Oracle® Cloud Native Environment

Getting Started for Release 1.2

F40897-04
December 2021

Oracle Legal Notices

Table of Contents

Preface

This document contains information about Oracle Cloud Native Environment. It includes information on installing and configuring Oracle Cloud Native Environment.

Document generated on: 2021-12-02 (revision: 1190)

Audience

This document is written for system administrators and developers who want to use Oracle Cloud Native Environment. It is assumed that readers have a general understanding of the Oracle Linux operating system and container concepts.

Related Documents

The latest version of this document and other documentation for this product are available at:

https://docs.oracle.com/en/operating-systems/olcne/

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

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

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture

that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Chapter 1 Oracle Cloud Native Environment Host Requirements

This chapter describes the hardware and operating system requirements for the hosts in Oracle Cloud Native Environment.

1.1 Hardware Requirements

Oracle Cloud Native Environment is a clustered environment that requires more than one node to form a cluster. Your environment should consist of two or more systems where Oracle Cloud Native Environment is installed.

You can install Oracle Cloud Native Environment on any of the following server types:

- Bare-metal server
- Oracle Linux Kernel-based Virtual Machine (KVM) instance
- Oracle Cloud Infrastructure bare-metal instance
- Oracle Cloud Infrastructure virtual instance
- Oracle Private Cloud Appliance virtual instance
- Oracle Private Cloud at Customer virtual instance

Oracle Cloud Native Environment is available for 64-bit x86 hardware only.

Oracle Cloud Native Environment does not require specific hardware; however, certain operations are CPU and memory intensive. For a list of certified bare-metal servers, see the Oracle Linux Hardware Certification List at:

https://linux.oracle.com/hardware-certifications

For information on the current Oracle x86 Servers, see:

https://www.oracle.com/servers/x86/

For information on creating an Oracle Linux KVM instance, see *Oracle® Linux: KVM User's Guide*.

The installation instructions for Oracle Private Cloud Appliance and Oracle Private Cloud at Customer, as well as information about the Oracle Cloud Native Environment releases that can be installed, are available in the Oracle Private Cloud Appliance and Oracle Private Cloud at Customer documentation at:

https://docs.oracle.com/en/engineered-systems/private-cloud-appliance/

The hardware requirements listed here are for the absolute minimum to run Oracle Cloud Native Environment. Your deployment is highly likely to require nodes with a larger footprint.

1.1.1 Kubernetes Control Plane Node Hardware

A minimum Kubernetes control plane node configuration is:

- 4 CPU cores (Intel VT-capable CPU)
- 16GB RAM

- 1GB Ethernet NIC
- XFS file system (the default file system for Oracle Linux)
- 40GB hard disk space in the `/var` directory

1.1.2 Kubernetes Worker Node Hardware

A minimum Kubernetes worker node configuration is:

- 1 CPU cores (Intel VT-capable CPU)
- 8GB RAM
- 1GB Ethernet NIC
- XFS file system (the default file system for Oracle Linux)
- 15GB hard disk space in the `/var` directory

1.1.3 Operator Node Hardware

A minimum operator node configuration is:

- 1 CPU cores (Intel VT-capable CPU)
- 8GB RAM
- 1GB Ethernet NIC
- 15GB hard disk space in the `/var` directory

1.1.4 Kubernetes High Availability Requirements

A minimum high availability (HA) configuration for a Kubernetes cluster is:

- 3 Kubernetes control plane nodes. At least 5 control plane nodes is recommended.
- 2 Kubernetes worker nodes. At least 3 worker nodes is recommended.

Important

The number of control plane nodes must be an odd number equal to or greater than three, for example, 3, 5, or 7.

1.1.5 Istio Requirements

A minimum configuration for deploying the Istio module for Oracle Cloud Native Environment is:

- 1 Kubernetes control plane node
- 2 Kubernetes worker nodes

These requirements are the minimum needed to successfully deploy Istio into a Kubernetes cluster. However, as your cluster expands and more nodes are added, Istio requires additional hardware

resources. For information on the hardware requirements of Istio, see the upstream documentation at: https://istio.io/latest/docs/ops/deployment/performance-and-scalability/.

1.2 Operating System Requirements

Oracle Cloud Native Environment is available for the following operating systems:

- Oracle Linux 7 (x86_64) running the Unbreakable Enterprise Kernel Release 5 (UEK R5) or Unbreakable Enterprise Kernel Release 6 (UEK R6). A minimum of Oracle Linux 7.5 is required.
- Oracle Linux 8 (x86_64) running the Unbreakable Enterprise Kernel Release 6 (UEK R6). A minimum of Oracle Linux 8.3 is required.

Chapter 2 Oracle Cloud Native Environment Prerequisites

This chapter describes the prerequisites for the systems to be used in an installation of Oracle Cloud Native Environment. This chapter also discusses how to enable the repositories to install the Oracle Cloud Native Environment packages.

2.1 Enabling Access to the Oracle Cloud Native Environment Packages

This section contains information on setting up the locations for the operating system on which you want to install the Oracle Cloud Native Environment software packages.

2.1.1 Oracle Linux 7

The Oracle Cloud Native Environment packages for Oracle Linux 7 are available on the Oracle Linux yum server in the `ol7_olcne11` and `ol7_olcne12` repositories, or on the Unbreakable Linux Network (ULN) in the `ol7_x86_64_olcne11` and `ol7_x86_64_olcne12` channels. However there are also dependencies across other repositories and channels, and these must also be enabled on each system where Oracle Cloud Native Environment is installed.

Warning

Oracle does not support Kubernetes on systems where the `ol7_preview`, `ol7_developer` or `ol7_developer_EPEL` yum repositories or ULN channels are enabled, or where software from these repositories or channels is currently installed on the systems where Kubernetes runs. Even if you follow the instructions in this document, you may render your platform unsupported if these repositories or channels are enabled or software from these channels or repositories is installed on your system.

2.1.1.1 Enabling Channels with ULN

If you are registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channels.

To subscribe to the ULN channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels.

Oracle Cloud Native Environment Release 1.2

To install Oracle Cloud Native Environment Release 1.2, subscribe the system to the following channels:

- `ol7_x86_64_olcne12`

- ol7_x86_64_kvm_utils
- ol7_x86_64_addons
- ol7_x86_64_latest
- ol7_x86_64_UEKR5 or ol7_x86_64_UEKR6

Make sure the systems are not subscribed to the following channels:

- ol7_x86_64_olcne
- ol7_x86_64_olcne11
- ol7_x86_64_developer

Oracle Cloud Native Environment Release 1.1

To install Oracle Cloud Native Environment Release 1.1, subscribe the system to the following channels:

- ol7_x86_64_olcne11
- ol7_x86_64_kvm_utils
- ol7_x86_64_addons
- ol7_x86_64_latest
- ol7_x86_64_UEKR5 or ol7_x86_64_UEKR6

Make sure the systems are not subscribed to the following channels:

- ol7_x86_64_olcne
- ol7_x86_64_olcne12
- ol7_x86_64_developer

5. Click **Save Subscriptions**.

2.1.1.2 Enabling Repositories with the Oracle Linux Yum Server

If you are using the Oracle Linux yum server for system updates, enable the required yum repositories.

