Oracle® Cloud

Using Oracle WebLogic Server for OKE (Release 21.3.2 or earlier)

F47617-02
April 2022

Oracle Cloud Using Oracle WebLogic Server for OKE (Release 21.3.2 or earlier),

F47617-02

Primary Author: Oracle Corporation

Contents

1 Get Started with Oracle WebLogic Server for OKE

2 Create a Domain with Oracle WebLogic Server for OKE

3 Update a Domain in Oracle WebLogic Server for OKE

4 Manage a Domain in Oracle WebLogic Server for OKE

5 Troubleshoot and Known Issues in Oracle WebLogic Server for OKE

1

Get Started with Oracle WebLogic Server for OKE

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

> **Note:**
>
> If you are using Oracle WebLogic Server for OKE (**Release 21.3.3 or later**), see Using Oracle WebLogic Server for OKE.

Topics:

About Oracle WebLogic Server for OKE

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

Oracle WebLogic Server for OKE is available as a set of applications in the Oracle Cloud Infrastructure Marketplace. After launching one of these applications, you use a simple wizard interface to configure and provision your domains along with any supporting cloud resources like Kubernetes clusters, file systems, compute instances, networks and load balancers. Each server in the domain runs in a separate container in the Kubernetes cluster.

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

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

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

- Supports the Oracle Application Development Framework (ADF)

- Connects to an existing database in Oracle Cloud Infrastructure

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

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

- Oracle WebLogic Server Enterprise Edition
- Oracle WebLogic Suite

Oracle WebLogic Server for OKE supports these billing options:

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

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

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

Oracle WebLogic Server for OKE requires Oracle Cloud Infrastructure Vault in order to securely store the passwords for your domains. See Oracle Cloud Infrastructure Vault FAQ.

WebLogic Domain Home Source Types

When using the WLS operator to start WebLogic Server instances from a domain, you can use either the Model in Image and Domain in Image source types.

To identify whether a version uses the Model in Image or the Domain in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

> **Important:**
>
> From Oracle WebLogic Server for OKE release 21.2.2 onwards, Domain in Image is deprecated.

The following tables highlights the difference between the domain home source types.

Table 1-1 Domain Home Source Types

Model in Image	Domain in Image
The WDT model and archive files must be embedded in docker image. The kubernetes secrets can be passed for macro references within the model in the docker image. The WLS Operator's domain introspector job uses the WDT artifacts in the docker image to create the domain, and then starts the domain's WebLogic pods.	All the files that are required to run your domain in Kubernetes (binaries, patches, configuration, applications, and so on) are stored in the Docker image for your domain.
Different domains can use the same image, but require different domainUID and may have different configuration.	Requires a different image for each domain, but all servers in that domain use the same image.
Runtime state should not be kept in the images. Application and configuration may be.	Runtime state should not be kept in the images, but applications and configuration are.
If you want to mutate the domain home configuration, then you can override it with additional model files supplied in a ConfigMap or you can supply a new image. If you want to deploy application updates, then you must create a new image.	If you want to mutate the domain home configuration, then you can apply configuration overrides or create a new image. If you want to deploy application updates, then you must create a new image.
You can deploy model files to a ConfigMap to mutate the domain, and may not need to restart the entire domain for the change to take effect. Instead, you can initiate a rolling upgrade, which restarts your WebLogic Server instance Pods one at a time. Also, the model file syntax is far simpler and less error prone than the configuration override syntax, and, unlike configuration overrides, allows you to directly add data sources and JMS modules.	You can use configuration overrides to mutate the domain configuration, but there are limitations.
Dynamic updates to domain configuration can be leveraged through `ConfigMap`.	Not supported.

About the Components of Oracle WebLogic Server for OKE

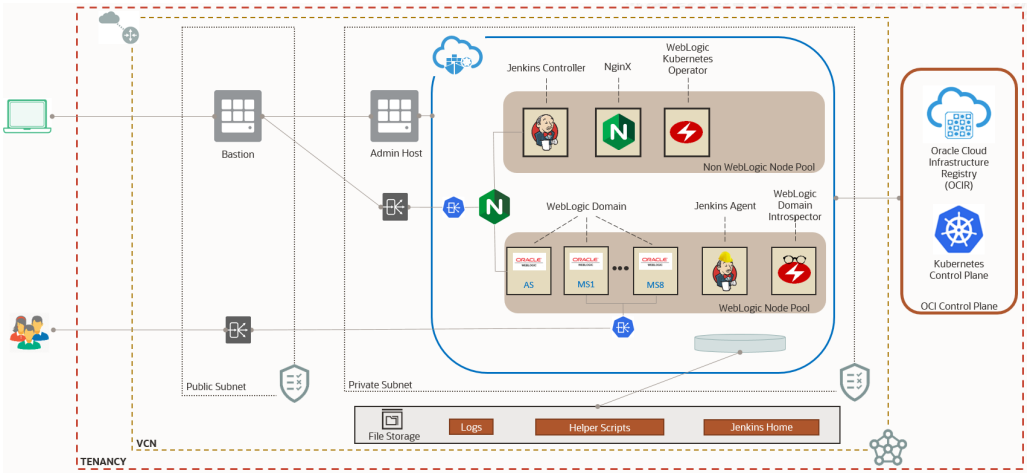
Learn about the Oracle Cloud Infrastructure components that comprise Oracle WebLogic Server for OKE.

Topics

- Oracle WebLogic Server
- Marketplace
- Resource Manager
- Container Engine for Kubernetes
- Registry
- WebLogic Server Kubernetes Operator
- Helm
- Jenkins
- Compute

- Storage
- Virtual Cloud Network
- Load Balancer
- Database
- Vault

The following diagram illustrates the components of a typical Oracle WebLogic Server for OKE 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 OKE supports these Oracle WebLogic Server editions:

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

Oracle WebLogic Server for OKE can create these domain configurations:

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

A JRF-enabled domain supports the Oracle Application Development Framework (ADF)

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

Marketplace

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

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

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

See Overview of Marketplace in the Oracle Cloud Infrastructure documentation.

Resource Manager

Oracle WebLogic Server for OKE uses Resource Manager in Oracle Cloud Infrastructure to provision the Kubernetes cluster, networks and other cloud resources that support your Oracle WebLogic Server 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 OKE is comprised of these components:

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

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

Container Engine for Kubernetes

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

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

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

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

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

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

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

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

When you scale a WebLogic cluster:

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

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

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

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

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

Registry

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

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

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

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

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

See Overview of Registry in the Oracle Cloud Infrastructure documentation.

WebLogic Server Kubernetes Operator

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

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

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

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

See Oracle WebLogic Server Kubernetes Operator.

Helm

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

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

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

See the Helm Documentation.

Jenkins

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

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

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

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

See Jenkins User Documentation.

Compute

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

A domain is comprised of these compute instances:

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

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

When you create a domain, you assign a shape to each of the compute instances. The shape determines the number of CPUs and the amount of memory allocated to the compute instance. Oracle Cloud Infrastructure offers a variety of bare metal (BM) and virtual machine (VM) shapes. However, Oracle WebLogic Server for OKE only supports the `VM.Standard2.x`, `VM.Standard.E2.x`, `BM.Standard2.x`, and `BM.Standard.E2.x` shapes. 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.

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

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

Storage

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

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

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

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

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

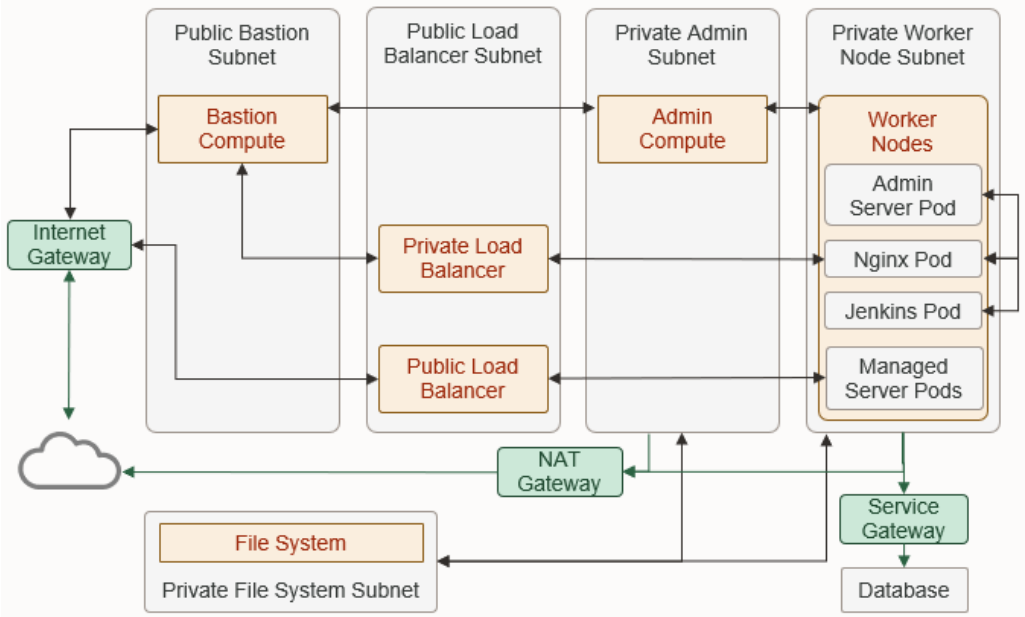
See Overview of File Storage in the Oracle Cloud Infrastructure documentation.

Virtual Cloud Network

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

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

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



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

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

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

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

See Overview of Networking in the Oracle Cloud Infrastructure documentation.

Load Balancer

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

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

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

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

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

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

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

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

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

See the following topics in the Oracle Cloud Infrastructure documentation:

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

Database

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

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

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

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

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

> **Note:**
>
> Free-Tier autonomous database is not supported.

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

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

Oracle WebLogic Server for OKE supports the same database versions and drivers as those for on-premise WebLogic Server installations. See *System Requirements and Supported Platforms for Oracle Fusion Middleware 12c (12.2.1.4.0)* at Oracle Fusion Middleware Supported System Configurations.

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

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

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

See these topics in the Oracle Cloud Infrastructure documentation:

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

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

Vault

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

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

Parameters for a new domain include:

- The password for the default Oracle WebLogic Server administrator
- The administrator password for an existing database, if you are creating a domain that includes the Java Required Files (JRF) components

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

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

See these topics in the Oracle Cloud Infrastructure documentation:

- Overview of Vault
- Oracle Cloud Infrastructure Vault FAQ

Identity

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

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

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

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

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

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

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

About the Application Lifecycle with Oracle WebLogic Server for OKE

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

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

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

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

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

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

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

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

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

About Oracle WebLogic Server for OKE Versions and Retirement Policy

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

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

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

For example:

```
WLS 12.2.1.4.200714.01
```

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 2

The 5th decimal place is a quarterly patch set release date. For example, `200714` is the July 2019 patch set.

For patch set update (PSU) naming purposes, releases in a Oracle WebLogic Server for OKE quarterly patch set 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 Oracle WebLogic Server for OKE scripts placed on the VM images, the Terraform scripts, and the Resource Manager schema.

A Oracle WebLogic Server for OKE 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 OKE 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 Included in Oracle WebLogic Server for OKE.

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

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:

```
12.2.1.4.200714.01
12.2.1.4.200714.02
12.2.1.4.190716.01
12.2.1.4.190716.02
```

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 OKE versions and images:

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

Before You Begin with Oracle WebLogic Server for OKE

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

Some tasks are required for any type of Oracle WebLogic Server domain that you create with Oracle WebLogic Server for OKE. 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 Compartment Policies
- Create Root Policies
- Create an Auth Token
- Create an Encryption Key
- Create Secrets with Passwords
- Create an SSH Key

Optional Tasks

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

Understand Service Requirements

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

- Identity and Access Management (dynamic groups and policies)

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

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

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

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

Create a Compartment

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

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

See Managing Compartments in the Oracle Cloud Infrastructure documentation.

Create Compartment Policies

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

Access to Oracle Cloud Infrastructure resources in a compartment is controlled through policies. Your Oracle Cloud Infrastructure user must have management access for Marketplace applications, Resource Manager stacks and jobs, Kubernetes clusters, compute instances, file systems, block storage volumes, load balancers, Key Management vaults and

keys, and IAM policies. If you want Oracle WebLogic Server for OKE to create network resources for a domain, then you must also have management access for these network resources.

A sample policy is shown below:

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

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

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

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

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

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

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

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

See Common Policies in the Oracle Cloud Infrastructure documentation.

Create Root Policies

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

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

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

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

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

See these topics in the Oracle Cloud Infrastructure documentation:

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

Create Dynamic Groups and Policies

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

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

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

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

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

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

Create a Dynamic Group

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

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

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

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

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

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

See Managing Dynamic Groups in the Oracle Cloud Infrastructure documentation.

Create Policies for the Dynamic Group

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

When you create a domain, compute instances in Oracle WebLogic Server for OKE need to access Oracle Cloud Infrastructure Vault secrets. If a load balancer is enabled, access to network resources is required. 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 network and database permissions to a dynamic group:

```
Allow dynamic-group MyInstancesPrincipalGroup to manage all-resources
in compartment MyCompartment
```

```
Allow service oke to read app-catalog-listing in compartment MyCompartment
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
compartment VaultCompartment where target.secret.id = '<OCID for weblogic
admin password secret>'
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
compartment VaultCompartment where target.secret.id = '<OCID for OCIR token
secret>'
Allow dynamic-group MyInstancesPrincipalGroup to read secret-bundles in
compartment VaultCompartment where target.secret.id = '<OCID for Database
password secret>'
Allow dynamic-group MyInstancesPrincipalGroup to use autonomous-transaction-
processing-family in compartment ATP_Database_Compartment
```

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

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

See these topics in the Oracle Cloud Infrastructure documentation:

- Common Policies
- Writing Policies for Dynamic Groups

Create an Auth Token

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

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

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

See Managing User Credentials in the Oracle Cloud Infrastructure documentation.

Create an Encryption Key

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

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

Create Secrets with Passwords

Use Oracle Cloud Infrastructure Vault to create secrets with passwords that you need to create a domain with Oracle WebLogic Server for OKE. Create one secret for each password.

You must provide OCID of the secrets:

- Administrator password for the new domain
- Auth token for a user with access to Oracle Cloud Infrastructure Registry
- Administrator password for an existing database, if you are creating a domain that includes the Java Required Files (JRF) components

To create secrets:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Identity & Security**, and then click **Vault**.
3. Select the **Compartment** where you want to create a secret.
4. From the list of vaults in the compartment, do one of the following:
   - Click the name of the vault where you want to create a secret.
   - Create a new vault for the secret by following the instructions in To create a new vault, and then click the name of the vault.
5. Click **Secrets**, and then click **Create Secret**.
6. In the **Create Secret** dialog box, choose a compartment from the **Create in Compartment** list.

   > **Tip:**
   >
   > Secrets can exist outside the compartment the vault is in.

7. Click **Name**, and then enter a name to identify the secret. Avoid entering any confidential information in this field
8. Click **Description**, and then enter a brief description of the secret to help identify it. Avoid entering any confidential information in this field.
9. Choose the encryption key that you want to use to encrypt the secret contents while they are imported to the vault.
10. Specify the format of the secret contents you are providing by choosing a template type from the **Secret Type Template** list.
11. Click **Secret Contents**, and then enter the secret contents.
12. Click **Create Secret**.

For information about optional fields, 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 OKE, you specify the public key. You then access the compute instances from an SSH client using the private key.

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

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

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

Create a Virtual Cloud Network

Oracle WebLogic Server for OKE can create a Virtual Cloud Network (VCN) in Oracle Cloud Infrastructure for a new Oracle WebLogic Server 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 resources assigned to a private subnet cannot be directly accessed from outside of Oracle Cloud. We recommend that you use private subnets for the Kubernetes cluster, administration compute instance, and file system.

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

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

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

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

See these topics in the Oracle Cloud Infrastructure documentation:

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

Create a Subnet for the Kubernetes Cluster

Oracle WebLogic Server for OKE can create a subnet for the Kubernetes cluster that hosts your Oracle WebLogic Server 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 OKE, the worker nodes in the Kubernetes cluster are assigned to a subnet. We recommend that you use a private subnet for the Kubernetes cluster.

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

If you want to use an existing subnet for the Kubernetes cluster when creating a 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) from the subnet that you plan to use for the administration compute instance.
- The subnet must have a security list that enables inbound access to all ports from the subnet that you plan to use for the load balancer.
- The subnet must have a security list that enables outbound access to the NFS port (2049) on the file system subnet.
- If you 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 use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.

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

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

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

Create a Subnet for the Administration Host

Oracle WebLogic Server for OKE can create a subnet for your domain's administration compute instance, 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 OKE, the administration compute instance is assigned to a subnet. We recommend that you use a private subnet.

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

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

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

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

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

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the Bastion Host

Oracle WebLogic Server for OKE can create a public subnet in Oracle Cloud Infrastructure for the bastion compute instance that is used to access your private Oracle WebLogic Server 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 OKE, we recommend that you use a private subnet for the Kubernetes cluster and the administration compute instance. Because these resources can not be directly accessed from outside of Oracle Cloud, Oracle WebLogic Server for OKE creates a bastion compute instance on a public subnet.

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

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

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

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

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

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

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the Load Balancer

Oracle WebLogic Server for OKE can create a subnet for the load balancer that is used to access an Oracle WebLogic Server 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, Oracle WebLogic Server for OKE creates two load balancers and assigns them to a public subnet.

- The public load balancer is used to access applications on the WebLogic managed servers.
- The private load balancer is used to access the WebLogic Server administration console and the Jenkins console. It is not assigned a public IP address from the subnet.

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

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

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

- The subnet must have a security list that enables outbound access to the WebLogic managed server ports (by default, 7003 and 7004) on the subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables outbound access to the Jenkins port (80) on the subnet that you plan to use for the Kubernetes cluster.

If you use an existing subnet, you can specify the subnet compartment that is different than the VCN compartment.
Network security groups are an alternative to security lists. After creating a 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).

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

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

Create a Subnet for the File System

Oracle WebLogic Server for OKE can create a subnet for the shared file system that you use to manage your Oracle WebLogic Server 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 OKE, the file system is assigned to a subnet. We recommend that you use a private subnet for the file system.

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

If you want to use an existing subnet for the file system when creating a 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 NFS port (2049) from the private subnet that you plan to use for the Kubernetes cluster.
- The subnet must have a security list that enables inbound access to the NFS port from the private subnet that you plan to use for the administration compute instance.

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

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

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

See VCNs and Subnets in the Oracle Cloud Infrastructure documentation.

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.

A JRF-enabled domain supports the Oracle Application Development Framework (ADF). When you create a domain with Oracle WebLogic Server for OKE and associate it with an existing database, Oracle WebLogic Server for OKE does the following:

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

Choose one of these database options:

- Oracle Autonomous Database
  - Create a serverless database. Oracle WebLogic Server for OKE does not yet support using a dedicated deployment database.
  - See Creating an Autonomous Database in the Oracle Cloud Infrastructure documentation.
- Oracle Cloud Infrastructure Database
  - Create a bare metal, virtual machine (VM), or Exadata DB system.
  - The Virtual Cloud Network (VCN) of the Oracle Cloud Infrastructure database must be same as the WebLogic Server VCN.
  - See Creating Bare Metal and Virtual Machine DB Systems or Managing Exadata DB Systems in the Oracle Cloud Infrastructure documentation.

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

- Oracle Autonomous Database - Update your access control list (ACL), if necessary.
- Oracle Cloud Infrastructure Database - Update the network security group that is assigned to the database, or update the security lists for the subnet on which the database was created, if necessary.

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

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

Oracle WebLogic Server for OKE supports the same database versions and drivers as those for on-premise WebLogic Server installations. See *System Requirements and Supported Platforms for Oracle Fusion Middleware 12c (12.2.1.4.0)* at Oracle Fusion Middleware Supported System Configurations.

Create a Confidential Application

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

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

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

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

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

1. From the Oracle Identity Cloud Service Console, click the navigation menu, and then select **Applications**.
2. Click **Add**.
3. Select **Confidential Application**.
4. Enter a **Name**, and then click **Next**.
5. Click **Configure this application as a client now**.
6. For **Allowed Grant Types**, select **Client Credentials**.
7. Below **Grant the client access to Identity Cloud Service Admin APIs**, click **Add**.
8. Select **Identity Domain Administrator**, and then click **Add**.
9. (Optional) For a WebLogic Server 12.2.1.4 domain only, add **Cloud Gate App Role**. You can add this role after you create your WebLogic Server domain but you may need to restart the domain.

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

10. Complete the Add Confidential Application wizard. Record the values of **Client ID** and **Client Secret**.
11. Select the check box for your application, click **Activate**, and then click **OK**.

12. In the Oracle Cloud Infrastructure console, create a secret in a vault to store the client secret of your confidential application.

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

Validate Existing Network Setup

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

The helper scripts perform the following validations and functions:

- Validates if the service gateway or the NAT gateway is created for the administration instance private subnet and the worker nodes private subnets.
- Validates if internet gateway is created for public bastion, file shared system and load balancer subnets.
- Checks if port 22 in WebLogic Server Subnet is open for access to the CIDR of the bastion instance subnet or bastion host IP.
- Checks if the private subnet for the Oracle WebLogic Server compute instances using the service gateway route rule has **All *<Region>* Services In Oracle Services Network** as the destination.
- Checks if the existing subnet for the load balancer has a security list that enables inbound access to ports 80 and 443.
- Validates if all protocols are open in private subnet for Kubernetes worker node for the Worker CIDR range.
- Validates if all protocols are open in private subnet for Kubernetes worker node for the VCN CIDR range.
- Validates if the file shared system has a security list that enables outbound access to ports 111 and 2048 (both TCP and UDP).
- Validates if the database port is accessible from WebLogic Server subnets.

Using the Validation Script

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

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

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

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

2. Run the `validateoke.sh` command.

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

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

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

3. Based on the domain type, run the following commands prior to provisioning:

   - Basic domain

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

   - JRF-enabled domain

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

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

```
validateoke.sh

example_user@cloudshell:~ (us-phoenix-1)$ ./validateoke.sh -b <Bastion Subnet OCID>
-a <Administration Host Subnet OCID> -w <Worker Subnet OCID> -f <File Shared System Subnet OCID>
-l <Load Balancer Subnet OCID> -d <DB Subnet OCID>
ERROR: SSH port 22 is not open for access by [0.0.0.0/0] in <Bastion Subnet
```

```
OCID>
WARNING: SSH port 22 is not open for access by Bastion Subnet CIDR
[10.0.0.0/24] in private Admin Host Subnet [<Administration Host
Subnet OCID>]
ERROR: Missing Service or NAT gateway in the VCN of the private
ADMIN_SUBNET Host subnet ocid [<Administration Host Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the BASTION_SUBNET
subnet [<Bastion Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the LB_SUBNET subnet
[<Load Balancer Subnet OCID>]
WARNING: Missing internet gateway in the VCN of the FSS_SUBNET_OCID
subnet [<File Shared System Subnet OCID>]
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 31474
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 10256
WARNING: For LB CIDR - Ports are not open in Workers Subnet CIDR 31804
WARNING: All Ports are not open for LB Subnet CIDR
WARNING: All Ports are not open for LB Subnet CIDR
WARNING: All Ports are not open for LB Subnet CIDR
ERROR: All Protocols are not open for WORKER's Subnet CIDR
ERROR: All Protocols are not open in WORKER's Subnet for VCN CIDR
ERROR: TCP -- 111 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2048 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2049 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2050 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 111 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2048 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2049 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: TCP -- 2050 -- Port is not open in FSS Subnet for VCN CIDR
ERROR: DB port 1521 is not open for access by VCN CIDR [10.0.0.0/16]
or Worker Subnet CIDR [10.0.1.0/24] in DB Subnet [<DB Subnet OCID>]
```

2

Create a Domain with Oracle WebLogic Server for OKE

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

Topics:

About Creating a Domain

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

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

- Domain Type

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

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

- Virtual Cloud Network (VCN)

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

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

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

- Subnets

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

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

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

- Network Access

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

- Load Balancers

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

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

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

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

Create a Stack

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

Launch a new stack from Marketplace. For a basic domain, 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 a Vault and an encryption key within that Vault or use an existing Vault and encryption key. See Create an Encryption Key

- Create the secret for the password that you want to use for the domain. See Create Secrets with Passwords.
- Create the secret for the auth token for a user with access to Oracle Cloud Infrastructure Registry. See Create Secrets with Passwords.

Oracle WebLogic Server for OKE 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 Subnet for the Kubernetes Cluster
- Create a Subnet for the Administration Host
- Create a Subnet for the Bastion Host
- Create a Subnet for the Load Balancer
- Create a Subnet for the File System

 Tutorial

Topics:

- Launch a Stack
- Configure Stack Information
- Configure WebLogic Server on Container Cluster
- Configure Advanced WebLogic Server Parameters
- Configure the Container Cluster
- Configure the Administration Instances
- Configure the Network
- Configure the Database
- Configure the File System
- Configure the Registry
- Create OCI Policies
- Configure WebLogic Authentication with Oracle Identity Cloud Service
- Create the 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  and select **Marketplace**.
3. Select an application that matches the edition of Oracle WebLogic Server that you want to provision.
   - **Oracle WebLogic Server Enterprise Edition for OKE BYOL**

- **Oracle WebLogic Server Enterprise Edition for OKE UCM**
- **Oracle WebLogic Suite for OKE BYOL**
- **Oracle WebLogic Suite for OKE UCM**

4. Select a required Oracle WebLogic Server version to use from the list of Model in Image or Domain in Image images.

   To identify whether a version uses the Model in Image or the Domain in Image WLS Operator pattern, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

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

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

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

   The Create Stack wizard is displayed.

Configure Stack Information

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

1. On the Stack Information page of the Create Stack wizard, enter a name for your stack.
2. Enter a description for the stack (optional).
3. Specify one or more tags for your stack (optional).
4. Click **Next**.

   The Configure Variables page opens.

Configure WebLogic Server on Container Cluster

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

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

   The maximum character length is 16.

   This prefix is used by all the created resources, except load balancers.

2. Enter the SSH public key, by either uploading the SSH key file or copy-pasting the SSH key information.
3. Select the number of running managed servers in the domain you want to create. You can specify up to `9`.

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

   Managed servers are members of a WebLogic Server cluster.

4. Enter a user name for the WebLogic Server administrator.
5. Enter the OCID of the secret for the password for the WebLogic Server administrator. See Create Secrets with Passwords.
6. If required, change the default domain name.

Configure the Network

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

1. In the 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 compute instances are created in the stack compartment that you selected earlier upon launching the stack. Select a network compartment if you want the network resources to be in a different compartment than the compute instances.
2. You can either create a new VCN, use an existing VCN but create *new* subnet resources, or an existing VCN and *existing* subnets.

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

   - To create resources in a new VCN, select **Create New VCN** from the **Virtual Cloud Network Strategy** dropdown, and then specify the following:
     - A CIDR for the new VCN
     - A shape for the private load balancer
     - A shape for the public load balancer.

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

   - To use an existing VCN but create *new* subnet resources, select **Use Existing VCN** from the **Virtual Cloud Network Strategy** dropdown, then do the following:
     a. From the **Existing Network** dropdown, select the name of an existing VCN.
     b. *Do not* select the **Use Existing Subnet** check box.

c. Specify public subnet CIDRs for the bastion host and load balancers.

d. Specify private subnet CIDRs for administration host, file system and mount target (storage) host, and Kubernetes cluster and node pool.

