3 are also available in the public images.

Create a Confidential Application

Before creating an Oracle WebLogic Server 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 OCI provisions an App Gateway and other security components in Oracle Identity Cloud Service. In order for Oracle WebLogic Server for OCI 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. You must use Oracle Cloud Infrastructure Vault to create a secret to store the client secret. You will asked to provide the OCID of the secret in the vault. See Create Secrets for Passwords.

Create a confidential application for Oracle WebLogic Server for OCI, 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*.

Create an Application Performance Monitoring Domain

Create an Application Performance Monitoring domain or use an existing Application Performance Monitoring domain.

When you create a domain with Oracle WebLogic Server for OCI, you can enable Application Performance Monitoring integration and provide the Application Performance Monitoring domain details. So, during provisioning, the Application Performance Monitoring Java agent is installed on each of the WebLogic compute instance. This agent records metrics and spans and sends them to the specified Application Performance Monitoring domain.

Metric-based autoscaling depends on the Application Performance Monitoring integration feature to be enabled during provisioning.

See Create an APM Domain in the Oracle Cloud Infrastructure documentation.

> **Note:**
>
> Application Performance Monitoring always free domain is not supported for autoscaling.

Create an Auth Token

In order for Oracle Cloud Infrastructure Registry to deploy autoscaling OCI Functions, you must provide an auth token.

Oracle WebLogic Server for OCI 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 to 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.

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 OCI. 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 private subnets.
- Validates if the database port is accessible from WebLogic Server subnets.
- Validates if port number of the administration server is accessible to the SSH and T3 ports from managed servers.
- Validates if internet gateway is created for public bastion and WebLogic Server subnets. This validation is optional and can be ignored for private or FastConnect network.
- 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 port 7003 in WebLogic Server subnet is open for access to the CIDR of the load balancer subnet.
- Checks if port 443 in load balancer subnet is open for access to `0.0.0.0/0` IP address.
- Checks if ports 111 (both TCP and UDP), 2048-2050 (TCP) and 2048 (UDP) in file system storage subnets are open for access.
- 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.

You must install the following tools before you run the validation scripts:

- OCI CLI 2.14.1 or later

See Install CLI.

- jq 1.5 or later

  See Download jq.

- Bash 4.0 or later

  See Upgrade Bash Version.

As these tools are installed in the Cloud Shell by default, it is recommended to run the validation scripts from the Cloud Shell. The scripts can be copied to Cloud Shell or any Linux terminal running bash 4 and above.

Validation scripts can be run prior to provisioning or post-provisioning.

Validation can be done post-provisioning if the network configuration changes causes the following issues:

- Scale out fails on reapply. This occurs when ports 22 and 9071 are not open to access Weblogic Server subnet CIDR in Weblogic Server subnet.
- Database connectivity fails. This occurs when the database port is not open to access Weblogic Server subnet CIDR in database subnet, or NAT gateway or service gateway for the WebLogic Server VCN is removed resulting in access failure to the Oracle Autonomous Database.

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 `validate.sh`. See Script File to Validate Network Setup to create the `validate.sh` file.

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

   ```
   chmod +x validate.sh
   ```

2. Based on the domain type, run the following commands using `validate.sh` script prior to provisioning:

   - Basic domain

     ```
     ./validate.sh -w <WLS_Subnet_OCID>
     ```

   - JRF-enabled domain

     ```
     ./validate.sh -w <WLS_Subnet_OCID> -d <DB_Subnet_OCID>
     ```

     If you configure a load balancer, the load balancer subnet must have a security list that enables access to port 443, and the WebLogic Server subnet must have a security list that allows load balancer subnet to access port 7003

     ```
     ./validate.sh -w <WLS_Subnet_OCID> -l <LB_Subnet_OCID
     ```

     If you configure file system storage, the file system storage subnet must have a security list that enables access to the ports 111 (both TCP and UDP), 2048-2050 (TCP) and 2048 (UDP).

     ```
     ./validate.sh -w <WLS_Subnet_OCID> -l <LB_Subnet_OCID
     ```

     If you are using an Application Datasource that is in a diffferent subnet than the database subnet, specify the `Application DB Subnet OCID`

```
./validate.sh -w <WLS_Subnet_OCID> -a <Application_DB_Subnet_OCID>
```

If you are using an Autonomous Database with private endpoint, specify the `Autonomous DB OCID`.

```
./validate.sh -w <WLS_Subnet_OCID> -t <Autonomous_DB_OCID>
```

- Basic domain in a private subnet

  ```
  ./validate.sh -b <Bastion_Subnet_OCID>
  ```

  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.

  ```
  ./validate.sh -b <Bastion_Subnet_OCID> -i <Bastion_Host_IP_CIDR>
  ```

  The bastion host IP CIDR must have `/32` suffix, else the following error is displayed:

  ```
  Bastion host IP CIDR is not valid: [1.1.1.1]
  ```

```
validate.sh

example_user@cloudshell:~ (us-phoenix-1)$ ./validate.sh -w
<WLS_Subnet_OCID> -d <DB_Subnet_OCID> /
-l <LB_Subnet_OCID> -f <FSS_Subnet_OCID> -a <AppDB_Subnet_OCID> -t
<ATPDB_OCID> -b <Bastion_Subnet_OCID> -i <Bastion_Host_IP_CIDR>
ERROR: Route Rule of private WLS subnet [<WLS_Subnet_OCID>] does not
use 'ALL Services in Oracle services network' destination
ERROR: Port 22 is not open for access by WLS Subnet CIDR [10.0.0.0/24]
in WLS Subnet [<WLS_Subnet_OCID>]
ERROR: Port 9071 is not open for access by WLS Subnet CIDR
[10.0.0.0/24] in WLS Subnet [<WLS_Subnet_OCID>]
ERROR: DB port 1521 is not open for access by WLS Subnet CIDR
[10.0.0.0/24] in DB Subnet [<DB_Subnet_OCID>]
ERROR: LB port 7003 is not open for access by LB Subnet CIDR
[10.0.5.0/24] in WLS Subnet [<WLS_Subnet_OCID>]
ERROR: Port [443] is not open for 0.0.0.0/0 in LB Subnet CIDR
[<LoadBalancer_Subnet_OCID>]
ERROR: TCP Port [111] is not open in FSS Subnet for VCN CIDR
ERROR: TCP Port [2048] is not open in FSS Subnet for VCN CIDR
ERROR: TCP Port [2049] is not open in FSS Subnet for VCN CIDR
ERROR: TCP Port [2050] is not open in FSS Subnet for VCN CIDR
ERROR: UDP Port [111] is not open in FSS Subnet for VCN CIDR
ERROR: UDP Port [2048] is not open in FSS Subnet for VCN CIDR
ERROR: DB port 1521 is not open for access by WLS Subnet CIDR
[10.0.0.0/24] in Application DB Subnet [<AppDB_Subnet_OCID>]
ERROR: DB port 1522 is not open for access by WLS Subnet CIDR
[10.0.0.0/24] in Autonomous DB Subnet [<ATPDB_Subnet_OCID>]
ERROR: SSH port 22 is not open for access by [0.0.0.0/0] in Bastion
Subnet [<Bastion_Subnet_OCID>]
WARNING: SSH port 22 is not open for access by Bastion Subnet CIDR
[1.1.1.1/32] in private WLS Subnet [<WLS_Subnet_OCID>]
```

2

Create a Stack

Learn how to create an Oracle WebLogic Server stack with Oracle WebLogic Server for OCI.

Topics:

- About Creating a Domain
- Create a Basic Domain
- Create a Basic Domain in a Private Subnet
- Create a JRF-Enabled Domain
- View the Cloud Resources for a Stack
- About the Resources in a Stack

About Creating a Domain

Learn about the options you have when creating a domain with Oracle WebLogic Server for OCI.

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

- Billing Type

  With Universal Credits (also called UCM), you are billed for the cost of the Oracle WebLogic Server license (based on OCPUs per hour) in addition to the cost of the compute resources. This option is not available for Oracle WebLogic Server Standard Edition.

  With Bring Your Own License (BYOL), you reuse your existing on-premises Oracle WebLogic Server licenses in Oracle Cloud. You are billed only for the cost of the compute resources.

- Domain Type and Database

  A basic domain does not require an existing database. See Create a Basic Domain.

  A JRF-enabled domain requires access to an existing Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System). A JRF-enabled domain includes the Java Required Files (JRF) components, and the database is used to contain the JRF schema.

  See Create a JRF-Enabled Domain and Create a Database.

- Virtual Cloud Network (VCN)

  Oracle WebLogic Server for OCI can create a VCN for you when you create a domain, or you can create a VCN before you create the domain. If you create a new VCN, you must specify a contiguous CIDR block of your choice.

  If you create a JRF-enabled domain and select an Oracle Cloud Infrastructure Database (DB System) in a different VCN, then Oracle WebLogic Server for OCI configures local peering between the two VCNs.

- Subnet

  Oracle WebLogic Server for OCI can create a new subnet for the WebLogic Server compute instances, or you can specify a subnet that you have already created. You must specify a CIDR if you create a new subnet.

  If you create a new VCN, you can only create a new regional subnet that spans the entire region.

- Network Access

  The subnet for the WebLogic domain can be public or private. If the subnet is private, the nodes cannot be accessed directly from outside of Oracle Cloud. When you create a domain on a private subnet, you can specify a public subnet for the bastion host. Oracle WebLogic Server for OCI creates this compute instance to enable you to administer the WebLogic nodes.

  If you already have an existing bastion to provide public access to the compute instances, or if you already have a VPN connection to your on-premise network, then you can delete the bastion created by Oracle WebLogic Server for OCI.

  See Create a Basic Domain in a Private Subnet.

  > **Note:**
  >
  > - Configuring a bastion is optional.
  >
  >   If you do not configure a bastion, no status is returned for provisioning. See Configure a Bastion.
  >
  > - It is recommended to not configure a bastion, that is deselect the **Provision Bastion Node on Public Subnet** option, only in network with fast connect setup

- Load Balancer

  When you create a domain, Oracle WebLogic Server for OCI can also create a load balancer to distribute application traffic to the WebLogic cluster. A 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.

  If you use a regional subnet for the WebLogic Server compute instances, use a regional subnet for the load balancer. The regional subnet is shared between both load balancer nodes.

  By default, the load balancer is public. You can also provision a public load balancer with a reserved public IP. If you create a domain in a private subnet, then you can provision a public or private load balancer. A private load balancer does not have a public IP address and cannot be accessed from outside of Oracle Cloud.

  Oracle WebLogic Server for OCI configures the load balancer to use Secure Socket Layer (SSL). A demonstration self-signed certificate is attached to the HTTPS listener. Oracle recommends you add your own SSL certificate to the load balancer after creating the domain. All traffic between the load balancer and compute instances uses HTTP.

- Oracle Cloud Infrastructure Root Policies and Dynamic Group

  When you create a domain, by default Oracle WebLogic Server for OCI creates a dynamic group and one or more root-level (tenancy) policies that allow the compute instances in the domain to access:

  - Keys and secrets in Oracle Cloud Infrastructure Vault
  - Load balancer resources
  - The database wallet if you're using Oracle Autonomous Database to contain the required infrastructure schemas for a JRF-enabled domain
  - The database network resources if you're using Oracle Cloud Infrastructure Database (DB System) to contain the required infrastructure schemas for a JRF-enabled domain

- Authentication

  By default, the domain is configured to use the local WebLogic Server identity store to maintain users, groups, and roles. Alternatively, a domain running WebLogic Server 12c can use Oracle Identity Cloud Service to authenticate users.

  In order to use Oracle Identity Cloud Service, you must create a domain that includes a load balancer.

- Security List for DB System

  When you create a JRF-enabled domain and use Oracle Cloud Infrastructure Database (DB System) to contain the JRF components, by default Oracle WebLogic Server for OCI creates a security list on the database's VCN that allows the WebLogic Server subnet to access the database.

Create a Basic Domain

Use Oracle WebLogic Server for OCI to create a stack that includes a basic Oracle WebLogic Server 12c domain, one or more WebLogic Server compute instances, network resources, and an optional load balancer.

Video

Launch a new stack from Marketplace. For a basic domain, you specify a public subnet for WebLogic Server (either a regional or availability domain-specific), and you do not specify a database.

Before you create a domain, you must first perform the following tasks:

- Create a compartment. See Create a Compartment.
- Create an SSH key. See Create an SSH Key.
- Create an encryption key to use for secrets. See Create an Encryption Key.
- Create secrets for the passwords you want to use for the domain. You will need to select the compartment where you have the secret and the secret that contains the password. See Create Secrets for PasswordsSee Create Secrets for Passwords.
- Create a confidential application in Oracle Identity Cloud Service if you want to use Oracle Identity Cloud Service for authentication in the domain. You will need the client ID and client secret for this confidential application. See Create a Confidential Application. You will also need to create a secret for the client secret and copy the OCID. See Create Secrets for Passwords.

- Create a FastConnect or a VPN connection if you want to use your own bastion host to administer your Compute instances. See VPN Connect or FastConnect in the Oracle Cloud Infrastructure documentation.

Oracle WebLogic Server for OCI can create the virtual cloud network (VCN) and subnets for your new domain. If you want to use an existing VCN or existing subnets for the domain, then they must meet certain requirements. See:

- Create a Virtual Cloud Network
- Create a Private Subnet for the Oracle WebLogic Server Nodes
- Create a Subnet for the Load Balancer (if you want to create a load balancer)

Tutorial

Topics:

- Launch a Stack
- Specify Stack Information
- Configure WebLogic Instance Parameters
- Configure Advanced Parameters for a Domain
- Configure Network Parameters
- Configure a WebLogic Console Port
- Configure a Load Balancer
- Configure File Storage
- Create OCI Policies
- Configure WebLogic Authentication
- Configure Database Parameters
- Configure a Data Source for an Application Database
- Set Local VCN Peering for an Application Database
- Configure Observability
- Configure Autoscaling
- Configure Tags
- Create the Domain Stack
- Use Your New Domain

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 ☰, select **Marketplace**, and then click **All Applications**.
3. Set the filter type to **Stack**.

4. Select an application that matches the edition of Oracle WebLogic Server that you want to provision, and also uses the type of billing you want (Universal Credits or Bring Your Own License).
   - **Oracle WebLogic Server Standard Edition BYOL**
   - **Oracle WebLogic Server Enterprise Edition BYOL**
   - **Oracle WebLogic Server Enterprise Edition UCM**
   - **Oracle WebLogic Suite BYOL**
   - **Oracle WebLogic Suite UCM**

   > **Note:**
   >
   > If you set the filter type to **Image**, follow the steps in Create an Instance Using the Marketplace in *Images for Oracle WebLogic Server for OCI*.

5. Select a version of Oracle WebLogic Server.

   The 14.1.1.0.0 patch level is the default.
   If multiple builds are available for the same patch level (`.01`, `.02`, `.03`, and so on), choose the latest build.

6. Select the compartment in which to create the stack.

   By default the stack compartment is used to contain the domain compute instances and network resources. If later on you specify a network compartment on the Configure Variables page of the Create Stack wizard, then only the compute instances are created in the stack compartment that you select here.

7. Select the **Oracle Standard Terms and Restrictions** check box, and then click **Launch Stack**.

   The Create Stack wizard is displayed.

Specify 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 Instance Parameters

Specify the parameters needed to configure the WebLogic instance domain.

1. In the WebLogic Server Instance section, enter the resource name prefix.

   The maximum character length is 16.

   This prefix is used by all the created resources.
2. Select the WebLogic Server shape for the compute instances.

The following shapes are supported:

- Standard: `VM.Standard2.x`, `VM.Standard.E2.x`, `BM.Standard2.x`, `BM.Standard.E2.x`, `BM.Standard3.64`
- Flexible: `VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`
- Optimized: `BM.Optimized3`, `VM.Optimized3.Flex`

> **Note:**
>
> In regional subnets, select the WebLogic Server shape that has sufficient service limits for an availability domain, else the provisioning fails.

3. For the flexible shapes (`VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`, `VM.Optimized3.Flex`), select the OCPU count for compute instances.

> **Note:**
>
> You can specify the OCPU count only for the flexible shapes; the maximum number of OCPUs that you can specify for `VM.Standard.E3.Flex` and `VM.Standard.E4.Flex` is 64 OCPUs, for `VM.Standard3.Flex` is 32 OCPUs, and for `VM.Optimized3.Flex` is 18 OCPUs. The memory, network bandwidth, and number of Virtual Network Interface Cards (VNICs) scale proportionately with the number of OCPUs.

4. Enter the SSH public key, by either uploading the SSH key file or pasting the contents of your SSH public key file.
5. Select the number of managed servers you want to create. For 12c and 14c versions and all the editions, you can specify up to `8` nodes, which can be scaled out to 30 when you edit the domain.

   The managed servers will be members of a cluster, unless you selected WebLogic Server Standard Edition.
6. Enter a user name for the WebLogic Server administrator.
7. Select the compartment where you have the WebLogic Server administration secret and then select the secret that contains the administration password. To create secrets, see Create Secrets for Passwords.
8. Select the JDK version for Oracle WebLogic Server. The default JDK version is 8.

> **Note:**
>
> You can select the JDK version only for Oracle WebLogic Server 14.1.1.0.

Configure Advanced Parameters for a Domain

You can optionally specify additional parameters by selecting **WLS Instance Advanced Configuration** on the Configure Variables page of the Create Stack wizard.

Select **WLS Instance Advanced Configuration** if you want to change the default port numbers or remove the sample application, and specify the WebLogic startup arguments. You can use the server startup arguments to provide arguments to the Java Virtual Machine for WebLogic Server instances.

- Cluster-related parameters are not applicable if you selected WebLogic Server Standard Edition.
- The port numbers 9071-9074 are reserved for internal domain communication.
- *Optional*: Specify the **WebLogic Server Startup Arguments** to scale out managed servers. When the servers are scaled out, any changes to the server startup arguments applies to the added nodes only. For example, `-Xms1024m -Xmx1024m`.

Configure Network Parameters

Define the Virtual Cloud Network (VCN) and subnet configuration for the basic domain. For this basic domain, the domain instance attaches to a public subnet.

1. Select a Virtual Cloud Network (VCN) strategy:
   - Select **Use Existing VCN**, and then select the name of the existing VCN.
   - Select **Create New VCN**, and then enter a name and CIDR for the new VCN.
2. In the WebLogic Server Network section of the Configure Variables page, select the **Network Compartment** in which to create the network resources for this domain.

   If you don't specify a network compartment, then all the network resources and the domain 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.
3. Select one of the following subnet strategies:
   - Select **Use Existing Subnet**.
   - Select **Create New Subnet**.

   > **Note:**
   >
   > If you're creating a new VCN, you can only create a new regional subnet.
4. If you use an existing VCN and subnet, validate the network and then enter `YES` in the **Validated Existing Network** field. To validate a network, see Validate Existing Network Setup.
5. Select the **Subnet Compartment** to use for the existing subnet.

   The subnet compartment is different than the VCN compartment. The subnets for the WebLogic Server nodes, load balancer and the bastion node use this same subnet compartment.

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

6. Keep the default **Use Public Subnet** selection.
7. For the WebLogic Server subnet, specify one of the following:
   - If you want to use an existing regional subnet, then choose the name of an existing regional subnet from the list of regional and availability domain-specific subnets.
   - If you are creating a new regional subnet, specify a CIDR for the new subnet.
8. *Optional*: If you are using a regional subnet, then from the WebLogic Admin Sever Availability Domain, select the availability domain in which you want to create the WebLogic administration server compute instance. If you do not select an availability domain, then by default, the compute instance is created in availability domain 1.
9. If your want to use a bastion compute instance with a reserved public IP, then select **Assign Reserved Public IP to Bastion Instance**.

Configure a WebLogic Console Port

If you are creating a new VCN with public subnets, then you have the option to disclose the Oracle WebLogic Administration Server Console port in a public subnet.

> **WARNING:**
>
> Oracle does not recommend that you open the WebLogic Administrative port to the internet when the WebLogic Server is in a public subnet. If your WebLogic Server is in a public subnet and you need to access WebLogic Administrative Console, then you can restrict the IP addresses by creating a security rule and open WebLogic Server Administrative port only to a CIDR block. Oracle WebLogic Server for OCI can configure this security rule for you during provisioning. That is, if you select a public subnet, ensure to limit the CIDR range to access WebLogic administration console ports, the defaults ports are `7001` and `7002` for `http` and `https` respectively.

1. Select **Enable Access to Administration Console**.
2. Specify the CIDR to create a security list to allow access to the WebLogic administration console port to the source CIDR range.

Configure a Load Balancer

You have the option to create a load balancer to distribute application traffic to the WebLogic Managed Servers. You can also use an existing load balancer for an existing VCN and an existing subnet, to distribute application traffic to the WebLogic Managed Servers.

> **Note:**
>
> If you enable autoscaling for WebLogic instances, you must configure a load balancer in Oracle Cloud Infrastructure when you create a stack, else the stack provisioning fails with a validation error.

To create a load balancer:

1. Select **Add Load Balancer**, if not already selected.

   This option is selected by default.

2. Configure the load balancer network.

   - If you chose to use an existing regional subnet for WebLogic Server, then select an existing regional subnet from the list of regional and availability domain-specific subnets. A load balancer can have only one regional subnet, which is shared between both nodes.

     > **Note:**
     >
     > This option is not applicable if you use an existing load balancer.

   - If you chose to create a regional subnet for WebLogic Server, then specify a CIDR for the new load balancer subnet.

3. Configure the load balancer:

   - If you chose to create a new Virtual Cloud Network (VCN) or use an existing VCN and a new subnet:

     a. Select **Create New Load Balancer**.

   - If you chose to use an existing VCN and an existing subnet, you can do one of the following:

     a. Select **Create Load Balancer**.

     b. Select **Use Existing Load Balancer**.

        > **Note:**
        >
        > The existing load balancer should be in the same compartment as the stack compartment, and should be in the same VCN as the existing VCN chosen for the stack.

        i. Specify the OCID for the existing load balancer.

        ii. Enter the name of the backend set for the existing load balancer that has a routing policy associated with the backend set. The backend set should not have any backends.
        See Configure the Load Balancer.

- If you create a new load balancer, select **Load Balancer with Reserved Public IP**, if you want to use a public load balancer with a reserved public IP. Then, specify the OCID of the public IP for the load balancer
- If you create a new load balancer, select a minimum and maximum flexible load balancer shape.
  By default, the minimum bandwidth size is set to 10Mbps and maximum to 400Mbps.

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

Configure File Storage

When you create an Oracle WebLogic Server for OCI domain, you can add a file storage.

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

To create a file storage:

1. Select **Add File System Storage**.
2. Configure the file storage:
   - If you chose to create a Virtual Cloud Network (VCN):
     a. *Optional:* Select the availability domain in which you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and a new subnet:
     a. *Optional:* Select the availability domain where you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and an existing subnet:
     – If you *do not* want to use an existing mount target or an existing file system:
       a. *Optional:* Select the availability domain where you want to create the file system and mount target.
       b. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
     – If you select **Existing Mount Target**:
       a. Select the compartment where you have the existing mount target. The mount target must reside within the subnet in the selected VCN.
       b. Specify the OCID of the existing mount target ID.
     – If you select **Existing File System**:

    a. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
    b. Select the compartment where you have the existing file system.
    c. Specify the OCID of the existing file system.

  - If you select both **Existing Mount Target** and **Existing File System**:
    a. Select the compartment where you have the existing mount target.
    b. Specify the OCID of the existing mount target ID.
    c. Select the compartment where you have the existing file system.
    d. Specify the OCID of the existing file system. The existing file system must be in the same availability domain as the existing mount target.

Create OCI Policies

When you create a basic domain, by default the **OCI Policies** check box is selected and Oracle WebLogic Server for OCI 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 domain.

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.

Configure WebLogic Authentication

You have the option to use Oracle Identity Cloud Service to authenticate application users for your domain.

This configuration is only available if the domain meets these requirements:

- Running WebLogic Server 12c
- Includes a load balancer

To use Oracle Identity Cloud Service for authentication:

1. Select **Enable Authentication Using Identity Cloud Service**.
2. Enter your Oracle Identity Cloud Service (IDCS) tenant name, which is also referred to as the instance ID.

   This ID is typically found in the URL that you use to access Oracle Identity Cloud Service, and has the format `idcs-<GUID>`.
3. Enter the client ID of an existing confidential application in this Oracle Identity Cloud Service instance.
4. Select the compartment where you have the IDCS secret and then select the secret that contains the confidential application password in IDCS. To create secrets, see Create Secrets for Passwords.
5. If necessary, you can override the default domain name and port that you use to access Oracle Identity Cloud Service, or the default port that is used for the App Gateway software appliance.

Configure Database Parameters

A basic WebLogic Server 12c domain does not require a database.

A database is required only if you want to create a domain that includes the Java Required Files (JRF) components. Do not select the **Provision with JRF** checkbox if you're not creating a JRF-enabled domain.

To create a domain that uses a database for JRF components, see Create a JRF-Enabled Domain.

Configure a Data Source for an Application Database

When you create an Oracle WebLogic Server for OCI domain, you can configure the application database to create a data source configuration that enables you to connect to Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System).

The database that you connect to is used to contain the schemas for the application database.

You can configure the application database only for Oracle WebLogic Server Enterprise Edition and Oracle WebLogic Suite.

> **Note:**
>
> You cannot configure the data source for an application database using database connect string.

On the Configure Variables page, select the **Configure Application Datasource** checkbox to display the Database options. Then in the Application Database section, select the **Application Database Strategy** for your application database and configure the database parameters.

- If using **Oracle Autonomous Database**, select or enter the following:
  - The compartment in which you've created the application database.
  - The autonomous database where you want to create the schemas for the application database.
  - The name of an autonomous database user to configure the application database.
  - Select the compartment where you have the application autonomous database secret and then select the secret that contains the application autonomous database user password in the autonomous database. To create secrets, see Create Secrets for Passwords.
  - The service level that the domain should use to connect to the application database for the selected autonomous database.
- If using **Database System**, select or enter the following:
  - The compartment in which you've created the application database.
  - The DB system to use for this application database.

  - The compartment in which the application database's VCN is found.
  - The VCN on which you've created the application database. If this VCN is different than the WebLogic Server VCN, they cannot have overlapping CIDRs. For example, you cannot create a domain on VCN `10.0.0.0/16` that uses a database on VCN `10.0.0.1/24`.
  - The database home within the selected application database system.
  - The version of the selected database home.
  - The database within the selected DB system where you want to create the schemas for the application database.
  - The Pluggable database (PDB) name, only if the selected application database is running Oracle Database 12c or later.
  - The name of a database user to configure the application database.
  - Select the compartment where you have the database secret and then select the secret that contains the application database user password. To create secrets, see Create Secrets for Passwords.
  - The application database listen port (1521 by default)
- If using **Database System**, then Oracle WebLogic Server for OCI creates a security list in the VCN on which you've created the application database. This security list allows the WebLogic Server subnet to access the application database port. If this step isn't required or you don't have the correct permissions to modify the database network, clear the **Create Application Database Security List** check box.

Set Local VCN Peering for an Application Database

If you selected the option to create a new VCN or selected the option to use an existing VCN with a new subnet for the WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database, you can either disable the local VCN peering or configure the local VCN peering for the Application Database.

Ensure that the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database are peered before creating the stack for the Oracle WebLogic Server for OCI domain. See Local VCN Peering to peer the VCNs manually. In this case, the stack is provisioned based on the database private IP address.

> **Note:**
>
> This is not applicable if you use an existing VCN and existing subnet, as the WebLogic server and database must be in the same VCN.

If you choose to create a virtual cloud network for an Oracle WebLogic Server domain, use Oracle WebLogic Server for OCI to create a Local Peering Gateway, else create a network with VCN peering and then use this existing network to provision the domain.

> **Note:**
>
> You must provision a bastion to use VCN peering, as there are SSH requirements which require a bastion.

If the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database system have not been peered, you can use Oracle WebLogic Server for OCI to update the two VCNs so that they can communicate.

Oracle WebLogic Server for OCI creates a public subnet in each VCN, and then creates a compute instance in each subnet. These compute instances run software to forward DNS requests across the VCNs.

You cannot use existing subnets for the DNS Forwarder compute instances.

1. Specify a CIDR for the new subnet in the WebLogic Server VCN.
2. Specify a CIDR for the new subnet in the application database VCN.
3. Select a shape for the new DNS Forwarder compute instance in each VCN.

Configure Observability

Oracle WebLogic Server for OCI can optionally export logs to OCI Logging Service, and provide visibility into the performance of applications using the Oracle Cloud Infrastructure Application Performance Monitoring (APM) service.

Select **Configure Observability** to enable logging and monitoring service integration for your WebLogic instances.

- Select **Enable exporting logs to OCI Logging Service** to enable logging for the WebLogic instances.
- Select **Enable Application Performance Monitoring** to export WebLogic metrics using Application Performance Monitoring (APM) Java Agent and create dashboards with WebLogic specific metrics. This is required for metric-based autoscaling of instances.
  Specify the OCID of the Application Performance Monitoring domain and the private data key for your existing Application Performance Monitoring domain.

When you enable Application Performance Monitoring, you can use autoscaling to scale out or scale in instances. See Configure Autoscaling.

> **Note:**
>
> If you enable autoscaling, you cannot use the always free Application Performance Monitoring domain.

Configure Autoscaling

When you create an Oracle WebLogic Server for OCI domain, you can enable autoscaling for WebLogic instances.

This configuration is only available if **Application Performance Monitoring** is enabled for the WebLogic instance. See Configure Observability. Application Performance Monitoring always free domain is not supported for autoscaling.

> **Note:**
>
> You cannot enable autoscaling after you have created the Oracle WebLogic Server for OCI domain.
> For autoscaling, ensure that you configure either public load balancer or load balancer with reserved IP.

To enable autoscaling:

1. Select **Enable Autoscaling**.
2. Select a performance metric for the WebLogic Monitoring Metrics.
3. Specify the threshold values as follows:
   - For *CPU Load* and *Used Heap Percent*, select the minimum and maximum threshold percentage.
   - For *Queue Length* and *Stuck threads*, select the minimum and maximum threshold counter values.
4. Enter a user name to access the image in the registry to deploy autoscaling OCI functions.
5. Select the compartment where you have the registry authentication token and then select the secret that contains the registry authentication token that you generated for the user to access the image registry.
6. *Optional*: Enter the email ID to receive scaling notifications.

   It is recommended to subscribe to email notifications.

Configure Tags

Oracle WebLogic Server for OCI can optionally assign tags to the resources (compute, network, and so on) that it creates for your domain.

Tagging allows you to define keys and values and associate them with resources. You can then use the tags to help you organize and find resources based on your business needs. There are separate fields to tag the stack and to tag the resources created within the stack.

1. Select **Create Tags**.
2. To assign a free-form tag, enter the **Tag Key** and **Value**.

   Free-form tag keys and values are case sensitive. For example, `costcenter` and `CostCenter` are treated as different tags.

3. To assign an existing tag, for **Tag Namespace**, select a defined tag, and then select the **Tag Key** and **Value**. If no value is displayed for the **Tag Key**, enter the **Value**.
4. Click **Additional Tag** to assign additional free-form or defined tags.

Create the Domain Stack

After you have specified the WebLogic instance variables, finish creating the domain stack.

On the Review page of the Create Stack wizard, review the information you have provided, and then click **Create**.

The Job Details page of the stack in Resource Manager is displayed. A stack creation job name has the format `ormjob`*`yyyymmddnnnnnn`*. (for example, `ormjob20190919165004`). 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. In the **Application Information** tab, you can directly access the OCI resources using the WebLogic instance IP and the bastion instance IP.

> **Note:**
>
> If there is an error during the creation of the stack, the compute, network, and other resources in the stack are not automatically deleted. If you want to delete the failed stack, see Delete a Stack.

Use Your New Domain

Access and manage your new domain after creating a stack with Oracle WebLogic Server for OCI.

Typical tasks that you might perform after creating a domain:

- View and manage the cloud resources that were created to support your domain. See View the Cloud Resources for a Stack.
- Use the WebLogic Server administration console to configure your domain. Create data sources, JMS modules, Coherence clusters, and so on, or deploy applications. See Access the WebLogic Console.
- Access the sample application that's deployed to your domain. See Access the Sample Application.
- Secure access to your applications using Oracle Identity Cloud Service. See Secure a Domain Using Identity Cloud Service.
- Add your own SSL certificate to the load balancer. See Add a Certificate to the Load Balancer.
- Troubleshoot a problem with your new stack. See Stack Creation Failed.

Create a Basic Domain in a Private Subnet

When you use Oracle WebLogic Server for OCI to create a stack and assign the Oracle WebLogic Server compute instances to a private subnet, the instances are not accessible from the public Internet.

To access the virtual machines (VMs) created in the private subnet, a bastion host is required. You must create a bastion host with a FastConnect or a VPN connection before you create a domain or you must choose to have a bastion host created for you.

To create a domain that uses a database for JRF components, see Create a JRF-Enabled Domain.

Before you create a domain, you must first perform the following tasks:

- Create a compartment. See Create a Compartment.
- Create an SSH key. See Create an SSH Key.
- Create an encryption key to use for secrets. See Create an Encryption Key.
- Create secrets for the passwords you want to use for the domain. You will need to select the compartment where you have the secret and the secret that contains the password. See Create Secrets for PasswordsSee Create Secrets for Passwords.
- Create a confidential application in Oracle Identity Cloud Service if you want to use Oracle Identity Cloud Service for authentication in the domain. You will need the client ID and client secret for this confidential application. See Create a Confidential Application. You will also need to create a secret for the client secret and copy the OCID. See Create Secrets for Passwords.
- Create a FastConnect or a VPN connection if you want to use your own bastion host to administer your Compute instances. See VPN Connect or FastConnect in the Oracle Cloud Infrastructure documentation.

Oracle WebLogic Server for OCI can create the virtual cloud network (VCN) and subnets for your new domain. If you want to use an existing VCN or existing subnets for the domain, then they must meet certain requirements. See:

- Create a Virtual Cloud Network
- Create a Private Subnet for the Oracle WebLogic Server Nodes
- Create a Subnet for the Bastion Node
- Create a Subnet for the Load Balancer (if you want to create a load balancer)

Topics:

- Launch a Stack
- Specify Stack Information
- Configure WebLogic Instance Parameters
- Configure Advanced Parameters for a Domain
- Configure Network Parameters
- Configure a Bastion
- Configure a Load Balancer

- Configure File Storage
- Create OCI Policies
- Configure WebLogic Authentication
- Configure Database Parameters
- Configure a Data Source for an Application Database
- Set Local VCN Peering for an Application Database
- Configure Observability
- Configure Autoscaling
- Configure Tags
- Create the Domain Stack
- Use Your New Domain

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 ☰, select **Marketplace**, and then click **All Applications**.
3. Set the filter type to **Stack**.
4. Select an application that matches the edition of Oracle WebLogic Server that you want to provision, and also uses the type of billing you want (Universal Credits or Bring Your Own License).
   - **Oracle WebLogic Server Standard Edition BYOL**
   - **Oracle WebLogic Server Enterprise Edition BYOL**
   - **Oracle WebLogic Server Enterprise Edition UCM**
   - **Oracle WebLogic Suite BYOL**
   - **Oracle WebLogic Suite UCM**

   > **Note:**
   >
   > If you set the filter type to **Image**, follow the steps in Create an Instance Using the Marketplace in *Images for Oracle WebLogic Server for OCI*.

5. Select a version of Oracle WebLogic Server.

   The latest 14.1.1.0.0 patch level is the default.
   If multiple builds are available for the same patch level (`.01`, `.02`, `.03`, and so on), choose the latest build.

6. Select the compartment in which to create the stack.

   By default the stack compartment is used to contain the domain compute instances and network resources. If later on you specify a network compartment on the Configure Variables page of the Create Stack wizard, then only the compute instances are created in the stack compartment that you select here.

7. Select the **Oracle Standard Terms and Restrictions** check box, and then click **Launch Stack**.

   The Create Stack wizard is displayed.

Specify 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 Instance Parameters

Specify the parameters needed to configure the WebLogic instance domain.

1. In the WebLogic Server Instance section, enter the resource name prefix.

   The maximum character length is 16.

   This prefix is used by all the created resources.

2. Select the WebLogic Server shape for the compute instances.

   The following shapes are supported:

   - Standard: `VM.Standard2.x`, `VM.Standard.E2.x`, `BM.Standard2.x`, `BM.Standard.E2.x`, `BM.Standard3.64`
   - Flexible: `VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`
   - Optimized: `BM.Optimized3`, `VM.Optimized3.Flex`

   > **Note:**
   >
   > In regional subnets, select the WebLogic Server shape that has sufficient service limits for an availability domain, else the provisioning fails.

3. For the flexible shapes (`VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`, `VM.Optimized3.Flex`), select the OCPU count for compute instances.

> **Note:**
>
> You can specify the OCPU count only for the flexible shapes; the maximum number of OCPUs that you can specify for `VM.Standard.E3.Flex` and `VM.Standard.E4.Flex` is 64 OCPUs, for `VM.Standard3.Flex` is 32 OCPUs, and for `VM.Optimized3.Flex` is 18 OCPUs. The memory, network bandwidth, and number of Virtual Network Interface Cards (VNICs) scale proportionately with the number of OCPUs.

4. Enter the SSH public key, by either uploading the SSH key file or pasting the contents of your SSH public key file.
5. Select the number of managed servers you want to create. For 12c and 14c versions and all the editions, you can specify up to `8` nodes, which can be scaled out to 30 when you edit the domain.

   The managed servers will be members of a cluster, unless you selected WebLogic Server Standard Edition.
6. Enter a user name for the WebLogic Server administrator.
7. Select the compartment where you have the WebLogic Server administration secret and then select the secret that contains the administration password. To create secrets, see Create Secrets for Passwords.
8. Select the JDK version for Oracle WebLogic Server. The default JDK version is 8.

> **Note:**
>
> You can select the JDK version only for Oracle WebLogic Server 14.1.1.0.

Configure Advanced Parameters for a Domain

You can optionally specify additional parameters by selecting **WLS Instance Advanced Configuration** on the Configure Variables page of the Create Stack wizard.

Select **WLS Instance Advanced Configuration** if you want to change the default port numbers or remove the sample application, and specify the WebLogic startup arguments. You can use the server startup arguments to provide arguments to the Java Virtual Machine for WebLogic Server instances.

- Cluster-related parameters are not applicable if you selected WebLogic Server Standard Edition.
- The port numbers 9071-9074 are reserved for internal domain communication.
- *Optional*: Specify the **WebLogic Server Startup Arguments** to scale out managed servers. When the servers are scaled out, any changes to the server startup arguments applies to the added nodes only. For example, `-Xms1024m -Xmx1024m`.

Configure Network Parameters

Define the Virtual Cloud Network (VCN) and subnet configuration for a private domain.

1. Select a Virtual Cloud Network (VCN) strategy:
   - Select **Use Existing VCN**, and then select the name of the existing VCN.
   - Select **Create New VCN**, and then enter a name and CIDR for the new VCN.
2. In the WebLogic Server Network section of the Configure Variables page, select the **Network Compartment** in which to create the network resources for this domain.

   If you don't specify a network compartment, then all the network resources and the domain 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.
3. Select one of the following subnet strategies:
   - Select **Use Existing Subnet**.
   - Select **Create New Subnet**.

   > **Note:**
   >
   > If you're creating a new VCN, you can only create a new regional subnet.
4. If you use an existing VCN and subnet, validate the network and then enter `YES` in the **Validated Existing Network** field. To validate a network, see Validate Existing Network Setup.
5. Select the **Subnet Compartment** to use for the existing subnet.

   The subnet compartment is different than the VCN compartment. The subnets for the WebLogic Server nodes, load balancer and the bastion node use this same subnet compartment.

   > **Note:**
   >
   > You can specify the subnet compartment only if you're using an existing subnet.
6. Keep the default **Use Public Subnet** selection.
7. For the WebLogic Server subnet, specify one of the following:
   - If you want to use an existing regional subnet, then choose the name of an existing regional subnet from the list of regional and availability domain-specific subnets.
   - If you are creating a new regional subnet, specify a CIDR for the new subnet.
8. *Optional*: If you are using a regional subnet, then from the WebLogic Admin Sever Availability Domain, select the availability domain in which you want to create the WebLogic administration server compute instance. If you do not select an availability domain, then by default, the compute instance is created in availability domain 1.

Configure a Bastion

You can configure a bastion compute instance on a public subnet to provide access to the WebLogic Server compute instances on a private subnet. However, creating the bastion node on public subnet is optional.

> **Note:**
>
> - By default, **Provision Bastion Node on Public Subnet** is selected when an existing private subnet is selected. If you do not select this option, no status is returned for provisioning, then you must check the status of provisioning by connecting to each compute instance and confirm that the `/u01/provStartMarker` file exists with details found in the file `/u01/logs/provisioning.log` file.
> - It is recommended to deselect the **Provision Bastion Node on Public Subnet** option only in network with fast connect setup.
> - The **Provision Bastion Node on Public Subnet** option is not available when you are creating a new subnet for a new VCN or existing VCN.

To configure a bastion:

1. If your want to use a bastion compute instance with a reserved public IP, then select **Assign Reserved Public IP to Bastion Instance**.
2. For the bastion host subnet, specify one of the following:
   - If you want to use an existing regional subnet, then choose the name of an existing regional subnet from the list of regional and availability domain-specific subnets.
   - If you are creating a new regional subnet, specify a CIDR for the new subnet.
3. Select a shape for the bastion compute instance.

> **Note:**
>
> You must provision a bastion to use VCN peering, as there are SSH requirements which require a bastion.

Configure a Load Balancer

You have the option to create a load balancer to distribute application traffic to the WebLogic Managed Servers. You can also use an existing load balancer for an existing VCN and an existing subnet, to distribute application traffic to the WebLogic Managed Servers.

> **Note:**
>
> If you enable autoscaling for WebLogic instances, you must configure a load balancer in Oracle Cloud Infrastructure when you create a stack, else the stack provisioning fails with a validation error.

To create a load balancer:

1. Select **Add Load Balancer**, if not already selected.

   This option is selected by default.

2. Configure the load balancer network.

   - If you chose to use an existing regional subnet for WebLogic Server, then select an existing regional subnet from the list of regional and availability domain-specific subnets. A load balancer can have only one regional subnet, which is shared between both nodes.

     > **Note:**
     >
     > This option is not applicable if you use an existing load balancer.

   - If you chose to create a regional subnet for WebLogic Server, then specify a CIDR for the new load balancer subnet.

3. Configure load balancer:

   - If you chose to create a new Virtual Cloud Network (VCN), or use an existing VCN and a new subnet:

     a. Select **Create New Load Balancer**.

   - If you chose to use an existing VCN and an existing subnet, you can do one of the following:

     a. Select **Create Load Balancer**.

     b. Select **Use Existing Load Balancer**.

        > **Note:**
        >
        > The existing load balancer should be in the same compartment as the stack compartment, and should be in the same VCN as the existing VCN chosen for the stack.

        i. Specify the OCID for the existing load balancer.

        ii. Enter the name of the backend set for the existing load balancer that has a routing policy associated with the backend set. The backend set should not have backends.
        See Configure the Load Balancer.

- If you create a new load balancer, select **Private Load Balancer**, if you do not want to assign a public IP address to the load balancer.
  This option is available only if you use a private subnet for WebLogic Server. You cannot create a load balancer in a private subnet.
- If you create a new load balancer, select **Load Balancer with Reserved Public IP**, if you want to use a public load balancer with a reserved public IP. Then, specify the OCID of the public IP for the load balancer.
- If you create a new load balancer, select a minimum and maximum flexible load balancer shape.
  By default, the minimum bandwidth size is set to 10Mbps and maximum to 400Mbps.

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

Configure File Storage

When you create an Oracle WebLogic Server for OCI domain, you can add a file storage.

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

To create a file storage:

1. Select **Add File System Storage**.
2. Configure the file storage:
   - If you chose to create a Virtual Cloud Network (VCN):
     a. *Optional:* Select the availability domain in which you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and a new subnet:
     a. *Optional:* Select the availability domain where you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and an existing subnet:
     – If you *do not* want to use an existing mount target or an existing file system:
       a. *Optional:* Select the availability domain where you want to create the file system and mount target.
       b. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
     – If you select **Existing Mount Target**:

   a. Select the compartment where you have the existing mount target. The mount target must reside within the subnet in the selected VCN.
   b. Specify the OCID of the existing mount target ID.

- If you select **Existing File System**:
   a. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
   b. Select the compartment where you have the existing file system.
   c. Specify the OCID of the existing file system.
- If you select both **Existing Mount Target** and **Existing File System**:
   a. Select the compartment where you have the existing mount target.
   b. Specify the OCID of the existing mount target ID.
   c. Select the compartment where you have the existing file system.
   d. Specify the OCID of the existing file system. The existing file system must be in the same availability domain as the existing mount target.

Create OCI Policies

When you create a basic domain in a private subnet, by default the **OCI Policies** check box is selected and Oracle WebLogic Server for OCI 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 domain.

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.

Configure WebLogic Authentication

You have the option to use Oracle Identity Cloud Service to authenticate application users for your domain.

This configuration is only available if the domain meets these requirements:

- Running WebLogic Server 12c
- Includes a load balancer

To use Oracle Identity Cloud Service for authentication:

1. Select **Enable Authentication Using Identity Cloud Service**.
2. Enter your Oracle Identity Cloud Service (IDCS) tenant name, which is also referred to as the instance ID.

   This ID is typically found in the URL that you use to access Oracle Identity Cloud Service, and has the format `idcs-<GUID>`.
3. Enter the client ID of an existing confidential application in this Oracle Identity Cloud Service instance.

4. Select the compartment where you have the IDCS secret and then select the secret that contains the confidential application password in IDCS. To create secrets, see Create Secrets for Passwords.
5. If necessary, you can override the default domain name and port that you use to access Oracle Identity Cloud Service, or the default port that is used for the App Gateway software appliance.

Configure Database Parameters

A basic WebLogic Server 12c domain does not require a database.

A database is required only if you want to create a domain that includes the Java Required Files (JRF) components. Do not select the **Provision with JRF** checkbox if you're not creating a JRF-enabled domain.

To create a domain that uses a database for JRF components, see Create a JRF-Enabled Domain.

Configure a Data Source for an Application Database

When you create an Oracle WebLogic Server for OCI domain, you can configure the application database to create a data source configuration that enables you to connect to Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System).

The database that you connect to is used to contain the schemas for the application database.

You can configure the application database only for Oracle WebLogic Server Enterprise Edition and Oracle WebLogic Suite.

> **Note:**
>
> You cannot configure the data source for an application database using database connect string.

If you selected the option to create a new VCN or selected the option to use an existing VCN with a new subnet, on the Configure Variables page, select the **Configure Application Datasource** checkbox to display the Database options. Then in the Application Database section, select the **Application Database Strategy** for your application database and configure the database parameters.

> **Note:**
>
> This is not applicable if you use an existing VCN and existing subnet, as the WebLogic server and database must be in the same VCN.

- If using **Oracle Autonomous Database**, select or enter the following:
  - The compartment in which you've created the application database.

    - The autonomous database where you want to create the schemas for the application database.
    - The name of an autonomous database user to configure the application database.
    - Select the compartment where you have the application autonomous database secret and then select the secret that contains the application autonomous database user password in the autonomous database. To create secrets, see Create Secrets for Passwords.
    - The service level that the domain should use to connect to the application database for the selected autonomous database.
- If using **Database System**, select or enter the following:
    - The compartment in which you've created the application database.
    - The DB system to use for this application database.
    - The compartment in which the application database's VCN is found.
    - The VCN on which you've created the application database. If this VCN is different than the WebLogic Server VCN, they cannot have overlapping CIDRs. For example, you cannot create a domain on VCN `10.0.0.0/16` that uses a database on VCN `10.0.0.1/24`.
    - The database home within the selected application database system.
    - The version of the selected database home.
    - The database within the selected DB system where you want to create the schemas for the application database.
    - The Pluggable database (PDB) name, only if the selected application database is running Oracle Database 12c or later.
    - The name of a database user to configure the application database.
    - Select the compartment where you have the database secret and then select the secret that contains the application database user password. To create secrets, see Create Secrets for Passwords.
    - The application database listen port (1521 by default)
- If using **Database System**, then Oracle WebLogic Server for OCI creates a security list in the VCN on which you've created the application database. This security list allows the WebLogic Server subnet to access the application database port. If this step isn't required or you don't have the correct permissions to modify the database network, clear the **Create Application Database Security List** check box.

Set Local VCN Peering for an Application Database

If you selected the option to create a new VCN or selected the option to use an existing VCN with a new subnet for the WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database, you can either disable the local VCN peering or configure the local VCN peering for the Application Database.

Ensure that the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database are peered before creating the stack for the Oracle WebLogic Server for OCI domain. See Local VCN Peering to peer the VCNs manually. In this case, the stack is provisioned based on the database private IP address.

> **Note:**
>
> This is not applicable if you use an existing VCN and existing subnet, as the WebLogic server and database must be in the same VCN.

If you choose to create a virtual cloud network for an Oracle WebLogic Server domain, use Oracle WebLogic Server for OCI to create a Local Peering Gateway, else create a network with VCN peering and then use this existing network to provision the domain.

> **Note:**
>
> You must provision a bastion to use VCN peering, as there are SSH requirements which require a bastion.

If the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database system have not been peered, you can use Oracle WebLogic Server for OCI to update the two VCNs so that they can communicate.

Oracle WebLogic Server for OCI creates a public subnet in each VCN, and then creates a compute instance in each subnet. These compute instances run software to forward DNS requests across the VCNs.

You cannot use existing subnets for the DNS Forwarder compute instances.

1. Specify a CIDR for the new subnet in the WebLogic Server VCN.
2. Specify a CIDR for the new subnet in the application database VCN.
3. Select a shape for the new DNS Forwarder compute instance in each VCN.

Configure Observability

Oracle WebLogic Server for OCI can optionally export logs to OCI Logging Service, and provide visibility into the performance of applications using the Oracle Cloud Infrastructure Application Performance Monitoring (APM) service.

Select **Configure Observability** to enable logging and monitoring service integration for your WebLogic instances.

- Select **Enable exporting logs to OCI Logging Service** to enable logging for the WebLogic instances.
- Select **Enable Application Performance Monitoring** to export WebLogic metrics using Application Performance Monitoring (APM) Java Agent and create dashboards with WebLogic specific metrics. This is required for metric-based autoscaling of instances.
  Specify the OCID of the Application Performance Monitoring domain and the private data key for your existing Application Performance Monitoring domain.

When you enable Application Performance Monitoring, you can use autoscaling to scale out or scale in instances. See Configure Autoscaling.

> **Note:**
>
> If you enable autoscaling, you cannot use the always free Application Performance Monitoring domain.

Configure Autoscaling

When you create an Oracle WebLogic Server for OCI domain, you can enable autoscaling for WebLogic instances.

This configuration is only available if **Application Performance Monitoring** is enabled for the WebLogic instance. See Configure Observability. Application Performance Monitoring always free domain is not supported for autoscaling.

> **Note:**
>
> You cannot enable autoscaling after you have created the Oracle WebLogic Server for OCI domain.
> For autoscaling, ensure that you configure either public load balancer, private load balancer, or load balancer with reserved IP.

