Oracle Cloud Native Environment

Quick Installation for Release 1.5

F59611-04
July 2023

Oracle Cloud Native Environment Quick Installation for Release 1.5,

F59611-04

Contents

Preface

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This document contains information about performing a quick installation of Oracle Cloud Native Environment and a Kubernetes cluster. The information in this document provides a fast installation method using default settings for the Oracle Cloud Native Environment platform and Kubernetes cluster. If you are deploying Oracle Cloud Native Environment for a production environment, you may want to use the installation method described in Getting Started.

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 https://www.oracle.com/corporate/accessibility/.

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.

1 Introduction

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This document contains information about performing a quick installation of Oracle Cloud Native Environment and a Kubernetes cluster. The information in this document provides a fast installation method using default settings for the Oracle Cloud Native Environment platform and Kubernetes cluster.

If you are not familiar with Oracle Cloud Native Environment, you should first read about the architecture, components, and terminology definitions in Concepts.

The installation examples in this document use the Oracle Cloud Native Environment Platform CLI (`olcnectl`) to set up the hosts and their operating systems, and the Oracle Cloud Native Environment platform and Kubernetes cluster. The installation options in this document offer a quick installation option, with commands that set up the hosts and perform the installation automatically, instead of using the longer procedure detailed in Getting Started.

The installation methods in this document require a minimum of options to be provided about your hosts and uses that information to set up each host. During the host set up, the appropriate network ports are opened, the software packages are installed, CA Certificates are created, the operating system services are configured, and finally, Kubernetes is deployed. This streamlines an installation of Oracle Cloud Native Environment to get you up and running quickly.

You can also perform more complex deployments, setting your own security options and installing optional modules using an Oracle Cloud Native Environment configuration file. Examples of using a configuration file are also included in this document.

Hosts

To start the installation, you need to set up a host with the Platform CLI (`olcnectl`). In this document, this installation host is referred to as the *operator* node.

You also need to set up the hosts you want to include in the Kubernetes cluster (the Kubernetes control plane and worker nodes).

SSH key-based authentication should be set up between the installation host (operator node) and the hosts to be included in the Kubernetes cluster.

You may also need to set up a load balancer if you want to use an external load balancer for High Availability of the Kubernetes control plane.

Platform CLI

The Platform CLI command, `olcnectl provision`, is the command to use to perform a quick installation, and is run from the operator node. This command sets up the Kubernetes nodes, configures the node operating systems, creates CA Certificates for the Kubernetes nodes, installs the software packages, and creates a Kubernetes cluster.

The installation examples in this document show you how to use the `olcnectl provision` command to perform an installation. The steps performed by this command are:

- Generate CA Certificates.
- Copy the CA Certificates to each node.
- Set up the operating system on each node, including opening network ports.
- Install the Oracle Cloud Native Environment software packages on each node.
- Start the Oracle Cloud Native Environment platform services (Platform API Server and Platform Agent).
- Create an Oracle Cloud Native Environment environment.
- Create, validate and install a Kubernetes module, which creates the Kubernetes cluster.
- Set up the Platform CLI certificates to `~/.olcne` on the operator node to access the environment using the `olcnectl` command.
- Install optional modules if using a configuration file.

If you use all the defaults when using the `olcnectl provision` command, you install the most straight-forward installation option, using private CA Certificates. It is recommended for a production environment that you use your own CA Certificates.

For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

Complex Deployments

You can perform more complex installations by writing an Oracle Cloud Native Environment configuration file and passing the file to the `olcnectl provision` command using the `--config-file` option. A configuration file allows you to specify security options such as the SELinux setting on the nodes, the location of pre-generated CA Certificates and whether to deploy the Kubernetes `externalIPs` service.

If you use a configuration file, you can also use the `olcnectl provision` command to install optional Oracle Cloud Native Environment modules, for example, the Istio module, or the Oracle Cloud Infrastructure Cloud Controller Manager module.