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

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

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

g. Select a minimum and maximum flexible shape for a public load balancer. By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

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

h. If you want to use a public load balancer with a reserved public IP, specify the OCID of the public IP for the load balancer.

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

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

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

  d. Keep the default selection for **Provision Bastion node on Public Subnet** check box.
  e. Select the name of an existing public subnet for the bastion host.
  f. Select the names of existing private subnets for the Kubernetes cluster and node pool, administration host, and the file system and mount target (storage) host.
  g. Select the name of an existing public subnet for the load balancer.
  h. Enter the Oracle Cloud Identifier (OCID) for an existing NAT gateway or service gateway.
  i. Select a minimum and maximum flexible shape for a private load balancer.
  By default, the minimum bandwidth size is set to 10 Mbps and maximum to 100 Mbps.

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

  j. Select a minimum and maximum flexible shape for a public load balancer.
  By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

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

  k. If you want to use a public load balancer with a reserved public IP, specify the OCID of the public IP for the load balancer.

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

- To use an existing VCN and *existing* subnets without bastion configuration, select **Use Existing VCN** from the **Virtual Cloud Network Strategy** dropdown, then do the following:
  a. From the **Existing Network** dropdown, select the name of an existing VCN.
  b. Select the **Use Existing Subnet** check box.
  c. Select the **Subnet Compartment** to use for the existing subnet.
  The subnet compartment is different than the VCN compartment. The subnets for the bastion host, load balancers, Kubernetes cluster and node pool, administration host, and the file system and mount target host, use this same subnet compartment.

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

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

> **Note:**
>
> It is recommended to deselect the **Provision Bastion Node on Public Subnet** check box only in network with fast connect setup.
>
> In this case, no status is returned for provisioning, then you must check the status of provisioning in the **Logs** under **Application Information** of the stack, and view the error or success messages in the `/u01/logs/provisioning.log` file on the administration instance.
>
> To get the internal and external load balancer IP addresses for acessing the Jenkins Console, WebLogic Console, and the WebLogic Cluster Load Balancer, see Access the Load Balancer IP for No Bastion Host.

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

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

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

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

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

i. Select a minimum and maximum flexible shape for a public load balancer. By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

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

j. If you want to use a public load balancer with a reserved public IP, specify the OCID of the public IP for the load balancer.

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

Configure the Container Cluster

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

- Create a Container Cluster
- Use an Existing Cluster

Create a Container Cluster

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

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

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

   In the WebLogic node pool: For 2 or more running managed servers, select a shape with 2 or more OCPUs. For example, `VM.Standard2.2` instead of `VM.Standard2.1`.

3. Specify a CIDR for the pods in the Kubernetes cluster.
4. Specify a CIDR for the Kubernetes services that are exposed.

5. *Optional:* To encrypt the Kubernetes secrets at rest in `etcd` by using the master encryption key in the OCI vault service, select **Kubernetes Secret Encryption** and enter the vault key OCID.

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

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

Use an Existing Cluster

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

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

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

Configure the Administration Instances

Specify where you want to create the Oracle WebLogic Server for OKE administration instances and select the shapes to use.

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

> **Note:**
>
> This option is not available if you deselect the **Provision Bastion Node on Public Subnet** check box.

Configure the Database

A basic domain does not require a database.

In the Database section of the Configure Variables page, for **Database Strategy**, select **No Database** if you are creating a basic domain.

To create a domain that uses an existing database, see Create a JRF-Enabled Domain.

Configure the File System

Specify where you want to create the shared file system.

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

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

2. Enter the existing mount target ID (optional).

   When you use an existing subnet to provision an Oracle WebLogic Server for OKE cluster, you can specify the existing mount target OCID. If not specified, a new mount target is created for the file system. This mount target should be in the same subnet where the new file system is created.

3. Select the compartment for the existing mount target (optional).

   You can use this option to create the mount target in a different compartment than the stack compartment.

Configure the Registry

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

1. In the **Registry User Name** field, enter a user name that Kubernetes uses to access the image in the registry.
2. In the **Secrets OCID for Registry Authentication Token** field, enter the OCID for the secret for the auth token generated for the registry user.

Create OCI Policies

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

If you are not an administrator, the necessary groups and policies must be in place before you can create a 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 with Oracle Identity Cloud Service

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

To use Oracle Identity Cloud Service for authentication:

1. Select **Enable Authentication Using Identity Cloud Service**.

   The default values of the IDCS host name and port name are displayed. If required, you can override the default domain name and port that you use to access Oracle 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, and OCID of the secret that contains the client secret of an existing confidential application in this Oracle Identity Cloud Service instance.

   You can override the default port used for the Oracle Identity Cloud Service App Gateway, if required.

Create the Stack

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

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

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

Use Your New Domain

Access and manage your new basic domain after creating a stack.

Typical tasks that you might perform after creating a domain:

- View the cloud resources that were created to support your domain. See View the Cloud Resources for a Domain
- Access the WebLogic Server Administration Console. See Access the WebLogic Console.
- Access the Jenkins build engine. See Access the Jenkins Console.
- Deploy an Java Enterprise Edition (Java EE) application. See Deploy a Sample Application.
- Update the domain (for example, deploy applications and libraries, apply a patch). See Update a Domain in Oracle WebLogic Server for OKE.

Create a JRF-Enabled Domain

Use Oracle WebLogic Server for OKE to create a stack that includes an Oracle WebLogic Server domain with the Java Required Files (JRF) components, network resources, Kubernetes cluster, compute instances, and load balancers.

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

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

- Create an autonomous database or DB System database. See Create a Database.
- Identify the pluggable database (PDB) name. This is required only for Oracle Cloud Infrastructure Database (DB System) running Oracle Database 12c or later.
- If you use Oracle Autonomous Database or DB System, the database can be in a different compartment than the Oracle WebLogic Server for OKE compute instances. Plan to use the same compartment in which you created the autonomous database, or create a compartment for the domain resources. See Create a Compartment.
- Create an SSH key. See Create an SSH Key.
- Create the secret for the password that you want to use for the WebLogic server domain. See Create Secrets with Passwords.
- Create the secret for the auth token for a user with access to Oracle Cloud Infrastructure Registry. This is the administrator password you provided when you created the database. See Create Secrets with Passwords.
- Create the secret for DB administrator password. See Create Secrets with Passwords.

If using an autonomous database, Oracle WebLogic Server for OKE can create the VCN and subnets for you when you create your new domain. If using an existing VCN and a DB System database or autonomous database, you can create new subnets or use existing subnets.

If you want to use an existing VCN or existing subnets, then they must meet certain requirements. See:

- Create a Virtual Cloud Network
- Create a Subnet for the Kubernetes Cluster
- Create a Subnet for the Administration Host
- Create a Subnet for the Bastion Host
- Create a Subnet for the Load Balancer
- Create a Subnet for the File System

Topics:

Launch a Stack

Sign in to Marketplace and specify initial stack information.

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

   To identify whether a version uses the Model in Image or the Domain in Image WLS Operator pattern, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

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

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

   > **Note:**
   >
   > You must select the compartment where the database you intend to use is located. The database can be in a different compartment than the domain compute instances.

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

   The Create Stack wizard is displayed.

Configure Stack Information

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

1. On the Stack Information page of the Create Stack wizard, enter a name for your stack.
2. Enter a description for the stack (optional).
3. Specify one or more tags for your stack (optional).
4. Click **Next**.

   The Configure Variables page opens.

Configure WebLogic Server on Container Cluster

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

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

   The maximum character length is 16.

   This prefix is used by all the created resources, except load balancers.

2. Enter the SSH public key, by either uploading the SSH key file or copy-pasting the SSH key information.
3. Select the number of running managed servers in the domain you want to create. You can specify up to `9`.

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

   Managed servers are members of a WebLogic Server cluster.

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

5. Enter the OCID of the secret for the password for the WebLogic Server administrator. See Create Secrets with Passwords.
6. If required, change the default domain name.

Configure the Network

Define the Virtual Cloud Network (VCN) and the subnets configuration for the JRF-enabled domain.

> **Note:**
>
> If you're using a database in Oracle Cloud Infrastructure Database (DB System), make sure you select **Use Existing VCN**. You cannot create a new VCN for a domain that uses a DB System database.

1. In the Network section of the Configure Variables page, select the **Network Compartment** in which to create the network resources for this domain. Take note of the following:
   - If you don't specify a network compartment, then all the network resources and compute instances are created in the stack compartment that you selected earlier upon launching the stack. Select a network compartment if you want the network resources to be in a different compartment than the compute instances.
   - If you're using a database in Oracle Cloud Infrastructure Database (DB System), the database VCN and the Kubernetes cluster VCN must be the same. Select the compartment in which the VCN of the database you intend to use is located.
2. You can either create a new VCN, use an existing VCN but create *new* subnet resources, or an existing VCN and *existing* subnets.

   For an existing VCN and existing subnet, you can configure a bastion compute instance to provide access to the WebLogic Server compute instances on a private subnet. However, creating the bastion node on public subnet is optional.
   - To create resources in a new VCN, select **Create New VCN** from the **Virtual Cloud Network Strategy** dropdown, and then specify the following:
     - A CIDR for the new VCN
     - A shape for the private load balancer
     - A shape for the public load balancer.

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

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

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

  g. Select a minimum and maximum flexible shape for a public load balancer. By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

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

h. If you want to use a public load balancer with a reserved public IP, specify the OCID of the public IP for the load balancer.

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

- To use an existing VCN and *existing* subnets with bastion configuration, select **Use Existing VCN** from the **Virtual Cloud Network Strategy** dropdown, then do the following:

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

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

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

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

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

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

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

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

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

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

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

j. Select a minimum and maximum flexible shape for a public load balancer. By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

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

k. If you want to use a public load balancer with a reserved public IP, specify the OCID of the public IP for the load balancer.

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

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

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

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

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

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

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

> **Note:**
>
> It is recommended to deselect the **Provision Bastion Node on Public Subnet** check box only in network with fast connect setup.
>
> In this case, no status is returned for provisioning, then you must check the status of provisioning in the **Logs** under **Application Information** of the stack, and view the error or success messages in the `/u01/logs/provisioning.log` file on the administration instance.
>
> To get the internal and external load balancer IP addresses for acessing the Jenkins Console, WebLogic Console, and the WebLogic Cluster Load Balancer, see Access the Load Balancer IP for No Bastion Host.

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

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

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

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

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

i. Select a minimum and maximum flexible shape for a public load balancer.
By default, the minimum bandwidth size is set to 10 Mbps and maximum to 400 Mbps.

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

j. If you want to use a public load balancer with a reserved public IP, specify the OCID of the public IP for the load balancer.

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

Configure the Container Cluster

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

- Create a Container Cluster
- Use an Existing Cluster

Create a Container Cluster

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

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

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

   In the WebLogic node pool: For 2 or more running managed servers, select a shape with 2 or more OCPUs. For example, `VM.Standard2.2` instead of `VM.Standard2.1`.

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

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

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

Use an Existing Cluster

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

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

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

Configure the Administration Instances

Specify where you want to create the Oracle WebLogic Server for OKE compute instances and select the shapes to use.

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

> **Note:**
>
> This option is not available if you deselect the **Provision Bastion Node on Public Subnet** check box.

Configure the Database

You must specify a database in Oracle Autonomous Database or Oracle Cloud Infrastructure Database (DB System) if you're creating a JRF-enabled domain.

> **Note:**
>
> If you're using a DB System database, make sure you've selected **Use Existing VCN** from the Virtual Cloud Network Strategy dropdown in the Network section of the Create Stack wizard. From the Existing Network dropdown, make sure also you've selected the VCN where you created the DB System.

In the Database section of the Configure Variables page, select the **Database Strategy** for your domain, then configure the database parameters.

- For **Autonomous Database** strategy, select or enter the following:
  - The compartment in which you've created the autonomous database.
  - The autonomous database where you want to create the JRF schemas for this WebLogic domain.
  - The OCID of the secret for the password for the ADMIN user to access the selected autonomous database.
  - The service level that the domain should use to connect to the selected autonomous database.
- For **Database System** strategy, select or enter the following:
  - The compartment where the DB System is found.

- The DB system to use for this JRF-enabled domain.
- The database home within the selected DB system.
- The database home version.
- The database within the 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 database administrator (DBA) privileges, and the OCID for the secret for the password for that database administrator.
- The database listen port (1521 by default)

Configure the File System

Specify where you want to create the shared file system.

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

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

2. Enter the existing mount target ID (optional).

   When you use an existing subnet to provision an Oracle WebLogic Server for OKE cluster, you can specify the existing mount target OCID. If not specified, a new mount target is created for the file system. This mount target should be in the same subnet where the new file system is created.

3. Select the compartment for the existing mount target (optional).

   You can use this option to create the mount target in a different compartment than the stack compartment.

Configure the Registry

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

1. In the **Registry User Name** field, enter a user name that Kubernetes uses to access the image in the registry.
2. In the **Secrets OCID for Registry Authentication Token** field, enter the OCID for the secret for the auth token generated for the registry user.

Create OCI Policies

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

If you are not an administrator, the necessary groups and policies must be in place before you can create a 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 with Oracle Identity Cloud Service

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

To use Oracle Identity Cloud Service for authentication:

1. Select **Enable Authentication Using Identity Cloud Service**.

   The default values of the IDCS host name and port name are displayed. If required, you can override the default domain name and port that you use to access Oracle 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, and OCID of the secret that contains the client secret of an existing confidential application in this Oracle Identity Cloud Service instance.

   You can override the default port used for the Oracle Identity Cloud Service App Gateway, if required.

Create the Stack

After you have specified the parameters for your JRF-enabled domain, finish creating the stack.

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

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

NOT_SUPPORTED:

Immediately after you create a JRF-enabled domain, the WebLogic Server pods might be still in `STARTING` state. See WebLogic Server Pods are still in Starting State.

Use Your New Domain

Access and manage your new JRF-enabled domain after creating a stack.

Typical tasks that you might perform after creating a domain:

- View the cloud resources that were created to support your domain. See View the Cloud Resources for a Domain
- Access the WebLogic Server Administration Console. See Access the WebLogic Console.
- Access the Jenkins build engine. See Access the Jenkins Console.
- Deploy an Java Enterprise Edition (Java EE) application. See Deploy a Sample Application.
- Update the domain (for example, deploy applications and libraries, apply a patch). See Update a Domain in Oracle WebLogic Server for OKE.

View the Cloud Resources for a Domain

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

Note:

Load balancers created for your domain are not listed in the Resource Manager. See Load Balancers.

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

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

   You can click **Logs** to find the private and public load balancer IP addresses.

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

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

You can also find your 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.

Access the Load Balancer IP for No Bastion Host

Use `kubectl` commands to access the internal and external load balancer IP addresses if you provision an Oracle WebLogic Server for OKE domain without a bastion.

To get the IP addresses of the load balancer.

1. Access the administration compute instance for your domain. See Access the Administration Instance.
2. Run the following command:
   - For the internal load balancer IP address:

     ```
     kubectl get svc -n ingress-nginx -l app.kubernetes.io/
     name=ingress-nginx /
     -o jsonpath='{.items[*].status.loadBalancer.ingress[0].ip}'
     ```

     Output example:

     ```
     <internal_lb_ip1>
     ```

     You can use this internal load balancer IP address to access the Jenkins Console and WebLogic Console. See Access the Jenkins Console and Access the WebLogic Console.
   - For the external load balancer IP address:

     ```
     kubectl get svc -n ingress-nginx -l app.kubernetes.io/
     name=ingress-nginx-external /
     -o jsonpath='{.items[*].status.loadBalancer.ingress[0].ip}'
     ```

     Output example:

     ```
     <external_lb_ip1>
     ```

     You can use this external load balancer IP address to access the WebLogic Cluster Load Balancer.

The URL format is:

```
weblogic_cluster_lb_url=https://<external_lb_ip1>/<application
context>
```

About the Resources in a Stack

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

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

Topics:

Compute Instances

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

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

- Bastion instance - Has the name *resourceprefix*-bastion
- Administration instance - Has the name *resourceprefix*-admin
- Two or more Kubernetes worker nodes - Each has the name oke-*generated-alphanumeric-string-n*

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

Network Resources

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

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

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

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

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

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

Load Balancers

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

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

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

The public load balancer distributes traffic across the managed servers in your domain, and is accessible from a single IP address. The public load balancer resource is provisioned with the following:

- A public IP address
- A backend set, which is identified by the name `TCP-443`. The backend set configures the load balancing policy.
- A rule set, which has the name `wls_ssl_rule`. The request header rule `WL-PROXY-SSL` is defined in the rule set.
- A listener named `HTTP-443`. The listener handles traffic on port 443 and uses SSL.
- A certificate with the name `oke-ssl-secret`.

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

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

Kubernetes Resources

Oracle WebLogic Server for OKE provisions a Kubernetes cluster with two worker nodes for your Oracle WebLogic Server domain in Oracle Cloud Infrastructure.

In the Oracle Cloud Infrastructure Console, use the navigation menu and select **Developer Services**. Under the **Containers** group, click **Kubernetes Clusters**. When you select the compartment you specified to use for the stack, you'll see the Kubernetes cluster provisioned for your WebLogic Server domain.

The cluster and node resource names are as follows:

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

File System Resources

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

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

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

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

Registry Resources

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

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

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

The list of registry resources provisioned include:

- *resourceprefix*/infra/cisystem-jenkins-controller
- *resourceprefix*/infra/cisystem-jenkins-agent
- *resourceprefix*/infra/nginx-ingress-controller
- *resourceprefix*/infra/oraclelinux
- *resourceprefix*/infra/weblogic-kubernetes-operator
- *resourceprefix*/*domainname*/wls-domain-base
- *resourceprefix*/*domainname*/wls-domain-base/*releasenumber*

domainname is the default domain name or the name you provided during stack creation. *releasenumber* is the stack's WebLogic Server release.

Identity Resources for Dynamic Group and Root Policies

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

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

- Keys and secrets in Oracle Cloud Infrastructure Vault

- The database wallet if you're using an Oracle Autonomous Database to contain the required infrastructure schemas for a JRF-enabled domain

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

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

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

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

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

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

The `osms` policy has the following statement:

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

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

- `Allow dynamic-group servicename-admin-instance-principal-group to use dynamic-groups in tenancy where target.group.id = <dynamic_group_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to manage all-resources in compartment id <stack_compartment_ocid>`
- `Allow service oke to read app-catalog-listing in compartment id <stack_compartment_ocid>`
- `Allow dynamic-group servicename-admin-instance-principal-group to read secret-bundles in tenancy where target.secret.id = <OCID for OCIR token secret>`
- `Allow dynamic-group servicename-admin-instance-principal-group to read secret-bundles in tenancy where target.secret.id = <OCID for weblogic admin password secret>`
- `Allow dynamic-group servicename-admin-instance-principal-group to read secret-bundles in tenancy where target.secret.id = <OCID for database password secret>`
  This policy applies if you create a domain that includes the Java Required Files (JRF) components.
- `Allow dynamic-group servicename-admin-instance-principal-group to use autonomous-transaction-processing-family in compartment ATP_Database_Compartment`
  This policy applies if you create a domain that includes the Java Required Files (JRF) components with ATP database.
- `Allow dynamic-group servicename-admin-instance-principal-group to use keys in tenancy where target.key.id = <oke_encryption_key_ocid>`
  This policy applies if cluster encryption is selected.

Identity Resources for Oracle Identity Cloud Service

If you configure your domain to use Oracle Identity Cloud Service for authentication, Oracle WebLogic Server for OKE provisions additional resources in Oracle Identity Cloud Service to support the domain. These resources are created on the Oracle Identity Cloud Service server.

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, 2021-02-20T21:46:21.288662

3

Update a Domain in Oracle WebLogic Server for OKE

Create and deploy custom domain images using the tools provided with Oracle WebLogic Server for OKE.

Topics:

About Updating a Domain

Learn about updating a domain with Oracle WebLogic Server for OKE.

Topics

Jenkins Pipeline

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

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

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

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

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

The features of Pipeline are:

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

Topics:

Pipeline Jobs

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

Table 3-1 Preconfigured Jobs

Job Type	Job Name	Description
Main	`update-domain`	Deploys and undeploys applications, shared libraries and resources such as Java Messaging Service (JMS) and datasources to a domain.
Main	`sample-app`	Deploys and undeploys applications and the correponding module Yaml using `sample-app.war` that is bundled with the Oracle WebLogic Server for OKE solution.
Main	`jdk-patch`	Applies a new JDK on the source image.
Main	`opatch-update`	Applies OPatches on the source image.
Main	`rebase-full-install`	Creates new image from Fusion Middleware installer, JDK installer, and JDK patches. This job reduces the image size but takes long time to execute as it builds the image from scratch and copies to the existing domain over the new image.

Table 3-1 (Cont.) Preconfigured Jobs

Job Type	Job Name	Description
Child	`backup-and-deploy-domain-job` This job is applicable only for **Model in Image**.	Creates a backup of the Yaml file of the running domain and deploys the changes on the running domain. This job is used by other pipeline jobs and cannot be activated manually.
Child	`test-and-deploy-domain-job` This job is applicable only for **Domain in Image**.	Uses the new domain image from OCIR to create a test domain and validates if all pods are in running state, and then deploys the new domain image to the currently running domain. This job is used by other pipeline jobs and cannot be activated manually.

Pipeline Job Stages in Model in Image

The Pipeline jobs go through different stages during build creation. These stages are distinct set of substeps that are performed through the Pipeline. The following section describes the stages of the Pipeline jobs when you are using **Model in Image**.

To identify a version that uses the Model in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

Table 3-2 Stages in Pipeline

Stage	Description	Applicable to Job Type
PRE-CHECK	• Checks if the domain is in running state. • Checks the status of the introspector pods. • Reads the model domain Yaml file and gets the replica count that are created from the running domain. • Checks if the pods (replica count +1) are in Ready status.	• `update-domain` • `sample-app` • `jdk-update` • `opatch-update` • `rebase-full-install`

Table 3-2 (Cont.) Stages in Pipeline

Stage	Description	Applicable to Job Type
BUILD_DOMAIN	• Runs the Oracle WebLogic Deploy Tooling (WDT) `updateDomain.sh` command and updates the domain with applications, libraries, and resources. • Creates a temporary docker build. • Copies the WDT archive zip file, the WDT domain model Yaml file, and Oracle WebLogic Image tool (WIT) to the temp directory. • Logs in to OCIR and uses docker build to create a new image.	• `update-domain` • `sample-app` • `jdk-update` • `opatch-update` • `rebase-full-install`
OCIR_UPLOAD	Pushes the updated domain image to OCIR.	• `update-domain` • `sample-app` • `jdk-update` • `opatch-update` • `rebase-full-install`
APPLY_JDK	• Uses docker script to apply new patch on top of the existing image. • In the first stage, copies JDK bundle to jdk tmp directory and extracts the contents of the jdk `tar.gz` bundle. • In the second stage, it removes contents of existing image from jdk directory, creates a new jdk directory, and copies jdk contents from first stage to jdk directory in second stage. • Logs in to OCIR and uses docker build to create a new image.	• `jdk-update`
APPLY_WLS_OPATCHES	• Uses WIT update command to apply WLS OPatch on domain image. • Reads the multiple WLS patches provided by user and runs `imagetool update` command to apply patches.	• `opatch-update`

Table 3-2 (Cont.) Stages in Pipeline

Stage	Description	Applicable to Job Type
REBASE_FULL_INSTALL	• Uses the WIT `create` command to create new image with Fusion Middleware installer, JDK installer, JDK patches, WLS OPatches, and the domain copied from source image. • Uses the `docker build` command to copy the artifacts from the existing image to the new Model in Image.	• `rebase-full-install`
DEPLOY_DOMAIN	• Reads the model domain Yaml file of the running domain and creates a timestamp directory in shared file system under domain directory for backups. • Copies the model domain Yaml file of the running domain as `prev-domain.yaml`. • Deletes the running domain and updates the domain Yaml image of the running domain with the new image tag. • Copies the new domain Yaml to the backup directory as `domain.yaml` and applies the new domain image to the currently running domain. If deployment to the running domain fails or the pods do not come up, then the job rollbacks to the previous working image.	• `backup-and-deploy-domain-job`
DOMAIN_VALIDATION	• Checks if the domain is in running state. • Checks the status of the introspector pods. • Reads the model domain Yaml file and gets the replica count created for the running domain. • Checks if the pods (replica count +1) are in Ready status. If deployment to the running domain fails or the pods do not come up, then the job rollbacks to the previous working image.	• `backup-and-deploy-domain-job`

Pipeline Job Stages in Domain in Image

The Pipeline jobs go through different stages during build creation. These stages are distinct set of substeps that are performed through the Pipeline. The following section describes the stages of the Pipeline jobs when you are using **Domain in Image**.

To identify a version uses the Domain in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

> **Important:**
>
> From Oracle WebLogic Server for OKE release 21.2.2 onwards, Domain in Image is deprecated.

Table 3-3 Stages in Pipeline

Stage	Description	Appicable to Job Type
PRE-CHECK	• Checks if the domain is in running state. • Reads the model domain Yaml file and gets the replica count that are created from the running domain. • Checks if the pods (replica count +1) are in Ready status.	• `update-domain` • `sample-app` • `jdk-update` • `opatch-update` • `rebase-full-install`
BUILD_DOMAIN	• Runs the Oracle WebLogic Deploy Tooling (WDT) `updateDomain.sh` command and updates the domain with applications, libraries, and resources. • Creates a temporary docker build. • Copies the WDT archive zip file, the WDT domain model Yaml file, and Oracle WebLogic Image tool (WIT) to the temp directory. • Logs in to OCIR and uses docker build to create a new image.	• `update-domain` • `sample-app` • `jdk-update` • `opatch-update` • `rebase-full-install`
REBASE	Uses WIT to reduce the image size.	• `update-domain` • `sample-app`

Table 3-3 (Cont.) Stages in Pipeline

Stage	Description	Appicable to Job Type
OCIR_UPLOAD	Pushes the updated domain image to OCIR.	• `update-domain` • `sample-app` • `jdk-update` • `opatch-update` • `rebase-full-install`
APPLY_JDK	• Uses docker script to apply new patch on top of the existing image. • In the first stage, copies JDK bundle to jdk tmp directory and extracts the contents of the jdk `tar.gz` bundle. • In the second stage, it removes contents of existing image from jdk directory, creates a new jdk directory, and copies jdk contents from first stage to jdk directory in second stage. • Logs in to OCIR and uses docker build to create a new image.	• `jdk-update`
APPLY_WLS_OPATCHES	• Uses WIT update command to apply WLS OPatch on domain image. • Reads the multiple WLS patches provided by user and runs `imagetool update` command to apply patches.	• `opatch-update`
REBASE_FULL_INSTALL	Uses WIT rebase command to create new image with Fusion Middleware installer, JDK installer, JDK patches, WLS OPatches, and the domain copied from source image.	• `rebase-full-install`
TEST_DOMAIN	Creates new test domain Yaml and applies the test domain Yaml to the running domain.	• `test-and-deploy-domain`

Table 3-3 (Cont.) Stages in Pipeline

Stage	Description	Appicable to Job Type
TEST_DOMAIN_VALIDATION	• Checks if the domain is in running state. • Reads the model domain Yaml file of the test domain and gets the replica count that are created from the running domain Yaml. • Checks if the pods (replica count +1) are in Ready status.	• `test-and-deploy-domain`
DEPLOY_DOMAIN	• Reads the model domain Yaml file of the running domain and creates a timestamp directory in shared file system under domain directory for backups. • Copies the model domain Yaml file of the running domain as `prev-domain.yaml`. • Deletes the running domain and updates the domain Yaml image of the running domain with the new image tag. • Copies the new domain Yaml to the backup directory as `domain.yaml` and applies the new domain image to the currently running domain. If deployment to the running domain fails or the pods do not come up, then the job rollbacks to the previous working image.	• `test-and-deploy-domain`
DOMAIN_VALIDATION	• Checks if the domain is in running state. • Reads the model domain Yaml file and gets the replica count created for the running domain. • Checks if the pods (replica count +1) are in Ready status. If deployment to the running domain fails or the pods do not come up, then the job rollbacks to the previous working image.	• `test-and-deploy-domain`

Project Components

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

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

All Jenkins files are found on the shared file system.