To enable the yum repositories:

1. Install the oracle-olcne-release-el7 release package to install the Oracle Cloud Native Environment yum repository configuration.

```
sudo yum install oracle-olcne-release-el7
```

2. Set up the repositories for the release you want to install.

Oracle Cloud Native Environment Release 1.2

To install Oracle Cloud Native Environment Release 1.2, enable the following yum repositories:

- ol7_olcne12
- ol7_kvm_utils
- ol7_addons
- ol7_latest
- ol7_UEKR5 or ol7_UEKR6

Use the yum-config-manager tool to enable the yum repositories:

```
sudo yum-config-manager --enable ol7_olcne12 ol7_kvm_utils ol7_addons ol7_latest
```

Make sure the ol7_olcne, ol7_olcne11, and ol7_developer yum repositories are disabled:

```
sudo yum-config-manager --disable ol7_olcne ol7_olcne11 ol7_developer
```

Oracle Cloud Native Environment Release 1.1

To install Oracle Cloud Native Environment Release 1.1, enable the following yum repositories:

- ol7_olcne11
- ol7_kvm_utils
- ol7_addons
- ol7_latest
- ol7_UEKR5 or ol7_UEKR6

Use the yum-config-manager tool to enable the yum repositories:

```
sudo yum-config-manager --enable ol7_olcne11 ol7_kvm_utils ol7_addons ol7_latest
```

Make sure the ol7_olcne and ol7_olcne12, and and ol7_developer yum repositories are disabled:

```
sudo yum-config-manager --disable ol7_olcne ol7_olcne12 ol7_developer
```

3. Use yum-config-manager to enable the UEK repository for your kernel:

```
sudo yum-config-manager --enable ol7_UEKR5
```

or

```
sudo yum-config-manager --enable ol7_UEKR6
```

2.1.2 Oracle Linux 8

The Oracle Cloud Native Environment packages for Oracle Linux 8 are available on the Oracle Linux yum server in the ol8_olcne12 repository, or on the Unbreakable Linux Network (ULN) in the ol8_x86_64_olcne12 channel. However there are also dependencies across other repositories and channels, and these must also be enabled on each system where Oracle Cloud Native Environment is installed.

Warning

Oracle does not support Kubernetes on systems where the `ol8_developer` or `ol8_developer_EPEL` yum repositories or ULN channels are enabled, or where software from these repositories or channels is currently installed on the systems where Kubernetes runs. Even if you follow the instructions in this document, you may render your platform unsupported if these repositories or channels are enabled or software from these channels or repositories is installed on your system.

2.1.2.1 Enabling Channels with ULN

If you are registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channels.

To subscribe to the ULN channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels. Subscribe the system to the following channels:
   - `ol8_x86_64_olcne12`
   - `ol8_x86_64_addons`
   - `ol8_x86_64_baseos_latest`
   - `ol8_x86_64_UEKR6`
5. Click **Save Subscriptions**.

2.1.2.2 Enabling Repositories with the Oracle Linux Yum Server

If you are using the Oracle Linux yum server for system updates, enable the required yum repositories.

To enable the yum repositories:

1. Install the `oracle-olcne-release-el8` release package to install the Oracle Cloud Native Environment yum repository configuration.

   ```
   sudo dnf install oracle-olcne-release-el8
   ```

2. Enable the following yum repositories:
   - `ol8_olcne12`
   - `ol8_addons`
   - `ol8_baseos_latest`
   - `ol8_UEKR6`

   Use the `dnf config-manager` tool to enable the yum repositories:

```
sudo dnf config-manager --enable ol8_olcne12 ol8_addons ol8_baseos_latest ol8_UEKR6
```

2.2 Accessing the Oracle Container Registry

The container images that are deployed by the Platform CLI are hosted on the Oracle Container Registry. For more information about the Oracle Container Registry, see the *Oracle® Linux: Oracle Container Runtime for Docker User's Guide*.

For a deployment to use the Oracle Container Registry, each node within the environment must be provisioned with direct access to the Internet.

You can optionally use an Oracle Container Registry mirror, or create your own private registry mirror within your network.

When you create a Kubernetes module you must specify the registry from which to pull the container images. This is set using the `--container-registry` option of the `olcnectl module create` command. If you use the Oracle Container Registry the container registry must be set to:

```
container-registry.oracle.com/olcne
```

If you use a private registry that mirrors the Oracle Cloud Native Environment container images on the Oracle Container Registry, make sure you set the container registry to the domain name and port of the private registry, for example:

```
myregistry.example.com:5000/olcne
```

When you set the container registry to use during an installation, it becomes the default registry from which to pull images during updates and upgrades of the Kubernetes module. You can set a new default value during an update or upgrade using the `--container-registry` option.

2.2.1 Using an Oracle Container Registry Mirror

The Oracle Container Registry has many mirror servers located around the world. You can use a registry mirror in your global region to improve download performance of container images. While the Oracle Container Registry mirrors are hosted on Oracle Cloud Infrastructure, they are also accessible external to Oracle Cloud Infrastructure. Using a mirror that is closest to your geographical location should result in faster download speeds.

To use an Oracle Container Registry mirror to pull images, use the format:

```
container-registry-region-key.oracle.com/olcne
```

For example, to use the Oracle Container Registry mirror in the US East (Ashburn) region, which has a region key of `IAD`, the registry should be set (using the using the `--container-registry` option) to:

```
container-registry-iad.oracle.com/olcne
```

For more information on Oracle Container Registry mirrors and finding the region key for a mirror in your location, see the Oracle Cloud Infrastructure documentation at:

https://docs.cloud.oracle.com/iaas/Content/General/Concepts/regions.htm

2.2.2 Using a Private Registry

In some cases, nodes within your environment may not be provisioned with direct access to the Internet. In these cases, you can use a private registry that mirrors the Oracle Cloud Native Environment container

images on the Oracle Container Registry. Each node requires direct access to the mirror registry host in this scenario.

You can use an existing container registry in your network, or create a private registry using Oracle Container Runtime for Docker. If you use an existing private container registry, skip the first step in the following procedure that creates a Docker registry.

To create a private registry:

1. Select a host to use for your Oracle Container Registry mirror service. The mirror host must have access to the Internet and should be able to pull images directly from the Oracle Container Registry, or alternately should have access to the correct image files stored locally. Ideally, the host should not be a node within your Oracle Cloud Native Environment, but should be accessible to all of the nodes that are part of the environment.

   On the mirror host, install Oracle Container Runtime for Docker, and set up a Docker registry container, following the instructions in the *Oracle® Linux: Oracle Container Runtime for Docker User's Guide*.

2. On the mirror host, enable access to the Oracle Cloud Native Environment software packages. For information on enabling access to the packages, see Section 2.1, “Enabling Access to the Oracle Cloud Native Environment Packages”.

3. Install the `olcne-utils` package so you have access to the registry mirroring utility.

   ```
   sudo yum install olcne-utils
   ```

4. Install the `podman` package.

   ```
   sudo yum install podman
   ```

5. Copy the required container images from the Oracle Container Registry to the private registry using the `registry-image-helper.sh` script with the required options:

   ```
   registry-image-helper.sh --to host.example.com:5000/olcne
   ```

   Where *`host.example.com:5000`* is the resolvable domain name and port on which your private registry is available.

   You can optionally use the `--from` option to specify an alternate registry from which to pull the images. For example, to pull the images from an Oracle Container Registry mirror:

   ```
   registry-image-helper.sh --from container-registry-iad.oracle.com/olcne --to host.example.com:5000/olcne
   ```

   If the host where you are running the script does not have access to the Internet, you can replace the `--from` option with the `--local` option to load the container images directly from a local directory. The local directory which contains the images should be either:

   - `/usr/local/share/kubeadm/`
   - `/usr/local/share/olcne/`

   The image files should be archives in TAR format. All TAR files in the directory are loaded into the private registry when the script is run with the `--local` option.

   You can use the `--version` option to specify the Kubernetes version you want to mirror. If not specified, the latest release is used. The available versions you can pull are those listed in *Release Notes*.

2.3 Setting up the Operating System

The following sections describe the requirements that must be met to install and configure Oracle Cloud Native Environment on Oracle Linux 7 and Oracle Linux 8 systems.

2.3.1 Setting up a Network Time Service

As a clustering environment, Oracle Cloud Native Environment requires that the system time is synchronized across each Kubernetes control plane and worker node within the cluster. Typically, this can be achieved by installing and configuring a Network Time Protocol (NTP) daemon on each node. Oracle recommends installing and setting up the chronyd daemon for this purpose.