If you want an installation that uses Vault to generate and authenticate the certificates for the Kubernetes nodes, you should use the full installation steps outlined in Getting Started.

For information on writing a configuration file, see Platform Command-Line Interface.

Install Examples

A number of different installation types are provided to help you get up and running quickly. Choose the option that best matches your deployment type and use the installation example as a guide to get you started with your own installation.

Table 1-1 Installation Examples

Install Type	Bare Metal or Virtual Machine	Oracle Cloud Infrastructure
Platform and Kubernetes	Quick Install	Quick Install
Platform and Kubernetes HA	Quick HA Install with Internal Load Balancer Quick HA Install with External Load Balancer	Quick HA Install on Oracle Cloud Infrastructure
Platform and Kubernetes with Optional Modules	Quick Install using Configuration File	Quick HA Install using Configuration File on Oracle Cloud Infrastructure

2
Hosts

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This chapter provides the requirements for the hosts you can include in the deployment. Each host is assigned a role to when included as a node in the deployment.

The following table lists the hosts you need to set up and their role(s). The number of nodes listed in the table are required when you use the quick installation option using the `olcnectl provision` command. For example, if you do a non-HA installation, you can only provide a single Kubernetes control plane node. If you provide more than one control plane node, you need to provide a load balancer for the control plane nodes. If you follow the installation process in Getting Started, these restrictions do not apply.

You can install the Platform API Server on the operator node, and this is the recommended method. If you prefer, you can set up separate hosts for the Platform CLI and Platform API Server.

Table 2-1 Host Roles

	Operator Node	Kubernetes Control Plane Nodes	Kubernetes Worker Nodes
Number of nodes	1 only	1 only	1 or more
Number of nodes for High Availability	1 only	3 See Kubernetes High Availability Requirements.	2 See Kubernetes High Availability Requirements.
Role	Install and manage the environment and Kubernetes cluster.	Manage the lifecycle of containerized applications.	Hosts the containerized applications.
Components	Platform CLI (`olcnectl`). Platform API Server.	Platform Agent. Kubernetes API Server. Kubernetes CLI (`kubectl`).	Platform Agent. Container Runtime engine.

Hardware Requirements

Oracle Cloud Native Environment is a clustered environment that requires more than one node to form a cluster. 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.

Kubernetes Control Plane Node Hardware

The Kubernetes control plane is the container orchestration layer that exposes the Kubernetes API and interfaces to create and manage the lifecycle of containers. The nodes that form the Kubernetes control plane are referred to as *control plane* nodes. A control plane node is a host that runs the daemons and services needed to manage the cluster and orchestrate containers, such as the Oracle Cloud Native Environment Platform Agent, etcd, the Kubernetes API Server, Scheduler, Controller Manager, and Cloud Controller Manager.

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

Kubernetes Worker Node Hardware

A Kubernetes worker node is a host that runs the daemons and services needed to run pods, such as the Platform Agent, kubelet, kube-proxy, CRI-O, RunC and Kata Runtime.

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
- XFS mount-point `/var/lib/containers` with dedicated space based on the number of container images going to be saved and leveraged.

Operator Node Hardware

An operator node is a host that contains the Oracle Cloud Native Environment Platform Command-Line Interface. This node may also likely include the Oracle Cloud Native Environment Platform API Server.

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

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.

Operating System Requirements

Oracle Cloud Native Environment is available for the following x86_64 operating systems.