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

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

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

Access the Jenkins Console

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

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

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

The URL format is:

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

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

> **Note:**
>
> If you provision an Oracle WebLogic Server for OKE domain without a bastion instance, you must get the internal load balancer IP address, and then use this IP address in the Jenkins URL. See Access the Load Balancer IP for No Bastion Host to get the internal load balancer IP address.
>
> For example, if the IP address obtained is `<internal_lb_ip1>`, the Jenkins URL format is:
>
> ```
> jenkins_console_url=http://<internal_lb_ip1>/jenkins
> ```

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

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

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

    The SSH command format is:

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

    For example:

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

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

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

Deploy a Sample Application

Use the Jenkins job to deploy or undeploy applications, shared libraries and resources to the new domain using a sample application.

To deploy or undeploy a sample application:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **sample-app**.
3. Click **Build with Parameters**.
4. Select the **Undeploy_Sample_App** check box to undeploy the sample application (optional).
5. Deselect the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).

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

   The **Rollback_On_Failure** check box is selected by default.
6. Click **Build** to run the Pipeline job.

   You can see the different stages of the Pipeline for the `sample-app` job. See Pipeline Job Stages in Domain in Image.
7. Access the sample-app with the URL:

   ```
   https://<external-LB-IP>/sample-app
   ```

   You can use the `kubectl` command to get the external load balancer IP address.

   For example:

   ```
   kubectl get service <resourceprefix>-external -n <resourceprefix>-domain-ns
   ```

> **Note:**
>
> To get the external load balancer IP address, when you provision an Oracle WebLogic Server for OKE domain without a bastion instance, see Access the Load Balancer IP for No Bastion Host.

You can use the WebLogic Server Administration Console to verify that your application is deployed to the domain. See Access the WebLogic Console.

Update the Domain Configuration

Use the Jenkins job to update a domain that you created with Oracle WebLogic Server for OKE for **Model in Image** and **Domain in Image**.

To identify whether a version uses the Model in Image or the Domain in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

> **Important:**
>
> From Oracle WebLogic Server for OKE release 21.2.2 onwards, Domain in Image is deprecated.

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

- File Upload - Uploads the file from the local system.
- Object Storage - Uses the pre-authenticated URL on the Object Storage.
- Shared File System - Uses the path of the shared file storage.

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

You can provide one of the files to update the domain:

- Model Yaml file
- Model Yaml file with variable properties file
- Archive file containing the model Yaml file
- Archive file containing the model Yaml file and the variable properties file
- Archive file containing the model Yaml file, overriding model Yaml file, and the variable properties file
- Archive file and the model Yaml file.

  This archive file should not contain the model Yaml file.
- Archive file, model Yaml file, and the variable properties file

  This archive file should not contain the model Yaml file.

If you are using **Model in Image**:

- The model Yaml file must contain the required secrets like WebLogic Admin Password, runtime encryption secret, and Repository Schema Utility (RCU) schema user password.
- In case of Oracle Cloud Infrastructure Database (DB system), the model Yaml file must include the datasource secret and in case of Oracle Autonomous Database, the model Yaml file must include the datasource secret and wallet with the keystore passwords. See About Data Sources.

If you are using **Domain in Image**:

- The model Yaml file and the variable properties can be encrypted using the `encryptModel` command. See Encrypt Model Tool in *WebLogic Deploy Tooling* documentation.
- You must specify a passphrase if the model Yaml file or the variable properties is encrypted with the Encrypt Model Tool.
- You can also update the Repository Schema Utiity (RCU) password for the model Yaml file; all schema passwords are updated to the same password. See step 10 to update the schema password.
  Before you update the schema password, follow the prerequisites in Prerequisites to Update the Repository Schema Utility Password.

  An example of model Yaml with RCU schema password:

```
domainInfo:
          RCUDbInfo:
            rcu_prefix:<schema_prefix>
              rcu_schema_password:'<new_password>'
```

See RCU Connection Information in *WebLogic Deploy Tooling* documentation.

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

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **update-domain**.
3. Click **Build with Parameters**.
4. Select the source of the archive zip file from the **Archive_Source** list.

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

5. For **Archive_File_Location**, either browse to select the zip file, specify the http pre-authenticated URL, or specify the path of the zip file on the shared file system.

   To know about the structure of the archive file, see archive file in *WebLogic Deploy Tooling* documentation.
6. Select the source of the domain model Yaml file from the **Domain_Model_Source** list.

   If the archive zip file contains the domain model Yaml file, you can skip this step.

   This step is optional if you are using **Domain in Image**.
7. For **Model_File_Location**, either browse to select the Yaml file, specify the http pre-authenticated URL, or specify the path of the Yaml on the shared file system.

   See model YAML file in *WebLogic Deploy Tooling* documentation.

   This step is optional if you are using **Domain in Image**.
8. Select the source of the variable properties file from the **Variable_Source** list.
9. For **Variable_File_Location**, either browse to select the file, specify the http pre-authenticated URL, or specify the path of the properties file on the shared file system.

10. Deselect the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).

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

    The **Rollback_On_Failure** check box is selected by default.

    > **Note:**
    >
    > If you are using **Domain in Image**:
    >
    > - Enter the **Encryption_Passphrase** to encrypt the passwords in the model Yaml or the variable properties file.
    > - Select the **Update_RCU_Schema_Password** check box to update the schema password for the model Yaml file.
    >
    > This option updates the password for all schemas.
    >
    > Select this check box only if you want to update the RCU schema password. Before you update the schema password, follow the prerequisites in Prerequisites to Update the Repository Schema Utility Password.

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

    You can see the different stages of the Pipeline for the `update-domain` job. See Pipeline Job Stages in Domain in Image.

12. Access the deployed application using the external load balancer.

    ```
    https://<external-LB-IP>/sample-app
    ```

    You can use the `kubectl` command to get the external load balancer IP address.

    For example:

    ```
    kubectl get service <resourceprefix>-external -n <resourceprefix>-
    domain-ns
    ```

    > **Note:**
    >
    > To get the external load balancer IP address, when you provision an Oracle WebLogic Server for OKE domain without a bastion instance, see Access the Load Balancer IP for No Bastion Host.

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

Prerequisites to Update the Repository Schema Utility Password

If you are using **Domain In Image**, perform the following prerequisite tasks before you update the Repository Schema Utility (RCU) password with the `update-domain` job.

After the password is updated on the database, WebLogic connection pool periodically tries connecting to the schema using the old password and fails to connect until the RCU password is updated. So, you must change the SQL profile associated with the RCU schema users.

Before you update the RCU schema password:

1. From your computer, open an SSH connection to the database as the `opc` user, then change to the `oracle` user, and start a connectionless command-line session.

   ```
   ssh -i <privKey> opc@<DB IP>
   sudo su oracle
   sqlplus /nolog
   ```

   where, `DB IP` is the public IP address of the database instance, and `privKey` is the full path and name of the file that contains the private key corresponding to the public key associated with the database instance that you want to access.

2. Connect to the database as `sysdba` user using `sqlplus`.

   ```
   connect sys/<password>@//dbhost.subnet1.vcn1.oraclevcn.com:1521/
   pdbName.subnet1.vcn1.oraclevcn.com as sysdba
   ```

   Example:

   ```
   connect sys/<password>@//sidb19-
   scan.admin.existingnetwork.oraclevcn.com:1521/
   sipdb.admin.existingnetwork.oraclevcn.com as sysdba
   ```

3. Set the login attempts for the `DEFAULT` profile to `UNLIMITED` and then check the limit for the `DEFAULT` profile.

   ```
   ALTER PROFILE DEFAULT LIMIT FAILED_LOGIN_ATTEMPTS UNLIMITED;
   select limit from dba_profiles where profile='DEFAULT' /
   and resource_name='FAILED_LOGIN_ATTEMPTS';
   LIMIT
   ----------------------------------------------------------------
   UNLIMITED
   ```

4. List all the dba users where username like `<schema_prefix>__%`.

   ```
   select username from dba_users where username like '<schema_prefix>__%';


   USERNAME
   ------------------------------
   <schema_prefix>_STB
   ```

```
<schema_prefix>_IAU_APPEND
<schema_prefix>_OPSS
<schema_prefix>_WLS
<schema_prefix>_IAU
<schema_prefix>_WLS_RUNTIME
<schema_prefix>_IAU_VIEWER
<schema_prefix>_UMS
<schema_prefix>_MDS
```

For example:

```
select username from dba_users where username like 'SP1601029287_%';
```

All user names are prefixed with `SP1601029287` as in `SP1601029287_STB`.

5. Change the password for the following users.

```
alter user <schema_prefix>_STB identified by <new_password>;
alter user <schema_prefix>_IAU_APPEND identified by <new_password>;
alter user <schema_prefix>_OPSS identified by <new_password>;
alter user <schema_prefix>_WLS identified by <new_password>;
alter user <schema_prefix>_IAU identified by <new_password>;
alter user <schema_prefix>_WLS_RUNTIME identified by <new_password>;
alter user <schema_prefix>_IAU_VIEWER identified by <new_password>;
alter user <schema_prefix>_UMS identified by <new_password>;
alter user <schema_prefix>_MDS identified by <new_password>;
```

For example:

```
alter user SP1601029287_STB identified by <new_password>;
```

6. Connect to the database for the MDS user.

```
connect <schema_prefix>_MDS/<password>//dbhost.example.com:1521
Connected
```

7. List the table names in the database for the current user.

```
select table_name from user_tables;
```

8. Exit SQL.

```
exit
```

Update the Repository Schema Utility Password using Secrets

If you are using **Model in Image** and modified the Repository Schema Utility (RCU) password, then you must update the schema password in the domain.

> **Note:**
>
> If you are using **Domain in Image**, the RCU password is handled in the update domain pipeline job by using the **Update_RCU_Schema_Password** check box. See Update the Domain Configuration.

During initial provisioning, we create a secret named `<resource_prefix>-rcu-access`, which contains all the RCU related information, like `db_connect_String`, schema prefix, and schema password.

Complete the following steps to update the schema password in the domain:

1. Shutdown the domain.

   Run the following command:

   ```
   kubectl edit domain -n <domain_ns> -o yaml
   ```

   Sample output:

   ```
   kind: Domain
     metadata:
       name: domain1
     spec:
       serverStartPolicy: "NEVER"
   ```

   Change the `serverStartPolicy` value, from `IF_NEEDED` to `NEVER`. See Starting and stopping servers.

2. If you have not changed the RCU schema password on the database, then complete this step.

   a. In the administration server, run the `rcu_secret.sh` script, which is located at `/u01/scripts/pipeline/helper-scripts`. This displays the existing `schemaPrefix` information.

   b. Connect to the database as `sysdba` user using `sqlplus`.

   ```
   connect sys/<password>@//dbhost.subnet1.vcn1.oraclevcn.com:1521/
   pdbName.subnet1.vcn1.oraclevcn.com as sysdba
   ```

   Example:

   ```
   connect sys/<password>@//sidb19-
   scan.admin.existingnetwork.oraclevcn.com:1521/
   sipdb.admin.existingnetwork.oraclevcn.com as sysdba
   ```

c. Set the login attempts for the `DEFAULT` profile to `UNLIMITED` and then check the limit for the `DEFAULT` profile.

```
ALTER PROFILE DEFAULT LIMIT FAILED_LOGIN_ATTEMPTS UNLIMITED;
select limit from dba_profiles where profile='DEFAULT' /
and resource_name='FAILED_LOGIN_ATTEMPTS';
LIMIT
----------------------------------------------------------------
---
UNLIMITED
```

d. List all the dba users where username like `<schema_prefix>__%`.

```
select username from dba_users where username like
'<schema_prefix>__%';



USERNAME
--------------------------------
<schema_prefix>_STB
<schema_prefix>_IAU_APPEND
<schema_prefix>_OPSS
<schema_prefix>_WLS
<schema_prefix>_IAU
<schema_prefix>_WLS_RUNTIME
<schema_prefix>_IAU_VIEWER
<schema_prefix>_UMS
<schema_prefix>_MDS
```

For example:

```
select username from dba_users where username like
'SP1601029287_%';
```

All user names are prefixed with `SP1601029287` as in `SP1601029287_STB`.

e. Change the password for the following users:

```
alter user <schema_prefix>_STB identified by <new_password>;
alter user <schema_prefix>_IAU_APPEND identified by
<new_password>;
alter user <schema_prefix>_OPSS identified by <new_password>;
alter user <schema_prefix>_WLS identified by <new_password>;
alter user <schema_prefix>_IAU identified by <new_password>;
alter user <schema_prefix>_WLS_RUNTIME identified by
<new_password>;
alter user <schema_prefix>_IAU_VIEWER identified by
<new_password>;
alter user <schema_prefix>_UMS identified by <new_password>;
alter user <schema_prefix>_MDS identified by <new_password>;
```

For example:

```
alter user SP1601029287_STB identified by <new_password>;
```

f. Connect to the database for the MDS user.

```
connect <schema_prefix>_MDS/<password>//dbhost.example.com:1521
Connected
```

g. List the table names in the database for the current user.

```
select table_name from user_tables;
```

h. Exit SQL.

```
exit
```

3. Delete the existing kubernetes secret: `<resource_prefix>-rcu-access`
4. Run the following command to obtain the name of the secret.

```
kubectl get secrets -n domain10-ns |grep rcu-access
```

5. Recreate the secret with the same name. The name that you obtained in the previous step.

   The `<resource_prefix>-rcu-access` secret has certain fields other than the schema password, which also needs to be specified based on the database type. When you run the `rcu_secret.sh` script, located at `/u01/scripts/pipeline/helper-scripts/`, it outputs all the other required fields in addition to the schema password for recreating the secret.

   For ATP database:

```
[opc@wrjrf1-admin helper-scripts]$./rcu_secret.sh
rcu_db_name = <atp_db_name_low>
rcu_prefix = <prefix>
rcu_wallet_password = <password>
[opc@wrjrf1-admin helper-scripts]$kubectl create secret generic -n
<domain_ns> '<resource_prefix>-rcu-access' --from-
literal=rcu_db_name=<atp_db_name_low> --from-literal=rcu_prefix=<prefix>
--from-literal=rcu_wallet_password=<password> --from-
literal=rcu_schema_password=<new_password>
```

   For OCI, SI, or RAC database:

```
[opc@wrjrf1-admin helper-scripts]$./rcu_secret.sh
rcu_admin_password = <admin_password>
rcu_db_conn_string = <connect_String>
rcu_db_user = sys
rcu_prefix = <prefix>
[opc@wrjrf1-admin helper-scripts]$kubectl create secret generic -n
<domain_ns> '<resource_prefix>-rcu-access' --from-
```

```
literal=rcu_admin_password=<admin_password> --from-
literal=rcu_db_conn_string=name-
scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:1521/
db0409_pdb1.subnet2ad2phx.paasdevjcsphx.oraclevcn.com --from-
literal=rcu_db_user=sys --from-literal=rcu_prefix=<prefix> --from-
literal=rcu_schema_password=<new_password>
```

6. Change the `serverStartPolicy` value, from `NEVER` to `IF_NEEDED`, and then increment the `restartVersion`.

   ```
   kind: Domain
     metadata:
       name: domain1
     spec:
       serverStartPolicy: "IF_NEEDED"
       restartVersion: "3"
   ```

   Change the `serverStartPolicy` value, from `NEVER` to `IF_NEEDED`. Then, increment the `restartVersion` value.

7. Wait for the domain to start for a rolling restart. Then, verify that the datasource `mds-owsm` tests okay in the WLS admin console.

Create a New Domain Image

Use the Jenkins job to create a new domain image from Fusion Middleware installer, JDK installer and WebLogic patches for Oracle WebLogic Server for OKE.

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

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

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

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

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

To create a new domain image:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **rebase-full-install**.
3. Click **Build with Parameters**.
4. Select the source of the JDK installer file from the **JDK_Installer** list.
5. For **JDK_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system.
6. Select the source of the Fusion Middleware installer file from the **FMW_Installer** list.
7. For **FMW_Installer_Location**, specify the http pre-authenticated URL or the path of the JAR file on the shared file system.

8. Select the source of the Weblogic patch file from the **WLS_Opatches** list.

   WLS_Opatches refers to any patch that may be required to patch WLS, ADF/JRF, or Coherence product binaries (including ADR patches, JRF patches, OPSS, OWSM, OPatches).

9. For **Opatches_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system.

   For multiple patches, specify the location in separate lines.

10. Select **Skip_Opatch_Update** check box to create a new image without OPatch update (optional).

11. Deselect the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).

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

    The **Rollback_On_Failure** check box is selected by default.

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

    You can see the different stages of the Pipeline for the `rebase-full-install` job. See Pipeline Job Stages in Domain in Image.

Update the JDK

Use the Jenkins job to replace the Java Development Kit (JDK) for a domain that you created with Oracle WebLogic Server for OKE.

You can use one of the following sources to specify the location of the JDK installer file:

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

  You must specify the location of a JDK that uses the Linux x64 compressed archive format (`.tar.gz`).

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

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

To update the JDK for the domain:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **jdk-patch**.
3. Click **Build with Parameters**.
4. Select the source of the JDK installer file from the **JDK_Installer** list.
5. For **JDK_Location**, specify the http pre-authenticated URL or the path of the zip file on the shared file system.
6. Deselect the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).

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

   The **Rollback_On_Failure** check box is selected by default.

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

You can see the different stages of the Pipeline for the `jdk-patch` job. See Pipeline Job Stages in Domain in Image.

Apply a WebLogic Server Patch

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

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

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

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

To apply the WebLogic patch file:

1. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
2. On the Dashboard page, click **opatch-update**.
3. Click **Build with Parameters**.
4. Select the source of the Weblogic patch file from the **WLS_Opatches** list.

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

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

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

   The **Rollback_On_Failure** check box is selected by default.
7. Click **Build** to run the Pipeline job.

   You can see the different stages of the Pipeline for the `opatch-update` job. See Pipeline Job Stages in Domain in Image.

About Data Sources

For **Model in Image**, learn about creating data sources after you create an Oracle WebLogic Server for OKE domain.

To identify a version that uses the Model in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

For mutable values such as database user names, passwords, and URLs, you must use model macros to reference arbitrary secrets from model Yaml files. All password fields in the model Yaml must use secret macros; passwords should not be directly

included in the model Yaml files. So, to create the data source for the database, you must create data source secrets.

Before creating data sources, you must set up the database to create a schema user. See Prerequisites to Create a Data Source.

You can create data sources that enable you to connect to either an Oracle Autonomous Database or an Oracle Cloud Infrastructure Database (DB System) database. You can also create multi data sources and Active GridLink (AGL) data source when using Oracle Real Application Clusters (RAC) database.

Topics:

Prerequisites to Create a Data Source

Before you create a data source, you must set up the database to create a schema user.

Complete the following step to create a schema user:

1. Create a schema user for the datasource.

   For example, let's create `hello` schema user:

   ```
   create user hello identified by "<password>";

   GRANT CONNECT TO hello;
   GRANT RESOURCE TO hello;
   GRANT DBA TO hello;

   GRANT CREATE SESSION TO hello;
   GRANT UNLIMITED TABLESPACE TO hello;

   ALTER SESSION SET CURRENT_SCHEMA = hello;

   CREATE TABLE products
   ( product_id number(10) not null,
   product_name varchar2(50) not null,
   CONSTRAINT products_pk PRIMARY KEY (product_id)
   );

   insert into products values
   (1, 'lamp');

   commit;
   ```

2. Add the schema user that you created in step 1 to the model Yaml template.

Example:

```
Properties:
            user:
              Value:<schema user name>
```

Create a Data Source for an ATP Database

Use the model Yaml templates to create a data source for an Oracle Autonomous Database.

Following is an example of model Yaml template for app deployment with ATP datasource properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    hellods:
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          JNDIName:
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-
secret:password@@'
          Properties:
            user:
              Value:
            javax.net.ssl.keyStore:
              Value:
            javax.net.ssl.keyStoreType:
              Value: JKS
            javax.net.ssl.keyStorePassword:
              Value: '@@SECRET:@@ENV:DOMAIN_UID@@-keystore-
secret:password@@'
            javax.net.ssl.trustStore:
              Value:
            javax.net.ssl.trustStoreType:
              Value: JKS
            javax.net.ssl.trustStorePassword:
              Value: '@@SECRET:@@ENV:DOMAIN_UID@@-keystore-
secret:password@@'
            oracle.net.ssl_version:
              Value: '1.2'
            oracle.net.ssl_server_dn_match:
              Value: true
```

```
      oracle.net.tns_admin:
        Value:
      oracle.jdbc.fanEnabled:
        Value: false
    JDBCConnectionPoolParams:
      InitialCapacity: 1
      MaxCapacity: 1
      TestTableName: SQL ISVALID
      TestConnectionsOnReserve: true
```

You must set up the database to create a schema user before you create the data source. See Prerequisites to Create a Data Source.

When you create a data source for an ATP database, you need the ATP client credentials or wallet files. So, you must run the download script before you create the data source. See Download the ATP Wallet.

To create a data source for an ATP database:

1. Update the `Properties` section in the model Yaml template.

   For example, update `keyStore` and `trustStore` file locations from the downloaded ATP wallet, `tns_admin` to point to the directory of the ATP wallet, and the schema user you created in Prerequisites to Create a Data Source.

   ```
   Properties:
     user:
       Value: <schema user>
     javax.net.ssl.keyStore:
       Value: /u01/shared/atp_wallet/keystore.jks
     javax.net.ssl.trustStore:
       Value: /u01/shared/atp_wallet/truststore.jks
     oracle.net.tns_admin:
       Value: /u01/shared/atp_wallet
   ```

2. Create the data source secrets in the model Yaml file as follows:

   a. Create `<domain_uid>-datasource-secret` with password and url using the `kubectl` command.

      Example:

      ```
      kubectl create secret generic <domain_uid>-datasource-secret --from-
      literal=password=<password>
      --from-literal=url='jdbc:oracle:thin:@(description= (retry_count=20)
      (retry_delay=3)
      address=(protocol=tcps)(port=1522)(host=adb.us-
      phoenix-1.oraclecloud.com))
      connect_data=(service_name=<service_name>))
      (security=(ssl_server_cert_dn="CN=adwc.uscom-
      east-1.oraclecloud.com,OU=Oracle
      BMCS US,O=Oracle Corporation,L=Redwood City,ST=California,C=US")))' -
      n <resource_prefix>-domain-ns
      ```

   b. Create the `<domain_uid>-keystore-secret` with the password used to download the ATP wallet using the `kubectl` command.

For example:

```
kubectl create secret generic <domain_name>-keystore-secret --
from-literal=password=<password> -n <resource_prefix>ly2-domain-
ns
```

Download the ATP Wallet

The download script unpacks and copies the ATP wallet contents to a node.

1. Open an SSH connection to the domain's Administration Server node as the `opc` user.

   ```
   ssh -i <path_to_private_key> opc@<node_public_ip>
   ```

2. Download the ATP wallet to the administration host using the following wallet script:

   ```
   python /u01/scripts/utils/download_atp_wallet.shutils/
   oci_api_utils.py <atp_database_ocid>
    <atp_wallet_password> <path_to_extract_wallet_files>
   ```

   Example:

   ```
   python /u01/scripts/utils/oci_api_utils.py
   download_atp_wallet_with_pwd
   ocid1.autonomousdatabase.oc1.phx.abyhqljtzrr25tfpdyzbjq5udz3lz2hkb5t
   xvtfejckwd25z6hjg6qbxm4ta <password>
   /u01/shared/atp_wallet
   ```

   Eight files are extracted to the subdirectory. The following is an example of the script response:

   ```
   <Apr 29, 2021 09:51:35 PM GMT> <INFO> <oci_api_utils> <(host:ly2-
   admin.okeworkdersregi.paasdevjcsphx.oraclevcn.com) - <WLSOKE-VM-
   INFO-0100> : ATP Wallet downloaded>
   Archive:  /tmp/atp_wallet.zip
     inflating: /u01/shared/atp_wallet/README
     inflating: /u01/shared/atp_wallet/cwallet.sso
     inflating: /u01/shared/atp_wallet/tnsnames.ora
     inflating: /u01/shared/atp_wallet/truststore.jks
     inflating: /u01/shared/atp_wallet/ojdbc.properties
     inflating: /u01/shared/atp_wallet/sqlnet.ora
     inflating: /u01/shared/atp_wallet/ewallet.p12
     inflating: /u01/shared/atp_wallet/keystore.jks
   ```

Create a Data Source for a DB System Database

Use the model Yaml templates to create a data source for an Oracle Cloud Infrastructure Database.

Following is an example of model Yaml template for app deployment with single-instance (SI) DB System data source properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    hellods:
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          JNDIName: jdbc/hellods
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-
secret:password@@'
          Properties:
            user:
              Value:
            oracle.net.CONNECT_TIMEOUT:
              Value: '120000'
            SendStreamAsBlob:
              Value: true
        JDBCConnectionPoolParams:
          InitialCapacity: 1
          MaxCapacity: 1
          TestTableName: SQL ISVALID
          TestConnectionsOnReserve: true
```

You must set up the database to create a schema user before you create the data source and update . See Prerequisites to Create a Data Source.

1. Update the schema user that you created in the `Properties` section in the model Yaml template.

   See Prerequisites to Create a Data Source.

   For example:

   ```
   Properties:
     user:
   ```

```
Value: <schema user>
```

2. Create the data source secrets in the model Yaml file as follows:
   a. Obtain the `tns` connect string using the python script.

      ```
      python /u01/scripts/db/db_util.py get-connect-string
      ```

   b. Create `<domain_uid>-datasource-secret` with password and url using the `kubectl` command.

      For example:

      ```
      kubectl create secret generic <domain_name>-datasource-secret --
      from-literal=password=<password>
      --from-literal=url='jdbc:oracle:thin:@//wrdb19-
      scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:1521/
      WRPDB.SUBNET2AD2PHX.PAASDEVJCSPHX.ORACLEVCN.COM'
      -n <resource_prefix>-domain-ns
      ```

Create a Multi Data Source for a RAC Database

Use the model Yaml templates to create a multi data source for an Oracle Real Application Cluster (RAC) database with WebLogic Enterprise Edition.