The `chronyd` service is enabled and started by default on Oracle Linux 8 systems.

To set up chronyd on Oracle Linux 7:

1. On each Kubernetes control plane and worker node, install the `chrony` package, if it is not already installed:

   ```
   sudo yum install chrony
   ```

2. Edit the NTP configuration in `/etc/chrony.conf`. Your requirements may vary. If you are using DHCP to configure the networking for each node, it is possible to configure NTP servers automatically. If you have not got a locally configured NTP service that your systems can sync to, and your systems have Internet access, you can configure them to use the public `pool.ntp.org` service. See https://www.ntppool.org/.

3. Make sure NTP is enabled to restart at boot and that it is started before you proceed with the Oracle Cloud Native Environment installation. For example:

   ```
   sudo systemctl enable --now chronyd.service
   ```

For information on configuring a Network Time Service, see the *Oracle® Linux 7: Administrator's Guide*.

2.3.2 Disabling Swap

You must disable swap on the Kubernetes control plane and worker nodes. To disable swap, enter:

```
sudo swapoff -a
```

To make this permanent over reboots, edit the `/etc/fstab` file to remove or comment out any swap disks.

2.3.3 Setting SELinux

SELinux runs in one of three modes, `disabled`, `enforcing` or `permissive`. The default is `enforcing` and this is the recommended mode.

SELinux must be enabled on the Kubernetes control plane and worker nodes to allow containers to access the host file system, which is required by pod networks. Oracle Cloud Native Environment can be run with either `permissive` or `enforcing` modes. Oracle Cloud Native Environment expects SELinux to be run in `permissive` mode by default, but it is recommended you use the operating system default of `enforcing`.

You can find the current setting on a host using:

```
sudo getenforce
```

For more information about SELinux, see *Oracle® Linux: Administering SELinux*.

Note

You can change the SELinux mode after you install a Kubernetes cluster. If you do this, you need to also update the Kubernetes module so the Platform API Server knows of the change. For information on changing the SELinux mode after you install a Kubernetes cluster, see *Container Orchestration*.

SELinux Permissive Mode

Oracle Cloud Native Environment expects SELinux to be set to `permissive` by default. You can set SELinux to `permissive` mode using:

```
sudo setenforce permissive
```

You must also set the SELinux mode over system restarts. Edit the `/etc/selinux/config` file and set the value of the `SELINUX` directive to `permissive`, or use `sed`:

```
sudo sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config
```

SELinux Enforcing Mode

You can optionally set SELinux to `enforcing` mode (the operating system default and the recommended mode) using:

```
sudo setenforce enforcing
```

You must also set the SELinux mode over system restarts. Edit the `/etc/selinux/config` file and set the value of the `SELINUX` directive to `enforcing`, or use `sed`:

```
sudo sed -i s/^SELINUX=.*$/SELINUX=enforcing/ /etc/selinux/config
```

If you set SELinux to `enforcing` you need to set this as an option using the Platform CLI when you create the Kubernetes module.

2.4 Setting up the Network

This section contains information about the networking requirements for Oracle Cloud Native Environment nodes.

The following table shows the network ports used by the services in a deployment of Kubernetes in an environment.

From Node Type	To Node Type	Port	Protocol	Reason
Worker	Operator	8091	TCP(6)	Platform API Server
Control plane	Operator	8091	TCP(6)	Platform API Server
Control plane	Control plane	2379-2380	TCP(6)	Kubernetes etcd (highly available clusters)

From Node Type	To Node Type	Port	Protocol	Reason
Operator	Control plane	6443	TCP(6)	Kubernetes API server
Worker	Control plane	6443	TCP(6)	Kubernetes API server
Control plane	Control plane	6443	TCP(6)	Kubernetes API server
Control plane	Control plane	6444	TCP(6)	Alternate Kubernetes API server (highly available clusters)
Operator	Control plane	8090	TCP(6)	Platform Agent
Control plane	Control plane	10250	TCP(6)	Kubernetes `kubelet` API server
		10251	TCP(6)	Kubernetes `kube-scheduler` (highly available clusters)
		10252	TCP(6)	
		10255	TCP(6)	Kubernetes `kube-controller-manager` (highly available clusters)
				Kubernetes `kubelet` API server for read-only access with no authentication
Control plane	Control plane	8472	UDP(11)	Flannel
Worker	Control plane	8472	UDP(11)	Flannel
Control plane	Control plane	N/A	VRRP(112)	Keepalived for Kubernetes API server (highly available clusters)
Operator	Worker	8090	TCP(6)	Platform Agent
Control plane	Worker	10250	TCP(6)	Kubernetes `kubelet` API server
		10255	TCP(6)	Kubernetes `kubelet` API server for read-only access with no authentication
Control plane	Worker	8472	UDP(11)	Flannel
Worker	Worker	8472	UDP(11)	Flannel

The following sections show you how to set up the network on each node to enable the communication between nodes in an environment.

2.4.1 Setting up the Firewall Rules

Oracle Linux 7 installs and enables `firewalld`, by default. The Platform CLI notifies you of any rules that you may need to add during the deployment of the Kubernetes module. The Platform CLI also provides the commands to run to modify your firewall configuration to meet the requirements.

Make sure that all required ports are open. The ports required for a Kubernetes deployment are:

- 2379/tcp: Kubernetes etcd server client API (on control plane nodes in highly available clusters)
- 2380/tcp: Kubernetes etcd server client API (on control plane nodes in highly available clusters)
- 6443/tcp: Kubernetes API server (control plane nodes)

- 8090/tcp: Platform Agent (control plane and worker nodes)
- 8091/tcp: Platform API Server (operator node)
- 8472/udp: Flannel overlay network, VxLAN backend (control plane and worker nodes)
- 10250/tcp: Kubernetes `kubelet` API server (control plane and worker nodes)
- 10251/tcp: Kubernetes `kube-scheduler` (on control plane nodes in highly available clusters)
- 10252/tcp: Kubernetes `kube-controller-manager` (on control plane nodes in highly available clusters)
- 10255/tcp: Kubernetes `kubelet` API server for read-only access with no authentication (control plane and worker nodes)

The commands to open the ports and to set up the firewall rules are provided below.

2.4.1.1 Non-HA Cluster Firewall Rules

For a cluster with a single control plane node, the following ports are required to be open in the firewall.

Operator Node

On the operator node, run:

```
sudo firewall-cmd --add-port=8091/tcp --permanent
```

Restart the firewall for these rules to take effect:

```
sudo systemctl restart firewalld.service
```

Worker Nodes

On the Kubernetes worker nodes run:

```
sudo firewall-cmd --zone=trusted --add-interface=cni0 --permanent
sudo firewall-cmd --add-port=8090/tcp --permanent
sudo firewall-cmd --add-port=10250/tcp --permanent
sudo firewall-cmd --add-port=10255/tcp --permanent
sudo firewall-cmd --add-port=8472/udp --permanent
```

If you are installing Oracle Cloud Native Environment Release 1.2.3 or earlier on Oracle Linux 7, you also need to enable masquerading. This is not required for all other installation types. On the Kubernetes worker nodes run:

```
sudo firewall-cmd --add-masquerade --permanent
```

Restart the firewall for these rules to take effect:

```
sudo systemctl restart firewalld.service
```

Control Plane Nodes

On the Kubernetes control plane nodes run:

```
sudo firewall-cmd --zone=trusted --add-interface=cni0 --permanent
sudo firewall-cmd --add-port=8090/tcp --permanent
```

```
sudo firewall-cmd --add-port=10250/tcp --permanent
sudo firewall-cmd --add-port=10255/tcp --permanent
sudo firewall-cmd --add-port=8472/udp --permanent
sudo firewall-cmd --add-port=6443/tcp --permanent
```

If you are installing Oracle Cloud Native Environment Release 1.2.3 or earlier on Oracle Linux 7, you also need to enable masquerading. This is not required for all other installation types. On the Kubernetes control plane nodes run:

```
sudo firewall-cmd --add-masquerade --permanent
```

Restart the firewall for these rules to take effect:

```
sudo systemctl restart firewalld.service
```

2.4.1.2 Highly Available Cluster Firewall Rules

For a highly available cluster, open all the firewall ports as described in Section 2.4.1.1, “Non-HA Cluster Firewall Rules”, along with the following **additional** ports on the control plane node.