Table 2-2 Operating Systems (x86_64)

Operating System	Release Number	Update Number	Kernel
Oracle Linux	8	Latest and latest-1	Unbreakable Enterprise Kernel Release 7 (UEK R7)
Oracle Linux	8	Latest and latest-1	Unbreakable Enterprise Kernel Release 6 (UEK R6)
Oracle Linux	8	Latest and latest-1	Red Hat Compatible Kernel (RHCK)
Red Hat Enterprise Linux	8	Latest and latest-1	Red Hat Kernel
Oracle Linux	7	9 or later	UEK R6

3
Prerequisites

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

Before you use the `olcnectl provision` command to perform a deployment of Oracle Cloud Native Environment, you should set up the hosts you want to include in the environment and synchronize the time between them. You should also include a host to use for the installation (the *operator* node). The operator node must have the Oracle Cloud Native Environment Platform CLI (`olcnectl`) software package installed. SSH key-based authentication should also be set up between the operator node, and all other nodes to use in the deployment, including the Platform API Server node (which is likely to be the operator node).

To prepare for an installation:

1. **Set up the nodes**. Set up the nodes you want to use for the installation. See Hosts for information on the number and configuration of hosts you can use for an installation.
2. **Set up SSH key-based authentication**. Set up SSH key-based authentication for the user that is to run the Platform CLI (`olcnectl`) installation commands to enable login from the operator node to other nodes in the system. In a typical setup, this user needs a private key on the operator node and the corresponding public key on each Kubernetes node as well as on the Platform API Server node. For information on setting up SSH key based authentication, see Set up SSH Key-based Authentication and Oracle Linux: Connecting to Remote Systems With OpenSSH.
3. **Set up the operator node**. On the operator node, enable access to the Oracle Cloud Native Environment software package repository and install the Platform CLI (`olcnectl`) software package.
4. **Synchronize time**. On Oracle Linux 7 hosts, synchronize the time between all nodes using a Network Time Service. This is not required for Oracle Linux 8 hosts.

Related Topics

- Hosts
- Set up the Operator Node on Oracle Linux 8
- Set up the Operator Node on Oracle Linux 7
- Set up a Network Time Service on Oracle Linux 7

Set up SSH Key-based Authentication

Set up and verify SSH key-based authentication from the operator node to the Kubernetes nodes. Do not set a passphrase when creating the key pair as this will prevent automatic processes from running seamlessly. Take appropriate steps to ensure the private key remains secure.

Set up SSH key-based authentication for the user that is to run the Platform CLI (`olcnectl`) installation commands to enable login from the operator node to the following nodes:

- Each Kubernetes node.
- The Platform API Server node.

The following steps demonstrate one way of setting up the SSH key-based authentication:

1. **Generate the private and public key pair:**

   On the operator node, run `ssh-keygen` as the user that you use to run `olcnectl` commands. Do not create a passphrase for the key (press `<Enter>` when prompted for a passphrase). For example:

   ```
   ssh-keygen
   ```

   Output similar to the following is displayed:

   ```
   Generating public/private rsa key pair.
   Enter file in which to save the key (/home/oracle/.ssh/id_rsa):<Enter>
   Enter passphrase (empty for no passphrase): <Enter>
   Enter same passphrase again: <Enter>
   Your identification has been saved in /home/oracle/.ssh/id_rsa.
   Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
   ...
   ```

   In this example and all subsequent steps, the `oracle` user is used. Your username may differ.

2. **Verify the location of the private and public key pair:**

   Verify the private and public key pair have been created at the location reported in the `ssh-keygen` command output:

   ```
   ls -l /home/oracle/.ssh/
   ```

   ```
   ...
   -rw-------. 1 oracle oracle 2643 Jan 10 14:55 id_rsa
   -rw-r--r--. 1 oracle oracle  600 Jan 10 14:55 id_rsa.pub
   ...
   ```

   The public key is indicated by the file with the "`.pub`" extension.

3. **Set up the public key on the target nodes:**

   Add the contents of the public key to the `$HOME/.ssh/authorized_keys` file on each target node for the user for which the key-based SSH is being set up. The following are possible ways of doing this:

   - **Run the `ssh-copy-id` Command:**

For systems with password authentication enabled, you have the option of running the `ssh-copy-id` command on the operator node. The syntax is as follows:

```
ssh-copy-id user@host
```

When prompted you enter the user's password for the host. Once the command successfully completes, the public key's contents will have been added to the copy of the user's `$HOME/.ssh/authorized_keys` file on the remote host.