Following is an example of model Yaml template for app deployment with multi data source (RAC as infra DB with WebLogic Enterprise Edition) properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    'db1-hellods':
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          GlobalTransactionsProtocol: None
          JNDIName: [ 'jdbc/db1-hellods' ]
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-db1datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-
db1datasource-secret:password@@'
          Properties:
            user:
              Value:
        JDBCConnectionPoolParams:
          StatementCacheSize: 0
          TestTableName: SQL ISVALID
```

```
          InitialCapacity: 1
          MaxCapacity: 1
    'db2-hellods':
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        JDBCDataSourceParams:
          GlobalTransactionsProtocol: None
          JNDIName: [ 'jdbc/db2-hellods' ]
        JDBCDriverParams:
          DriverName: oracle.jdbc.OracleDriver
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-db2datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-db2datasource-
secret:password@@'
          Properties:
            user:
              Value:
        JDBCConnectionPoolParams:
          StatementCacheSize: 0
          TestTableName: SQL ISVALID
          InitialCapacity: 1
          MaxCapacity: 1
    'hellods':
       Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
       JdbcResource:
         DatasourceType: MDS
         JDBCDataSourceParams:
           AlgorithmType: 'Load-Balancing'
           DataSourceList: [ 'db1-hellods','db2-hellods' ]
           JNDIName: [ jdbc/hellods ]
```

You must set up the database to create a schema user before you create the data source. See Prerequisites to Create a Data Source.

1. Update the schema user that you created in the `Properties` section in the model Yaml template.

   See Prerequisites to Create a Data Source.

   For example:

   ```
   Properties:
     user:
       Value: <schema user>
   ```

2. Create the data source secrets in the model Yaml file as follows:

   a. Open an SSH connection to the domain's Administration Server node as the `opc` user.

      ```
      ssh -i <path_to_private_key> opc@<node_public_ip>
      ```

b. Go to `u01/shared/helper-scripts` location and obtain the connect string for each node using the following commands:

```
./url.sh
URL1 = jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=<host_name>)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=<service_name>)
(INSTANCE_NAME=<instance1_name>)))
URL2 = jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=<host_name>)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=<service_name>)
(INSTANCE_NAME=<instance2_name>))
```

c. Create `<domain_uid>-db1datasource-secret/<domain_uid>-db2datasource-secret<domain_uid>-datasource-secret` with password and url using the `kubectl` commands.

```
kubectl create secret generic <domain_name>-db1datasource-secret
--from-literal=password=<password>
--from-
literal=url='jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TC
P)
HOST=<host1_name>)
(PORT=1521))CONNECT_DATA=(SERVICE_NAME=<service_name>)
(INSTANCE_NAME=<instance_name>)))'
-n <resource_prefix>-domain-ns

kubectl create secret generic <domain_name>-db2datasource-secret
--from-literal=password=<password>
--from-
literal=url='jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TC
P)
HOST=<host2_name)
(PORT=1521))CONNECT_DATA=(SERVICE_NAME=<service_name>)
(INSTANCE_NAME=<instance_name>)))'
-n <resource_prefix>-domain-ns
```

Create an Active GridLink Data Source for a RAC Database

Use the model Yaml templates to create an Active GridLink (AGL) data source for an Oracle Real Application Cluster (RAC) database with WebLogic Suite Edition.

Following is an example of model Yaml template for app deployment with AGL datasource (RAC as infra DB with WebLogic Enterprise Edition) properties:

> **Note:**
>
> The template files are located at `/u01/shared/scripts/pipeline/templates` on the administration host.

```
resources:
  JDBCSystemResource:
    'hellods':
      Target: '@@ENV:RESOURCE_PREFIX@@-cluster'
      JdbcResource:
        DatasourceType: AGL
        JDBCConnectionPoolParams:
          StatementCacheSize: 0
          TestTableName: SQL ISVALID
        JDBCDataSourceParams:
          GlobalTransactionsProtocol: None
          JNDIName: [ jdbc/hellods ]
        JDBCDriverParams:
          DriverName: oracle.jdbc.replay.OracleDataSourceImpl
          URL: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-secret:url@@'
          PasswordEncrypted: '@@SECRET:@@ENV:DOMAIN_UID@@-datasource-
secret:password@@'
          Properties:
            user:
              Value:
        JDBCOracleParams:
          FanEnabled: true
          ActiveGridlink: true
          OnsNodeList:
```

You must set up the database to create a schema user before you create the data source. See Prerequisites to Create a Data Source.

1. Update the `Properties` section in the model Yaml template to set `OnsNodeList`.
   a. Open an SSH connection to a node as the `opc` user.

      ```
      ssh -i <path_to_private_key> opc@<node_public_ip>
      ```

   b. Go to `u01/shared/helper-scripts` location and invoke the following script:

      ```
      ./scan_address.sh
      <db_hostname>-scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:6200
      ```

2. Create the data source secrets in the model YAML file as follows:
   a. Obtain the `tns` connect string using the python script.

      ```
      python /u01/scripts/db/db_util.py get-connect-string
      ```

   b. Create `<domain_uid>-datasource-secret` with password and url using the `kubectl` command.

For example:

```
kubectl create secret generic <domain_name>-datasource-secret --
from-literal=password=<password>
--from-literal=url='jdbc:oracle:thin:@//<db_hostname>-
scan.subnet2ad2phx.paasdevjcsphx.oraclevcn.com:1521/
<service_name>'
-n <resource_prefix>-domain-ns
```

Configure RAC Infra Datasources

If you are using **Model in Image** and created a JRF domain that uses a RAC database as the infra database, the RCU datasources created during domain creation are generic. You must convert the generic datasources to Multi-Data source for WebLogic Enterpise Edition or Active GridLink (AGL) for WebLogic suite edition.

To identify whether a version uses the Model in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

For WebLogic Enterprise Edition:

To use the RCU datasources template in `/u01/scripts/pipeline/templates/ee_generic_to_mds.yaml` file with the helper `ee_rac_node_urls.sh` script, complete the following steps:

1. Go to the location: `/u01/scripts/pipeline/helper-scripts/`
2. Run the `ee_rac_node_urls.sh` script.
   This displays the required URLs. URL1 for node1 and URL2 for node2.
3. In the model template file, replace the placeholders from the script: "`$(URL1)`" with the value of URL1 and "`$(URL2)`".
4. Run the update domain pipeline job by using the model template file.

For WebLogic Suite edition:

To use RCU datasources template in `/u01/scripts/pipeline/templates/suite_generic_to_agl_datasource.yaml` file with the helper `rac_suite_scan_address.sh` script and add scan address or port for OnsNodeList, complete the following steps:

1. Go to the location: `/u01/scripts/pipeline/helper-scripts/`
2. Run the `rac_suite_scan_address.sh` script.
   This displays the scan address of the RAC.
3. In the model template file, replace the placeholders "`$(scan_address)`" with the output from the script.
4. Run the update domain pipeline job by using the model template file.

Create JMS Resources

If you are using **Model in Image**, Java Messaging Service (JMS) resources can be created using the provided templates.

To identify whether a version uses the Model in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

JMS resources can be added to both JRF and non-JRF domains. It is recommended to use JDBC persistent stores for JMS and Transaction stores to be in the database. So, we need to create a datasource and corresponding leasing table for JDBC store for both the JRF and non-JRF domains.

Complete the following steps:

1. Create a table for database leasing:
2. Format of the tablename should be: `<datasourceschemausername>.<tablename>`
   Example:

```
create table mydbuser.mytable
(
SERVER VARCHAR2(255) NOT NULL,
INSTANCE VARCHAR2(255) NOT NULL,
DOMAINNAME VARCHAR2(255) NOT NULL,
CLUSTERNAME VARCHAR2(255) NOT NULL,
TIMEOUT DATE,
PRIMARY KEY (SERVER, DOMAINNAME, CLUSTERNAME)
);
```

3. The tablename has to be specified in the model template yaml for `AutoMigrationTableName`

The sample model template yaml files for are located at `/u01/shared/scripts/pipeline/templates`:

- `jrf_domain_jms_resources.yaml`
- `non_jrf_domain_jms_resources.yaml`

These are basic sample template model yaml files that can be used and has to be modified according to your requirement. These are not the final model template.

> **Note:**
>
> Irrespective of the number of managed servers, increase the initial and max capacity of `mydatasource` to a higher value.

For best practices about creating JMS resources, see Best Practices for JMS Beginners and Advanced Users in *Administering JMS Resources for Oracle WebLogic Server*.

4

Manage a Domain in Oracle WebLogic Server for OKE

Learn how to access the administration console, stop, start, back up, scale, and perform other management operations on your domain by using the tools provided with Oracle WebLogic Server for OKE.

Topics:

About Managing a Domain

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

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

For information about scaling, see Scaling in *Oracle WebLogic Server Kubernetes Operator*.

About the Security Checkup Tool

Oracle WebLogic Server Administration console includes a security checkup tool that displays security check warnings.

In case of Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied, the message `Security warnings`

`detected. Click here to view the report and recommended remedies` is displayed at the top of the Oracle WebLogic Server Administration console. When you click the message, a list of security warnings are displayed as listed in the following table.

The warning messages listed in the table are examples.

Security Warnings

Warning Message	Resolution
`Production mode is enabled but the file or directory <directory_name>/startWebLogic.sh is insecure since its permission is not a minimum of umask 027.`	Run the following command in the administration server as `oracle` user: `chmod 640 /u01/data/domains/<domain_name>/bin`
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Set the java properties for anonymous RMI T3 and IIOP requests during server start up. See Set the Java Properties.

> **Note:**
>
> For existing Oracle WebLogic Server for OKE instances (created before July 20, 2021), you see the SSL host name verification warnings. For details, see Security Checkup Tool Warnings.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OKE instances created after July 20, 2021, though the java properties to disable anonymous requests for preventing anonymous RMI access are configured, the warnings still appear. This is a known issue in Oracle WebLogic Server.

Set the Java Properties

To set the java properties for anonymous RMI T3 and IIOP requests:

1. Edit the `domain.yaml` located in `/u01/shared/weblogic-domains/<domain_name>/domain.yaml` for all instances of `serverPod` definitions as follows:

```
serverPod:
      env:
      - name: USER_MEM_ARGS
        #admin server memory is explicitly set to min of 256m and
```

```
max of 512m and GC algo is G1GC
        value: "-Xms256m -Xmx512m -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
      - name: JAVA_OPTIONS
        value: "-Dweblogic.store.file.LockEnabled=false
        -Dweblogic.rjvm.allowUnknownHost=true
        -Dweblogic.security.remoteAnonymousRMIT3Enabled=false
        -Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

2. Apply the `domain.yaml` using the `kubectl` command:

```
kubectl -f <path_to_domain.yaml>
```

Access the WebLogic Console

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

> **Note:**
>
> Do not use the WebLogic console to make any configuration changed. All configuration changes should be should done through jobs, this ensure that the changes are persistent.
>
> Security check warnings are displayed at the top of the console. See About the Security Checkup Tool for the warnings and how to handle them.

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

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

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
3. Select the **Compartment** in which your domain is created.
4. Click the stack for your domain.
5. Click **Jobs**.
6. In the **Jobs** table, click the Apply job for the stack.
7. Click **Outputs**.

8. Identify and make a note of the public IP address of the bastion compute instance, `bastion_instance_public_ip`.
9. Click **Logs**.
10. Search for the attribute `weblogic_console_url`. Copy the URL.

> **Note:**
>
> If you provision an Oracle WebLogic Server for OKE domain without a bastion instance, you must obtain the internal load balancer IP address, and then use this IP address in the WebLogic Console URL. See Access the Load Balancer IP for No Bastion Host.
>
> For example, if the IP address obtained is `<internal_lb_ip2>`, the WebLogic Console URL format is:
>
> ```
> weblogic_console_url=http://<internal_lb_ip2>/console
> ```

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

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

    Specify the `-D` option to use dynamic port forwarding. Provide the path to the private key that corresponds to the public key that you specified when you created the 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.
12. In your browser settings, set up the SOCKS (version 5) proxy configuration. Specify your local computer and the same SOCKS port that you used in your SSH command.
13. Browse to the console URL.
14. Sign in using the administrator credentials for your domain.

Access the Administration Instance

Access the administration compute instance for a domain that you created with Oracle WebLogic Server for OKE.

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

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

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

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

   Connect as the `opc` user.

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

   The SSH command format is:

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

   For example:

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

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

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

Create a Private Load Balancer for WebLogic Cluster Ingress

After provisioning the stack, you can create a WebLogic cluster ingress load balancer (an internal load balancer).

> **Note:**
>
> If you create a WebLogic cluster ingress load balancer using the steps in the following section, the private and public load balancers created during the Oracle WebLogic Server for OKE provisioning are deleted. So, you must change the URL that is used to access the WebLogic Server Administration Console and the Jenkins console to use the new private ingress load balancer.

To create a WebLogic cluster ingress load balancer:

1. From your computer, create an SSH connection to the administration instance's IP address, and also specify the bastion instance's IP address as a proxy.

   Connect as the `opc` user.

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

   The SSH command format is:

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

2. Uninstall the current ingress controller.

   ```
   helm uninstall ingress-controller
   ```

3. Change to the `/u01/scripts/ingress-controller/templates` directory and open `_nginx-load-balancer.tpl` in text editor.
4. In the `external LB definition`, after `service.beta.kubernetes.io/oci-load-balancer-backend-protocol: "HTTP""`, add the following line:

   ```
   service.beta.kubernetes.io/oci-load-balancer-internal: "true"
   ```

5. Save and exit the text editor.
6. Copy files to the `tmp` folder.

   ```
   cp /u01/provisioning-data/ingress-controller-input-values.yaml /tmp
   ```

7. Verify if the existing load balancers are removed from the Oracle Cloud Infrastructure console.
8. Install the ingress-controller again.

   ```
   /u01/scripts/bootstrap/install_ingress_controller.sh /tmp/ingress-controller-input-values.yaml
   ```

   This command creates two private load balancers, one for admin-ingress, which has a different IP address, and other for cluster-ingress. You can now access the applications deployed to the cluster only through HTTPS and an SSH tunnel through the bastion compute instance. See Access the WebLogic Console and Access the Jenkins Console.

Authenticate by using an External LDAP Server

This section describes the steps required to add authenticators for external systems, like OpenLDAP. This helps you to use the groups and users defined in the system for your applications deployed in the Oracle WebLogic Server for OKE domain. Also, it provides information about how to use an OpenLDAP server.

In Oracle WebLogic Server for OKE, you cannot add users and groups to the WebLogic embedded LDAP server for your applications. As the embedded LDAP has limited use when using the domain home in the image model, if you add users to the embedded LDAP through the Administration console, the users are not persisted in the image and disappear when you restart the admin server pod. Also, it is not recommended to change the domain from the administration console.

Prerequisites

Before you authenticate by using an external LDAP sever, ensure that you have completed the required prerequisites.

1. You must have created a domain with Oracle WebLogic Server for OKE instance. See Create a Domain with Oracle WebLogic Server for OKE.
2. You must have an OpenLDAP server, which is ready to use.
   The OpenLDAP server must be accessible from the Oracle WebLogic Server for OKE nodes, where your WebLogic domain is running and from the admin host. That is, OpenLDAP server must connect to the OpenLDAP host (either by name or IP address) and use the port where the LDAP server is listening.

   Default ports are `389` for LDAP and `636` for LDAPS (LDAP over SSL).

   If the LDAP server is on premises, then connect the VCN where the stack was created with your datacenter, by using either FastConnect or VPN connect. See Access to Your On-Premises Network.

Add a new OpenLDAP Authenticator to the Domain in Model in Image

For **Model in Image**, define the authenticators that you want to add to your WebLogic domain in a WDT model file, and then apply this model by using the `update-domain` pipeline job.

To identify a version that uses the Model in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

1. Create an `OpenLDAP_authenticator.yaml` file.

The following is an example of a model file specifying an OpenLDAP authenticator and helps to connect to an OpenLDAP server by using LDAP protocol, that is with SSL disabled. This model is based on the models presented in Modeling Security Providers.

> **Note:**
>
> You must create a secret for the administrator password, and you use this secret when you run the `update-domain` pipeline job.

Example of an `OpenLDAP_authenticator.yaml` file:

```
topology:
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True

EnableGroupMembershipLookupHierarchyCaching: True
                            Host: myldap.example.com
                            Port: 389
                            UserObjectClass: inetOrgPerson
                            GroupHierarchyCacheTTL: 600
                            SSLEnabled: False
                            UserNameAttribute: cn
                            Principal:
'cn=foo,ou=users,dc=example,dc=com'
                            CredentialEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-secret:password@@'
                            UserBaseDn: 'ou=users,dc=example,dc=com'
                            UserSearchScope: subtree
                            UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                            GroupBaseDN:
'ou=groups,dc=example,dc=com'
                            StaticGroupObjectClass: groupofnames
                            StaticGroupNameAttribute: cn
                            StaticMemberDNAttribute: member
                            StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                            UseRetrievedUserNameAsPrincipal: True
                            KeepAliveEnabled: True
                            GuidAttribute: uuid
                    DefaultAuthenticator:
                        DefaultAuthenticator:
                            ControlFlag: SUFFICIENT
                    DefaultIdentityAsserter:
                        DefaultIdentityAsserter:
```

In order to keep `DefaultAuthenticator` and `DefaultIdentityAsserter` while changing or adding providers, they must be specified in the model with any non-default attributes, as in the example. The order of providers in the model will be the order the providers set in the WebLogic security configuration. See Modeling Security Providers.

2. Run the `update-domain` pipeline job to add the authenticators. See Update the Domain Configuration.

Enable SSL Support in Model in Image

For **Model in Image**, to enable SSL support, you need to perform a few additional steps. Here you need to configure both a trust keystore and an identity keystore, although only trust keystore is required for one-way SSL connection to the LDAP server. You must also configure SSL with the host name verifier.

To identify a version that uses the Model in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

1. Obtain the root Certificate Authority (CA) certificate for the LDAP server.
2. Create a trust keystore by using the preceding certificate or if you already have an existing trust keystore, import the certificate to the trust keystore.
   Following is an example to create the keystore *myTrust.jks* with the root CA certificate `rootca.pem`, by using the `keytool` command:

   ```
   keytool -import -keystore ./myTrust.jks -trustcacerts -alias oidtrust -
   file rootca.pem -storepass TrustKeystorePwd -noprompt
   ```

3. Create an identity keystore, if you do not have an existing identity keystore.
   Following is an example to create the identity keystore:

   ```
   keytool -genkeypair -alias server_cert -keyalg RSA -sigalg SHA256withRSA -
   keysize 2048 -dname
   "CN=example.com,OU=Support,O=Example,L=Reading,ST=Berkshire,C=GB" -
   keystore ./myIdentity.jks
   ```

4. Copy the trust and identity keystores to the `u01` shared location, and specify the location of these files in the `model.yaml` file.
5. Create a `model.yaml` file, specifying the OpenLDP Authentication Provider with SSL enabled, the `DefaultAuthenticator` and `DefaultIdentityAsserter` information, and the custom keystores for admin server and the servers that are part of the dynamic cluster configured in the domain.

> **Note:**
>
> You must create a secret for the administrator password, and provide this secret in the `model.yaml` file; this secret is used when you run the `update-domain` job. For information see, Update the Domain Configuration.
>
> You must use ENV macros for server names as specified in the following `model.yaml` example files for non-JRF anf JRFdomain.

- Following is a sample of the `model.yaml` file for a non-JRF domain:

In this `model.yaml` file, `ServerPrivateKeyAlias`, refers to the alias used when you created the `DemoIdentity` keystore and `ServerPrivateKeyPassPhraseEncrypted` refers to the password set for `ServerPrivateKeyAlias`.

```
topology:
    Server:
        '@@ENV:RESOURCE_PREFIX@@-adminserver':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/
DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/
myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
    ServerTemplate:
        '@@ENV:RESOURCE_PREFIX@@-cluster-template':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/
DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/
myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True

EnableGroupMembershipLookupHierarchyCaching: True
                            Host: 'pg-openldap'
                            Port: 636
                            UserObjectClass: inetOrgPerson
                            GroupHierarchyCacheTTL: 600
                            SSLEnabled: True
```

```
                                UserNameAttribute: cn
                                Principal: 'cn=admin,dc=wlsoketest-
ldap,dc=com'
                                CredentialEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-secret:password@@'
                                UserBaseDn: 'ou=people,dc=wlsoketest-
ldap,dc=com'
                                UserSearchScope: subtree
                                UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                                GroupBaseDN: 'ou=groups,dc=wlsoketest-
ldap,dc=com'
                                StaticGroupObjectClass: groupofnames
                                StaticGroupNameAttribute: cn
                                StaticMemberDNAttribute: member
                                StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                                UseRetrievedUserNameAsPrincipal: True
                                KeepAliveEnabled: True
                                GuidAttribute: entryuuid
                        DefaultAuthenticator:
                            DefaultAuthenticator:
                                ControlFlag: SUFFICIENT
                        DefaultIdentityAsserter:
                            DefaultIdentityAsserter:
```

- Following is a sample of the `model.yaml` file for a JRF domain:

  In this `model.yaml` file, `ServerPrivateKeyAlias`, refers to the alias used when you created the `DemoIdentity` keystore and `ServerPrivateKeyPassPhraseEncrypted` refers to the password set for `ServerPrivateKeyAlias`.

  > **Note:**
  >
  > – If you want to add the LDAP SSL Authenticator to managed servers, then update the model yaml file with the custom identity trust and keystore values to each server in the cluster.
  >
  > – If you scale-out the cluster, then update the domain with the `update-domain` job. You need to provide the yaml file containing the contents, as listed in the sample yaml file, with the name of the scaled-out managed server. However, you do not have to update the `SecurityConfiguration` details.

```
topology:
    Server:
        '@@ENV:RESOURCE_PREFIX@@-adminserver':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/
DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
```

```
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/
myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
        '@@ENV:RESOURCE_PREFIX@@-managed-server1':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/
DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/
myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
        '@@ENV:RESOURCE_PREFIX@@-managed-server2':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: '/u01/shared/
DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demosecret:password@@'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: '/u01/shared/
myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-trustsecret:password@@'
            SSL:
                ServerPrivateKeyAlias: DemoIdentity
                ServerPrivateKeyPassPhraseEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-demokeysecret:password@@'
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True

EnableGroupMembershipLookupHierarchyCaching: True
                            Host: 'pg-openldap'
                            Port: 636
                            UserObjectClass: inetOrgPerson
```

```
                                GroupHierarchyCacheTTL: 600
                                SSLEnabled: True
                                UserNameAttribute: cn
                                Principal: 'cn=admin,dc=wlsoketest-
ldap,dc=com'
                                CredentialEncrypted:
'@@SECRET:@@ENV:DOMAIN_UID@@-ldap-secret:password@@'
                                UserBaseDn: 'ou=people,dc=wlsoketest-
ldap,dc=com'
                                UserSearchScope: subtree
                                UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                                GroupBaseDN: 'ou=groups,dc=wlsoketest-
ldap,dc=com'
                                StaticGroupObjectClass: groupofnames
                                StaticGroupNameAttribute: cn
                                StaticMemberDNAttribute: member
                                StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                                UseRetrievedUserNameAsPrincipal: True
                                KeepAliveEnabled: True
                                GuidAttribute: entryuuid
                        DefaultAuthenticator:
                            DefaultAuthenticator:
                                ControlFlag: SUFFICIENT
                        DefaultIdentityAsserter:
                            DefaultIdentityAsserter:
```

6. Apply the model and archive to the running WebLogic Server domain.
   To run the `update-domain` CI/CD pipeline to update the running domain and add the Authentication Providers and custom keystores, complete the following steps:
   a. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
   b. On the Dashboard page, click **update-domain**.
   c. Click **Build with Parameters**.
   d. Select **Shared File System** from the **Archive_Source** list.
   e. For **Archive_File_Location**, browse to select the archive zip file or specify the path of the zip file on the shared file system.
   f. Select **Shared File System** from the **Domain_Model_Source** list.
   g. For **Model_File_Location**, browse to select the YAML file or specify the path of the YAML on the shared file system.
   h. Select **None** from the **Variable_Source** list.
   i. Select the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).
      If you deselected this check box, you can rollback to the previous image later from the backup.

      The **Rollback_On_Failure** check box is selected by default.
   j. Click **Build** to run the Pipeline job.

Configure SSL with host name verifier

1. In the `model.yaml` file, add the SSL configuration with custom `HostnameVerifier` class:
   - For a non-JRF domain, add the configuration in the admin server and the cluster template.
   - For a JRF domain, add the configuration in the admin sever and managed server.

   Following is a sample `model.yaml` file for a non-JRF domain:

```
topology:
  Server:
    '@@ENV:RESOURCE_PREFIX@@-adminserver':
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
  ServerTemplate:
    '@@ENV:RESOURCE_PREFIX@@-cluster-template':
      ListenPort: 8001
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        ListenPort: 8100
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
```

   Following is a sample `model.yaml` file for a JRF domain with 2 replicas:

```
topology:
  Server:
    '@@ENV:RESOURCE_PREFIX@@-adminserver':
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
      '@@ENV:RESOURCE_PREFIX@@-managed-server1':
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
      '@@ENV:RESOURCE_PREFIX@@-managed-server2':
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
```

```
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
```

> **Note:**
>
> You must augment the sample `model.yaml` based on the number of replicas. For example, in case of a JRF domain with 3 replicas, in the sample `model.yaml` file, you must add `@ENV:RESOURCE_PREFIX@@-managed-server1`, `@ENV:RESOURCE_PREFIX@@-managed-server2`, and `@ENV:RESOURCE_PREFIX@@-managedserver3`.

2. Apply the `model.yaml` to the running WebLogic Server domain. See step 6 in Enable SSL Support in Model in Image.

Add a new OpenLDAP Authenticator to the Domain in Domain in Image

For **Domain in Image**, define the authenticators that you want to add to your WebLogic domain in a WDT model file, and then apply this model by using the `update-domain` pipeline job.

To identify a version that uses the Domain in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

> **Important:**
>
> From Oracle WebLogic Server for OKE release 21.2.2 onwards, Domain in Image is deprecated.

1. Create an `OpenLDAP_authenticator.yaml` file.
The following is an example of a model file specifying an OpenLDAP authenticator and helps to connect to an OpenLDAP server by using LDAP protocol, that is with SSL disabled. This model is based on the models presented in Modeling Security Providers.

> **Note:**
>
> The admin user credentials need to be encrypted. And, you will need the encryption password when you run the `update-domain` pipeline job.

Example of an `OpenLDAP_authenticator.yaml` file:

```
topology:
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
```

```
                                PropagateCauseForLoginException: True

    EnableGroupMembershipLookupHierarchyCaching: True
                                Host: myldap.example.com
                                Port: 389
                                UserObjectClass: inetOrgPerson
                                GroupHierarchyCacheTTL: 600
                                SSLEnabled: False
                                UserNameAttribute: cn
                                Principal:
    'cn=foo,ou=users,dc=example,dc=com'
                                CredentialEncrypted:
    '{AES}<encrypted_credential>'
                                UserBaseDn: 'ou=users,dc=example,dc=com'
                                UserSearchScope: subtree
                                UserFromNameFilter: '(&(cn=%u)
    (objectclass=inetOrgPerson))'
                                GroupBaseDN:
    'ou=groups,dc=example,dc=com'
                                StaticGroupObjectClass: groupofnames
                                StaticGroupNameAttribute: cn
                                StaticMemberDNAttribute: member
                                StaticGroupDNsfromMemberDNFilter:
    '(&(member=%M)(objectclass=groupofnames))'
                                UseRetrievedUserNameAsPrincipal: True
                                KeepAliveEnabled: True
                                GuidAttribute: uuid
                        DefaultAuthenticator:
                            DefaultAuthenticator:
                                ControlFlag: SUFFICIENT
                        DefaultIdentityAsserter:
                            DefaultIdentityAsserter:
```

In order to keep `DefaultAuthenticator` and `DefaultIdentityAsserter` while changing or adding providers, they must be specified in the model with any non-default attributes, as in the example. The order of providers in the model will be the order the providers set in the WebLogic security configuration. See Modeling Security Providers.