To enable autoscaling:

1. Select **Enable Autoscaling**.
2. Select a performance metric for the WebLogic Monitoring Metrics.
3. Specify the threshold values as follows:
   - For *CPU Load* and *Used Heap Percent*, select the minimum and maximum threshold percentage.
   - For *Queue Length* and *Stuck threads*, select the minimum and maximum threshold counter values.
4. Enter a user name to access the image in the registry to deploy autoscaling OCI functions.
5. Select the compartment where you have the registry authentication token and then select the secret that contains the registry authentication token that you generated for the user to access the image registry.
6. *Optional*: Enter the email ID to receive scaling notifications.

   It is recommended to subscribe to email notifications.

Configure Tags

Oracle WebLogic Server for OCI can optionally assign tags to the resources (compute, network, and so on) that it creates for your domain.

Tagging allows you to define keys and values and associate them with resources. You can then use the tags to help you organize and find resources based on your business needs. There are separate fields to tag the stack and to tag the resources created within the stack.

1. Select **Create Tags**.
2. To assign a free-form tag, enter the **Tag Key** and **Value**.

   Free-form tag keys and values are case sensitive. For example, `costcenter` and `CostCenter` are treated as different tags.
3. To assign an existing tag, for **Tag Namespace**, select a defined tag, and then select the **Tag Key** and **Value**. If no value is displayed for the **Tag Key**, enter the **Value**.
4. Click **Additional Tag** to assign additional free-form or defined tags.

Create the Domain Stack

After you have specified the WebLogic instance variables, finish creating the domain stack.

On the Review page of the Create Stack wizard, review the information you have provided, and then click **Create**.

The Job Details page of the stack in Resource Manager is displayed. A stack creation job name has the format `ormjob`*`yyyymmddnnnnnn`*. (for example, `ormjob20190919165004`). 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. In the **Application Information** tab, you can directly access the OCI resources using the WebLogic instance IP and the bastion instance IP.

> **Note:**
>
> If there is an error during the creation of the stack, the compute, network, and other resources in the stack are not automatically deleted. If you want to delete the failed stack, see Delete a Stack.

Use Your New Domain

Access and manage your new domain after creating a stack with Oracle WebLogic Server for OCI.

Typical tasks that you might perform after creating a domain:

- View and manage the cloud resources that were created to support your domain. See View the Cloud Resources for a Stack.
- Use the WebLogic Server administration console to configure your domain. Create data sources, JMS modules, Coherence clusters, and so on, or deploy applications. See Access the WebLogic Console in a Private Subnet.
- Access the sample application that's deployed to your domain. See Access the Sample Application in a Private Subnet.
- Secure access to your applications using Oracle Identity Cloud Service. See Secure a Domain Using Identity Cloud Service.
- Add your own SSL certificate to the load balancer. See Add a Certificate to the Load Balancer.
- Troubleshoot a problem with your new stack. See Stack Creation Failed.

- If you already have an existing bastion to provide public access to the domain, or if you already have a VPN connection to your on-premise network, then you can delete the new bastion compute instance that was created for your domain.

Create a JRF-Enabled Domain

Creating a JRF-enabled domain is similar to creating a basic domain in Oracle WebLogic Server for OCI; however, a database in Oracle Cloud Infrastructure is required as the domain is provisioned with the Java Required Files (JRF) components.

> **Note:**
>
> Oracle WebLogic Server 14c does not support JRF, so you cannot create a JRF-enabled domain using Oracle WebLogic Server 14.1.1.0.

From Marketplace, create a stack by entering parameters that automatically create a domain. When creating a JRF-enabled domain, you specify an autonomous database or DB system database. You can also specify a public subnet (either a regional or availability domain-specific) or a private subnet for the domain.

> **Note:**
>
> Oracle WebLogic Server 12c must be specified as the version for a JRF-enabled domain if you intend to use an Oracle Autonomous Database.

Before you create a domain, you must first perform the following tasks:

- Create database in Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System). See Create a Database.
- Create a compartment for your domain resources, or use the same compartment in which you created the database. See Create a Compartment.
- Create an SSH key. See Create an SSH Key.
- Create an encryption key to use for secrets. See Create an Encryption Key.
- Create secrets for the passwords you want to use for the domain. You will need to select the compartment where you have the secret and the secret that contains the password. See Create Secrets for PasswordsSee Create Secrets for Passwords.
- Identify the pluggable database (PDB) name. This is required only for Oracle Cloud Infrastructure Database (DB System) running Oracle Database 12c or later.
- Create a confidential application in Oracle Identity Cloud Service if you want to use Oracle Identity Cloud Service for authentication in the domain. You will need the client ID and client secret for this confidential application. See Create a Confidential Application. You will also need to create a secret for the client secret and copy the OCID. See Create Secrets for Passwords.

Oracle WebLogic Server for OCI can create the virtual cloud network (VCN) and subnets for your new domain. If you want to use an existing VCN or existing subnets for the domain, then they must meet certain requirements. See:

- Create a Virtual Cloud Network
- Create a Private Subnet for the Oracle WebLogic Server Nodes
- Create a Subnet for the Bastion Node
- Create a Subnet for the Load Balancer (if you want to create a load balancer)

Tutorial (using an autonomous database)

Tutorial (using a DB System database)

Topics:

- Launch a Stack
- Specify Stack Information
- Configure WebLogic Instance Parameters
- Configure Advanced Parameters for a Domain
- Configure Network Parameters
- Configure a WebLogic Console Port
- Configure a Load Balancer
- Configure File Storage
- Create OCI Policies
- Configure WebLogic Authentication
- Configure Database Parameters
- Set Local VCN Peering
- Configure a Data Source for an Application Database
- Set Local VCN Peering for an Application Database
- Configure Observability
- Configure Autoscaling
- Configure Tags
- Create the Domain Stack
- Use Your New Domain

Launch a Stack

Use Marketplace to specify initial stack information.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Marketplace**, and then click **All Applications**.
3. Set the filter type to **Stack**.
4. Select an application that matches the edition of Oracle WebLogic Server that you want to provision, and also uses the type of billing you want (Universal Credits or Bring Your Own License).

- **Oracle WebLogic Server Standard Edition BYOL**
- **Oracle WebLogic Server Enterprise Edition BYOL**
- **Oracle WebLogic Server Enterprise Edition UCM**
- **Oracle WebLogic Suite BYOL**
- **Oracle WebLogic Suite UCM**

> **Note:**
>
> If you set the filter type to **Image**, follow the steps in Create an Instance Using the Marketplace in *Images for Oracle WebLogic Server for OCI*.

5. Select a version of Oracle WebLogic Server 12c.

   The latest 14.1.1.0.0 patch level is the default version.
   If multiple builds are available for the same patch level (`.01`, `.02`, `.03`, and so on), choose the latest build.

   > **Note:**
   >
   > To use an Oracle Autonomous Database, you must select a 12c version.

6. Select the compartment in which to create the stack.

   By default the stack compartment is used to contain the domain compute instances and network resources. If later on you specify a network compartment on the Configure Variables page of the Create Stack wizard, then only the compute instances are created in the stack compartment that you select here.

7. Select the **Oracle Standard Terms and Restrictions** check box, and then click **Launch Stack**.

   The Create Stack wizard is displayed.

Specify 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 Instance Parameters

Specify the parameters needed to configure the WebLogic instance domain.

1. In the WebLogic Server Instance section, enter the resource name prefix.

   The maximum character length is 16.

This prefix is used by all the created resources.

2. Select the WebLogic Server shape for the compute instances.

   The following shapes are supported:

   - Standard: `VM.Standard2.x`, `VM.Standard.E2.x`, `BM.Standard2.x`, `BM.Standard.E2.x`, `BM.Standard3.64`
   - Flexible: `VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`
   - Optimized: `BM.Optimized3`, `VM.Optimized3.Flex`

   > **Note:**
   >
   > In regional subnets, select the WebLogic Server shape that has sufficient service limits for an availability domain, else the provisioning fails.

3. For the flexible shapes (`VM.Standard.E3.Flex`, `VM.Standard.E4.Flex`, `VM.Standard3.Flex`, `VM.Optimized3.Flex`), select the OCPU count for compute instances.

   > **Note:**
   >
   > You can specify the OCPU count only for the flexible shapes; the maximum number of OCPUs that you can specify for `VM.Standard.E3.Flex` and `VM.Standard.E4.Flex` is 64 OCPUs, for `VM.Standard3.Flex` is 32 OCPUs, and for `VM.Optimized3.Flex` is 18 OCPUs. The memory, network bandwidth, and number of Virtual Network Interface Cards (VNICs) scale proportionately with the number of OCPUs.

4. Enter the SSH public key, by either uploading the SSH key file or pasting the contents of your SSH public key file.

5. Select the number of managed servers you want to create. For 12c and 14c versions and all the editions, you can specify up to `8` nodes, which can be scaled out to 30 when you edit the domain.

   The managed servers will be members of a cluster, unless you selected WebLogic Server Standard Edition.

6. Enter a user name for the WebLogic Server administrator.

7. Select the compartment where you have the WebLogic Server administration secret and then select the secret that contains the administration password. To create secrets, see Create Secrets for Passwords.

Configure Advanced Parameters for a Domain

You can optionally specify additional parameters by selecting **WLS Instance Advanced Configuration** on the Configure Variables page of the Create Stack wizard.

Select **WLS Instance Advanced Configuration** if you want to change the default port numbers or remove the sample application, and specify the WebLogic startup arguments. You can use the server startup arguments to provide arguments to the Java Virtual Machine for WebLogic Server instances.

- Cluster-related parameters are not applicable if you selected WebLogic Server Standard Edition.
- The port numbers 9071-9074 are reserved for internal domain communication.
- *Optional*: Specify the **WebLogic Server Startup Arguments** to scale out managed servers. When the servers are scaled out, any changes to the server startup arguments applies to the added nodes only. For example, `-Xms1024m -Xmx1024m`.

Configure Network Parameters

Define the Virtual Cloud Network (VCN) and subnet configuration for the domain.

1. Select a Virtual Cloud Network (VCN) strategy:
   - Select **Use Existing VCN**, and then select the name of the existing VCN.
   - Select **Create New VCN**, and then enter a name and CIDR for the new VCN.
2. In the WebLogic Server Network section of the Configure Variables page, select the **Network Compartment** in which to create the network resources for this domain.

   If you don't specify a network compartment, then all the network resources and the domain 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.
3. Select one of the following subnet strategies:
   - Select **Use Existing Subnet**.
   - Select **Create New Subnet**.

   > **Note:**
   >
   > If you're creating a new VCN, you can only create a new regional subnet.
4. If you use an existing VCN and subnet, validate the network and then enter `YES` in the **Validated Existing Network** field. To validate a network, see Validate Existing Network Setup.
5. Select the **Subnet Compartment** to use for the existing subnet.

   The subnet compartment is different than the VCN compartment. The subnets for the WebLogic Server nodes, load balancer and the bastion node use this same subnet compartment.

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

6. If you selected to use an existing subnet, then for subnet type, select **Use Public Subnet** or **Use Private Subnet**. Compute instances in a private subnet are not directly accessible from outside of Oracle Cloud.
7. For the WebLogic Server subnet, specify one of the following:
   - If you want to use an existing regional subnet, then choose the name of an existing regional subnet from the list of regional and availability domain-specific subnets.
   - If you are creating a new regional subnet, specify a CIDR for the new subnet.
8. If you selected **Use Private Subnet**, then configure the bastion compute instance.
   a. Specify one of the following for the bastion subnet:
      - If you want to use an existing regional subnet, then choose the name of an existing regional subnet from the list of regional and availability domain-specific subnets.
      - If you are creating a new regional subnet, then specify a CIDR for the new subnet.

   b. *Optional*: If you are using a regional subnet, then from the WebLogic Admin Sever Availability Domain, select the availability domain in which you want to create the WebLogic administration server compute instance. If you do not select an availability domain, then by default, the compute instance is created in availability domain 1.

   c. Select a shape for the bastion compute instance.
9. *Optional*: If you are using a regional subnet, then from the WebLogic Admin Sever Availability Domain, select the availability domain in which you want to create the WebLogic administration server compute instance. If you do not select an availability domain, then by default, the compute instance is created in availability domain 1.
10. If your want to use a bastion compute instance with a reserved public IP, then select **Assign Reserved Public IP to Bastion Instance**.

Configure a WebLogic Console Port

If you are creating a new VCN with public subnets, then you have the option to disclose the Oracle WebLogic Administration Server Console port in a public subnet.

> **WARNING:**
>
> Oracle does not recommend that you open the WebLogic Administrative port to the internet when the WebLogic Server is in a public subnet. If your WebLogic Server is in a public subnet and you need to access WebLogic Administrative Console, then you can restrict the IP addresses by creating a security rule and open WebLogic Server Administrative port only to a CIDR block. Oracle WebLogic Server for OCI can configure this security rule for you during provisioning. That is, if you select a public subnet, ensure to limit the CIDR range to access WebLogic administration console ports, the defaults ports are `7001` and `7002` for `http` and `https` respectively.

1. Select **Enable Access to Administration Console**.
2. Specify the CIDR to create a security list to allow access to the WebLogic administration console port to the source CIDR range.

Configure a Load Balancer

You have the option to create a load balancer to distribute application traffic to the WebLogic Managed Servers. You can also use an existing load balancer for an existing VCN and an existing subnet, to distribute application traffic to the WebLogic Managed Servers.

> **Note:**
>
> If you enable autoscaling for WebLogic instances, you must configure a load balancer in Oracle Cloud Infrastructure when you create a stack, else the stack provisioning fails with a validation error.

To create a load balancer:

1. Select **Add Load Balancer**, if not already selected.

   This option is selected by default.
2. Configure the load balancer network.
   - If you chose to use an existing regional subnet for WebLogic Server, then select an existing regional subnet from the list of regional and availability domain-specific subnets. A load balancer can have only one regional subnet, which is shared between both nodes.

     > **Note:**
     >
     > This option is not applicable if you use an existing load balancer.

- If you chose to create a regional subnet for WebLogic Server, then specify a CIDR for the new load balancer subnet.

3. Configure load balancer:
    - If you chose to create a new Virtual Cloud Network (VCN), or use an existing VCN and a new subnet:
        a. Select **Create New Load Balancer**.
    - If you chose to use an existing VCN and an existing subnet, you can do one of the following:
        a. Select **Create Load Balancer**.
        b. Select **Use Existing Load Balancer**.

            > **Note:**
            >
            > The existing load balancer should be in the same compartment as the stack compartment, and should be in the same VCN as the existing VCN chosen for the stack.

            i. Specify the OCID for the existing load balancer.

            ii. Enter the name of the backend set for the existing load balancer that has a routing policy associated with the backend set. The backend set should not have backends.
            See Configure the Load Balancer.
    - If you create a new load balancer, select **Private Load Balancer**, if you do not want to assign a public IP address to the load balancer.
      This option is available only if you use a private subnet for WebLogic Server. You cannot create a load balancer in a private subnet.
    - If you create a new load balancer, select **Load Balancer with Reserved Public IP**, if you want to use a public load balancer with a reserved public IP. Then, specify the OCID of the public IP for the load balancer.
    - If you create a new load balancer, select a minimum and maximum flexible load balancer shape.
      By default, the minimum bandwidth size is set to 10Mbps and maximum to 400Mbps.

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

Configure File Storage

When you create an Oracle WebLogic Server for OCI domain, you can add a file storage.

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

To create a file storage:

1. Select **Add File System Storage**.
2. Configure the file storage:
   - If you chose to create a Virtual Cloud Network (VCN):
     a. *Optional:* Select the availability domain in which you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and a new subnet:
     a. *Optional:* Select the availability domain where you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and an existing subnet:
     – If you *do not* want to use an existing mount target or an existing file system:
       a. *Optional:* Select the availability domain where you want to create the file system and mount target.
       b. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
     – If you select **Existing Mount Target**:
       a. Select the compartment where you have the existing mount target. The mount target must reside within the subnet in the selected VCN.
       b. Specify the OCID of the existing mount target ID.
     – If you select **Existing File System**:
       a. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
       b. Select the compartment where you have the existing file system.
       c. Specify the OCID of the existing file system.
     – If you select both **Existing Mount Target** and **Existing File System**:
       a. Select the compartment where you have the existing mount target.
       b. Specify the OCID of the existing mount target ID.
       c. Select the compartment where you have the existing file system.
       d. Specify the OCID of the existing file system. The existing file system must be in the same availability domain as the existing mount target.

Create OCI Policies

When you create a JRF-enabled domain, by default the **OCI Policies** check box is selected and Oracle WebLogic Server for OCI 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 domain.

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.

Configure WebLogic Authentication

You have the option to use Oracle Identity Cloud Service to authenticate application users for your domain.

This configuration is only available if the domain meets these requirements:

- Running WebLogic Server 12c
- Includes a load balancer

To use Oracle Identity Cloud Service for authentication:

1. Select **Enable Authentication Using Identity Cloud Service**.
2. Enter your Oracle Identity Cloud Service (IDCS) tenant name, which is also referred to as the instance ID.

   This ID is typically found in the URL that you use to access Oracle Identity Cloud Service, and has the format `idcs-<GUID>`.
3. Enter the client ID of an existing confidential application in this Oracle Identity Cloud Service instance.
4. Select the compartment where you have the IDCS secret and then select the secret that contains the confidential application password in IDCS. To create secrets, see Create Secrets for Passwords.
5. If necessary, you can override the default domain name and port that you use to access Oracle Identity Cloud Service, or the default port that is used for the App Gateway software appliance.

Configure Database Parameters

You must specify a database in Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System) when you create an Oracle WebLogic Server for OCI domain that includes the Java Required Files (JRF) components.

The database you specify is used to contain the required infrastructure schemas for the JRF-enabled domain.

On the Configure Variables page, select the **Provision with JRF** checkbox to display the Database options. Then in the Database section, select the **Database Strategy** for your domain and configure the database parameters.

- If using **Autonomous Database**, select or enter the following:

  - The compartment in which you've created the database.
  - The database where you want to create the JRF schemas for this WebLogic domain.
  - The service level that the domain should use to connect to the selected autonomous database.
  - Select the compartment where you have the autonomous database secret and then select the secret that contains the administration user password in the autonomous database. To create secrets, see Create Secrets for Passwords.
- If using **Database System**, select or enter the following:
  - The compartment in which you've created the database.
  - The compartment in which the database's VCN is found.
  - The VCN on which you've created the database. If this VCN is different than the WebLogic Server VCN, they cannot have overlapping CIDRs. For example, you cannot create a domain on VCN `10.0.0.0/16` that uses a database on VCN `10.0.0.1/24`.
  - The DB system to use for this WebLogic domain.
  - The database home within the selected DB system.
  - The database home version.
  - The database within the selected DB system where you want to create the JRF schemas for this domain.
  - The Pluggable database (PDB) name, only if the selected database is running Oracle Database 12c or later.
  - The name of a database user with SYSDBA privileges.
  - The OCID of the secret that contains the password for the SYSDBA user.
  - Select the compartment where you have the database secret and then select the secret that contains the database administration password. To create secrets, see Create Secrets for Passwords.
  - The database listen port (1521 by default)
- If using **Database System**, then Oracle WebLogic Server for OCI creates a security list in the VCN on which you've created the database. This security list allows the WebLogic Server subnet to access the database port. If this step isn't required or you don't have the correct permissions to modify the database network, clear the **Create DB Security List** check box.
- If using **Database System** with connect string, select the **Use Database Connection String** check box and enter the following:
  - The connect string to connect to the database.

> **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 2-1 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>`

- The name of a database user with SYSDBA privileges.
- Select the compartment where you have the database secret and then select the secret that contains the password of the SYSDBA user. To create secrets, see Create Secrets for Passwords.

> **Note:**
>
> Oracle recommends you to use the connect string for Exadata Database systems.
>
> If you use database connect string, then Oracle WebLogic Server for OCI 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.

Set Local VCN Peering

If you selected the option to create a new VCN or selected the option to use an existing VCN with a new subnet for the WebLogic Server compute instances and the Oracle Cloud Infrastructure Database (DB System), you can either disable the local VCN peering or configure the local VCN peering for the Infrastructure Database.

Ensure that the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database are peered before creating the stack for the Oracle WebLogic Server for OCI domain. See Local VCN Peering to peer the VCNs manually. In this case, the stack is provisioned based on the database private IP address.

> **Note:**
>
> This is not applicable if you use an existing VCN and existing subnet, as the WebLogic server and database must be in the same VCN.

If you choose to create a virtual cloud network for an Oracle WebLogic Server domain, use Oracle WebLogic Server for OCI to create a Local Peering Gateway, else create a network with VCN peering and then use this existing network to provision the domain.

> **Note:**
>
> You must provision a bastion to use VCN peering, as there are SSH requirements which require a bastion.

If the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Database system have not been peered, you can use Oracle WebLogic Server for OCI to update the two VCNs so that they can communicate.

Oracle WebLogic Server for OCI creates a public subnet in each VCN, and then creates a compute instance in each subnet. These compute instances run software to forward DNS requests across the VCNs.

You cannot use existing subnets for the DNS Forwarder compute instances.

1. Specify a CIDR for the new subnet in the WebLogic Server VCN.
2. Specify a CIDR for the new subnet in the database VCN.
3. Select a shape for the new DNS Forwarder compute instance in each VCN.

Configure a Data Source for an Application Database

When you create an Oracle WebLogic Server for OCI domain, you can configure the application database to create a data source configuration that enables you to connect to Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System).

The database that you connect to is used to contain the schemas for the application database.

You can configure the application database only for Oracle WebLogic Server Enterprise Edition and Oracle WebLogic Suite.

> **Note:**
>
> You cannot configure the data source for an application database using database connect string.

If you selected the option to create a new VCN or selected the option to use an existing VCN with a new subnet, on the Configure Variables page, select the **Configure Application Datasource** checkbox to display the Database options. Then in the Application Database section, select the **Application Database Strategy** for your application database and configure the database parameters.

> **Note:**
>
> This is not applicable if you use an existing VCN and existing subnet, as the WebLogic server and database must be in the same VCN.

- If using **Oracle Autonomous Database**, select or enter the following:
  - The compartment in which you've created the application database.
  - The autonomous database where you want to create the schemas for the application database.
  - The name of an autonomous database user to configure the application database.
  - Select the compartment where you have the application autonomous database secret and then select the secret that contains the application autonomous database user password in the autonomous database. To create secrets, see Create Secrets for Passwords.
  - The service level that the domain should use to connect to the application database for the selected autonomous database.
- If using **Database System**, select or enter the following:
  - The compartment in which you've created the application database.
  - The DB system to use for this application database.
  - The compartment in which the application database's VCN is found.
  - The VCN on which you've created the application database. If this VCN is different than the WebLogic Server VCN, they cannot have overlapping CIDRs. For example, you cannot create a domain on VCN `10.0.0.0/16` that uses a database on VCN `10.0.0.1/24`.
  - The database home within the selected application database system.
  - The version of the selected database home.
  - The database within the selected DB system where you want to create the schemas for the application database.
  - The Pluggable database (PDB) name, only if the selected application database is running Oracle Database 12c or later.
  - The name of a database user to configure the application database.
  - Select the compartment where you have the database secret and then select the secret that contains the application database user password. To create secrets, see Create Secrets for Passwords.
  - The application database listen port (1521 by default)
- If using **Database System**, then Oracle WebLogic Server for OCI creates a security list in the VCN on which you've created the application database. This security list allows the WebLogic Server subnet to access the application database port. If this step isn't required or you don't have the correct permissions to modify the database network, clear the **Create Application Database Security List** check box.

Set Local VCN Peering for an Application Database

If you selected the option to create a new VCN or selected the option to use an existing VCN with a new subnet for the WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database, you can either disable the local VCN peering or configure the local VCN peering for the Application Database.

Ensure that the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database are peered before creating the stack for the Oracle WebLogic Server for OCI domain. See Local VCN Peering to peer the VCNs manually. In this case, the stack is provisioned based on the database private IP address.

> **Note:**
>
> This is not applicable if you use an existing VCN and existing subnet, as the WebLogic server and database must be in the same VCN.

If you choose to create a virtual cloud network for an Oracle WebLogic Server domain, use Oracle WebLogic Server for OCI to create a Local Peering Gateway, else create a network with VCN peering and then use this existing network to provision the domain.

> **Note:**
>
> You must provision a bastion to use VCN peering, as there are SSH requirements which require a bastion.

If the VCNs for WebLogic Server compute instances and the Oracle Cloud Infrastructure Application Database system have not been peered, you can use Oracle WebLogic Server for OCI to update the two VCNs so that they can communicate.

Oracle WebLogic Server for OCI creates a public subnet in each VCN, and then creates a compute instance in each subnet. These compute instances run software to forward DNS requests across the VCNs.

You cannot use existing subnets for the DNS Forwarder compute instances.

1. Specify a CIDR for the new subnet in the WebLogic Server VCN.
2. Specify a CIDR for the new subnet in the application database VCN.
3. Select a shape for the new DNS Forwarder compute instance in each VCN.

Configure Observability

Oracle WebLogic Server for OCI can optionally export logs to OCI Logging Service, and provide visibility into the performance of applications using the Oracle Cloud Infrastructure Application Performance Monitoring (APM) service.

Select **Configure Observability** to enable logging and monitoring service integration for your WebLogic instances.

- Select **Enable exporting logs to OCI Logging Service** to enable logging for the WebLogic instances.
- Select **Enable Application Performance Monitoring** to export WebLogic metrics using Application Performance Monitoring (APM) Java Agent and create dashboards with WebLogic specific metrics. This is required for metric-based autoscaling of instances.
  Specify the OCID of the Application Performance Monitoring domain and the private data key for your existing Application Performance Monitoring domain.

When you enable Application Performance Monitoring, you can use autoscaling to scale out or scale in instances. See Configure Autoscaling.

> **Note:**
>
> If you enable autoscaling, you cannot use the always free Application Performance Monitoring domain.

Configure Autoscaling

When you create an Oracle WebLogic Server for OCI domain, you can enable autoscaling for WebLogic instances.

This configuration is only available if **Application Performance Monitoring** is enabled for the WebLogic instance. See Configure Observability. Application Performance Monitoring always free domain is not supported for autoscaling.

> **Note:**
>
> You cannot enable autoscaling after you have created the Oracle WebLogic Server for OCI domain.
> For autoscaling, ensure that you configure either public load balancer, private load balancer, or load balancer with reserved IP.

To enable autoscaling:

1. Select **Enable Autoscaling**.
2. Select a performance metric for the WebLogic Monitoring Metrics.
3. Specify the threshold values as follows:
   - For *CPU Load* and *Used Heap Percent*, select the minimum and maximum threshold percentage.
   - For *Queue Length* and *Stuck threads*, select the minimum and maximum threshold counter values.
4. Enter a user name to access the image in the registry to deploy autoscaling OCI functions.
5. Select the compartment where you have the registry authentication token and then select the secret that contains the registry authentication token that you generated for the user to access the image registry.

6. *Optional*: Enter the email ID to receive scaling notifications.

   It is recommended to subscribe to email notifications.

Configure Tags

Oracle WebLogic Server for OCI can optionally assign tags to the resources (compute, network, and so on) that it creates for your domain.

Tagging allows you to define keys and values and associate them with resources. You can then use the tags to help you organize and find resources based on your business needs. There are separate fields to tag the stack and to tag the resources created within the stack.

1. Select **Create Tags**.
2. To assign a free-form tag, enter the **Tag Key** and **Value**.

   Free-form tag keys and values are case sensitive. For example, `costcenter` and `CostCenter` are treated as different tags.
3. To assign an existing tag, for **Tag Namespace**, select a defined tag, and then select the **Tag Key** and **Value**. If no value is displayed for the **Tag Key**, enter the **Value**.
4. Click **Additional Tag** to assign additional free-form or defined tags.

Create the Domain Stack

After you have specified the WebLogic instance variables, finish creating the domain stack.

On the Review page of the Create Stack wizard, review the information you have provided, and then click **Create**.

The Job Details page of the stack in Resource Manager is displayed. A stack creation job name has the format `ormjob`*`yyyymmddnnnnnn`*. (for example, `ormjob20190919165004`). 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. In the **Application Information** tab, you can directly access the OCI resources using the WebLogic instance IP and the bastion instance IP.

> **Note:**
>
> If there is an error during the creation of the stack, the compute, network, and other resources in the stack are not automatically deleted. If you want to delete the failed stack, see Delete a Stack.

Use Your New Domain

Access and manage your new domain after creating a stack with Oracle WebLogic Server for OCI.

Typical tasks that you might perform after creating a domain:

- View and manage the cloud resources that were created to support your domain. See View the Cloud Resources for a Stack.

- Use the WebLogic Server administration console to configure your domain. Create data sources, JMS modules, Coherence clusters, and so on, or deploy applications. See Access the WebLogic Console.
- Access the sample application that's deployed to your domain. See Access the Sample Application.
- Secure access to your applications using Oracle Identity Cloud Service. See Secure a Domain Using Identity Cloud Service.
- Add your own SSL certificate to the load balancer. See Add a Certificate to the Load Balancer.
- Troubleshoot a problem with your new stack. See Stack Creation Failed.

You can also use the Fusion Middleware Control Console to monitor, configure, and manage a JRF-enabled domain. See Access the Fusion Middleware Control Console.

View the Cloud Resources for a Stack

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

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 job whose type is Apply.
6. Click **Associated Resources**.

   A list of resources in this stack displays. The list might include compute instances, virtual cloud networks (VCN), subnets, security lists, gateways, and block storage volumes.
7. Click the name of a resource to manage it and to view its details.

   For example, click a compute instance to view its IP address or to reboot it.

Alternatively, you can find your domain'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, network, and other resources created by a stack in Oracle Cloud Infrastructure for a domain in Oracle WebLogic Server for OCI.

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

Topics:

- Compute Instances

- Network Resources
- Load Balancer
- Identity Resources for Dynamic Group and Root Policy
- Identity Resources for Oracle Identity Cloud Service

Compute Instances

Depending on the number of nodes you specify for your Oracle WebLogic Server for OCI stack configuration, one or more compute instances are created for your domain.

Each WebLogic Server compute instance name has the following format:

`servicename``-wls-`*`n`*

Where:

- *`servicename`* is the resource name prefix you provided during stack creation
- *`n`* is `0`, `1`, `2`, and so on

For example, a domain with two nodes would have the following compute instances if the resource prefix is `thestack`:

- `thestack-wls-0`
- `thestack-wls-1`

The first compute instance (with the suffix `-wls-0`) runs the WebLogic Administration server `thestack_adminserver` and the first Managed Server `thestack_server_1`. The second compute instance (with the suffix `-wls-1`) runs the second Managed Server `thestack_server_2`, and so on.

If you specified a private subnet for your domain, a bastion instance is created, which is identified by:

`servicename``-bastion-instance`

If you created a JRF-enabled domain, and WebLogic Server and the database are on different VCNs, then Domain Name Service (DNS) compute instances are created:

- *`servicename`*`-wlsdns-0` - DNS Forwarder in the WebLogic Server VCN
- *`servicename`*`-dbsystem-dns` - DNS Forwarder in the database VCN

Network Resources

Several network resources for route tables, security lists, and gateways are created for your Oracle WebLogic Server for OCI domain.

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

Your domain configuration determines the type and number of network resources created. The names of all network resources begin with the resource name prefix you provided during stack creation. The following table provides a summary of the resources that can be created.

Resource Name	Type
servicename-*vcnname*	WebLogic VCN
servicename-wls-subnet	WebLogic regional subnet
servicename-wls-subnet-*adname*	WebLogic availability domain-specific subnet
servicename-bastion-subnet	public subnet for the bastion compute instance
servicename-lb-subnet-1	load balancer regional subnet
servicename-lb-subnet-1-*adname1*	availability domain-specific subnet 1 for load balancer node 1
servicename-lb-subnet-1-*adname2*	availability domain-specific subnet 2 for load balancer node 2
servicename-wls-dns-subnet-*adname*	public subnet for the DNS Forwarder in the WebLogic VCN, for local VCN peering
servicename-dbsystem-dns-subnet-*adname*	public subnet for the DNS Forwarder in the database VCN, for local VCN peering
Default route table for *servicename*-*vcnname*	default route table for the WebLogic VCN
servicename-public-routetable	route table for a subnet
servicename-dbsystem-routetable	database route table, for local VCN peering
servicename-internet-gateway	internet gateway for the WebLogic VCN
servicename-service-gateway	service gateway for the WebLogic VCN
servicename-wls-lpg	local peering gateway in the WebLogic VCN
servicename-dbsystem-lpg	local peering gateway in the database VCN
Default security list for *servicename*-*vcnname*	default security list for the VCN
servicename-internal-security-list	security list for the WebLogic subnet
servicename-bastion-security-list	security list for the bastion subnet
servicename-wls-bastion-security-list	security list for the bastion and WebLogic subnets
servicename-wls-ms-security-list	security list for the WebLogic Managed Servers
servicename-lb-security-list	security list for the load balancer regional subnet
servicename-wls-lb-security-list-1	security list for the load balancer node 1 and WebLogic subnets
servicename-wls-lb-security-list-2	security list for the load balancer node 2 and WebLogic subnets
servicename-wls_dns_security_list	security list for the DNS subnet in the WebLogic VCN, for local VCN peering
servicename-dbsystem-dns-security-list	security list for the DNS subnet in the database VCN, for local VCN peering
Default DHCP Options for *servicename*-*vcnname*	default set of Dynamic Host Configuration Protocol (DHCP) options for a new VCN
servicename-dhcpOptions	copy of the default DHCP options in the WebLogic VCN
servicename-wls-dns-dhcp-option	custom DNS routing in the WebLogic VCN, for local VCN peering
servicename-dbsystem-dns-dhcp-option	custom DNS routing in the database VCN, for local VCN peering

Load Balancer

If you chose to create a load balancer for your domain, it is accessible from a single IP address and it distributes traffic across the managed servers in the domain.

The name of the load balancer resource has the following format:

`servicename-lb`

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

The backend resource (which configures the load balancing policy) is identified by the name:

`servicename-lb-backendset`

The default listener is named `https` and it handles traffic on port 443. Attached to the listener are the following:

- The rule set created with the name `SSLHeaders`. The rule set has the header rules `WL-Proxy-SSL` (value is `true`) and `is_ssl` (value is `ssl`).
- The certificate `demo_cert`.

  Oracle recommends you add your own SSL certificate.

  See Managing SSL Certificates in the Oracle Cloud Infrastructure documentation and Add a Certificate to the Load Balancer.

Identity Resources for Dynamic Group and Root Policy

Oracle WebLogic Server for OCI creates a dynamic group and a single policy for your domain if the **OCI Policies** check box remains selected during stack creation.

The dynamic group and root-level (tenancy) policy allow compute instances in the domain to access:

- Launch compute instances and manage block storage volumes.
- Keys and secrets in Oracle Cloud Infrastructure Vault
- Load balancer resources
- The database wallet if you're using an Oracle Autonomous Database to contain the required infrastructure schemas for a JRF-enabled domain
- The database if you're using Oracle Cloud Infrastructure Database (DB System) to contain the required infrastructure schemas for a JRF-enabled domain

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

- `servicename-wlsc-principal-group` (dynamic group)
- `servicename-oci-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 `service` policy has the following statements:

- `Allow dynamic-group` *`servicename`*`-wlsc-principal-group to read secret-bundles in tenancy` where `target.secret.id=<`*`OCID of the secret`*`>`.

  The OCID of the secret can be the Administrator password, the Database password, Autonomous Database (ATP) password, the Application Database password, and the IDCS client secret password.

- `Allow dynamic-group` *`servicename`*`-wlsc-instance-principal-group to manage virtual-network-family in compartment` where `target.vcn.id=<`*`OCID of the existing_VCN_ID`*`>`.

  The following policies grants the OS Management service:

- `Allow dynamic-group` *`servicename`*`-wlsc-instance-principal-group to read instance-family in tenancy`
- `Allow dynamic-group` *`servicename`*`-wlsc-instance-principal-group to use osms-managed-instances in compartment`

  The following policy is created if you provision a load balancer:

- `Allow dynamic-group` *`servicename`*`-wlsc-instance-principal-group to use load balancers in compartment`

  The following policy is created if you provision a JRF stack using an autonomous database:

- `Allow dynamic-group` *`servicename`*`-wlsc-principal-group to use autonomous-transaction-processing-family in tenancy`

Identity Resources for Oracle Identity Cloud Service

If you configure your domain to use Oracle Identity Cloud Service for authentication, Oracle WebLogic Server for OCI provisions additional resources in Oracle Identity Cloud Service to support the domain.

These resources are not components of the stack, and so they are not visible in Resource Manager. In addition, they are not deleted automatically when you destroy the stack.

The names of the Oracle Identity Cloud Service resources have the following formats:

- *`servicename`*`_confidential_idcs_app_`*`timestamp`* - Confidential Application
- *`servicename`*`_enterprise_idcs_app_`*`timestamp`* - Enterprise Application
- *`servicename`*`_app_gateway_`*`timestamp`* - App Gateway

Where:

- *`servicename`* is the resource name prefix you provided during stack creation.
- *`timestamp`* is the date and time on which the stack was created. For example, `2019-09-24T21:46:21.288662`

3 Manage a Domain

Learn how to access the administration console for an Oracle WebLogic Server for OCI domain, access the sample application, secure the domain, and delete the domain when you no longer need it.

Topics:

- About Managing a Domain
- About the Default Ports
- About the Security Checkup Tool
- Access the WebLogic Console
- Access the Fusion Middleware Control Console
- Access the Sample Application
- Start and Stop a Domain
- Scale a Stack
- Add IDCS after Creating a Domain
- Back Up and Restore a Domain
- Add a Certificate to the Load Balancer
- Secure a Domain Using Identity Cloud Service
- Secure Web Services Using Identity Cloud Service
- Integrate OPSS User and Group APIs with Identity Cloud Service
- Upgrade the Oracle Identity Cloud Service App Gateway Version
- Configure Session Persistence
- Update the Password Secret OCID and Policy
- Manage Data Sources
- Connect to a Domain Using Oracle JDeveloper
- Delete a Stack

About Managing a Domain

Learn about managing an Oracle WebLogic Server domain after creating it with Oracle WebLogic Server for OCI.

In general, you configure, manage, and maintain an Oracle WebLogic Server for OCI domain just like an on-premise domain. For example, to deploy an application:

- Roadmap for Deploying Applications in WebLogic Server (14.1.1.0)
- Roadmap for Deploying Applications in WebLogic Server (12.2.1.4)

This chapter provides additional help, best practices, and tools for some WebLogic Server tasks:

- Access the WebLogic Console
- Access the Fusion Middleware Control Console
- Access the Sample Application
- Start and Stop a Domain
- Scale a Stack
- Add IDCS after Creating a Domain
- Back Up and Restore a Domain
- Add a Certificate to the Load Balancer
- Secure a Domain Using Identity Cloud Service
- Secure Web Services Using Identity Cloud Service
- Integrate OPSS User and Group APIs with Identity Cloud Service
- Configure Session Persistence
- Manage Data Sources
- Connect to a Domain Using Oracle JDeveloper
- Delete a Stack

About the Default Ports

Oracle WebLogic Server for OCI configures listen ports and network channels in a new domain.

Each server has both internal and external ports. Internal ports are not accessible from outside of Oracle Cloud. External ports support HTTP and HTTPS only. They do not support the T3 and T3S protocols, and they do not support HTTP tunneling. Oracle does not recommend enabling the T3 protocol or HTTP tunneling on network channels that are accessible from outside of Oracle Cloud.

If you chose to use a public subnet for the domain's compute instances, then the instances are assigned public IP addresses, and the external server ports are accessible from the Internet. If you used a private subnet, the external server ports are accessible only from the Oracle Cloud network, or from your on-premises data center over a VPN network.

Administration Server

Channel	Port, Protocol	Purpose
Default listen port	9071, T3	• Internal domain communication • Use administration clients like the WebLogic Scripting Tool (WLST) within this virtual cloud network (VCN) • Access T3 applications within this VCN

Channel	Port, Protocol	Purpose
Default SSL listen port	9072, T3S	• Internal domain communication • Use administration tools within this VCN • Access T3S applications within this VCN
`ExternAdmin`	7001, HTTP	• Access the administration console • Access web applications
`SecuredExternAdmin`	7002, HTTPS	• Access the administration console • Access web applications

Managed Servers

Channel	Port, Protocol	Purpose
Default listen port	9073, T3	• Internal domain communication • Use administration tools within this VCN • Access T3 applications within this VCN
Default SSL listen port	9074, T3S	• Internal domain communication • Use administration tools within this VCN • Access T3S applications within this VCN
`ExternAdmin`	7003, HTTP	Access web applications
`SecuredExternAdmin`	7004, HTTPS	Access web applications

About the Security Checkup Tool

Oracle WebLogic Server Administration console includes a security checkup tool that displays security check warnings. These security check warnings are displayed for Oracle WebLogic Server for OCI instances that are created using WebLogic Server versions 12.2.1.4 and 14.1.1.0.

In case of Oracle WebLogic Server for OCI 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 OCI instances created after October 20, 2021, or the instances on which the October PSUs are applied.
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Disable the anonymous RMI T3 and IIOP requests in the WebLogic Server Administration Console as soon as possible unless your deployment requires anonymous T3 or IIOP (not typical). See Disable Remote Anonymous RMI T3 and IIOP Requests.

> **Note:**
>
> For existing Oracle WebLogic Server for OCI instances created before release 21.3.2 (August 17, 2021), you see the SSL host name verification warnings. 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 OCI 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.

If you want to perform anonymous RMI requests, you must disable the java properties. Go to the `nodemanager.properties` file located under `DOMAIN_HOME/nodemanager` and remove the `weblogic.startup.Arguments` property.

Disable Remote Anonymous RMI T3 and IIOP Requests

To disable the remote anonymous RMI T3 and IIOP requests 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 the domain name, and then select the **Security** tab.
3. Expand **Advanced** and deselect **Remote anonymous RMI access via IIOP** and **Remote anonymous RMI access via T3**.

After saving the changes, return to **Change Center** and click **Activate Changes**.

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.

Access the WebLogic Console

Use the WebLogic Server Administration Console to access a domain in Oracle WebLogic Server for OCI.

> **Note:**
>
> 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.

- Access the WebLogic Console in a Public Subnet
- Access the WebLogic Console in a Private Subnet
- Access the WebLogic Console Through a Load Balancer

Access the WebLogic Console in a Public Subnet

Oracle WebLogic Server compute instances assigned to a public subnet are accessible from the public Internet.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the domain instance that has the Administration Server node.

   The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
5. Copy the public IP address value.
6. In a browser, specify the URL of the WebLogic Server Administration Console in the following format, using the public IP address and appropriate port:

   ```
   https://IP-address:port/console
   ```

   The default SSL port is `7002`, unless it was changed during stack creation.

   For example:

   ```
   https://192.0.2.1:7002/console
   ```
7. When prompted, enter the WebLogic Server administrator user name and password for this domain.

Access the WebLogic Console in a Private Subnet

Oracle WebLogic Server compute instances assigned to a private subnet are not accessible from the public Internet.

To access the WebLogic Server Administration Console to administer such instances, you can use:

- Bastion instance that's created on a public subnet and dynamic port forwarding with a secure shell (SSH) utility. See Access by Using the Bastion Instance.
- Bastion service and dynamic port forwarding with a secure shell (SSH) utility. See Access by Using the Bastion Service.

By opening an SSH tunnel with dynamic port forwarding, the SSH client becomes a Socket Secure (SOCKS) proxy listening on the port you specify. All traffic that routes to the proxy port is forwarded to its destination through the proxy server. Then when you configure your browser to use a SOCKS proxy, you can access a private domain's administration console.

Access by Using the Bastion Instance

To access the WebLogic console in a private subnet by using the bastion instance, complete the following steps;

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the domain instance that has the Administration Server node.

   The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
5. Copy the private IP address value.
6. Return to the Compute Instances page.
7. Click the name of the bastion instance that's associated with the domain.

   The domain's bastion instance is identified by *servicename*`-bastion-instance`. For example: `abcde7xy-bastion-instance`
8. Copy the public IP address value.
9. From your computer, open an SSH tunnel to an unused port on the bastion node as the `opc` user. For example, you can use port `1088` for SOCKS proxy. It can be any other unused port.

   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 -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.

10. 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.
11. Specify the URL of the WebLogic Server Administration Console in the following format, using the private IP address:

    ```
    http://private_ip_address:port/console
    ```

    The default port is `7001`, unless it was changed during stack creation.

    For example:

    ```
    http://192.0.2.254:7001/console
    ```

12. When prompted, enter the WebLogic Server administrator user name and password.

Access by Using the Bastion Service

To access the WebLogic console in a private subnet by using the bastion service, complete the following steps:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. If you have already created a Bastion service and a session in the required VCN, navigate to the Bastion and the Session, and then continue from step 18.
3. Click the navigation menu ☰, select **Identity & Security**. Under the **Identity & Security** group, click **Bastion**.
4. From the **Compartment** dropdown, select the compartment in which your domain is created.
5. Click **Create bastion**.
6. Enter a name for the bastion.
7. Under **Configure networking**, select the **Target virtual cloud network** of the target resource that you intend to connect to by using sessions hosted on this bastion.
8. Select the **Target subnet**.

    The subnet must either be the same as the target resource's subnet or it must be a subnet from which the target resource's subnet accepts network traffic.

9. In **CIDR block allowlist**, add one or more address ranges in CIDR notation that you want to allow to connect to sessions hosted by this bastion.

    Enter a CIDR block, and then either click the value or press **Enter** to add the value to the list. The maximum allowed number of CIDR blocks is 20.

10. From the list of Bastion, click the name of the Bastion you created.
11. Click **Create session**.
12. From the **Session type** dropdown, select **SSH port forwarding session**.
13. Under **Connect to the target host** option, select **Instance name**.
14. From the Compute instance, select the domain instance that has the Administration Server node.

    The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
15. In the **Port** field, enter the WebLogic administration server's listener port number, where the administration console is accessible. By default, the port number `7002`.
16. Under **Add SSH Key**, provide the public key file of the SSH key pair that you want to use for the session.
17. Click **Create Session**.
18. Click the **Actions** icon for the session you created, and select **View SSH command**.
19. In the **View SSH command** window, click **Copy** to copy the SSH command.
20. Paste the SSH command in a text editor.
21. In the SSH command, replace <*privateKey*> with the path to the private key that corresponds to the public key that you specified when you created the domain.
22. In the SSH command, replace the <*localPort*> with the preferred local port number.
23. Copy the updated SSH command.
24. From your computer, open a SSH utility.
25. Paste the SSH command and then press **Enter**.

    If prompted to continue connecting, type `Yes` and press **Enter**.
26. Specify the URL of the WebLogic Server Administration Console in the following format:

    ```
    https://localhost:<local-port>/console
    ```

    The default port is `7002`, unless it was changed during stack creation.

    For example:

    ```
    https://localhost:7002/console
    ```
27. When prompted, enter the WebLogic Server administrator user name and password.

Access the WebLogic Console Through a Load Balancer

Oracle WebLogic Server compute instances assigned to a private subnet are not accessible from the public Internet. So, if you have created the load balancer manually, you can access the administration console via load balancer's public IP address.

However, it is recommended to access the WebLogic Server Administration Console using the bastion instance that's created on a public subnet and dynamic port forwarding with a secure shell (SSH) utility. See Access the WebLogic Console in a Private Subnet.

> **Note:**
>
> If you created the load balancer in a different network than the Oracle WebLogic Server domain network with nonoverlapping CIDRs, you must create a Local Peering Gateway on both the domain network and load balancer network and add a route table for the subnet for WebLogic domain and the load balancer subnet to use the respective Local Peering Gateway.

For SSL connections that terminate at the load balancer, you must configure the load balancer to include the `WL-Proxy-SSL` header in the rule set and enable the WebLogic Proxy Plugin on the cluster and the administration server.

To access the WebLogic administration console through the load balancer:

1. Configure the load balancer to include the `WL-Proxy-SSL` header in the rule set. See Create an HTTPS Listener for the Load Balancer.
2. Enable WebLogic Proxy Plugin for the administration server by adding the following line in domain `config.xml` under `AdminServer` config:

   ```
   <weblogic-plugin-enabled>true</weblogic-plugin-enabled>
   ```

3. Restart the domain using the following script:

   ```
   /opt/scripts/restart_domain.sh -o restart
   ```

4. Create a backend set for the load balancer.
   a. Sign in to the Oracle Cloud Infrastructure Console.
   b. From the navigation menu, click **Networking**, and then click **Load Balancers**.
   c. Click the name of the Compartment that contains the load balancer you want to modify, and then click the load balancer's name.
   d. Under **Resources** menu, click **Backend Sets**,

      The list of Backend Sets is displayed.
   e. Click **Create Backend Set**.
   f. Specify the backend set details.
      i. Enter a **Name** for the backend set.
      ii. Select the load balancer policy for the backend set.
      For policies, see Load Balancing Policies.
      iii. Under **Health Check**, specify the parameters to confirm the health of backend servers.
         - **Protocol**: *HTTP*
         - **Port**: *7001*
         - **URL Path**: Private IP address of the admin VM WebLogic admin VM

         Optionally, you can specify the other test parameters to confirm the health of backend servers, as required.
   g. Click **Create Backend Set**.

5. Specify the backend server for the backend set.
   a. Click the name of the backend set that you created in step 4.
   b. Under **Resources** menu, click **Backends**.
   c. Click **Add Backends**.
   d. Select **IP Addresses**.
   e. Enter the private IP address of a backend server you want to add to the backend set.
   f. For **Port**, enter `7001`.
   g. Click **Add**.
6. In the Oracle Cloud Infrastructure Console., from the navigation menu, click **Compute**. Under the **Compute** group, click **Instances**.
7. From the **Compartment** dropdown, select the compartment in which your domain is created.
8. Click the name of the domain instance that has the Administration Server node.

   The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
9. Copy the public IP address value.
10. In a browser, specify the URL of the WebLogic Server Administration Console in the following format, using the public IP address and port *7001*:

    ```
    https://IP-address:7001/console
    ```
11. When prompted, enter the WebLogic Server administrator user name and password.

Access the Fusion Middleware Control Console

If your Oracle WebLogic Server for OCI domain includes the Java Required Files (JRF) components, you can use the Fusion Middleware Control to access and manage the domain and its Fusion Middleware components.

Topics:

- Access the Fusion Middleware Control Console in a Public Subnet
- Access the Fusion Middleware Control Console in a Private Subnet

Access the Fusion Middleware Control Console in a Public Subnet

Oracle WebLogic Server compute instances assigned to a public subnet are accessible from the public Internet.

To access the Fusion Middleware Control:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.

4. Click the name of the domain instance that has the Administration Server node.

   The instance with the Adminstration Server node has `wls-0` appended to the name. For example: `abcde1xy-wls-0`

5. Copy the public IP address value.
6. In a browser, specify the URL of the Fusion Middleware Control in the following format, using the public IP address and appropriate port:

   ```
   https://IP-address:port/em
   ```

   The default SSL port is `7002`, unless it was changed during stack creation.