On the Kubernetes control plane nodes run:

```
sudo firewall-cmd --add-port=10251/tcp --permanent
sudo firewall-cmd --add-port=10252/tcp --permanent
sudo firewall-cmd --add-port=2379/tcp --permanent
sudo firewall-cmd --add-port=2380/tcp --permanent
```

Restart the firewall for these rules to take effect:

```
sudo systemctl restart firewalld.service
```

2.4.2 Setting up Other Network Options

This section contains information on other network related configuration that affects an Oracle Cloud Native Environment deployment. You may not need to make changes from this section, but they are provided to help you understand any issues you may encounter related to network configuration.

2.4.2.1 Internet Access

The Platform CLI checks it is able to access the container registry, and possibly other Internet resources, to be able to pull any required container images. Unless you intend to set up a local registry mirror for container images, the systems where you intend to install Oracle Cloud Native Environment must either have direct internet access, or must be configured to use a proxy.

2.4.2.2 Flannel Network

The Platform CLI configures a flannel network as the network fabric used for communications between Kubernetes pods. This overlay network uses VxLANs to facilitate network connectivity. For more information on flannel, see the upstream documentation at:

https://github.com/coreos/flannel

By default, the Platform CLI creates a network in the `10.244.0.0/16` range to host this network. The Platform CLI provides an option to set the network range to an alternate range, if required, during installation. Systems in an Oracle Cloud Native Environment deployment must not have any network devices configured for this reserved IP range.

2.4.2.3 br_netfilter Module

The Platform CLI checks whether the `br_netfilter` module is loaded and exits if it is not available. This module is required to enable transparent masquerading and to facilitate Virtual Extensible LAN (VxLAN) traffic for communication between Kubernetes pods across the cluster. If you need to check whether it is loaded, run:

```
sudo lsmod|grep br_netfilter

br_netfilter           24576  0
bridge                155648  2 br_netfilter,ebtable_broute
```

If you see the output similar to shown, the `br_netfilter` module is loaded. Kernel modules are usually loaded as they are needed, and it is unlikely that you need to load this module manually. If necessary, you can load the module manually and add it as a permanent module by running:

```
sudo modprobe br_netfilter
sudo sh -c 'echo "br_netfilter" > /etc/modules-load.d/br_netfilter.conf'
```

2.4.2.4 Bridge Tunable Parameters

Kubernetes requires that packets traversing a network bridge are processed for filtering and for port forwarding. To achieve this, tunable parameters in the kernel bridge module are automatically set when the `kubeadm` package is installed and a sysctl file is created at `/etc/sysctl.d/k8s.conf` that contains the following lines:

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

If you modify this file, or create anything similar yourself, run the following command to load the bridge tunable parameters:

```
sudo /sbin/sysctl -p /etc/sysctl.d/k8s.conf
```

2.4.2.5 Network Address Translation

Network Address Translation (NAT) is sometimes required when one or more Kubernetes worker nodes in a cluster are behind a NAT gateway. For example, you might want to have a control plane node in a secure company network while having other worker nodes in a publicly accessible demilitarized zone which is less secure. The control plane node would access the worker nodes through the worker node's NAT gateway. Or you may have a worker node in a legacy network that you want to use in your cluster that is primarily on a newer network. The NAT gateway, in these cases, translates requests for an IP address accessible to the Kubernetes cluster into the IP address on the subnet behind the NAT gateway.

Note

Only worker nodes can be behind a NAT. Control plane nodes cannot be behind a NAT.

Regardless of what switches or network equipment you use to set up your NAT gateway, you must configure the following for a node behind a NAT gateway:

- The node's interface behind the NAT gateway must have an public IP address using the `/32` subnet mask that is reachable by the Kubernetes cluster. The `/32` subnet restricts the subnet to one IP address, so that all traffic from the Kubernetes cluster flows through this public IP address.

- The node's interface must also include a private IP address behind the NAT gateway that your switch uses NAT tables to match the public IP address to.

For example, you can use the following command to add the reachable IP address on the `ens5` interface:

```
sudo ip addr add 192.168.64.6/32 dev ens5
```

You can then use the following command to add the private IP address on the same interface:

```
sudo ip addr add 192.168.192.2/18 dev ens5
```

2.5 Setting FIPS Mode

You can optionally configure Oracle Cloud Native Environment operator, control plane, and worker hosts to run in Federal Information Processing Standards (FIPS) mode as described in *Oracle® Linux 8: Enhancing System Security*. Oracle Cloud Native Environment uses the cryptographic binaries of OpenSSL from Oracle Linux 8 when the host runs in FIPS mode.

Note

You cannot use Oracle Cloud Native Environment on Oracle Linux 7 hosts running in FIPS mode.

Chapter 3 Installing Oracle Cloud Native Environment

This chapter discusses how to prepare the nodes to be used in an Oracle Cloud Native Environment deployment. When the nodes are prepared, they must be installed with the Oracle Cloud Native Environment software packages. When the nodes are set up with the software, you can use the Platform CLI to perform a deployment of a Kubernetes cluster and optionally a service mesh.

This chapter shows you how to perform the steps to set up the hosts and install the Oracle Cloud Native Environment software, ready to perform a deployment of modules. When you have set up the nodes, deploy the Kubernetes module to install a Kubernetes cluster using the steps in *Container Orchestration*.

3.1 Installation Overview

The high level overview of setting up Oracle Cloud Native Environment is described in this section.

To install Oracle Cloud Native Environment:

1. **Prepare the operator node**: An *operator* node is a host that is used to perform and manage the deployment of environments. The operator node must be set up with the Platform API Server, and the Platform CLI (`olcnectl`).
2. **Prepare the Kubernetes nodes**: The Kubernetes control plane and worker nodes must to be set up with the Platform Agent.
3. **Set up a load balancer**: If you are deploying a highly available Kubernetes cluster, set up a load balancer. You can set up your own load balancer, or use the container-based load balancer deployed by the Platform CLI.
4. **Set up X.509 Certificates**: X.509 Certificates are used to provide secure communication between the Kubernetes nodes. You must set up the certificates before you create an environment and perform a deployment.
5. **Start the services**: Start the Platform API Server and Platform Agent services on nodes using the X.509 Certificates.
6. **Create an environment**: Create an environment into which you can install the Kubernetes module and any other optional modules.

3.2 Setting up the Nodes

This section discusses setting up nodes to use in an Oracle Cloud Native Environment. The nodes are used to form a Kubernetes cluster.

An *operator* node should be used to perform the deployment of the Kubernetes cluster using the Platform CLI and the Platform API Server. An operator node may be a node in the Kubernetes cluster, or a separate host. In examples in this book, the operator node is a separate host, and not part of the Kubernetes cluster.

On each Kubernetes node (both control plane and worker nodes) the Platform Agent must be installed. Before you set up the Kubernetes nodes, you must prepare them. For information on preparing the nodes, see Chapter 2, *Oracle Cloud Native Environment Prerequisites*.

During the installation of the required packages on, an `olcne` user is created. This user is used to start the Platform API Server or Platform Agent services and has the minimum operating system privileges to perform that task. The `olcne` user should not be used for any other purpose.

3.2.1 Setting up the Operator Node

This section discusses setting up the operator node. The *operator* node is a host that is used to perform and manage the deployment of environments, including deploying the Kubernetes cluster.