2. Run the `update-domain` pipeline job to add the authenticators. See Update the Domain Configuration.

Enable SSL Support in Domain in Image

For **Domain in Image**, to enable SSL support, you need to perform a few additional steps. Here you need to configure both a trust keystore and an identity keystore, although only trust keystore is required for one-way SSL connection to the LDAP server. You must also configure SSL with the host name verifier.

To identify a version that uses the Domain in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

> **Important:**
>
> From Oracle WebLogic Server for OKE release 21.2.2 onwards, Domain in Image is deprecated.

1. Obtain the root Certificate Authority (CA) certificate for the LDAP server.
2. Create a trust keystore by using the preceding certificate or if you already have an existing trust keystore, import the certificate to the trust keystore.
   Following is an example to create the keystore *myTrust.jks* with the root CA certificate `rootca.pem`, by using the `keytool` command:

   ```
   keytool -import -keystore ./myTrust.jks -trustcacerts -alias oidtrust -
   file rootca.pem -storepass TrustKeystorePwd -noprompt
   ```

3. Create an identity keystore, if you do not have an existing identity keystore.
   Following is an example to create the identity keystore:

   ```
   keytool -genkeypair -alias server_cert -keyalg RSA -sigalg SHA256withRSA -
   keysize 2048 -dname
   "CN=example.com,OU=Support,O=Example,L=Reading,ST=Berkshire,C=GB" -
   keystore ./myIdentity.jks
   ```

4. Create an archive file with the trust and identity keystores that will be configured in the admin and managed servers.
   Following is a sample of the contents of the archive file:

   ```
   $ unzip -l archive.zip
   Archive:  archive.zip
     Length      Date    Time    Name
   ---------  ---------- -----   ----
           0  11-25-2020 17:22   wlsdeploy/
           0  11-25-2020 17:18   wlsdeploy/servers/
           0  11-25-2020 19:35   wlsdeploy/servers/myoke-adminserver/
        2337  11-25-2020 00:49   wlsdeploy/servers/myoke-adminserver/
   myIdentity.jks
         853  11-25-2020 00:47   wlsdeploy/servers/myoke-adminserver/
   myTrust.jks
   ---------                     -------
        2190                     5 files
   ```

5. Create a `model.yaml` file, specifying the OpenLDP Authentication Provider with SSL enabled, the `DefaultAuthenticator` and `DefaultIdentityAsserter` information, information, and the custom keystores for admin server and the servers that are part of the dynamic cluster configured in the domain.

> **Note:**
>
> The credentials for the administrator purpose and the passphrase for the keystores have to be encrypted. You need the encryption passphrase when you run the `update-domain` job. For information see, Update the Domain Configuration.

- Following is a sample of the `model.yaml` file for a non-JRF domain:

```
topology:
    Server:
        'myoke-adminserver':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: 'wlsdeploy/servers/
myoke-adminserver/myIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'{AES}<encrypted_passphrase>'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: 'wlsdeploy/servers/
myoke-adminserver/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'{AES}<encrypted_passphrase>'
    ServerTemplate:
        'myoke-cluster-template':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: 'wlsdeploy/servers/
myoke-adminserver/myIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'{AES}<encrypted_passphrase>'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: 'wlsdeploy/servers/
myoke-adminserver/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'{AES}<encrypted_passphrase>'
    SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True

EnableGroupMembershipLookupHierarchyCaching: True
                            Host: myldap.example.com
                            Port: 636
                            UserObjectClass: inetOrgPerson
                            GroupHierarchyCacheTTL: 600
                            SSLEnabled: True
                            UserNameAttribute: cn
                            Principal:
```

```
'cn=foo,ou=users,dc=example,dc=com'
                              CredentialEncrypted:
'{AES}<encrypted_credential>'
                              UserBaseDn: 'ou=users,dc=example,dc=com'
                              UserSearchScope: subtree
                              UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                              GroupBaseDN: 'ou=groups,dc=wlsoketest-
ldap,dc=com'
                              StaticGroupObjectClass: groupofnames
                              StaticGroupNameAttribute: cn
                              StaticMemberDNAttribute: member
                              StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                              UseRetrievedUserNameAsPrincipal: True
                              KeepAliveEnabled: True
                              GuidAttribute: uuid
                        DefaultAuthenticator:
                            DefaultAuthenticator:
                                ControlFlag: SUFFICIENT
                        DefaultIdentityAsserter:
                            DefaultIdentityAsserter:
```

- Following is a sample of the model.yaml file for a JRF domain:

> **Note:**
>
> – If you want to add the LDAP SSL Authenticator to managed servers, then update the model yaml file with the custom identity trust and keystore values to each server in the cluster.
>
> – If you scale-out the cluster, then update the domain with the update-domain job. You need to provide the yaml file containing the contents, as listed in the sample yaml file, with the name of the scaled-out managed server. However, you do not have to update the `SecurityConfiguration` details.

```
topology:
   Server:
      'wlsoke-adminserver':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: 'wlsdeploy/servers/wlsoke-
adminserver/DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'DemoIdentityKeyStorePassPhrase'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: 'wlsdeploy/servers/wlsoke-
adminserver/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted: 'TrustKeystorePwd'
      'wlsoke-managed-server1':
            KeyStores: CustomIdentityAndCustomTrust
```

```
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: 'wlsdeploy/servers/
wlsoke-adminserver/DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'DemoIdentityKeyStorePassPhrase'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: 'wlsdeploy/servers/
wlsoke-adminserver/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'TrustKeystorePwd'
      'wlsoke-managed-server2':
            KeyStores: CustomIdentityAndCustomTrust
            CustomIdentityKeyStoreType: jks
            CustomIdentityKeyStoreFileName: 'wlsdeploy/servers/
wlsoke-adminserver/DemoIdentity.jks'
            CustomIdentityKeyStorePassPhraseEncrypted:
'DemoIdentityKeyStorePassPhrase'
            CustomTrustKeyStoreType: jks
            CustomTrustKeyStoreFileName: 'wlsdeploy/servers/
wlsoke-adminserver/myTrust.jks'
            CustomTrustKeyStorePassPhraseEncrypted:
'TrustKeystorePwd'
   SecurityConfiguration:
        Realm:
            myrealm:
                AuthenticationProvider:
                    My OpenLDAP authenticator:
                        OpenLDAPAuthenticator:
                            ControlFlag: SUFFICIENT
                            PropagateCauseForLoginException: True

EnableGroupMembershipLookupHierarchyCaching: True
                            Host: myldap.example.com
                            Port: 636
                            UserObjectClass: inetOrgPerson
                            GroupHierarchyCacheTTL: 600
                            SSLEnabled: True
                            UserNameAttribute: cn
                            Principal:
'cn=foo,ou=users,dc=example,dc=com'
                            CredentialEncrypted:
'{AES}<encrypted_credential>'
                            UserBaseDn:
'ou=users,dc=example,dc=com'
                            UserSearchScope: subtree
                            UserFromNameFilter: '(&(cn=%u)
(objectclass=inetOrgPerson))'
                            GroupBaseDN:
'ou=groups,dc=wlsoketest-ldap,dc=com'
                            StaticGroupObjectClass: groupofnames
                            StaticGroupNameAttribute: cn
                            StaticMemberDNAttribute: member
                            StaticGroupDNsfromMemberDNFilter:
'(&(member=%M)(objectclass=groupofnames))'
                            UseRetrievedUserNameAsPrincipal: True
```

```
            KeepAliveEnabled: True
            GuidAttribute: uuid
    DefaultAuthenticator:
        DefaultAuthenticator:
            ControlFlag: SUFFICIENT
    DefaultIdentityAsserter:
        DefaultIdentityAsserter:
```

6. Apply the model and archive to the running WebLogic Server domain.
   To run the `update-domain` CI/CD pipeline to update the running domain and add the Authentication Providers and custom keystores, complete the following steps:
   a. Sign in to the Jenkins console for your domain. See Access the Jenkins Console.
   b. On the Dashboard page, click **update-domain**.
   c. Click **Build with Parameters**.
   d. Select **Shared File System** from the **Archive_Source** list.
   e. For **Archive_File_Location**, browse to select the archive zip file or specify the path of the zip file on the shared file system.
   f. Select **Shared File System** from the **Domain_Model_Source** list.
   g. For **Model_File_Location**, browse to select the YAML file or specify the path of the YAML on the shared file system.
   h. Select **None** from the **Variable_Source** list.
   i. For **Encryption_Passphrase**, enter the encrytion passphrase to encrypt the passwords in the model YAML or the variable properties file.
   j. Select the **Rollback_On_Failure** check box if you do not want to rollback to the previous working domain image (optional).
      If you deselected this check box, you can rollback to the previous image later from the backup.

      The **Rollback_On_Failure** check box is selected by default.
   k. Click **Build** to run the Pipeline job.

Configure SSL with host name verifier

1. In the `model.yaml` file, add the SSL configuration with custom `HostnameVerifier` class:
   - For a non-JRF domain, add the configuration in the admin server and the cluster template.
   - For a JRF domain, add the configuration in the admin sever and managed server.

   Following is a sample `model.yaml` file for a non-JRF domain:

```
topology:
  Server:
    '@@ENV:RESOURCE_PREFIX@@-adminserver':
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
  ServerTemplate:
    '@@ENV:RESOURCE_PREFIX@@-cluster-template':
```

```
      ListenPort: 8001
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        ListenPort: 8100
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
```

Following is a sample `model.yaml` file for a JRF domain with 2 replicas:

```
topology:
  Server:
    '@@ENV:RESOURCE_PREFIX@@-adminserver':
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
      '@@ENV:RESOURCE_PREFIX@@-managed-server1':
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
      '@@ENV:RESOURCE_PREFIX@@-managed-server2':
      Cluster: '@@ENV:RESOURCE_PREFIX@@-cluster'
      SSL:
        OutboundCertificateValidation: BuiltinSSLValidationOnly
        HostnameVerifier:
weblogic.security.utils.SSLWLSWildcardHostnameVerifier
        InboundCertificateValidation: BuiltinSSLValidationOnly
```

> **Note:**
>
> You must augment the sample `model.yaml` based on the number of replicas. For example, in case of a JRF domain with 3 replicas, in the sample `model.yaml` file, you must add `@ENV:RESOURCE_PREFIX@@-managed-server1`, `@ENV:RESOURCE_PREFIX@@-managed-server2`, and `@ENV:RESOURCE_PREFIX@@-managedserver3`.

2. Apply the `model.yaml` to the running WebLogic Server domain. See step 6 in Enable SSL Support in Domain in Image.

Verify the Authenticator

Verify that the authentication provider is created successfully and the expected LDAP provider users and groups are synced.

Check the Health of a Domain

Monitor the health of key components of the service in Oracle WebLogic Server for OKE.

The following topic describes how to check the health of such components:

- Check the Health of a Cluster
- Check the Health of a Load Balancer
- Check the Health of a WebLogic Domain

Check the Health of a Cluster

Learn how to view cluster metrics that help you to monitor the health, capacity, and performance of the instance's Kubernetes cluster manged by Oracle WebLogic Server for OKE.

Check the Metrics for Clusters

You can view metrics information for the clusters in your domain.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu and select **Developer Services**. Under the **Containers** group, click **Kubernetes Clusters**.
3. Select the **Compartment** containing the cluster for which you want to view metrics.
4. Click the name of the cluster for which you want to view metrics.
5. Under Resources on the left, click **Metrics**.

   The **Metrics** tab displays a chart for each metric for the cluster that is emitted by the metric namespace. For more information about the displayed metrics, see Available Metrics: oci_oke.

Check the Metrics for Node Pool Clusters

You can view metrics information of the node pools of the instance's Kubernetes cluster and also view the metrics information for each node.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Click the navigation menu and select **Developer Services**. Under the **Containers** group, click **Kubernetes Clusters**.
3. Select the **Compartment** containing the cluster for which you want to view metrics.
4. Click the name of the cluster for which you want to view metrics.
5. Under Resources on the left, click **Node pools**.
6. On the **Node Pools** tab, click the name of a node pool for which you want to see detailed status.
7. Under Resources on the left, click **Metrics**.

   This displays more granular information about the health, capacity, and performance of the node pool.

8. Under Resources on the left, click **Nodes**.

   This displays the summary status of each worker node in the node pool

9. Click View Mertics beside the node to view more granular information about the health, capacity, and performance of that node.

   For more information about the displayed metrics, see Available Metrics: oci_oke.

Check the Health of a Load Balancer

Learn how to view the status of a load balancers associated with the instance's Kubernetes cluster manged by Oracle WebLogic Server for OKE.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.

2. Run the following command:

```
kubectl get services --all-namespaces
```

   Output example:

```
NAMESPACE                 NAME
TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                     AGE
default                   kubernetes
ClusterIP      10.96.0.1       <none>
443/TCP                     27h
ingress-nginx             okename-internal
LoadBalancer   10.96.185.81    100.121.170.271
80:32144/TCP                27h
jenkins-ns                jenkins-service
ClusterIP      10.96.121.100   <none>            8080/
TCP,50000/TCP           27h
kube-system               kube-dns
ClusterIP      10.96.7.5       <none>            53/UDP,53/
TCP,9153/TCP       27h
kube-system               tiller-deploy
ClusterIP      10.96.76.135    <none>
44134/TCP                   27h
okename-domain-ns         mydomain-cluster-okename-cluster
ClusterIP      10.96.143.98    <none>
8001/TCP                    27h
okename-domain-ns         mydomain-okename-adminserver
ClusterIP      None            <none>            30012/
TCP,7001/TCP          27h
okename-domain-ns         mydomain-okename-managed-server1
ClusterIP      None            <none>
8001/TCP                    27h
okename-domain-ns         mydomain-okename-managed-server2
ClusterIP      None            <none>
8001/TCP                    27h
okename-domain-ns         okename-external
LoadBalancer   10.96.162.263   144.25.10.101     80:32148/
TCP,443:31808/TCP   27h
```

```
okename-operator-ns       internal-weblogic-operator-svc
ClusterIP      10.96.92.254    <none>
8082/TCP                   27h
```

Make a note of the IP address of the services with type `LoadBalancer` and in the `EXTERNAL-IP` column.

3. Disconnect from the administration compute instance.
4. Sign in to the Oracle Cloud Infrastructure Console.
5. Click the navigation menu and select **Networking** > **Load Balancers**.
6. Select the **Compartment** that contains your stack.
7. Find the required load balancer by searching with the IP addresses that you noted in Step2.
8. Click the name of the load balance against the IP address you searched for.
9. Under Resources on the left, click **Metrics**.

   The **Metrics** tab displays a default set of charts for the selected load balancer.

Check the Health of a WebLogic Domain

Learn how to view the status of a WebLogic domain manged by Oracle WebLogic Server for OKE.

Each server that is part of the domain runs in a pod. You can run `kubectl` commands to check the status of the pods that are part of the domain.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.
2. Run the following commands:

```
kubectl get pods -n <service>-domain-
ns                                          # list all the pods in the
domain namespace
kubectl describe pod -n <service>-domain-ns <domain>-<service>-
adminserver          # get details of admin server.
kubectl describe pod -n <service>-domain-ns <domain>-<service>-managed-
server1      # get details of managed server 1.
```

Where, <*service*> is the name of the domain and <*domain*> is the domain name.

Output example for command `kubectl get pods -n <service>-domain-ns`:

```
NAME                                READY   STATUS    RESTARTS   AGE
mydomain-nameoke-adminserver        1/1     Running   0          1d4h
mydomain-nameoke-managed-server1    0/1     Running   0          1d4h
mydomain-nameoke-managed-server2    1/1     Running   0          1d4h
```

See the `READY` column to know the status of the respective server.

- `1/1`: servers are running and ready to accept request.

- `0/1`: pod is running, but is not ready to accept request.

Start and Stop Servers

Oracle WebLogic Server for OKE provides utilities to manage the servers in your domain.

With these utilities you can start and stop the admin server and the managed servers in your domain.

> **Note:**
>
> Do not use the WebLogic Server Administration Console to start or stop servers.

1. Identify the following IP address of the node in your domain:
   - The public IP address to the Administration Server node.
   - The public IP address of the bastion and the private IP address of the compute instance.
2. From your computer, run the `ssh` command to connect to the domain's Administration Server node as the `opc` user.

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

3. Run the following command:

   ```
   cd /u01/scripts/wls-domain-lifecycle
   ```

   The scripts to run the lifecycle operations on the WebLogic pods are displayed.
4. Run the following commands:

Command	Result
`sh startServer.sh -h` This help command can be used with all the scripts that are available in `wls-domain-lifecycle`.	Displays the help information that includes the command format and the parameters that can be used with the script.
`sh stopServer.sh -s <server name> -n <namespace> -d <domain uid>`	Stops the managed server
`sh startServer.sh -s <server name> -n <namespace> -d <domain uid>`	Starts the managed server
`sh stopCluster.sh -n <namespace> -d <domain uid> -c <cluster name>`	Stops all the managed servers running in your domain
`sh startCluster.sh -n <namespace> -d <domain uid> -c <cluster name>`	Starts all the managed servers running in your domain
`sh stopDomain.sh -n <namespace> -d <domain uid>`	Stops the admin server and the managed servers running in your domain

Command	Result
`sh startDomain.sh -n <namespace> -d <domain uid>`	Starts the admin server and the managed servers running in your domain

For additional information on the scripts, see the `readme` file in Oracle WebLogic Kubernetes Operator.

There are fields on the Domain that specify which servers should be running, which servers should be stopped, and the desired initial state. You can also modify these fields on the Domain to start and stop servers. See Starting and stopping servers in *Oracle WebLogic Server Kubernetes Operator*.

Scale a WebLogic Cluster

You can change the number of cluster pods or nodes in your Oracle WebLogic Server for OKE stack to increase performance or to reduce costs.

Add pods to scale out, or remove pods to scale in.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.

2. Use `kubectl` to modify the domain.

   ```
   kubectl edit domain <domain_name> -n [namespace]
   ```

   Where, `<domain_name>` is the domain that you want to scale.

   The opens the Domain definition in an editor.

3. Edit the `replicas` value to the desired number of pods.

4. Save and commit the Domain definition.

   You will be notified of the change and the domain immediately scales the corresponding cluster by reconciling the number of running pods with the `replicas` value you specified.

For more information about scaling, see Scaling in *Oracle WebLogic Server Kubernetes Operator*.

Set the JVM Arguments Definition

To explicitly set the Java Virtual Machine (JVM) heap size in the WebLogic Server pods that are created, modify the domain and specify the JVM settings in the domain YAML file.

To set the JVM heap size:

1. Access the administration compute instance for your stack.

   See Access the Administration Instance.

2. Modify the domain using the `kubectl` command.

   ```
   kubectl edit domain <domain_name> -n <domain-namespace> -o yaml
   ```

   This command opens the Domain definition in an editor.

3. Specify the following configuration for the `USER_MEM_ARGS` variable:

```
name: USER_MEM_ARGS
value: '-Xms256M -Xmx512M -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom
-Dweblogic.security.SSL.ignoreHostnameVerification=true '
```

Session Persistence Considerations

You can configure session persistence when deploying Java EE applications to a WebLogic cluster. You must configure session persistence by updating the `weblogic.xml` deployment descriptor's *`session-descriptor`* element, specifically the *`persistent-store-type`* element, whose default value is *`memory`*.

To edit the `weblogic.xml` deployment descriptor's *`session-descriptor`* element, see session-descriptor.

For applications deployed to a cluster, use a value suitable for clustered applications, for example, *`replicated_if_clustered`*. For information, see Using Sessions and Session Persistence.

For ADF applications, see additional considerations at High Availability Checklist for ADF Applications.

Enabling session affinity or sticky sessions at the ingress controller

To send all client requests of a session to the same Oracle WebLogic Server, you must edit the Kubernetes ingress *wls-cluster-ingress* and add session affinity annotations.

To ensure session affinity annotations work, we also need to specify a host in the ingress *wls-cluster-ingress*. You can edit the kubectl to set session affinity.

Run the following command to view the contents of your *wls-cluster-ingress* file:

```
kubectl get ingress -n myoke-domain-ns wls-cluster-ingress -o yaml
```

Sample output:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx-applications
    meta.helm.sh/release-name: ingress-controller
    meta.helm.sh/release-namespace: default
    nginx.ingress.kubernetes.io/configuration-snippet: |
      more_clear_input_headers "WL-Proxy-Client-IP" "WL-Proxy-SSL";
      more_set_input_headers "X-Forwarded-Proto: https";
      more_set_input_headers "WL-Proxy-SSL: true";
  creationTimestamp: "2020-11-30T20:28:48Z"
  generation: 1
  labels:
    app.kubernetes.io/managed-by: Helm
  name: wls-cluster-ingress
```

```
  namespace: myoke-domain-ns
  resourceVersion: "2414741"
  selfLink: /apis/extensions/v1beta1/namespaces/myoke-domain-ns/ingresses/
wls-cluster-ingress
  uid: f5aa919c-7e93-4ca8-a4ca-ddf791c126dd
spec:
  rules:
  - http:
      paths:
      - backend:
          serviceName: mydomain-test-cluster-myoke-cluster
          servicePort: 8001
        path: /
status:
  loadBalancer:
    ingress:
    - ip: <public_ip>
```

Complete the following steps:

1. Run the following command to enable session affinity:

   ```
   kubectl edit ingress -n myoke-domain-ns wls-cluster-ingress
   ```

2. Add the session affinity annotations and the host.

   > **Note:**
   >
   > The minimum set of annotations to add is: `nginx.ingress.kubernetes.io/affinity`

   You can add more annotations to modify the default behavior. For example, you can add `nginx.ingress.kubernetes.io/affinity-mode` and for maximum stickiness set its value to `persistent`, or `nginx.ingress.kubernetes.io/session-cookie-name` to change the default name of the cookie.

   For information about session affinity annotations, see Sticky sessions and Session Affinity.

3. Save the contents.
4. Run the following command to view the contents of your *wls-cluster-ingress* file:

   ```
   kubectl get ingress -n myoke-domain-ns wls-cluster-ingress -o yaml
   ```

   In the following sample output, the text with comments show the modifications of the ingress. In this case, we enabled session affinity and configured the expiration time for the session cookie:

   ```
   [opc@myoke-admin templates]$ kubectl get ingress -n myoke-domain-ns   wls-
   cluster-ingress -o yaml
   apiVersion: extensions/v1beta1
   kind: Ingress
   ```

```
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx-applications
    meta.helm.sh/release-name: ingress-controller
    meta.helm.sh/release-namespace: default
    nginx.ingress.kubernetes.io/affinity: cookie  # New annotation
    nginx.ingress.kubernetes.io/configuration-snippet: |
      more_clear_input_headers "WL-Proxy-Client-IP" "WL-Proxy-SSL";
      more_set_input_headers "X-Forwarded-Proto: https";
      more_set_input_headers "WL-Proxy-SSL: true";
    nginx.ingress.kubernetes.io/session-cookie-expires: "172800"
#New annotation
    nginx.ingress.kubernetes.io/session-cookie-max-age: "172800"
#New annotation
  creationTimestamp: "2020-11-30T20:28:48Z"
  generation: 2
  labels:
    app.kubernetes.io/managed-by: Helm
  name: wls-cluster-ingress
  namespace: myoke-domain-ns
  resourceVersion: "2419743"
  selfLink: /apis/extensions/v1beta1/namespaces/myoke-domain-ns/
ingresses/wls-cluster-ingress
  uid: f5aa919c-7e93-4ca8-a4ca-ddf791c126dd
spec:
  rules:
  - host: my-server   #New host
    http:
      paths:
      - backend:
          serviceName: mydomain-test-cluster-myoke-cluster
          servicePort: 8001
        path: /
status:
  loadBalancer:
    ingress:
    - ip: <public_ip>
```

> **Note:**
>
> If you set the host name in the ingress, you will not be able to access your applications by using the load balancer public IP. To access your applications use the host name you specified in the ingress. That is, you need to add the host name to your DNS servers, or manually map the host name to the public IP of the load balancer. For example, by editing the `/etc/hosts` file.

Back Up and Restore a Domain

Oracle WebLogic Server for OKE provides the ability to backup and restore your domain.

Back up a domain:

- By default, every time you run a job, a back up of the domain is created automatically.
- There is no job available to periodically back up your domain.
- You can also manually take a backup by copying the current `yaml` file to your preferred location. A `yaml` file has information about the current setup of the domain.

The backups are location at: `/u01/shared/weblogic-domains/<domain_name>/backups`

A sample backup:

```
[opc@oracle-admin oracledomain]$ pwd
/u01/shared/weblogic-domains/oracledomain
[opc@oracle-admin oracledomain]$ ll -tR
.:
total 25
-rwxr-xr-x. 1 opc opc 3728 Sep 24 16:14 domain.yaml
drwxr-xr-x. 3 opc opc 1 Sep 24 16:14 backups
-rw-rw-r--. 1 opc opc 4642 Sep 23 17:16 provisioning_metadata.json
-rw-rw-r--. 1 opc opc 1495 Sep 23 16:55 create-domain-inputs.yaml

./backups:
total 1
drwxr-xr-x. 2 opc opc 2 Sep 24 16:14 20-09-24_15-57-44

./backups/20-09-24_15-57-44:
total 16
-rw-r--r--. 1 opc opc 3728 Sep 24 16:14 domain.yaml
-rw-r--r--. 1 opc opc 3725 Sep 24 16:14 prev-domain.yaml
```

Where:

- `prev-domain.yaml`, is the previous domain that was running before the current job was completed.
- `domain.yaml`, is the existing domain after the current job was completed.

Restore a domain:

1. Access the administration compute instance for your domain.
   See Access the Administration Instance.
2. Go to the backup location, where the domain `yaml` that you want to apply is located.
3. Open the domain `yaml` and make a note of the image id.
   For example:

   ```
   image: "phx.ocir.io/ax8cfrmecktw/oracle/oracle_domain/wls-domain-
   base:12.2.1.4.200714-200819-20-09-23_14-51-16"
   ```

4. Run the following commands to set the required environment variables:

   ```
   export wls_domain_namespace=<domain namespace>
   export wls_domain_uid=<domain UID>
   export ocir_url=<region>.ocir.io
   ```

5. Run the following command:

   ```
   /u01/shared/scripts/pipeline/common/pipeline_common.sh -i <image_id>
   ```

   Where, <*image_id*> is the image id you noted in step 2.

   For example:

   ```
   /u01/shared/scripts/pipeline/common/pipeline_common.sh -i
   phx.ocir.io/ax8cfrmecktw/oracle/oracle_domain/wls-domain-
   base:12.2.1.4.200714-200819-20-09-23_14-51-16
   ```

Delete the Resources and Stack

You can delete the resources and stack your created for Oracle WebLogic Server for OKE.

1. Access the administration compute instance for your domain.

   See Access the Administration Instance.

2. Run the following command:

   ```
   /u01/shared/scripts/lcm/delete_rcu.sh -d <DB Admin Password>
   ```

   This drops the Repository Configuration Utility (RCU) schema repositories from the database for a JRF-Enabled domain.

3. Run the following command to delete the resources:

   ```
   /u01/shared/scripts/lcm/delete_resources.sh -p <OCIR Auth Token> -l
   ```

   This deletes the OCIR repos created during provisioning and the OCI Load Balancer associated with the internal and external ingress services.

4. Click the navigation menu  , select **Developer Services**. Under the **Resource Manager** group, click **Stacks**.
5. Click the stack you want to delete.
6. Click **Terraform Actions**, and then select **Destroy**.
7. When prompted for confirmation, click **Destroy**.
8. Periodically monitor the progress of the Destroy job until it is finished.

   If an email address is associated with your user profile, you will receive an email notification.

   Ensure that all the resources of the stack are deleted successfully.

9. Click **Delete Stack**.

Delete the Identity Cloud Service Resources

If the Oracle WebLogic Server for OKE 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.

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...
   ```

5

Troubleshoot and Known Issues in Oracle WebLogic Server for OKE

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

To identify whether a version uses the Model in Image or the Domain in Image source type, see the *Supported Image* column in Patches Included in Oracle WebLogic Server for OKE.