The following example shows how command `ssh-copy-id` can be used to add the public key to the `authorized_keys` file for user `oracle` on host `192.0.2.255` :

```
ssh-copy-id oracle@192.0.2.255

/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
oracle/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys

oracle@192.0.2.255's password:

Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'oracle@192.0.2.255'"and check
to make sure that only the key(s) you wanted were added.
```

- **Manually setting the `authorized_keys` file**

  If you do not have access to the `ssh-copy-id` command or are unable to access the system remotely with a password, you must populate the `$HOME/.ssh/authorized_keys` file on the target nodes manually. The following steps show how this can be done:

  a. On the operator node, open the public key you have created in a file. Continuing with our example we would do this as follows:

     ```
     cat /home/oracle/.ssh/id_rsa.pub

     ssh-rsa AQRayc2EAAAABIwAAAQEA6OabJhWABsZ4F3mcjEPT3sxnXx1OoUcvuC...
     ...OEKX9Kp9QWH+IfASI8q09xQ= oracle@svr-operator-node
     ```

  b. Copy the public key you have created.

  c. Log in to one of the target servers.

  d. Confirm the location of the `$HOME/.ssh/authorized_keys` file belonging to the user whose key-based access you are setting up. For example:

     ```
     ls /home/oracle/.ssh/authorized_keys

     /home/oracle/.ssh/authorized_keys
     ```

  e. Append the public key to the `authorized_keys` file.

  f. Ensure that the permissions of the user's `$HOME/.ssh` and `$HOME/.ssh/authorized_keys` file are set correctly:

     – `$HOME/.ssh`: The recommended permissions are read/write/execute for the user, and not accessible by others, as shown in the following example output:

       ```
       drwx------. 2 oracle oracle 66 Jan 11 17:33 /home/oracle/.ssh
       ```

- `$HOME/.ssh/authorized_keys`: The recommended permissions are read/write for the user, and not accessible by others, as shown in the following example output:

  ```
  -rw-------. 1 oracle oracle 1004 Jan 11 17:36 /home/oracle/.ssh/
  authorized_keys
  ```

g. Repeat the steps for the remaining target nodes.

4. **Verify your user has SSH key-based access from the operator node:**

   On the operator node, use `ssh` to connect to each of the other nodes and confirm login succeeds without being prompted for a password.

   For example, if the user in our case is `oracle`, and one of the target nodes is host `192.0.2.255`, we would confirm key-based SSH access by running the `ssh` command on the operator node as follows:

   ```
   ssh oracle@192.0.2.255
   ```

For more information on setting up SSH key-based authentication, see Oracle Linux: Connecting to Remote Systems With OpenSSH

Set up the Operator Node on Oracle Linux 8

Set up the operator node on an Oracle Linux 8 host. The operator node is used to perform an installation of Oracle Cloud Native Environment and a Kubernetes cluster.

To set up the operator node on Oracle Linux 8:

1. Install the `oracle-olcne-release-el8` release package:

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

2. Enable the yum repositories.

   For hosts running UEK R7, use:

   ```
   sudo dnf config-manager --enable ol8_olcne15 ol8_addons ol8_baseos_latest
   ol8_appstream ol8_kvm_appstream ol8_UEKR7
   ```

   For hosts running UEK R6, use:

   ```
   sudo dnf config-manager --enable ol8_olcne15 ol8_addons ol8_baseos_latest
   ol8_appstream ol8_kvm_appstream ol8_UEKR6
   ```

   For hosts running RHCK, use:

   ```
   sudo dnf config-manager --enable ol8_olcne15 ol8_addons ol8_baseos_latest
   ol8_appstream ol8_kvm_appstream
   ```

3. Disable the following repositories:

   ```
   sudo dnf config-manager --disable ol8_olcne14 ol8_olcne13 ol8_olcne12
   ol8_developer
   ```

4. Install the `olcnectl` software package:

   ```
   sudo dnf install olcnectl
   ```

Set up the Operator Node on Oracle Linux 7

Set up the operator node on an Oracle Linux 7 host. The operator node is used to perform an installation of Oracle Cloud Native Environment and a Kubernetes cluster.