   For example:

   ```
   https://192.0.2.3:7002/em
   ```

7. When prompted, enter the WebLogic Server administrator user name and password for this domain.

Access the Fusion Middleware Control Console in a Private Subnet

Oracle WebLogic Server compute instances assigned to a private subnet are not accessible from the public Internet.

To access the Fusion Middleware Control Console for a private domain, you can use the bastion instance that's created on a public subnet and dynamic port forwarding with a secure shell (SSH) utility.

By opening an SSH tunnel with dynamic port forwarding, the SSH client becomes a Socket Secure (SOCKS) proxy listening on the port you specify. All traffic that routes to the proxy port is forwarded to its destination through the proxy server. Then when you configure your browser to use a SOCKS proxy, you can access a private domain and its Fusion Middleware components.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the domain instance that has the Administration Server node.

   The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`

5. Copy the private IP address value.
6. Return to the Compute Instances page.
7. Click the name of the bastion instance that's associated with the domain.

   The domain's bastion instance is identified by *`servicename`*`-bastion-instance`. For example: `abcde7xy-bastion-instance`

8. Copy the public IP address value.

9. From your computer, open an SSH tunnel to an unused port on the bastion node as the `opc` user. For example, you can use port `1088` for SOCKS proxy. It can be any other unused port.

   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 -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.

10. 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.

11. Specify the URL of the Fusion Middleware Control Console in the following format, using the private IP address:

    ```
    http://private_ip_address:port/em
    ```

    The default port is `7001`, unless it was changed during stack creation.

    For example:

    ```
    http://192.0.2.3:7001/em
    ```

12. When prompted, enter the WebLogic Server administrator user name and password for this domain.

Access the Sample Application

By default when you create an Oracle WebLogic Server for OCI domain, sample applications are deployed automatically.

How to Access the Application

To access the `sample-app` application in a browser, enter a URL that's similar to the following:

```
https://IP_address:port/sample-app
```

The IP address and port you'll use depends on whether your domain is in a public or private subnet, and whether a load balancer is configured. See:

- Access the Sample Application Through a Public Load Balancer
- Access the Sample Application Through a Private Load Balancer
- Access the Sample Application Without a Load Balancer
- Access the Sample Application in a Private Subnet

If your domain is configured to use Oracle Identity Cloud Service, then a second application is deployed named `idcs-sample-app`. See Access the Sample Application Using Identity Cloud Service.

What the Application Does

From the `sample-app` application, you can find documentation, tutorials, and other resources for Oracle WebLogic Server for OCI in the Oracle Help Center.

Use the `idcs-sample-app` application to test the integration with Oracle Identity Cloud Service. After signing in, the application displays information about the current user.

How to Manage the Application

You can verify that the sample applications are deployed and running by viewing the Deployments table in the WebLogic Server Administration Console. From the Deployments table, you can stop, start, and undeploy the applications.

Access the Sample Application Through a Public Load Balancer

If your Oracle WebLogic Server for OCI domain is associated with a public load balancer, use the public IP address of the load balancer to access the sample application.

This procedure applies to a domain created in a public or private subnet.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Networking**, and then click **Load Balancers**.
3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack is initially created, this is the compartment that contains the compute instances and network resources for the domain, or this is the network compartment that has only the network resources for the domain.
4. Click the name of the load balancer instance that's associated with your domain instance.

   The load balancer instance has `-lb` appended to the domain name. For example: `abcde1xy-lb`
5. Copy the public IP address value.
6. In a browser, specify the URL of the sample application. By default, the load balancer listens for HTTPS requests on port `443`.

   ```
   https://IP-address/sample-app/
   ```

Access the Sample Application Through a Private Load Balancer

If your Oracle WebLogic Server for OCI domain is in a private subnet and is associated with a private load balancer, it is not accessible from the public Internet.

To access the sample application, you can use the bastion instance that's created on a public subnet and dynamic port forwarding with a secure shell (SSH) utility.

By opening an SSH tunnel with dynamic port forwarding, the SSH client becomes a Socket Secure (SOCKS) proxy listening on the port you specify. All traffic that routes to the proxy port is forwarded to its destination through the proxy server. Then when you configure your browser to use a SOCKS proxy, you can access the load balancer's private IP address.

Alternatively, you can configure a virtual private network (VPN) between your VCN and your on-premise data center. See VPN Connect in the Oracle Cloud Infrastructure documentation.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Networking**, and then click **Load Balancers**.
3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack is initially created, this is the compartment that contains the compute instances and network resources for the domain, or this is the network compartment that has only the network resources for the domain.
4. Click the name of the load balancer instance that's associated with your domain instance.

   The load balancer instance has `-lb` appended to the domain name. For example: `abcde1xy-lb`
5. Copy the IP address value.
6. From the navigation menu, click **Compute**, and then click **Instances**.
7. Select the **Compartment** in which the compute instances for your domain were created.
8. Click the name of the bastion instance that's associated with the domain.

   The domain's bastion instance is identified by *`servicename`*`-bastion-instance`. For example: `abcde1xy-bastion-instance`
9. Copy the public IP address value.
10. From your computer, open an SSH tunnel to an unused port on the bastion node as the `opc` user. For SOCKS proxy, you can use port `1088`.

    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 -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.
11. 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.
12. Specify the URL of the sample application in the following format, using the private IP address of the load balancer:

    ```
    https://private_ip_address/sample-app
    ```

By default, the load balancer listens for HTTPS requests on port `443`.

For example:

```
https://192.0.2.254/sample-app
```

Access the Sample Application Without a Load Balancer

If your Oracle WebLogic Server for OCI domain is not associated with a load balancer, use the IP address of a Managed Server node to access the sample application.

> **Note:**
>
> This procedure does not apply to a domain created in a private subnet. See Access the Sample Application in a Private Subnet.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the compute instance that has the Administration Server and the first Managed Server.

   The instance has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
5. Copy the public IP address value.
6. In a browser, specify the URL of the sample application using the public IP address and the Managed Server port, in one of the following formats:

   ```
   https://IP-address:7004/sample-app/
   ```

   ```
   http://IP-address:7003/sample-app/
   ```

   Unless the ports were changed during stack creation, the default Managed Server ports are `7004` (SSL) and `7003`.

   For example:

   ```
   https://198.51.100.1:7004/sample-app/
   ```

   ```
   http://198.51.100.1:7003/sample-app/
   ```

Access the Sample Application in a Private Subnet

Oracle WebLogic Server compute instances assigned to a private subnet are not accessible from the public Internet.

If your private domain is configured with a load balancer, see:

- Access the Sample Application Through a Public Load Balancer
- Access the Sample Application Through a Private Load Balancer

To access the sample application deployed to a private domain without a load balancer, you can use the bastion instance that's created on a public subnet and dynamic port forwarding with a secure shell (SSH) utility.

By opening an SSH tunnel with dynamic port forwarding, the SSH client becomes a Socket Secure (SOCKS) proxy listening on the port you specify. All traffic that routes to the proxy port is forwarded to its destination through the proxy server. Then when you configure your browser to use a SOCKS proxy, you can access the domain's private IP address.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the compute instance that has the Administration Server and the first Managed Server.

   The instance has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
5. Copy the private IP address value.
6. Return to the Compute Instances page.
7. Click the name of the bastion instance that's associated with the domain.

   The domain's bastion instance is identified by `*servicename*-bastion-instance`. For example: `abcde7xy-bastion-instance`
8. Copy the public IP address value.
9. From your computer, open an SSH tunnel to an unused port on the bastion node as the `opc` user. For SOCKS proxy, you can use port `1088`.

   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 -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.

10. 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.
11. Specify the URL of the sample application in the following format, using the private IP address:

    ```
    http://private_ip_address:port/sample-app
    ```

    The default Managed Server port is `7003`, unless it was changed during stack creation.

    For example:

    ```
    http://192.0.2.254:7003/sample-app
    ```

Access the Sample Application Using Identity Cloud Service

If your Oracle WebLogic Server for OCI domain is integrated with Oracle Identity Cloud Service, then a second sample application is automatically deployed so that you can test this integration.

The application is protected by Oracle Identity Cloud Service, so you must sign in as a valid user in Oracle Identity Cloud Service.

A domain that uses Oracle Identity Cloud Service for authentication always includes a load balancer. This procedure applies to a domain created in a public or private subnet.

Part of the application requires the user to be a member of a group named `SampleAppAdmin`. Create this group in Oracle Identity Cloud Service, and add at least one user to the group. See Create Groups in Oracle Cloud Infrastructure documentation.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Networking**, then click **Load Balancers**.
3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack is initially created, this is the compartment that contains the compute instances and network resources for the domain, or this is the network compartment that has only the network resources for the domain.
4. Click the name of the load balancer instance that's associated with your domain instance.

   The load balancer instance has `-lb` appended to the domain name. For example: `abcde1xy-lb`
5. Copy the public IP address value.
6. Sign out from the Oracle Cloud Infrastructure Console.
7. In a browser, specify the URL of the sample application. By default, the load balancer listens for HTTPS requests on port `443`.

   ```
   https://IP-address/__protected/idcs-sample-app
   ```

The URL includes two underscore characters.

For example:

```
https://192.0.2.254/__protected/idcs-sample-app
```

8. Sign in to the Oracle Identity Cloud Service as a user who is a member of the `SampleAppAdmin` group.
9. Click **Protected page**.

To protect other applications in the domain, see Secure a Domain Using Identity Cloud Service.

Start and Stop a Domain

Oracle WebLogic Server for OCI provides utilities to manage the server processes in your domain.

With these utilities you can stop, start, or restart (stop and then start) the WebLogic Server and Node Manager processes in your domain.

1. Identify the IP address of the node in your domain.

   The name of the node is *`servicename`*`-wls-`*`n`*, where `servicename` is the resource name prefix you provided during stack creation. The Administration Server runs on the first node, *`servicename`*`-wls-0`

   - If your domain is on a public subnet, then use the public IP address of the compute instance.
   - If your domain is on a private subnet, then use the public IP address of the bastion and the private IP address of the compute instance.

2. Open an SSH connection to the node as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

   Or,

   ```
   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. Change to the `oracle` user.

   ```
   sudo su - oracle
   ```

4. Execute the `restart_domain.sh` script.

   ```
   /opt/scripts/restart_domain.sh -o [start|stop|stopMS|restart]
   ```

   - `start` - start the Node Manager and all servers on this node
   - `stop` - stop the Node Manager and all servers on this node
   - `stopMS` - stop only the Managed Server process on this node

- `restart` - stop and then start the Node Manager and all servers on this node

For the `stop` operation, the Administration Server and Node Manager must be running.

To start, stop, or restart all servers in the domain, run the following command:

```
/opt/scripts/restart_domain.sh -o [start|stop|restart] -servers all
```

> **Note:**
>
> To stop all servers in your domain, the Administration Server must be running, and to start all servers in your domain, the Node Manager in the virtual machine of the Administration server must be running.

If you modified certain settings after creating the domain, like the administrator password or port number, then you must provide additional parameters to `restart_domain.sh`:

- `-u` - User name for the domain administrator
- `-p` - Password for the domain administrator
- `--adminhost` - Hostname of the administration server
- `--port` - Port number of the administration server
- `--domain-name` - Name of the domain
- `--domain-home` - Home directory for the domain
- `--admin-servername` - Name of the administration server

Add IDCS after Creating a Domain

After you create a domain, you can add Oracle Identity Cloud Service (IDCS) to your Oracle WebLogic Server for OCI instance.

> **Note:**
>
> This procedure applies to domains that are created from November 2021 (Release 21.4.2) onwards. For previous releases, contact *Support*.

Prerequisites:

- Create a confidential application in IDCS to use IDCS for authentication in the domain. You will need the client ID and client secret for this confidential application. See Create a Confidential Application.
- An OCI secret with the IDCS client secret value in the tenancy. Create Secrets for Passwords. Copy the Secret OCID.

- At the root compartment level, create an OCI policy with the following policy statement:

  ```
  Allow dynamic-group <service-prefix>-wlsc-principal-group to read secret-bundles in tenancy where target.secret.id ='<secret-ocid>'
  ```

  Where, *<secret-ocid>* is the OCI secret that you obtained in the previous step.
- Add a Load Balancer, if not already configured. See Add a Load Balancer.

Complete the following steps to add IDCS to your domain:

1. Create a JSON file that contains the following information:

   ```
   {
     "is_idcs_selected" : "true",
     "idcs_host" : "<Domain name to access IDCS> (typically, identity.oraclecloud.com)",
     "idcs_port" : "443",
     "idcs_tenant" : "<IDCS Instance ID> (format is idcs-<GUID>)",
     "idcs_client_id" : "<Client ID of the confidential application in IDCS>",
     "idcs_client_secret_ocid" : "<Client secret of the confidential application>",
     "idcs_cloudgate_port" : "9999",
     "idcs_cloudgate_docker_image_tar" : "/u01/zips/APP-GATEWAY/21.2.2/appgateway-21.2.2-2105050509.tar.gz",
     "load_balancer_id" : "<OCID of the Load Balancer>",
     "lbip" : "<IP address of the Load Balancer>"
   }
   ```

   In the JSON file, use the client ID and client secret that you created for the confidential application. See Prerequisites.
2. Log in as a root user to the Administration server.
3. Save the JSON file to an accessible location.
4. Run the following command to verify if a Docker Engine is available and displays the status as loaded:

   ```
   systemctl status docker
   ```

5. Run the following command:

   ```
   python3 /opt/scripts/idcs/configure_idcs.py <json-file-location>
   ```

6. Log in to each of the nodes and complete step 3 through step 5:
7. If the domain is a JRF domain, then add the OPSS SCIM template to the domain.
   a. Change to the oracle user.

      ```
      sudo su - oracle
      ```

b. Run the following commands only on the VM where the Administrator server is running:

```
/u01/app/oracle/middleware/oracle_common/common/bin/wlst.sh
readDomain("/u01/data/domains/<domain_name>")
addTemplate("/u01/app/oracle/middleware/oracle_common/common/
templates/wls/oracle.opss_scim_template.jar")
updateDomain()
closeDomain()
exit()
```

8. Restart the Administration server

9. If the domain is a JRF domain and you added the OPSS SCIM template, then restart the managed servers.

Back Up and Restore a Domain

Use the backup and restore capabilities of Oracle Cloud Infrastructure Block Volumes to return your Oracle WebLogic Server for OCI domain to a specific state, or to recover from a failure.

Topics:

- About Volume Backups
- About Database Backups
- Create a Backup
- Restore a Backup
- Repair a Boot Volume

About Volume Backups

The backup feature of Oracle Cloud Infrastructure Block Volumes lets you make a point-in-time backup of data on a volume. You can create a new block volume from a backup, and then attach the new volume to a domain's compute instance.

Each Oracle WebLogic Server for OCI compute instance is comprised of three volumes:

- *Boot* volume - This volume contains the Linux operating system. To restore a boot volume backup, you must create a new compute instance.

  > **Note:**
  >
  > Restoring from a boot volume is not recommended. It must be performed only as a last option.

- *Block* volume (*Domain* volume) - This volume has the contents of the `/u01/data` folder, including the domain configuration.

- *Block* volume (*Middleware* volume) - This volume has the contents of the `/u01/app` folder. The *Middleware* volume is introduced in instances created after release 20.3.3 (September 29, 2020).

Oracle recommends that you create regular backups of all boot volumes and block volumes for your domain. You can create backups manually, or you can configure policies that automatically create backups based on a specific schedule, and retain the backups for a specific period of time.

The first backup of a volume is a *full* backup, which includes all changes since the volume was created. After the first backup, you can choose to create additional full backups, or to create *incremental* backups, which include only the changes since the last backup. Both backup types enable you to restore the full volume contents to the point-in-time snapshot of the volume when the backup was taken.

Backups are encrypted and stored in Oracle Cloud Infrastructure Object Storage. Therefore, backups can be restored as new volumes to any availability domain within the same region. You can also copy volume backups across regions for disaster recovery purposes.

You must restart WebLogic Server (or restart the entire compute instance) when you restore a block volume backup. To avoid downtime and ensure your applications remain available to users, Oracle recommends that you create a cluster with multiple compute instances, and that you restore the block volume for one compute instance at a time. This approach is also called a *rolling recovery*.

When you restore a block volume backup for the first compute instance, the domain's administration server will be temporarily unavailable. However, your applications do not depend on the administration server and will not be affected.

See Overview of Block Volume Backups in the Oracle Cloud Infrastructure documentation.

About Database Backups

If your Oracle WebLogic Server for OCI domain includes the Java Required Files (JRF) components, then Oracle recommends that you back up the database containing the JRF schemas.

When you restore the block storage volumes for a domain from a backup, then you can also restore the database from a backup that was taken at the same time.

The backup and restoration procedures vary depending on type of database: Oracle Autonomous Database or Oracle Cloud Infrastructure Database.

Database Type	Details
Oracle Autonomous Database	• Oracle automatically backs up your autonomous databases and retains these backups for 60 days. Automatic backups are weekly full backups and daily incremental backups. • Backing Up an Autonomous Database Manually • Restoring an Autonomous Database
Oracle Cloud Infrastructure Database	• Backing Up a Database • Recovering a Database

Create a Backup

Create a backup of all block volumes for your Oracle WebLogic Server for OCI domain.

You can either take an immediate backup of the volumes, or you can configure a backup policy that takes a future backup according to the policy's schedule.

> **Note:**
>
> Creating a new instance by restoring from a boot volume is not recommended. It must be preformed only as a last option. To backup a boot volume, see Backing Up a Volume.

To backup a block volume, complete the following steps:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. Select the **Compartment** in which your domain was created.
4. Click the name of the first compute instance for your domain.

   The instance has `wls-0` appended to the name. For example: `mydomain-wls-0`
5. Scroll down and under **Resources**, select **Attached Block Volumes**.
6. Create a manual backup of the block volume, or assign it a backup policy.
   - Create a manual backup. See Backing Up a Volume.
   - Assign a backup policy. See Policy-Based Backups.
7. Repeat from step 2 through step 6 for the remaining compute instances in your domain.

Restore a Backup

Restore the volumes for your Oracle WebLogic Server for OCI domain from a backup.

There are two main scenarios where you would restore a backup:

- Restore to a previous working domain
- Recover from an Operating System failure

Restore to a Previous Domain

If your domain configuration is corrupted, accidentally destroyed, or the domain is not behaving as expected after a change to the binaries, then you can restore the block volumes to a previous working backup.

> **Tip:**
>
> If you have a corrupted or missing domain, follow the restoration steps in this section using the data volume. Format: `<my_domain_name>-data-block-node`.
> If you have patched or updated the WebLogic binaries and had unexpected results, follow the restoration steps in this section using the `mw` block volume. Format: `<my_domain_name>-mw-block-node` .

To avoid downtime, Oracle recommends that you restore the block volume for one compute instance in your domain at a time. The remaining compute instances can continue to process client requests.

When you attach a block volume to a virtual machine (VM) compute instance, you have two options for attachment type: iSCSI or paravirtualized. Paravirtualized attachments greatly simplify the process of configuring your block storage but IOPS performance is greater for iSCSI attachments. Bare metal compute instances must use iSCSI attachments. See Overview of Block Volume.

To restore a block volume, complete the following steps:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. Select the **Compartment** in which your domain was created.
4. Click the name of the first compute instance for your domain.

   The instance has `wls-0` appended to the name. For example: `mydomain-wls-0`
5. Click **Attached Block Volumes**.
6. Identify the name of the block volume for this compute instance, and the **Attachment Type**: `paravirtualized` or `iscsi`.
7. Create a new block volume from the backup. See Restoring a Backup to a New Volume.

   For example, `mydomain-data-block-0-new` for a domain volume or `mydomain-mw-block-0-new` for the binaries volume.
8. Shut down the WebLogic Server processes on the compute instance if any are running.
   a. Connect to the compute instance as the `opc` user with a secure shell (SSH).
   b. Switch to the `oracle` user.

      ```
      sudo su - oracle
      ```

   c. Identify the process IDs of all server and Node Manager processes.

```
ps -ef | grep weblogic.
oracle <processID> ... weblogic.NodeManager
oracle <processID> ... weblogic.Server
oracle <processID> ... weblogic.Server
```

   d. Kill the Node Manager process ID, and then kill the server process IDs.

```
kill -9 <processID>
```

9. Detach the current block volume from the compute instance's file system.
   a. Connect to the compute instance as the `opc` user with SSH.
   b. Unmount `/u01/data`.

```
sudo umount /u01/data
```

10. Detach the current block volume from the compute instance. See Detaching a Volume.

    If the volume attachment type is iSCSI, then connect to the compute instance as the `opc` user with SSH, and follow the instructions in the **Detach Block Volume** dialog.

11. Attach the new block volume to the compute instance. See Attaching a Volume.

    Select the same attachment type as the original block volume: paravirtualized or iSCSI.

12. If the attachment type is iSCSI, then connect the new block volume to the compute instance's file system. See Connecting to a Volume.

    You can copy the required commands from the **iSCSI Commands and Information** dialog.

13. From the Oracle Cloud Infrastructure Console, click **Reboot** to reboot the compute instance.

    The WebLogic Server processes start automatically.

14. Repeat all of these steps for each of the remaining compute instances in your domain.

After you restore the domain, verify that you can access the WebLogic Server administration console and your applications. See:

- Access the WebLogic Console
- Access the Fusion Middleware Control Console
- Access the Sample Application

Recover from an Operating System Failure

If you encounter an Operating System (OS) failure, you have different options to recover your domain.

To recover your domain, complete the following:

1. Repair the boot volume. See Repair a Boot Volume.

Did this recover your domain?

- **Yes**: Skip the next steps.
- **No**: Go to next step.

2. Move the WebLogic Server instance. See Move a WebLogic Server Instance.

   Did this recover your domain?

   - **Yes**: Skip the next step.
   - **No**: Go to next step.

3. If step 1 or step 2 does not recover your domain, then as a last option complete this step.

   Delete the instance and then create an instance from a boot volume backup. See Restoring a Boot Volume.

> **Note:**
>
> If you restore a boot volume:
>
> - All the stack metadata is lost. So, no post-provisioning actions, like restart script is available.
> - If there are no other stack instances in the same compartment, then the new instance will not inherit the UCM subscription pricing.

Repair a Boot Volume

If the boot volume for one of your Oracle WebLogic Server for OCI compute instances becomes corrupted and will not boot, you can attach it to another compute instance to troubleshoot and repair it.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. Select the **Compartment** in which your domain was created.
4. Click the compute instance that you need to repair.
5. Identify the region and availability domain for the compute instance.
6. If the compute instance is running, click **Stop**.
7. Click **Boot Volume**.
8. Click the **Actions** icon for the boot volume, and then select **Detach**.
9. Identify a working Linux compute instance in the same region and availability domain, or create a new Linux compute instance.

   See Creating an Instance.
10. Access the new compute instance with SSH.
11. Use the `lsblk` command to identify the devices and partitions that are currently configured on the compute instance.

Example:

```
sudo lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0 46.6G  0 disk
...
```

12. Return to the Oracle Cloud Infrastructure Console.
13. From the Instances page, click the new compute instance.
14. Click **Attached Block Volumes**.
15. Click **Attach Block Volume**.
16. For **Block Volume**, select the boot volume that you need to repair.
17. Click **Attach**.
18. Return to the SSH session.
19. Run the `lsblk` command again, and identify the new device and partitions.

    Example:

    ```
    sudo lsblk
    NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
    sdb      8:16   0 46.6G  0 disk
    ¿¿sdb2   8:18   0    8G  0 part
    ¿¿sdb3   8:19   0 38.4G  0 part
    ¿¿sdb1   8:17   0  200M  0 part
    sda      8:0    0 46.6G  0 disk
    ...
    ```

20. Use the `fsck` command to perform a consistency check of each partition on the new device.

    ```
    sudo fsck -V /dev/<partition>
    ```

    Example:

    ```
    sudo fsck -V /dev/sdb1
    sudo fsck -V /dev/sdb2
    sudo fsck -V /dev/sdb3
    ```

21. When prompted, correct any errors found in the partitions.
22. Return to the Oracle Cloud Infrastructure Console.
23. Detach the repaired boot volume from the compute instance's block volumes. See Detaching a Volume.

    If the volume attachment type is iSCSI, then connect to the compute instance with SSH, and follow the instructions in the **Detach Block Volume** dialog.

24. Navigate to the original compute instance.
25. Attach the repaired boot volume to original compute instance. See Attaching a Volume.
26. Click **Start** to start the compute instance.

Add a Certificate to the Load Balancer

When you create a domain with a load balancer, Oracle WebLogic Server for OCI configures the load balancer to use Secure Socket Layer (SSL) and also adds a demonstration self-signed certificate. Oracle recommends you upload your own SSL certificate, and then associate the certificate with the HTTPS listener.

> **Note:**
>
> This procedure applies only to domains that were created after June 2020. For domains created before June 2020, see Configure SSL for a Domain.

You can use a custom, self-signed SSL certificate, or a certificate that you've obtained from a Certificate Authority (CA). For production WebLogic Server environments, Oracle recommends that you use a CA-issued SSL certificate, which reduces the chances of experiencing a man-in-the-middle attack.

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Load Balancers**.
3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack was initially created, this might be the same compartment that contains the compute instances for the domain.
4. Click the load balancer that was provisioned as part of your stack, `*prefix*-lb`.
5. Click **Certificates**.
6. Click **Add Certificate**.
7. Enter a name for your certificate.
8. Either upload the certificate file, or paste its contents into the text area.
9. If applicable, specify a CA certificate or a private key file.

   For example, if you are using a self-signed certificate, upload the corresponding private key file. See Managing SSL Certificates in the Oracle Cloud Infrastructure documentation.
10. Click **Add Certificate**, and then click **Close**.
11. After the certificate was successfully added, click **Listeners**.
12. Edit the `https` listener.
13. Select your new certificate.
14. Click **Save Changes**, and then click **Close**.

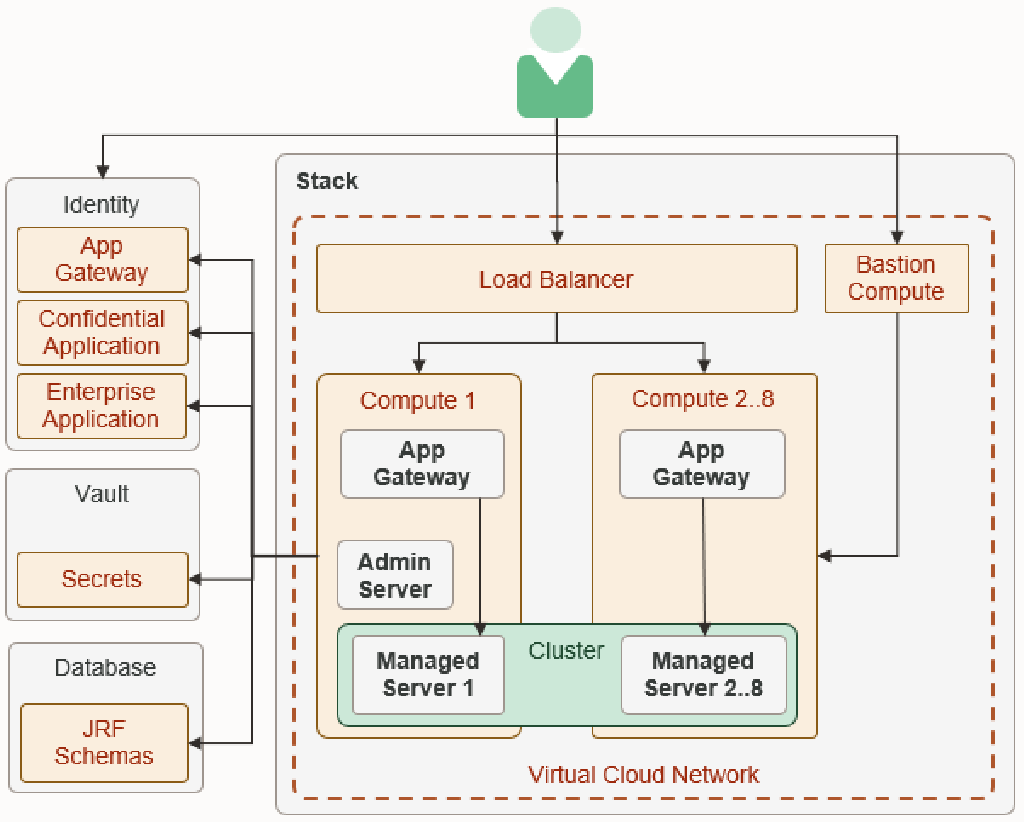
You cannot modify an existing load balancer certificate. You must add a new certificate, and then associate the listener with the new certificate.

Secure a Domain Using Identity Cloud Service

Use Oracle Identity Cloud Service to protect applications and restrict administrative access for an Oracle WebLogic Server domain that you created with Oracle WebLogic Server for OCI.

By default, a 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.

When you create a domain running WebLogic Server 12c, you can also choose to enable Oracle Identity Cloud Service for authentication. The following diagram illustrates this configuration.



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

Topics:

- Create WebLogic Administrator Groups
- Update WebLogic Administrator Roles
- Update Protected Application Resources
- Update Application Deployment Descriptors

Create WebLogic Administrator Groups

Create groups in Oracle Identity Cloud Service to grant users administrative access to your domain.

Global roles in WebLogic Server control the administrative operations that a user can perform in the domain. For example, users with the `Deployer` role can deploy Java applications to the domain. By default, these roles are assigned to group names like `Deployers`. After creating these groups in Oracle Identity Cloud Service, you can add users to them.

When you create a domain, you specify a default administrative user. This user is configured in the default WebLogic Server identity store. You can use standard WebLogic Server tools like the Administration Console in order to modify this user or to change its password.

1. From the Identity Cloud Service console, expand the Navigation Drawer, and then click **Groups**.
2. Create the following groups.
   - `Administrators`
   - `Deployers`
   - `Operators`
   - `Monitors`

   By default, members of these groups will have administrative access to all domains that you create with Oracle WebLogic Server for OCI and that you configure to use Oracle Identity Cloud Service.
3. Optional: Create groups to control administrative access to a specific domain.

   For example:
   - `MyDomain_Administrators`
   - `MyDomain_Deployers`
   - `MyDomain_Operators`
   - `MyDomain_Monitors`

See these topics in *Administering Oracle Identity Cloud Service*:

- Create Groups
- Assign User Accounts to the Group

Update WebLogic Administrator Roles

Map groups in Oracle Identity Cloud Service to the administrator roles in your domain.

By default, the global administrator roles in a domain are mapped to these groups.

- `Administrators`
- `Deployers`
- `Operators`
- `Monitors`

If you do not modify the administrator roles in a domain, members of these groups in Oracle Identity Cloud Service will have access to all domains that you create with Oracle WebLogic Server for OCI and that you configure to use Oracle Identity Cloud Service.

Sign in to the WebLogic Server Administration Console for your domain. See Access the WebLogic Console.

1. From the WebLogic Server Administration Console, click **Security Realms**.
2. Click the default realm.
3. Click the **Roles and Policies** tab.
4. From the Roles table, expand **Global Roles**, and then expand **Roles**.
5. Click **View Role Conditions** for the `Admin` role.
6. Click the group name assigned to this role. The default is **Administrators**.
7. Enter the name of the Oracle Identity Cloud Service group to which you want to map to this role.
8. Click **OK**, and then click **Save**.
9. From the breadcrumb links at the top of the page, click **Realm Roles**.
10. Repeat from step 4 for each administrator role that you want to update.

Update Protected Application Resources

Configure the URL patterns that Oracle Identity Cloud Service uses to determine which application requests require authentication for your domain.

Oracle WebLogic Server for OCI provisions each compute instance in the domain with the App Gateway software appliance. 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 OCI also creates an enterprise application in Oracle Identity Cloud Service for the domain. The enterprise application defines which resources require the user to be authenticated, and which resources don't require authentication.

By default, all requests whose URI begins with `/__protected` (two underscore characters followed by the word "protected") are protected. For example, a client request to the URL `https://`*`<lb_host>`*`/`**`__protected`**`/myapp/doaction` requires authentication, while a request to `https://`*`<lb_host>`*`/myapp/doaction` does not.

1. From the Identity Cloud Service console, expand the Navigation Drawer, and then click **Applications**.
2. Click the enterprise application associated with your domain.

   The name of the application is *`<stack>`*`_enterprise_idcs_app_`*`<timestamp>`*. For example, `myweblogic_enterprise_idcs_app_2019-08-01T01:02:01.123456`.
3. Click the **SSO Configuration** tab.
4. Expand **Resources**.
5. Add, remove, or update the resources in this application.

   You can use regular expressions (regex) to define the protected URL patterns. If you use regular expressions, you must select the **Regex** check box.

   For example, suppose the Java applications deployed to this domain are configured to use the context roots `store` and `marketplace`. To protect all requests to the marketplace application, and also requests to the path `/store/cart`, add the following resources:

   - `/marketplace/.*`
   - `/store/cart/.*`

6. Expand **Authentication Policy**.
7. Under Managed Resources, add, remove, or update the policy for each resource.

   You can also change the order in which the policies are evaluated.

   Set a policy's **Authentication Method** to `Public` if you want the specified resource to be visible to anyone.

See About Enterprise Applications in *Administering Oracle Identity Cloud Service*.

Update Application Deployment Descriptors

Secure a Java application that's deployed to your domain by updating the application's context path, security constraints, and role assignments.

Oracle WebLogic Server supports the Java Enterprise Edition declarative model for securing web applications with XML deployment descriptors.

1. Update the value of `context-root` in the application's `weblogic.xml` file. Prefix the current value with one of the protected resource URLs that are defined in the Oracle Identity Cloud Service enterprise application for this domain.

   By default, the only protected context root is `/__protected` (two underscore characters followed by the word "protected"). For example:

   ```
   <context-root>/__protected/store</context-root>
   ```

2. Create one or more `security-role` elements in the application's `web.xml` file.

   Simply list the user roles for your application. For example:

   ```
   <security-role>
     <role-name>HRAdmin</role-name>
   </security-role>
   ```

3. Create one or more `security-constraint` elements in the application's `web.xml` file.

   Each security constraint grants access to one or more URL patterns in your application, and to specific roles. For example:

   ```
   <security-constraint>
     <web-resource-collection>
       <web-resource-name>AdminPages</web-resource-name>
       <url-pattern>/admin/*</url-pattern>
     </web-resource-collection>
     <auth-constraint>
       <role-name>HRAdmin</role-name>
   ```

```
  </auth-constraint>
</security-constraint>
```

Do not include the context root path in the URL patterns.

4. Create one or more `security-role-assignment` elements in the application’s `weblogic.xml` file.

   Map your application roles to specific users and/or groups found in Oracle Identity Cloud Service. For example:

```
<security-role-assigment>
  <role-name>HRAdmin</role-name>
  <principal-name>HRManagersGroup</principal-name>
</security-role-assigment>
```

5. Set the login configuration of the application to `CLIENT-CERT`.

```
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>default</realm-name>
</login-config>
```

6. Redeploy your application for these changes to take effect.

   For example, use the WebLogic Server Administration Console.

Secure Web Services Using Identity Cloud Service

Use Oracle Identity Cloud Service and Oracle Web Services Manager to protect web service applications and clients that you deploy to Oracle WebLogic Server for OCI domains.

This configuration is applicable only for domains that you created with Oracle WebLogic Server for OCI, and that meet all of these requirements:

- Is JRF-enabled
- Includes a load balancer that is configured for HTTPS. For domains that were created before June 2020, see Configure SSL for a Domain.
- Uses Oracle Identity Cloud Service for authentication. See Access the Sample Application Using Identity Cloud Service.

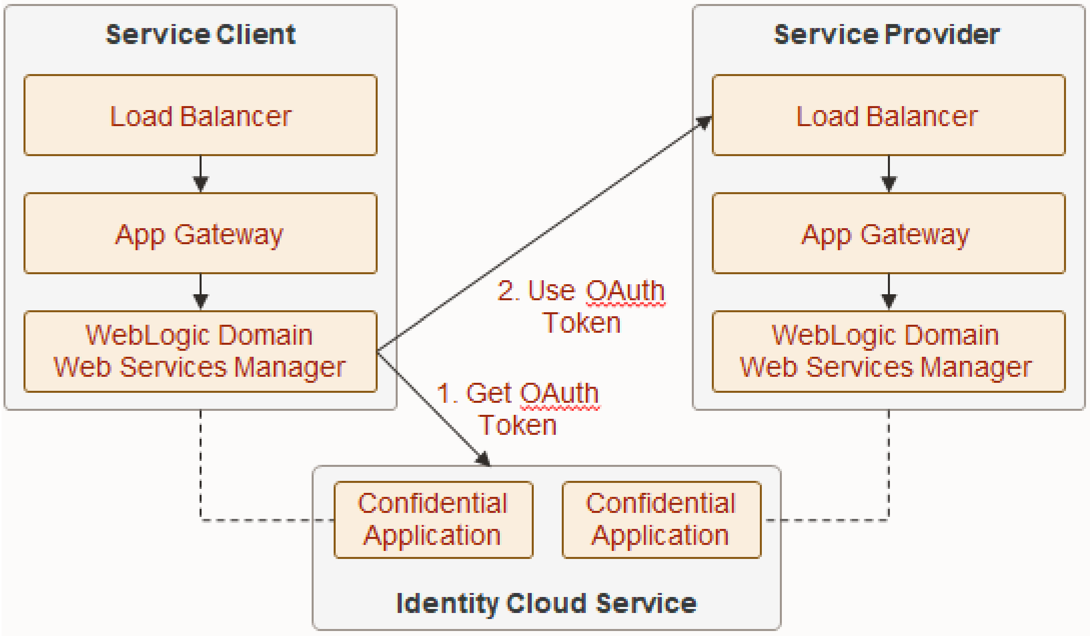
All JRF-enabled domains include Oracle Web Services Manager (OWSM), which provides a policy framework to manage and secure web services consistently across your organization. Both Oracle Web Services Manager and Oracle Identity Cloud Service support the OAuth protocol. The web service client requests an access token by authenticating with the authorization server (Oracle Identity Cloud Service) and presenting the authorization grant. The Oracle Web Services Manager server-side agent validates the access token and then accepts the client request if valid.

See Using OAuth2 with Oracle Web Services Manager in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

When you secure web service communication, the following terms are used:

- *Provider* - The Oracle WebLogic Server for OCI stack (WebLogic Server domain, load balancer, and so on) that hosts your web service application.
- *Client* - The Oracle WebLogic Server for OCI stack that hosts your web service client application.

The following diagram illustrates this security configuration.



Topics:

- Deploy a Sample Web Service Client Application
- Get Information About Identity Cloud Service
- Configure OAuth for the Web Services Provider
- Update the Confidential Application for the Web Services Provider
- Configure OAuth for the Web Services Client
- Update the Confidential Application for the Web Services Client
- Test the Sample Web Service Client

Deploy a Sample Web Service Client Application

To quickly verify the OAuth integration between Oracle Web Services Manager and Oracle Identity Cloud Service, you can build and deploy a sample client application.

This sample web application consists of a single page with an HTML form, and a single Servlet that invokes the specified web service URL. The client uses the OAuth policy `oracle/http_oauth2_token_over_ssl_idcs_client_policy`.

Alternatively, you can deploy and test your own web service client application.

1. Connect to the Administration Server node in your client domain as the `opc` user.

```
ssh -i <path_to_private_key> opc@<node_public_ip>
```

2. Change to the `oracle` user.

   ```
   sudo su - oracle
   ```

3. Create the web application directory structure.

   ```
   mkdir -p /home/oracle/oauth-client/WEB-INF/classes
   ```

4. Copy and paste the following text into a new file named `/home/oracle/oauth-client/index.jsp`.

   ```
   <html>
   <head>
       <meta http-equiv="Content-Type" content="text/html"/>
       <title>Test web services call using wss-oauth</title>
   </head>
   <body>
   <form name="oauth-test" method="post"
   action="helloworldclientservlet">
       <table>
           <tr>
               <th>Address</th>
               <td>
                   <input name="address" type="text" size="110"
   value=""/>
               </td>
           </tr>
           <tr>
               <th>Scope</th>
               <td>
                   <input name="scope" type="text" size="110"
   value=""/>
               </td>
           </tr>
           <tr>
               <th>Test</th>
               <td>
                   <input name="submit" type="submit" size="20"
   value="Test"/>
               </td>
           </tr>
       </table>
   </form>
   </body>
   </html>
   ```

5. Copy and paste the following text into a new file named `/home/oracle/HelloWorldClientServlet.java`.

   ```
   import java.util.*;
   import java.io.*;
   import java.lang.*;
   import java.security.AccessController;
   import javax.security.auth.Subject;
   ```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import weblogic.jaxrs.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import com.sun.jersey.api.client.filter.LoggingFilter;
import com.sun.jersey.api.client.ClientResponse;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
import oracle.wsm.security.util.SecurityConstants;

public class HelloWorldClientServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        String BASE_URI = request.getParameter("address");
        String SCOPE_URI = request.getParameter("scope");
        PropertyFeature scope = new
PropertyFeature(SecurityConstants.ConfigOverride.CO_SCOPE, SCOPE_URI);
        PolicyReferenceFeature clientPRF = new
PolicyReferenceFeature("oracle/
http_oauth2_token_over_ssl_idcs_client_policy", scope);
        DefaultClientConfig cc = new DefaultClientConfig();
        Map<String, Object> properties = cc.getProperties();
        properties.put(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE, new
PolicySetFeature(clientPRF));
        Client client = Client.create(cc);
        OutputStream baos = new ByteArrayOutputStream();
        PrintStream ps = new PrintStream(baos);
        client.addFilter(new LoggingFilter(ps));
        try {
            Subject subject =
Subject.getSubject(AccessController.getContext());
            Set<java.security.Principal> principals =
subject.getPrincipals();
            for (java.security.Principal principal : principals) {
                out.println("principal : " + principal.toString());
            }
            WebResource webResource = client.resource(BASE_URI);
            ClientResponse res = webResource.get(ClientResponse.class);
            ps.flush();
            out.println(baos.toString());
        } catch (Exception e) {
            String[] messages = getAllInnerErrorMessages(e);
            int count = 0;
            String allMsg = null;
            for (String mes : messages) {
                if (count != 0) {
                    allMsg += "\n|";
                    allMsg = allMsg + "\n+" + (getString("-", count
* 3) + "-> Caused By : " + mes);
                } else {
                    allMsg = mes;
                }
                count++;
            }
            out.println("The client was not able to call the
service using OAuth. The exception causes are: \n" + allMsg);
        }
        out.close();
    }

    private String[] getAllInnerErrorMessages(Throwable th) {
        List<String> all = new ArrayList<String>();
        boolean msgFound = false;
        while (th != null) {
            if (th.getMessage() != null && !
th.getMessage().trim().equals("")) {
                if (!msgFound && all.size() > 0) {
                    all.add(0, th.getMessage());
                }
                all.add(th.getMessage());
                msgFound = true;
            } else {
                all.add(th.getClass().getName());
            }
            th = th.getCause();
        }
        return all.toArray(new String[0]);
    }

    private String getString(String sp, int count) {
        StringBuffer ret = new StringBuffer();
        for (int i = 0; i < count; i++) {
            ret.append(sp);
        }
        return ret.toString();
    }

}
```

6. Copy and paste the following text into a new file named `/home/oracle/oauth-client/WEB-INF/web.xml`.

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5" xmlns="http://java.sun.com/xml/ns/javaee">
    <servlet>
        <servlet-name>HelloWorldClientServlet</servlet-name>
        <servlet-class>HelloWorldClientServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloWorldClientServlet</servlet-name>
        <url-pattern>/helloworldclientservlet</url-pattern>
    </servlet-mapping>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Success</web-resource-name>
            <url-pattern>/index.jsp</url-pattern>
        </web-resource-collection>
    </security-constraint>
    <login-config>
        <auth-method>CLIENT-CERT</auth-method>
    </login-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

7. Run `setWLSEnv.sh`.

   ```
   source /u01/app/oracle/middleware/wlserver/server/bin/setWLSEnv.sh
   ```

8. Compile the Servlet and put the class file in `/home/oracle/oauth-client/WEB-INF/classes`.

   ```
   cd /home/oracle
   javac -cp $CLASSPATH:/u01/app/oracle/middleware/oracle_common/modules/
   clients/com.oracle.webservices.wls.jaxws-owsm-client.jar \
   -d /home/oracle/oauth-client/WEB-INF/classes \
   HelloWorldClientServlet.java
   ```

9. Package the application.

   ```
   cd oauth-client
   zip -r ../oauth-client.war *
   cd ..
   ```

10. Use the WebLogic Scripting Tool (WLST) to deploy `oauth-client.war` to your domain.

    ```
    /u01/app/oracle/middleware/oracle_common/common/bin/wlst.sh
    connect('<admin_user>','<admin_password>','t3://
    <admin_server_IP>:<admin_server_port>')
    deploy(appName='oauth-client',path='/home/oracle/oauth-
    client.war',targets='<server_or_cluster>',upload='true')
    ```

    Example:

    ```
    /u01/app/oracle/middleware/oracle_common/common/bin/wlst.sh
    connect('weblogic','<admin_password>','t3://203.0.113.20:7001')
    deploy(appName='oauth-client',path='/home/oracle/oauth-
    client.war',targets='wlsoci_cluster',upload='true')
    ```

Get Information About Identity Cloud Service

Record configuration details about your Oracle Identity Cloud Service instance, and also download its certificates.

Before you begin, identify the confidential application in Oracle Identity Cloud Service that was created for your web services client domain. See Identity Resources for Oracle Identity Cloud Service.

1. Access the Oracle Identity Cloud Service console.
2. From the navigation menu, click **Applications**.
3. Click the confidential application that was created for your client domain.
4. Click the **Configuration** tab.
5. Under General Information, copy the **Client ID**.
6. Click **Show Secret**, and then copy the secret value.
7. From the navigation menu, click **Settings**, and then click **Default Settings**.
8. If it's not already enabled, select **Access Signing Certificate**, click **Save**, and then click **Yes**.
9. Access the following URL: `https://idcs-GUID.identity.oraclecloud.com/admin/v1/SigningCert/jwk`

   You can obtain the `GUID` for your Oracle Identity Cloud Service instance from the console URL.

   The response contains two certificates, separated by a comma.

   ```
   {"keys":[{"kty":"RSA",...,"kid":"SIGNING_KEY",...:
   ["<idcs_cert>","<idcs_root_ca_cert>"],"key_ops":
   ["verify","encrypt"],"alg":"RS256",...}]}
   ```

10. Copy and paste the first certificate, `idcs_cert`, into a new file named `idcs.cert`.

    Add the standard certificate header and footer lines.

    ```
    -----BEGIN CERTIFICATE-----
    <idcs-cert>
    -----END CERTIFICATE-----
    ```

11. Copy and paste the second certificate, `idcs_root_ca_cert`, into a new file named `idcs_ca.cert`.

    Add the standard certificate header and footer lines.

    ```
    -----BEGIN CERTIFICATE-----
    <idcs_root_ca_cert>
    -----END CERTIFICATE-----
    ```

12. Use `openssl` to view `idcs.cert` in text format.

    ```
    openssl x509 -in idcs.cert -text
    ```

13. Copy the contents of the `Subject` field, which contains a list of distinguished names (DN).

    ```
    Certificate:
        Data:
            ...
            Validity
                Not Before: Aug 27 09:20:19 2019 GMT
                Not After : Aug 27 09:20:19 2029 GMT
            Subject: <dn_list>
            ...
    ```

14. Use `openssl` to view `idcs_ca.cert` in text format, and to verify that it has no validation errors.

    ```
    openssl x509 -in idcs_ca.cert -text
    ```

15. Access the following URL: `https://idcs-GUID.identity.oraclecloud.com/.well-known/idcs-configuration`

16. From the response, copy the values for `issuer` and `token_endpoint`.

    For example:

    ```
    {
      ...
      "openid-configuration" : {
        "issuer" : "https://identity.oraclecloud.com/",
        "authorization_endpoint" : "https://idcs-
    GUID.identity.oraclecloud.com/oauth2/v1/authorize",
        "token_endpoint" : "https://idcs-GUID.identity.oraclecloud.com/
    oauth2/v1/token",
        ...
    ```

17. Return to the Oracle Identity Cloud Service console, click **Settings**, and then click **Default Settings**.

18. If **Access Signing Certificate** was disabled previously, then disable it again. Click **Save**, and then click **Yes**.

Configure OAuth for the Web Services Provider

Establish trust between Oracle Web Services Manager in your provider domain and Oracle Identity Cloud Service by importing certificates and creating global policy attachments.

The global policy affects all web services deployed to this domain. Alternatively, you can create policies for individual web services.

1. Copy the files `idcs.cert` and `idcs_ca.cert` to the Administration Server node in your provider domain as the `opc` user.

   Place the files in the `/tmp` directory.

   ```
   scp -i <path_to_private_key> idcs.cert opc@<node_public_ip>:/tmp
   scp -i <path_to_private_key> idcs_ca.cert opc@<node_public_ip>:/tmp
   ```

2. Connect to the Administration Server node in your domain as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

3. Change the owner of the certificate files to `oracle`.

   ```
   sudo chown oracle:oracle /tmp/*.cert
   ```

4. Change to the `oracle` user.

   ```
   sudo su - oracle
   ```

5. Copy and paste the following text into a new file named `config_owsm_provider.py`.

   ```
   ip='<admin_server_IP>'
   port='<admin_server_port>'
   user='<admin_user>'
   pwd='<password>'
   dn_list='<idcs_dn_list>'
   trusted_issuer='<issuer>'
   idcs_cert_path = '/tmp/idcs.cert'
   idcs_ca_cert_path = '/tmp/idcs_ca.cert'
   ```

6. Update the variables in `config_owsm_provider.py`.
   - The IP address of this node
   - The port number of the Administration Server (the default is `7001`)
   - The user name and password of the domain administrator
   - The distinguished name (DN) from the Subject field in `idcs.cert`.
   - The Oracle Identity Cloud Service issuer name

   Reverse the order of the DN entries. For example, change `CN = abc,CN = def` to `CN = def,CN = abc`.