To set up the operator node:

1. On the operator node, install the Platform CLI, Platform API Server, and utilities.

   On Oracle Linux 7 enter:

   ```
   sudo yum install olcnectl olcne-api-server olcne-utils
   ```

   On Oracle Linux 8 enter:

   ```
   sudo dnf install olcnectl olcne-api-server olcne-utils
   ```

2. Enable the `olcne-api-server` service, but do *not* start it. The `olcne-api-server` service is started when you configure the X.509 Certificates.

   ```
   sudo systemctl enable olcne-api-server.service
   ```

   For information on configuration options for the Platform API Server, see Section 4.1, “Configuring the Platform API Server”.

3.2.2 Setting up Kubernetes Nodes

This section discusses setting up the nodes to use in a Kubernetes cluster. Perform these steps on both Kubernetes control plane and worker nodes.

To set up the Kubernetes nodes:

1. On each node to be added to the Kubernetes cluster, install the Platform Agent package and utilities.

   On Oracle Linux 7 enter:

   ```
   sudo yum install olcne-agent olcne-utils
   ```

   On Oracle Linux 8 enter:

   ```
   sudo dnf install olcne-agent olcne-utils
   ```

2. Enable the `olcne-agent` service, but do *not* start it. The `olcne-agent` service is started when you configure the X.509 Certificates.

   ```
   sudo systemctl enable olcne-agent.service
   ```

   For information on configuration options for the Platform Agent, see Section 4.2, “Configuring the Platform Agent”.

3. If you use a proxy server, configure it with CRI-O. On each Kubernetes node, create a CRI-O `systemd` configuration directory:

   ```
   sudo mkdir /etc/systemd/system/crio.service.d
   ```

   Create a file named `proxy.conf` in the directory, and add the proxy server information. For example:

   ```
   [Service]
   Environment="HTTP_PROXY=proxy.example.com:3128"
   Environment="HTTPS_PROXY=proxy.example.com:3128"
   ```

```
Environment="NO_PROXY=mydomain.example.com"
```

4. If the `docker` service is running, stop and disable it.

```
sudo systemctl disable --now docker.service
```

5. If the `containerd` service is running, stop and disable it.

```
sudo systemctl disable --now containerd.service
```

3.3 Setting up a Load Balancer for Highly Available Clusters

A highly available (HA) cluster needs a load balancer to provide high availability of control plane nodes. A load balancer communicates with the Kubernetes API server on the control plane nodes.

There are two methods of setting up a load balancer to create an HA cluster:

- Using your own external load balancer instance
- Using the load balancer that can be deployed by the Platform CLI on the control plane nodes

3.3.1 Setting up Your Own Load Balancer

If you want to use your own load balancer implementation, it should be set up and ready to use before you perform an HA cluster deployment. The load balancer hostname and port is entered as an option when you create the Kubernetes module. The load balancer should be set up with the following configuration:

- The listener listening on TCP port 6443.
- The distribution set to round robin.
- The target set to TCP port 6443 on the control plane nodes.
- The health check set to TCP.

For more information on setting up your own load balancer, see the *Oracle® Linux 7: Administrator's Guide*, or *Oracle® Linux 8: Setting Up Load Balancing*.

If you are deploying to Oracle Cloud Infrastructure, set up a load balancer.

To set up a load balancer on Oracle Cloud Infrastructure:

1. Create a load balancer.
2. Add a backend set to the load balancer using weighted round robin. Set the health check to be TCP port 6443.
3. Add the control plane nodes to the backend set. Set the port for the control plane nodes to port 6443.
4. Create a listener for the backend set using TCP port 6443.

For more information on setting up a load balancer in Oracle Cloud Infrastructure, see the Oracle Cloud Infrastructure documentation.

3.3.2 Setting up the In-built Load Balancer

If you want to use the in-built load balancer that can be deployed by the Platform CLI, you need to perform the following steps to prepare the control plane nodes. These steps should be performed on each control plane node.

To prepare control plane nodes for the load balancer deployed by the Platform CLI:

1. Set up the control plane nodes as described in Section 3.2.2, “Setting up Kubernetes Nodes”.
2. Nominate a virtual IP address that can be used for the primary control plane node. This IP address should not be in use on any node, and is assigned dynamically to the control plane node assigned as the primary controller by the load balancer. If the primary node fails, the load balancer reassigns the virtual IP address to another control plane node, and that, in turn, becomes the primary node. The virtual IP address used in examples in this documentation is `192.0.2.100`.
3. Open port 6444. When you use a virtual IP address, the Kubernetes API server port is changed from the default of 6443 to 6444. The load balancer listens on port 6443 and receives the requests and passes them to the Kubernetes API server.

   ```
   sudo firewall-cmd --add-port=6444/tcp
   sudo firewall-cmd --add-port=6444/tcp --permanent
   ```

4. Enable the Virtual Router Redundancy Protocol (VRRP) protocol:

   ```
   sudo firewall-cmd --add-protocol=vrrp
   sudo firewall-cmd --add-protocol=vrrp --permanent
   ```

3.4 Setting up X.509 Certificates for Kubernetes Nodes

Communication between the Kubernetes nodes is secured using X.509 certificates.

Before you deploy Kubernetes, you need to configure the X.509 certificates used to manage the communication between the nodes. There are a number of ways to manage and deploy the certificates. You can use:

- **Vault**: The certificates are managed using the HashiCorp Vault secrets manager. Certificates are created *during* the deployment of the Kubernetes module. You need to create a token authentication method for Oracle Cloud Native Environment.
- **CA Certificates**: Use your own certificates, signed by a trusted Certificate Authority (CA), and copied to each Kubernetes node *before* the deployment of the Kubernetes module. These certificates are unmanaged and must be renewed and updated manually.
- **Private CA Certificates**: Using generated certificates, signed by a private CA you set up, and copied to each Kubernetes node *before* the deployment of the Kubernetes module. These certificates are unmanaged and must be renewed and updated manually. A script is provided to help you set this up.

A software-based secrets manager is recommended to manage these certificates. The HashiCorp Vault secrets manager can be used to generate, assign and manage the certificates. Oracle recommends you implement your own instance of Vault, setting up the appropriate security for your environment.

For more information on installing and setting up Vault, see the HashiCorp documentation at:

https://learn.hashicorp.com/vault/operations/ops-deployment-guide

If you do not want to use Vault, you can use your own certificates, signed by a trusted CA, and copied to each node. A script is provided to generate a private CA which allows you to generate certificates for each node. This script also gives you the commands needed to copy the certificates to the nodes.

3.4.1 Setting up Vault Authentication

To configure Vault for use with Oracle Cloud Native Environment, set up a Vault token with the following properties:

- A PKI secret engine with a CA certificate or intermediate, located at `olcne_pki_intermediary`.
- A role under that PKI, named `olcne`, configured to not require a common name, and allow any name.
- A token authentication method and policy that attaches to the `olcne` role and can request certificates.

For information on setting up the Vault PKI secrets engine to generate dynamic X.509 certificates, see:

https://www.vaultproject.io/docs/secrets/pki/index.html

For information on creating Vault tokens, see:

https://www.vaultproject.io/docs/commands/token/create.html

3.4.2 Setting up CA Certificates

This section shows you how to use your own certificates, signed by a trusted CA, without using a secrets manager such as Vault. To use your own certificates, copy them to all Kubernetes nodes, and to the Platform API Server node.

To make sure the Platform Agent on each Kubernetes node, and the Platform API Server have access to certificates, make sure you copy them into the `/etc/olcne/certificates/` directory on each node. The path to the certificates is used when setting up the Platform Agent and Platform API Server, and when creating an environment.

The examples in this book use the `/etc/olcne/configs/certificates/production/` directory for certificates. For example:

- **CA Certificate**: `/etc/olcne/configs/certificates/production/ca.cert`
- **Node Key**: `/etc/olcne/configs/certificates/production/node.key`
- **Node Certificate**: `/etc/olcne/configs/certificates/production/node.cert`

3.4.3 Setting up Private CA Certificates

This section shows you how to create a private CA, and use that to generate signed certificates for the nodes. This section also contains information on copying the certificates to the nodes. Additionally this section contains information on generating additional certificates for nodes that you want to scale into a Kubernetes cluster.