To set up the operator node on Oracle Linux 7:

1. Install the `oracle-olcne-release-el7` release package:

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

2. Enable the yum repositories:

   ```
   sudo yum-config-manager --enable ol7_olcne15 ol7_kvm_utils ol7_addons ol7_latest
   ol7_UEKR6
   ```

3. Disable the following repositories:

   ```
   sudo yum-config-manager --disable ol7_olcne14 ol7_olcne13 ol7_olcne12 ol7_olcne11
   ol7_olcne ol7_developer
   ```

4. Install the `olcnectl` software package:

   ```
   sudo yum install olcnectl
   ```

Set up a Network Time Service on Oracle Linux 7

This sets up a Network Time Service on Oracle Linux 7 hosts using `chronyd` to synchronize time on the hosts to use in the deployment.

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
   ```

4
Quick Install

Install Oracle Cloud Native Environment on bare metal hosts or virtual machines, including a Kubernetes cluster.

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This is the fastest method to set up a basic deployment of Oracle Cloud Native Environment on bare metal hosts or virtual machines. This method sets up the nodes, installs the Oracle Cloud Native Environment platform and installs a Kubernetes cluster.

Security Considerations: You should consider the following security settings when you use this installation example:

- Private CA Certificates are used to secure network communication between the Kubernetes nodes.
- SELinux is set to `permissive` mode on the host operating system on each node.
- The Kubernetes `externalIPs` service is not deployed.

If you want to perform a more complex deployment and change these security settings, use a configuration file as shown in Quick Install using Configuration File.

Nodes Required: At least three nodes. They are:

- **Operator Node**: One node to use as the operator node, which is used to perform the installation using the Platform CLI (`olcnectl`), and to host the Platform API Server.
- **Kubernetes control plane**: One node to use as a Kubernetes control plane node. You cannot use more than one control plane node without a load balancer.
- **Kubernetes worker**: At least one node to use as a Kubernetes worker node.

Before you begin: Complete the prerequisite set up. See Prerequisites.

To do a quick install with no load balancer:

1. On the operator node, use the `olcnectl provision` command to start the installation. The mandatory syntax is:

```
olcnectl provision
--api-server host
--master-nodes host
--worker-nodes hosts
```

```
--environment-name name
--name name
```

Use the `--api-server` option to set the FQDN of the node on which the Platform API Server should be set up.

Use the `--master-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes control plane nodes. This is a comma separated list.

Use the `--worker-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes worker nodes. This is a comma separated list.

Use the `--environment-name` option to set the name to identify the environment.

Use the `--name` option to set the name to identify the Kubernetes module.

There are a number of other command options that you may need, such as the SSH log in credentials, proxy server information, and the option to automatically accept any prompts using the `--yes` option. For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

For example:

```
olcnectl provision \
--api-server operator.example.com \
--master-nodes control1.example.com \
--worker-nodes worker1.example.com,worker2.example.com \
--environment-name myenvironment \
--name mycluster
```

2. A list of the steps to be performed on each node is displayed and a prompt is displayed to proceed. For example, on a control plane node, the changes may look similar to:

```
? Apply control-plane configuration on control1.example.com:
* Install oracle-olcne-release
...
* Install and enable olcne-agent