> **Important:**
>
> From Oracle WebLogic Server for OKE release 21.2.2 onwards, Domain in Image is deprecated.

Topics

Troubleshoot and Known Issues in Model In Image

Following are the common problems in Oracle WebLogic Server for OKE Model in Image. Learn how to diagnose and solve them.

Topics

Free-Tier Autonomous Database

Free-Tier autonomous database is not supported.

Introspection Failed during Initial Provisioning

When you are running Pipeline jobs the initial provisioning fails, this might be because the introspector is in `ERROR` or `FAILED` state.

Issue: Initial provisioning fails.

Following is an example of the error message:

```
<May 01, 2021 04:44:18 PM GMT> <ERROR> <apply_domain.py> <(host:wrong-
admin.okemb1admin.vcnmb.oraclevcn.com)
- <WLSOKE-VM-ERROR-0093> : Introspector job is in failed state in
weblogic domain namespace [wrong-domain-ns]. [exit_code : -1]. Please
check the introspector logs at location /u01/shared/logs.
<May 01, 2021 04:44:18 PM GMT> <ERROR> <markers.py> <(host:wrong-
admin.okemb1admin.vcnmb.oraclevcn.com)
<May 01, 2021 16:44:18 PM GMT> - <WLS-OKE-ERROR-014> - Failed to apply
domain changes to OKE cluster.[exit_code : -1]
```

Fix:

The typical root cause is invalid weblogic password. as specified in `introspector_script.out` located at `/u01/shared/logs`.

Run the following command:

```
tail -15 introspector_script.out
```

Sample output:

```
<Mar 19, 2021 8:09:53 PM> <SEVERE> <create> <main> <WLSDPLY-12409>
<createDomain failed to create the domain: Failed to set attribute
Password in path /Security/domainmb/User/weblogic to value <masked>:
60455: Invalid password.
60455: The password must be at least 8 alphanumeric characters with at
least one number or special character.
60455: Correct the password.>Issue Log for createDomain version 1.9.8
running WebLogic version 12.2.1.4.0 offline mode:SEVERE Messages: 1.
WLSDPLY-12409: createDomain failed to create the domain: Failed to set
attribute Password in path /Security/domainmb/User/weblogic to value
<masked>: 60455: Invalid password.
60455: The password must be at least 8 alphanumeric characters with at
least one number or special character.
60455: Correct the password.Total: WARNING : 0 SEVERE :
1createDomain.sh failed (exit code = 2)
```

Update the weblogic password.

Introspection Failed when Running Pipeline Jobs

In some instances, the Kubernetes job (`DOMAIN_UID-introspector`) created for the introspection fails. When the initial introspection fails, the operator does not start any WebLogic Server instances. If there are already WebLogic Server instances running, then a failed introspection leaves the existing WebLogic Server instances running without making any changes to the operational state of the domain. The introspection is periodically retried and then eventually timeout with the Domain status indicating the processing failed. To recover from a failed state, you need correct the underlying problem and update the `introspectVersion` or `restartVersion`.

Check the introspector job

If your introspector job failed, then examine the `kubectl describe` of the job and its pod. Also, examine its log, located at `/u01/shared/weblogic-domains/<domain-name>/logs/introspector_script.out`.

For example, assuming your domain UID is `sample-domain1` and your domain namespace is `sample-domain1-ns`, following is a failed introspector job pod among the domain's pods:

```
$ kubectl -n sample-domain1-ns get pods -l weblogic.domainUID=sample-domain1

NAME                                        READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server                 1/1     Running   0          19h
sample-domain1-introspector-v2l7k           0/1     Error     0          75m
sample-domain1-managed-server1              1/1     Running   0          19h
sample-domain1-managed-server2              1/1     Running   0          19h
```

Let us look at the job's describe:

```
$ kubectl -n sample-domain1-ns describe job/sample-domain1-introspector
```

Now, let us look at the job's pod describe, in particular look at its events:

```
$ kubectl -n sample-domain1-ns describe pod/sample-domain1-introspector-v2l7k
```

Finally, let us look at the job's pod's log:

```
$ kubectl -n sample-domain1-ns logs job/sample-domain1-introspector
```

Alternative log command (will have same output as previous):

```
$ kubectl -n sample-domain1-ns logs pod/sample-domain1-introspector-v2l7k
```

A common reason for the introspector job to fail is because of an error in a model file. Following is a sample log output from an introspector job that displays such a failure:

```
...
SEVERE Messages:
    1. WLSDPLY-05007: Model file /u01/wdt/models/model1.yaml,/weblogic-
operator/wdt-config-map/..2020_03_19_15_43_05.993607882/datasource.yaml
contains an
unrecognized section: TYPOresources. The recognized sections are domainInfo,
topology, resources, appDeployments, kubernetes
```

Initiating Rolling Restart

If a model file error references a model file in your `spec.configuration.model.configMap` file, then you can correct the error by redeploying the ConfigMap with a corrected model file and then initiating a domain restart or roll. Similarly, if a model file error references a model file in your model image, then you can correct the error by deploying a corrected image, modifying your Domain YAML file to reference the new image under `spec.image`, and then initiating a domain restart or roll.

To continue to use the pipeline jobs to update the running domain, we need to ensure that the introspector is in `Success` status, which can be achieved by rolling the domain to the previous successful image.

To rollback to the previous previous successful image, run the following command:

```
/u01/shared/scripts/pipeline/common/pipeline_common.sh -i <image_name>
```

Where, *<image_name>* is the image ID in the `prev-domain.yaml` file, located in the backup directory at `/u01/shared/weblogic-domains/<domain_name>/backups`.

> **Note:**
>
> `prev-domain.yaml` is the previous domain that was running before the current job completed.

As the introspector was in the failure status, the domain pods did not restart and would be in the previous image. Once the above function is invoked, introspector succeeds and the pipeline jobs can be reused.

If the error is due to `configmap`, initiate the rolling restart by completing the following steps:

1. Rectify the error in the yaml file.
2. Increment the value of `spec.restartVersion`.
   a. Perform a `kubectl edit domain -n <domain_ns> -o yaml`.
      This opens the yaml file in the VI editor.
   b. Under spec, search for the `restartVersion` flag and increment the value.
3. Save the yaml file.

Run the following command to verify the fix:

```
kubectl get pods -A
```

The age for the pod must not correspond to the time when the update-domain job completed.

Sample output:

```
NAME                                             READY   STATUS
RESTARTS   AGE
sample-domain1-admin-server                      1/1     Running
0          19h
```

```
sample-domain1-managed-server1              1/1     Running   0          19h
sample-domain1-managed-server2              1/1     Running   0          19h
```

Introspector is not Launching

In update-domain pipeline job, where a Model in Image was created to add additional domain configuration, the introspector job does not start.

Issue: The introspector job does not start when the domain resource is updated with the new image. This results in the WebLogic pods not initiating the rolling restart. In this scenario, the domain configuration did not take effect as the WebLogic pods are still using the old domain configuration and relevant debug information is not available in the operator logs.

Workaround: Run the Destroy job on the stack and reapply the job to recreate the required resources.

Data Source Deployed on Server and Cluster

This section covers the known issue when you create data sources in your Oracle WebLogic Server for OKE domain.

If the user adds a new data source and deploys the data source to a cluster only, by default, the data source is deployed to both the managed server and the administration server in the cluster.

WebLogic Server Pods are still in Starting State

After you create a JRF-enabled domain, the WebLogic Server pods are still be in `STARTING` state.

Issue: Immediately after you create a JRF-enabled domain, you might want to launch a CI/CD pipeline job. However, the WebLogic Server pods are still in `STARTING` state.

Workaround:

Complete the following steps:

1. Log in to the Admin VM and check the introspector logs for any errors.
   The introspector log is located at: `/u01/shared/logs/introspector_script.out`
2. Run the following command:
   `kubectl get pods -A`

   Sample output:

```
NAMESPACE            NAME                                        READY
STATUS    RESTARTS   AGE
name-domain1         sample-domain1-admin-server                 1/1
Running   0          19h
name-domain1         sample-domain1-managed-server1              1/1
Running   0          19h
name-domain1         sample-domain1-managed-server2              1/1
Running   0          19h
```

3. If the WebLogic pods are in `RUNNING` status with `READY` state of `1/1`, then you can launch the CI/CD pipeline jobs.

Handling NFS Locking Errors

By default, the WebLogic stores are mount to the shared file system, which use Network File System (NFS) version 3 and is disabled. Therefore, the file locks on the different WebLogic stores and may not release if the VM of any node pool in the WebLogic Node pool is abruptly shut down. This is encountered in different scenarios, like, when a VM is stopped, restarted, or terminated, and there are WebLogic pods assigned to the worker node that is being terminated.

Issue: The WebLogic Server Pod (Admin Server or any manged server) fails to start and displays the following error in the WebLogic logs:

```
[Store:280105]The persistent file store "_WLS_myinstance-admin-server"
cannot open file _WLS_<instanceName>-<ServerName>000000.DAT.
```

Workaround:

To solve this issue, complete the following steps:

> **Note:**
>
> Even if you are using an earlier version of WebLogic Server you need to complete these steps.

1. Apply patch 32471832 by using the opatch update job, which is available in July 2021 PSUs.
2. For administration and managed server pods in the cluster, update the `domain.yaml` file by adding the `Dweblogic.store.file.LockEnabled=false` parameter.
   Following is an example, where the `Dweblogic.store.file.LockEnabled=false` parameter is added:

```
serverPod:
  env:
  - name: USER_MEM_ARGS
    #Default to G1GC algo
    value: "-XX:+UseContainerSupport -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
  - name: JAVA_OPTIONS
    value: "-Dweblogic.store.file.LockEnabled=false -
Dweblogic.rjvm.allowUnknownHost=true -
Dweblogic.security.SSL.ignoreHostnameVerification=true -
Dweblogic.security.remoteAnonymousRMIT3Enabled=false -
Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

3. Run the following command to apply `domain.yaml`.

```
kubectl -f <domain.yaml-file-path>
```

> **Note:**
>
> If you have created Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied, a few Security warnings are displayed. See About the Security Checkup Tool.

Unable to Access the Console or the Application

Troubleshoot problems accessing the console or the application after the Oracle WebLogic Server for OKE domain is successfully created.

Error accessing the console or the application

If you receive 502 bad gateway error when accessing the Jenkins console and WebLogic Server console, or the application using load balancer, use the `kubectl` command to get the node ports that are used by the system and ensure that these node ports are open for access via the load balancer subnet.

For example:

```
kubectl describe service --all-namespaces | grep -i nodeport
NodePort: http 32062/TCP
NodePort: https 30305/TCP
```

To check port access:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Virtual Cloud Networks**.
3. Select the compartment in which you created the domain.
4. Select the virtual cloud network in which the domain was created.
5. Select the subnet where the WebLogic Server compute instance is provisioned.
6. Select the security list assigned to this subnet.
7. For an Oracle WebLogic Server for OKE cluster using a private and public subnet, make sure the following ingress rules exist:

```
Source: <LB Subnet CIDR>
IP Protocol: TCP
Source Port Range: All
Destination Port Range: 32062


Source: <LB Subnet CIDR>
IP Protocol: TCP
```

```
Source Port Range: All
Destination Port Range: 30305
```

For a domain on a private and public subnet, set the `Source` to the CIDR of the load balancer subnet.

Stack Creation Failed

Troubleshoot a failed Oracle WebLogic Server domain that you created using Oracle WebLogic Server for OKE.

Failed to install WebLogic Operator

Stack provisioning might fail when you create a domain with Oracle WebLogic Server for OKE in a new subnet for an existing VCN due to error in installation of WebLogic Server Kubernetes Operator.

Example message:

```
module.provisioner.null_resource.check_provisioning_status_1  (remote-
exec):
<Aug 27, 2020 07:01:31 PM GMT> <INFO>  <install_wls_operator.sh>
<(host:wrjrf8-admin.wrjrf8admin.existingnetwork.oraclevcn.com) -
<WLSOKE-VM-INFO-0020> :
Installing weblogic operator in namespace [wrjrf8-operator-ns]>
module.provisioner.null_resource.check_provisioning_status_1  (remote-
exec): <Aug 27, 2020
07:02:12 PM GMT> <ERROR>  <install_wls_operator.sh>
<(host:wrjrf8-admin.wrjrf8admin.existingnetwork.oraclevcn.com) -
<WLSOKE-VM-ERROR-0013> : Error
installing weblogic operator. Exit  code[1]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to create service account

Stack provisioning might fail with `HTTP 409` conflict error if the service account creation fails.

Example message:

```
module.provisioner.null_resource.check_provisioning_status_1 (remote-
exec):
HTTP response body: {"kind":"Status","apiVersion":"v1","metadata":
{},"status":"Failure","message":
"Operation cannot be fulfilled on serviceaccounts \"default\": the
object has been modified;
please apply your changes to the latest version and try
again","reason":"Conflict","details":
{"name":"default","kind":"serviceaccounts"}

,"code":409}
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to login to OCIR

Stack provisioning might fail if the docker login to OCI registry is not succcesful.

Example message:

```
[phx.ocir.io]>module.provisioner.null_resource.check_provisioning_status_1
(remote-exec):
<Sep 22, 2020 02:33:46 PM GMT> <ERROR> <docker_init.sh> <(host:wrfinal2-
admin.admin.existingnetwork.oraclevcn.com)
- <WLSOKE-VM-ERROR-0003> : Unable to login to custom OCIR
[phx.ocir.io]>module.provisioner.null_resource.check_provisioning_status_1
(remote-exec):
]>module.provisioner.null_resource.check_provisioning_status_1 (remote-exec):
<Sep 22, 2020 02:33:46 PM GMT> <ERROR> <docker_init.py> <(host:wrfinal2-
admin.admin.existingnetwork.oraclevcn.com)
- <WLSOKE-VM-ERROR-0020> : Error executing sh /u01/scripts/bootstrap/
docker_init.sh. Exit code [1]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to download ATP wallet

Stack provisioning might fail if you create a JRF-enabled domain running WebLogic Server 12c and using an Oracle Autonomous Database.

Example message in apply log:

```
module.provisioner.null_resource.check_provisioning_status_1 (remote-exec):
<Sep 22, 2020 12:31:11 PM GMT> <ERROR> <markers.py> <(host:wrfinal2-
admin.admin.existingnetwork.oraclevcn.com
- <Sep 22, 2020 12:31:11> - <WLS-OKE-ERROR-003> - Failed to verify oke
cluster nodes status.
[Exit code : {'status': 500, 'message': u'An internal server error has
occurred.',
'code': u'InternalServerError', 'opc-request-id':
'768603269A9D460D9B979632FC04C181/37A72EDA76A2687A5E24499AA6A70F9B/
7823A7DF9CDD435D869F3CB42C46B39E'}]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to verify OKE cluster node status

Stack provisioning fails if the OKE cluster worker nodes are inactive when you create the WebLogic domain with Oracle WebLogic Server for OKE.

Example message:

```
<INFO> <oke_worker_status.py>
<(host:wlsatpte-admin.nevcnokeadmin.nevcnokevcn.oraclevcn.com) - <WLSOKE-VM-
INFO-0011> : Waiting
```

```
for the workers nodes to be Active. Retrying...><Dec 17, 2020 04:47:56
PM GMT> <ERROR>
<markers.py> <(host:wlsatpte-
admin.nevcnokeadmin.nevcnokevcn.oraclevcn.com) - <Dec 17, 2020
16:47:56> - <WLS-OKE-ERROR-003> - Failed to verify oke cluster nodes
status. [Exit code : Status
check timed out]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Load Balancer Creation Failed

After provisioning a stack, you might encounter an issue where the internal Load Balancer (LB) is missing.

When you run the following command, the external IP for the LB would is displayed as `<pending>`:

```
kubectl get svc -n ingress-nginx
```

In this case, the IP allocation for the LB fails and the LB instance is not created. This is because the quota for the selected LB shape is not available.

Reinstall Load Balancers for Jenkins

When provisioning, creating a load balancer can fail for different reasons. However, provisioning does not stop as the load balancers can be created later. Follow the steps in this section to recreate the load balancers.

When you create a Oracle WebLogic Server for OKE instance, two load balancers are created. One with a public IP, that provides access to the applications installed in the WebLogic cluster, and another with a private IP, to provide access to the WebLogic console and Jenkins console.

Following are the reasons load balancer creation fails during provisioning:

1. Lack of quota for the selected LB shapes.
2. Lack of available public IPs (for external load balancer) or private IPs (for internal load balancer) in the VCN or subnets selected during provisioning.

Check the Status of the Load Balancers

You can view the status of the load balancers by checking the Resource Manager job log, the load balancer services, and the provisioning logs.

Resource Manager job log: When both the load balancers are created successfully, the resource manager job log includes the following:

```
module.provisioner.null_resource.check_provisioning_status_3 (remote-
exec): {
module.provisioner.null_resource.check_provisioning_status_3 (remote-
exec): "weblogic_console_url": "http://<IP_adressess>/console",
module.provisioner.null_resource.check_provisioning_status_3 (remote-
```

```
exec): "jenkins_console_url": "http://<IP_adressess>/jenkins",
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec):
"weblogic_cluster_lb_url": "https://<IP_adressess>/<application context>"
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec): }
```

The first two lines are for the private load balancer. The third line is for the public load balancer. If any of the load balancers are not created, you will not see any of the above lines in the Resource Manager job log.

Load Balancer Services:

To check the load balancer services, run the following command:

```
kubectl get svc -n ingress-nginx
```

If the output lists any of the load balancer services as `<pending>`, under the `EXTERNAL-IP` column, then the load balancers are not created.

Sample output:

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
rsh3oke-external    LoadBalancer   10.1.1.1     <pending>
443:30618/TCP    11m
rsh3oke-internal    LoadBalancer   10.1.1.2    100.1.1.1    80:30790/TCP     11m
```

Provisioning logs:

If the internal or external load balancer is *not* created successfully, the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error message:

```
<WLSOKE-VM-INFO-0058> : Installing ingress controller charts for jenkins
[ ingress-controller ]>
<WLSOKE-VM-ERROR-0058> : Error installing ingress controller with Helm. Exit
code [1]>
```

And, in the `/u01/logs/provisioning_cmd.out` file, you would see the following error message:

```
<install_ingress_controller.sh>  -  Error: timed out waiting for the
condition
```

Reinstall the Load Balancers

After identifying and fixing the cause of the failure, like increased quota for the selected LB shape, you can reinstall the load balancers in the instance.

1. Run the following command to get the values required to install the ingress-controller:

   ```
   helm get values ingress-controller -o yaml > ingress_values.yaml
   ```

2. Run the following command to remove the existing helm release:

```
helm uninstall ingress-controller
```

> **Note:**
>
> This command will delete both the external and internal load balancers.

3. Run the following command to install both the external and internal load balancers:

```
/u01/scripts/bootstrap/install_ingress_controller.sh
ingress_values.yaml
```

Sample Output:

```
<Nov 20, 2020 08:01:01 PM GMT> <INFO>
<install_ingress_controller.sh> <(host:host_name) - <WLSOKE-VM-
INFO-0058> : Installing ingress controller charts for jenkins
[ ingress-controller ]>
<Nov 20, 2020 08:03:27 PM GMT> <INFO>
<install_ingress_controller.sh> <(host:host_name) - <WLSOKE-VM-
INFO-0059> : Successfully installed ingress controller>
```

4. Run the following command to verify if load balancer services are created and have external IP addresses:

```
kubectl get svc -n ingress-nginx
```

Sample output:

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
rsh3oke-external    LoadBalancer   10.96.115.139   144.25.19.38
443:31162/TCP    12m
rsh3oke-internal    LoadBalancer   10.96.249.188   100.111.191.133
80:30605/TCP     12m
```

Install Jenkins Manually

When you create a Oracle WebLogic Server for OKE instance, Jenkins is installed by installing a Helm release called *jenkins-oke*. During provisioning, Jenkins installation may fail, but provisioning is not stopped, because Jenkins can be installed after provisioning. This section explains how to install Jenkins manually, if Jenkins installation has failed during provisioning.

Check if Jenkins Install Failed during Provisioning

You can know if the Jenkins install failed by trying to access the Jenkins console, checking the provisioning logs, and checking the Kubernetes resources (pods, services, and so on) under the `jenkins-ns` namespace.

Access the Jenkins console:

Try accessing the Jenkins console, as described in Access the Jenkins Console.

If you are not able to access the console, then continue to the next section to check the logs.

Provisioning logs:

If Jenkins is *not* installed successfully, then the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error:

```
<WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<WLSOKE-VM-ERROR-0052> : Error installing jenkins charts. Exit code[1]>
```

And, you would see the details of the failure in the `/u01/logs/provisioning_cmd.out` file.

Kubernetes resources:

To check the Kubernetes resources in the `jenkins-ns` namespace, run the following command:

```
kubectl get all -n jenkins-ns
```

Following is a sample output, where Jenkins was installed correctly:

```
NAME                                      READY   STATUS    RESTARTS   AGE
pod/jenkins-deployment-5bb55586b9-vn8sk   1/1     Running   0          26m

NAME                         TYPE        CLUSTER-IP    EXTERNAL-IP
PORT(S)              AGE
service/jenkins-service   ClusterIP   10.96.149.6   <none>        8080/
TCP,50000/TCP   26m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/jenkins-deployment   1/1     1            1           26m

NAME                                             DESIRED   CURRENT   READY
AGE
replicaset.apps/jenkins-deployment-5bb55586b9   1         1         1
26m
```

Install Jenkins Manually

After identifying and fixing the cause of the failure, install Jenkins in your instance.

1. Check if the `provisioning_metadata.properties` file exists, at the `/u01/shared/weblogic-domains/<domain>` directory.
   Does the `provisioning_metadata.properties` file exist?
   - **Yes**: Continue with the next step.
   - **No**: Run the following command:

     ```
     python /u01/scripts/metadata/provisioning_metadata.py
     ```

     Continue with the next step.
2. Run the following command to remove the existing helm release:

   ```
   helm uninstall jenkins-oke
   ```

3. Run the following command to install Jenkins:

   ```
   /u01/scripts/bootstrap/install_jenkins.sh  /u01/provisioning-data/jenkins-inputs.yaml
   ```

   Where, `jenkins-inputs.yaml` file contains the required variables.

   Sample Output:

   ```
   <Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.py>
   <(host:host_name) - updated /u01/provisioning-data/jenkins-inputs.yaml>
   <Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.sh>
   <(host:host_name) - <WLSOKE-VM-INFO-0098> : Creating configmap [wlsoke-metadata-configmap]>
   <Nov 23, 2020 05:10:09 PM GMT> <INFO> <install_jenkins.sh>
   <(host:host_name) - <WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
   <Nov 23, 2020 05:10:22 PM GMT> <INFO> <install_jenkins.sh>
   <(host:host_name) - <WLSOKE-VM-INFO-0057> : Successfully installed jenkins in namespace [ jenkins-ns ]>
   ```

You have successfully installed the Jenkins console. Try accessing the Jenkins console, as described in Access the Jenkins Console.

Security Checkup Tool Warnings

Learn about the security check warnings that are displayed in the Oracle WebLogic Server Administration console and how to troubleshoot them.

At the top of the WebLogic Server Administration console, the message `Security warnings detected. Click here to view the report and recommended remedies` is displayed for Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied.

When you click the message, a list of security warnings are displayed as listed in the following table.

The warning messages listed in the table are examples.

Security Warnings

Warning Message	Resolution
`SSL hostname verification is disabled by the SSL configuration.`	Review your applications before you make any changes to address these SSL host name security warnings. For applications that connect to SSL endpoints with a host name in the certificate, which does not match the local machine's host name, the connection fails if you configure the BEA host name verifier in Oracle WebLogic Server. For applications that connect to Oracle provided endpoints such as Oracle Identity Cloud Service (for example,`*.identity.oraclecloud.com`), the connection fails if you did not configure the wildcard host name verifier or a custom host name verifier that accepts wildcard host names. If you are not sure of the SSL configuration settings you should configure to address the warning, Oracle recommends that you configure the wildcard host name verifier. You see the SSL host name verification warnings in case of existing Oracle WebLogic Server for OKE instances (created before July 20, 2021). To address this warning, you must configure SSL with host name verifier. See Configure SSL with host name verifier.
`Production mode is enabled but the file or directory <directory_name>/ startWebLogic.sh is insecure since its permission is not a minimum of umask 027`	Run the following command in the administration server as `oracle` user: `chmod 640 /u01/data/domains/<domain_name>/bin`
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Set the java properties for anonymous RMI T3 and IIOP requests during server start up. See Set the Java Properties.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OKE instances created after July 20, 2021, though the java properties to disable anonymous requests for preventing anonymous RMI access are configured, the warnings still appear. This is a known issue in Oracle WebLogic Server.

Set the Java Properties

To set the java properties for anonymous RMI T3 and IIOP requests:

1. Edit the `domain.yaml` located in `/u01/shared/weblogic-domains/<domain_name>/domain.yaml` for all instances of `serverPod` definitions as follows:

```
serverPod:
      env:
      - name: USER_MEM_ARGS
        #admin server memory is explicitly set to min of 256m and
max of 512m and GC algo is G1GC
        value: "-Xms256m -Xmx512m -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
      - name: JAVA_OPTIONS
        value: "-Dweblogic.store.file.LockEnabled=false
        -Dweblogic.rjvm.allowUnknownHost=true
        -Dweblogic.security.remoteAnonymousRMIT3Enabled=false
        -Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

2. Apply the `domain.yaml` using the `kubectl` command:

```
kubectl -f <path_to_domain.yaml>
```

Get Additional Help

Use online help, email, customer support, and other tools if you have questions or problems with Oracle WebLogic Server for OKE.

For general help with Oracle Cloud Marketplace, see How Do I Get Support in *Using Oracle Cloud Marketplace*.

Troubleshoot and Known Issues in Domain in Image

Following are the common problems in Oracle WebLogic Server for OKE Domain In Image. Learn how to diagnose and solve them.

Topics

Free-Tier Autonomous Database

Free-Tier autonomous database is not supported.

RCU Datasources have Targets only to Administration Server

If you are using Domain In Image, for RAC database, then data sources that you create with the Enterprise Edition are targeted to only the administration server. Some of the data sources. like `mds-owsm`, `opss-audit-DBDS`, `opss-audit-viewDS`, `opss-data-source` need to be targeted to the WLS cluster. You need to update this after provisioning, by using the `update-domain` pipeline job.

Issue: Some of the data sources are not targeted to the WLS cluster.

Workaround:

Complete the following steps:

1. Create a model yaml file with the name `ee_datasource.yaml` and save it to a preferred location.
2. Open the `ee_datasource.yaml` file and copy-paste the following resources information:

> **Note:**
>
> Replace the administration server and cluster names in the target place holders <*adminserver-name*> and <*cluster-name*> respectively.