3.4.3.1 Creating and Copying Certificates

This section shows you how to create a private CA, and use that to generate signed certificates for the nodes.

To generate certificates using a private CA:

1. (Optional) You can set up keyless SSH between the operator node and the Kubernetes nodes to make it easier to copy the certificates to the nodes. For information on setting up keyless SSH, see *Oracle® Linux: Connecting to Remote Systems With OpenSSH*.

2. Use the `/etc/olcne/gen-certs-helper.sh` script to generate a private CA and certificates for the nodes.

Tip

The `gen-certs-helper.sh` script saves the certificate files to the directory from which you run the script. The `gen-certs-helper.sh` script also creates a script you can use to copy the certificates to each Kubernetes node (`olcne-transfer-certs.sh`). If you run the `gen-certs-helper.sh` script from the `/etc/olcne` directory, it uses the default certificate directory used in this book (`/etc/olcne/certificates/`) when creating the `olcne-transfer-certs.sh` script. This means you can start up the Platform API Server, and the Platform Agent on Kubernetes nodes, using the default certificate directory locations as shown in this book. You could also use the `--cert-dir` option to specify the location to save the certificates and transfer script.

Provide the nodes for which you want to create certificates using the `--nodes` option. You should create a certificate for each node that runs the Platform API Server or Platform Agent. That is, for the operator node, and each Kubernetes node. If you are deploying a highly available Kubernetes cluster using a virtual IP address, you do not need to create a certificate for a virtual IP address.

Provide the private CA information using the `--cert-request*` options (some, but not all, of these options are shown in the example). You can get a list of all command options using the `gen-certs-helper.sh --help` command.

For example:

```
cd /etc/olcne
sudo ./gen-certs-helper.sh \
--cert-request-organization-unit "My Company Unit" \
--cert-request-organization "My Company" \
--cert-request-locality "My Town" \
--cert-request-state "My State" \
--cert-request-country US \
--cert-request-common-name cloud.example.com \
--nodes operator.example.com,control1.example.com,worker1.example.com,worker2.example.com,worker3.example.c
```

The certificates and keys for each node are generated and saved to the directory:

`/path/configs/certificates/tmp-olcne/node/`

Where *path* is the directory from which you ran the `gen-certs-helper.sh` script, or the location you set with the `--cert-dir` option; and *node* is the name of the node for which the certificate was generated.

The private CA certificate and key files are saved to the directory:

`/path/configs/certificates/production/`

3. Copy the certificate generated for a node from the `/path/configs/certificates/tmp-olcne/node/` directory to that node.

To make sure the Platform Agent on each Kubernetes node, and the Platform API Server have access to certificates, make sure you copy them into the `/etc/olcne/certificates/` directory on each

node. The path to the certificates is used when setting up the Platform Agent and Platform API Server, and when creating an environment.

The examples in this book use the `/etc/olcne/configs/certificates/production/` directory as the location for certificates on nodes.

A script is created to help you copy the certificates to the nodes, `/path/configs/certificates/olcne-transfer-certs.sh`. You can use this script and modify it to suit your needs, or transfer the certificates to the nodes using some other method.

Important

If you set up keyless SSH, change the `USER` variable in this script to the user you set up with keyless SSH.

Run the script to copy the certificates to the nodes:

```
bash -ex /path/configs/certificates/olcne-tranfer-certs.sh
```

4. Make sure the `olcne` user on each node that runs the Platform API Server or Platform Agent is able to read the directory in which you copy the certificates. If you used the default path for certificates of `/etc/olcne/certificates/`, the `olcne` user has read access.

   If you used a different path, check the `olcne` user can read the certificate path. On the operator node, and each Kubernetes node, run:

   ```
   sudo -u olcne ls /path/configs/certificates/production

   ca.cert  node.cert  node.key
   ```

   You should see a list of the certificates and key for the node.

3.4.3.2 Creating Additional Certificates

This section contains information about generating certificates for any additional nodes that you want to add to a Kubernetes cluster.

To generate additional certificates using a private CA:

1. On the operator node, generate new certificates for the nodes using the `/etc/olcne/gen-certs-helper.sh` script. For example:

   ```
   cd /etc/olcne
   sudo ./gen-certs-helper.sh \
   --cert-request-organization-unit "My Company Unit" \
   --cert-request-organization "My Company" \
   --cert-request-locality "My Town" \
   --cert-request-state "My State" \
   --cert-request-country US \
   --cert-request-common-name cloud.example.com \
   --nodes control4.example.com,control5.example.com \
   --byo-ca-cert /path/configs/certificates/production/ca.cert \
   --byo-ca-key /path/configs/certificates/production/ca.key
   ```

   The private key to generate the new certificates is specified with the `--byo-ca-key` option and the CA certificate with the `--byo-ca-cert` option. In this example, the private CA certificate and key files are located in the directory:

   `/path/configs/certificates/production/`

Where *path* is the directory from which you originally ran the gen-certs-helper.sh script (it is most likely /etc/olcne), or the location you set with the --cert-dir option if you used that option.

2. When you have generated the new certificates, copy them to the nodes. A script is created to help you copy the certificates to the nodes, */path*/configs/certificates/olcne-transfer-certs.sh. You can use this script and modify it to suit your needs, or transfer the certificates to the nodes using some other method.

   Run the script to copy the certificates to the nodes:

   ```
   bash -ex /path/configs/certificates/olcne-tranfer-certs.sh
   ```

3.5 Setting up X.509 Certificates for the externalIPs Kubernetes Service

Important

You do not need to perform the steps in this section if you are using Oracle Cloud Native Environment Releases 1.2.0 and 1.1.8 or lower. The set up steps in this section are for Releases 1.2.2 and 1.1.10 or later.

When you deploy Kubernetes, a service is deployed to the cluster that controls access to externalIPs in Kubernetes services. The service is named externalip-validation-webhook-service and runs in the externalip-validation-system namespace. This Kubernetes service requires X.509 certificates be set up prior to deploying Kubernetes. You can use Vault to generate the certificates, or use your own certificates for this purpose. You can also generate certificates using the gen-certs-helper.sh script. The certificates must be available on the operator node. The examples in this book use the /etc/olcne/configs/certificates/restrict_external_ip/production/ directory for these certificates.

3.5.1 Setting up Vault Certificates

You can use Vault to generate a certificates for the externalIPs Kubernetes service. The Vault instance must be configured in the same way as described in Section 3.4.1, “Setting up Vault Authentication”.

You need to generate certificates for two nodes, named:

externalip-validation-webhook-service.externalip-validation-system.svc

externalip-validation-webhook-service.externalip-validation-system.svc.cluster.local

The certificate information should be generated in PEM format.