Proceed? yes/no(default) yes
```

Enter `yes` to continue. The node is set up.

Information about the changes on each node is displayed. You need to confirm the set up steps for each node.

> **Tip:**
>
> If you want to avoid accepting the changes on each node, use the `--yes` command option with the `olcnectl provision` command.

3. The nodes are set up with the Oracle Cloud Native Environment platform and a Kubernetes module is installed to set up a Kubernetes cluster. You can show information about the environment using the syntax:

```
olcnectl module instances
--api-server host_name:8091
--environment-name name
```

> **Tip:**
>
> To avoid having to enter the `--api-server` option in future `olcnectl` commands, add the `--update-config` option.

For example:

```
olcnectl module instances \
--api-server operator.example.com:8091 \
--environment-name myenvironment \
--update-config
```

The output looks similar to:

```
INFO[...] Global flag configuration for myenvironment has been written to
the
local Platform config and you don't need to specify them for any future
calls
INSTANCE                    MODULE      STATE
control1.example.com:8090   node        installed
...
mycluster                   kubernetes  installed
```

If you want to see more information about the deployment, use the `olcnectl module report` command. For example:

```
olcnectl module report \
--environment-name myenvironment \
--name mycluster \
--children
```

4. Set up the Kubernetes CLI (`kubectl`) on a control plane node. The `kubectl` command is installed on each control plane node in the cluster. To use it to access the cluster, you need to configure it using the Kubernetes configuration file.

   Log into a control plane node and copy and paste these commands to a terminal in your home directory:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
export KUBECONFIG=$HOME/.kube/config
echo 'export KUBECONFIG=$HOME/.kube/config' >> $HOME/.bashrc
```

Verify that you can use the `kubectl` command using any `kubectl` command such as:

```
kubectl get pods --all-namespaces
```

The output looks similar to:

```
NAMESPACE              NAME                              READY
STATUS    RESTARTS   AGE
externalip-validat...  externalip-validation-...         1/1
Running   0          1h
kube-system            coredns-...                       1/1
Running   0          1h
kube-system            coredns-...                       1/1
Running   0          1h
kube-system            etcd-...                          1/1
Running   2          1h
kube-system            etcd-...                          1/1
Running   2          1h
kube-system            kube-apiserver-...                1/1
Running   2          1h
kube-system            kube-apiserver-...                1/1
Running   2          1h
kube-system            kube-controller-manager-...       1/1
Running   5 (1h ago) 1h
kube-system            kube-controller-manager-...       1/1
Running   2          1h
kube-system            kube-flannel-...                  1/1
Running   0          1h
kube-system            kube-flannel-...                  1/1
Running   0          1h
kube-system            kube-flannel-...                  1/1
Running   0          1h
kube-system            kube-proxy-...                    1/1
Running   0          1h
kube-system            kube-proxy-...                    1/1
Running   0          1h
kube-system            kube-proxy-...                    1/1
Running   0          1h
kube-system            kube-scheduler-...                1/1
Running   5 (1h ago) 1h
kube-system            kube-scheduler-...                1/1
Running   2          1h
kubernetes-dashboard   kubernetes-dashboard-...          1/1
Running   0          1h
```

> **Note:**
>
> After the deployment, a Kubernetes configuration file is created in the local directory of the operator node. The file is named `kubeconfig.environment_name.cluster_name` and contains information about the Kubernetes cluster. This file is created for your convenience and is not required to set up `kubectl` on the control plane nodes.
>
> You may want to use this file to add to a larger Kubernetes configuration file if you have multiple clusters. See the upstream Kubernetes documentation for more information on configuring access to multiple clusters.

Related Topics

- Creating a MetalLB Module
- Creating a Gluster Container Storage Interface Module
- Creating an Operator Lifecycle Manager Module
- Creating an Istio Module

5

Quick HA Install with Internal Load Balancer

Install a Highly Available Oracle Cloud Native Environment on bare metal hosts or virtual machines, including a Kubernetes cluster. This example uses the internal containerized NGINX and Keepalived load balancer deployed by the Platform CLI.