   For example:

   ```
   ip='203.0.113.20'
   port='7001'
   user='weblogic'
   pwd='<password>'
   dn_list='CN=idcs-GUID,CN=Cloud9,CN=sslDomains'
   trusted_issuer='https://identity.oraclecloud.com/'
   ```

7. Copy and paste the following text into `config_owsm_provider.py`, and after the variable declarations.

   ```
   connect(user, pwd,'t3://' + ip + ':' + port)

   try:
      beginWSMSession()
      createWSMPolicySet('oauth-ps-ws-service','ws-
   service','Domain("*")','Global policy for default interactions for
   ```

```
ws-service',true)
   attachWSMPolicy('oracle/
wss11_saml_or_username_token_with_message_protection_service_policy')
   validateWSMPolicySet('oauth-ps-ws-service')
   displayWSMPolicySet('oauth-ps-ws-service')
finally:
   commitWSMSession()

try:
   beginWSMSession()
   createWSMPolicySet('oauth-ps-rest-resource','rest-
resource','Domain("*")','Global policy for default interactions for rest-
resource',true)
   attachWSMPolicy('oracle/multi_token_over_ssl_rest_service_policy')
   validateWSMPolicySet('oauth-ps-rest-resource')
   displayWSMPolicySet('oauth-ps-rest-resource')
finally:
   commitWSMSession()

try:
   beginWSMSession()
   createWSMTokenIssuerTrustDocument('trust-doc',None)
   setWSMConfiguration(None, 'TokenIssuerTrust', 'name', None, ['trust-
doc'])
   selectWSMTokenIssuerTrustDocument('trust-doc')
   setWSMTokenIssuerTrust('dns.jwt',trusted_issuer,[dn_list])
   setWSMTokenIssuerTrust('dns.sv',trusted_issuer,[dn_list])
   setWSMTokenIssuerTrust('dns.hok',trusted_issuer,[dn_list])
   setWSMTokenIssuerTrust('dns.jwt','www.oracle.com',[])
   setWSMTokenIssuerTrust('dns.sv','www.oracle.com',[])
   setWSMTokenIssuerTrust('dns.hok','www.oracle.com',[])
finally:
   commitWSMSession()

refreshWSMCache()

svc=getOpssService(name='KeyStoreService')
try:
   svc.createKeyStore(appStripe='owsm', name='keystore',
password='',permission=true)
except Exception, ex:
   ex_msg = str(ex)
   if 'Keystore keystore in stripe owsm already exists' in ex_msg:
      print 'Keystore keystore in stripe owsm already exists. Continue...'
   else:
      raise
svc.importKeyStoreCertificate(appStripe='owsm', name='keystore',
password='', alias='idcs-oauthkey', keypassword='',
type='TrustedCertificate', filepath=idcs_cert_path)
svc.importKeyStoreCertificate(appStripe='owsm', name='keystore',
password='', alias='idcs-oauthkeyca', keypassword='',
type='TrustedCertificate', filepath=idcs_ca_cert_path)
```

8. Run `config_owsm_provider.py` using WLST.

   ```
   /u01/app/oracle/middleware/oracle_common/common/bin/wlst.sh
   config_owsm_provider.py
   ```

   Verify that the script ran successfully:

   ```
   ...
   Session started for modification.
   The policy set was created successfully in the session.
   Policy reference "oracle/
   wss11_saml_or_username_token_with_message_protection_service_policy"
    added.
   ...
   Creating policy set oauth-ps-ws-service in repository.
   Session committed successfully.
   Session started for modification.
   The policy set was created successfully in the session.
   Policy reference "oracle/multi_token_over_ssl_rest_service_policy"
   added.
   ...
   Creating policy set oauth-ps-rest-resource in repository.
   Session committed successfully.
   Session started for modification.
   New Token Issuer Trust document named trust-doc created.
   ...
   Creating tokenissuertrust trust-doc in repository.
   Session committed successfully.
   ...
   Keystore created
   Certificate imported.
   Certificate imported.
   ```

Update the Confidential Application for the Web Services Provider

Update the resources that are protected by OAuth in the domain's confidential application found in Oracle Identity Cloud Service.

1. Access the Oracle Identity Cloud Service console.
2. From the navigation menu, click **Applications**.
3. Click the confidential application that was created for your provider domain.
4. Click the **Configuration** tab.
5. Under Resources, click **Register Resources**.
6. For **Primary Audience**, enter the URL of the load balancer that directs traffic to the provider domain.

   ```
   https://<lb_public_ip>:443
   ```

7. For **Scopes**, click **Add**.
8. Set the name of the new scope to `external`, and then click **Add**.

9. Click **Save**.

Configure OAuth for the Web Services Client

Establish trust between Oracle Web Services Manager in your client domain and Oracle Identity Cloud Service by importing certificates and creating global policy attachments.

The global policy affects all web service clients deployed to this domain. Alternatively, you can create policies for individual applications.

1. Connect to the Administration Server node in your client domain as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

2. Change to the `oracle` user.

   ```
   sudo su - oracle
   ```

3. Copy and paste the following text into a new file named `config_owsm_client.py`.

   ```
   def config_policyset(policy_set_name,subject_type):
       try:
           beginWSMSession()

   createWSMPolicySet(policy_set_name,subject_type,resource_scope,desc,is_ena
   bled)
           attachWSMPolicy(policy)
           setWSMPolicyOverride(policy,'token.uri',token_uri)
           setWSMPolicyOverride(policy,'oauth2.client.csf.key',csf_key)
           validateWSMPolicySet(policy_set_name)
       finally:
           commitWSMSession()
           displayWSMPolicySet(policy_set_name)

   ip='<admin_server_IP>'
   port='<admin_server_port>'
   user='<admin_user>'
   pwd='<password>'
   client_id='<idcs_app_client_id>'
   client_secret='<idcs_app_client_secret>'
   token_uri = '<token_endpoint>'
   dn_list = '<dn_list>'
   app_resource = 'resource=<client_app_name>'
   ```

4. Update the variables in `config_owsm_client.py`.
   - The IP address of this node
   - The port number of the Administration Server (the default is `7001`)
   - The user name and password of the domain administrator
   - The client ID and secret of the confidential application in Oracle Identity Cloud Service that was created for the domain
   - The Oracle Identity Cloud Service token endpoint

- The distinguished name (DN) to use for the generated certificate in Oracle Web Services Manager
- The name of the web service client application that is deployed to the domain

For example:

```
ip='203.0.113.30'
port='7001'
user='weblogic'
pwd='<password>'
client_id='ABCD1234efgh5678IJKL9012'
client_secret='<idcs_app_client_secret>'
token_uri = 'https://
idcs-1234abcd5678EFGH9012ijkl.identity.oraclecloud.com/oauth2/v1/
token'
dn_list = 'CN=OWSM, OU=ST, O=Oracle, L=RedWood, ST=CA, C=US'
app_resource = 'resource=oauth-client'
```

5. Copy and paste the following text into `config_owsm_client.py`, and after the variable declarations.

```
connect(user, pwd,'t3://' + ip + ':' + port)

csf_key = 'idcs.oauth2.client.credentials'
createCred(map='oracle.wsm.security',key=csf_key,user=client_id,pass
word=client_secret)

is_enabled = true
policy = 'oracle/oauth2_config_client_policy'
resource_scope = 'Domain("*")'
desc = ''
config_policyset('oauth-ps-config-rest-connection','rest-
connection')
config_policyset('oauth-ps-config-rest-client','rest-client')
config_policyset('oauth-ps-config-ws-connection','ws-connection')
config_policyset('oauth-ps-config-ws-client','ws-client')
config_policyset('oauth-ps-config-ws-callback','ws-callback')
refreshWSMCache()

svc = getOpssService(name='KeyStoreService')
try:
   svc.createKeyStore(appStripe='owsm', name='keystore',
password='',permission=true)
except Exception, ex:
   ex_msg = str(ex)
   if 'Keystore keystore in stripe owsm already exists' in ex_msg:
      print 'Keystore keystore in stripe owsm already exists.
Continue...'
   else:
      raise
svc.generateKeyPair(appStripe='owsm', name='keystore', password='',
dn=dn_list, keysize='2048', alias='orakey', keypassword='')

svc.exportKeyStoreCertificate(appStripe='owsm', name='keystore',
password='', alias='orakey', keypassword='',
type='TrustedCertificate', filepath='/tmp/orakey.cert')

grantPermission(appStripe=None, codeBaseURL='file:$
{common.components.home}/modules/oracle.wsm.common/wsm-agent-
core.jar',principalClass=None,principalName=None,permClass='oracle.wsm.sec
urity.WSIdentityPermission',permTarget=app_resource, permActions='assert')
```

6. Run `config_owsm_client.py` using WLST.

   ```
   /u01/app/oracle/middleware/oracle_common/common/bin/wlst.sh
   config_owsm_client.py
   ```

   Verify that the script ran successfully:

   ```
   ...
   Credential created successfully.
   Session started for modification.
   The policy set was created successfully in the session.
   Policy reference "oracle/oauth2_config_client_policy" added.
   ...
   Creating policy set oauth-ps-config-rest-connection in repository.
   Session committed successfully.
   ...
   Session started for modification.
   The policy set was created successfully in the session.
   Policy reference "oracle/oauth2_config_client_policy" added.
   ...
   Creating policy set oauth-ps-config-rest-client in repository.
   Session committed successfully.
   ...
   Keystore created
   Key pair generated
   Certificate exported.
   ```

7. Change the owner of the generated certificate file to the `opc` user.

   ```
   exit
   sudo chown opc:opc /tmp/orakey.cert
   ```

8. Download the certificate file as the `opc` user.

   ```
   scp -i <path_to_private_key> opc@<node_public_ip>:/tmp/orakey.cert .
   ```

Update the Confidential Application for the Web Services Client

Import the Oracle Web Services Manager certificate into the domain's confidential application found in Oracle Identity Cloud Service.

1. Access the Oracle Identity Cloud Service console.
2. From the navigation menu, click **Applications**.
3. Click the confidential application that was created for your client domain.
4. Click the **Configuration** tab.

5. Under Client Configuration, select **Trusted** for **Client Type**.
6. For **Certificate**, click **Import**.
7. For **Certificate Alias**, enter a unique name.

   For example, `orakey_oauthclient`
8. Select the file `orakey.cert`, and then click **Import**.
9. Click **Add Scope**.
10. Select the confidential application of the domain that is the web services provider, and then click **Add**.
11. Click **Save**.

Test the Sample Web Service Client

If you deployed the sample client application, you can use it to quickly verify the OAuth integration between Oracle Web Services Manager and Oracle Identity Cloud Service.

1. Use the load balancer to access the sample application on your client domain.

   ```
   https://<client_lb_public_ip>/oauth-client
   ```

2. For **Address**, enter the URL of your web service provider.

   ```
   https://<provider_lb_public_ip>/<service_endpoint>
   ```

   For example:

   ```
   https://203.0.113.21/myapp/myservice
   ```

3. For **Scope**, enter this value.

   ```
   https://<provider_lb_public_ip>:443external
   ```

   For example:

   ```
   https://203.0.113.21:443external
   ```

4. Click the **Test** button.

Integrate OPSS User and Group APIs with Identity Cloud Service

Update your domain's confidential application in Oracle Identity Cloud Service to support the user and group lookup APIs in Oracle Platform Security Services.

This configuration is applicable only for domains that you created with Oracle WebLogic Server for OCI, and that meet all of these requirements:

- Is JRF-enabled

- Uses Oracle Identity Cloud Service for authentication. See Access the Sample Application Using Identity Cloud Service.

All JRF-enabled domains include Oracle Platform Security Services (OPSS), which provides an abstraction layer in the form of APIs that insulates developers from security and identity management implementation details. For example, developers do not need to know the details of accessing the security repository or managing keys and certificates. See Introduction to Oracle Platform Security Services in *Securing Applications with Oracle Platform Security Services*.

A domain that uses Oracle Identity Cloud Service is associated with a confidential application, which grants WebLogic Server one or more Oracle Identity Cloud Service client roles. By default, this confidential application has a single role, `Authenticator Client`, which enables Java applications to use the OPSS authentication APIs. If your Java applications use the OPSS APIs to look up user and group information, then you must add more roles to the confidential application. See AppRole Permissions in *REST API for Oracle Identity Cloud Service*.

> **Note:**
>
> Oracle recommends that you secure Java applications that access user and group information, to ensure they are accessed only by authorized users.

1. Access the Oracle Identity Cloud Service console.
2. From the navigation menu, click **Applications**.
3. Click the confidential application that was created for your domain.
4. Click the **Configuration** tab.
5. Under Client Configuration, locate **Grant the client access to Identity Cloud Service Admin APIs**, and then click **Add**.
6. Select one or more roles.

Role	Allowed Operations
Cloud Gate	Query users
User Administrator	Query and manage users and groups
Identity Domain Administrator	Access to all Identity Cloud Service operations

7. Click **Add**.
8. Click **Save**.

Upgrade the Oracle Identity Cloud Service App Gateway Version

If your Oracle WebLogic Server for OCI domain uses Oracle Identity Cloud Service for authentication, you must upgrade the App Gateway on each compute instance in the domain as an `opc` user. The latest App Gateway version is 21.2.2.

The upgrade steps are required only if both of these are true:

- You selected the **Enable Authentication Using Identity Cloud Service** option when creating the domain.
- The Oracle Identity Cloud Service App Gateway version is later than 19.2.1.

To upgrade the Oracle Identity Cloud Service App Gateway version, perform the following steps:

> **Note:**
>
> You must delete the existing container and recreate the container with a new version of the image.

1. Download the Oracle Identity Cloud Service App Gateway Docker image.
   a. Access the Identity Cloud Service console.
   b. Expand the **Navigation Drawer**, click **Settings**, and then click **Downloads**.
   c. In the Downloads page, click **Download** to the right of **App Gateway Docker Image for Identity Cloud Service**, and download the file, `idcs-appgateway-docker<version>.zip` to a location on your system.
   d. Navigate to the directory where you downloaded the file, and extract the contents of the zip file.

      Example:

      ```
      unzip idcs-appgateway-docker-<version>.zip
      ```

      After unzip, the file, `appgateway-<version>.tar.gz` is created.
2. Download the Oracle Identity Cloud Service App Gateway wallet tool (optional).
   a. Repeat steps a and b from step 1.
   b. In the Downloads page, download the wallet file. For example, `idcs-appgateway-wallet-tool-<version>.zip`.
3. Copy the Oracle Identity Cloud Service App Gateway Docker image and App Gateway wallet tool to one of the virtual machines in the Oracle WebLogic Server for OCI instance.

   For example, copy the files, `appgateway-<version>.tar.gz` and `idcs-appgateway-wallet-tool-<version>.zip`.
4. Deploy the Oracle Identity Cloud Service App Gateway Docker image.
   a. In the Oracle WebLogic Server for OCI virtual machine (VM) instance, load the `.tar.gz` file to the local Docker registry.

      ```
      sudo docker load -i <.tar.gz file>
      ```

      Example:

      ```
      sudo docker load --input /tmp/appgateway-<version>.tar.gz
      ```

b. Verify that you see the image in the local Docker registry.

```
sudo docker images
```

5. Deploy the Oracle Identity Cloud Service App Gateway wallet file (optional).

a. Create a new `wallet_tool` directory, `/usr/lib/wallet_tool`.

```
sudo mkdir -p /usr/lib/wallettool/
```

b. Extract the `idcs-appgateway-wallet-tool zip` to `/usr/lib/wallet_tool`.

```
sudo unzip /tmp/idcs-appgateway-wallet-tool-<version>.zip -d /usr/lib/
wallet_tool/
```

6. Create the `cwallet.sso` file (optional).

If the wallet file is not deleted, you can use the existing wallet file (`cwallet.sso`) to upgrade to the latest App Gateway version, or upgrade the App Gateway wallet tool and generate a new `cwallet.sso` file.

Use one of the following methods to create the `cwallet.sso` file.

- Manual:

  a. Retrieve the client ID and client secret of the `app_gateway` using information in the `idcs_artfacts.txt` in the `/u01/data` directory.

  ```
  cat /u01/data/.idcs_artifacts.txt
  ```

  b. Take a note of the `displayName` of the `app_gateway` in `/u01/data/.idcs_artifacts.txt`.

  Example:

  ```
  {
    "confidential_app": {
      "meta": {
        "location": "https://idcs-
  <GUID>.identity.oraclecloud.com:443/admin/v1/Apps/
  <confidential_app_ID>"
             }
    },
    "app_gateway": {
      "meta": {
        "location": "https://idcs-
  <GUID>.identity.oraclecloud.com:443/admin/v1/CloudGates/
  <app_gateway_ID>"
             },
      "displayName":
  "idcs0706_app_gateway_2021-06-07T14:57:22.297066",
      "id": "< app_gateway_ID>"
    },
    "enterprise_app": {
      "meta": {
        "location": "https://idcs-
  <GUID>.identity.oraclecloud.com:443/admin/v1/Apps/
  ```

```
<enterprise_app_ID>"
            }
  }
}
```

> **Note:**
>
> Note: You must belong to the *Administrator* group in Oracle Identity Cloud Service to access this information.

c. In the Oracle Identity Cloud Service console, expand the **Navigation Drawer**, click **Security**, and then click **App Gateways**. In the App Gateways page, search for the App Gateway with the noted `displayName` and take a note of the client ID and client secret.

d. Navigate to `/u01/data/cloudgate_config/` directory and create the `cwallet.sso` file.

```
cd /u01/data/cloudgate_config/
export LD_LIBRARY_PATH=/usr/lib/wallet_tool/lib/
echo <client_secret>  | /usr/lib/wallet_tool/cgwallettool --
create -i <client_id>
```

- Using Scripts:

> **Note:**
>
> You can use scripts to create the `cwallet.sso` file for Oracle WebLogic Server for OCI version 21.2.3 or later; version 21.2.3 has the latest scripts to support Oracle Identity Cloud Service App Gateway version 21.2.2.

a. Add the Oracle Identity Cloud Service client ID and client secret to `/u01/data/cloudgate_config/appgateway-env`.

b. Run the `create_idcs_cloudgate_cwallet.sh` script as a root user.

> **Note:**
>
> Make sure you are using the latest version of the `create_idcs_cloudgate_cwallet.sh` script.

Example:

```
sudo echo "" >> /u01/data/cloudgate_config/appgateway-env
sudo echo "CG_APP_NAME=<client_id>" >> /u01/data/
cloudgate_config/appgateway-env
sudo echo "CG_APP_SECRET=<client_secret>" >> /u01/data/
```

```
cloudgate_config/appgateway-env
sudo sh /opt/scripts/idcs/create_idcs_cloudgate_cwallet.sh
```

7. Stop and remove the existing App Gateway container.

   ```
   sudo docker container stop appgateway
   sudo docker container rm appgateway
   ```

8. Create and start the new App Gateway container.

   Use one of the following methods to create and start the new App Gateway container:

   - Manual:

     a. Run the `update_metadata` script to update the metadata for `docker_image_version` and `docker_image_name` to point to the latest version.

        Example:

        ```
        sudo python3 /opt/scripts/utils/update_metadata.py -k
        idcs_cloudgate_docker_image_version -v <version>
        sudo python3 /opt/scripts/utils/update_metadata.py -k
        idcs_cloudgate_docker_image_name -v idcs/idcs-appgateway
        ```

        > **Note:**
        >
        > This step to update the metadata script is required if you upgrade the Oracle Identity Cloud Service App Gateway version during scale out.

     b. Navigate to the `/u01/data/cloudgate_config/` directory and change the permissions to `777` and the owner to `8000:8000` for this directory.

        Example:

        ```
        cd /u01/data/cloudgate_config/
        sudo chmod -R 777 /u01/data/cloudgate_config/
        sudo chown -R 8000:8000 /u01/data/cloudgate_config/*
        ```

     c. Start the App Gateway container using the `docker run` command.

        > **Note:**
        >
        > You must mount the local folder, `/u01/data/cloudgate_config` volume to the directory, `/usr/local/nginx/conf/` inside the container `<my-container>`.
        >
        > The `cwallet.sso` file that contains the client ID and client secret must be copied to the folder, `/usr/local/nginx/conf/` in the container so that the container can reference the wallet file.

Example:

```
sudo docker run -it -d --name <my-container> --env-file /u01/
data/cloudgate_config/appgateway-env /
--env HOST_MACHINE=`hostname -f` --env
CLOUDGATE_VERSION=<version> /
--volume /u01/data/cloudgate_config/:/usr/local/nginx/
conf/:z /
--net=host idcs/idcs-appgateway:<version>
```

- Using Scripts:

  > **Note:**
  >
  > You can use scripts to create the `cwallet.sso` file for Oracle WebLogic Server for OCI version 21.2.3 or later; version 21.2.3 has the latest scripts to support Oracle Identity Cloud Service App Gateway version 21.2.2.

  a. Run the `update_metadata` script to update the metadata for `docker_image_version` and `docker_image_name` to point to latest version.

  Example:

  ```
  sudo python3 /opt/scripts/utils/update_metadata.py -k
  idcs_cloudgate_docker_image_version -v <version>
  sudo python3 /opt/scripts/utils/update_metadata.py -k
  idcs_cloudgate_docker_image_name -v idcs/idcs-appgateway
  ```

  b. Start the App Gateway container using `run_cloudgate.sh`.

  ```
  sudo sh /opt/scripts/idcs/run_cloudgate.sh
  ```

9. Verify the upgrade.

   a. Check the App Gateway container logs.

   ```
   sudo docker logs appgateway
   ```

   b. Log in and connect to the container using `bash`.

   Example:

   ```
   sudo docker exec -it appgateway bash
   ```

   c. Navigate to the `bin` folder in the container, and check the `cloudgate-env` file.

   Example:

   ```
   cd /usr/local/nginx/logs/
   cd /scratch/oracle/idcs-cloudgate/latest/bin/
   ./cg-env
   ```

10. Remove the existing Oracle Identity Cloud Service Docker image.

    ```
    sudo docker image rm opc-delivery.docker.oraclecorp.com/idcs/
    <container_name:existing_version>
    ```

11. Repeat from **step 3** for all remaining compute instances in this domain.

Configure Session Persistence

If you created a domain with Oracle WebLogic Server for OCI, and your web applications require session persistence, you can update the load balancer for the domain.

Session persistence is a method to direct all requests originating from a single logical client to a single backend server. By default, session persistence is enabled on the load balancer with the `Enable load balancer cookie persistence` option, but you can update the load balancer after creating a domain.

Oracle Cloud Infrastructure Load Balancing supports two types of session persistence (stickiness):

- Application cookie persistence - The load balancer activates session persistence when a backend server sends a `Set-Cookie` response header containing a recognized cookie name.
- Load balancer cookie persistence - The load balancer inserts a cookie into the response, which enables session persistence. This method is useful when you have applications that cannot generate their own cookies.

To configure session persistence:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Networking**, and then click **Load Balancers**.
3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack is initially created, this is the compartment that contains the compute instances and network resources for the domain, or this is the network compartment that has only the network resources for the domain.
4. Click the name of the load balancer that's associated with your domain.

   The load balancer has `-lb` appended to the domain name. For example: `abcde1xy-lb`
5. Click **Backend Sets**.
6. Click the backend set for this load balancer.
7. Click **Edit**.
8. Modify the **Session Persistence** configuration.

   For example, if you select **Enable Application Cookie Persistence**, and set **Cookie Name** to `'*'`, then responses that contain any cookie will be directed to the same server in the domain.

See these topics in the Oracle Cloud Infrastructure documentation:

- Session Persistence
- Managing Backend Sets

Enable Session Affinity or Sticky Sessions

If you have clustered JMS resources and you access it by using an external client through a load balancer by rmi-tunneling, then you need to enable session affinity or sticky sessions.

Complete the following steps:

1. Access the WebLogic console of the clone instance. See Access the WebLogic Console.
2. Under **Domain Structure**, expand **Environment** > **Clusters**, and then click **Cluster-1**.
3. In the right pane, change the **Default Load Algorithm** property to `RoundRobinAffinity`.
4. Click **Save**.
5. Restart the WebLogic Server instances.

Update the Password Secret OCID and Policy

If you have moved your password to new secret or vault or updated the password, then you must update the password secret OCID and policy to read the secrets with new secrets OCID.

Moved your password to new secret or vault

If you have moved your password to new secret or vault, complete the following steps:

1. Update the metadata on each node by using `update_metadata.py` script, which is located at `/opt/scripts/utils/`:

   ```
   python3 /opt/scripts/update_metadata.py -k wls_admin_password_ocid -
   v <new_secrets_ocid>
   ```

   > **Note:**
   >
   > The command creates a backup of previous setup under `/opt/scripts/utils/metadata_backup_<time_stamp>.txt`.

2. Update the *`servicename`*`-secrets-policy` policy to read the secrets with new secrets OCID.

Update your password in the existing secret

If you have updated your password, complete the following steps:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Identity & Security**, and then click **Vault**.
3. Under **List Scope**, in the **Compartment** list, click the name of the compartment that contains the vault with the secret you want to provide with new secret contents.

4. From the list of vaults in the compartment, click the vault name.
5. Click **Secrets**, and then click the name of the secret with the secret contents you want to update.
6. Click **Create Secret Version**.
7. Specify the format of the secret contents you're providing by choosing a template type from the **Secret Type Template** list.
8. Click **Secret Contents**, and then enter the secret contents.
9. Click **Create Secret Version**.
10. Update the `*servicename*-secrets-policy` policy to read the secrets with new secrets OCID.

Manage Data Sources

After you create an Oracle WebLogic Server for OCI domain, 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.

Creating a data source for a DB System database or Oracle Autonomous Database is similar with one exception. With an Oracle Autonomous Database, you need the Oracle Autonomous Database client credentials or wallet files.

Oracle WebLogic Server for OCI 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)

> **Note:**
>
> From release 21.4.3 (December 9, 2021) onwards, you cannot provision a domain in Oracle Oracle WebLogic Server for OCI for Oracle WebLogic server versions 11g (10.3.6.0) and 12c (12.2.1.3) from the Marketplace. However, you can provision a domain for Oracle WebLogic server version 12c (12.2.1.3) using the terraform scripts See Create a Domain for Oracle WebLogic Server 12.2.1.3.0 Using Terraform. The WebLogic binaries for 12.2.1.3 are also available in the public images.

Topics:

- Create a Data Source for an Oracle Autonomous Database
- Create a Data Source for a DB System Database

Create a Data Source for an Oracle Autonomous Database

Oracle WebLogic Server for OCI provides two utility scripts to help you create Oracle Autonomous Database:

- A download script that downloads the Oracle Autonomous Database wallet files to a node
- A create script that creates the data source using the downloaded Oracle Autonomous Database wallet files and data source properties you provide
- *Prerequisite*: At the `root` compartment level, create an OCI policy with the following policy statement:

  ```
  Allow dynamic-group <service-prefix>-wlsc-principal-group to use
  autonomous-transaction-processing-family in compartment id
  <compartment-id>
  ```

To run the scripts, you need to access the nodes in your WebLogic domain as the `opc` user. The scripts are located in `/opt/scripts/utils` and can only be run as the `oracle` user.

The Oracle Autonomous Database must allow the WebLogic Server compute instances to access the database listen port (1521 by default). Update your access control list (ACL), if necessary. See Security Tools for Serverless Deployments.

Tasks:

- Download the Oracle Autonomous Database Wallet
- Configure a Data Source for an Oracle Autonomous Database

Download the Oracle Autonomous Database Wallet

The download script unpacks and copies the Oracle Autonomous Database wallet contents to a node.

> **Note:**
>
> The download script must be run before the create script that configures a data source.

If the data source target is the domain cluster, you must run the download script on every node in the cluster. If the target is individual servers, then run the script on those servers.

You need the public IP address of each node on which you plan to run the download script. Find the IP address on the compute instance details page in the Oracle Cloud Infrastructure console. Look up the bastion's public IP address and the private IP address of a node if the WebLogic domain is in a private subnet.

1. Open an SSH connection to a node as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

   Or,

   ```
   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>
   ```

2. Change to the `oracle` user.

   ```
   sudo su oracle
   ```

3. At the `root` compartment level, create an OCI policy with the following policy statement:

   ```
   Allow dynamic-group <service-prefix>-wlsc-principal-group  to read 
   autonomous-database in compartment id <atp-compartment-id>
   ```

4. Run the script `download_atp_wallet.sh` by providing the following parameters:

   - OCID of the Oracle Autonomous Database- You can find the OCID from the Oracle Autonomous Database details page in the Oracle Cloud Infrastructure console.
   - Password for the Oracle Autonomous Database wallet - The password must be at least 8 characters long, and includes at least 1 letter and either 1 numeric character or 1 special character.
   - Path to save the extracted Oracle Autonomous Database wallet files - The path to a directory on the domain where the script saves the extracted Oracle Autonomous Database wallet files. For example: `/u01/data/domains/thestack_domain/config/atp`

     The directory must be identical on every node where you run the script.

   Command:

   ```
   /opt/scripts/utils/download_atp_wallet.sh <atp_database_ocid> 
   <atp_wallet_password> <path_to_extract_wallet_files>
   ```

   Example:

   ```
   /opt/scripts/utils/download_atp_wallet.sh 
   ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v password /u01/data/
   domains/servicename_domain/config/atp
   ```

   The download script creates a subdirectory in the path you provide using the Oracle Autonomous Database OCID value. For example:

   ```
   /u01/data/domains/servicename_domain/config/atp/
   ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v
   ```

   Seven files are extracted to the subdirectory. The following is an example of the script response:

   ```
   <Aug 22, 2019 10:39:50 PM GMT> <INFO> <oci_utils> <(host:servicename-
   wls-0.subnet_dns_domain_name) - <WLSC-VM-INFO-001> ATP Wallet downloaded>
   Archive:  /tmp/atp_wallet.zip
     inflating: /u01/data/domains/servicename_domain/config/atp/
   ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/cwallet.sso
     inflating: /u01/data/domains/servicename_domain/config/atp/
   ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/tnsnames.ora
     inflating: /u01/data/domains/servicename_domain/config/atp/
   ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/truststore.jks
     inflating: /u01/data/domains/servicename_domain/config/atp/
   ```

```
ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/
ojdbc.properties
  inflating: /u01/data/domains/servicename_domain/config/atp/
ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/sqlnet.ora
  inflating: /u01/data/domains/servicename_domain/config/atp/
ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/ewallet.p12
  inflating: /u01/data/domains/servicename_domain/config/atp/
ocid1.autonomousdatabase.oc1.phx.a1b2c3d4e56z7y8x9w10v/keystore.jks
```

5. Repeat steps 1 through 3 on each node where you have to run the download script. Depending on the data source target, run the download script on every node in the cluster or on individual servers.

Configure a Data Source for an Oracle Autonomous Database

The create script configures a JDBC data source using the downloaded Oracle Autonomous Database wallet files and the data source properties you provide.

> **Note:**
>
> You must run the download script before you run the create script to configure the data source.

When you execute the create script, you can let the script prompt you for the properties one at a time, or you can supply the properties in a configuration file.

1. Before you run the create script, prepare the required information for the following data source properties. If you plan to provide a properties configuration file when you run the create script, put each property name and value pair on one line. For example:

```
ds.name=myds1
ds.jndi.name=jdbc/myds1
```

Property Description	Property Name	Required
Name for the JDBC data source	`ds.name`	Y
OCID of the Oracle Autonomous Database	`atp.db.id`	Y
Name of the Oracle Autonomous Database	`atp.db.name`	Y
Path to the extracted Oracle Autonomous Database wallet files	`atp.wallet.path`	Y
Oracle Autonomous Databaseuser name	`db.user`	Y
Oracle Autonomous Databaseuser password	`db.password`	Y

Property Description	Property Name	Required
Oracle Autonomous Database wallet password. This is the password you provided when you ran the download script.	db.wallet.password	Y
WebLogic Server administrator user name	wls.admin.user	Y
WebLogic Server administrator password	wls.admin.password	Y
t3 or t3s URL to the WebLogic Administration Server node	wls.admin.url	Y
Type of target on which to deploy the data source. Value: Cluster or Server	ds.target.type	Y
Name of the cluster or a comma separated list of server names	ds.target.names	Y
JNDI namespace Default is jdbc/<ds.name>	ds.jndi.name	N
Oracle Autonomous Database service for the data source connection. Value: low (default), medium, tp, or tpurgent	db.level	N
JDBC driver class Default is oracle.jdbc.OracleDriver	ds.jdbc.driver	N

2. Open an SSH connection to any node as the opc user, then change to the oracle user.
3. Run the script create_atp_datasource.sh in one of the following ways:
   - Execute and follow user prompts:

     ```
     /opt/scripts/utils/create_atp_datasource.sh
     ```

     At each prompt, enter a value or press Enter to accept the default value. The following is an example of the script response after entering the values:

     ```
     INFO: Found wallet config file
     INFO: Verifying existing datasources.
     INFO: Verified that no existing data source has the same name.

     INFO: Created datasource configuration file /tmp/.ds_config
     INFO: Creating the datasource ==> datasource1
     INFO: Connecting to the admin server [t3://servicename-wls-0:7001]...
     INFO: Adding properties to datasource
     INFO: Target Type : Server
     INFO: Targets : servicename_server_1
     ```

```
INFO: Setting targets
[[com.bea:Name=servicename_server_1,Type=Server]]
INFO: Successfully create datasource [datasource1]
INFO: Validating the Datasource [datasource1]
INFO: Verify datasource on Server AdminServer
-- Datasource datasource1 not found on server AdminServer.
INFO: Verify datasource on Server servicename_server_1
-- datasource1:      State[Running] Connection Test is OK
```

- Execute using a properties configuration file:

  ```
  /opt/scripts/utils/create_atp_datasource.sh
  <path_to_configuration_file_and_file_name>
  ```

  For example:

  ```
  /opt/scripts/utils/create_atp_datasource.sh /u01/.ds_config
  ```

  The following is an example of the script response:

  ```
  INFO: Creating the datasource ==> datasource1
  INFO: Connecting to the admin server [t3://servicename-
  wls-0:7001]...
  INFO: Adding properties to datasource
  INFO: Target Type : Server
  INFO: Targets : servicename_server_1,servicename_server_2
  INFO: Setting targets
  [[com.bea:Name=servicename_server_1,Type=Server,
  com.bea:Name=servicename_server_2,Type=Server]]
  INFO: Successfully create datasource [datasource1]
  INFO: Validating the Datasource [datasource1]
  INFO: Verify datasource on Server servicename_server_2
  -- datasource1:  State[Running] Connection Test is OK
  INFO: Verify datasource on Server servicename_server_3
  -- Datasource datasource1 not found on server
  servicename_server_3.
  INFO: Verify datasource on Server servicename_server_1
  -- datasource1:  State[Running] Connection Test is OK
  INFO: Verify datasource on Server AdminServer
  -- Datasource datasource1 not found on server AdminServer.
  ```

Create a Data Source for a DB System Database

Use the WebLogic Server Administration Console to create a data source and establish a connection with an Oracle Cloud Infrastructure Database (DB System).

The DB System must allow your Oracle WebLogic Server for OCI domain to access the database listen port (1521 by default). 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. See Security Rules for the DB System.

Tasks:

- Verify the PDB Name
- Configure a Data Source for a DB System Database

Verify the PDB Name

You need the PDB name if you have a 12c database in your DB System.

If you did not specify a PDB name when you created the 12c database, the default PDB name could be the database name appended with `_pdb1`. For example: `<db_name>_pdb1`

To verify you have the correct PDB name:

1. Open an SSH connection to the database node.
2. Use SQL*Plus to connect to the database as the `sys` user.
3. Execute the command:

   ```
   show pdbs
   ```

Configure a Data Source for a DB System Database

You use the WebLogic Server Administration Console to create a Java Database Connectivity (JDBC) data source for the DB System database.

1. Access the WebLogic Server Administration Console.
2. In the Change Center box, click **Lock & Edit**.
3. In the Domain Structure box, expand Services (by clicking the + next to it) and click **Data Sources**.
4. On the Summary of JDBC Data Sources page , click **New**, and then select **Generic Data Source**.
5. On the first page of the Create a New JDBC Data Source wizard, do the following:
   a. In the **Name** field, enter a name for your data source.
   b. In the **JNDI Name** field, enter a name.
   c. In the **Database Type** dropdown list, accept the default value **Oracle**.
6. On the second page of the wizard, in the **Database Driver** dropdown list, select the value that's appropriate for your domain version.

   For example, select ***Oracle's Driver (Thin) for Service connections; Versions:Any** for a 12c domain.
7. On the third page of the wizard, accept all options.
8. On the fourth page of the wizard, do the following:
   a. Enter the **Database Name** and **Host Name**. Use the appropriate format for your database version and type.

Database Version	Database Type	Database Name	Host Name
12c	VM	*<pdb_name>.<db_domain>*	*<db_hostname>-*scan.*<db_domain>*
12c	Bare Metal	*<pdb_name>.<db_domain>*	*<db_hostname>.<db_domain>*

b. In the **Port** field, enter the database listen port number (`1521` by default).

c. In the **Database User Name** field, enter a user who has been granted access to the database.

d. In the **Password** field, enter the password for the database user and re-enter the password in **Confirm Password** field.

9. On the fifth page of the wizard, verify that the **URL** has the appropriate format based on your database version and type:

Database Verion	Database Type	URL
12c	VM	`jdbc:oracle:thin:@//`*<db_hostname>-*`scan.`*<db_domain>*`:`*<db_port>*`/`*<pdb_name>.<db_domain>*
12c	Bare Metal	`jdbc:oracle:thin:@//`*<db_hostname>.<db_domain>*`:`*<db_port>*`/`*<pdb_name>.<db_domain>*

10. Click **Test Configuration** to verify if a connection to the database can be established based on the information that you provided.

    - If the connection test fails, click **Back** and review the entries that you made for the data source and correct any errors. For example, make sure there is a `/` after the port number. If there are no errors in the entries and the test still fails, make sure that your database is running.
    - The connection is successful when the message `Connection test succeeded` is displayed.

11. On the last page of the wizard, select the target for the JDBC data source and then click **Finish**.

12. In the Change Center, click **Activate Changes**.

Connect to a Domain Using Oracle JDeveloper

Before you can use Oracle JDeveloper to connect to an Oracle WebLogic Server for OCI domain, add an ingress rule to restrict Internet access to a new network channel, and then create a network channel that supports the T3S protocol to enable remote connections.

> **Note:**
>
> - Oracle recommends you use Oracle Cloud Infrastructure FastConnect and private IP addresses to set up the connection between Oracle JDeveloper and your Oracle WebLogic Server for OCI domain. See FastConnect Overview.
> - Install Oracle JDeveloper on a separate compute instance in the same VCN as the WebLogic subnet. This allows access from the JDeveloper instance to the ports on the WLS subnet.

Topics:

- Add an Ingress Rule
- Create a Network Channel
- Create a Connection to the Domain Through Oracle JDeveloper

Add an Ingress Rule

Before you establish a network channel that supports the T3 (and secure T3) protocol, use the Oracle Cloud Infrastructure Console to create an ingress rule to restrict access from the Internet.

> **Note:**
>
> RMI communications in Oracle WebLogic Server use the T3/T3S protocol to transport data between WebLogic Server and other Java programs, including clients. When using a port that supports T3/T3S, Oracle recommends you limit access only to known clients.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the domain instance that has the Administration Server node.

   The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
5. Click the **Subnet** value.
6. Click the name of the default security list.
7. Click **Add Ingress Rules**.
8. In the **Source CIDR** field, do one of the following:
   - For a domain in a public subnet, you must limit access to the T3S port by using a restricted CIDR matching the IP range for the running JDeveloper instances.

- For a domain in a private subnet, you can enter `0.0.0.0/0` or set up a security rule using the IP range for the running JDeveloper instances.

9. In the **Destination Port Range** field, enter `8002`.
10. Click **Add Ingress Rules**.

Create a Network Channel

When you use Oracle WebLogic Server for OCI to create a domain, the administration server is configured with network channels that do not support the T3 and T3S protocols. To connect to the administration server with tools that use the T3 and T3S protocols, you must create a new network channel.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. From the **Compartment** dropdown, select the compartment in which your domain is created.
4. Click the name of the domain instance that has the Administration Server node.

   The instance with the Administration Server node has `wls-0` appended to the name. For example: `abcde7xy-wls-0`
5. Copy the **Public IP Address** value.
6. Copy the **Private IP Address** value.
7. Copy the **Internal FQDN** value.
8. In a browser, access the WebLogic Server Administration Console using the public IP address.
9. In the Change Center box, click **Lock & Edit**.
10. In the Domain Structure box, expand **Environment** (by clicking the + next to it) and click **Servers**.
11. Click the name of the Administration Server node.
12. On the Settings page for the node, click **Protocols** and then click **Channels**.
13. Under Network Channels, click **New**.
14. On the first page of the Create a New Network Channel wizard, do the following:
    a. In the **Name** field, enter a name for the channel.
    b. In the **Protocol** dropdown list, select **t3s**.
15. On the second page of the Create a New Network Channel wizard, do the following:
    a. In the **Listen Address** field, enter the Internal FQDN value you copied.
    b. In the **Listen Port** field, enter `8002`.
    c. In the **External Listen Address** field, enter the Private IP Address value you copied.
    d. In the **External Listen Port** field, enter `8002`.
16. Click **Finish**.

17. In the Change Center, click **Activate Changes**.

Create a Connection to the Domain Through Oracle JDeveloper

After creating the ingress rule and network channel, establish and test a connection between Oracle JDeveloper and your Oracle WebLogic Server for OCI domain.

You'll need to provide the name of your Oracle WebLogic Server for OCI domain.

1. Open the Oracle JDeveloper Console.
2. Open the Create Application Server Connection dialog.
3. On the Configuration page, enter the following:
    a. For **WebLogic Hostname (Administration Server)**, enter the private IP address of the compute instance that runs the WebLogic Administration Server.
    b. For **Port**, enter `0`.
    c. For **SSL Port**, enter `8002`.
    d. Select the **Always Use SSL** check box.
    e. For **WebLogic Domain**, enter your WebLogic Server domain name in Oracle Cloud Infrastructure. The name has the suffix `_domain`. For example: `thestack_domain`
4. Ensure **No Proxy** is selected in your Web Browser and Proxy setting.
5. Test the application server connection in JDeveloper.

Upgrade a Domain

For an existing Oracle WebLogic Server for OCI domain, you can upgrade the WebLogic Server release from 12c (12.2.1.3) to WebLogic Server release 12c (12.2.1.4).

Following are the advantages of this documented process over the traditional in-place upgrade:

- The stack has the latest scripts on disk volumes.
- There is an easier roll back process. Since the original stack with 12.2.1.3 is still available, only the database has to be rolled back for a JRF instance.
- The WebLogic binaries are the latest. You do not have to locate the WebLogic installers and apply the required patches.

Topics:

- Backup the Database
- Create a 12.2.1.4 Instance
- Stop the WebLogic Server Processes on the Source 12.2.1.3 Instance
- Replace the Domain on the Target 12.2.1.4 Instance with Cloning
- Set up VNC Server
- Perform Readiness Check
- Upgrade Infrastructure Schemas
- Reconfigure the Domain
- Upgrade the Domain

- Restart Servers
- Post Upgrade
- Roll Back Upgrade

Backup the Database

1. Complete the *Create an on-demand full backup of a database* procedure in Backing Up a Container Database to Oracle Cloud Infrastructure Object Storage in the Oracle Cloud Infrastructure documentation.
2. Return here to continue with the next step.

Create a 12.2.1.4 Instance

1. Create a 12.2.1.4 instance, which is identical to the 12.2.1.3 instance that you plan to upgrade.

   Ensure that you provide a unique name to the 12.2.1.4 instance and use the same networking as the 12.2.1.3 instance. See Create a Stack.

   > **Note:**
   >
   > If you are using a reserved IP with your load balancer, then delete the load balancer on the source instance to free up the reserved IP address.

2. Return here to continue with the next step.

Stop the WebLogic Server Processes on the Source 12.2.1.3 Instance

1. Log in to the source instance as an `opc` user.
2. Run the following command:

   ```
   sudo su - oracle
   /opt/scripts/restart_domain.sh -o stop
   ```

   Run the following command to confirm if there are any running processes:

   ```
   jps
   ```

   If there are any processes running, then run `kill -9` against each of the processes.
3. Log in to each of the non-administration compute instance as an `opc` user.
4. Run the following command:

   ```
   sudo su - oracle
   /opt/scripts/restart_domain.sh -o stop
   ```

Run the following command to confirm if there are any running processes:

```
jps
```

If there are any processes running, then run `kill -9` against each of the processes.

5. Log in to administration compute instances as an `opc` user.
6. Run the following command:

```
sudo su - oracle
/opt/scripts/restart_domain.sh -o stop
```

Run the following command to confirm if there are any running processes:

```
jps
```

If there are any processes running, then run `kill -9` against each of the processes.

Replace the Domain on the Target 12.2.1.4 Instance with Cloning

1. Clone the source 12.2.1.3 domain contents. See Clone a JRF or non-JRF Instance using a Script.

> **Note:**
>
> - Clone only the data block volumes. *Do not* use to the `-m` argument to clone the Middleware volumes.
> - Cloning might fail at the `StartServers` step, as your applications may not be upgraded to handle 12.2.1.4 binaries. If this error occurs, ignore the error and continue with the next step.
>   Run the following command:
>
>   ```
>   python3 create_clone.py -p StartAppGateway
>   ```

2. After you have run the cloning script on all the VMs, stop the WebLogic servers by running the following command on each VM:

```
sudo su - oracle
/opt/scripts/restart_domain.sh -o stop
```

Run the following command to confirm if there are any running processes:

```
jps
```

If there are any processes running, then run `kill -9` against each of the processes.

3. Are you upgrading a JRF instance?

- Yes: Continue with the next procedure.
- No: Go to Restart Servers.

Set up VNC Server

To use the Reconfiguration Wizard and Upgrade Assistant (Fusion Middleware tools) during the upgrade process, you need a graphical user interface (GUI) environment. The instructions in this step explain how to set up a VNC server and use port forwarding through a bastion host. If you are familiar with using X11, then X11 forwarding can be used to forward the GUI to your local desktop and you can skip this step.

> **Note:**
>
> This step is *not applicable* for a non-JRF instance.

1. Log in to administration compute instance as an `opc` user.
2. Run the following command to install the GUI packages on the target 12.2.1.4 instance:

   ```
   sudo bash
   yum group install "Server with GUI"
   # enter 'y' when prompted.
   ```

3. As an `opc` user, run the following command to set up the VNC server for the `oracle` user on the administration compute instance:

   ```
   sudo bash
   yum install tigervnc-server -y
   exit
   sudo su - oracle
   vncpasswd
   # enter the password and confirmation password.
   # respond with "n" when prompted if this should be a view only
   password
   exit
   sudo bash
   cp /lib/systemd/system/vncserver@.service /etc/systemd/system/
   vncserver@\:1.service
   vi /etc/systemd/system/vncserver@\:1.service
    Replace <USER> with oracle
   systemctl daemon-reload
   systemctl enable vncserver@\:1.service
   systemctl start vncserver@\:1.service
   systemctl status vncserver@\:1.service
   # Confirm it is running

   # Set up the firewall to allow the VNC server port to be accessed:
   iptables -I INPUT -m state --state NEW -p tcp --destination-port
   5901 -j ACCEPT
   ```

4. Sign in to the Oracle Cloud Infrastructure Console and update the subnet to have a security list that enables inbound access to the VNC server port `5901`.

> **Note:**
>
> If you have not already set up your bastion VM to be able to access VMs as the `opc` user, then place your private key pem for the opc user on the disk. This is used for port forwarding.

5. Create the tunnel by accessing the bastion host as the `opc` user:

```
ssh -i <privatekey.ppk> -L <vnc_port_on_wlsm-private-ip>:<wlsvm-private-
ip>:<port_on_bastion> <wlsvm-private-ip>
```

Following is an example, where the IP address of the administration compute instance is `10.1.1.1` and the private key for the `opc` user is in `~/.ssh/id_rsa`

```
ssh -i ~/.ssh/id_rsa -L 5901:10.1.1.1:5901 10.1.1.1
```

Windows instructions for launching GUI

1. Install and launch PuTTY.
2. For **Host Name**, type the bastion IP address.
3. For **Saved Sessions**, type `bastion`.
4. Under **Category**, go to **Connection** > **Data**.
5. For **Auto-logion username**, type `opc`.
6. Under **Category**, go to **Connection** > **Data** > **SSH** > **Tunnels**.
7. Type the following values for the respective fields:
   - Source port: `5901`
   - Destination: `localhost:5901`
8. Click **Add**.
9. Under **Category**, go to **Connection** > **Data** > **SSH** > **Auth**.
10. For **Private key file for authentication**, browse and select the xperiment private key that you have created.
11. Under the **Category**, select **Session**.
12. Select **Save** and then select **Open** to establish the connection
13. Verify that you connected successfully to the putty session.
14. Install a VNC Viewer and set up a new connection to use `localhost:5901` to verify that you can connect correctly.

> **Note:**
>
> Ensure that you have set up the vncserver as the oracle user, as this creates a session with the oracle user even though you have port forwarding via the opc user ssh keys.

Perform Readiness Check

Perform a readiness check to determine if your service instance is ready for upgrade.

> **Note:**
>
> This step is *not applicable* for a non-JRF instance.

1. By using the VNC viewer session, start the Upgrade Assistant.

   ```
   export USER_MEM_ARGS=-Djava.security.egd=file:/dev/urandom
   /u01/app/oracle/middleware/oracle_common/upgrade/bin/ua -readiness
   ```

   Setting USER_MEM_ARGS to use the /dev/urandom device reduces the time it takes to run the Oracle Fusion Middleware upgrade tools.

2. Use the Upgrade Assistant to perform a readiness check. See Upgrade from a previous 12c release to 12.2.1.4 in *Upgrading to the Oracle Fusion Middleware Infrastructure*

3. On the **Readiness Check Type** screen, select the domain-based readiness check.

   The domain-based readiness check enables the Upgrade Assistant to discover and select all upgrade-eligible schemas or component configurations in the domain specified in the Domain Directory field.

4. On the **End of Readiness** screen in the Upgrade Assistant, review the results of the readiness check (Readiness Success or Readiness Failure).

   - If the readiness check is successful, click **View Readiness Report** to review the complete report. Oracle recommends that you review the Readiness Report before you perform the upgrade even when the readiness check is successful. Use the **Find** option to search for a particular word or phrase within the report. The report also indicates where the completed Readiness Check Report file is located.

   - If the readiness check encounters an issue or error, click **View Log** to review the log file, identify and correct the issues, and then restart the readiness check.

Upgrade Infrastructure Schemas

This step helps to identify if you have an earlier version of infrastructure database schemas or have installed other Oracle products.

> **Note:**
>
> This step is *not applicable* for a non-JRF instance.

1. Start the Upgrade Assistant if you have not already done so. For example:

   ```
   export USER_MEM_ARGS=-Djava.security.egd=file:/dev/./urandom
   /u01/app/oracle/middleware/oracle_common/upgrade/bin/ua
   ```

2. Upgrade the schemas. See Upgrading Schemas Using the Upgrade Assistant in *Upgrading to the Oracle Fusion Middleware Infrastructure*:
3. On the Selected Schemas screen, select **All Schemas Used by a Domain**, and then enter a domain directory name in the **Domain Directory** field.

   The **All Schemas Used by a Domain** selection allows the Upgrade Assistant to discover and select all components that have a schema available to upgrade in the domain specified in the **Domain Directory** field. This is also known as a domain-assisted schema upgrade. In addition, the Upgrade Assistant prepopulates connection information on the schema input screens.
4. On the Upgrade Progress screen in the Upgrade Assistance, monitor the schema upgrade progress.
5. Finish the schema upgrade process.
   - If the schema upgrade succeeds, click **Close** to complete the upgrade and close the wizard.
   - If the upgrade fails, click **View Log** to view and troubleshoot the errors. The logs are available in the following directory:

     ```
     /u01/app/oracle/middleware/oracle_common/upgrade/logs
     ```

6. Remedy the database connection failure if one occurs. See Problems with Database Connectivity When Upgrading the Infrastructure Schema Database in *Administering Oracle Java Cloud Service*.
7. Verify the schema upgrade was successful by checking that the schemas in `schema_version_registry` have been properly updated. See Problems with Database Connectivity When Upgrading the Infrastructure Schema Database in *Upgrading to the Oracle Fusion Middleware Infrastructure*.