For example:

```
vault write olcne_pki_intermediary/issue/olcne \
    alt_names=externalip-validation-webhook-service.externalip-validation-system.svc,externalip-validation-web
    format=pem_bundle
```

The output is displayed. Look for the section that starts with certificate. This section contains the certificates for the node names (set with the alt_names option). Save the output in this section to a file named node.cert. The file should look something like:

```
-----BEGIN RSA PRIVATE KEY-----
```

```
MIIEpQIBAAKCAQEAymg8uHy+mpwlelCyC4WrnfLwUmJ5vZmSos85QnIlZvyycUPK
...
X3c8LNaJDfQx1wKfTc/c0czBhHYxgwfau0G6wjqScZesPi2xY0xyslE=
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIID2TCCAsGgAwIBAgIUZ/M/D7bAjhyGx7DivsjBb9oeLhAwDQYJKoZIhvcNAQEL
...
9bRwnen+JrxUn4GV59GtsTiqzY6R2OKPm+zLl8E=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDnDCCAoSgAwIBAgIUMapl4aWnBXE/02qTW0zOZ9aQVGgwDQYJKoZIhvcNAQEL
...
kV8w2xVXXAehp7cg0BakVA==
-----END CERTIFICATE-----
```

Look for the section that starts with `issuing_ca`. This section contains the CA certificate. Save the output in this section to a file named `ca.cert`. The file should look something like:

```
-----BEGIN CERTIFICATE-----
MIIDnDCCAoSgAwIBAgIUMapl4aWnBXE/02qTW0zOZ9aQVGgwDQYJKoZIhvcNAQEL
...
kV8w2xVXXAehp7cg0BakVA==
-----END CERTIFICATE-----
```

Look for the section that starts with `private_key`. This section contains the private key for the node certificates. Save the output in this section to a file named `node.key`. The file should look something like:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAymg8uHy+mpwlelCyC4WrnfLwUmJ5vZmSos85QnIlZvyycUPK
...
X3c8LNaJDfQx1wKfTc/c0czBhHYxgwfau0G6wjqScZesPi2xY0xyslE=
-----END RSA PRIVATE KEY-----
```

Copy the three files (`node.cert`, `ca.cert` and `node.key`) to the operator node and set the ownership of the files as described in Section 3.5.2, “Setting up CA Certificates”.

3.5.2 Setting up CA Certificates

If you are using your own certificates, you should copy them to a directory under `/etc/olcne/certificates/` on the operator node. For example:

- **CA Certificate**: `/etc/olcne/configs/certificates/restrict_external_ip/production/ca.cert`
- **Node Key**: `/etc/olcne/configs/certificates/restrict_external_ip/production/node.key`
- **Node Certificate**: `/etc/olcne/configs/certificates/restrict_external_ip/production/node.cert`

You should copy these certificates to a different location on the operator node than the certificates and keys used for the Kubernetes nodes as set up in Section 3.4, “Setting up X.509 Certificates for Kubernetes Nodes”. This makes sure you do not overwrite those certificates and keys. You need to generate certificates for two nodes, named:

`externalip-validation-webhook-service.externalip-validation-system.svc`

`externalip-validation-webhook-service.externalip-validation-system.svc.cluster.local`

The certificates for these two nodes should be saved as a single file as `node.cert`.

Make sure the permissions of the output directory where the certificates are located can be read by the user on the operator node that you intend to use use to run the `olcnectl` commands to install Kubernetes. In this example the `opc` user is to be used on the operator node, so ownership of the directory is set to the `opc` user:

```
sudo chown -R opc:opc /etc/olcne/configs/certificates/restrict_external_ip/
```

3.5.3 Setting up Private CA Certificates

You can use the `gen-certs-helper.sh` script to generate the certificates. Run the script on the operator node and enter the options required for your environment.

The `--cert-dir` option sets the location where the certificates are to be saved.

The `--nodes` option must be set to the name of the Kubernetes service, as shown:

```
--nodes externalip-validation-webhook-service.externalip-validation-
system.svc,externalip-validation-webhook-service.externalip-validation-
system.svc.cluster.local
```

Use the `--one-cert` option to save the certificates for the two service names to a single file.

```
cd /etc/olcne
sudo ./gen-certs-helper.sh \
--cert-dir /etc/olcne/configs/certificates/restrict_external_ip/ \
--cert-request-organization-unit "My Company Unit" \
--cert-request-organization "My Company" \
--cert-request-locality "My Town" \
--cert-request-state "My State" \
--cert-request-country US \
--cert-request-common-name cloud.example.com \
--nodes externalip-validation-webhook-service.externalip-validation-system.svc,externalip-validation-webhook-s
--one-cert
```

You can use the same CA certificate and private key you used to generate the Kubernetes node certificates by using the `--byo-ca-cert` and `--byo-ca-key` options. For example, add the following lines to the command:

```
--byo-ca-cert /path/configs/certificates/production/ca.cert \
--byo-ca-key /path/configs/certificates/production/ca.key
```

Make sure the permissions of the output directory where the certificates are located can be read by the user on the operator node that you intend to use use to run the `olcnectl` commands to install Kubernetes. In this example the `opc` user is to be used on the operator node, so ownership of the directory is set to the `opc` user:

```
sudo chown -R opc:opc /etc/olcne/configs/certificates/restrict_external_ip/
```

3.6 Starting the Platform API Server and Platform Agent Services

This section discusses using certificates to set up secure communication between the Platform API Server and the Platform Agent on nodes in the cluster. You can set up secure communication using certificates managed by Vault, or using your own certificates copied to each node. You must configure the Platform API Server and the Platform Agent to use the certificates when you start the services.

For information on setting up the certificates with Vault, see Section 3.4, “Setting up X.509 Certificates for Kubernetes Nodes”.

For information on creating a private CA to sign certificates that can be used during testing, see Section 3.4.3, “Setting up Private CA Certificates”.

3.6.1 Starting the Services Using Vault

This section shows you how to set up the Platform API Server and Platform Agent services to use certificates managed by Vault.

To set up and start the services using Vault:

1. On the operator node, use the `/etc/olcne/bootstrap-olcne.sh` script to configure the Platform API Server to retrieve and use a Vault certificate. Use the `bootstrap-olcne.sh --help` command for a list of options for this script. For example:

   ```
   sudo /etc/olcne/bootstrap-olcne.sh \
   --secret-manager-type vault \
   --vault-token s.3QKNuRoTqLbjXaGBOmO6Psjh \
   --vault-address https://192.0.2.20:8200 \
   --force-download-certs \
   --olcne-component api-server
   ```

   The certificates are generated and downloaded from Vault. By default, the certificates are saved to the `/etc/olcne/certificates/` directory. You can alternatively specify a path for the certificates, for example, by including the following options in the `bootstrap-olcne.sh` command:

   ```
   --olcne-ca-path /etc/olcne/configs/certificates/production/ca.cert \
   --olcne-node-cert-path /etc/olcne/configs/certificates/production/node.cert \
   --olcne-node-key-path /etc/olcne/configs/certificates/production/node.key \
   ```

   The Platform API Server is configured to use the certificates, and started. You can confirm the service is running using:

   ```
   systemctl status olcne-api-server.service
   ```

2. On each Kubernetes node, use the `/etc/olcne/bootstrap-olcne.sh` script to configure the Platform Agent to retrieve and use a certificate. For example:

   ```
   sudo /etc/olcne/bootstrap-olcne.sh \
   --secret-manager-type vault \
   --vault-token s.3QKNuRoTqLbjXaGBOmO6Psjh \
   --vault-address https://192.0.2.20:8200 \
   --force-download-certs \
   --olcne-component agent
   ```

   The certificates are generated and downloaded from Vault.

   The Platform Agent is configured to use the certificates, and started. You can confirm the service is running using:

   ```
   systemctl status olcne-agent.service
   ```

3.6.2 Starting the Services Using Certificates

This section shows you how to set up the Platform API Server and Platform Agent services to use your own certificates, which have been copied to each node. This example assumes the certificates are available on all nodes in the `/etc/olcne/configs/certificates/production/` directory.

To set up and start the services using certificates:

1. On the operator node, use the `/etc/olcne/bootstrap-olcne.sh` script to configure the Platform API Server to use the certificates. Use the `bootstrap-olcne.sh --help` command for a list of options for this script. For example:

   ```
   sudo /etc/olcne/bootstrap-olcne.sh \
   --secret-manager-type file \
   --olcne-node-cert-path /etc/olcne/configs/certificates/production/node.cert \
   --olcne-ca-path /etc/olcne/configs/certificates/production/ca.cert \
   --olcne-node-key-path /etc/olcne/configs/certificates/production/node.key \
   --olcne-component api-server
   ```

   The Platform API Server is configured to use the certificates, and started. You can confirm the service is running using:

   ```
   systemctl status olcne-api-server.service
   ```

2. On each Kubernetes node, use the `/etc/olcne/bootstrap-olcne.sh` script to configure the Platform Agent to use the certificates. For example:

   ```
   sudo /etc/olcne/bootstrap-olcne.sh \
   --secret-manager-type file \
   --olcne-node-cert-path /etc/olcne/configs/certificates/production/node.cert \
   --olcne-ca-path /etc/olcne/configs/certificates/production/ca.cert \
   --olcne-node-key-path /etc/olcne/configs/certificates/production/node.key \
   --olcne-component agent
   ```

   The Platform Agent is configured to use the certificates, and started. You can confirm the service is running using:

   ```
   systemctl status olcne-agent.service
   ```

3.7 Creating an Environment

The first step to creating a Kubernetes cluster is to create an environment. You can create multiple environments, with each environment potentially containing multiple modules. Naming each environment and module makes it easier to manage the deployed components of Oracle Cloud Native Environment.

Note

You should not use the same node in more than one environment.

Use the `olcnectl environment create` command on the **operator** node to create an environment. For more information on the syntax for the `olcnectl environment create` command, see *Platform Command-Line Interface*.

This section shows you how to create an environment using Vault, and using your own certificates copied to the file system on each node. For information on setting up X.509 certificates, see Section 3.4, “Setting up X.509 Certificates for Kubernetes Nodes”.

3.7.1 Creating an Environment using Certificates Managed by Vault

This section shows you how to create an environment using Vault to provide and manage the certificates.

On the **operator** node, use the `olcnectl environment create` command to create an environment. For example, to create an environment named `myenvironment` using certificates generated from a Vault instance located at `https://192.0.2.20:8200`