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This is the fastest method to set up a basic Highly Available deployment of Oracle Cloud Native Environment on bare metal hosts or virtual machines. This method sets up the nodes, installs the Oracle Cloud Native Environment platform and installs a Kubernetes cluster. A load balancer is deployed by the Platform CLI to the control plane nodes and configured with the Kubernetes cluster. The load balancer is a container-based deployment of NGINX and Keepalived.

Security Considerations: You should consider the following security settings when you use this installation example:

- Private CA Certificates are used to secure network communication between the Kubernetes nodes.
- SELinux is set to `permissive` mode on the host operating system on each node.
- The Kubernetes `externalIPs` service is not deployed.

If you want to perform a more complex deployment and change these security settings, use a configuration file as shown in Quick Install using Configuration File.

Nodes Required: As many nodes as required for High Availability. (See Kubernetes High Availability Requirements). They are:

- **Operator Node**: One node to use as the operator node, which is used to perform the installation using the Platform CLI (`olcnectl`), and to host the Platform API Server.
- **Kubernetes control plane**: At least three nodes to use as Kubernetes control plane nodes.
- **Kubernetes worker**: At least two nodes to use as Kubernetes worker nodes.

Before you begin: Complete the prerequisite set up. See Prerequisites.

To do a quick install with an internal load balancer:

1. Use the `--virtual-ip` option when creating the Kubernetes module to 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.

2. On the operator node, use the `olcnectl provision` command to start the installation. The mandatory syntax is:

```
olcnectl provision
--api-server host
--master-nodes hosts
--worker-nodes hosts
--environment-name name
--name name
--virtual-ip IP_address
```

Use the `--api-server` option to set the FQDN of the node on which the Platform API Server should be set up.

Use the `--master-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes control plane nodes. This is a comma separated list.

Use the `--worker-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes worker nodes. This is a comma separated list.

Use the `--environment-name` option to set the name to identify the environment.

Use the `--name` option to set the name to identify the Kubernetes module.

Use the `--virtual-ip` option to set the virtual IP address.

There are a number of other command options that you may need, such as the SSH log in credentials, proxy server information, and the option to automatically accept any prompts using the `--yes` option. For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

For example:

```
olcnectl provision \
--api-server operator.example.com \
--master-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--virtual-ip 192.0.2.100
```

3. A list of the steps to be performed on each node is displayed and a prompt is displayed to proceed. For example, on a control plane node, the changes may look similar to:

```
? Apply control-plane configuration on control1.example.com:
* Install oracle-olcne-release
...
```

```
* Install and enable olcne-agent

Proceed? yes/no(default) yes
```

Enter `yes` to continue. The node is set up.

Information about the changes on each node is displayed. You need to confirm the set up steps for each node.

> **Tip:**
>
> If you want to avoid accepting the changes on each node, use the `--yes` command option with the `olcnectl provision` command.

4. The nodes are set up with the Oracle Cloud Native Environment platform and a Kubernetes module is installed to set up a Kubernetes cluster. You can show information about the environment using the syntax:

```
olcnectl module instances
--api-server host_name:8091
--environment-name name
```

> **Tip:**
>
> To avoid having to enter the `--api-server` option in future `olcnectl` commands, add the `--update-config` option.

For example:

```
olcnectl module instances \
--api-server operator.example.com:8091 \
--environment-name myenvironment \
--update-config
```

The output looks similar to:

```
INFO[...] Global flag configuration for myenvironment has been written to
the
local Platform config and you don't need to specify them for any future
calls
INSTANCE                    MODULE      STATE
control1.example.com:8090   node        installed
...
mycluster                   kubernetes  installed
```

If you want to see more information about the deployment, use the `olcnectl module report` command. For example:

```
olcnectl module report \
--environment-name myenvironment \
--name mycluster \
--children
```

5. Set up the Kubernetes CLI (`kubectl`) on a control plane node. The `kubectl` command is installed on each control plane node in the cluster. To use it to access the cluster, you need to configure it using the Kubernetes configuration file.