   One way to verify the schema upgrade is to use SQL*Plus commands to obtain data from the `SCHEMA_VERSION_REGISTRY`.
   1. Find the Oracle Java Cloud Service instance's schema prefix in the Upgrade Assistant log file at `/u01/app/oracle/middleware/oracle_common/upgrade/logs`.

b. Connect to the database as a user having Oracle DBA privileges and run the following commands from SQL*Plus to get the current version numbers.

```
sqlplus / as sysdba
SQL> connect <user_name>/<password>@<host_name>:<port>/
<service_name> as sysdba
SQL> SELECT MRC_NAME,COMP_ID,OWNER,VERSION,STATUS,UPGRADED FROM
SCHEMA_VERSION_REGISTRY WHERE MRC_NAME like 'SP1556690734';
```

Example output for the `SCHEMA_VERSION_REGISTRY`:

MRC_NAME	COMP_ID	OWNER	VERSION	STATUS	UPGRADED
SP1556690734	IAU	SP1556690734_IAU	12.2.1.2.0	VALID	Y
SP1556690734	IAU_APPEND	SP1556690734_IAU_APPEND	12.2.1.2.0	VALID	N
SP1556690734	IAU_VIEWER	SP1556690734_IAU_VIEWER	12.2.1.2.0	VALID	Y
SP1556690734	MDS	SP1556690734_MDS	12.2.1.3.0	VALID	Y
SP1556690734	OPSS	SP1556690734_OPSS	12.2.1.0.0	VALID	Y
SP1556690734	STB	SP1556690734_STB	12.2.1.3.0	VALID	Y
SP1556690734	UCSUMS	SP1556690734_UMS	12.2.1.0.0	VALID	N
SP1556690734	WLS	SP1556690734_WLS	12.2.1.0.0	VALID	N

Reconfigure the Domain

> **Note:**
>
> - This step is *not applicable* for a non-JRF instance.
> - Running the reconfiguration wizard is not required for an upgrade from 12.2.1.3. However, if it is not run and you do not replace 12.2.1.3 with 12.2.1.4 in config.xml, then you will encounter the Incorrect Version Numbers After a Reduced Downtime Upgrade issue.

1. Start the Reconfiguration Wizard as user `oracle` with the following logging options, with `log_file` as the absolute path of the log file you'd like to create for the domain reconfiguration session. This can be helpful if you need to troubleshoot the reconfiguration process.

For example:

```
/u01/app/oracle/middleware/oracle_common/common/bin/reconfig.sh -
log_priority=all -log="/u01/reconfig0212.log"
```

2. Perform the reconfiguration tasks as described in *Upgrading to the Oracle Fusion Middleware Infrastructure*. See Reconfiguring the Domain with the Reconfiguration Wizard.
3. On the Advanced Configuration screen of the Reconfiguration Wizard, select **Deployment and Services**.
4. Target the `wsm-pm` app to the cluster containing the managed servers.
5. Click **Reconfig**.
6. Check the End of Configuration screen to learn whether the reconfiguration process completed successfully or failed.
    - If the reconfiguration is successful, **Oracle WebLogic Server Reconfiguration Succeeded** is displayed. The location of the domain that was reconfigured as well as the Administration Server URL (including the listen port) are displayed as well.
    - If the reconfiguration process did not complete successfully, an error message is displayed which indicates the reason. Take appropriate action to resolve the error.

Upgrade the Domain

> **Note:**
>
> This step is *not applicable* for a non-JRF instance.

1. Start the Upgrade Assistant, for example:

```
export USER_MEM_ARGS=-Djava.security.egd=file:/dev/./urandom
/u01/app/oracle/middleware/oracle_common/upgrade/bin/ua
```

2. Use the Upgrade Assistant to upgrade the domain configurations. See Upgrading the Domain Configurations with the Upgrade Assistant in *Upgrading to the Oracle Fusion Middleware Infrastructure*.
3. On the All Configurations screen, select **All Configurations Used by a Domain** and specify your domain location in the **Domain Directory** field. Enter the domain directory directly or click **Browse** to select a valid domain directory.
4. On the Upgrade Summary page, review the summary of the options you have selected for the component configuration upgrade, and then click **Upgrade** to start the upgrade process.
5. View the Upgrade Progress page to monitor the upgrade.
6. View the results and finish the upgrade.
    - If the upgrade succeeds, the Upgrade Success page is displayed. Click **Close** to complete the upgrade and close the wizard.

- If the upgrade fails, the Upgrade Failure screen is displayed. Click **View Log** to view and troubleshoot the errors. The logs are available at

  ```
  ORACLE_HOME/oracle_common/upgrade/logs
  ```

Restart Servers

Access the 12.2.1.4 instance as the `opc` user, and on each VM run the following command:

```
sudo su - oracle
/opt/scripts/restart_domain.sh
```

Post Upgrade

If the upgrade was successful, complete following tasks if they apply to your instance:

- If you use a Hosting Provider to manage DNS, then reset the `CNAME` records at your Hosting Provider to point to the new IP addresses of the load balancer and WebLogic VMs.
- Destroy the source 12.2.1.3 instance. See Destroy Stack Resources.

  > **WARNING:**
  >
  > Do not Delete a JRF Database Schema or Delete the Identity Cloud Service Resources as these resources are required in the upgraded cloned instance.

- Run the following commands to remove the UI libraries. This stops the VNC server, removes the VNC server package, and removes the `Server with GUI` group packages.

  ```
  # Remove VNC server
  sudo su - oracle
  vncserver -list
  # Locate the X Display value and kill this - typically this is :1
  vncserver -kill :1
  # Remove the Linux service
  exit
  sudo bash
  systemctl stop vncserver@\:1.service
  systemctl disable vncserver@\:1.service
  # Uninstall package
  yum remove tigervnc-server -y
  # Remove GUI
  yum group remove "Server with GUI"
  ```

Roll Back Upgrade

If the upgrade fails, you can roll back the upgrade. Since the source 12.2.1.3 instance is still present, only the database needs to be rolled back if you upgraded a JRF domain.

1. Complete the *Restore a database using a specific backup from Object Storage* procedure in Recovering a Container Database from Object Storage in the Oracle Cloud Infrastructure documentation.
2. If you use a reserved IP address on the load balancer, then you must delete the load balancer on the source instance prior to creating the 12.2.1.4 instance.
   a. Delete the load balancer or destroy the 12.2.1.4 stack to release the reserved IP address. See Destroy Stack Resources.
   b. Create a load balancer in Oracle Cloud Infrastructure again using the reserved IP address. See Configure SSL for a Domain.

4

Scale a Stack

Resize your Oracle WebLogic Server for OCI domain to meet changing workload requirements.

> **Note:**
>
> If you have moved your password to new secret or vault or updated the password, then you must update the password secret OCID and policy to read the secrets with new secrets OCID. See Update the Password Secret OCID and Policy.

Topics

- Add or Remove WebLogic Server Nodes
- Change the Shape of the Existing Compute Instances
- Add a Load Balancer
- Remove the Load Balancer
- Add a File Storage
- Remove the File Storage

Add or Remove WebLogic Server Nodes

You can change the number of nodes (compute instances) in your Oracle WebLogic Server for OCI stack to increase performance or to reduce costs. Add nodes to scale out, or remove nodes to scale in.

> **Note:**
>
> You cannot use this procedure to scale a domain that was created before June 29, 2020.

To scale the domain, edit the node count variable for the stack and then run an Apply job.

Oracle WebLogic Server for OCI performs these tasks for a scaling operation:

- Add or remove compute instances
- Add or remove managed servers in the domain configuration (optional)
- Scale up or scale down the new compute instances (optional)
- Add managed servers to the existing cluster in the domain configuration, if the domain is not running Oracle WebLogic Server Standard Edition (optional)

- Update the backend set of the load balancer, if your stack includes a load balancer (optional)

If you customized your domain configuration after creating it (changing port numbers, changing server names, and so on), there is no guarantee that the domain modification part of the scaling job will succeed. You can disable this feature and perform these tasks manually after the scaling job completes.

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. Click **Variables**.
6. Click **Edit Variables**.
7. Edit **WebLogic Server Node Count**, and then change the shape of the new compute instances or use the default shape.

   For the compute instance shape:

   - If the default shape is a flexible shape and you choose to use this default shape, you can edit the OCPU count for the additional instances.

     > **Note:**
     >
     > If you do not increase the node count and edit the OCPU count for any of the flexible shapes, the OCPU count remains unchanged.

   - If you select the standard shape for the new compute instance, you cannot specify the OCPU count for the instance.

   If you increase the node count and modify the **SSH Public Key**, the key will be used for new compute instances only. The keys for existing compute instances remain unchanged.
8. Optional: If you have updated the WebLogic Server administrator password, then enter the OCID of the secret that contains the password for the WebLogic Server administrator. See Create Secrets for Passwords.
9. Optional: If you want Oracle WebLogic Server for OCI to create compute instances but not update your domain configuration, select **Do Not Update Domain Configuration For Scale Out**.
10. Click **Next**.
11. Click **Save Changes**.
12. Click **Terraform Actions**, and then click **Apply**.
13. Enter a name for the job, and then click **Apply**.
14. Periodically monitor the progress of the Apply job until it is finished.
15. Click **Outputs**.

16. Locate `WebLogic_Instances`, and verify the number of compute instances in the updated stack.

If you selected **Do Not Update Domain Configuration For Scale Out**, then you must manually update your domain configuration and add the managed servers. Use the WebLogic Server Administration Console or the WebLogic Server Scripting Tool (WLST).

If your domain is in a private subnet, the bastion compute instance is deleted and recreated. As a result, the bastion might have a different IP address. See Access the WebLogic Console in a Private Subnet.

If you scale out a domain created after June 24th, 2021, any OPatches you added after provisioning are automatically applied to the new virtual machines.

For domains that were created before June 25th, 2021 or the domains on which the scaling operation did not successfully add a new WebLogic managed server, perform the binary synchronization step manually on the new added virtual machines using the following commands:

```
adminHost=$(python3 /opt/scripts/databag.py wls_admin_host)

# Step 1: rsync middleware
rsync -e "ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
LogLevel=ERROR" -a --delete --timeout 60 /
--exclude=user_projects --exclude=logs oracle@$adminHost:/u01/app/oracle/
middleware/ /u01/app/oracle/middleware
# Step 2: rsync jdk
rsync -e "ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
LogLevel=ERROR" -a --delete --timeout 60 /
--exclude=user_projects --exclude=logs oracle@$adminHost:/u01/app/
oracle/jdk/ /u01/app/oracle/jdk
```

Add UCM WebLogic Server Node to a BYOL Stack

You can add Universal Credits (UCM) WebLogic server nodes to a Bring Your Own License (BYOL) stack in your Oracle WebLogic Server for OCI stack.

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. Click **Variables**.
6. Click **Edit Variables**.
7. From the **Image for Scale Out** dropdown, select the required WebLogic server image for scale out.

> **Note:**
>
> - If you have a BYOL stack you can add either BYOL or UCM nodes. If you have a UCM stack you can add only UCM nodes.
> - If you have a BYOL stack with UCM nodes, to add BYOL nodes you must scale in the UCM nodes before adding BYOL nodes.
> - If you have a BYOL stack that is enabled for autoscaling, you can scale out the stack with UCM nodes or BYOL nodes.
> - If you select a BYOL image, it requires a WebLogic License with a valid support contract. If you select a UCM image, it is charged per OCPU per hour for the entitlement and WebLogic support.

8. Select **Terms of use**.
9. Click **Next**.
10. Click **Save Changes**.
11. Click **Terraform Actions**, and then click **Apply**.
12. Enter a name for the job, and then click **Apply**.
13. Periodically monitor the progress of the Apply job until it is finished.

Manage Autoscaling Resources

If you created an Oracle WebLogic Server for OCI with autoscaling enabled, you can create and update the Alarm Definition, configure the parameters of the scaling functions, and reenable autoscaling for a stack.

If you have not enabled autoscaling for your WebLogic instance, see Configure Autoscaling.

Topics:

- Create Alarm Definitions
- Update Alarm Definitions
- Configure Function Application
- Reenable Autoscaling for a Stack

Create Alarm Definitions

Alarms are used to push messages to configured destinations. You can create custom alarms to receive notifications for WebLogic domain metrics, in addition to the Scale Out Alarm Definition and Scale In Alarm Definition that are created when autoscaling is enabled during provisioning.

For more information on alarm definitions, see Autoscaling.

To create the alarm definitions:

1. Access the Oracle Cloud Infrastructure console.

2. From the navigation menu, select **Observability & Management**. Under **Monitoring**, click **Alarm Definitions**.
3. Click **Create Alarm**.
4. Under **Define alarm**, enter an **Alarm name** and select the **Alarm severity**.
5. Under **Metric description**, select the **Compartment** where the Application Performance Monitoring domain is located, select `oracle_apm_monitoring` **Metric namespace**, and then for **Metric name**, select a WebLogic metric.

   For monitoring concepts, metric namespaces, and the default Application Performance Monitoring metrics, see Monitoring Concepts and Application Performance Monitoring Metrics in Oracle Cloud Infrastructure documentation.
6. Under **Metric dimensions**, do the following:
   a. For **Dimension name**, select *AppserverClusterName* and for **Dimension value**, select the WebLogic cluster created by the stack.
   b. Select **Aggregate metric streams** to return the combined value of all metric streams for the selected statistic.
7. Under **Trigger rule**, specify the **Operator**, **Value**, and **Trigger delay minutes**.

   The default value for **Trigger delay minutes** of the alarm definition is five minutes, which is the number of minutes that the condition is maintained before the alarm state changes from "Ok" to "Firing". You can specify the number of minutes that the condition must be maintained before the alarm is in firing state.
8. Under **Destinations**, select the following:
   - For **Destination service**, select *Notification Service*.
   - For **Compartment**, select your stack compartment.
   - For **Topic**, select the notification topic for scale out or scale in.
9. Click **Save alarm**. The new alarm is listed on the Alarm Definitions page.

   See Create Alarms in the Oracle Cloud Infrastructure documentation.

Update Alarm Definitions

You can update alarm definitions if you want to change the number of alarm definitions and the settings such as threshold value and alarm metrics that are configured when autoscaling is enabled during provisioning.

To update the alarm definitions:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Observability & Management**. Under **Monitoring**, click **Alarm Definitions**.
3. Click the name of the alarm definition.
4. Click **Actions**, and click **Edit alarm**.
5. Under **Define alarm**, **Metric description**, and **Metric dimensions**, update the alarm settings as needed.
6. Under **Trigger rule**, update the default settings for **Value** and **Trigger delay minutes**.

> **Note:**
>
> The default value for **Trigger delay minutes** of the alarm definition is five minutes, which is the number of minutes that the condition is maintained before the alarm state changes from "Ok" to "Firing". You can specify the number of minutes that the condition must be maintained before the alarm is in firing state.

7. Under **Notifications**, update the default settings for **Notification frequency**.

   > **Note:**
   >
   > You can enter the **Notification frequency** only if **Repeat notification?** is selected. The default period of time to wait before resending the notification is 20 minutes for scale out and 30 minutes for scale in.

   See Update Alarm in the Oracle Cloud Infrastructure documentation.

Configure Function Application

In autoscaling, you can use functions to invoke the Resource Manager APIs to scale the WebLogic Server domain in Oracle WebLogic Server for OCI. If you want to edit the Function Application configuration that is set when autoscaling is enabled during provisioning, you can configure the parameters of the Function Application.

To know the function application that are created during autoscaling, see Autoscaling.

To configure the parameters of the function application:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Developer Services**. Under **Functions**, click **Applications**.
3. Select the **Compartment** that contains your stack.
4. Click the name of the application.
5. Under **Resources**, click **Configuration**.
6. Update the following parameters:
   - *offline_ms1_from_lb* - Set this parameter to *true,* if you want the managed server coresiding on the backend of the administration node to be set offline temporarily. This setting is required when the threshold condition for metric is breached and when the administration server is unable to configure the node on the cluster as the WebLogic Server domain is overloaded.
     After the managed server is set to offline, the managed server is suspended (server changes from "Running" state to "Admin" state), and it only accepts administrative requests. As a result, the administration server has CPU time to activate the domain configuration on the new node as no traffic (requests) is sent to managed server during scale out. After stack apply job is complete, the managed server backend is set to online (server changes from "Admin" to "Running" state).

> **Note:**
>
> By default, *offline_ms1_from_lb* is set to false.

- *debug* - Set this parameter to *true* to enable debug logging for functions.
- *min_wls_node_count* - Provide the minimum WebLogic node count for scale in so that the scaling happens up to the *min_wls_node_count* value.
  By default, *min_wls_node_count* is set to the number of WebLogic nodes specified during initial stack provisioning. For example, if you selected two nodes during initial stack provisioning, the *min_wls_node_count* is set to two.

  You must not edit the parameters, *stack_ocid* and *wlsc_email_notification_topic_id*.

At the function-level, you can add a parameter to update and override any inherited parameter value, the selected function has inherited from the Function Application configuration.
To override the parameter values of the function application:

1. Under **Resources**, click **Functions**.
2. Click the name of the function.
3. Under **Resources**, click **Configuration**.
4. If you want to disable autoscaling, set the **use_autoscaling** parameter to *None*.

Reenable Autoscaling for a Stack

If the scale in or scale out fails, you receive an email notification with the job status and information about the tag **disabled_for_autoscaling:true** enabled for the stack.

You can check the Scale Apply job logs in the Oracle Cloud Infrastructure console and the Function logs to identify the cause of the failure. After you fix the issue, reenable autoscaling by setting the tag **disabled_for_autoscaling** to false or removing the tag from the stack.

To reenable autoscaling for the stack:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Developer Services**. Under **Resource Manager**, click **Stacks**.
3. Select the **Compartment** that contains your stack.
4. Click the name of your stack.
5. Click the **Tags** tab and do one of the following:
   - Set **disabled_for_autoscaling** to *false*.
   - Click **Remove Tag** to remove the tag from the stack.

Change the Shape of the Existing Compute Instances

You can scale up the existing compute resources for your Oracle WebLogic Server for OCI domain to increase performance, or you can scale down the existing compute resources to reduce costs.

> **Note:**
>
> Do not use Resource Manager to change the shape of the compute instances in your domain. You must use the Compute service.

When you change the shape of an existing compute instance, you select a different processor, number of cores, amount of memory, network bandwidth, and maximum number of VNICs for the instance. The instance's public and private IP addresses, volume attachments, and VNIC attachments remain the same. For example, changing the shape of an instance from `VM-Standard2.2` to `VM-Standard2.4` doubles the capacity of the node from two OCPUs to four OCPUs, and also doubles the amount of memory allocated to the node.

The original shape of the compute instance determines which shapes you can select as a target for the new shape. You cannot modify the shape of an instance that uses a bare metal shape or certain virtual machine (VM) shapes. See Changing the Shape of an Instance.

Oracle recommends that you use the same shape for all compute instances that comprise a single WebLogic Server cluster. This allows traffic to be distributed uniformly across the cluster.

When you change a shape, the compute instance must be restarted. To avoid downtime and ensure your applications remain available to users, Oracle recommends that you create a cluster with multiple compute instances, and that you change the shape for one compute instance at a time.

When you change the shape of the first compute instance, the domain's administration server will be temporarily unavailable. However, your applications do not depend on the administration server and will not be affected.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu ☰, select **Compute**. Under the **Compute** group, click **Instances**.
3. Select the **Compartment** in which your domain was created.
4. Click the name of the first compute instance for your domain.

   The instance has `wls-0` appended to the name. For example: `mydomain-wls-0`
5. Click **Edit**, and then select **Edit Shape**.
6. Select a new shape, and then click **Save Changes** > **Reboot Instance**.
7. Repeat from **step 2** for the remaining compute instances in your domain.

Change Reserved Public IP Usage

You can update an existing Oracle WebLogic Server for OCI domain to either use a bastion compute instance with a reserved IP or remove the assigned reserved IP.

Edit the reserved IP variable for the stack and then run an *Apply* job.

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. Click **Variables**.
6. Click **Edit Variables**.
7. Configure Reserved IP:
   - If your want to use a bastion compute instance with a reserved public IP, then select **Assign Reserved Public IP to Bastion Instance**
   - If you have already used a bastion compute instance with a reserved public IP and *do not* want to use a reserved public IP now, then unselect **Assign Reserved Public IP to Bastion Instance**.
8. Click **Next**.
9. Click **Save Changes**.
10. Click **Terraform Actions**, and then click **Apply**.
11. Enter a name for the job, and then click **Apply**.
12. Periodically monitor the progress of the Apply job until it is finished.
13. Click **Outputs**.
14. Identify if the reserver IP address of the load balancer in the updated stack.

Add a Load Balancer

You can add a load balancer to an existing Oracle WebLogic Server for OCI domain that was originally created without a load balancer.

> **Note:**
>
> You cannot use this procedure on a domain that was created before June 29, 2020. If you enabled autoscaling for your domain and did not configure the load balancer on stack creation, you must add the load balancer when you edit the stack.

Oracle WebLogic Server for OCI configures the new load balancer to distribute application traffic to the managed servers in your domain.

Edit the load balancer variables for the stack and then run an Apply job.

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. Click **Variables**.
6. Click **Edit Variables**.
7. Select **Add Load Balancer**.
8. Configure the load balancer network.
   - If you chose to use an existing regional subnet for WebLogic Server, then select an existing regional subnet from the list of regional and availability domain-specific subnets. A load balancer can have only one regional subnet, which is shared between both nodes.

     > **Note:**
     >
     > This option is not applicable if you use an existing load balancer.

   - If you chose to create a regional subnet for WebLogic Server, then specify a CIDR for the new load balancer subnet.
9. Configure load balancer:
   - Select **Create Load Balancer**.
   - Select **Use Existing Load Balancer**.

     > **Note:**
     >
     > You can select an existing load balancer for an existing VCN and existing subnet only.
     > The existing load balancer should be in the same compartment as the stack compartment, and should be in the same VCN as the existing VCN chosen for the stack.
     >
     > a. Specify the OCID for the existing load balancer.
     >
     > b. Enter the name of the backend set for the existing load balancer that has a routing policy associated with the backend set. The backend set should not have any backends.

   - If you create a new load balancer, select **Private Load Balancer**, if you do not want to assign a public IP address to the load balancer.
     If you change the load balancer for an existing domain from public to private, or change the load balancer from private to public, then you must also change the load balancer subnet CIDR.
   - If you create a new load balancer, select **Load balancer with Reserved Public IP**, if you want to use a public load balancer with a reserved public IP. Then, specify the OCID of the public IP for the load balancer.

10. If you create a new load balancer, select a minimum and maximum flexible load balancer shape.

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

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

11. Click **Next**.
12. Click **Save Changes**.
13. Click **Terraform Actions**, and then click **Apply**.
14. Enter a name for the job, and then click **Apply**.
15. Periodically monitor the progress of the Apply job until it is finished.
16. Click **Outputs**.
17. Identify the IP address of the load balancer in the updated stack.

If your domain includes the sample application, you can access it using the load balancer. See Access the Sample Application.

Remove the Load Balancer

If you created an Oracle WebLogic Server for OCI domain with a load balancer, and no longer require the load balancer, you can remove it.

> **Note:**
>
> You cannot use this procedure on a domain that was created before June 29, 2020.

Edit the load balancer variables for the stack and then run an Apply job.

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. Click **Variables**.
6. Click **Edit Variables**.
7. Clear **Add Load Balancer**.
8. Click **Next**.
9. Click **Save Changes**.
10. Click **Terraform Actions**, and then click **Apply**.

11. Enter a name for the job, and then click **Apply**.
12. Periodically monitor the progress of the Apply job until it is finished.

Add a File Storage

You can add a file storage to an existing Oracle WebLogic Server for OCI domain that was originally created without a file storage.

> **Note:**
>
> You cannot use this procedure on a domain that was created before Release 21.3.3 (September 16, 2021). For these existing domains, you must manually add the file storage.

Edit the file storage variables for the stack and then run an Apply job.

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. Click **Variables**.
6. Click **Edit Variables**.
7. Select **Add File System Storage**.
8. Configure the file storage:
   - If you chose to create a Virtual Cloud Network (VCN):
     a. *Optional:* Select the availability domain in which you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and a new subnet:
     a. *Optional:* Select the availability domain where you want to create the file system and mount target.
     b. Specify the CIDR of the new subnet.
   - If you chose to use an existing VCN and an existing subnet:
     – If you *do not* want to use an existing mount target or an existing file system:
       a. *Optional:* Select the availability domain where you want to create the file system and mount target.
       b. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
     – If you select **Existing Mount Target**:

   a. Select the compartment where you have the existing mount target. The mount target must reside within the subnet in the selected VCN.
   b. Specify the OCID of the existing mount target ID.
- If you select **Existing File System**:
   a. Select an existing subnet to use for the mount target. This subnet must be available in the selected VCN.
   b. Select the compartment where you have the existing file system.
   c. Specify the OCID of the existing file system.
- If you select both **Existing Mount Target** and **Existing File System**:
   a. Select the compartment where you have the existing mount target.
   b. Specify the OCID of the existing mount target ID.
   c. Select the compartment where you have the existing file system.
   d. Specify the OCID of the existing file system. The existing file system must be in the same availability domain as the existing mount target.

9. Click **Next**.
10. Click **Save Changes**.
11. Click **Terraform Actions**, and then click **Apply**.
12. Enter a name for the job, and then click **Apply**.
13. Periodically monitor the progress of the Apply job until it is finished.

Remove the File Storage

If you created an Oracle WebLogic Server for OCI domain with a file storage , and no longer require the file storage, you can remove it.

> **Note:**
>
> You cannot use this procedure on a domain that was created before Release 21.3.3 (September 16, 2021).

To remove a file storage:

1. Run the following command to manually unmount from all the VMs:

```
umount -l /u01/shared
```

2. Remove the /etc/fstab entries on all VMs.

> **Note:**
>
> Even if you do not unmount the file storage, the VM instance is functional as long as there are no domain dependencies. The VMs restart successfully, even if it fails to mount the deleted file storage. After removing the file storage, you can remove the `/etc/fstab` entries on all VMs.

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. Select the **Compartment** that contains your stack.
6. Click the name of your stack.
7. Click **Variables**.
8. Click **Edit Variables**.
9. Clear **Add File System Storage**.
10. Click **Next**.
11. Click **Save Changes**.
12. Click **Terraform Actions**, and then click **Apply**.
13. Enter a name for the job, and then click **Apply**.
14. Periodically monitor the progress of the Apply job until it is finished.

5 Clone a Stack

You can clone an Oracle WebLogic Server for OCI instance by cloning the block volumes that contain the middleware binaries (`mw`) and the domain configuration (`data`).

> **Note:**
>
> To ensure that a cloned instance uses the appropriate image, do not clone the boot volume.

Following are the different methods to clone an instance:

- **Move the instance**
  In this method, the binaries and data are moved from one instance to another. Here, the same database, load balancer, and IDCS application (if IDCS is enabled), is used in the source and the cloned instance.

  Example: You want a newer version of the OS and cannot get the current instance's kernel updated. In this use case, you would want to move binaries and data from one instance at a lower kernel version to another instance, where the kernel is newer.

  > **Note:**
  >
  > In the JRF case, when following the manual cloning method, where infra database is used, to ensure that the cloned instance can connect to the database, it is recommended to:
  >
  > – Verify that the cloned instance security rules allow access to the existing infra database.
  >
  > – Have the Compartment and VCN of the cloned instance be the same as the database.

- **Copy the instance**
  In this method, you will copy the binaries and data from one instance to another. Here, load balancer or IDCS application (if IDCS is enabled), is different for the source and instance.

  Example: You have a test instance and a production instance. After you have completed testing updates in the test instance you want to copy the contents to the production instance.

  > **Note:**
  >
  > Currently, in this method, cloning for an instance with JRF database is not supported.

Topics:

- Move a WebLogic Server Instance
- Copy a WebLogic Server Instance

Move a WebLogic Server Instance

Use the clone script to clone and move a WebLogic server instance, where the script performs various procedural steps to clone your instance. You can also manually clone an instance by following the steps provided for a JRF or non-JRF instance.

For a non-JRF instance, you can clone to move the binaries and data, and then reuse the load balancer and IDCS of the original instance. For a JRF instance, you can clone and check if it is able to connect to the database through the data source, as no database-specific configuration updates are required.

> **Note:**
>
> For instances created after 22.3.2 (August 2022), if Application Performance Monitoring (APM) is not configured on the `Original instance`, but Application Performance Monitoring is configured on the `Cloning instance`, then the clone script adds Application Performance Monitoring to the cloned stack.

Clone a JRF or non-JRF Instance using a Script

Learn how to clone an instance by using the clone script.

> **Note:**
>
> This cloning procedure is not recommended for upgrading WLS versions.

In this procedure,

- `Original instance`, is the WebLogic Server instance you want to clone.
- `Cloning instance`, is the WebLogic Server instance you will create to clone the `Original instance`.

Methods: There are two methods to clone an instance.

- Method 1: Do not destroy the `Original instance` stack and run the script on the `Cloning instance` to clone the required volumes from the `Original instance`.
- Method 2: Manually create cloned volumes, destroy the `Original instance` stack, and run the script on the `Cloning instance` to attach the cloned volumes.

Method 1: Do not destroy the `Original instance` stack and run the script on the `Cloning instance` to clone the required volumes from the `Original instance`.

1. Create a domain. This is the `Cloning instance`. See Create a Stack.

   > **Note:**
   >
   > - You must create the `Cloning instance` using the same configuration as the `Original instance`, in the same compartment and region.
   > - It is recommended to stop the original instance servers before running the cloning script. This is specifically recommended for JRF instances, as having 2 WebLogic domains using the same infrastructure schemas is not supported by Oracle WebLogic Server.
   > - If your instance uses an existing VCN with a new subnet, then the CIDR range for the subnet created in the `Cloning instance` differs from the CIDR range of the subnet in the `Original instance`.

2. In the `Original instance`, delete any load balancer using a reserved IP. See Remove the Load Balancer.
3. In the `Original instance`, if you have assigned an SSL certificate to the original load balancer, then set up SSL certificates on the load balancer in `Cloning instance`. See Add a Certificate to the Load Balancer.
4. Stop all the servers in the `Original instance`. See Start and Stop a Domain.
5. Log in to each node of the `Cloning instance` as an `opc` user.
6. Run the `create_clone.py` script on each of the nodes. The `create_clone.py` script is located at `/opt/scripts/cloning`:
   - If you want to clone only the data block volumes:

     ```
     python3 create_clone.py -s <Original_instance_stack_OCID>
     ```

   - If you want to clone both the data block and Middleware volumes:

     ```
     python3 create_clone.py -s <Original_instance_stack_OCID> -m true
     ```

   > **Note:**
   >
   > Do not run the script on `Cloning instance` nodes that correspond to nodes added on the `Original instance` with the **Do Not Update Domain Configuration for Scale Out** selected. The metadata for these nodes will be missing, resulting in failures at various steps in the script. On such nodes you must extend the domain in the same manner as you did on the `Original instance`.

   If the clone script fails at a particular stage, complete the steps in Clone Script Failed, and then return to this procedure to continue with the next step.
7. Access the WebLogic console of the `Cloning instance` to verify the updates you performed to the instance. See Access the WebLogic Console.

8. Destroy and Delete the `Original instance`. See Delete a Stack.

Method 2: Manually create cloned volumes, destroy the `Original instance` stack, and run the script on the `Cloning instance` to attach the cloned volumes.

> **Note:**
>
> This method is recommended when you are using VCN peering or running low on limits, such as, compute or load balancer.

1. Clone the data block volumes of the `Original instance`. See Cloning a Volume.
   As required, you can also clone the Middleware volumes.
2. Destroy and then delete the stack in `Original instance`. See Destroy Stack Resources and Delete the Stack.
3. Create a domain. This is the `Cloning instance`. See Create a Stack.

   > **Note:**
   >
   > You must create the `Cloning instance` using the same configuration as the `Original instance`, in the same compartment and region.

4. Log in to each node of the `Cloning instance` as an `opc` user.
5. Run the `create_clone.py` script on each of the nodes. The `create_clone.py` script is located at `/opt/scripts/cloning`:
   - If you have cloned only the data block volumes:

     ```
     python3 create_clone.py -d <Original_instance_data_volume_OCID>
     ```

   - If you have cloned both the data block and Middleware volumes:

     ```
     python3 create_clone.py -d <Original_instance_data_volume_OCID> -
     m <Original_instance_Middleware_volume_OCID>'
     ```

   > **Note:**
   >
   > Do not run the script on `Cloning instance` nodes that correspond to nodes added on the `Original instance` with the **Do Not Update Domain Configuration for Scale Out** selected. The metadata for these nodes will be missing, resulting in failures at various steps in the script. On such nodes you must extend the domain in the same manner as you did on the `Original instance`.

   If the clone script fails at a particular stage, complete the steps in Clone Script Failed, and then return to this procedure to continue with the next step.

6. Access the WebLogic console of the `Cloning instance` to verify the updates you performed to the instance. See Access the WebLogic Console.

You have successfully cloned the instance.

If you run terraform apply after cloning, complete the following steps:

1. If you add nodes, it restores any volumes that was previously destroyed by the cloning script.
   The restored volumes can be identified if they have the following name formata:
   - `<clone_prefix>-data-block-<number>`
   - `<clone_prefix>-mw-block-<number>`

   Detach and delete the restored volumes. See Deleting a Volume.
2. If OCI logging is added then OCI logging has to be run independently on the compute instance running the administration server.
   Run the following commands on the administration server:

   ```
   python3 update_metadata.py -k use_oci_logging -v true
   python3 update_logging.py
   ```

Copy a WebLogic Server Instance

For a non-JRF instance, you can clone to copy the binaries and data from the original instance, and then use the load balancer and IDCS in the cloned instance. In this setup, the cloned instance would have the same topology as the original instance.

> **Note:**
>
> Currently, in this method, cloning for an instance with JRF database is not supported.

Topics:

- Clone a non-JRF Instance
- Scale Out the Cloned Instance

Clone a non-JRF Instance

Complete the following steps:

In this procedure,

- `Original instance`, is the instance you want to clone.
- `Cloning instance`, is the instance you will create to clone the original instance.

1. Create a non-JRF domain. This is the `Cloning instance`. See Create a Basic Domain.

> **Note:**
>
> If the source instance uses IDCS, the IDCS configuration steps assume that the new instance was created with the **Enable Authentication Using Identity Cloud Service** option selected.

2. Complete the following steps if you have created an instance by using the Identity Cloud Service:
   1. Open the `/u01/data/domains/idcss_domain/config/config.xml` file and make a note of the `idcs:client-id` value.
   2. Take a backup of the following to the `/tmp` location:

      ```
      /u01/data/cloudgate_config
      ```

3. Detach the block volumes that were created in the cloning instance.
   1. Stop the Domain. See Start and Stop a Domain.
   2. Remove the mount volume:

      ```
      sudo umount <mount>
      ```

      Where *<mount>* is `/u01/data` or `/u01/app`.
   3. From the navigation menu, click **Compute**. Under the **Compute** group, click **Instances**.
   4. From the Compartment dropdown, select the compartment in which your instance is created.
   5. Click the instance you created.
   6. In the instance page, under Resources, click **Attached Block Volume**.
   7. Against the block volume that was created when creating the domain, click the menu icon and then click **Detach**.
      The `Detach Block Volume` dialog box is displayed. It provides information about your volume and the iSCSI commands you will need. The commands are ready to use with the appropriate information included.

      Copy these commands.
   8. Click **Continue Detachment**.
   9. Access the node and then run the iSCSI commands that you copied in step 3e.
   10. Repeat step 3e through step 3g for all the other block volumes that were created when creating the domain.
4. Clone both the middleware and data block volumes of the `Original instance`. See Cloning a Volume.

> **Note:**
>
> Follow the next steps, to first attach the middleware volume and then attach the data block volume to the `Cloning instance`. This sequence ensures that the `mountVolume.sh` script works as desired, which you will be running later in this procedure.

5. To the `Cloning instance`, attach the middleware volume that your cloned in step 4.
   1. In the instance page, under Resources, click **Attached Block Volume** > **Attach Block Volume**.
   2. In the **Attach Block Volume** page, select the compartment where you cloned the block volume earlier.
   3. Click the drop-down and select the block volume you clone earlier.
   4. Select the Attachment Type as **ISCSI**.
   5. Click **Attach**.
      An `Attach Block Volume` message appears.
   6. Click **Close**.
6. To the `Cloning instance`, attach the data block volume that your cloned in step 4.
   1. In the instance page, under Resources, click **Attached Block Volume** > **Attach Block Volume**.
   2. In the **Attach Block Volume** page, select the compartment where you cloned the block volume earlier.
   3. Click the drop-down and select the block volume you clone earlier.
   4. Select the Attachment Type as **ISCSI**.
   5. Click **Attach**.
      An `Attach Block Volume` message appears.
   6. Click **Close**.
7. Run the following script:

   ```
   /opt/scripts/cloning/mountVolume.sh -m <cloned-middleware-volume-name> -d
   <cloned-data-volume-name>
   ```

   This script runs the iSCSI commands to attach both the middleware and data volumes. Also, the script updates the UUID entries of both the volumes in `/etc/fstab`, which ensures that the mount is persistent across reboot.
8. Update the metadata on each node by using `update_metadata.py` script, which is located at `/opt/scripts/utils/`:

   We need to update the metadata for reboot to work correctly. This also starts the node manager and administration server automatically after every restart. There are 5 domain related values in metadata that need to be updated for reboot to work.

   ```
   python3 /tmp/update_metadata.py -k wls_domain_name -v
   <resource_prefix>_domain
   python3 /tmp/update_metadata.py -k wls_machine_name -v
   ```

```
<resource_prefix>_machine_
python3 /tmp/update_metadata.py -k wls_cluster_name -v
<resource_prefix>_cluster
python3 /tmp/update_metadata.py -k wls_admin_server_name -v
<resource_prefix>_adminserver
python3 /tmp/update_metadata.py -k wls_ms_server_name -v
<resource_prefix>_server_
```

The script updates one value at time. Also, for every command, it creates the backup of previous setup under `/opt/scripts/utils/metadata_backup_<timestamp>.txt`.

> **Note:**
>
> After the restart works as desired, you can delete these backed up files.

9. Run the following `update_hostname.sh` script, which is located at `/opt/scripts/cloning`:

   ```
   ./update_hostname.sh <FQDN-of-source-instance>
   ```

   For example: `./update_hostname.sh source12c-wls-0.subnet1fd5ed7.idcsvcn.oraclevcn.com`

   This updates the hostname to the cloned hostname in the `nodemanager.properties` and `config.xml` files. It also updates the `startup.properties` file to reflect the correct admin url.

10. Reboot the instance.

    This will automatically first start the node manager and then start the administration server.

11. Repeat step 2 through step 10 on all the nodes.

    You have now cloned the WebLogic Server Instance. Continue with the next steps to configure the load balancer and IDCS.

> **Caution:**
>
> If you are not using IDCS, then do not continue with the next steps.

12. Access the IDCS console and copy the `idcs:client-secret` value:

    a. Access the IDCS console.

    b. From the left navigation, Click **Applications**.

    c. Search for the confidential application that is used by the Authentication Provider for the instance. It will have the instance name and the word `confidential` in it.

    d. Select the application, and then click **Configuration**.

    e. Under **General Information**, click **Show Secret** against **Client Secret**. The Client Secret information is displayed.

   f. Copy the **Client Secret** value.

13. In the WebLogic console, update the `idcs:client-id` and `idcs:client-secret`.

   a. Access the WebLogic console of the clone instance. See Access the WebLogic Console.

   b. Under **Domain Structure**, click **Security Realms**.

   c. In the **Summary of Security Realms** page, select **myrealm** > **Providers**.

   d. Click **IDCSIntegrator** > **Provider Specific**.

   e. In the left navigation, under **Change Center**, click **Lock & Edit**.

   f. In the **Provider Specific** page, enter the values for `Client Id`, `Client Secret`, and `Confirm Client Secret`.
   Where,

      - `Client Id` : is the client ID that you made a note of in step 2a.

      - `Client Secret` or `Confirm Client Secret`: is the Client Secret that you copied in step 12.

   g. Click **Save**.

   h. In the left navigation, under **Change Center**, click **Release Configuration.**

14. In the cloned instance, restore the backup of `/u01/data/cloudgate_config` from the `/tmp` to `/u01/data` location.

    This is the backup which you performed in step 2b.

15. Restart the container.

```
sudo systemctl status appgateway.service
sudo systemctl start appgateway.service
sudo docker ps -a
sudo systemctl status appgateway.service
sudo /opt/scripts/idcs/run_cloudgate.sh
```

16. Verify if `nginx` is redirecting traffic to WebLogic server through port `9999`:

```
ip=$(hostname -i)
cloudgate_url=${ip}:9999
curl -v ${cloudgate_url}
curl -s -o /dev/null -w "%{http_code}\n" ${cloudgate_url}
```

17. Repeat step 14 through step 16 on all nodes.

> **Note:**
>
> Here you will use the load balancer of the cloned instance. So, there is no update required to the load balancer.

18. Verify if you can access sample app. See Access the Sample Application.

19. After every reboot, manually start the `appgateway`:

    ```
    sudo systemctl start appgateway
    sudo docker ps -a
    ip=$(hostname -i)
    cloudgate_url=${ip}:9999
    curl -v ${cloudgate_url}
    curl -v ${cloudgate_url}/sample-app
    ```

You have successfully cloned the instance.

Access the respective WebLogic console to verify the updates you performed to the instance. See Access the WebLogic Console.

After you verified the updates you performed to the instance, delete the block volumes that you detach in step 6. See Deleting a Volume.

Scale Out the Cloned Instance

You can scale out the instance that you cloned.

> **Note:**
>
> If you have updated the WebLogic Server password, then create a version of the Secret and update the metadata scripts to use the new Secret. See *To update a secret's contents to create a new secret version* under Managing Secrets in the Oracle Cloud Infrastructure documentation.

To scale out the cloned instance, edit the node count variable for the stack and complete the required steps:

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. Click **Variables**.
6. Click **Edit Variables**.
7. Edit **WebLogic Server Node Count** to the increase the number of compute instances.
8. Select **Do Not Update Domain Configuration for Scale Out**.
9. Click **Next**.
10. Click **Save Changes**.

    The apply job is run to update the stack.
11. Periodically monitor the progress of the Apply job until it is finished.
12. Click **Outputs**.

13. Locate `WebLogic_Instances`, and verify the number of compute instances in the updated stack.

14. Update the metadata on each node by using `update_metadata.py` script, which is located at `/opt/scripts/utils/`:

    We need to update the metadata for reboot to work correctly. This also starts the node manager and administrator server automatically after every restart. There are 5 domain related values in metadata that need to be updated for reboot to work.

    ```
    python3 /tmp/update_metadata.py -k wls_domain_name -v
    <resource_prefix>_domain
    python3 /tmp/update_metadata.py -k wls_machine_name -v
    <resource_prefix>_machine_
    python3 /tmp/update_metadata.py -k wls_cluster_name -v
    <resource_prefix>_cluster
    python3 /tmp/update_metadata.py -k wls_admin_server_name -v
    <resource_prefix>_adminserver
    python3 /tmp/update_metadata.py -k wls_ms_server_name -v
    <resource_prefix>_server_
    ```

    The script updates one value at time. Also, for every command, it creates the backup of previous setup under `/opt/scripts/utils/metadata_backup_<timestamp>.txt`.

    > **Note:**
    >
    > After the restart works as desired, you can delete these backed up files.

15. Run the manual scale out by running extend domain scripts on the added instance.

    > **Note:**
    >
    > You must change `WLST_PROPERTIES` for a custom SSL set up, to avoid SSL handshake errors.
    >
    > For example, if you are using Custom Identity and Custom Trust, then `weblogic.security.TrustKeyStore=CustomTrust`, `weblogic.security.CustomTrustKeyStoreFileName`, and `weblogic.security.CustomTrustKeyStoreType` need to be set.

    a. Run the following command:

    ```
    export WLST_PROPERTIES="-
    Dweblogic.security.SSL.minimumProtocolVersion=TLSv1.2 -
    Dweblogic.security.SSL.ignoreHostnameVerification=true
    -Dweblogic.security.TrustKeyStore=DemoTrust -
    Djava.security.egd=file:///dev/urandom -
    Dweblogic.ssl.JSSEEnabled=true
    -Dweblogic.security.SSL.enableJSSE=true -Dwlst.offline.log=/u01/logs/
    wlst_extend_domain.log"
    ```

b. Run the following extend domain script for WebLogic Server 12.2.1.4 or 14.1.1.0:

```
/opt/scripts/decryptStrings.sh 1 | /u01/app/oracle/middleware/
oracle_common/common/bin/wlst.sh -skipWLSModuleScanning /opt/
scripts/extend_12c_domain.py
```

Access the respective WebLogic console to verify the updates you performed to the instances. See Access the WebLogic Console.

6
Delete a Stack

Use Resource Manager to destroy and delete the stack when you no longer need an Oracle WebLogic Server for OCI domain.

You perform two separate actions:

- Destroy the stack – A destroy job terminates the compute instance or instances for the domain but the stack's state and job history remain.
- Delete the stack – A delete job permanently removes the stack and all related resources that were created for the domain, such as compute instances, networking components, and load balancer components.

If your domain includes the Java Required Files (JRF) components, then you must also delete the JRF schema before you destroy the stack.

If your domain uses Oracle Identity Cloud Service, then you must also delete the security resources before you destroy the stack.

Tasks:

- Delete a JRF Database Schema
- Delete the Identity Cloud Service Resources
- Delete Autoscaling Resources
- Destroy Stack Resources
- Delete the Stack
- Delete the Database Security List
- Delete the Identity Cloud Service Resources Manually

Delete a JRF Database Schema

If the Oracle WebLogic Server for OCI domain you want to delete was created with the Java Required Files (JRF) components, you must remove the JRF schema before you destroy the stack.

> **WARNING:**
>
> Skip this procedure for an instance that was cloned, as database schemas are required in the cloned instance.

You'll need the following to delete the JRF schema:

- The secure shell (SSH) private key that corresponds to the public key that was specified when you created the domain

- The public IP address to the Administration Server node. If the WebLogic domain is in a private subnet, look up the bastion's public IP address and the private IP address of the administration server node.
- The WebLogic Server administrator password
- The SYSDBA user password of the database associated with the domain

To delete the schema associated with a JRF-enabled domain:

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
   ```

   Or,

   ```
   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
   ```

   For example:

   ```
   ssh -i /home/myuser/mykey.openssh opc@203.0.113.13
   ```

   ```
   ssh -i ~/.ssh/mykey.openssh -o ProxyCommand="ssh -W %h:%p -i ~/.ssh/
   mykey.openssh opc@198.51.100.1" opc@192.0.2.254
   ```

2. If prompted, enter the passphrase for the private key.
3. Change to the `oracle` user.

   ```
   sudo su - oracle
   ```

4. Run the following command to delete the JRF schemas, providing the passwords for the WebLogic Server administrator and SYSDBA user.

   ```
   /opt/scripts/delete_rcu.sh domain_password database_password
   ```

   For example:

   ```
   /opt/scripts/delete_rcu.sh wlsadminpassword dbadminpassword
   ```

5. Wait for the script to run. The operation is completed when you see output similar to the following:

```
Component schemas dropped:
Component                                     Status       Logfile
Common Infrastructure Services                Success      /tmp/
RCU2019-06-10_numstring/logs/stb.log
Oracle Platform Security Services             Success      /tmp/
RCU2019-06-10_numstring/logs/opss.log
User Messaging Service                        Success      /tmp/
RCU2019-06-10_numstring/logs/ucsums.log
```

```
Audit Services                                Success      /tmp/RCU2019-06-10_numstring/
logs/iau.log
Audit Services Append                         Success      /tmp/RCU2019-06-10_numstring/
logs/iau_append.log
Audit Services Viewer                         Success      /tmp/RCU2019-06-10_numstring/
logs/iau_viewer.log
Metadata Services                             Success      /tmp/RCU2019-06-10_numstring/
logs/mds.log
WebLogic Services                             Success      /tmp/RCU2019-06-10_numstring/
logs/wls.log
Repository Creation Utility - Drop : Operation Completed
>
<Jun 11, 2019 05:15:12 PM GMT> <INFO> <cleanup.py> <(host:resourcename-
wls-0.mysubnet.ocidbvcnterrafo.oraclevcn.com) - Successfully deleted rcu schemas for
prefix = SP1560222222>
```

6. If necessary, review the entire output for exceptions or failures that caused a failed or incomplete deletion. For example, a failure to connect to the domain could be caused by an invalid WebLogic Server administrator password. Re-execute the script after fixing the issues.

Delete the Identity Cloud Service Resources

If the Oracle WebLogic Server for OCI domain you want to delete was configured to use Oracle Identity Cloud Service for authentication, then you must delete the security resources for the domain before you destroy the stack.

> **WARNING:**
>
> Skip this procedure for an instance that was cloned, as Oracle Identity Cloud Service resources are required in the cloned instance.

You'll need the client ID and secret of an existing confidential application in Oracle Identity Cloud Service. See Create a Confidential Application.

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
   ```

   For example:

   ```
   ssh -i /home/myuser/mykey opc@203.0.113.13
   ```

2. If prompted, enter the passphrase for the private key.
3. Run the following command to delete the security resources for this domain.

Provide the client ID and secret of the confidential application in Oracle Identity Cloud Service.

```
sudo su oracle -c '/opt/scripts/idcs/delete_idcs_applications.sh 
idcs_app_client_id idcs_app_client_secret'
```

Sample output:

```
Deactivating App Gateway gateway_name...
Deleting App Gateway gateway_name...
Deactivating application enterprise_app_name...
Deleting application enterprise_app_name...
Deactivating application confidential_app_name...
Deleting application confidential_app_name...
```

Delete Autoscaling Resources

The autoscaling resources, Functions, Event Rule, and Notification Subscriptions are created using Oracle Cloud Infrastructure SDK APIs from WebLogic Administration instance during provisioning. So, you must destroy these autoscaling resources before destroying the stack.

Run the command in Cloud Shell to destroy autoscaling resources using `remove_resources.py` script. To create `remove_resources.py`, see Script File to Delete Resources .

> **Note:**
>
> A user who is not an administrator can also run the `pre-destroy` command.

```
python3 remove_resources.py pre-destroy <service_name_prefix> -f 
autoscaling
```

If you run this script from a nonhome region, use the following command:

```
python3 remove_resources.py pre-destroy <service_name_prefix> -f 
autoscaling -r <region_name>
```

Example:

```
python3 remove_resources.py pre-destroy abcstack -f autoscaling -r us-
phoenix-1
```

Destroy Stack Resources

To delete an Oracle WebLogic Server for OCI domain, use Resource Manager to destroy the stack associated with the domain before you execute the Delete Stack action.

> **Note:**
>
> If your domain includes the Java Required Files (JRF) components, be sure to delete the JRF schema before you destroy the stack.
> If autoscaling is enabled, you must destroy the autoscaling resources before you destroy the stack. See Delete Autoscaling Resources.

A destroy action terminates the compute instance or instances for the domain but the stack's state and job history remain until you execute the Delete Stack action.