```
resources:
  JDBCSystemResource:
    'db1-mds-owsm':
      Target: '<adminserver-name>, <cluster-name>'
    'db2-mds-owsm':
      Target: '<adminserver-name>, <cluster-name>'
    'db1-opss-audit-DBDS':
      Target: '<adminserver-name>, <cluster-name>'
    'db2-opss-audit-DBDS':
      Target: '<adminserver-name>, <cluster-name>'
    'db1-opss-audit-viewDS':
      Target: '<adminserver-name>, <cluster-name>'
    'db2-opss-audit-viewDS':
      Target: '<adminserver-name>, <cluster-name>'
    'db1-opss-data-source':
      Target: '<adminserver-name>, <cluster-name>'
    'db2-opss-data-source':
      Target: '<adminserver-name>, <cluster-name>'
    'mds-owsm':
      Target: '<adminserver-name>, <cluster-name>'
    'opss-audit-DBDS':
      Target: '<adminserver-name>, <cluster-name>'
    'opss-audit-viewDS':
      Target: '<adminserver-name>, <cluster-name>'
```

```
'opss-data-source':
  Target: '<adminserver-name>, <cluster-name>'
```

3. Run the `update-domain` pipeline job, with the location of the model yaml file.
4. After the job succeeds, from the administration console verify that the following data sources are targeted to administration server and the WLS cluster.
   - `mds-owsm`
   - `opss-audit-DBDS`
   - `opss-audit-viewDS`
   - `opss-data-source`

Handling NFS Locking Errors

By default, the WebLogic stores are mount to the shared file system, which use Network File System (NFS) version 3 and is disabled. Therefore, the file locks on the different WebLogic stores and may not release if the VM of any node pool in the WebLogic Node pool is abruptly shut down. This is encountered in different scenarios, like, when a VM is stopped, restarted, or terminated, and there are WebLogic pods assigned to the worker node that is being terminated.

Issue: The WebLogic Server Pod (Admin Server or any manged server) fails to start and displays the following error in the WebLogic logs:

```
[Store:280105]The persistent file store "_WLS_myinstance-admin-server"
cannot open file _WLS_<instanceName>-<ServerName>000000.DAT.
```

Workaround:

To solve this issue, complete the following steps:

> **Note:**
>
> Even if you are using an earlier version of WebLogic Server you need to complete these steps.

1. Apply patch 32471832 by using the opatch update job, which is available in July 2021 PSUs.
2. For administration and managed server pods in the cluster, update the `domain.yaml` file by adding the `Dweblogic.store.file.LockEnabled=false` parameter.
   Following is an example, where the `Dweblogic.store.file.LockEnabled=false` parameter is added:

```
serverPod:
  env:
  - name: USER_MEM_ARGS
    #Default to G1GC algo
    value: "-XX:+UseContainerSupport -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
```

```
  - name: JAVA_OPTIONS
    value: "-Dweblogic.store.file.LockEnabled=false -
Dweblogic.rjvm.allowUnknownHost=true -
Dweblogic.security.SSL.ignoreHostnameVerification=true -
Dweblogic.security.remoteAnonymousRMIT3Enabled=false -
Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

3. Run the following command to apply `domain.yaml`.

```
kubectl -f <domain.yaml-file-path>
```

> **Note:**
>
> If you have created Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied, a few Security warnings are displayed. See About the Security Checkup Tool.

Unable to Access the Console or the Application

Troubleshoot problems accessing the console or the application after the Oracle WebLogic Server for OKE domain is successfully created.

Error accessing the console or the application

If you receive 502 bad gateway error when accessing the Jenkins console and WebLogic Server console, or the application using load balancer, use the `kubectl` command to get the node ports that are used by the system and ensure that these node ports are open for access via the load balancer subnet.

For example:

```
kubectl describe service --all-namespaces | grep -i nodeport
NodePort: http 32062/TCP
NodePort: https 30305/TCP
```

To check port access:

1. Access the Oracle Cloud Infrastructure console.
2. From the navigation menu, select **Networking**, and then click **Virtual Cloud Networks**.
3. Select the compartment in which you created the domain.
4. Select the virtual cloud network in which the domain was created.
5. Select the subnet where the WebLogic Server compute instance is provisioned.
6. Select the security list assigned to this subnet.
7. For an Oracle WebLogic Server for OKE cluster using a private and public subnet, make sure the following ingress rules exist:

```
Source: <LB Subnet CIDR>
IP Protocol: TCP
```

```
Source Port Range: All
Destination Port Range: 32062


Source: <LB Subnet CIDR>
IP Protocol: TCP
Source Port Range: All
Destination Port Range: 30305
```

For a domain on a private and public subnet, set the `Source` to the CIDR of the load balancer subnet.

Stack Creation Failed

Troubleshoot a failed Oracle WebLogic Server domain that you created using Oracle WebLogic Server for OKE.

Failed to install WebLogic Operator

Stack provisioning might fail when you create a domain with Oracle WebLogic Server for OKE in a new subnet for an existing VCN due to error in installation of WebLogic Server Kubernetes Operator.

Example message:

```
module.provisioner.null_resource.check_provisioning_status_1  (remote-
exec):
<Aug 27, 2020 07:01:31 PM GMT> <INFO>  <install_wls_operator.sh>
<(host:wrjrf8-admin.wrjrf8admin.existingnetwork.oraclevcn.com) -
<WLSOKE-VM-INFO-0020> :
Installing weblogic operator in namespace [wrjrf8-operator-ns]>
module.provisioner.null_resource.check_provisioning_status_1  (remote-
exec): <Aug 27, 2020
07:02:12 PM GMT> <ERROR>  <install_wls_operator.sh>
<(host:wrjrf8-admin.wrjrf8admin.existingnetwork.oraclevcn.com) -
<WLSOKE-VM-ERROR-0013> : Error
installing weblogic operator. Exit  code[1]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to create service account

Stack provisioning might fail with `HTTP 409` conflict error if the service account creation fails.

Example message:

```
module.provisioner.null_resource.check_provisioning_status_1 (remote-
exec):
HTTP response body: {"kind":"Status","apiVersion":"v1","metadata":
{},"status":"Failure","message":
"Operation cannot be fulfilled on serviceaccounts \"default\": the
object has been modified;
```

```
please apply your changes to the latest version and try
again","reason":"Conflict","details":
{"name":"default","kind":"serviceaccounts"}

,"code":409}
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to login to OCIR

Stack provisioning might fail if the docker login to OCI registry is not succcesful.

Example message:

```
[phx.ocir.io]>module.provisioner.null_resource.check_provisioning_status_1
(remote-exec):
<Sep 22, 2020 02:33:46 PM GMT> <ERROR> <docker_init.sh> <(host:wrfinal2-
admin.admin.existingnetwork.oraclevcn.com)
- <WLSOKE-VM-ERROR-0003> : Unable to login to custom OCIR
[phx.ocir.io]>module.provisioner.null_resource.check_provisioning_status_1
(remote-exec):
]>module.provisioner.null_resource.check_provisioning_status_1 (remote-exec):
<Sep 22, 2020 02:33:46 PM GMT> <ERROR> <docker_init.py> <(host:wrfinal2-
admin.admin.existingnetwork.oraclevcn.com)
- <WLSOKE-VM-ERROR-0020> : Error executing sh /u01/scripts/bootstrap/
docker_init.sh. Exit code [1]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to download ATP wallet

Stack provisioning might fail if you create a JRF-enabled domain running WebLogic Server 12c and using an Oracle Autonomous Database.

Example message in apply log:

```
module.provisioner.null_resource.check_provisioning_status_1 (remote-exec):
<Sep 22, 2020 12:31:11 PM GMT> <ERROR> <markers.py> <(host:wrfinal2-
admin.admin.existingnetwork.oraclevcn.com
- <Sep 22, 2020 12:31:11> - <WLS-OKE-ERROR-003> - Failed to verify oke
cluster nodes status.
[Exit code : {'status': 500, 'message': u'An internal server error has
occurred.',
'code': u'InternalServerError', 'opc-request-id':
'768603269A9D460D9B979632FC04C181/37A72EDA76A2687A5E24499AA6A70F9B/
7823A7DF9CDD435D869F3CB42C46B39E'}]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Failed to verify OKE cluster node status

Stack provisioning fails if the OKE cluster worker nodes are inactive when you create the WebLogic domain with Oracle WebLogic Server for OKE.

Example message:

```
<INFO> <oke_worker_status.py>
<(host:wlsatpte-admin.nevcnokeadmin.nevcnokevcn.oraclevcn.com) -
<WLSOKE-VM-INFO-0011> : Waiting
for the workers nodes to be Active. Retrying...><Dec 17, 2020 04:47:56
PM GMT> <ERROR>
<markers.py> <(host:wlsatpte-
admin.nevcnokeadmin.nevcnokevcn.oraclevcn.com) - <Dec 17, 2020
16:47:56> - <WLS-OKE-ERROR-003> - Failed to verify oke cluster nodes
status. [Exit code : Status
check timed out]>
```

Run a Destroy job on the stack and apply the job again to recreate the resources using the same database.

Load Balancer Creation Failed

After provisioning a stack, you might encounter an issue where the internal Load Balancer (LB) is missing.

When you run the following command, the external IP for the LB would is displayed as `<pending>`:

```
kubectl get svc -n ingress-nginx
```

In this case, the IP allocation for the LB fails and the LB instance is not created. This is because the quota for the selected LB shape is not available.

Previous Domain Image in Sample Application

You may encounter this issue in the WebLogic Server Console for a sample application Jenkins job.

After the sample application is successfully created in Jenkins using the `sample-app` job, on the WebLogic Server console, the sample application is not deployed with the new domain image, and still shows the previous domain image.

Example message:

```
10:15:10 + echo 'Publishing image [iad.ocir.io/ax8cfrmecktw/rsht1/
rsht1_domain/wls-domain-base: \
12.2.1.4.200714-200819-20-09-11_16-59-12] to domain...'
10:15:10 Publishing image [iad.ocir.io/ax8cfrmecktw/rsht1/rsht1_domain/
wls-domain-base: \
12.2.1.4.200714-200819-20-09-11_16-59-12] to domain...
10:15:10 + local running_domain_yaml=/tmp/running-
domain-20-09-11_16-59-12.yaml
```

```
10:15:10 + kubectl get domain rsht1domain -n rsht1-domain-ns -o yaml
10:15:10 + mkdir -p /u01/shared/weblogic-domains/rsht1domain/backups/
20-09-11_16-59-12
10:15:10 + cp /tmp/running-domain-20-09-11_16-59-12.yaml \
/u01/shared/weblogic-domains/rsht1domain/backups/20-09-11_16-59-12/prev-
domain.yaml
10:15:10 + sed -i -e 's|\(image: \).*|\1 "iad.ocir.io/ax8cfrmecktw/rsht1/
rsht1_domain/wls-domain-base: \
12.2.1.4.200714-200819-20-09-11_16-59-12"|g' /tmp/running-
domain-20-09-11_16-59-12.yaml
10:15:10 + kubectl apply -f /tmp/running-domain-20-09-11_16-59-12.yaml
10:15:21 Error from server (Conflict): error when applying patch:
10:15:21 {"metadata":{"annotations":{"kubectl.kubernetes.io/last-applied-
configuration": \
"{\"apiVersion\":\"weblogic.oracle/v8\",\"kind\":\"Domain\"
```

As a workaround, run the sample application again using the `sample-app` Pipeline job.

Reinstall Load Balancers for Jenkins

When provisioning, creating a load balancer can fail for different reasons. However, provisioning does not stop as the load balancers can be created later. Follow the steps in this section to recreate the load balancers.

When you create a Oracle WebLogic Server for OKE instance, two load balancers are created. One with a public IP, that provides access to the applications installed in the WebLogic cluster, and another with a private IP, to provide access to the WebLogic console and Jenkins console.

Following are the reasons load balancer creation fails during provisioning:

1. Lack of quota for the selected LB shapes.
2. Lack of available public IPs (for external load balancer) or private IPs (for internal load balancer) in the VCN or subnets selected during provisioning.

Check the Status of the Load Balancers

You can view the status of the load balancers by checking the Resource Manager job log, the load balancer services, and the provisioning logs.

Resource Manager job log: When both the load balancers are created successfully, the resource manager job log includes the following:

```
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec): {
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec):
"weblogic_console_url": "http://<IP_adressess>/console",
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec):
"jenkins_console_url": "http://<IP_adressess>/jenkins",
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec):
"weblogic_cluster_lb_url": "https://<IP_adressess>/<application context>"
module.provisioner.null_resource.check_provisioning_status_3 (remote-exec): }
```

The first two lines are for the private load balancer. The third line is for the public load balancer. If any of the load balancers are not created, you will not see any of the above lines in the Resource Manager job log.

Load Balancer Services:

To check the load balancer services, run the following command:

```
kubectl get svc -n ingress-nginx
```

If the output lists any of the load balancer services as `<pending>`, under the `EXTERNAL-IP` column, then the load balancers are not created.

Sample output:

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
rsh3oke-external    LoadBalancer   10.1.1.1     <pending>
443:30618/TCP    11m
rsh3oke-internal    LoadBalancer   10.1.1.2    100.1.1.1
80:30790/TCP     11m
```

Provisioning logs:

If the internal or external load balancer is *not* created successfully, the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error message:

```
<WLSOKE-VM-INFO-0058> : Installing ingress controller charts for
jenkins [ ingress-controller ]>
<WLSOKE-VM-ERROR-0058> : Error installing ingress controller with
Helm. Exit code [1]>
```

And, in the `/u01/logs/provisioning_cmd.out` file, you would see the following error message:

```
<install_ingress_controller.sh>  -  Error: timed out waiting for the
condition
```

Reinstall the Load Balancers

After identifying and fixing the cause of the failure, like increased quota for the selected LB shape, you can reinstall the load balancers in the instance.

1. Run the following command to get the values required to install the ingress-controller:

   ```
   helm get values ingress-controller -o yaml > ingress_values.yaml
   ```

2. Run the following command to remove the existing helm release:

   ```
   helm uninstall ingress-controller
   ```

> **Note:**
>
> This command will delete both the external and internal load balancers.

3. Run the following command to install both the external and internal load balancers:

```
/u01/scripts/bootstrap/install_ingress_controller.sh ingress_values.yaml
```

Sample Output:

```
<Nov 20, 2020 08:01:01 PM GMT> <INFO> <install_ingress_controller.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0058> : Installing ingress controller
charts for jenkins [ ingress-controller ]>
<Nov 20, 2020 08:03:27 PM GMT> <INFO> <install_ingress_controller.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0059> : Successfully installed
ingress controller>
```

4. Run the following command to verify if load balancer services are created and have external IP addresses:

```
kubectl get svc -n ingress-nginx
```

Sample output:

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
rsh3oke-external    LoadBalancer   10.96.115.139   144.25.19.38
443:31162/TCP    12m
rsh3oke-internal    LoadBalancer   10.96.249.188   100.111.191.133
80:30605/TCP     12m
```

Install Jenkins Manually

When you create a Oracle WebLogic Server for OKE instance, Jenkins is installed by installing a Helm release called *jenkins-oke*. During provisioning, Jenkins installation may fail, but provisioning is not stopped, because Jenkins can be installed after provisioning. This section explains how to install Jenkins manually, if Jenkins installation has failed during provisioning.

Check if Jenkins Install Failed during Provisioning

You can know if the Jenkins install failed by trying to access the Jenkins console, checking the provisioning logs, and checking the Kubernetes resources (pods, services, and so on) under the `jenkins-ns` namespace.

Access the Jenkins console:

Try accessing the Jenkins console, as described in Access the Jenkins Console.

If you are not able to access the console, then continue to the next section to check the logs.

Provisioning logs:

If Jenkins is *not* installed successfully, then the `/u01/logs/provisioning.log` file would include an error message.

Sample of the error:

```
<WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<WLSOKE-VM-ERROR-0052> : Error installing jenkins charts. Exit code[1]>
```

And, you would see the details of the failure in the `/u01/logs/provisioning_cmd.out` file.

Kubernetes resources:

To check the Kubernetes resources in the `jenkins-ns` namespace, run the following command:

```
kubectl get all -n jenkins-ns
```

Following is a sample output, where Jenkins was installed correctly:

```
NAME                                        READY   STATUS    RESTARTS
AGE
pod/jenkins-deployment-5bb55586b9-vn8sk   1/1     Running   0
26m

NAME                        TYPE        CLUSTER-IP    EXTERNAL-IP
PORT(S)              AGE
service/jenkins-service   ClusterIP   10.96.149.6   <none>        8080/
TCP,50000/TCP   26m

NAME                                  READY   UP-TO-DATE   AVAILABLE
AGE
deployment.apps/jenkins-deployment   1/1     1            1
26m

NAME                                             DESIRED   CURRENT
READY   AGE
replicaset.apps/jenkins-deployment-5bb55586b9   1         1
1       26m
```

Install Jenkins Manually

After identifying and fixing the cause of the failure, install Jenkins in your instance.

1. Check if the `provisioning_metadata.properties` file exists, at the `/u01/shared/weblogic-domains/<domain>` directory.
   Does the `provisioning_metadata.properties` file exist?
   - **Yes**: Continue with the next step.
   - **No**: Run the following command:

     ```
     python /u01/scripts/metadata/provisioning_metadata.py
     ```

Continue with the next step.

2. Run the following command to remove the existing helm release:

```
helm uninstall jenkins-oke
```

3. Run the following command to install Jenkins:

```
/u01/scripts/bootstrap/install_jenkins.sh  /u01/provisioning-data/jenkins-inputs.yaml
```

Where, `jenkins-inputs.yaml` file contains the required variables.

Sample Output:

```
<Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.py>
<(host:host_name) - updated /u01/provisioning-data/jenkins-inputs.yaml>
<Nov 23, 2020 05:10:07 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0098> : Creating configmap [wlsoke-metadata-configmap]>
<Nov 23, 2020 05:10:09 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0056> : Installing jenkins jenkins-ns>
<Nov 23, 2020 05:10:22 PM GMT> <INFO> <install_jenkins.sh>
<(host:host_name) - <WLSOKE-VM-INFO-0057> : Successfully installed jenkins in namespace [ jenkins-ns ]>
```

You have successfully installed the Jenkins console. Try accessing the Jenkins console, as described in Access the Jenkins Console.

Security Checkup Tool Warnings

Learn about the security check warnings that are displayed in the Oracle WebLogic Server Administration console and how to troubleshoot them.

At the top of the WebLogic Server Administration console, the message `Security warnings detected. Click here to view the report and recommended remedies` is displayed for Oracle WebLogic Server for OKE instances created after July 20, 2021, or the instances on which the July 2021 PSUs are applied.

When you click the message, a list of security warnings are displayed as listed in the following table.

The warning messages listed in the table are examples.

Security Warnings

Warning Message	Resolution
`SSL hostname verification is disabled by the SSL configuration.`	Review your applications before you make any changes to address these SSL host name security warnings. For applications that connect to SSL endpoints with a host name in the certificate, which does not match the local machine's host name, the connection fails if you configure the BEA host name verifier in Oracle WebLogic Server. For applications that connect to Oracle provided endpoints such as Oracle Identity Cloud Service (for example,`*.identity.oraclecloud.com`), the connection fails if you did not configure the wildcard host name verifier or a custom host name verifier that accepts wildcard host names. If you are not sure of the SSL configuration settings you should configure to address the warning, Oracle recommends that you configure the wildcard host name verifier. You see the SSL host name verification warnings in case of existing Oracle WebLogic Server for OKE instances (created before July 20, 2021). To address this warning, you must configure SSL with host name verifier. See Configure SSL with host name verifier.
`Production mode is enabled but the file or directory <directory_name>/startWebLogic.sh is insecure since its permission is not a minimum of umask 027`	Run the following command in the administration server as `oracle` user: `chmod 640 /u01/data/domains/<domain_name>/bin`
`Remote Anonymous RMI T3 or IIOP requests are enabled. Set the RemoteAnonymousRMIT3Enabled and RemoteAnonymousRMIIIOPEnabled attributes to false.`	Set the java properties for anonymous RMI T3 and IIOP requests during server start up. See Set the Java Properties.

After you address the warnings, you must click **Refresh Warnings** to see the warnings removed in the console.

For Oracle WebLogic Server for OKE instances created after July 20, 2021, though the java properties to disable anonymous requests for preventing anonymous RMI access are configured, the warnings still appear. This is a known issue in Oracle WebLogic Server.

Set the Java Properties

To set the java properties for anonymous RMI T3 and IIOP requests:

1. Edit the `domain.yaml` located in `/u01/shared/weblogic-domains/<domain_name>/domain.yaml` for all instances of `serverPod` definitions as follows:

```
serverPod:
      env:
      - name: USER_MEM_ARGS
        #admin server memory is explicitly set to min of 256m and max of
512m and GC algo is G1GC
        value: "-Xms256m -Xmx512m -XX:+UseG1GC -
Djava.security.egd=file:/dev/./urandom"
      - name: JAVA_OPTIONS
        value: "-Dweblogic.store.file.LockEnabled=false
        -Dweblogic.rjvm.allowUnknownHost=true
        -Dweblogic.security.remoteAnonymousRMIT3Enabled=false
        -Dweblogic.security.remoteAnonymousRMIIIOPEnabled=false"
```

2. Apply the `domain.yaml` using the `kubectl` command:

```
kubectl -f <path_to_domain.yaml>
```

Get Additional Help

Use online help, email, customer support, and other tools if you have questions or problems with Oracle WebLogic Server for OKE.

For general help with Oracle Cloud Marketplace, see How Do I Get Support in *Using Oracle Cloud Marketplace*.

A

Patches Included in Oracle WebLogic Server for OKE

Each Oracle WebLogic Server for OKE release includes patches from several products, namely, Oracle WebLogic Server, Oracle JDeveloper, Oracle Java Development Kit, Oracle Platform Security Services and Oracle Web Services Manager.

> **Note:**
>
> If you are using Oracle WebLogic Server for OKE (**Release 21.3.3 or later**), see Using Oracle WebLogic Server for OKE.

Patches in an Oracle WebLogic Server for OKE release are not automatically applied to existing domains. You have to apply the patches manually if you wish to update your existing domain to match the latest or other supported release.

The following table shows the patches in a Oracle WebLogic Server for OKE release. Use your Oracle Support account to locate and download the patch you wish to apply.

> **Tip:**
>
> For a list of new features and enhancements that were added recently to improve your Oracle WebLogic Server for OKE experience, see What's New for Oracle WebLogic Server for OKE.

WebLogic Server Version	Domain Home Source Type	Patches	Oracle WebLogic Server for OKE Release	Patch List
12.2.1.4.210720.02-MII	Model in Image	opatch: • 30385564 - Oracle XML Developers Kit patch • 32784652 - OPSS Patch for April 2021 • 31544353 - ADR for WebLogic Server • 1221410 (32973297) - Coherence 12.2.1.4 cumulative Patch 10 (12.2.1.4.10) • 33084721 - ADF bundle patch 12.2.1.4.210706 • 33059296 - WLS patch set update 12.2.1.4.210629 • 32880070 - FMW common third-party SPU 12.2.1.4.0 for April 2021 CPU • 32772437 - FMW Platform 12.2.1.4.0 for April 2021 CPU • 32471832 - System prop to disable all file store locks. • 32905339 - OWSM bundle patch 12.2.1.4.210520	21.3.2	July 2021 PSUs
12.2.1.4.210720.01-MII	Model in Image	opatch: • 30385564 - Oracle XML Developers Kit patch • 32784652 - OPSS Patch for April 2021 • 31544353 - ADR for WebLogic Server • 1221410 (32973297) - Coherence 12.2.1.4 cumulative Patch 10 (12.2.1.4.10) • 33084721 - ADF bundle patch 12.2.1.4.210706 • 33059296 - WLS patch set update 12.2.1.4.210629 • 32880070 - FMW common third-party SPU 12.2.1.4.0 for April 2021 CPU • 32772437 - FMW Platform 12.2.1.4.0 for April 2021 CPU • 32471832 - System prop to disable all file store locks.	21.3.1	July 2021 PSUs
12.2.1.4.210420.05-MII	Model in Image	opatch: • 32698246 - WLS CPU patch April 2021 • 32684757 - ADF/JDev bundle patch for April 2021 • 122148 (32581859) - WLS Coherence patch for April 2021 • 32784652 - OPSS Patch for April 2021 • 31544353 - ADR for WebLogic Server • 30385564 - Oracle XML Developers Kit patch • 32652899 - FMW common third-party SPU 12.2.1.4.0 for April 2021 CPU • 32772437 - FMW Platform 12.2.1.4.0 for April 2021 CPU	21.2.3	April 2021 PSUs

WebLogic Server Version	Domain Home Source Type	Patches	Oracle WebLogic Server for OKE Release	Patch List
12.2.1.4.210420.04-MII	Model in Image	opatch: • 32698246 - WLS CPU patch April 2021 • 32684757 - ADF/JDev bundle patch for April 2021 • 122148 (32581859) - WLS Coherence patch for April 2021 • 32784652 - OPSS Patch for April 2021 • 31544353 - ADR for WebLogic Server • 30385564 - Oracle XML Developers Kit patch • 32652899 - FMW common third-party SPU 12.2.1.4.0 for April 2021 CPU • 32772437 - FMW Platform 12.2.1.4.0 for April 2021 CPU	21.2.2	April 2021 PSUs
12.2.1.4.210420.03-DII	Domain in Image	opatch: • 32698246 - WLS CPU patch April 2021 • 32684757 - ADF/JDev bundle patch for April 2021 • 122148 (32581859) - WLS Coherence patch for April 2021 • 32622685 - OPSS bundle patch • 31544353 - ADR for WebLogic Server • 30385564 - Oracle XML Developers Kit patch	21.2.1	April 2021 PSUs

Download Patches Using the Patching Tool Utility

Oracle WebLogic Server for OKE provides the patching tool utility to download the patches for the WebLogic Server instances. This utility can be used if you do not have access to the support portal to download the required patches.

You can use this patching tool utility on the Administration host and the bastion instance.

With this patching tool utility, you can list, download, and upgrade the patches for the WebLogic Server instances. To apply the patches, see Apply a WebLogic Server Patch.

1. Identify the following IP address of the node in your domain:
   - The public IP address to the Administration Server node.
   - The public IP address of the bastion and the private IP address of the compute instance.
2. From your computer, run the `ssh` command to connect to the domain's Administration Server node as the `opc` user.

   ```
   ssh -i <path_to_private_key> 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>
```

3. Run the following commands:

Command	Output/Result	Description

`patch-utils -v`	`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.`
		Displays the build version, and Oracle license and copyright information

`patch-utils setup`	`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]`
		Configures the region from where to download the patches and creates the configuration file in the specified middleware Home.
		Note: The user can download the patches from the five specified regions only.

Command	Output/Result	Description
`patch-utils list`	*<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>*	Lists the patches in the patch catalog for the applicable WebLogic Server version. **Note:** You must set up the configuration file before running the `patch-utils list` command.

Command	Output/Result	Description

`patch-utils list -a`	`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`
		Lists all the current patches based on OPatch utility for 12c and BSU (BEA Smart Update) for 11g.
		Note: You must set up the configuration file before running the `patch-utils list -a` command.