```
olcnectl --api-server 127.0.0.1:8091 environment create \
--environment-name myenvironment \
--update-config \
--secret-manager-type vault \
--vault-token s.3QKNuRoTqLbjXaGBOmO6Psjh \
--vault-address https://192.0.2.20:8200
```

The `--update-config` option saves the certificate generated by Vault on the local host. When you use this option, you do not need to enter the certificate information again when managing the environment. This option also saves the connection information for the Platform API Server. You do not need to provide this information again when connecting to the environment. That is, the next time you connect to the environment you do not need to provide the `--api-server` option. For more information on setting the Platform API Server see *Platform Command-Line Interface*.

The `--secret-manager-type` option sets the certificate manager to Vault. Replace `--vault-token` with the token to access Vault. Replace `--vault-address` with the location of your Vault instance.

By default, the certificate is saved to `$HOME/.olcne/certificates/`*`environment_name/`*. If you want to specify a different location to save the certificate, use the `--olcne-node-cert-path`, `--olcne-ca-path`, and `--olcne-node-key-path` options. For example, add the following options to the `olcnectl environment create` command:

```
--olcne-node-cert-path /etc/olcne/configs/certificates/production/node.cert \
--olcne-ca-path /etc/olcne/configs/certificates/production/ca.cert \
--olcne-node-key-path /etc/olcne/configs/certificates/production/node.key
```

3.7.2 Creating an Environment using Certificates

This section shows you how to create an environment using your own certificates, copied to each node. This example assumes the certificates are available on all nodes in the `/etc/olcne/configs/certificates/production/` directory.

On the **operator** node, create the environment using the `olcnectl environment create` command. For example:

```
olcnectl --api-server 127.0.0.1:8091 environment create \
--environment-name myenvironment \
--update-config \
--secret-manager-type file \
--olcne-node-cert-path /etc/olcne/configs/certificates/production/node.cert \
--olcne-ca-path /etc/olcne/configs/certificates/production/ca.cert \
--olcne-node-key-path /etc/olcne/configs/certificates/production/node.key
```

The `--update-config` option saves the connection information for the Platform API Server. You do not need to provide this information again when connecting to the environment. That is, the next time you connect to the environment you do not need to provide the `--api-server` option. For more information on setting the Platform API Server see *Platform Command-Line Interface*.

The `--secret-manager-type` option sets the certificate manager to use file-based certificates.

You can optionally set the location for the certificate files using environment variables; `olcnectl` uses these if they are set.

The environment variables map to the `olcnectl environment create` command options:

- `$OLCNE_SM_CERT_PATH` sets the value used with the `--olcne-node-cert-path` option.
- `$OLCNE_SM_CA_PATH` sets the value used with the `--olcne-ca-path` option.

- `$OLCNE_SM_KEY_PATH` sets the value used with the `--olcne-node-key-path` option.

For example:

```
export OLCNE_SM_CA_PATH=/etc/olcne/configs/certificates/production/ca.cert
export OLCNE_SM_CERT_PATH=/etc/olcne/configs/certificates/production/node.cert
export OLCNE_SM_KEY_PATH=/etc/olcne/configs/certificates/production/node.key
olcnectl --api-server 127.0.0.1:8091 environment create --environment-name myenvironment \
--update-config \
--secret-manager-type file
```

3.8 Next Steps

After you create an environment, you can add any modules you want to the environment. A base installation requires a Kubernetes module which is used to create a Kubernetes cluster. For information on creating and installing a Kubernetes module, see *Container Orchestration*.

When you have created and installed a Kubernetes module, you can optionally install a service mesh using the Istio module. For information on installing the Istio module to create a service mesh, see *Service Mesh*.

Chapter 4 Configuring Oracle Cloud Native Environment Services

This chapter contains information about any configuration options for Oracle Cloud Native Environment, including configuring the Platform API Server and Platform Agent services.

4.1 Configuring the Platform API Server

The Platform API Server runs as a Systemd service, named `olcne-api-server`. You can get logs for this service using:

```
sudo journalctl -u olcne-api-server
```

By default, the service runs on TCP port 8091. You can change this and other Platform API Server settings by editing the Systemd service unit file so that the binary is invoked to use additional options.

`olcne-api-server` options:

- `[-p|--port] port_number`

  Specifies the port that the Platform API Server binds to. Defaults to `8091` if unspecified.

- `[-i|--installables] installables_path`

  Specifies the path to the directory of installable modules. Defaults to `/etc/olcne/modules`.

- `[-x|--insecure]`

  Allows the gRPC server to accept clients that do not securely establish their identity.

To reconfigure the Platform API Server to use any of these options, you can edit the Systemd unit file at `/usr/lib/systemd/system/olcne-api-server.service` and append the option to the `ExecStart` line. For example:

```
[Unit]
Description=Platform API Server for Oracle Cloud Native Environments
Wants=network.target
After=network.target

[Service]
ExecStart=/usr/libexec/olcne-api-server -i /etc/olcne/modules --port 9083
WorkingDirectory=/var/olcne
User=olcne
Group=olcne
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

If you edit the Systemd unit file, you must run the following commands for the changes to take effect:

```
sudo systemctl daemon-reload
sudo systemctl restart olcne-api-server.service
```

Note

If you change the port value for this service, you should take this into account for all other instructions provided in the documentation.

4.2 Configuring the Platform Agent

The Platform Agent runs as a Systemd service, named `olcne-agent`. You can get logs for this service using:

```
sudo journalctl -u olcne-agent
```

By default, the service runs on TCP port 8090. You can change this and other Platform Agent settings by editing the Systemd service unit file so that the binary is invoked to use additional options.

Additional options available to `olcne-agent` include:

`olcne-agent` options:

- `[-p|--port]` *`port-number`*

  Specifies the port that the Platform Agent service binds to. Defaults to `8090` if unspecified.

- `[-x|--insecure]`

  Allows the gRPC server to accept clients that do not securely establish their identity.

To reconfigure the Platform Agent to use any of these options, you can edit the Systemd unit file at `/usr/lib/systemd/system/olcne-agent.service` and append the option to the `ExecStart` line.

If you edit the Systemd unit file, you must run the following commands for the changes to take effect:

```
sudo systemctl daemon-reload
sudo systemctl restart olcne-agent.service
```

Note

If you change the port value for this service, you should take this into account for all other instructions provided in the documentation.