To destroy a stack:

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. From the **Compartment** dropdown, select the compartment where your stack is located.
4. Click the name of your stack.
5. Click **Terraform Actions**, then click **Destroy**.
6. When prompted for confirmation, click **Destroy**.

   A job with type `Destroy` and state `Accepted` is added to the top of the table under **Jobs**. After a few minutes, the state changes to `In Progress`.
7. Wait for the destroy job state to change to `Succeeded` before you delete the stack.

To verify the stack has been destroyed, navigate to the Compute Instances page. Instances associated with a destroyed stack are labeled `Terminated`.
If there are networking resources created by the stack and those resources are still in use by other compute instances (not created by the stack):

- The destroy action on the stack fails because related networking resources are still in use. Those networking resources will not be deleted.
- The compute instances created by the stack are terminated. You can proceed to delete the stack.

Delete the Stack

To delete the Oracle WebLogic Server for OCI stack, use Resource Manager to delete the stack associated with the stack.

> **NOT_SUPPORTED:**
>
> - Deleting a stack does not destroy the stack resources. Ensure that you first Destroy Stack Resources and then delete the stack.
> - To verify a stack has been destroyed, navigate to the Compute Instances page. Instances associated with a destroyed stack are labeled `Terminated`.

A delete action permanently removes the stack and all related resources that were created for the domain, such as compute instances, network components, and load balancer components. If any of the network resources are being used by other stacks, the delete action:

- Does not remove those network components
- Does remove the compute instances created by the stack for the domain

To delete a stack:

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. From the **Compartment** dropdown, select the compartment where your stack is located.
4. Click the name of your stack.
5. Click **Delete Stack**.

   > **Note:**
   >
   > Ensure that you first Destroy Stack Resources and then delete the stack.

6. When prompted for confirmation, click **Delete**.

The Stacks page redisplays immediately. You'll no longer see your stack listed on the page.

Delete the Database Security List

If your Oracle WebLogic Server for OCI domain is JRF-enabled and is connected to an Oracle Cloud Infrastructure Database (DB System), then you can delete the security list that grants the domain access to the database.

> **WARNING:**
>
> Do not the delete this security list if other domains are in the same VCN and are using the same database.

This security list is not a component of the stack for your domain, and is not automatically deleted when you destroy the stack.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. From the navigation menu, click **Networking**, and then select **Virtual Cloud Networks**.
3. Select the **Compartment** where your database's virtual cloud network (VCN) is located.
4. Click the name of the database's VCN.
5. Click **Security Lists**.
6. Click the security list for your domain, `*servicename*-wls-to-db-seclist`.

   `*servicename*` is the resource name prefix you provided during stack creation.
7. Click **Terminate**.
8. When prompted for confirmation, click **Delete**.

Delete the Identity Cloud Service Resources Manually

If the Oracle WebLogic Server for OCI domain you deleted was configured to use Oracle Identity Cloud Service for authentication, you can manually delete the security resources for the domain.

If the domain's compute instances still exist, you can delete the security resources using a script. See Delete the Identity Cloud Service Resources.

1. Delete the App Gateway that's associated with your domain.

   From Identity Cloud Service console, expand the navigation drawer, click **Security**, and then click **App Gateways**.

   The name of the gateway is `*servicename*_app_gateway_*timestamp*`. For example, `mywls_app_gateway_2019-08-01T01:02:01.123456`.
2. Delete the enterprise application that's associated with your domain.

   From Identity Cloud Service console, expand the navigation drawer, and then click **Applications**.

   The name of the application is `*servicename*_enterprise_idcs_app_*timestamp*`. For example, `mywls_enterprise_idcs_app_2019-08-01T01:02:01.123456`.
3. Delete the confidential application that's associated with your domain.

The name of the application is `servicename_confidential_idcs_app_timestamp`. For example, `mywls_confidential_idcs_app_2019-08-01T01:02:01.123456`.

7

Troubleshoot

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

Topics

- Check Known Issues
- Clone Script Failed
- Stack Creation Failed
- Error in Mounting the Volume During Provisioning
- Unable to Access the Domain
- Load Balancer does not send Cookie `X-Oracle-BMC-LBS-Route`
- Autoscaling Failed to Create Functions
- Management Agents Are Not Deleted on Instance Termination
- Enterprise Manager Console Is Not Loading
- Scale Out Fails on the Administration Compute Instance
- Enable OS Management to Install Patches
- Security Checkup Tool Warnings
- Running `python3` Command Fails
- Get Additional Help and Contact Support

Check Known Issues

Learn about known problems in Oracle WebLogic Server for OCI and how to work around them.

See Known Issues in Oracle WebLogic Server for Oracle Cloud Infrastructure.

Clone Script Failed

Issue: When you run the clone script it might fail at a particular stage.

Workaround: By using the error message, you can identify the stage where the error occurred, fix the error, and then run the following command to continue the cloning script from the required stage.

```
python3 /opt/scripts/cloning/create_clone.py -p <stage_name>
```

AD Mismatch:

In an Availability Domain (AD), if the threshold of set limits are reached, then when you create the `Cloning instance` the Compute instances might be placed in another Availability Domain that does not match the Availability Domain of the `Original instance`. Due to this mismatch, the cloned volumes cannot be attached.

Workaround: Use the Oracle WebLogic Server for OCI console to create backup of data volumes, and optionally, the Middleware volumes. Make a note of the OCIDs of backed up volumes and then complete the steps in Method 2: Manually create cloned volumes and destroy the source stack.

Cleanup Resources of a Deleted Instance

If you create an Oracle WebLogic Server for OCI instance and delete some resources outside of terraform, the terraform destroy may fail.

Issue:

You might have deleted an instance without destroying the instance. In this scenario, some of the resources you had created for the instance are not deleted.

Workaround:

In such scenarios, you can run the following script to remove all the resources of the instance.

1. Copy the following script in Cloud Shell.
   For example, copy the script and save the file as `remove_resources.py`.

```
"""
#
# Copyright (c) 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 os
import sys
import oci

"""
Lists and deletes the resources like instances, policies, volumes,
VCN related resources, logs and tags etc..
"""


class CleanUpResources:

    def __init__(self):
        # delegate token should be present at /etc/oci/
delegation_token in cloud shell
        if os.path.exists('/etc/oci/delegation_token'):
            with open('/etc/oci/delegation_token', 'r') as file:
                delegation_token = file.read()
            self.signer =
oci.auth.signers.InstancePrincipalsDelegationTokenSigner(delegation_
token=delegation_token)
```

```
        else:
            print("ERROR: In the Cloud shell the delegation token does 
not exist at location /etc/oci/delegation_token."
                  "Run the script from the Cloud shell, where you need to 
delete the resources.")
            sys.exit(1)
        self.vcn_client = oci.core.VirtualNetworkClient(config={}, 
signer=self.signer)
        self.virtual_network_composite_operations = 
oci.core.VirtualNetworkClientCompositeOperations(self.vcn_client)
        self.log_client = oci.logging.LoggingManagementClient(config={}, 
signer=self.signer)
        self.log_composite_operations = 
oci.logging.LoggingManagementClientCompositeOperations(self.log_client)
        self.identity_client = oci.identity.IdentityClient(config={}, 
signer=self.signer)
        self.identity_client_composite_operations = 
oci.identity.IdentityClientCompositeOperations(self.identity_client)

    # Lists all the resources based on the service name prefix
    def list_all_resources(self, service_name_prefix):
        search_client = 
oci.resource_search.ResourceSearchClient(config={}, signer=self.signer)
        running_resources = ["RUNNING", "Running", "AVAILABLE", 
"STOPPED", "Stopped", "ACTIVE", "CREATED", "INACTIVE"]
        resource_not_required = ["PrivateIp", "Vnic"]
        structured_search = 
oci.resource_search.models.StructuredSearchDetails(
            query="query all resources where displayname =~ 
'{}'".format(service_name_prefix),
            type='Structured',

matching_context_type=oci.resource_search.models.SearchDetails.MATCHING_CO
NTEXT_TYPE_NONE)

        resources = search_client.search_resources(structured_search)
        resources_details = []
        no_of_resources = 0
        tagname_resource = "wlsoci-" + service_name_prefix
        default_rt = "Default Route Table for " + service_name_prefix
        print(
            "Resource Name                               Resource 
Type                        Resource Lifecycle State                      
OCID         DOC")
        print(

"=========================================================================
========================================================================")
        for resource in resources.data.items:
            resource_name = resource.display_name
            if (resource_name.startswith(service_name_prefix) or 
tagname_resource in resource_name or default_rt in resource_name) and (
                    resource.lifecycle_state in running_resources) and (
                    resource.resource_type not in resource_not_required):
                resources_details.append(resource)
                no_of_resources = no_of_resources + 1
                print("{}             {}          {}
{}           {}".format(resource.display_name,

                  resource.resource_type,

                  resource.lifecycle_state,

                  resource.identifier,

                  resource.time_created))
        print(

"===================================================================
====================================================================
=========")
        print("Total number of resources
{}".format(len(resources_details)))
        return resources_details

    # Removes all resources based on the service name prefix
    def cleanup_resources(self, delete_list):
        print("Deleting the resources")
        self.delete_policies(delete_list)
        self.delete_instance(delete_list)
        self.delete_block_volumes(delete_list)
        self.delete_load_balancer(delete_list)
        self.delete_subnet(delete_list)
        self.delete_sec_list(delete_list)
        self.delete_route_table(delete_list)
        self.delete_dhcp_options(delete_list)
        self.delete_internet_gateway(delete_list)
        self.delete_service_gateway(delete_list)
        self.delete_local_peering_gateway(delete_list)
        self.delete_nat_gateway(delete_list)
        self.delete_vcn_resources(delete_list)
        self.delete_unified_agent_configuration(delete_list)
        self.delete_log(delete_list)
        self.delete_log_group(delete_list)
        self.delete_mount_targets(delete_list)
        self.delete_fss(delete_list)
        self.delete_tag_namespace(delete_list)
        self.delete_boot_volumes(delete_list)

    # Delete Policies
    def delete_policies(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "Policy":
                policy_ocid = resource.identifier
                print("Deleting policy: {0}, with ocid:
{1}".format(resource.display_name, policy_ocid))
                try:

self.identity_client_composite_operations.delete_policy_and_wait_for
_state(
                        policy_ocid,

wait_for_states=[oci.identity.models.Policy.LIFECYCLE_STATE_DELETED])
                    print("Deleted policy successfully!")
                except Exception as e:
                    print("Error while deleting the policy {0}, policy id
{1}, Error message {2}".format(
                        resource.display_name, policy_ocid, str(e)))

    # Delete Dynamic Group
    def delete_dynamic_group(self, service_name_prefix):
        tenancy = os.environ['OCI_TENANCY']
        dynamic_group_list =
self.identity_client.list_dynamic_groups(tenancy).data
        for d_group in dynamic_group_list:
            if service_name_prefix in d_group.name:
                print("Deleting the dynamic group: {0}, with ocid:
{1}".format(d_group.name, d_group.id))
                try:

self.identity_client_composite_operations.delete_dynamic_group_and_wait_fo
r_state(
                        d_group.id,
wait_for_states=[oci.identity.models.DynamicGroup.LIFECYCLE_STATE_DELETED]
)
                    print("Deleted the dynamic group successfully!")
                except Exception as e:
                    print("Error while deleting the dynamic group name
{}, ocid {}, Error message {}".format(
                        d_group.name, d_group.id, str(e)))

    # Delete Block Volumes
    def delete_block_volumes(self, delete_list):
        bv_client = oci.core.BlockstorageClient(config={},
signer=self.signer)
        bv_composite_operations =
oci.core.BlockstorageClientCompositeOperations(bv_client)
        for resource in delete_list:
            if resource.resource_type == "Volume":
                bv_ocid = resource.identifier
                try:
                    print(
                        "Deleting the block volume: {0}, with ocid
{1}".format(resource.display_name, bv_ocid))

bv_composite_operations.delete_volume_and_wait_for_state(
                        bv_ocid,
wait_for_states=[oci.core.models.Volume.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted the block volume successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the block volume {0}, ocid
{1}, Error message {2}".format(
                            resource.display_name, bv_ocid, str(e)))
    # Delete all compute instances
    def delete_instance(self, delete_list):
        compute_client = oci.core.ComputeClient(config={},
signer=self.signer)
        compute_composite_operations =
oci.core.ComputeClientCompositeOperations(compute_client)
        for resource in delete_list:
            if resource.resource_type == "Instance":
                instance_ocid = resource.identifier
                instance_name = resource.display_name
                print("Deleting the compute instance: {0}, with
ocid {1}".format(instance_name, instance_ocid))
                try:

compute_composite_operations.terminate_instance_and_wait_for_state(
                        instance_ocid,
wait_for_states=[oci.core.models.Instance.LIFECYCLE_STATE_TERMINATED
])
                    print("Deleted the compute instance
successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the instance {0},
ocid {1}, Error message {2}".format(
                            instance_name, instance_ocid, str(e)))

    # Delete all Subnets in the VCN
    def delete_subnet(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "Subnet":
                subnet_ocid = resource.identifier
                print(
                    "Deleting subnet: {0}, with ocid
{1}".format(resource.display_name, resource.identifier))
                try:

self.virtual_network_composite_operations.delete_subnet_and_wait_for
_state(
                        subnet_ocid,

wait_for_states=[oci.core.models.Subnet.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted subnet successfully!")
                except Exception as e:
                    print("Error while deleting the subnet {0},
ocid {1}, Error message {2}".format(resource.display_name,

                          subnet_ocid, str(e)))

    # Delete Security lists
    def delete_sec_list(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "SecurityList":
                sec_list_name = resource.display_name
                sec_list_ocid = resource.identifier
                if not ("Default" in sec_list_name):
                    print(
                        "Deleting the security list: {0}, with ocid 
{1}".format(resource.display_name,

     resource.identifier))
                    try:

self.virtual_network_composite_operations.delete_security_list_and_wait_fo
r_state(
                            sec_list_ocid,

wait_for_states=[oci.core.models.SecurityList.LIFECYCLE_STATE_TERMINATED])
                        print("Deleted the security list successfully!")
                    except Exception as e:
                        print(
                            "Error while deleting the security list {0}, 
ocid {1}, Error message {2}".format(
                                resource.display_name, sec_list_ocid, 
str(e)))

    # Delete Load balancers
    def delete_load_balancer(self, delete_list):
        lb_client = oci.load_balancer.LoadBalancerClient(config={}, 
signer=self.signer)
        lb_composite_operations = 
oci.load_balancer.LoadBalancerClientCompositeOperations(lb_client)
        for resource in delete_list:
            if resource.resource_type == "LoadBalancer":
                lb_name = resource.display_name
                lb_ocid = resource.identifier
                print("Deleting Load balancer {0} with ocid 
{1}".format(lb_name, lb_ocid))
                try:

lb_composite_operations.delete_load_balancer_and_wait_for_state(
                        lb_ocid,

wait_for_states=[oci.load_balancer.models.WorkRequest.LIFECYCLE_STATE_SUCC
EEDED])
                    print("Load balancer deleted successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the loadbalancer {0}, ocid 
{1}, Error message {2}".format(
                            lb_name, lb_ocid, str(e)))

    # Delete Route tables
    def delete_route_table(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "RouteTable":
                route_table_name = resource.display_name
                route_table_ocid = resource.identifier
                # Removing the route rules from the tables
                rt_details = oci.core.models.UpdateRouteTableDetails()
                rt_details.route_rules = []
```

```
self.virtual_network_composite_operations.update_route_table_and_wai
t_for_state(
                    route_table_ocid, rt_details,

wait_for_states=[oci.core.models.RouteTable.LIFECYCLE_STATE_AVAILABL
E])

self.vcn_client.update_route_table(route_table_ocid, rt_details)
                # Default route table can't be deleted from VCN
                if not ("Default" in route_table_name):
                    print(
                        "Deleting the route table: {0}, with ocid
{1}".format(resource.display_name,

         resource.identifier))
                    try:

self.virtual_network_composite_operations.delete_route_table_and_wai
t_for_state(
                            route_table_ocid,

wait_for_states=[oci.core.models.RouteTable.LIFECYCLE_STATE_TERMINAT
ED])

                        print("Deleted the route table
successfully!")
                    except Exception as e:
                        print("Error while deleting the route table
{0}, ocid {1}, Error message {2}".format(
                            resource.display_name,
route_table_ocid, str(e)))
                        if "associated with Subnet" in str(e):
                            try:

self.delete_subnet_route_table_association(route_table_ocid)
                                # After removing the association
again retrying the removal of route table
                                # This is for Db subnet route table

self.virtual_network_composite_operations.delete_route_table_and_wai
t_for_state(
                                    route_table_ocid,

wait_for_states=[oci.core.models.RouteTable.LIFECYCLE_STATE_TERMINAT
ED])
                                print("Deleted the route table
successfully!")
                            except Exception as e:
                                print("Error while deleting the
route table after removing the association "
                                      "{0}, ocid {1}, Error message
{2}".format
                                      (resource.display_name,
route_table_ocid, str(e)))
```

```
    # Delete Subnet and route table association to remove route table
    def delete_subnet_route_table_association(self, route_table_ocid):
        default_rt_id_in_vcn = ""
        print("Route table is associated with a subnet. Removing the
association between the subnet and route table")
        rt_res = self.vcn_client.get_route_table(route_table_ocid).data
        vcn_id = rt_res.vcn_id
        compartment_id = rt_res.compartment_id
        list_route_rables_vcn =
self.vcn_client.list_route_tables(compartment_id=compartment_id,

vcn_id=vcn_id).data
        for rt in list_route_rables_vcn:
            if "Default Route" in rt.display_name:
                default_rt_id_in_vcn = rt.id
        list_subnets =
self.vcn_client.list_subnets(compartment_id=compartment_id,
vcn_id=vcn_id).data
        for subnet in list_subnets:
            subnet_ocid = subnet.id
            if subnet.route_table_id == route_table_ocid:
                subnet_details = oci.core.models.UpdateSubnetDetails()
                subnet_details.route_table_id = default_rt_id_in_vcn
                try:

self.virtual_network_composite_operations.update_subnet_and_wait_for_stat
e(
                        subnet_ocid, subnet_details,

wait_for_states=[oci.core.models.Subnet.LIFECYCLE_STATE_AVAILABLE])
                    print("Removed the association between the subnet and
route table.")
                except Exception as e:
                    print("Error while removing the association between
the subnet and route table {}".format(str(e)))

    # Delete DHCP Options
    def delete_dhcp_options(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "DHCPOptions":
                dhcp_name = resource.display_name
                dhcp_ocid = resource.identifier
                if not ("Default" in dhcp_name):
                    print(
                        "Deleting the DHCP options: {0}, with ocid
{1}".format(resource.display_name, dhcp_ocid))
                    try:

self.virtual_network_composite_operations.delete_dhcp_options_and_wait_for
_state(dhcp_ocid,

                               wait_for_states=[
oci.core.models.DhcpOptions.LIFECYCLE_STATE_TERMINATED])
                        print("Deleted the DHCP options
successfully!")
                    except Exception as e:
                        print(
                            "Error while deleting the DHCP options
{0}, ocid {1}, Error message {2} ".format(
                                resource.display_name, dhcp_ocid,
str(e)))

    # Delete Internet Gateway
    def delete_internet_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "InternetGateway":
                ig_ocid = resource.identifier
                print("Deleting the Internet Gateway: {0}, with
ocid {1}".format(resource.display_name,

            ig_ocid))
                try:

self.virtual_network_composite_operations.delete_internet_gateway_an
d_wait_for_state(ig_ocid,

                                     wait_for_states=[


oci.core.models.InternetGateway.LIFECYCLE_STATE_TERMINATED])

                    print("Deleted the Internet Gateway
successfully!")
                except Exception as e:
                    print("Error while deleting the Internet
Gateway {0}, ocid {1}, Error message {2}".format(
                        resource.display_name,
                        ig_ocid, str(e)))

    # Delete Service Gateway
    def delete_service_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "ServiceGateway":
                svc_gateway_ocid = resource.identifier
                print("Deleting the service gateway: {0}, with ocid
{1}".format(resource.display_name,

           svc_gateway_ocid))
                try:

self.virtual_network_composite_operations.delete_service_gateway_and
_wait_for_state(
                        svc_gateway_ocid,
wait_for_states=[oci.core.models.ServiceGateway.LIFECYCLE_STATE_TERM
INATED])

                    print("Deleted the service gateway
successfully!")
                except Exception as e:
                    print("Error while deleting the service gateway {0}, 
ocid {1}, Error message {2}".format(
                        resource.display_name,
                        svc_gateway_ocid, str(e)))

    # Delete Local Peering Gateway
    def delete_local_peering_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "LocalPeeringGateway":
                lpg_ocid = resource.identifier
                print("Deleting the local peering gateway: {0}, with ocid 
{1}".format(resource.display_name,

           lpg_ocid))
                try:

self.virtual_network_composite_operations.delete_local_peering_gateway_and
_wait_for_state(
                        lpg_ocid,
wait_for_states=[oci.core.models.LocalPeeringGateway.LIFECYCLE_STATE_TERMI
NATED])

                    print("Deleted local peering gateway successfully!")
                except Exception as e:
                    print("Error while deleting the local peering gateway 
{0}, ocid {1}, Error message {2}".format(
                        resource.display_name, lpg_ocid, str(e)))

    # Delete Nat Gateway
    def delete_nat_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "NatGateway":
                nat_ocid = resource.identifier
                print("Deleting the NAT gateway: {0}, with ocid 
{1}".format(resource.display_name,

 nat_ocid))
                try:

self.virtual_network_composite_operations.delete_nat_gateway_and_wait_for_
state(
                        nat_gateway_id=nat_ocid,

wait_for_states=[oci.core.models.NatGateway.LIFECYCLE_STATE_TERMINATED]
                    )
                    print("Deleted the NAT gateway successfully!")
                except Exception as e:
                    print("Error while deleting the NAT gateway {0}, ocid 
{1}, Error message {2}".format(
                        resource.display_name, nat_ocid, str(e)))

    # Delete VCN
    def delete_vcn_resources(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "Vcn":
                vcn_ocid = resource.identifier
                vcn_name = resource.display_name
                print("Deleting the VCN: {0}, with ocid
{1}".format(vcn_name, vcn_ocid))
                try:

self.virtual_network_composite_operations.delete_vcn_and_wait_for_st
ate(vcn_ocid,


oci.core.models.Vcn.LIFECYCLE_STATE_TERMINATED)
                    print("Deleted the VCN successfully!")
                except Exception as e:
                    print("Error while deleting the VCN {0}, VCN id
{1}, Error message {2}".format(vcn_name, vcn_ocid,

                              str(e)))

    # Deleting the Unified Agent Configuration
    def delete_unified_agent_configuration(self, delete_list):
        for resource in delete_list:
            if resource.resource_type ==
"UnifiedAgentConfiguration":
                uac_ocid = resource.identifier
                print("Deleting the unified agent configuration:
{}, with ocid {}".format(resource.display_name, uac_ocid))
                try:

self.log_composite_operations.delete_unified_agent_configuration_and
_wait_for_state(
                        uac_ocid,

wait_for_states=[oci.logging.models.WorkRequest.STATUS_SUCCEEDED])
                    print("Deleted the unified agent configuration
successfully!")
                except Exception as e:
                    print("Error while deleting the unified agent
configuration name {0}, ocid {1} - Error message {2}".format(
                        resource.display_name, uac_ocid, str(e)))

    # Delete logs in a Log groups
    def delete_log(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "LogGroup":
                log_group_ocid = resource.identifier
                list_logs =
self.log_client.list_logs(log_group_ocid).data
                for log in list_logs:
                    print("Deleting the log name {0}, with log ocid
{1}".format(log.display_name, log.id))
                    try:

self.log_composite_operations.delete_log_and_wait_for_state(
                            log_group_ocid, log.id,
wait_for_states=[oci.logging.models.WorkRequest.STATUS_SUCCEEDED])
                        print("Deleted the log successfully!")
                    except Exception as e:
                        print("Error while deleting the log name {}, log 
ocid {}, Error message {}".format(
                            log.display_name, log.id, str(e)))

    # Delete Log Group
    def delete_log_group(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "LogGroup":
                log_group_ocid = resource.identifier
                print("Deleting the log group: {0}, with ocid 
{1}".format(resource.display_name,

log_group_ocid))
                try:

self.log_composite_operations.delete_log_group_and_wait_for_state(
                        log_group_ocid,
wait_for_states=[oci.logging.models.WorkRequest.STATUS_SUCCEEDED])

                    print("Deleted log group successfully!")
                except Exception as e:
                    print("Error while deleting the log group {0}, ocid 
{1}, Error message {2}".format(
                        resource.display_name, log_group_ocid, str(e)))

    # Delete the Mount targets
    def delete_mount_targets(self, delete_list):
        mt_client = oci.file_storage.FileStorageClient(config={}, 
signer=self.signer)
        mt_composite_operations = 
oci.file_storage.FileStorageClientCompositeOperations(mt_client)
        for resource in delete_list:
            if resource.resource_type == "MountTarget":
                mt_ocid = resource.identifier
                print("Deleting the mount target {0}, with ocid 
{1}".format(resource.display_name, mt_ocid))
                try:

mt_composite_operations.delete_mount_target_and_wait_for_state(
                        mt_ocid,
wait_for_states=[oci.file_storage.models.MountTarget.LIFECYCLE_STATE_DELET
ED])
                    print("Deleted the mount target successfully!")
                except Exception as e:
                    print("Error while deleting the mount target {0}, 
ocid {1}, Error message {2}".format(
                        resource.display_name, mt_ocid, str(e)))

    # Delete FSS
    def delete_fss(self, delete_list):
        fss_client = oci.file_storage.FileStorageClient(config={},
signer=self.signer)
        fss_composite_operations =
oci.file_storage.FileStorageClientCompositeOperations(fss_client)
        for resource in delete_list:
            if resource.resource_type == "FileSystem":
                fss_ocid = resource.identifier
                try:
                    # Get the list of exports to delete
                    list_exports =
fss_client.list_exports(file_system_id=fss_ocid).data
                    for export in list_exports:
                        export_ocid = export.id
                        print("Deleting the export id
{}".format(export_ocid))

fss_composite_operations.delete_export_and_wait_for_state(
                            export_id=export_ocid,

wait_for_states=[oci.file_storage.models.Export.LIFECYCLE_STATE_DELE
TED])
                        print("Deleted the exports successfully!")
                except Exception as e:
                    print("Error while deleting the export, Error
message {}".format(str(e)))
                try:
                    print("Deleting the FSS: {0}, with ocid
{1}".format(resource.display_name, fss_ocid))

fss_composite_operations.delete_file_system_and_wait_for_state(
                        fss_ocid,
wait_for_states=[oci.file_storage.models.FileSystem.LIFECYCLE_STATE_
DELETED])
                    print("Deleted the FSS successfully!")
                except Exception as e:
                    print("Error while deleting the FSS name {0},
ocid {1}, Error message {2}".format(
                        resource.display_name, fss_ocid, str(e)))

    # Deletion of TagNamespace
    def delete_tag_namespace(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "TagNamespace":
                tag_ns_name = resource.display_name
                tag_ns_ocid = resource.identifier
                print("Deleting the tag namespace {0}, with ocid
{1}".format(tag_ns_name, tag_ns_ocid))
                try:
                    # Retiring the tag namespace
                    tag_status =
self.identity_client.get_tag_namespace(tag_namespace_id=tag_ns_ocid)
.data
                    print("Tag namespace: {} and isRetired:
{}".format(tag_ns_name, tag_status.is_retired))

                    if not tag_status.is_retired:
                        print("Retiring the tag namespace
{}".format(tag_ns_name))
                        tag_ns_details =
oci.identity.models.UpdateTagNamespaceDetails()
                        tag_ns_details.is_retired = True

self.identity_client_composite_operations.update_tag_namespace_and_wait_fo
r_state(
                            tag_namespace_id=tag_ns_ocid,
                            update_tag_namespace_details=tag_ns_details,
                            wait_for_states=[

oci.identity.models.TagNamespace.LIFECYCLE_STATE_INACTIVE])
                        tag_status =
self.identity_client.get_tag_namespace(tag_namespace_id=tag_ns_ocid).data
                        print("Tag status before deleting
{}".format(tag_status.is_retired))
                    print("Deleting the tag namespace
{}".format(tag_ns_name))
                    # Tag namespace deletion is taking too long time. So
not waiting for the completion.

self.identity_client.cascade_delete_tag_namespace(tag_namespace_id=tag_ns_
ocid)
                    print("Asynchronous deletion of Tag namespaces is
enabled."
                          "Check the deletion status manually. Tag name
{0} with ocid {1}".format(tag_ns_name,

                       tag_ns_ocid))
                except Exception as e:
                    print("Error while deleting the Tag namespace {0},
ocid {1}, Error message {2} "
                          .format(tag_ns_name, tag_ns_ocid, str(e)))

    # Deleting the unattached boot volumes
    def delete_boot_volumes(self, delete_list):
        bv_client = oci.core.BlockstorageClient(config={},
signer=self.signer)
        bv_composite_operations =
oci.core.BlockstorageClientCompositeOperations(bv_client)
        for resource in delete_list:
            if resource.resource_type == "BootVolume" and
resource.lifecycle_state == "AVAILABLE":
                bv_ocid = resource.identifier
                bv_name = resource.display_name
                print("Deleting the boot volume {}, with ocid {}
".format(bv_name, bv_ocid))
                try:

bv_composite_operations.delete_boot_volume_and_wait_for_state(
                        boot_volume_id=bv_ocid,

wait_for_states=[oci.core.models.BootVolume.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted the boot volume successfully!")
                except Exception as e:
                    print("Error while deleting the boot volume 
name {}, ocid {}, Error message {}".format(bv_name,

                                       bv_ocid,

                                       str(e)))


if __name__ == '__main__':
    no_of_args = len(sys.argv)
    if no_of_args < 2:
        print("Usage: ")
        print("To list all the resources based on service name 
prefix:")
        print("python3 remove_resources.py <service_name_prefix>")
        print("To remove all the resources based on service name 
prefix:")
        print("python3 remove_resources.py <service_name_prefix> 
delete")
        sys.exit(1)

    service_prefix = sys.argv[1]
    print("Service prefix name:" + service_prefix)
    cleanup_util = CleanUpResources()
    if len(service_prefix) >= 16:
        service_prefix = service_prefix[0:16]
    service_prefix = service_prefix + "-"
    if no_of_args < 3:
        print("Listing all resources with service prefix name" + 
service_prefix)
        cleanup_resources = 
cleanup_util.list_all_resources(service_prefix)
    elif no_of_args < 4 and sys.argv[2] == "delete":
        print("Deleting all resources with service prefix name" + 
service_prefix)
        cleanup_resources = 
cleanup_util.list_all_resources(service_prefix)
        cleanup_util.cleanup_resources(cleanup_resources)
        cleanup_util.delete_dynamic_group(service_prefix)
```

2. Run the following command to list all the resources:

   ```
   python3 remove_resources.py <full_service_prefix_name>
   ```

3. Check that list and ensure that you want to delete these resources.
4. Run the following command to deleted all the resources of the instance:

   ```
   python3 remove_resources.py <full_service_prefix_name> delete
   ```

Stack Creation Failed

Troubleshoot a failed Oracle WebLogic Server domain that you attempted to create with Oracle WebLogic Server for OCI.

View the stack log files

Use the Terraform job logs in Resource Manager to identify the cause of the failure.

1. Click the navigation menu ☰, select **Developer Services**. Under the **Resource Manager** group, click **Jobs**.
2. Identify and click the job for your stack.
    - The Type is Apply.
    - The State is Failed.
    - The Stack is the name of your Oracle WebLogic Server for OCI stack.
3. From the Logs section, search the log for error messages.
   You can optionally **Download** the log files and search the files offline.
4. See below for details about specific error messages.

Modify the stack configuration

If necessary, delete the current stack resources, modify your stack configuration, and then apply the changes.

1. Click the navigation menu ☰, select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
2. Click the name of your stack.
3. Click **Terraform Actions** and select **Destroy**.
   Wait for the destroy job to complete.
4. Click **Edit Stack**.
5. When done, click **Save Changes**.
6. Click **Terraform Actions** and select **Apply**.

Cannot launch a stack in Marketplace

Example message: `Unable to accept Terms of Use`

In Marketplace, you might see the message when you click **Launch Stack**, after you've selected a stack version and compartment, and checked the Oracle Standard Terms and Restrictions box.

You likely don't have permission to:

- Create Marketplace applications in the selected compartment. Verify that this policy exists in the compartment where you want to create the stack.
  ```
  Allow group Your_Group to manage app-catalog-listing in compartment
  Your_Compartment
  ```
- Access the selected compartment. Choose another compartment or ask your administrator.

Cannot determine home region

Example message:

```
data.oci_core_app_catalog_subscriptions.mp_image_subscription[0]:
Refreshing state...
Error: Null value found in list ... "oci_identity_regions" "home-
region"
```

If you are not an administrator, ask them to verify that the following root-level policy exists in your tenancy:

```
Allow group Your_Group to inspect tenancies in tenancy
```

Cannot find dynamic group and secrets policy

Example messages:

```
Error: Service error:NotAuthorizedOrNotFound. Authorization failed or
requested resource not found. http status code: 404.
 Opc request id: request_id on modules/policies/groups.tf line 8, in
resource...
 "oci_identity_dynamic_group" "wlsc_instance_principal_group" {


Error: Service error:NotAuthorizedOrNotFound. Authorization failed or
requested resource not found. http status code: 404.
 Opc request id: request_id on wlsc-policies.tf line 10, in resource...
 "oci_identity_policy" "wlsc_secret-service-policy" {
```

When the **OCI Policies** check box is selected (by default), Oracle WebLogic Server for OCI creates a dynamic group and one or more root-level policies in your tenancy.

You must be an Oracle Cloud Infrastructure administrator, or be granted root-level permissions to create domains. If you are not an administrator, ask them to verify that root-level policies exist in your tenancy. For example:

```
Allow group Your_Group to manage dynamic-groups in tenancy
Allow group Your_Group to manage policies in tenancy
Allow group Your_Group to use secret-family in tenancy
```

See:

- Create Root Policies
- Create Policies for the Dynamic Group

Maximum number of dynamic groups has exceeded

Example message:

```
<WLSC-VM-ERROR-0119> : Failed to get secret content for
[ocid1.vaultsecret.oc1.iad.alongstring123]: [{'status': 400,
'message': "This instance principal matches more than '5' dynamic groups,
update your dynamic groups' matching rules"...'}]>
```

When the **OCI Policies** check box is selected (by default), Oracle WebLogic Server for OCI creates a dynamic group and one or more root-level policies in your tenancy. The maximum number of dynamic groups allowed is 5.

Solution:

1. Add the following policy that uses the existing dynamic group to access the new secrets for the new stack:

   ```
   Allow dynamic-group <existing-dyanmic-group-name> to read secret-bundles
   in tenancy where target.secret.id = '<OCID_of_the_secret>'
   ```

2. Deselect the **OCI Policies** check box and try to create the stack again.

Unable to get secret content or decrypted credential

Example messages:

- `Failed to get secret content for Your_vault_secret_OCID`
- `Authorization failed or requested resource not found`
- `Error retrieving %s password from Secret Vault`
- `Failed in create domain due to exception [Failed to retrieve WebLogic Password from Secrets Vault]`
- `Failed to retrieve IDCS Client Secret from Secrets Vault`
- `Unable to get decrypt credential`
- `Key or Vault does not exist or you are not authorized to access them.`

When you create a domain with Oracle WebLogic Server for OCI, you provide the OCID values of the secrets that contain the passwords to use for the domain and during provisioning. The compute instances use this information to decrypt the passwords. The compute instances are granted access to vault secrets using policies.

You must be an Oracle Cloud Infrastructure administrator, or be granted root-level permissions to create domains. If you are not an administrator, ask them to verify that relevant vault secret policies exist in your tenancy and compartment. For example:

```
Allow group Your_Group to use secret-family in tenancy
Allow dynamic-group Your_DynamicGroup to use secret-family in compartment
MyCompartment
Allow dynamic-group Your_DynamicGroup to use keys in compartment
MyCompartment
Allow dynamic-group Your_DynamicGroup to use vaults in compartment
MyCompartment
```

If the policies exist, check that the OCID of the compartment in listed in dynamic group.

See:

- Create Secrets for Passwords
- Create Policies for the Dynamic Group

- Create Root Policies

Unable to get decrypted credential when creating a stack in a private subnet

Example message: `<WLSC-VM-ERROR-001> Unable to get decrypt credential [HTTPSConnectionPool(host='auth.us-phoenix-1.oraclecloud.com', port=443): Max retries exceeded with url: /v1/x509 (Caused by ConnectTimeoutError(<oci._vendor.urllib3.connection.VerifiedHTTPSConnection object at 0x1e5110>, 'Connection to auth.us-phoenix-1.oraclecloud.com timed out. (connect timeout=10)'))]>`

When you create a domain with Oracle WebLogic Server for OCI in an existing private subnet, provisioning fails if the WebLogic Server subnet is using a route table that does not include a service gateway or a Network Address Translation (NAT) gateway.

Modify the private subnet, and select a route table that uses a service gateway or NAT gateway. Or select a virtual cloud network (VCN) whose default route table uses a service gateway or NAT gateway. Refer to these topics:

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

Failed to download Oracle Autonomous Database wallet

Example message: `module.provisioners.null_resource.status_check[0] (remote-exec): <Nov 23, 2019 09:37:17 PM GMT> <ERROR> <oci_api_utils> <(host:stackname-wls-0.subnetxxx.stacknamevcn.oraclevcn.com) - <WLSC-VM-ERROR-0052> : Unable to download atp wallet. [{'status': 403, 'message': u'Forbidden', 'code': u'Forbidden', 'opc-request-id': 'FA6C16D8B'}]`

You must be an Oracle Cloud Infrastructure administrator, or be granted root-level and compartment-level permissions to create domains. Access to the database wallet is needed when you create a JRF-enabled domain that uses an autonomous database. If you are not an administrator, ask them to verify that relevant policies for autonomous databases exist in your tenancy and compartment. For example:

```
Allow group Your_Group to inspect autonomous-transaction-processing-
family in compartment Your_ATP_Compartment
Allow dynamic-group Your_DynamicGroup to inspect autonomous-
transaction-processing-family in compartment Your_ATP_Compartment
```

See:

- Create Policies for the Dynamic Group
- Create Root Policies
- Create Compartment Policies

Failed to validate DB connectivity

When you create a domain that includes the Java Required Files (JRF) components, you must select an existing database and provide connection details. The compute instances use this information to connect to the database and provision the JRF database schemas.

Possible causes for this error include:

- You entered the wrong database password or a plain text password.
- The database does not allow the compute instances to access its listen port (1521 by default).
  - Oracle Autonomous Database - Check your access control list (ACL).
  - Oracle Cloud Infrastructure Database - Check the network security group that was assigned to the database, and the security lists for the subnet on which the database was created.
- You selected an Oracle Cloud Infrastructure Database running Oracle Database 12c or later, and you did not provide the name of a pluggable database (PDB).

Invalid or overlapping network CIDR

Stack provisioning fails if you specify subnets with overlapping CIDRs or use the same subnet for WebLogic Server and the load balancer.

Example messages:

```
Error: module.network-wls-public-subnet.oci_core_subnet.wls-subnet: 1 error(s)
occurred: oci_core_subnet.wls-subnet: Service error:InvalidParameter. The
requested CIDR 10.0.3.0/24 is invalid: subnet ocid1.subnet.oc1.iad.aaan4a with
CIDR 10.0.3.0/24 overlaps with this CIDR.. http status code: 400.
```

```
Error: module.validators.null_resource.duplicate_lb2_subnet_cidr: : invalid or
unknown key: WLSC-ERROR: Load balancer subnet 2 CIDR has to be unique value.
```

```
Error: module.validators.null_resource.duplicate_wls_subnet_cidr: : invalid or
unknown key: WLSC-ERROR: Weblogic subnet CIDR has to be unique value.
```

Possible causes for these errors include:

- You chose to create new subnets for WebLogic Server, the load balancer, or the bastion, and the CIDR you specified for these subnets overlaps with the CIDRs for existing subnets in the same virtual cloud network (VCN).
- You chose to use an existing subnet when provisioning a stack with a load balancer, and you specified the same subnet for WebLogic Server and the load balancer.
- You created a JRF-enabled domain, your Oracle Cloud Infrastructure Database and WebLogic domain are in different VCNs, and the VCNs have overlapping CIDRs. For example, you cannot create a WebLogic domain on VCN `10.0.0.0/16` that uses a database on VCN `10.0.0.1/24`.

Job is still running or has timed out

Most stack creation jobs for Oracle WebLogic Server for OCI should complete within an hour. Some internal provisioning problems might cause the job to run indefinitely until it eventually times out after 24 hours.

After the current Apply job times out, run a new Apply job on the same stack. This will destroy any resources that were created, and then attempt to create the resources again. If the problem occurs again, contact support.

Failed to check database port is open for Exadata DB system

When you create a domain that includes Java Required Files (JRF) components, for Exadata DB systems, the database port open check does not work if the **Create DB Security List** checkbox is selected. In this case, the provisioning fails if the database subnet has more than five security lists.

So, when provisioning, deselect the **Create DB Security List** check box to avoid creating an additional security list for the database port in the VCN, and manually open the database port (1521 by default).

Error in Mounting the Volume During Provisioning

Troubleshoot a failed Oracle WebLogic Server for OCI domain due to Oracle Cloud Infrastructure (OCI) policy not in effect.

Issue:

When you create a domain with the **OCI Policies** check box selected, at times, the policy creation takes longer time than it takes to reach the first OCI API call, that is, the call to create a volume mount, and an error message is displayed in the bootstrap log.

Example message:

```
Exception stacktrace [
{'opc-request-id': '<GUID>', 'code': 'NotAuthorizedOrNotFound',
'message': 'Authorization failed or requested resource not found.',
'status': 404}
]
Error executing volume mounting.. Exiting provisioning
```

Workaround:

1. Create a dynamic group using the following rule:
   `All { instance.compartment.id = '<resource_compartment ocid>') }`

   To create a dynamic group, see Create a Dynamic Group.
2. Create the policy for the dynamic group as follows:
   Example policy:

   ```
   Allow dynamic-group MyInstancesPrincipalGroup to use volumes in
   compartment MyCompartment where any { request.operation =
   'AttachVolume', request.operation = 'DetachVolume'}"
   ```

   To create other policies for the dynamic group, see Create Policies for the Dynamic Group.
3. Wait for 10 to 15 minutes and then create a domain without selecting the **OCI Policies** check box.
4. Delete the dynamic group and policy created manually after the domain is created.

Unable to Access the Domain

Troubleshoot problems accessing an Oracle WebLogic Server domain after it's successfully created.

Cannot access the WebLogic Console from the Internet

By default the WebLogic Server Administration Console is accessed through port 7001 or 7002.

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 a domain that's not on a private subnet, make sure the following ingress rules exist:

   ```
   Source: 0.0.0.0/0
   IP Protocol: TCP
   Source Port Range: All
   Destination Port Range: 7002
   ```

   ```
   Source: 0.0.0.0/0
   IP Protocol: TCP
   Source Port Range: All
   Destination Port Range: 7001
   ```

   For a domain on a private subnet, set the `Source` to the CIDR of the bastion instance subnet.

Cannot access the sample application using the load balancer: Not Found

On a domain running Oracle WebLogic Server Standard Edition, the sample application is deployed only to the first Managed Server. If your Standard Edition domain has multiple Managed Servers and you access the sample application using a load balancer, the Managed Servers that aren't hosting the sample application will respond with the code `404` (Not Found).

You can use the WebLogic Server Administration Console to update the targets for the sample application, and add the remaining Managed Servers.

Cannot access applications using the load balancer: Bad Gateway

If you restart the compute instances running your Managed Servers, or you restart the compute instances running the App Gateway, the backend set of the load balancer will temporarily be in an unhealthy state. By default, a load balancer in this state will respond with the code `502` (Bad Gateway). After the WebLogic Server and App Gateway processes are running, the load balancer should return to the OK state.

To check the status of the load balancer and backend servers:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Load Balancers**.
3. Click the load balancer that was created for your domain, *prefix*-lb.
4. Click **Backend Sets**, and then click *prefix*-lb-backendset.
5. Click **Backends**, and then check the state of each backend.
6. Access the WebLogic compute instances using a secure shell (SSH) client. Check that the Managed Server process is listening on its assigned port (the default is 7003).

   ```
   curl -s -o /dev/null -w "%{http_code}\n" http://private_ip:7003
   ```

   A `404` response indicates that the Managed Server is running.
7. If you enabled authentication with Oracle Identity Cloud Service, then access the App Gateway compute instances using an SSH client. Check that the App Gateway process is listening on its assigned port (the default is 9999).

   ```
   curl -s -o /dev/null -w "%{http_code}\n" http://private_ip:9999
   ```

   A `404` response indicates that the App Gateway is running.

See Managing Backend Servers in the Oracle Cloud Infrastructure documentation.

Cannot access the Fusion Middleware Control Console from the Internet

If you enabled authentication with Oracle Identity Cloud Service on a WebLogic Server 12.2.1.4 domain, you might be redirected to an error page when you try to log in to the Fusion Middleware Control Console.

Example message:

```
<Error> <oracle.help.web.rich.OHWFilter>
<BEA-000000> <ADFSHARE-00120: Error encountered while creating the MDS
Session. Application state will be reset. Please logout and log back
in if
problem persists.
oracle.adf.share.ADFShareException: ADFSHARE-00120: Error encountered
while
creating the MDS Session. Application state will be reset. Please
logout and
log back in if problem persists.
```

To access the Fusion Middleware Control Console:

1. Add the Cloud Gate App Role to your confidential application that you created for the domain.
   a. Access the Oracle Identity Cloud Service console.

   b. From the navigation menu, click **Applications**.
   c. Click the confidential application that was created for your domain.
   d. Click the **Configuration** tab.
   e. Under Client Configuration, locate **Grant the client access to Identity Cloud Service Admin APIs**, and then click **Add**.
   f. Select the **Cloud Gate** App Role and click **Add**.
   g. Click **Save**.
2. Restart your WebLogic Server domain and log in to the Fusion Middleware Control Console again.

See Create a Confidential Application.

Load Balancer does not send Cookie `X-Oracle-BMC-LBS-Route`

When you setup Oracle WebLogic Server for OCI with a load balancer, the load balancer does not send cookie `X-Oracle-BMC-LBS-Route`.

Scenario:

1. Create a 2-node WebLogic instance with load balancer by using a Oracle WebLogic Server for OCI listings in marketplace.
2. Access the sample app through load balancer.
3. In your web browser, go to **WebDeveloper** > **Web Console** > **Network**.
4. Click **Reload**.
5. Click on **Get request** of the sample app, then select the **Cookies** tab
   The cookies tab is empty.

Workaround:

1. Sign in to the Oracle Cloud Infrastructure console.
2. From the navigation menu, click **Networking**, and then click **Load Balancers**.
3. Click the name of the **Compartment** that contains the load balancer you want to modify, and then click the load balancer's name.
4. , and then click the name of the backend set you want to modify.
5. In the **Resources** menu, click **Backend Sets**. Deselect **HTTP Only**.
6. Save the changes.
7. Undo the changes you did in step 5 and then save the changes.
8. Reload the sample app browser.
   Now you can view that the cookie with name `X-Oracle-BMC-LBS-Route` is passed properly.

Autoscaling Failed to Create Functions

When you create an Oracle WebLogic Server for OCI domain with autoscaling enabled, functions may not be created during provisioning.

Issue:

During provisioning, if autoscaling failed to create the functions due to invalid OCIR auth token value, provisioning is successful but the following error is displayed in the log:

```
module.provisioners.null_resource.print_service_info[0] (remote-exec):
************************************************************
module.provisioners.null_resource.print_service_info[0] (remote-exec):
This service is configured with the following options .....
module.provisioners.null_resource.print_service_info[0] (remote-exec):
WebLogic Server for OCI Version : 22.1.1-220208100422
module.provisioners.null_resource.print_service_info[0] (remote-exec):
WebLogic Server Version: 12.2.1.4
module.provisioners.null_resource.print_service_info[0] (remote-exec):
WebLogic Server Edition: SUITE
module.provisioners.null_resource.print_service_info[0] (remote-exec):
Virtual Cloud Network : NEW VCN
module.provisioners.null_resource.print_service_info[0] (remote-exec):
Network Type: PRIVATE Network with BASTION
module.provisioners.null_resource.print_service_info[0] (remote-exec):
Domain Type: Plain WebLogic Server Domain (non-JRF)
module.provisioners.null_resource.print_service_info[0] (remote-exec):
APM agent enabled : [True]
module.provisioners.null_resource.print_service_info[0] (remote-exec):
APM agent installed : [True]
module.provisioners.null_resource.print_service_info[0] (remote-exec):
Failed to create autoscaling resources, Check provisioning logs for
details
module.provisioners.null_resource.print_service_info[0] (remote-exec):
User can invoke remove_resources.py script and then rerun the
configure_autoscaling.sh script on Admin VM to recreate the resources.
module.provisioners.null_resource.print_service_info[0] (remote-exec):
************************************************************
```

Workaround:

Perform the following steps to create the function resources:

1. Update the OCIR auth token secret to the valid OCIR auth token as follows:
   - In the script `/opt/scripts/observability/autoscaling/configure_autoscaling.sh`, replace the line `ocir_auth_token=$(python3 /opt/scripts/wls_credentials.py ocirAuthToken)` with `ocir_auth_token='<*VALID_AUTH_TOKEN*>'`.
2. Run the script to delete resources to clean up any resources created by autoscaling during provisioning from WebLogic administration instance.
   `python3 remove_resources.py pre-destroy <service_name_prefix> -f autoscaling`
3. Log in as a `root` user to the Administration server and run the `configure_autoscaling.sh` script.
   `/opt/scripts/observability/autoscaling/configure_autoscaling.sh`

   This script creates the autoscaling functions using the valid OCIR auth token from step 1. If you encounter any errors when you run the script, see `/u01/logs/provisioning.log`.

4. In the OCI console, verify if:
   - Autoscaling functions are created under the function application.
   - Notification subscriptions are created for Scale Out and Scale In notification topics.
   - Event rule is created for the stack.

Management Agents Are Not Deleted on Instance Termination

In case of an Oracle WebLogic Server for OCI domain with autoscaling enabled, when you destroy the stack, the management agent resources associated with the compute instance are not destroyed.

Issue:

When you destroy the stack, the compute instance for the domain is terminated but the associated management agent resources are active.

Workaround:

Perform the following steps to manually delete the management agent resources:

> **Note:**
>
> You must run the following commands from the Cloud Shell. You can also install OCI CLI and create the config file on your host, and then run the following commands. See Installing the CLI.

1. List the management agents in the stack compartment.

```
oci management-agent agent list --compartment-id <stack_compartment_ocid>
oci management-agent agent list --compartment-id
<stack_compartment_ocid>  | grep ocid1.managementagent | grep '"id":'
```

2. Delete the management agents associated with your service.

```
oci management-agent agent delete --agent-id
<OCID_of_the_managementagent_from_step1> --force
```

Enterprise Manager Console Is Not Loading

Issue: After you create a JRF-enabled domain without Oracle Identity Cloud Service in Oracle WebLogic Server for OCI, you are unable to log into the Enterprise Manager console.