Command	Output/Result	Description
	Interim patches (2): 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>* Patch *<Patch number>*: applied on *<day month date time>* Unique Patch ID: *<Patch ID number>* Patch description: "ADF Bundle patch *<WebLogic 12c version number>*" Created on *<date month year time>* Bugs fixed:*<Bug number1>*, *<Bug number2>*, *<Bug number3>* OPatch succeeded.	

Command	Output/Result	Description

`patch-utils info -n <Patch ID>`	`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`
	`....`
		Displays information of the specified patch.
		The WebLogic Server patches include the `readme` file that provides the patch details and other useful information about patching.
		Note: You can use the `-l` parameter to print the specified number of lines from the `readme` file.
		Example:
		`patch-utils info -n <Patch ID> -l 25`

`patch-utils download -l <Patch ID> -p <Location to download>`	`Successfully downloaded following patches.`
	`Please copy them to weblogic hosts and apply them locally.`
	`['<Patch ID_Generic.zip']`
		Downloads the patches to the specified location.
		To download multiple patches, specify the patch IDs as comma separated values.
		Note: You can also download unencrypted patches.

`patch-utils upgrade` **Note:** This command is used to upgrade VMs if the NAT Gateway is enabled on the WebLogic Server subnet.	`Successfully updated patch-utils to [<Patch Utils version number>]. Please rerun patch-utils.`	Upgrades the patching tool utility to the latest version.

B

Oracle Cloud Identifiers and Listings in Oracle WebLogic Server for OKE

Learn about the list of Oracle WebLogic Server for OKE images available on the Partner Image Catalog that contains the entitlement to use the different versions of Oracle WebLogic software for Oracle WebLogic Server for OKE UCM application. These images are priced at the same rate as the Oracle WebLogic Server Enterprise Edition for OKE and Oracle WebLogic Suite for OKE stacks.

See Oracle WebLogic Server Enterprise Edition for OKE and Oracle WebLogic Suite for OKE.

> **Note:**
>
> To view the pricing details, click the **Get App** link.

The following table shows the listings and the OCIDs for the different Oracle WebLogic Server editions of Oracle WebLogic Server for OKE.

WebLogic Server Edition	Image Name	Image ID	Listing ID	Resource Version ID
Oracle WebLogic Server Enterprise Edition	wlsoke-custom-np-image-ee-UCM-20.4.1-20091703004	ocid1.image.oc1..aaaaaaaaibbsg23uasf77j4kdldnjnbmgkfjxd5gqywabs3hwx2jw45pj24q	ocid1.appcataloglisting.oc1..aaaaaaaabw6dti6ejlfe4h5vcdtuemmzcbxc6myje2t4au6fox5excyiy2ma	20.4.1-20091703004-092120202314
Oracle WebLogic Suite Edition	wlsoke-custom-np-image-suite-UCM-20.4.1-20091703004	ocid1.image.oc1..aaaaaaaaznbtycmdn7747itt3qmipjvnui4xnnjgiztteszgghnjepjrbknq	ocid1.appcataloglisting.oc1..aaaaaaaaln2a5njbk3mtcqmokjrptv62cqeoqrm4ntyjojko5lqypqbgucua	20.4.1-20091703004-092120202313

C

License Information for Oracle WebLogic Server for OKE

Learn about the licensed third-party technology associated with Oracle WebLogic Server for OKE.

Open Source or Other Separately Licensed Software

Required notices for open source or other separately licensed software products or components distributed in Oracle WebLogic Server for OKE are identified in the following table along with the applicable licensing information. Additional notices and/or licenses may be found in the included documentation or `readme` files of the individual third party software.

Provider	Component(s)	Licensing Information
Docker Inc.	Docker 19.03.11.ol-4.el7	Apache License Version 2
Jenkins CI	Jenkins 2.235.5	MIT License
Jenkins CI	Jenkins active choices-plugin 2.3	MIT License
Jenkins CI	Jenkins Mask Passwords Plug-in 2.13	MIT License
Kubernetes	kubectl 1.17.9	Apache License Version 2
Python Software Foundation	Python 3.6.8-13.0.1.el7	Python License 2.0
The Helm Authors	Helm 3.2.4	Apache License Version 2
The Kubernetes Authors	Kubernetes Python Client 11.0.0	Apache License Version 2
The pip developers	pip 20.1.1	MIT License
Yichun Zhang	OpenResty 1.15.8.2	BSD License

D

Script File To Validate Network Setup

You must create a script file to validate if the existing WebLogic Server subnet and the database subnets meet the prerequisites to provision the WebLogic instance in Oracle WebLogic Server for OKE. You can copy the following scripts in Cloud Shell to perform the validation. For example, copy the scripts and save the file as `validateoke.sh`.

```
# Script to validate existing public, private and database subnets meet the
prerequisites
# for provisioning and proper functioning of Oracle WebLogic Server for OKE.
#
version="1.0.0"

# Set Flags
# -----------------------------------
# Flags which can be overridden by user input.
# Default values are below
# -----------------------------------
DB_PORT=1521
SSH_PORT=22
BASTION_SUBNET_OCID=""
ADMIN_SUBNET_OCID=""
WORKER_SUBNET_OCID=""
FSS_SUBNET_OCID=""
LB_SUBNET_OCID=""

DB_SUBNET_OCID=""
BASTION_HOST_IP_CIDR=""

debug=false
args=()

function ip_to_int() {
  local ip_addr="${1}"
  local ip_1 ip_2 ip_3 ip_4

  ip_1=$(echo "${ip_addr}" | cut -d'.' -f1)
  ip_2=$(echo "${ip_addr}" | cut -d'.' -f2)
  ip_3=$(echo "${ip_addr}" | cut -d'.' -f3)
  ip_4=$(echo "${ip_addr}" | cut -d'.' -f4)

  echo $(( ip_1 * 256**3 + ip_2 * 256**2 + ip_3 * 256 + ip_4 ))
}

#################################################
# Determine whether IP address is in the specified subnet.
#
# Args:
#   cidr_subnet: Subnet, in CIDR notation.
```

```
#   ip_addr: IP address to check.
#
# Returns:
#   0|1
#######################################################
function in_cidr_range() {
  local cidr_subnet="${1}"
  local ip_addr="${2}"
  local subnet_ip cidr_mask netmask ip_addr_subnet subnet rval

  subnet_ip=$(echo "${cidr_subnet}" | cut -d'/' -f1)
  cidr_mask=$(echo "${cidr_subnet}" | cut -d'/' -f2)

  netmask=$(( 0xFFFFFFFF << $(( 32 - ${cidr_mask} )) ))

  # Apply netmask to both the subnet IP and the given IP address
  ip_addr_subnet=$(( netmask & $(ip_to_int ${ip_addr}) ))
  subnet=$(( netmask & $(ip_to_int ${subnet_ip}) ))

  # Subnet IPs will match if given IP address is in CIDR subnet
  [ "${ip_addr_subnet}" == "${subnet}" ] && rval=0 || rval=1

  return $rval
}

#######################################################
# Validates if one of service or nat gateways exist in the specified
private subnet.
#
# Returns:
#   0|1
#######################################################
function validate_service_or_nat_gw_exist() {
  local subnet_ocid=$1
  local vcn_ocid=""
  local vcn_compartment_ocid=""
  is_private_subnet=$(oci network subnet get --subnet-id "$
{subnet_ocid}" | jq -r '.data["prohibit-public-ip-on-vnic"]')

  if [[ $is_private_subnet = true ]]
  then
    vcn_ocid=$(oci network subnet get --subnet-id "${subnet_ocid}" |
jq -r '.data["vcn-id"]')
    vcn_compartment_ocid=$(oci network vcn get --vcn-id "${vcn_ocid}"
| jq -r '.data["compartment-id"]')
    # Check if NAT gateway exists in the VCN
    res=$(oci network nat-gateway list --compartment-id $
{vcn_compartment_ocid} --vcn-id ${vcn_ocid})
    nat_gw_found=$(if [[ -n $res ]]; then echo 0; else echo 1; fi)

    # Check if Service gateway exists in the VCN
    res=$(oci network service-gateway list --compartment-id $
{vcn_compartment_ocid} --vcn-id ${vcn_ocid})
    svc_gw_found=$(if [[ -n $res ]]; then echo 0; else echo 1; fi)
```

```
    # One of NAT or Service Gateway must exist
    if [[ $nat_gw_found -ne 0 ]] && [[ $svc_gw_found -ne 0 ]]
    then
      echo 1
      return
    fi

    # Admin subnet should be using either NAT or service gateway or both in
its routetable
    rt_ocid=$(oci network subnet get --subnet-id ${subnet_ocid} | jq -r
'.data["route-table-id"]')
    rt_rules=$(oci network route-table get --rt-id ${rt_ocid} | jq -r
'.data["route-rules"]')
    rt_rules_count=$(echo $rt_rules | jq '.|length')

    nat=""
    svc=""
    nat_gw_id=""
    svc_gw_id=""

    for ((i = 0 ; i < $rt_rules_count ; i++))
    do
      network_entity_ocid=$(echo $rt_rules | jq -r --arg i "$i" '.[$i|
tonumber]["network-entity-id"]')
      nat_id=$(echo $network_entity_ocid | grep natgateway)
      if [[ -n $nat_id ]]; then nat_gw_id=$nat_id; fi

      svc_id=$(echo $network_entity_ocid | grep servicegateway)
      if [[ -n $svc_id ]]; then svc_gw_id=$svc_id; fi
    done

    if [[ (-z $nat_gw_id  && -z $svc_gw_id) ]]; then
      echo 2
      return
    fi

    # If WLS subnet route table has a rule to use service gateway then it
should be using
    # all-<region-code>-services-in-oracle-services-network destination
    echo ""
    if [[ -n $svc_gw_id ]]
    then
      is_all_services_name=$(oci network service-gateway get --service-
gateway-id $svc_gw_id | jq -r '.data.services[0]["service-name"]' | grep -i
"all.*services in oracle services network")
      if [[ -z $is_all_services_name ]]
      then
        echo 3
        return
      fi
      for ((i = 0 ; i < $rt_rules_count ; i++))
      do
        network_entity_ocid=$(echo $rt_rules | jq -r --arg i "$i" '.[$i|
tonumber]["network-entity-id"]')
        res=$(echo $network_entity_ocid | grep servicegateway)
```

```
        if [[ -n $res ]]
        then
          all_services_destination=$(echo $rt_rules | jq -r --arg i
"$i" '.[$i|tonumber].destination'  | grep -i "all-.*-services-in-
oracle-services-network")
          if [[ -z $all_services_destination ]]
          then
            echo 4
            return
          fi
        fi
      done
    fi
  fi
  echo 0
}

#####################################################
# Validates if the internet gateway exists in the VCN of Admin subnet.
# Without Internet gateway in Admin Subnet VCN, SSH access from ORM
will not work.
# When using terraform CLI from within private network, internet
gateway is not required.
# Hence this check will give a warning and not an error.
#
# Returns:
#   0|1
#####################################################
function validate_internet_gw_exist() {
  local subnet_ocid=$1
  local vcn_ocid=""
  local vcn_compartment_ocid=""

  vcn_ocid=$(oci network subnet get --subnet-id ${subnet_ocid} | jq -r
'.data["vcn-id"]')
  vcn_compartment_ocid=$(oci network vcn get --vcn-id ${vcn_ocid} | jq
-r '.data["compartment-id"]')
  # Check if Service gateway exists in the VCN
  res=$(oci network internet-gateway list --compartment-id $
{vcn_compartment_ocid} --vcn-id ${vcn_ocid})
  if [[ -n $res ]]; then
    echo 0
  else
    echo 1
  fi
}

#####################################################
# Checks if specified port is open to specified source CIDR in the
specified seclist's ingress rules.
#
# Args:
#     seclist_ocid: Security list OCID for the security list to check
ingress rules for.
#     port: destination port to check
```

```
#     source: Source CIDR (either block/range of IPs or single IP (with /32
suffix)
#
# Returns:
#   0|1
#######################################################
function check_tcp_port_open_in_seclist() {
  local seclist_ocid=$1
  local port=$2
  local source=$3
  local port_is_open=false
  local tcp_protocol="6"

  ingress_rules=$(oci network security-list get --security-list-
id $seclist_ocid | jq -r '.data["ingress-security-rules"]')
  ingress_rules_count=$(echo $ingress_rules | jq '.|length')

  for ((i = 0 ; i < $ingress_rules_count ; i++))
  do
    ingress_protocol=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].protocol')
    ingress_source=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].source')
    tcp_options=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]
["tcp-options"]')
    port_min=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["tcp-
options"]["destination-port-range"].min')
    port_max=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["tcp-
options"]["destination-port-range"].max')

    source_in_cidr_range=1
    if [[ $source = "0.0.0.0/0" ]]
    then
      if [[ $ingress_source = $source ]]
      then
        source_in_cidr_range=0
      else
        source_in_cidr_range=1
      fi
    else
      source_in_cidr_range=$(in_cidr_range $ingress_source $source ; echo $?)
    fi

    if [[ ($ingress_protocol = "all" || $ingress_protocol = $tcp_protocol )
&& ( $tcp_options = "null" || ( $port -ge $port_min && $port -
le $port_max ) ) && $source_in_cidr_range -eq 0 ]]
    then
       port_is_open=true
       echo 0
       return
    fi
  done
  echo 1
}
```

```
#####################################################
# Validates if the specified TCP port is open for the WLS subnet CIDR.
#
# Args:
#     port:         Destination port
#     source_cidr:  Source CIDR
#
# Returns:
#   0|1
#####################################################
function validate_subnet_port_access() {
  local port_found_open=1
  local subnet=$1
  local port=$2
  local source_cidr=$3
  local protocol=$4 # Default protocol is TCP, if it is UDP then need
to pass this param
  sec_lists=$(oci network subnet get --subnet-id ${subnet} | jq -c
'.data["security-list-ids"]')
  # Convert to bash array
  declare -A seclists_array

  while IFS="=" read -r key value
  do
      seclists_array[$key]="$value"
  done < <(jq -r 'to_entries|map("\(.key)=\(.value|tostring)")|.[]'
<<< "$sec_lists")
  # Check the ingress rules for specified destination port is open for
access by source CIDR
  for seclist_ocid in "${seclists_array[@]}"
  do
    if [[ $port_found_open -ne 0 ]]; then
      if [[ -z $protocol ]]; then # default is TCP
        port_found_open=$(check_tcp_port_open_in_seclist $seclist_ocid
"${port}" "$source_cidr")
      else # protocol param is non empty then udp
        port_found_open=$(check_udp_port_open_in_seclist $seclist_ocid
"${port}" "$source_cidr")
      fi
    fi
  done
  echo $port_found_open
}

#####################################################
# Checks if specified UDP port is open to specified source CIDR in the
specified seclist's ingress rules.
#
# Args:
#     seclist_ocid: Security list OCID for the security list to check
ingress rules for.
#     port: destination port to check
#     source: Source CIDR (either block/range of IPs or single IP
(with /32 suffix)
#
```

```
# Returns:
#   0|1
#####################################################
function check_udp_port_open_in_seclist() {
  local seclist_ocid=$1
  local port=$2
  local source=$3
  local port_is_open=false
  local udp_protocol="17"

  ingress_rules=$(oci network security-list get --security-list-
id $seclist_ocid | jq -r '.data["ingress-security-rules"]')
  ingress_rules_count=$(echo $ingress_rules | jq '.|length')

  for ((i = 0 ; i < $ingress_rules_count ; i++))
  do
    ingress_protocol=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].protocol')
    ingress_source=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|
tonumber].source')
    udp_options=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]
["udp-options"]')
    port_min=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["udp-
options"]["destination-port-range"].min')
    port_max=$(echo $ingress_rules | jq -r --arg i "$i" '.[$i|tonumber]["udp-
options"]["destination-port-range"].max')

    source_in_cidr_range=1
    if [[ $source = "0.0.0.0/0" ]]
    then
      if [[ $ingress_source = $source ]]
      then
        source_in_cidr_range=0
      else
        source_in_cidr_range=1
      fi
    else
      source_in_cidr_range=$(in_cidr_range $ingress_source $source ; echo $?)
    fi

    if [[ ($ingress_protocol = "all" || $ingress_protocol = $udp_protocol )
&& ( $udp_options = "null" || ( $port -ge $port_min && $port -
le $port_max ) ) && $source_in_cidr_range -eq 0 ]]
    then
       port_is_open=true
       echo 0
       return
    fi
  done
  echo 1
}


#####################################################
# Validates if CIDR is a valid single host IP (must end with /32 suffix).
```

```
#
# Args:
#     ip_cidr: Single host IPv4 Address in CIDR format
#
# Returns:
#   0|1
#####################################################
function is_valid_ip_cidr() {
  local ip_cidr=$1

  is_valid=$(echo ${ip_cidr} | grep -E '^(([0-9]|[1-9][0-9]|1[0-9]{2}|
2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|
25[0-5])(\/(32))$')
  if [[ -n $is_valid ]]; then
    echo 0
  else
    echo 1
  fi
}

############## Begin Options and Usage ##################

# Print usage
usage() {
  echo -n "$0 [OPTIONS]...

 This script is used to validate existing subnets for OKE - Bastion,
Admin, Worker, FSS, LB subnets (and optionally database subnets) are
setup correctly.
 ${bold}Options:${reset}
  -b, --bastionsubnet Bastion Subnet OCID (Required)
  -a, --adminsubnet   Admin Subnet OCID (Required)
  -w  --workersubnet  Workers Subnet OCID (Required)
  -f  --fsssubnet     FSS Subnet OCID (Required)
  -l  --lbsubnet      LB Subnet OCID (Required)
  -d, --dbsubnet      DB Subnet OCID
  -i, --bastionipcidr Bastion Host IP CIDR (should be suffixed
with /32)
      --debug         Runs script in BASH debug mode (set -x)
  -h, --help          Display this help and exit
      --version       Output version information and exit
  "
}

# Iterate over options breaking -ab into -a -b when needed and --
foo=bar into
# --foo bar
optstring=h
unset options
while (($#)); do
  case $1 in
    # If option is of type -ab
    -[!-]?*)
      # Loop over each character starting with the second
      for ((i=1; i < ${#1}; i++)); do
```

```
        c=${1:i:1}

        # Add current char to options
        options+=("-$c")

        # If option takes a required argument, and it's not the last char
make
        # the rest of the string its argument
        if [[ $optstring = *"$c:"* && ${1:i+1} ]]; then
          options+=("${1:i+1}")
          break
        fi
      done
      ;;

    # If option is of type --foo=bar
    --?*=*) options+=("${1%%=*}" "${1#*=}") ;;
    # add --endopts for --
    --) options+=(--endopts) ;;
    # Otherwise, nothing special
    *) options+=("$1") ;;
  esac
  shift
done
set -- "${options[@]}"
unset options

# Print help if no arguments were passed.
[[ $# -eq 0 ]] && set -- "--help"

# Read the options and set stuff
while [[ $1 = -?* ]]; do
  case $1 in
    -h|--help) usage >&2; exit 0 ;;
    --version) echo "$(basename $0) ${version}"; exit 0 ;;
    -b|--bastionsubnet) shift; BASTION_SUBNET_OCID=${1} ;;
    -a|--adminsubnet) shift; ADMIN_SUBNET_OCID=${1} ;;
    -w|--workersubnet) shift; WORKER_SUBNET_OCID=${1} ;;
    -f|--fsssubnet) shift; FSS_SUBNET_OCID=${1} ;;
    -l|--lbsubnet) shift; LB_SUBNET_OCID=${1} ;;
    -d|--dbsubnet) shift; DB_SUBNET_OCID=${1} ;;
    -i|--bastionipcidr) shift; BASTION_HOST_IP_CIDR=${1} ;;
    --debug) debug=true;;
    --endopts) shift; break ;;
    *) "invalid option: '$1'." ; usage >&2; exit 1 ;;
  esac
  shift
done

# Store the remaining part as arguments.
args+=("$@")

############## End Options and Usage ###################

# ############# ############# #############
```

```
# ##       MAIN SCRIPT BODY             ##
# ##                                    ##
# ##                                    ##
# ############ ############ ############

# Set IFS to preferred implementation
IFS=$'\n\t'

# Exit on error. Append '||true' when you run the script if you expect
an error.
set -o errexit

# Run in debug mode, if set
if ${debug}; then set -x ; fi

# Bash will remember & return the highest exitcode in a chain of pipes.
# This way you can catch the error in case mysqldump fails in
`mysqldump |gzip`, for example.
set -o pipefail

# Validate all required params are present
if [[ -z ${BASTION_SUBNET_OCID} || -z ${ADMIN_SUBNET_OCID} || -z $
{WORKER_SUBNET_OCID} || -z ${FSS_SUBNET_OCID} || -z $
{LB_SUBNET_OCID} ]]
then
  echo "One or more required params are not specified. Please provide
either bastion and Admin subnet OCIDs"
  usage >&2
  exit
fi

vcn_ocid=$(oci network subnet get --subnet-id "${BASTION_SUBNET_OCID}"
| jq -r '.data["vcn-id"]')
vcn_cidr=$(oci network vcn get --vcn-id "${vcn_ocid}" | jq -r
'.data["cidr-block"]')

# Check if SSH port - 22 is open for access by Bastion Subnet
if [[ -n ${BASTION_SUBNET_OCID} || -n ${BASTION_HOST_IP_CIDR} ]]
then
  all_ips="0.0.0.0/0"
  res=$(validate_subnet_port_access "${BASTION_SUBNET_OCID}" "$
{SSH_PORT}" "${all_ips}")

  if [[ ${res} -ne 0 ]]
    then
      echo "ERROR: SSH port ${SSH_PORT} is not open for access by
[$all_ips] in -- ${BASTION_SUBNET_OCID}"
  fi

  # Check if bastion host IP is valid CIDR
  bastion_cidr_block=""
  if [[ -n ${BASTION_HOST_IP_CIDR} ]]
  then
    is_valid_cidr=$(is_valid_ip_cidr "${BASTION_HOST_IP_CIDR}")
    if [[ $is_valid_cidr -ne 0 ]]
```

```
    then
      echo "Bastion host IP CIDR is not valid: [${BASTION_HOST_IP_CIDR}]"
      usage >&2
      exit
    fi
    bastion_cidr_block=${BASTION_HOST_IP_CIDR}
  else
    bastion_cidr_block=$(oci network subnet get --subnet-id "$
{BASTION_SUBNET_OCID}" | jq -r '.data["cidr-block"]')
  fi

  # Check if bastion CIDR has access to SSH port on ADMIN subnet
  res=$(validate_subnet_port_access "${ADMIN_SUBNET_OCID}" "${SSH_PORT}" "$
{bastion_cidr_block}")

  if [[ $res -ne 0 ]]
  then
    echo "WARNING: SSH port ${SSH_PORT} is not open for access by Bastion
Subnet CIDR [$bastion_cidr_block] in private Admin Subnet
[$ADMIN_SUBNET_OCID]"
  fi
fi

# Check if service or NAT gateway exists in ADMIN & WORKER subnet's VCN.
if [[ -n ${ADMIN_SUBNET_OCID} && -n ${WORKER_SUBNET_OCID} ]]
then
  subnet_names=('ADMIN_SUBNET' 'WORKER_SUBNET')
  i=0
  for subnet_ocid in ${ADMIN_SUBNET_OCID} ${WORKER_SUBNET_OCID}; do
    res=$(validate_service_or_nat_gw_exist "${subnet_ocid}")
    if [[ $res -eq 1 ]]
    then
      echo "ERROR: Missing Service or NAT gateway in the VCN of the private $
{subnet_names[i]} subnet ocid [$subnet_ocid]"
    elif [[ $res -eq 2 ]]
    then
      echo "ERROR: Private ${subnet_names[i]} subnet [$subnet_ocid] does not
use NAT or Service gateway"
    elif [[ $res -eq 3 ]]
    then
      echo "ERROR: Service Gateway in VCN of private ${subnet_names[i]}
subnet [$subnet_ocid] does not allow access to all services in Oracle
services network"
    elif [[ $res -eq 4 ]]
    then
      echo "ERROR: Route Rule of private ${subnet_names[i]} subnet
[$subnet_ocid] does not use 'ALL Services in Oracle services network'
destination"
    fi
  done
fi

# Check if internet gateway exists in BASTION & LB & FSS subnet's VCN.
subnet_names=('BASTION_SUBNET' 'LB_SUBNET' 'FSS_SUBNET_OCID')
i=0
```

```
for subnet_ocid in ${BASTION_SUBNET_OCID} ${LB_SUBNET_OCID} $
{FSS_SUBNET_OCID}; do
  res=$(validate_internet_gw_exist "${subnet_ocid}")

  if [[ $res -ne 0 ]]
  then
    echo "WARNING: Missing internet gateway in the VCN of the $
{subnet_names[i]} subnet [$subnet_ocid]"
  fi
  i=$((i+1))
done

# Check if LB Subnet ports are open  0.0.0.0/0 all, 443, 80
all_ips="0.0.0.0/0"
for port in 'all' '443' '80'; do
  res=$(validate_subnet_port_access "${LB_SUBNET_OCID}" "${port}" "$
{all_ips}")

  if [[ $res -ne 0 ]]
  then
    echo "WARNING: Port [$port] is not open for 0.0.0.0/0 in LB Subnet
CIDR [${LB_SUBNET_OCID}]"
  fi
done

# Check if Worker Subnet all protocols are open for workers subnet
worker_subnet_cidr=$(oci network subnet get --subnet-id "$
{WORKER_SUBNET_OCID}" | jq -r '.data["cidr-block"]')
res=$(validate_subnet_port_access "${WORKER_SUBNET_OCID}" "all" $
{worker_subnet_cidr})

if [[ $res -ne 0 ]]
then
  echo "ERROR: All Protocols are not open for WORKER's Subnet CIDR [$
{worker_subnet_cidr}]"
fi

# FSS subnet verification - Checking All TCP Ports are open in FSS
SUBNET OCID for VCN CIDR
for port in '111' '2048' '2049' '2050'; do
  res=$(validate_subnet_port_access "${FSS_SUBNET_OCID}" "${port}" "$
{vcn_cidr}")
  if [[ $res -ne 0 ]]
  then
    echo "ERROR: TCP Port [${port}] is not open in FSS Subnet for VCN
CIDR"
  fi
done

# FSS subnet verification - UDP - '111' '2048' in FSS SUBNET OCID for
VCN CIDR"
for port in '111' '2048'; do
  res=$(validate_subnet_port_access "${FSS_SUBNET_OCID}" "${port}" "$
{vcn_cidr}" "UDP")
  if [[ $res -ne 0 ]]
```

```
  then
    echo "ERROR: UDP Port [${port}] is not open in FSS Subnet for VCN CIDR"
  fi
done

# Check if DB port is open for access by Worker's subnet CIDR in DB subnet
(only if DB subnet is provided)
if [[ -n ${DB_SUBNET_OCID} ]]
then
  res=$(validate_subnet_port_access ${DB_SUBNET_OCID} ${DB_PORT} ${vcn_cidr})
  res1=$(validate_subnet_port_access ${DB_SUBNET_OCID} ${DB_PORT} $
{worker_subnet_cidr})

  if [[ (${res} -ne 0) || (${res1} -ne 0) ]]
  then
    echo "ERROR: DB port ${DB_PORT} is not open for access by VCN CIDR
[$vcn_cidr] or Worker Subnet CIDR [$worker_subnet_cidr] in DB Subnet
[$DB_SUBNET_OCID]"
  fi
fi
```