> **Note:**
>
> This issue is applicable for stacks created between 31st August, 2022 and 20th September, 2022 (22.3.2 release).

Workaround:

1. Edit the `jps-config.xml` located in `/u01/data/domain/<domain-name>/config/fmwconfig/`. Replace `idstore.scim` with `idstore.ldap`.

   ```
   <serviceInstanceRef ref="idstore.ldap"/>
   ```

2. Execute the `restart_domain.sh` script.

   ```
   /opt/scripts/restart_domain.sh -o restart
   ```

Scale Out Fails on the Administration Compute Instance

Scale out fails due to high CPU usage on the administration compute instance.

Issue:

If the administration compute instance is stressed to 100 percent utilization, the scale out fails as the `pack` commands that are run during scale out on the administration compute instance do not give any results and time out.

Workaround:

Verify if the control group `wlsmcg` exists under `/sys/fs/cgroup/cpu`. If it exists, assign process IDs for the administration server and managed server to the control group, and assign CPU shares to the control group to manage the CPU usage. So, you can successfully run the `pack` command.

> **Note:**
>
> If you start the servers on the administration compute instance using scripts or through the WebLogic console, the new process IDs for the server are not added to the control group.

To reassign the process ID for the administration server and managed server to the control group:

1. Check if current server process ID is part of the control group `wlsmcg`.

   ```
   cat /sys/fs/cgroup/cpu/wlsmscg/tasks | grep "<processID for managed server>"
   ```

2. If the process ID is not found in the step 1, run the following script as the `opc` user to create control groups and assign process IDs for the administration server and managed server.

   ```
   sudo /opt/scripts/create_control_groups.sh
   ```

3. Verify that managed server process ID is assigned to the control group, *wlsmscg*.

   ```
    cat /sys/fs/cgroup/cpu/wlsmscg/tasks | grep "<processID for managed server>"
   ```

Enable OS Management to Install Patches

For an existing Oracle WebLogic Server for OCI instance, you might encounter issues when you use the OS Management to apply patches.

So, to enable the OS Management, create the following policies:

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

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 OCI 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 OCI instances created after October 20, 2021, or the instances on which the October PSUs are applied.

Warning Message	Resolution
`SSL hostname verification is disabled by the SSL configuration.` You see the SSL host name verification warnings in case of existing Oracle WebLogic Server for OCI instances created before release 21.3.2 (August 17, 2021).	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. See Using the BEA Host Name Verifier in Administering Security for 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. See Using the Wildcard Host Name Verifier in *Administering Security for Oracle WebLogic Server*. **Note**: For WebLogic Server 14.1.1.0.0, the default host name verifier is set to the wildcard host name verifier.
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Disable the anonymous RMI T3 and IIOP requests in the WebLogic Server Administration Console as soon as possible unless your deployment requires anonymous T3 or IIOP (not typical). See Disable Remote Anonymous RMI T3 and IIOP Requests.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OCI 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.

If you want to perform anonymous RMI requests, you must disable the java properties. Go to the `nodemanager.properties` file located under `DOMAIN_HOME/nodemanager` and remove the `weblogic.startup.Arguments` property.

Disable Remote Anonymous RMI T3 and IIOP Requests

To disable the remote anonymous RMI T3 and IIOP requests 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 the domain name, and then select the **Security** tab.
3. Expand **Advanced** and deselect **Remote anonymous RMI access via IIOP** and **Remote anonymous RMI access via T3**.

After saving the changes, return to **Change Center** and click **Activate Changes**.

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.

Running `python3` Command Fails

Issue: When you run a command that uses `python3`, it might fail on the stacks that were created earlier than May 25, 2022.

Workaround: If your stack is created earlier than May 25, 2022, then in the command use `python`. For stacks created after May 25, 2022, use `python3`.

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 OCI.

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 OCI 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 OCI** 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 OCI 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** 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 OCI 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 a `FUSER could not be located` error is displayed when applying a patch, complete the following steps:
>
> 1. Set the environment variable: `export OPATCH_NO_FUSER=TRUE`.
> 2. In the Unix shell where step 1 is run or `OPATCH_NO_FUSER=TRUE` is set, apply the required patch.

Patches in a new release of Oracle WebLogic Server for OCI are not automatically applied to existing domains that you created with Oracle WebLogic Server for OCI. You have to apply the patches manually if you wish to update your existing domain to match the latest release, or to match a specific supported release.

A Patch Set Update (PSU) is a group of related patches that is identified by a specific version number. When you create a domain with Oracle WebLogic Server for OCI, you choose a version of WebLogic Server in this format: `<major_version>.<patch_level>.<build>`. For example, `12.2.1.4.191121.01`.

> **Tip:**
>
> For a list of new features and enhancements that were added recently to improve your Oracle WebLogic Server for OCI experience, see What's New for Oracle WebLogic Server for OCI.

JDK Patching

To patch JDK, download and install JDK from Oracle Technology Network to the `/u01/app/oracle/jdk` location. Do not use `/u01/jdk` location as this location is symlink to `/u01/app/oracle/jdk`.

Patches for Oracle WebLogic Server for OCI

Table 8-1 table lists the patches that are found in the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases. Use your Oracle Support account to locate and download the patch you wish to apply.

> **Note:**
>
> For a list of patches that are found in the Oracle WebLogic Server for OCI 11.1.1.7.0 and 12.2.1.3 releases, see Table 8-2.

Table 8-1 Patches for the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases

WebLogic Server Version	Patches	WebLogic Server for OCI Release	Patch List
14.1.1.0.221018.01	opatch: • 34686388 - WLS patch set update 14.1.1.0.221010 • 34545599 - Coherence 14.1.1.0 Cumulative Patch 11 (14.1.1.0.11) New Opatch Version: • Bug 28186730 - Opatch 13.9.4.2.11 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.3.0, 12.2.1.4.0, and 14.1.1.0.0	22.4.1	October 2022 PSUs
12.2.1.4.221018.01	opatch: • 34566592 - OWSM Bundle Patch 12.2.1.4.220905 • 34604561 - FMW Third party Bundle Patch 12.2.1.4.220915 • 34545596 - Coherence 12.2.1.4 Cumulative Patch 15 (12.2.1.4.15) • 34535558 - ADF Bundle Patch 12.2.1.4.220825 • 34653267 - WLS Patch set update 12.2.1.4.220929 • 34542329 - Merge request on top of 12.2.1.4.0 for bugs 34280277, 26354548, 26629487, 29762601 • 33093748 - FMW Platform 12.2.1.4.0 SPU for APRCPU2021 • 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 - Patch for OAM Console Login Fails After Applying 1.80.331 JDK New Opatch Version: • Bug 28186730 - Opatch 13.9.4.2.11 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.3.0, 12.2.1.4.0, and 14.1.1.0.0	22.4.1	October 2022 PSUs

Table 8-1 (Cont.) Patches for the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases

WebLogic Server Version	Patches	WebLogic Server for OCI Release	Patch List
12.2.1.3.221018.01	New Opatch Version: • Bug 28186730 - Opatch 13.9.4.2.11 for EM 13.4, 13.5 and for FMW/WLS 12.2.1.3.0, 12.2.1.4.0, and 14.1.1.0.0 opatch: • 34697822 - WLS patch set update 12.2.1.3.221013 • 34545595 - Coherence 12.2.1.3 Cumulative Patch 20 (12.2.1.3.20) • 34636492 - FMW Third party Bundle Patch 12.2.1.3.220926 • 34667652 - OWSM Bundle Patch 12.2.1.3.221004 • 32982708 - FMW Platform 12.2.1.3.0 SPU for APR CPU 2021 • 26394536 - Managed server going into failed state due to deadlock • 31544340 - ADR for WebLogic July CPU 2020 • 30186876 - 12.2.1.4.0 OIM: SOA composer deployment fails during SOA_Server startup (intermittent) • 30252137 - Xpath-functions current-datetime should consider DST • 27263211 - EM configuration fails in FOH (12.2.1.3.1) • 31464643 - Merge request on top of 12.2.1.3.0 for bugs 29011959 and 30385564 • 34243945 - ADF Bundle Patch 12.2.1.3.220604 • 29840258 - RCU 12.2.1.3 fails due to bad password generation for FMW registry user • 28659321 - New exception showing up in managed server logs • 32397127 - OPSS patch for April 2021 • 26045997 - Enabling driver fan without running ONS daemons causes connect request error • 27608287 - Upgrade assistant readiness received error when schema prefix is mixed case • 24738720 - 12.2.1.2 EM template can't be added with JRF Cloud template added • 33903365 - OAM console login fails after applying 1.80.331 JDK • 18345580 - XSLT grouping using muenchian method does not work at runtime • 26355633 - PoolDataSource fails connect to 12c non container DB using service name	22.4.1	October 2022 PSUs

Table 8-1 (Cont.) Patches for the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases

WebLogic Server Version	Patches	WebLogic Server for OCI Release	Patch List
	• 26287183 - PSR:PERF:WLS 12.2.1.3 : ~49% regression in JDBC benchmarks • 26261906 - Merge request on top of 12.2.0.1.0 for bugs 24811916, 25232931, and 25559137 • 26051289 - Invalid arguments while using Preparestatement (string, string[]) with WE8ISO8859		
14.1.1.0.220719.04	opatch: • 34429365 - WebLogic patch set update 14.1.1.0.220727 • 34248968 - Coherence 14.1.1.0 Cumulative Patch 10 (14.1.1.0.10) • 32589626 - Bug fix: `err_http2_protocol_error` when accessing the HTTPS channel network for HTTP2 protocol.	22.3.1	July 2022 PSUs
12.2.1.4.220719.04	opatch: • 34341032 - OWSM Bundle Patch 12.2.1.4.220701 • 33093748 - FMW Platform 12.2.1.4.0 SPU for APRCPU2021 • 34287807 - FMW Third party Bundle Patch 12.2.1.4.220616 • 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 • 1221414 (34248976) - Coherence 12.2.1.4 Cumulative Patch 14 (12.2.1.4.14) • 34247006 - ADF Bundle Patch 12.2.1.4.220606 • 34236279 - WLS Patch set update 12.2.1.4.220602 • 33903365 - OAM console login fails after applying 1.80.331 JDK	22.3.1	July 2022 PSUs
14.1.1.0.220419.05	opatch: • 32589626 - Bug fix: `err_http2_protocol_error` when accessing the HTTPS channel network for HTTP2 protocol. • 141109 (33902209) - Bundle patch for Oracle Coherence Version 14.1.1.0.9 • 34011596 - WebLogic patch set update 14.1.1.0.220329	22.2.3	April 2022 PSUs

Table 8-1   (Cont.) Patches for the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases

WebLogic Server Version	Patches	WebLogic Server for OCI Release	Patch List
12.2.1.4.220419.05	opatch: • 33903365 - Patch for OAM console login fails after applying 1.80.331 JDK • 30874677 - Offline wlst not encrypting credential after updatedomain() • 33618954 - OWSM bundle patch 12.2.1.4.211129 • 32772437 - FMW Platform 12.2.1.4.0 SPU for April 2021 CPU • 34044738 - FMW Thirdparty Bundle Patch 12.2.1.4.220406 • 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 • 1221413 (33902201) - Bundle patch for Oracle Coherence Version 12.2.1.4.13 • 33958532 - ADF bundle patch 12.2.1.4.220314 • 34012040 - WebLogic patch set update 12.2.1.4.220329	22.2.3	April 2022 PSUs
14.1.1.0.220419.04	opatch: • 32589626 - Bug fix: `err_http2_protocol_error` when accessing the HTTPS channel network for HTTP2 protocol. • 141109 (33902209) - Bundle patch for Oracle Coherence Version 14.1.1.0.9 • 34011596 - WebLogic patch set update 14.1.1.0.220329	22.2.2	April 2022 PSUs

Table 8-1 (Cont.) Patches for the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases

WebLogic Server Version	Patches	WebLogic Server for OCI Release	Patch List
12.2.1.4.220419.04	opatch: • 33903365 - Patch for OAM console login fails after applying 1.80.331 JDK • 30874677 - Offline wlst not encrypting credential after updatedomain() • 33618954 - OWSM bundle patch 12.2.1.4.211129 • 32772437 - FMW Platform 12.2.1.4.0 SPU for April 2021 CPU • 34044738 - FMW Thirdparty Bundle Patch 12.2.1.4.220406 • 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 • 1221413 (33902201) - Bundle patch for Oracle Coherence Version 12.2.1.4.13 • 33958532 - ADF bundle patch 12.2.1.4.220314 • 34012040 - WebLogic patch set update 12.2.1.4.220329	22.2.2	April 2022 PSUs
14.1.1.0.220419.03	opatch: • 32589626 - Bug fix: `err_http2_protocol_error` when accessing the HTTPS channel network for HTTP2 protocol. • 141109 (33902209) - Bundle patch for Oracle Coherence Version 14.1.1.0.9 • 34011596 - WebLogic patch set update 14.1.1.0.220329	22.2.1	April 2022 PSUs

Table 8-1 (Cont.) Patches for the Oracle WebLogic Server for OCI 14.1.1.0 and 12.2.1.4 releases

WebLogic Server Version	Patches	WebLogic Server for OCI Release	Patch List
12.2.1.4.220419.03	opatch: • 33903365 - Patch for OAM console login fails after applying 1.80.331 JDK • 30874677 - Offline wlst not encrypting credential after updatedomain() • 33618954 - OWSM bundle patch 12.2.1.4.211129 • 32772437 - FMW Platform 12.2.1.4.0 SPU for April 2021 CPU • 34044738 - FMW Thirdparty Bundle Patch 12.2.1.4.220406 • 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 • 1221413 (33902201) - Bundle patch for Oracle Coherence Version 12.2.1.4.13 • 33958532 - ADF bundle patch 12.2.1.4.220314 • 34012040 - WebLogic patch set update 12.2.1.4.220329	22.2.1	April 2022 PSUs

The following table lists the patches for 11.1.1.7.0 and 12.2.1.3 releases. Use your Oracle Support account to locate and download the patch you wish to apply.

Table 8-2 Patches for 11.1.1.7.0 and 12.2.1.3 releases

WebLogic Server Version	Patches
12.2.1.3	opatch: • 33903365 - Patch for OAM console login fails after applying 1.80.331 JDK • 1221318 (33902200) - Coherence 12.2.1.3 cumulative patch 18 (12.2.1.3.18) • 33949366 - ADF bundle patch 12.2.1.3.220310 • 34010914 - WebLogic patch set update 12.2.1.3.220329 • 33791665 - log4j v1 - CVE-2021-4104, CVE-2022-23302, CVE-2022-23305, CVE-2022-23307 • 33735326 - log4j v2 - CVE-2021-44832 • 33699205 - WebLogic patch set update 12.2.1.3.211222 • 33618953 - OWSM Bundle Patch 12.2.1.3.211129 • 32997257 - ADF bundle patch 12.2.1.3.210614 • 33591009 - Coherence 12.2.1.3 Cumulative Patch 17 (12.2.1.3.17) • 33590225 - ADF bundle patch 12.2.1.3.211119 • 33671996 - WebLogic overlay patch for October 2021 PSU for CVE-2021-44228 and CVE-2021-45046 • 32772477 - FMW Platform 12.2.1.3.0 SPU for April 2021 CPU • 32651962 - FMW common third-party SPU 12.2.1.3.0 for April 2021 CPU • 26394536 - Oracle Process Mgmt and Notification patch • 31544340 - ADR for WebLogic JULY CPU 2020 • 30186876 - SOA composer patch • 30252137 - XML Developers Kit patch • 27263211 - Enterprise Manager patch • 31464643 - Merge request on top of 12.2.1.3.0 for bugs 29011959 and 30385564 • 1221316 (33286132) - Coherence 12.2.1.3 cumulative patch 16 (12.2.1.3.16) • 33313934 - ADF bundle patch 12.2.1.3.210903 • 33412599 - WebLogic patch set update 12.2.1.3.210929 • 29840258 - RCU patch • 28659321 - Managed Server logs patch • 32397127 - OPSS patch for April 2021 • 26045997 - JDBC patch • 27608287 - Upgrade Assistant Readiness patch • 24738720 - Enterprise Manager patch • 32917014 - OWSM bundle patch 12.2.1.3.210524 • 18345580 - XML Developers Kit patch • 26355633 - JDBC patch • 26287183 - JDBC patch • 26261906 - Universal Connection Pool patch • 26051289 - JDBC patch

Table 8-2 (Cont.) Patches for 11.1.1.7.0 and 12.2.1.3 releases

WebLogic Server Version	Patches
10.3.6.0 (11.1.1.7.0)	bsu: • 21Y4 - WebLogic patch set update 10.3.6.0.211019 • CW7X - ADR for WebLogic Server • I1EV - WebLogic Server patch • SY38 - WebLogic Server patch • TTGM - Patch to update certgen and related artifacts to support new demo certs. opatch: • 27214515 - ADF patch • 27846936 - OPSS bundle patch • 17617649 - FMW bug fixes • 22577934 - FMW Control patch • 22852289 - FMW bug fixes

To identify the version of WebLogic Server on which your domain is running:

1. Sign in to the WebLogic Server Administration Console for your domain. See Access the WebLogic Console.
2. From the Domain Structure panel, expand **Environments**, and then click **Servers**.
3. Click the administration server.
4. Click **Monitoring**.
5. Locate the **Patch List**.

About Patching Utility Tool

Oracle WebLogic Server for OCI 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.

> **Note:**
>
> - The patching utility tool is available only for Oracle WebLogic Server for OCI instances provisioned after release 20.4.3 (December 23, 2020).
> - For instances provisioned prior to release 20.4.3 (before December 23, 2020), download the WebLogic Server patches by using the My Oracle Support website.
> - For UCM only license users, who does not have access to My Oracle Support, open a Support Ticket to get the unique link to download the quarterly Patch Set Updates (PSUs).

From release 21.3.3 onwards, you can use the patching utility tool to download patches for custom images that are created from Oracle WebLogic Server for OCI instance.

For custom images created from existing instances with older version of patching tool, the patching utility does not work. So, before you create custom images from existing instances, Oracle recommends you to upgrade the patching utility to the latest version using the `patch-utils upgrade` command. With the patching tool upgrade, the images use the latest version of patching utility tool with two keys for decryption.

For existing instances created from the Marketplace, you need not upgrade the patching tool. You can use the older version of the patching tool to download the patches.

You can use this patching tool utility on the WebLogic Server compute instance and the bastion instance.

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.

These patches can then be applied only on the WebLogic Server VMs using the patching utility tool.

Patch Management Using Patching Utility

Use the patching utility tool in Oracle WebLogic Server for OCI to list, download, apply, and roll back patches. You can also view the patching tool version and upgrade the patching tool.

> **Note:**
>
> Using this utility, you cannot apply the patch for a bastion instance.

You can perform the following tasks using the patching utility tool:

- View Patching Tool Version
- Configure Initial Setup
- List Patches
- List Pending Patches
- View Patch Details
- Download Patches
- Apply Patches
- Roll Back Patches
- Upgrade Patching Tool

View Patching Tool Version

You can view the build version along with the Oracle license and copyright information.

1. Connect to the compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

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

You can configure the region from where to download the patches and create the configuration file in the specified Middleware Home.

The user can download the patches from the five regions, *us-phoenix-1*, *us-ashburn-1*, *eu-frankfurt-1*, *ap-mumbai-1*, *ap-tokyo-1*, *sa-saopaulo-1*, only.

> **Note:**
>
> You must set up the configuration before you run the patching tool on any provisioned VM.

1. Connect to the compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

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

You can 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 compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

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 and BSU (BEA Smart Update) for 11g.

  ```
  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.

  > **Note:**
  >
  > Ensure that you have upgraded the patching tool before you use the `patch utils` to list the latest patches. See Upgrade Patching Tool.

  For WebLogic Server compute instances, the latest patches are listed for a given Middleware Home. In case of multiple Middleware Homes, you must 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 bastion 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)
```

List Pending Patches

You can list the pending patches in the patching tool repository that need to be applied on a middleware home.

> **Note:**
>
> Ensure that you have upgraded the patching tool before using the `patch utils` to list pending patches. See Upgrade Patching Tool.

1. Connect to the compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

2. Run the following command:

   ```
   patch-utils diff
   ```

   Sample output:

   ```
   
   Patch Id         Patch Components     Description
   ----------       ------------------
   --------------------------------------------------------------------------
   -
   <Patch number>     WLS, FMW          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, FMW          Merge Request on Top of 12.2.1.4.0
   for Bugs 34280277 26354548 26629487
                                          29762601
   <Patch number>     WLS, FMW          Coherence 12.2.1.4 Cumulative Patch
   15 (12.2.1.4.15)
   ```

   To list the pending patches for a given Middleware Home, use the following command:

   ```
   patch-utils diff -m <MW_HOME>
   ```

   If you don't use `-m` option, you must first set up the configuration using the provided Middleware Home. See Configure Initial Setup.

   When you apply the latest patches, if the Middleware Home is not present, you receive the following error message:

   ```
   Unable to find middleware home [<MW_HOME>]. Please run on node with
   Middleware home present or to update the middleware home, run - patch-
   utils setup.
   ```

View Patch Details

You can 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 compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

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

You can 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 compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

2. Download patches.

   ```
   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']
```

Apply Patches

You can apply the latest patches on a given Middleware Home.

The patches that are already installed in the Middleware Home are not applied. You can apply multiple patches by specifying the patch numbers as comma separated lists. The patches are applied in the order they are specified.

In case of private subnets that do not have network access to outside of Oracle Cloud, you must download the patching zip file on the bastion host, and copy the zip file that has the tool and the VM key to the Weblogic Server VM in the private subnet. After download of patches is complete, you can run `patch-utils apply -f <downloaded zip path>` to apply the patch.

When you run the apply command, if you receive an error that the Weblogic servers are running on the host, you must stop the servers. See. Start and Stop a Domain.

1. Connect to the compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

2. Run the following commands to apply patches:
   - Apply latest patches.

     > **Note:**
     >
     > Ensure that you have upgraded the patching tool before you apply the latest patches. See Upgrade Patching Tool.

     ```
     patch-utils apply -L
     ```

Sample output:

```
Applying latest patches ['<Patch number1>', '<Patch number2>']
---------------------------------------------------------

Thu Nov 25 11:56:25 GMT 2022 - Applying OPATCH upgrade p<Patch
number1>_122140_1394211_Generic
---------------------------------------------------------

Latest OPATCH version p28186730_122140_1394211_Generic is
applied on the middleware home

-----------------------------------------------------------------
--------------------------

Thu Nov 25 11:56:26 GMT 2022 - Applying OPATCH patch p<Patch
number2>_122140_Generic
-----------------------------------------------------------------
---------------------------

Oracle Interim Patch Installer version 13.9.4.2.11
Copyright (c) 2022, 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    : 13.9.4.2.11
OUI version       : 13.9.4.0.0
Log file location : /u01/app/oracle/middleware/cfgtoollogs/
opatch/opatch2022-11-25_11-56-26AM_1.log


OPatch detects the Middleware Home as "/u01/app/oracle/
middleware"

Verifying environment and performing prerequisite checks...
OPatch continues with these patches:   <Patch number2>

Do you want to proceed? [y|n]
Y (auto-answered by -silent)
User Responded with: Y
All checks passed.

Please shutdown Oracle instances running out of this ORACLE_HOME
on the local system.
(Oracle Home = '/u01/app/oracle/middleware')


Is the local system ready for patching? [y|n]
Y (auto-answered by -silent)
User Responded with: Y
Backing up files...
Applying interim patch '<Patch number2>' to OH '/u01/app/oracle/
middleware'
```

```
ApplySession: Optional component(s) [ oracle.sysman.fmw.plugin.soa,
12.2.1.4.0 ] , [ oracle.sysman.fmw.plugin.wc, 12.2.1.4.0 ] ,
[ oracle.sysman.fmw.plugin.wc, 12.2.1.4.0 ] ,
[ oracle.sysman.fmw.plugin.beam, 12.2.1.4.0 ] ,
[ oracle.sysman.fmw.plugin.idm, 12.2.1.4.0 ] ,
[ oracle.sysman.fmw.plugin.idm, 12.2.1.4.0 ] ,
[ oracle.sysman.fmw.plugin.webtier.otd, 12.2.1.4.0 ]  not present in
the Oracle Home or a higher version is found.

Patching component oracle.sysman.fmw.as, 12.2.1.4.0...

Patching component oracle.sysman.fmw.as, 12.2.1.4.0...

Patching component oracle.sysman.fmw.agent, 12.2.1.4.0...

Patching component oracle.sysman.fmw.core, 12.2.1.4.0...
Patch <Patch number2> successfully applied.
Log file location: /u01/app/oracle/middleware/cfgtoollogs/opatch/
opatch2022-11-25_11-56-26AM_1.log

OPatch succeeded.
Oracle Interim Patch Installer version 13.9.4.2.11
Copyright (c) 2022, 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    : 13.9.4.2.11
OUI version       : 13.9.4.0.0
Log file location : /u01/app/oracle/middleware/cfgtoollogs/opatch/
opatch2022-11-25_11-57-30AM_1.log


OPatch detects the Middleware Home as "/u01/app/oracle/middleware"

Invoking utility "cleanup"
OPatch will clean up 'restore.sh,make.txt' files and 'scratch,backup'
directories.
You will be still able to rollback patches after this cleanup.
Do you want to proceed? [y|n]
Y (auto-answered by -silent)
User Responded with: Y

Backup area for restore has been cleaned up. For a complete list of
files/directories
deleted, Please refer log file.

OPatch succeeded.

Successfully applied patch p<Patch number2>_122140_Generic
```

To apply the patches for a given Middleware Home, use the following command:

```
patch-utils apply -L -m <MW_HOME>
```

If you don't use `-m` option, you must first set up the configuration using the provided Middleware Home. See Configure Initial Setup.

When you apply the latest patches, if the Middleware Home is not present, you receive the following error message:

```
Unable to find middleware home [<MW_HOME>]. Please run on node
with Middleware home present or to update the middleware home,
run - patch-utils setup.
```

- Apply OPatch.
  This command leverages the OPATCH utility found in the Middleware Home to apply individual patches.

  ```
  patch-utils apply -l <Patch ID>
  ```

  Sample output:

  ```
  Applying OPATCH patch <OPatch ID>
  Oracle Interim Patch Installer version <OPatch 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"
  Verifying environment and performing prerequisite checks...
  OPatch succeeded.
  ```

- Apply local patch.
  This command applies the patch using the local zip file.

  > **Note:**
  >
  > In case of private subnets that do not have network access to outside of Oracle Cloud, before you apply the local patch, you must download the patching zip file on the bastion host, and copy the zip file that has the tool and the VM key to the Weblogic Server VM in the private subnet.

  ```
  patch-utils apply -f /tmp/<Patch ID>.zip
  ```

Sample output:

```
Listing opatch inventory before applying new patches
Oracle Interim Patch Installer version <OPatch 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"
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
Applying OPATCH patch <_ OPatch ID.zip>
Oracle Interim Patch Installer version <OPatch 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"
Verifying environment and performing prerequisite checks...
OPatch succeeded.
```

Roll Back Patches

You can roll back the latest applied patches.

The `patch-utils rollback -L` command rolls back all patches that are applied using the `patch-utils apply -L` command.

Note:

Ensure that you have upgraded the patching tool before you roll back the latest patches. See Upgrade Patching Tool.

1. Connect to the compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

2. Run the following command to roll back the applied patches:

   ```
   patch-utils rollback -L
   ```

   Sample output:

   ```
   Rolling back last applied patches <Patch number1>,<Patch number2>
   --------------------------------------------------------------------
   ------------------------

   Thu Nov 25 11:35:55 GMT 2022 - Rollback OPATCH patch <Patch
   number1>,<Patch number2>
   --------------------------------------------------------------------
   ------------------------

   Oracle Interim Patch Installer version 13.9.4.2.11
   Copyright (c) 2022, 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    : 13.9.4.2.11
   OUI version       : 13.9.4.0.0
   Log file location : /u01/app/oracle/middleware/cfgtoollogs/opatch/
   opatch2022-11-25_11-35-55AM_1.log


   OPatch detects the Middleware Home as "/u01/app/oracle/middleware"

   Following patches are not present in the Oracle Home.
   <Patch number2>

   Patches will be rolled back in the following order:
   <Patch number1>
   The following patch(es) will be rolled back: <Patch number1>

   Please shutdown Oracle instances running out of this ORACLE_HOME on
   the local system.
   (Oracle Home = '/u01/app/oracle/middleware')


   Is the local system ready for patching? [y|n]
   Y (auto-answered by -silent)
   ```

```
User Responded with: Y

Rolling back patch <Patch number>...

RollbackSession rolling back interim patch '<Patch number1>' from OH
'/u01/app/oracle/middleware'

Patching component oracle.sysman.fmw.as, 12.2.1.4.0...

Patching component oracle.sysman.fmw.as, 12.2.1.4.0...

Patching component oracle.sysman.fmw.agent, 12.2.1.4.0...

Patching component oracle.sysman.fmw.core, 12.2.1.4.0...
RollbackSession removing interim patch '<Patch number1>' from inventory
Log file location: /u01/app/oracle/middleware/cfgtoollogs/opatch/
opatch2022-11-25_11-35-55AM_1.log

OPatch succeeded.
Oracle Interim Patch Installer version 13.9.4.2.11
Copyright (c) 2022, 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    : 13.9.4.2.11
OUI version       : 13.9.4.0.0
Log file location : /u01/app/oracle/middleware/cfgtoollogs/opatch/
opatch2022-11-25_11-36-44AM_1.log


OPatch detects the Middleware Home as "/u01/app/oracle/middleware"

Invoking utility "cleanup"
OPatch will clean up 'restore.sh,make.txt' files and 'scratch,backup'
directories.
You will be still able to rollback patches after this cleanup.
Do you want to proceed? [y|n]
Y (auto-answered by -silent)
User Responded with: Y

Backup area for restore has been cleaned up. For a complete list of files/
directories
deleted, Please refer log file.

OPatch succeeded.
```

If the roll back command fails for a particular patch rollback, the original state is restored and a message stating the reason for failure is displayed.

To roll back the patches for a given Middleware Home, use the following command:

```
patch-utils rollback -L -m <MW_HOME>
```

If you don't use `-m` option, you must first set up the configuration using the provided Middleware Home. See Configure Initial Setup.

When you roll back the latest applied patches, if the Middleware Home is not present, you receive the following error message:

```
Unable to find middleware home [<MW_HOME>]. Please run on node with
Middleware home present or to update the middleware home, run -
patch-utils setup.
```

Upgrade Patching Tool

You can upgrade the patching tool utility to the latest version.

1. Connect to the compute instance or the bastion instance as the `opc` user.

   ```
   ssh -i path_to_private_key opc@node_public_ip
   ```

   Or,

   ```
   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
   ```

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

License Information

Learn about the licensed third-party technology associated with Oracle WebLogic Server for OCI.

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 OCI 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
Simon Kelly	dnsmasq	GNU General Public License Version 3

B

Configure SSL for a Domain

Secure Socket Layer (SSL) is the most commonly-used method of securing data sent across the internet. For domains created before June2020, you can configure SSL between clients and the load balancer used to access your Oracle WebLogic Server for OCI domain.

> **Note:**
>
> This procedure applies only to domains that were created before June 29, 2020.
>
> To set up custom SSL for Oracle WebLogic Server for OCI instances, see Overview of Configuring SSL in WebLogic Server.

In this configuration, SSL connections (the HTTPS protocol) terminate at the load balancer. Connections from the load balancer to the compute instances running Oracle WebLogic Server do not use SSL; they use the HTTP protocol.

If you selected the **Prepare Load Balancer for HTTPS** option when creating the domain, then you only need to perform these tasks:

- Add a Certificate to the Load Balancer

If you did not select this option when creating the domain, then you must perform all of the tasks:

- Create an HTTPS Listener for the Load Balancer
- Add a Certificate to the Load Balancer
- Update the App Gateway for HTTPS (if the domain uses Oracle Identity Cloud Service)

Create an HTTPS Listener for the Load Balancer

Update the load balancer for your domain. Create a listener for the HTTPS port, and then configure the SSL request headers for Oracle WebLogic Server.

> **Note:**
>
> This procedure applies only to domains that were created before June 2020. The steps are required only if you did *not* select the **Prepare Load Balancer for HTTPS** option when creating the domain.

The SSL request headers instruct WebLogic Server to use the HTTPS protocol in external URLs that it generates, such as in web application links.

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Load Balancers**.

3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack was initially created, this might be the same compartment that contains the compute instances for the domain.

4. Click the load balancer that was provisioned as part of your stack, `*prefix*-lb`.
5. Click **Rule Sets**.
6. Click **Create Rule Set**
7. For **Name**, enter `SSLHeaders`.
8. Click **Specify Request Header Rules**.
9. Specify these header parameters.
   - **Action**: Add Request Header
   - **Header**: `WL-Proxy-SSL`
   - **Value**: `true`
10. Click **Another Request Header Rule**.
11. Specify these header parameters.
    - **Action**: Add Request Header
    - **Header**: `is_ssl`
    - **Value**: `ssl`
12. Click **Create**, and then click **Close**.
13. Click **Listeners**.
14. Click **Create Listener**
15. For **Name**, enter `https`.
16. For **Port**, enter `443`.
17. For **Backend Set**, select the backend resource created for the load balancer.
18. Click **Create**, and then click **Close**.
19. After the `https` listener is created, edit it.
20. Click **Additional Rule Set**, and then select `SSLHeaders`.
21. Click **Save Changes**, and then click **Close**.
22. If you provisioned a new load balancer subnet as part of your stack, update the security list for this subnet and permit access to the HTTPS port.
    a. From the navigation menu, select **Networking**, and then click **Virtual Cloud Networks**.
    b. Click the virtual cloud network (VCN) used by your domain.
    c. Click **Security Lists**.
    d. Click the load balancer security list that was provisioned for your stack.
       - *prefix*`-lb-security-list`, if you created a single regional subnet
       - *prefix*`-wls-lb-security-list-1` and *prefix*`-wls-lb-security-list-2`, if you created subnets for specific availability domains

e. Edit the existing ingress rule for port 80.

f. Change the **Destination Port Range** to 443.

g. Click **Save Changes**.

If you want to delete this stack at a later time, you will not be able to destroy the stack using Resource Manager. Because of the changes to the load balancer resources, you will have to manually delete the load balancer.

See these topics in the Oracle Cloud Infrastructure documentation:

- Managing Listeners
- Managing Rule Sets
- Security Lists

Add a Certificate to the Load Balancer

Upload your SSL certificate, and then associate the certificate with the HTTPS listener.

> **Note:**
>
> This procedure applies only to domains that were created before June 2020.

You can use a custom, self-signed SSL certificate, or a certificate that you've obtained from a Certificate Authority (CA). For production WebLogic Server environments, Oracle recommends that you use a CA-issued SSL certificate, which reduces the chances of experiencing a man-in-the-middle attack.

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Load Balancers**.
3. Select the **Compartment** in which the network resources for your domain were created.

   Depending on how the stack was initially created, this might be the same compartment that contains the compute instances for the domain.
4. Click the load balancer that was provisioned as part of your stack, `*prefix*-lb`.
5. Click **Certificates**.
6. Click **Add Certificate**.
7. Enter a name for your certificate.
8. Either upload the certificate file, or paste its contents into the text area.
9. If applicable, specify a CA certificate or a private key file.

   For example, if you are using a self-signed certificate, upload the corresponding private key file. See Managing SSL Certificates in the Oracle Cloud Infrastructure documentation.
10. Click **Add Certificate**, and then click **Close**.
11. After the certificate was successfully added, click **Listeners**.
12. Edit the `https` listener.
13. Click **Use SSL**, and then select your new certificate.

14. Click **Save Changes**, and then click **Close**.
15. If a listener exists named `http`, delete this listener.

    This is the default load balancer listener if you did not select the **Prepare Load Balancer for HTTPS** option when creating the domain.

You cannot modify an existing load balancer certificate. You must add a new certificate, and then associate the listener with the new certificate.

Update the App Gateway for HTTPS

If your Oracle WebLogic Server domain uses Oracle Identity Cloud Service for authentication, update and restart the App Gateway on each compute instance in the domain.

> **Note:**
>
> This procedure applies only to domains that were created before June 2020. The steps are required only if both of these are true:
>
> - This procedure applies only to domains that were created before June 2020.
> - You did *not* select the **Prepare Load Balancer for HTTPS** option when creating the domain.
> - You selected the **Enable Authentication Using Identity Cloud Service** option when creating the domain.

1. Open an SSH connection to the first compute instance in the domain, as the `opc` user.

   Example:

   ```
   ssh -i mykey opc@203.0.113.13
   ```

2. Create a backup of the folder `/u01/data/cloudgate_config`.

   ```
   sudo cp -avr /u01/data/cloudgate_config /u01/data/
   cloudgate_config_bak
   ```

3. Edit the file `/u01/data/cloudgate_config/appgateway-env`.
4. Edit the variable named `CG_CALLBACK_PREFIX`. Replace `http` with `https`.

   ```
   CG_CALLBACK_PREFIX=https://%hostid%
   ```

5. Stop and remove the App Gateway container.

   ```
   sudo docker container stop appgateway
   sudo docker container rm appgateway
   ```

6. Delete the contents of the folder `/u01/data/cloudgate_config`, *except* for the following files.

- `appgateway-env`
- `cwallet.sso`
- `origin_conf`

7. Start the App Gateway container.

   ```
   sudo /opt/scripts/idcs/run_cloudgate.sh
   ```

8. Repeat from **step 1** for all remaining compute instances in this domain.

C

Script Files

This section list all the required script files required for Oracle WebLogic Server for OCI.

Topics:

- Script File to Validate Network Setup
- Script File to Create Policies for Autoscaling Functions Post Provisioning

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 OCI.

You can copy the following scripts in Cloud Shell to perform the validation. For example, copy the scripts and save the file as `validate.sh`.

> **Note:**
>
> This script applies only to Oracle WebLogic Server for OCI releases 22.4.1 and earlier. For the script applicable to the later releases of Oracle WebLogic Server for OCI, see Network Validation Script.

```
# ##################################################
# Script to validate existing WebLogic and DB subnets meet the pre-requisite
for
# provisioning and proper functioning of WebLogic Server for Oracle Cloud
Infrastructure.
#

# Set Flags
# -----------------------------------
# Flags which can be overridden by user input.
# Default values are below
# -----------------------------------
DB_PORT=1521
ATP_DB_PORT=1522
SSH_PORT=22
T3_PORT=9071
WLS_LB_PORT=7003
LB_PORT=443
NETWORK_COMPARTMENT_OCID=""
WLS_SUBNET_OCID=""
DB_SUBNET_OCID=""
APP_DB_SUBNET_OCID=""
BASTION_SUBNET_OCID=""
```

```
BASTION_HOST_IP_CIDR=""
LB_SUBNET_OCID=""
FSS_SUBNET_OCID=""

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 vcn_ocid=""
  local vcn_compartment_ocid=""

  is_private_subnet=$(oci network subnet get --subnet-id ${WLS_SUBNET_OCID}
| jq -r '.data["prohibit-public-ip-on-vnic"]')

  if [[ $is_private_subnet = true ]]
  then
    vcn_ocid=$(oci network subnet get --subnet-id ${WLS_SUBNET_OCID} | jq -r
'.data["vcn-id"]')
    vcn_compartment_ocid=$(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

    # WLS subnet should be using either NAT or service gateway or both in
its routetable
    rt_ocid=$(oci network subnet get --subnet-id ${WLS_SUBNET_OCID} | jq -r
'.data["route-table-id"]')
    rt_rules=$(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
    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 WLS subnet.
# Without Internet gateway in WLS 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 vcn_ocid=""
  local vcn_compartment_ocid=""

  vcn_ocid=$(oci network subnet get --subnet-id ${WLS_SUBNET_OCID} |
jq -r '.data["vcn-id"]')
  vcn_compartment_ocid=$(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 ingress
rules of specified seclist or nsg.
#
# Args:
#     seclist_or_nsg_ocid:  OCID for the security list or nsg.
#     port: destination port to check
#     source: Source CIDR (either block/range of IPs or single IP (with /32
suffix)
#     ocid_type: Valid values: "nsg" for Network Security Group OCID,
"seclist" for Security List OCID (default)
#
# Returns:
#   0|1
####################################################
function check_tcp_port_open_in_seclist_or_nsg() {
  local seclist_or_nsg_ocid=$1
  local port=$2
  local source=$3
  local ocid_type=$4
  local port_is_open=false
  local tcp_protocol="6"

  if [[ $ocid_type = "nsg" ]]; then
      ingress_rules=$(oci network nsg rules list --nsg-
id $seclist_or_nsg_ocid | jq -r '.data')
  else
      ingress_rules=$(oci network security-list get --security-list-
id $seclist_or_nsg_ocid | jq -r '.data["ingress-security-rules"]')
  fi

  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
}

####################################################
# 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 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_or_nsg $seclist_ocid "${port}"
"$source_cidr" "seclist")
      else # protocol param is non empty then udp
          port_found_open=$
(check_udp_port_open_in_seclist $seclist_ocid "${port}" "$source_cidr")
      fi
    fi
  done
  echo $port_found_open
}

####################################################
# Validates if the ATP_PORT is open for the WLS subnet CIDR.
# This is applicable for ATP with private endpoint only.
# Args:
#     subnet: ATP private endpoint subnet
#     atp_id: ATP OCID to check port access
#     source_cidr: WLS subnet CIDR
#
# Returns:
#   0|1
####################################################
function validate_atp_port_access() {
  local port_found_open=1
  local subnet=$1
  local atp_id=$2
  local source_cidr=$3

  nsg_list=$(oci db autonomous-database get --autonomous-database-id $
{atp_id} | jq -c '.data["nsg-ids"]')

  if [[ $nsg_list = "null" ]]; then
```

```
      #skip validation if nsg is not present (ATP has no private endpoint)
      port_found_open=0
  else
      # Convert to bash array
      declare -A nsg_list_array

      while IFS="=" read -r key value
      do
          nsg_list_array[$key]="$value"
      done < <(jq -r 'to_entries|map("\(.key)=\(.value|tostring)")|.[]' <<<
"$nsg_list")

      # Check the NSG ingress rules for specified destination port is open
for access by source CIDR
      for nsg_ocid in "${nsg_list_array[@]}"
      do
        if [[ $port_found_open -ne 0 ]]; then
          port_found_open=$(check_tcp_port_open_in_seclist_or_nsg $nsg_ocid
"${ATP_DB_PORT}" "$source_cidr" "nsg")
        fi
      done

      if [[ port_found_open -ne 0 ]]; then
          # If ATP port is not opened by NSG ingress rules, check subnet
security list
          # doc: "if you choose a subnet with a security list, the security
rules for the database
          # will be a union of the rules in the security list and the NSGs."
          port_found_open=$(validate_subnet_port_access ${subnet} $
{ATP_DB_PORT} ${source_cidr})
      fi
  fi

  echo $port_found_open
}

####################################################
# Validates if custom resolver has 169.254.169.254 dns entry for WLS subnet.
# Args:
#     subnet_id:    Subnet Id
#
# Returns:
#   0|1
####################################################
function validate_dhcp_options() {
  local subnet_id=$1
  dhcp_id=$(oci network subnet get --subnet-id ${subnet_id} | jq -r
'.data["dhcp-options-id"]')
  resolver_type=$(oci network dhcp-options get --dhcp-id ${dhcp_id} | jq -r
'.data["options"][0]["server-type"]')

  if [[ $resolver_type == "CustomDnsServer" ]]; then
    dhcp_options=$(oci network dhcp-options get --dhcp-id ${dhcp_id})
    dns_servers=($(jq -r '.data["options"][0]["custom-dns-servers"]|.[]' <<<
"$dhcp_options"))
```

```
    for dns_server in "${dns_servers[@]}"
    do
      if [[ $dns_server == '169.254.169.254' ]];then
        echo 0
        return
      fi
    done
  else
    echo 0
    return
  fi
  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 WebLogic subnet (and
optionally DB Subnet) are setup correctly.
 ${bold}Options:${reset}
  -w, --wlssubnet     WebLogic Subnet OCID (Required)
  -d, --ocidbsubnet   OCI Database System Subnet OCID
  -a, --appdbsubnet   Application Database Subnet OCID
  -t, --atpid         ATP DB OCID
  -b, --bastionsubnet Bastion Subnet OCID
  -i, --bastionipcidr Bastion Host IP CIDR (should be suffixed
with /32)
  -l, --lbsubnet      LB Subnet OCID
  -f, --fsssubnet     FSS Subnet OCID
```

```
      --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) ${version}"; exit 0 ;;
    -w|--wlssubnet) shift; WLS_SUBNET_OCID=${1} ;;
    -d|--ocidbsubnet) shift; DB_SUBNET_OCID=${1} ;;
    -a|--appdbsubnet) shift; APP_DB_SUBNET_OCID=${1} ;;
    -t|--atpid) shift; ATP_OCID=${1} ;;
    -b|--bastionsubnet) shift; BASTION_SUBNET_OCID=${1} ;;
    -i|--bastionipcidr) shift; BASTION_HOST_IP_CIDR=${1} ;;
    -l|--lbsubnet) shift; LB_SUBNET_OCID=${1} ;;
    -f|--fsssubnet) shift; FSS_SUBNET_OCID=${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 ${WLS_SUBNET_OCID} ]]
then
  echo "One or more required params are not specified."
  usage >&2
  exit
fi

# Check if service or NAT gateway exists in WLS subnet's VCN.
res=$(validate_service_or_nat_gw_exist)

if [[ $res -eq 1 ]]
then
  echo "ERROR: Missing Service or NAT gateway in the VCN of the
private WLS subnet [$WLS_SUBNET_OCID]"
elif [[ $res -eq 2 ]]
then
  echo "ERROR: Private WLS subnet [$WLS_SUBNET_OCID] does not use NAT
```

```
or Service gateway"
elif [[ $res -eq 3 ]]
then
  echo "ERROR: Service Gateway in VCN of private WLS subnet
[$WLS_SUBNET_OCID] does not allow access to all services in Oracle services
network"
elif [[ $res -eq 4 ]]
then
  echo "ERROR: Route Rule of private WLS subnet [$WLS_SUBNET_OCID] does not
use 'ALL Services in Oracle services network' destination"
fi

#Check for custom resolver
res=$(validate_dhcp_options ${WLS_SUBNET_OCID})

if [[ $res -ne 0 ]]
then
  echo "WARNING: Missing OCI dns server [169.254.169.254] in the DNS Servers
entries for the custom resolver"
fi

# Check if internet gateway exists in WLS subnet's VCN.
res=$(validate_internet_gw_exist)

if [[ $res -ne 0 ]]
then
  echo "WARNING: Missing internet gateway in the VCN of the WLS subnet
[$WLS_SUBNET_OCID]"
fi

wls_subnet_cidr_block=$(oci network subnet get --subnet-id $
{WLS_SUBNET_OCID} | jq -r '.data["cidr-block"]')

# Check if SSH port is open for access by WLS subnet CIDR
res=$(validate_subnet_port_access ${WLS_SUBNET_OCID} ${SSH_PORT} $
{wls_subnet_cidr_block})

if [[ $res -ne 0 ]]
then
  echo "ERROR: Port ${SSH_PORT} is not open for access by WLS Subnet CIDR
[$wls_subnet_cidr_block] in WLS Subnet [$WLS_SUBNET_OCID]"
fi

# Check if t3 port is open for access by WLS subnet CIDR
res=$(validate_subnet_port_access ${WLS_SUBNET_OCID} ${T3_PORT} $
{wls_subnet_cidr_block})
if [[ $res -ne 0 ]]
then
  echo "ERROR: Port ${T3_PORT} is not open for access by WLS Subnet CIDR
[$wls_subnet_cidr_block] in WLS Subnet [$WLS_SUBNET_OCID]"
fi

# Check if DB port is open for access by WLS 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} $
{wls_subnet_cidr_block})
  if [[ $res -ne 0 ]]
  then
    echo "ERROR: DB port ${DB_PORT} is not open for access by WLS
Subnet CIDR [$wls_subnet_cidr_block] in DB Subnet [$DB_SUBNET_OCID]"
  fi
fi

# Check if DB port is open for access by WLS subnet CIDR in
Application DB subnet (only if Application DB subnet is provided)
if [[ -n ${APP_DB_SUBNET_OCID} ]]
then
  res=$(validate_subnet_port_access ${APP_DB_SUBNET_OCID} ${DB_PORT} $
{wls_subnet_cidr_block})
  if [[ $res -ne 0 ]]
  then
    echo "ERROR: DB port ${DB_PORT} is not open for access by WLS
Subnet CIDR [$wls_subnet_cidr_block] in Application DB Subnet
[$APP_DB_SUBNET_OCID]"
  fi
fi

# Check if port for ATP with private endpoint is open for access by
WLS subnet CIDR. The ATP has associated NSG.
# No-op for ATP with no private endpoint (network security group is
not present for ATP)
if [[ -n ${ATP_OCID} ]]
then
  atp_subnet_ocid=$(oci db autonomous-database get --autonomous-
database-id ${ATP_OCID} | jq -r '.data["subnet-id"]')
  if [[ $atp_subnet_ocid != null ]]; then
    #skip validation if ATP has no private endpoint (no subnet-id)
    res=$(validate_atp_port_access ${atp_subnet_ocid} ${ATP_OCID} $
{wls_subnet_cidr_block})
    if [[ $res -ne 0 ]]; then
      echo "ERROR: ATP port ${ATP_PORT} is not open for access by WLS
Subnet CIDR [$wls_subnet_cidr_block] in ATP Subnet [$
{atp_subnet_ocid}]"
    fi
  fi
fi

# Check if SSH port is open for access by Bastion Subnet or Host CIDR
in WLS Private Subnet
if [[ -n ${BASTION_SUBNET_OCID} || -n ${BASTION_HOST_IP_CIDR} ]]
then
  is_private_subnet=$(oci network subnet get --subnet-id $
{WLS_SUBNET_OCID} | jq -r '.data["prohibit-public-ip-on-vnic"]')

  if [[ $is_private_subnet = true ]]
  then
    # Check if Bastion subnet has SSH port open for 0.0.0.0/0
    if [[ -n ${BASTION_SUBNET_OCID} ]]
    then
      all_ips="0.0.0.0/0"
      res=$(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 [$BASTION_SUBNET_OCID]"
      fi
    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 WLS subnet
    res=$(validate_subnet_port_access ${WLS_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 WLS Subnet [$WLS_SUBNET_OCID]"
    fi
  fi
fi

# Check if 7003 port is open for access by LB Subnet or Host CIDR in WLS
Subnet
if [[ -n ${LB_SUBNET_OCID} ]]
then
    lbsubnet_cidr_block=$(oci network subnet get --subnet-id "$
{LB_SUBNET_OCID}" | jq -r '.data["cidr-block"]')
    res=$(validate_subnet_port_access "${WLS_SUBNET_OCID}" ${WLS_LB_PORT} "$
{lbsubnet_cidr_block}")
    if [[ $res -ne 0 ]]
    then
      echo "ERROR: LB port ${WLS_LB_PORT} is not open for access by LB
Subnet CIDR - [$lbsubnet_cidr_block] in WLS Subnet [$WLS_SUBNET_OCID]"
    fi
fi

# Check if LB Subnet port is open in LB Subnet - 443
```

```
if [[ -n ${LB_SUBNET_OCID} ]]
then
    all_ips="0.0.0.0/0"
    res=$(validate_subnet_port_access "${LB_SUBNET_OCID}" ${LB_PORT} "$
{all_ips}")
    if [[ $res -ne 0 ]]
    then
      echo "ERROR: Port [$LB_PORT] is not open for 0.0.0.0/0 in LB
Subnet CIDR [${LB_SUBNET_OCID}]"
    fi
fi

# FSS subnet verification - Checking All TCP Ports are open in FSS
SUBNET OCID for VCN CIDR
if [[ -n ${FSS_SUBNET_OCID} ]]
then
  vcn_ocid=$(oci network subnet get --subnet-id "${WLS_SUBNET_OCID}" |
jq -r '.data["vcn-id"]')
  vcn_cidr=$(oci network vcn get --vcn-id "${vcn_ocid}" | jq -r
'.data["cidr-block"]')

  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
fi
```

Script File to Create Policies for Autoscaling Functions Post Provisioning

Use the script file to create policies for autoscaling functions after creating the Oracle WebLogic Server for OCI domain.

If the **OCI Policies** check box is not selected during domain creation (`create_policies` set to `false`), you can use the script file to create function's dynamic group and policies after creating the domain. The script verifies if the stack has `create_policies` set to `false`, in which case it returns an error that policies would be automatically created via stack. However, you can use the `--force` option to override and create dynamic group and policies when `create_policies == true`.

Copy the following scripts in Cloud Shell to create polices for autoscaling. For example, copy the scripts and save the file as `create_autoscaling_policies.py`

> **Note:**
>
> For instructions on how to use the script to create function's dynamic group and policies, see Create Dynamic Groups and Policies for Observability.

After you run the script, you must enable alarms from the Oracle Cloud Infrastructure console. See To enable an alarm in Oracle Cloud Infrastructure documentation.

```
"""
#
# Copyright (c) 2022, Oracle Corporation and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at https://
oss.oracle.com/licenses/upl.
"""
import argparse
import logging
import oci
import os
import sys
import textwrap
from oci.identity import models as identity_models

logger = logging.getLogger()
logging.basicConfig()
REGION = ''


def get_config():
    if REGION != '':
        return {"region": REGION}
    return {}


def get_variable(variables, name, default=None):
    """
    First read from stack variables
    If not found and default provided, return default
    :param variables:
    :param name:
    :param default:
    :return:
    """
    ret = None
    found = True

    if name in variables:
        ret = variables.get(name)
    else:
        if default is not None:
            return default
        else:
            logger.info("ERROR: value for the variable [%s] could not
be found!" % (name))
            found = False

    if found:
        logger.debug("Value of variable [%s] is [%s]" % (name, ret))

    return ret


def inspect_stack(signer, stack_ocid):
    client =
oci.resource_manager.ResourceManagerClient(config=get_config(),
signer=signer)
    try:
        stack = client.get_stack(stack_id=stack_ocid)
    except(Exception, ValueError) as ex:
        logger.exception("ERROR: accessing resource manager stack
failed")
        raise
    resp = stack.data

    return resp


def create_functions_dynamic_group(signer, stack_compartment_id,
tenancy_id, service_name):
    """
    Creates functions dynamic group in root compartment.

    :param signer:
    :param stack_compartment_id:
    :param tenancy_id:
    :param service_name:
    :return:
    """
    client = oci.identity.IdentityClient(config={}, signer=signer)
    client_composite_ops =
oci.identity.IdentityClientCompositeOperations(client)
    dg_name = "{0}-functions-dynamic-group-manual".format(service_name)
    try:
        create_dynamicgrp_resp =
client_composite_ops.create_dynamic_group_and_wait_for_state(

create_dynamic_group_details=identity_models.CreateDynamicGroupDetails(
                compartment_id=tenancy_id,
                description="Autoscaling Functions Dynamic Group of
stack: {0}".format(service_name),
                matching_rule="All {resource.type = 'fnfunc', "
                              "resource.compartment.id='%s'}" %
(stack_compartment_id),
                name=dg_name
            ),

wait_for_states=[identity_models.DynamicGroup.LIFECYCLE_STATE_ACTIVE]
        )
    except(Exception, ValueError) as ex:
        logger.exception("Failed to create functions dynamic group
{0}".format(dg_name))
        raise
    return create_dynamicgrp_resp.data


def create_policies_for_functions_dynamic_group(signer,
stack_compartment_id, tenancy_id, service_name,
                                                network_compartment_id,

apm_domain_compartment_id=None,
                                                atp_db_compartment_id=None,
                                                ocidb_compartment_id=None,

app_atp_db_compartment_id=None,

app_ocidb_compartment_id=None,
                                                fss_compartment_id=None):
    """
    Creates policies for functions dynamic group.

    :param signer:
    :param stack_compartment_id:
    :param tenancy_id:
    :param service_name:
    :param network_compartment_id:
    :param apm_domain_compartment_id:
    :param atp_db_compartment_id:
    :param ocidb_compartment_id:
    :param app_atp_db_compartment_id:
    :param app_ocidb_compartment_id:
    :param fss_compartment_id:
    :return:
    """
    client = oci.identity.IdentityClient(config={}, signer=signer)
    client_composite_ops =
oci.identity.IdentityClientCompositeOperations(client)
    dg_name = "{0}-functions-dynamic-group-manual".format(service_name)
    try:
        policies = [
            "Allow dynamic-group {0} to inspect tenancies in
tenancy".format(dg_name),
            "Allow dynamic-group {0} to inspect limits in
tenancy".format(dg_name),
            "Allow dynamic-group {0} to manage volume-family in compartment
id {1}".format(dg_name,

              stack_compartment_id),
            "Allow dynamic-group {0} to manage instance-family in
compartment id {1}".format(dg_name,

                      stack_compartment_id),
            "Allow dynamic-group {0} to use app-catalog-listing in
compartment id {1}".format(dg_name,

                       stack_compartment_id),
            "Allow dynamic-group {0} to manage virtual-network-family
in compartment id {1}".format(dg_name,

                             network_compartment_id),
            "Allow dynamic-group {0} to manage load-balancers in
compartment id {1}".format(dg_name,

                     network_compartment_id),
            "Allow dynamic-group {0} to manage orm-family in
compartment id {1}".format(dg_name, stack_compartment_id),
            "Allow dynamic-group {0} to read metrics in compartment id
{1}".format(dg_name, stack_compartment_id),
            "Allow dynamic-group {0} to manage repos in
tenancy".format(dg_name),
            "Allow dynamic-group {0} to manage functions-family in
compartment id {1}".format(dg_name,

                       stack_compartment_id),
            "Allow dynamic-group {0} to manage ons-topics in
compartment id {1}".format(dg_name, stack_compartment_id),
            "Allow dynamic-group {0} to manage instance-agent-command-
family in compartment id {1}".format(dg_name,

                                    stack_compartment_id),
            "Allow dynamic-group {0} to manage alarms in compartment
id {1}".format(dg_name, stack_compartment_id),
            "Allow dynamic-group {0} to manage log-groups in
compartment id {1}".format(dg_name, stack_compartment_id),
            "Allow dynamic-group {0} to use unified-configuration in
compartment id {1}".format(dg_name,

                         stack_compartment_id),
            "Allow dynamic-group {0} to inspect dynamic-groups in
tenancy".format(dg_name),
            "Allow dynamic-group {0} to manage policies in
tenancy".format(dg_name),
            "Allow dynamic-group {0} to use tag-namespaces in
tenancy".format(dg_name)
        ]
        if apm_domain_compartment_id is not None:
            policies.append("Allow dynamic-group {0} to use apm-
domains in compartment id {1}".format(dg_name,

                               apm_domain_compartment_id))
        if atp_db_compartment_id is not None:
            policies.append(
                "Allow dynamic-group {0} to inspect autonomous-
```

```
transaction-processing-family in compartment id {1}".format(
                    dg_name,
                    atp_db_compartment_id))
        if ocidb_compartment_id is not None:
            policies.append("Allow dynamic-group {0} to inspect database-
family in compartment id {1}".format(dg_name,

                                 ocidb_compartment_id))
        if app_ocidb_compartment_id is not None:
            policies.append("Allow dynamic-group {0} to inspect database-
family in compartment id {1}".format(dg_name,

                                 app_ocidb_compartment_id))
        if app_atp_db_compartment_id is not None:
            policies.append(
                "Allow dynamic-group {0} to use autonomous-transaction-
processing-family in compartment id {1}".format(
                    dg_name,
                    app_atp_db_compartment_id))
        if fss_compartment_id is not None:
            policies.append("Allow dynamic-group {0} to manage mount-targets
in compartment id {1}".format(dg_name,

                              fss_compartment_id))
            policies.append("Allow dynamic-group {0} to manage file-systems
in compartment id {1}".format(dg_name,

                             fss_compartment_id))
            policies.append("Allow dynamic-group {0} to manage export-sets
in compartment id {1}".format(dg_name,

                            fss_compartment_id))
        create_policy_resp =
client_composite_ops.create_policy_and_wait_for_state(
            create_policy_details=identity_models.CreatePolicyDetails(
                compartment_id=tenancy_id,
                description="Policy for Autoscaling Functions Dynamic Group
of stack: {0}".format(service_name),
                statements=policies,
                name="{0}-wlsc-policy-manual".format(service_name)
            ),
            wait_for_states=[identity_models.Policy.LIFECYCLE_STATE_ACTIVE]
        )
    except(Exception, ValueError) as ex:
        logger.exception("Failed to create policies for functions dynamic
group {0}".format(dg_name))
        raise
    return create_policy_resp.data


def get_apm_compartment_id(signer, apm_domain_id):
    client = oci.apm_control_plane.ApmDomainClient(config=get_config(),
signer=signer)
    try:
        resp = client.get_apm_domain(apm_domain_id)
    except(Exception, ValueError) as ex:
        logger.exception("Failed to get compartment for APM domain ID
[{0}]".format(apm_domain_id))
        raise
    apm_domain = resp.data
    return apm_domain.compartment_id


def main():
    signer = None
    # delegate token should be present at /etc/oci/delegation_token in
cloud shell
    if os.path.exists('/etc/oci/delegation_token'):
        with open('/etc/oci/delegation_token', 'r') as file:
            delegation_token = file.read()
        signer =
oci.auth.signers.InstancePrincipalsDelegationTokenSigner(delegation_tok
en=delegation_token)
    else:
        logger.error("In the Cloud shell the delegation token does not
exist at location /etc/oci/delegation_token. "
                     "Run the script from the Cloud shell, where you
need to delete the resources.")
        sys.exit(1)

    if signer is None:
        logger.error('Instance principal signer is not initialized')
        sys.exit(1)

    parser = argparse.ArgumentParser(description=textwrap.dedent('''
        This script creates Functions Dynamic Group and Policies for
Autoscaling.
    '''), formatter_class=argparse.RawDescriptionHelpFormatter)
    parser.add_argument('stack_ocid', help='Stack OCID')
    parser.add_argument('-r', '--region', default='', help='Region
other than home region must be specified.')
    parser.add_argument('-d', '--debug', action='store_const',
const=True, help='Enable Debug')
    parser.add_argument('-f', '--force', action='store_const',
const=True, help='Force creation of dynamic group and '

         'policies')

    args = parser.parse_args()

    stack_ocid = args.stack_ocid
    global REGION

    REGION = args.region
    debug = args.debug
    if debug:
        logger.setLevel(logging.DEBUG)
    else:
        logger.setLevel(logging.INFO)
    force = args.force
    # Get stack variables
    resp = inspect_stack(signer, stack_ocid)
    logger.debug(resp)
    variables = resp.variables
    create_policies = get_variable(variables, 'create_policies', "true")

    if create_policies == 'true':
        if not force:
            logger.error('Stack has create_policies enabled so manual 
creation of autoscaling functions dynamic group '
                         'and policies is not required. Bailing out!')
            return
        else:
            logger.warning('Stack has create_policies enabled so manual 
creation of autoscaling functions dynamic '
                           'group and policies is not required.')

    stack_compartment_id = get_variable(variables, 'compartment_ocid')
    tenancy_id = get_variable(variables, 'tenancy_ocid')
    service_name = get_variable(variables, 'service_name')

    # create functions dynamic group
    create_functions_dynamic_group(signer, stack_compartment_id, tenancy_id, 
service_name)

    # create policies for the functions dynamic group
    apm_domain_id = get_variable(variables, 'apm_domain_id', "")
    atp_db_id = get_variable(variables, "atp_db_id", "")
    ocidb_dbsystem_id = get_variable(variables, "ocidb_dbsystem_id", "")
    app_atp_db_id = get_variable(variables, "app_atp_db_id", "")
    appdb_dbsystem_id = get_variable(variables, "appdb_dbsystem_id", "")
    add_fss = get_variable(variables, "add_fss", "false")

    network_compartment_id = get_variable(variables, 
'network_compartment_id', stack_compartment_id)
    apm_domain_compartment_id = get_apm_compartment_id(signer, 
apm_domain_id) if apm_domain_id != '' else None
    atp_db_compartment_id = get_variable(variables, 'atp_db_compartment_id',
                                         stack_compartment_id) if atp_db_id !
= '' else None
    ocidb_compartment_id = get_variable(variables, 'ocidb_compartment_id',
                                        stack_compartment_id) if 
ocidb_dbsystem_id != '' else None
    app_atp_db_compartment_id = get_variable(variables, 
'app_atp_db_compartment_id',
                                             stack_compartment_id) if 
app_atp_db_id != '' else None
    app_ocidb_compartment_id = get_variable(variables, 
'app_ocidb_compartment_id',
                                            stack_compartment_id) if 
appdb_dbsystem_id != '' else None
    fss_compartment_id = get_variable(variables, 'fss_compartment_id',
                                      stack_compartment_id) if add_fss == 
'true' else None
```

```
    create_policies_for_functions_dynamic_group(signer,
stack_compartment_id, tenancy_id, service_name,

network_compartment_id, apm_domain_compartment_id,
                                                atp_db_compartment_id,
                                                ocidb_compartment_id,

app_atp_db_compartment_id,

app_ocidb_compartment_id,
                                                fss_compartment_id)


if __name__ == '__main__':
    main()
```

Script File to Delete Resources

You must create a script file to remove the autoscaling resources before you destroy the stack in Oracle WebLogic Server for OCI.

You can copy the following scripts to the script file, `remove_resources.py` file in Cloud Shell.

```
"""
#
# Copyright (c) 2021, 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.
"""

import os
import sys
import oci
from oci.events import models as event_models
from oci.functions import models as fn_models
from oci.monitoring import models as monitoring_models
from oci.artifacts import models as artifacts_models
from oci.ons import models as ons_models
from oci import pagination
import argparse
import textwrap

REGION = ''


def get_config():
    if REGION != '':
        return {"region": REGION}
    return {}


"""
```

```
Lists and deletes the resources like instances, policies, volumes, VCN
related resources, logs and tags etc..
"""


class CleanUpResources:

    def __init__(self, service_name_prefix):
        self.service_name_prefix = service_name_prefix
        # delegate token should be present at /etc/oci/delegation_token in
cloud shell
        if os.path.exists('/etc/oci/delegation_token'):
            with open('/etc/oci/delegation_token', 'r') as file:
                delegation_token = file.read()
            self.signer =
oci.auth.signers.InstancePrincipalsDelegationTokenSigner(delegation_token=del
egation_token)
        else:
            print("ERROR: In the Cloud shell the delegation token does not
exist at location /etc/oci/delegation_token."
                  "Run the script from the Cloud shell, where you need to
delete the resources.")
            sys.exit(1)
        self.vcn_client = oci.core.VirtualNetworkClient(config=get_config(),
signer=self.signer)
        self.virtual_network_composite_operations =
oci.core.VirtualNetworkClientCompositeOperations(self.vcn_client)
        self.log_client =
oci.logging.LoggingManagementClient(config=get_config(), signer=self.signer)
        self.log_composite_operations =
oci.logging.LoggingManagementClientCompositeOperations(self.log_client)
        self.identity_client = oci.identity.IdentityClient(config={},
signer=self.signer)
        self.identity_client_composite_operations =
oci.identity.IdentityClientCompositeOperations(self.identity_client)
        self.events_client = oci.events.EventsClient(config=get_config(),
signer=self.signer)
        self.events_client_composite_operations =
oci.events.EventsClientCompositeOperations(self.events_client)
        self.fn_client =
oci.functions.FunctionsManagementClient(config=get_config(),
signer=self.signer)
        self.fn_composite_operations =
oci.functions.FunctionsManagementClientCompositeOperations(client=self.fn_cli
ent)
        self.monitoring_client =
oci.monitoring.MonitoringClient(config=get_config(), signer=self.signer)
        self.monitoring_composite_operations =
oci.monitoring.MonitoringClientCompositeOperations(
            client=self.monitoring_client)
        self.artifacts_client =
oci.artifacts.ArtifactsClient(config=get_config(), signer=self.signer)
        self.artifacts_composite_operations =
oci.artifacts.ArtifactsClientCompositeOperations(
            client=self.artifacts_client)
        self.ons_control_plane_client =
oci.ons.NotificationControlPlaneClient(config=get_config(),
signer=self.signer)

    # Lists all the resources based on the service name prefix
    def list_all_resources(self):
        search_client =
oci.resource_search.ResourceSearchClient(config=get_config(),
signer=self.signer)
        running_resources = ["RUNNING", "Running", "AVAILABLE",
"STOPPED", "Stopped", "ACTIVE", "CREATED", "INACTIVE"]
        resource_not_required = ["PrivateIp", "Vnic"]
        # https://docs.oracle.com/en-us/iaas/Content/Search/Concepts/
queryoverview.htm#resourcetypes
        structured_search =
oci.resource_search.models.StructuredSearchDetails(
            query="query all resources where displayname =~
'{}'".format(self.service_name_prefix),
            type='Structured',

matching_context_type=oci.resource_search.models.SearchDetails.MATCHING
_CONTEXT_TYPE_NONE)

        resources = search_client.search_resources(structured_search)
        resources_details = []
        no_of_resources = 0
        # Tags and default route table doesn't start with service
prefix
        tagname_resource = "wlsoci-" + self.service_name_prefix
        default_rt = "Default Route Table for " +
self.service_name_prefix
        # Logs and unified agent config resources use service_prefix
with underscore instead of hyphen
        log_resource = self.service_name_prefix[:-1] + "_"
        print(
            "Resource Name                              Resource
Type                        Resource Lifecycle State
OCID         DOC")
        print(

"======================================================================
=======================================================================
====")
        for resource in resources.data.items:
            resource_name = resource.display_name
            if (resource_name.startswith(
                    self.service_name_prefix) or tagname_resource in
resource_name or default_rt in resource_name or log_resource in
resource_name) and (
                    resource.lifecycle_state in running_resources) and
(
                    resource.resource_type not in
resource_not_required):
                resources_details.append(resource)
                no_of_resources = no_of_resources + 1
                print("{}             {}          {}          {}
{}".format(resource.display_name,

         resource.resource_type,

         resource.lifecycle_state,

         resource.identifier,

         resource.time_created))
        print(

"============================================================================
====================================================================")
        print("Total number of resources {}".format(len(resources_details)))
        return resources_details

    # Removes all resources based on the service name prefix
    def cleanup_resources(self, delete_list):
        print("Deleting the resources")
        self.delete_all_autoscaling_resources(delete_list)
        self.delete_policies(delete_list)
        self.delete_instance(delete_list)
        self.delete_block_volumes(delete_list)
        self.delete_load_balancer(delete_list)
        self.delete_subnet(delete_list)
        self.delete_sec_list(delete_list)
        self.delete_route_table(delete_list)
        self.delete_dhcp_options(delete_list)
        self.delete_internet_gateway(delete_list)
        self.delete_service_gateway(delete_list)
        self.delete_local_peering_gateway(delete_list)
        self.delete_nat_gateway(delete_list)
        self.delete_vcn_resources(delete_list)
        self.delete_unified_agent_configuration(delete_list)
        self.delete_log(delete_list)
        self.delete_log_group(delete_list)
        self.delete_mount_targets(delete_list)
        self.delete_fss(delete_list)
        self.delete_tag_namespace(delete_list)
        self.delete_boot_volumes(delete_list)

    # Delete Policies
    def delete_policies(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "Policy":
                policy_ocid = resource.identifier
                print("Deleting policy: {0}, with ocid:
{1}".format(resource.display_name, policy_ocid))
                try:

self.identity_client_composite_operations.delete_policy_and_wait_for_state(
                        policy_ocid,

wait_for_states=[oci.identity.models.Policy.LIFECYCLE_STATE_DELETED])
```

```
                    print("Deleted policy successfully!")
                except Exception as e:
                    print("Error while deleting the policy {0}, policy
id {1}, Error message {2}".format(
                        resource.display_name, policy_ocid, str(e)))

    # Delete Dynamic Group
    def delete_dynamic_group(self):
        tenancy = os.environ['OCI_TENANCY']
        dynamic_group_list =
self.identity_client.list_dynamic_groups(tenancy).data
        for d_group in dynamic_group_list:
            if self.service_name_prefix in d_group.name:
                print("Deleting the dynamic group: {0}, with ocid:
{1}".format(d_group.name, d_group.id))
                try:

self.identity_client_composite_operations.delete_dynamic_group_and_wait
_for_state(
                        d_group.id,
wait_for_states=[oci.identity.models.DynamicGroup.LIFECYCLE_STATE_DELET
ED])
                    print("Deleted the dynamic group successfully!")
                except Exception as e:
                    print("Error while deleting the dynamic group name
{}, ocid {}, Error message {}".format(
                        d_group.name, d_group.id, str(e)))

    # Delete Block Volumes
    def delete_block_volumes(self, delete_list):
        bv_client = oci.core.BlockstorageClient(config=get_config(),
signer=self.signer)
        bv_composite_operations =
oci.core.BlockstorageClientCompositeOperations(bv_client)
        for resource in delete_list:
            if resource.resource_type == "Volume":
                bv_ocid = resource.identifier
                try:
                    print(
                        "Deleting the block volume: {0}, with ocid
{1}".format(resource.display_name, bv_ocid))

bv_composite_operations.delete_volume_and_wait_for_state(
                        bv_ocid,
wait_for_states=[oci.core.models.Volume.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted the block volume successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the block volume {0},
ocid {1}, Error message {2}".format(
                            resource.display_name, bv_ocid, str(e)))

    # Delete all compute instances
    def delete_instance(self, delete_list):
        compute_client = oci.core.ComputeClient(config=get_config(),
signer=self.signer)
        compute_composite_operations =
oci.core.ComputeClientCompositeOperations(compute_client)
        for resource in delete_list:
            if resource.resource_type == "Instance":
                instance_ocid = resource.identifier
                instance_name = resource.display_name
                print("Deleting the compute instance: {0}, with ocid
{1}".format(instance_name, instance_ocid))
                try:

compute_composite_operations.terminate_instance_and_wait_for_state(
                        instance_ocid,
wait_for_states=[oci.core.models.Instance.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted the compute instance successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the instance {0}, ocid {1},
Error message {2}".format(
                            instance_name, instance_ocid, str(e)))

    # Delete all Subnets in the VCN
    def delete_subnet(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "Subnet":
                subnet_ocid = resource.identifier
                print(
                    "Deleting subnet: {0}, with ocid
{1}".format(resource.display_name, resource.identifier))
                try:

self.virtual_network_composite_operations.delete_subnet_and_wait_for_state(
                        subnet_ocid,

wait_for_states=[oci.core.models.Subnet.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted subnet successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the subnet {0}, ocid {1},
Error message {2}".format(resource.display_name,

                     subnet_ocid, str(e)))

    # Delete Security lists
    def delete_sec_list(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "SecurityList":
                sec_list_name = resource.display_name
                sec_list_ocid = resource.identifier
                if not ("Default" in sec_list_name):
                    print(
                        "Deleting the security list: {0}, with ocid
{1}".format(resource.display_name,

   resource.identifier))
```

```
                    try:

self.virtual_network_composite_operations.delete_security_list_and_wait
_for_state(
                            sec_list_ocid,

wait_for_states=[oci.core.models.SecurityList.LIFECYCLE_STATE_TERMINATE
D])
                        print("Deleted the security list
successfully!")
                    except Exception as e:
                        print(
                            "Error while deleting the security list
{0}, ocid {1}, Error message {2}".format(
                                resource.display_name, sec_list_ocid,
str(e)))

    # Delete Load balancers
    def delete_load_balancer(self, delete_list):
        lb_client =
oci.load_balancer.LoadBalancerClient(config=get_config(),
signer=self.signer)
        lb_composite_operations =
oci.load_balancer.LoadBalancerClientCompositeOperations(lb_client)
        for resource in delete_list:
            if resource.resource_type == "LoadBalancer":
                lb_name = resource.display_name
                lb_ocid = resource.identifier
                print("Deleting Load balancer {0} with ocid
{1}".format(lb_name, lb_ocid))
                try:

lb_composite_operations.delete_load_balancer_and_wait_for_state(
                        lb_ocid,

wait_for_states=[oci.load_balancer.models.WorkRequest.LIFECYCLE_STATE_S
UCCEEDED])
                    print("Load balancer deleted successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the loadbalancer {0},
ocid {1}, Error message {2}".format(
                            lb_name, lb_ocid, str(e)))

    # Delete Route tables
    def delete_route_table(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "RouteTable":
                route_table_name = resource.display_name
                route_table_ocid = resource.identifier
                # Removing the route rules from the tables
                rt_details = oci.core.models.UpdateRouteTableDetails()
                rt_details.route_rules = []

self.virtual_network_composite_operations.update_route_table_and_wait_f
```

```
or_state(
                    route_table_ocid, rt_details,

wait_for_states=[oci.core.models.RouteTable.LIFECYCLE_STATE_AVAILABLE])
                self.vcn_client.update_route_table(route_table_ocid,
rt_details)
                # Default route table can't be deleted from VCN
                if not ("Default" in route_table_name):
                    print(
                        "Deleting the route table: {0}, with ocid
{1}".format(resource.display_name,

 resource.identifier))
                    try:

self.virtual_network_composite_operations.delete_route_table_and_wait_for_sta
te(
                            route_table_ocid,

wait_for_states=[oci.core.models.RouteTable.LIFECYCLE_STATE_TERMINATED])

                        print("Deleted the route table successfully!")
                    except Exception as e:
                        print("Error while deleting the route table {0},
ocid {1}, Error message {2}".format(
                            resource.display_name, route_table_ocid, str(e)))
                        if "associated with Subnet" in str(e):
                            try:

self.delete_subnet_route_table_association(route_table_ocid)
                                # After removing the association again
retrying the removal of route table
                                # This is for Db subnet route table

self.virtual_network_composite_operations.delete_route_table_and_wait_for_sta
te(
                                    route_table_ocid,

wait_for_states=[oci.core.models.RouteTable.LIFECYCLE_STATE_TERMINATED])
                                print("Deleted the route table
successfully!")
                            except Exception as e:
                                print("Error while deleting the route table
after removing the association "
                                      "{0}, ocid {1}, Error message
{2}".format
                                      (resource.display_name,
route_table_ocid, str(e)))

    # Delete Subnet and route table association to remove route table
    def delete_subnet_route_table_association(self, route_table_ocid):
        default_rt_id_in_vcn = ""
        print("Route table is associated with a subnet. Removing the
association between the subnet and route table")
        rt_res = self.vcn_client.get_route_table(route_table_ocid).data
        vcn_id = rt_res.vcn_id
        compartment_id = rt_res.compartment_id
        list_route_rables_vcn =
self.vcn_client.list_route_tables(compartment_id=compartment_id,

vcn_id=vcn_id).data
        for rt in list_route_rables_vcn:
            if "Default Route" in rt.display_name:
                default_rt_id_in_vcn = rt.id
        list_subnets =
self.vcn_client.list_subnets(compartment_id=compartment_id,
vcn_id=vcn_id).data
        for subnet in list_subnets:
            subnet_ocid = subnet.id
            if subnet.route_table_id == route_table_ocid:
                subnet_details = oci.core.models.UpdateSubnetDetails()
                subnet_details.route_table_id = default_rt_id_in_vcn
                try:

self.virtual_network_composite_operations.update_subnet_and_wait_for_st
ate(
                        subnet_ocid, subnet_details,

wait_for_states=[oci.core.models.Subnet.LIFECYCLE_STATE_AVAILABLE])
                    print("Removed the association between the subnet
and route table.")
                except Exception as e:
                    print("Error while removing the association
between the subnet and route table {}".format(str(e)))

    # Delete DHCP Options
    def delete_dhcp_options(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "DHCPOptions":
                dhcp_name = resource.display_name
                dhcp_ocid = resource.identifier
                if not ("Default" in dhcp_name):
                    print(
                        "Deleting the DHCP options: {0}, with ocid
{1}".format(resource.display_name, dhcp_ocid))
                    try:

self.virtual_network_composite_operations.delete_dhcp_options_and_wait_
for_state(dhcp_ocid,

                                  wait_for_states=[


oci.core.models.DhcpOptions.LIFECYCLE_STATE_TERMINATED])
                        print("Deleted the DHCP options successfully!")
                    except Exception as e:
                        print(
                            "Error while deleting the DHCP options
{0}, ocid {1}, Error message {2} ".format(
                                resource.display_name, dhcp_ocid,
str(e)))

    # Delete Internet Gateway
    def delete_internet_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "InternetGateway":
                ig_ocid = resource.identifier
                print("Deleting the Internet Gateway: {0}, with ocid 
{1}".format(resource.display_name,

    ig_ocid))
                try:

self.virtual_network_composite_operations.delete_internet_gateway_and_wait_fo
r_state(ig_ocid,

                            wait_for_states=[


oci.core.models.InternetGateway.LIFECYCLE_STATE_TERMINATED])

                    print("Deleted the Internet Gateway successfully!")
                except Exception as e:
                    print("Error while deleting the Internet Gateway {0}, 
ocid {1}, Error message {2}".format(
                        resource.display_name,
                        ig_ocid, str(e)))

    # Delete Service Gateway
    def delete_service_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "ServiceGateway":
                svc_gateway_ocid = resource.identifier
                print("Deleting the service gateway: {0}, with ocid 
{1}".format(resource.display_name,

   svc_gateway_ocid))
                try:

self.virtual_network_composite_operations.delete_service_gateway_and_wait_for
_state(
                        svc_gateway_ocid,
wait_for_states=[oci.core.models.ServiceGateway.LIFECYCLE_STATE_TERMINATED])

                    print("Deleted the service gateway successfully!")
                except Exception as e:
                    print("Error while deleting the service gateway {0}, 
ocid {1}, Error message {2}".format(
                        resource.display_name,
                        svc_gateway_ocid, str(e)))

    # Delete Local Peering Gateway
    def delete_local_peering_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "LocalPeeringGateway":
                lpg_ocid = resource.identifier
                print("Deleting the local peering gateway: {0}, with
ocid {1}".format(resource.display_name,

               lpg_ocid))
                try:

self.virtual_network_composite_operations.delete_local_peering_gateway_
and_wait_for_state(
                        lpg_ocid,
wait_for_states=[oci.core.models.LocalPeeringGateway.LIFECYCLE_STATE_TE
RMINATED])

                    print("Deleted local peering gateway
successfully!")
                except Exception as e:
                    print("Error while deleting the local peering
gateway {0}, ocid {1}, Error message {2}".format(
                        resource.display_name, lpg_ocid, str(e)))

    # Delete Nat Gateway
    def delete_nat_gateway(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "NatGateway":
                nat_ocid = resource.identifier
                print("Deleting the NAT gateway: {0}, with ocid
{1}".format(resource.display_name,

     nat_ocid))
                try:

self.virtual_network_composite_operations.delete_nat_gateway_and_wait_f
or_state(
                        nat_gateway_id=nat_ocid,

wait_for_states=[oci.core.models.NatGateway.LIFECYCLE_STATE_TERMINATED]
                    )
                    print("Deleted the NAT gateway successfully!")
                except Exception as e:
                    print("Error while deleting the NAT gateway {0},
ocid {1}, Error message {2}".format(
                        resource.display_name, nat_ocid, str(e)))

    # Delete VCN
    def delete_vcn_resources(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "Vcn":
                vcn_ocid = resource.identifier
                vcn_name = resource.display_name
                print("Deleting the VCN: {0}, with ocid
{1}".format(vcn_name, vcn_ocid))
                try:

self.virtual_network_composite_operations.delete_vcn_and_wait_for_stat
e(vcn_ocid,
               oci.core.models.Vcn.LIFECYCLE_STATE_TERMINATED)
                    print("Deleted the VCN successfully!")
                except Exception as e:
                    print("Error while deleting the VCN {0}, VCN id {1},
Error message {2}".format(vcn_name, vcn_ocid,

                      str(e)))

    # Deleting the Unified Agent Configuration
    def delete_unified_agent_configuration(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "UnifiedAgentConfiguration":
                uac_ocid = resource.identifier
                print("Deleting the unified agent configuration: {}, with
ocid {}".format(resource.display_name,

             uac_ocid))
                try:

self.log_composite_operations.delete_unified_agent_configuration_and_wait_for
_state(
                        uac_ocid,

wait_for_states=[oci.logging.models.WorkRequest.STATUS_SUCCEEDED])
                    print("Deleted the unified agent configuration
successfully!")
                except Exception as e:
                    print(
                        "Error while deleting the unified agent
configuration name {0}, ocid {1} - Error message {2}".format(
                            resource.display_name, uac_ocid, str(e)))

    # Delete logs in a Log groups
    def delete_log(self, delete_list, name=None):
        """
        Delete log resources for the service.

        :param delete_list:
        :param name: if name is set, only delete the named log resource,
otherwise delete all log resources in the delete_list
        :return:
        """
        for resource in delete_list:
            if resource.resource_type == "LogGroup":
                log_group_ocid = resource.identifier
                list_logs = self.log_client.list_logs(log_group_ocid).data
                for log in list_logs:
                    to_delete = True

                    if name is not None:
                        if log.display_name != name:
                            to_delete = False
                    if to_delete:
                        print("Deleting the log name {0}, with log ocid
{1}".format(log.display_name, log.id))
                        try:

self.log_composite_operations.delete_log_and_wait_for_state(
                                log_group_ocid, log.id,

wait_for_states=[oci.logging.models.WorkRequest.STATUS_SUCCEEDED])
                            print("Deleted the log {}
successfully!".format(log.display_name))

                        except Exception as e:
                            print("Error while deleting the log name
{}, log ocid {}, Error message {}".format(
                                log.display_name, log.id, str(e)))

    # Delete Log Group
    def delete_log_group(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "LogGroup":
                log_group_ocid = resource.identifier
                print("Deleting the log group: {0}, with ocid
{1}".format(resource.display_name,

   log_group_ocid))
                try:

self.log_composite_operations.delete_log_group_and_wait_for_state(
                        log_group_ocid,
wait_for_states=[oci.logging.models.WorkRequest.STATUS_SUCCEEDED])

                    print("Deleted log group successfully!")
                except Exception as e:
                    print("Error while deleting the log group {0},
ocid {1}, Error message {2}".format(
                        resource.display_name, log_group_ocid, str(e)))

    # Delete the Mount targets
    def delete_mount_targets(self, delete_list):
        mt_client =
oci.file_storage.FileStorageClient(config=get_config(),
signer=self.signer)
        mt_composite_operations =
oci.file_storage.FileStorageClientCompositeOperations(mt_client)
        for resource in delete_list:
            if resource.resource_type == "MountTarget":
                mt_ocid = resource.identifier
                print("Deleting the mount target {0}, with ocid
{1}".format(resource.display_name, mt_ocid))
                try:

mt_composite_operations.delete_mount_target_and_wait_for_state(
                        mt_ocid,
wait_for_states=[oci.file_storage.models.MountTarget.LIFECYCLE_STATE_DE
LETED])
                    print("Deleted the mount target successfully!")
                except Exception as e:
                    print("Error while deleting the mount target {0}, ocid
{1}, Error message {2}".format(
                        resource.display_name, mt_ocid, str(e)))

    # Delete FSS
    def delete_fss(self, delete_list):
        fss_client = oci.file_storage.FileStorageClient(config=get_config(),
signer=self.signer)
        fss_composite_operations =
oci.file_storage.FileStorageClientCompositeOperations(fss_client)
        for resource in delete_list:
            if resource.resource_type == "FileSystem":
                fss_ocid = resource.identifier
                try:
                    # Get the list of exports to delete
                    list_exports =
fss_client.list_exports(file_system_id=fss_ocid).data
                    for export in list_exports:
                        export_ocid = export.id
                        print("Deleting the export id
{}".format(export_ocid))

fss_composite_operations.delete_export_and_wait_for_state(
                            export_id=export_ocid,

wait_for_states=[oci.file_storage.models.Export.LIFECYCLE_STATE_DELETED])
                        print("Deleted the exports successfully!")
                except Exception as e:
                    print("Error while deleting the export, Error message
{}".format(str(e)))
                try:
                    print("Deleting the FSS: {0}, with ocid
{1}".format(resource.display_name, fss_ocid))

fss_composite_operations.delete_file_system_and_wait_for_state(
                        fss_ocid,
wait_for_states=[oci.file_storage.models.FileSystem.LIFECYCLE_STATE_DELETED])
                    print("Deleted the FSS successfully!")
                except Exception as e:
                    print("Error while deleting the FSS name {0}, ocid {1},
Error message {2}".format(
                        resource.display_name, fss_ocid, str(e)))

    # Deletion of TagNamespace
    def delete_tag_namespace(self, delete_list):
        for resource in delete_list:
            if resource.resource_type == "TagNamespace":
                tag_ns_name = resource.display_name
                tag_ns_ocid = resource.identifier
                print("Deleting the tag namespace {0}, with ocid
{1}".format(tag_ns_name, tag_ns_ocid))
                try:
                    # Retiring the tag namespace
                    tag_status =
self.identity_client.get_tag_namespace(tag_namespace_id=tag_ns_ocid).da
ta
                    print("Tag namespace: {} and isRetired:
{}".format(tag_ns_name, tag_status.is_retired))

                    if not tag_status.is_retired:
                        print("Retiring the tag namespace
{}".format(tag_ns_name))
                        tag_ns_details =
oci.identity.models.UpdateTagNamespaceDetails()
                        tag_ns_details.is_retired = True

self.identity_client_composite_operations.update_tag_namespace_and_wait
_for_state(
                            tag_namespace_id=tag_ns_ocid,

update_tag_namespace_details=tag_ns_details,
                            wait_for_states=[

oci.identity.models.TagNamespace.LIFECYCLE_STATE_INACTIVE])
                        tag_status =
self.identity_client.get_tag_namespace(tag_namespace_id=tag_ns_ocid).da
ta
                        print("Tag status before deleting
{}".format(tag_status.is_retired))
                    print("Deleting the tag namespace
{}".format(tag_ns_name))
                    # Tag namespace deletion is taking too long time.
So not waiting for the completion.

self.identity_client.cascade_delete_tag_namespace(tag_namespace_id=tag_
ns_ocid)
                    print("Asynchronous deletion of Tag namespaces is
enabled."
                          "Check the deletion status manually. Tag
name {0} with ocid {1}".format(tag_ns_name,

                           tag_ns_ocid))
                except Exception as e:
                    print("Error while deleting the Tag namespace {0},
ocid {1}, Error message {2} "
                          .format(tag_ns_name, tag_ns_ocid, str(e)))

    # Deleting the unattached boot volumes
    def delete_boot_volumes(self, delete_list):
        bv_client = oci.core.BlockstorageClient(config=get_config(),
signer=self.signer)
        bv_composite_operations =
oci.core.BlockstorageClientCompositeOperations(bv_client)
        for resource in delete_list:
            if resource.resource_type == "BootVolume" and
resource.lifecycle_state == "AVAILABLE":
                bv_ocid = resource.identifier
                bv_name = resource.display_name
                print("Deleting the boot volume {}, with ocid {}
".format(bv_name, bv_ocid))
                try:

bv_composite_operations.delete_boot_volume_and_wait_for_state(
                        boot_volume_id=bv_ocid,

wait_for_states=[oci.core.models.BootVolume.LIFECYCLE_STATE_TERMINATED])
                    print("Deleted the boot volume successfully!")
                except Exception as e:
                    print("Error while deleting the boot volume name {},
ocid {}, Error message {}".format(bv_name,

                              bv_ocid,

                              str(e)))

    def delete_functions(self, delete_list):
        """
        Deletes OCI functions and function application for the service.

        :param delete_list:
        :return:
        """
        for resource in delete_list:
            if resource.resource_type == "FunctionsApplication":
                fn_app_id = resource.identifier
                fn_app_name = resource.display_name
                try:
                    result = pagination.list_call_get_all_results(
                        self.fn_client.list_functions,
                        fn_app_id
                    )

                    # Delete all functions within function application
                    print("Deleting functions within function application
{}".format(fn_app_name))
                    for fn in result.data:

self.fn_composite_operations.delete_function_and_wait_for_state(
                            function_id=fn.id,

wait_for_states=[fn_models.Function.LIFECYCLE_STATE_DELETED]
                        )
                        print("Deleted the function {}
successfully!".format(fn.display_name))

                except Exception as e:
                    print("Error while deleting the functions within
application id {}, Error message {}".format(fn_app_id, str(e)))


    def delete_functions_app(self, delete_list):
        """
        Deletes function application for the service.
        :param delete_list:
        :return:
        """
        for resource in delete_list:
            if resource.resource_type == "FunctionsApplication":
                fn_app_id = resource.identifier
                try:
                    # Delete the function application

self.fn_composite_operations.delete_application_and_wait_for_state(
                        application_id = fn_app_id,
                        wait_for_states =
[fn_models.Application.LIFECYCLE_STATE_DELETED]
                    )
                except Exception as e:
                    print("Error while deleting the Functions
application with id {}, Error message {}".format(fn_app_id, str(e)))

    def delete_event_rules(self, delete_list):
        """
        Deletes the event rules for the service.
        :param delete_list:
        :return:
        """
        for resource in delete_list:
            if resource.resource_type == "EventRule":
                event_rule_id = resource.identifier
                event_rule_name = resource.display_name
                try:
                    print("Deleting event rule
{}".format(event_rule_name))

self.events_client_composite_operations.delete_rule_and_wait_for_state(
                        rule_id=event_rule_id,

wait_for_states=[event_models.Rule.LIFECYCLE_STATE_DELETED]
                    )
                    print("Deleted the event rule {}
successfully!".format(event_rule_name))
                except Exception as e:
                    print("Error while deleting the event rule id {},
Error message {}".format(event_rule_id, str(e)))


    def delete_autoscaling_logs(self, delete_list):
        """
        Deletes the event rule invoke log for the service.

        :param delete_list:
        :return:
        """
        # trim the extra hyphen char if present from the
service_name_prefix
        service_name = self.service_name_prefix[:-1] if
self.service_name_prefix[
                                                               -1] == '-' else
self.service_name_prefix
        self.delete_log(delete_list,
name="{0}_event_rule_invoke_log".format(service_name))
        self.delete_log(delete_list,
name="{0}_autoscaling_log".format(service_name))

    def predestroy_autoscaling_resources(self, delete_list):
        """
        Deletes pre-destroy resources associated with autoscaling.
        This is to be invoked when user needs to delete only the resources
created via API during provisioning outside
        of terraform preferably prior to running terraform destroy action.

        :param delete_list:
        :return:
        """
        print("Executing pre-destroy of resources created for autoscaling
feature")
        self.delete_functions(delete_list)
        self.delete_event_rules(delete_list)
        self.delete_autoscaling_logs(delete_list)

    def delete_alarms(self, delete_list):
        """
        Delete all alarms created for autoscaling for the service.

        :param delete_list:
        :return:
        """
        for resource in delete_list:
            if resource.resource_type == "Alarm":
                alarm_id = resource.identifier
                alarm_name = resource.display_name
                try:
                    print("Deleting alarm {}".format(alarm_name))

self.monitoring_composite_operations.delete_alarm_and_wait_for_state(
                        alarm_id=alarm_id,

wait_for_states=[monitoring_models.Alarm.LIFECYCLE_STATE_DELETED]
                    )
                    print("Deleted the alarm {}
successfully!".format(alarm_name))
                except Exception as e:
                    print("Error while deleting the alarm id {}, Error
message {}".format(alarm_id, str(e)))

    def delete_container_repos(self, delete_list):
        """
        Deletes container repos created for autoscaling for the service.
        Deleting the repos also deletes the container images in those repos.

        :param delete_list:
        :return:
```

```
        """
        for resource in delete_list:
            if resource.resource_type == "ContainerRepo":
                repo_id = resource.identifier
                repo_name = resource.display_name
                try:
                    print("Deleting container repo
{}".format(repo_name))

self.artifacts_composite_operations.delete_container_repository_and_wai
t_for_state(
                        repository_id=repo_id,

wait_for_states=[artifacts_models.ContainerRepository.LIFECYCLE_STATE_D
ELETED]
                    )
                    print("Deleted the container repo {}
successfully!".format(repo_name))
                except Exception as e:
                    print("Error while deleting the container repo id
{}, Error message {}".format(repo_id, str(e)))

    def delete_notification_topics(self, delete_list):
        """
        Deletes the notification topics created for autoscaling for
the service.
        Deleting the notification topics also deletes the associated
subscriptions for those topics.

        :param delete_list:
        :return:
        """
        for resource in delete_list:
            if resource.resource_type == "OnsTopic":
                topic_id = resource.identifier
                topic_name = resource.display_name
                try:
                    print("Deleting notification topic
{}".format(topic_name))
                    self.ons_control_plane_client.delete_topic(
                        topic_id=topic_id
                    )
                    oci.wait_until(self.ons_control_plane_client,
self.ons_control_plane_client.get_topic(topic_id),
                                   'lifecycle_state',
ons_models.NotificationTopic.LIFECYCLE_STATE_DELETING)
                    print("Deleted the notification topic {}
successfully!".format(topic_name))
                except Exception as e:
                    print(
                        "Error while deleting the notification topic
id {}, Error message {}".format(topic_id, str(e)))

    def delete_all_autoscaling_resources(self, delete_list):
        """
        Deletes all resources created when autoscaling feature is enabled.

        :param delete_list:
        :return:
        """
        self.predestroy_autoscaling_resources(delete_list)
        self.delete_container_repos(delete_list)
        self.delete_alarms(delete_list)
        self.delete_notification_topics(delete_list)
        self.delete_functions_app(delete_list)


def main():
    parser = argparse.ArgumentParser(description=textwrap.dedent('''
        This script is used for:
        - pre-destroying autoscaling resources (prior to running destroy job
on the stack from OCI Console)
        - delete all infra resources created for the stack (identified by
its service name prefix)
    '''), formatter_class=argparse.RawDescriptionHelpFormatter)
    # Required position params
    parser.add_argument('command', choices=['delete', 'list', 'pre-
destroy'], help='Command')
    parser.add_argument('service_name_prefix', help='Stack service name')

    # Optional params
    parser.add_argument('-r', '--region', default='', help='Region other
than home region must be specified.')
    parser.add_argument('-f', '--feature', default='autoscaling',
help='Feature to run pre-destroy for.')

    args = parser.parse_args()

    command = args.command
    service_prefix = args.service_name_prefix

    global REGION

    REGION = args.region
    feature = args.feature

    print("Service prefix name:" + service_prefix)

    if len(service_prefix) >= 16:
        service_prefix = service_prefix[0:16]
    service_prefix = service_prefix + "-"
    cleanup_util = CleanUpResources(service_prefix)

    if command == 'list':
        print("Listing all resources with service prefix name " +
service_prefix)
        cleanup_resources = cleanup_util.list_all_resources()
    elif command == 'delete':
        print("Deleting all resources with service prefix name " +
service_prefix)
```

```
        cleanup_resources = cleanup_util.list_all_resources()
        cleanup_util.cleanup_resources(cleanup_resources)
        cleanup_util.delete_dynamic_group()
    elif command == 'pre-destroy' and feature == 'autoscaling':
        print('Deleting pre-destroy autoscaling resources with service 
prefix name ' + service_prefix)
        cleanup_resources = cleanup_util.list_all_resources()

cleanup_util.predestroy_autoscaling_resources(cleanup_resources)


if __name__ == '__main__':
    main()
```

D

Migrate Terraform Scripts

You can access the Terraform Scripts of an Oracle WebLogic Server for OCI and modify it as required. See Terraform Scripts in Oracle WebLogic Server for OCI.

Support ended for Terraform v0.11.x. So, you would not be able to scale out or destroy the stacks created with Terraform v0.11.x. However, you can configure the stacks in your local environment by downloading the terraform configuration, variables, and terraform state file from your Oracle Resource Manager (ORM) stacks. You can continue using terraform v0.11.x with the downloaded scripts and manage their stacks outside ORM.

Complete the following steps:

1. Download the previous versions of the terraform from https://releases.hashicorp.com/terraform/.
2. Install the terraform on the host, where you want to run the scripts.
3. Download the configuration and terraform state file from the last apply job:
   a. Click the navigation menu ☰, and select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
   b. Select the **Compartment** that contains your stack.
   c. Click the name of your stack.
   d. In the **Jobs** section, select the latest apply job.
   e. Click **Download Terraform Configuration** and save the file to the preferred location.
   f. Click **Download Terraform State** and save the file to the preferred location
4. Extract variables from the stack.
   Run the following OCI CLI command to generate the variables in json format:

   ```
   #ensure that oci cli and jq are installed in cloud shell.
   <user>@cloudshell:~ (<domain>-1)$ oci resource-manager stack get --stack-
   id <OCID> | jq -r .data.variables
   ```

5. Copy the variables into the `.json` file.
   You can edit the file to reapply parameters, like, node count, VM shape, SSH keys, and so on. For information about the variables, see Variables in Terraform Scripts.
6. Unzip the terraform configuration files to the working folder (`~/wlsoci/`).
7. Copy the terraform state and the varaibles.json fles to the working folder (`~/wlsoci/`).
8. As required, by using reapply, scale out or destroy the stacks. See Invoke Terraform Scripts.

E

Create a Domain for Oracle WebLogic Server 12.2.1.3.0 Using Terraform

Create an Oracle WebLogic Server domain for Oracle WebLogic Server 12.2.1.3.0 using the terraform configuration from the 12.2.1.4.0 Oracle WebLogic Server for OCI stack.

> **Note:**
>
> Ensure that you have created the 12.2.1.4.0 Oracle WebLogic Server for OCI stack.

Complete the following steps:

1. Download the terraform configuration file for the 12.2.1.4 stack. See Download a Terraform Configuration File.

   > **Note:**
   >
   > As you have already created the stack, you can follow steps 2 to 10 in **Download a Terraform Configuration File**.

2. In the `variables.tf` terraform configuration file, change the `wls_version` to *12.2.1.3.0*.
3. Complete step 1 through step 4 in Invoke Terraform Scripts.

   > **Note:**
   >
   > Use `env_vars_template` to create `env_vars` and source it as: source `./env_vars` before running `terraform init`.