   Log into a control plane node and copy and paste these commands to a terminal in your home directory:

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   export KUBECONFIG=$HOME/.kube/config
   echo 'export KUBECONFIG=$HOME/.kube/config' >> $HOME/.bashrc
   ```

   Verify that you can use the `kubectl` command using any `kubectl` command such as:

   ```
   kubectl get pods --all-namespaces
   ```

   The output looks similar to:

   ```
   NAMESPACE              NAME                           READY
   STATUS    RESTARTS   AGE
   externalip-validat...  externalip-validation-...      1/1
   Running   0          1h
   kube-system            coredns-...                    1/1
   Running   0          1h
   kube-system            coredns-...                    1/1
   Running   0          1h
   kube-system            etcd-...                       1/1
   Running   2          1h
   kube-system            etcd-...                       1/1
   Running   2          1h
   kube-system            kube-apiserver-...             1/1
   Running   2          1h
   kube-system            kube-apiserver-...             1/1
   Running   2          1h
   kube-system            kube-controller-manager-...    1/1
   Running   5 (1h ago) 1h
   kube-system            kube-controller-manager-...    1/1
   Running   2          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   ```

```
kube-system            kube-proxy-...                1/1     Running
0          1h
kube-system            kube-proxy-...                1/1     Running
0          1h
kube-system            kube-proxy-...                1/1     Running
0          1h
kube-system            kube-scheduler-...            1/1     Running   5
(1h ago) 1h
kube-system            kube-scheduler-...            1/1     Running
2          1h
kubernetes-dashboard   kubernetes-dashboard-...      1/1     Running
0          1h
```

> **Note:**
>
> After the deployment, a Kubernetes configuration file is created in the local directory of the operator node. The file is named `kubeconfig.`*`environment_name`*`.`*`cluster_name`* and contains information about the Kubernetes cluster. This file is created for your convenience and is not required to set up `kubectl` on the control plane nodes.
>
> You may want to use this file to add to a larger Kubernetes configuration file if you have multiple clusters. See the upstream Kubernetes documentation for more information on configuring access to multiple clusters.

Related Topics

- Creating a MetalLB Module
- Creating a Gluster Container Storage Interface Module
- Creating an Operator Lifecycle Manager Module
- Creating an Istio Module

6
Quick HA Install with External Load Balancer

Install a Highly Available Oracle Cloud Native Environment on bare metal hosts or virtual machines, including a Kubernetes cluster. This example uses an external load balancer.

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This is the fastest method to set up a basic Highly Available deployment of Oracle Cloud Native Environment on bare metal hosts or virtual machines. This method sets up the nodes, installs the Oracle Cloud Native Environment platform and installs a Kubernetes cluster. An external load balancer is used for the Kubernetes cluster.

Security Considerations: You should consider the following security settings when you use this installation example:

- Private CA Certificates are used to secure network communication between the Kubernetes nodes.
- SELinux is set to `permissive` mode on the host operating system on each node.
- The Kubernetes `externalIPs` service is not deployed.

If you want to perform a more complex deployment and change these security settings, use a configuration file as shown in Quick Install using Configuration File.

Nodes Required: As many nodes as required for High Availability. (See Kubernetes High Availability Requirements). They are:

- **Operator Node**: One node to use as the operator node, which is used to perform the installation using the Platform CLI (`olcnectl`), and to host the Platform API Server.
- **Kubernetes control plane**: At least three nodes to use as Kubernetes control plane nodes.
- **Kubernetes worker**: At least two nodes to use as Kubernetes worker nodes.

Before you begin: Complete the prerequisite set up. See Prerequisites.

1. Set up an external load balancer. 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.

2. On the operator node, use the `olcnectl provision` command to start the installation. The mandatory syntax is:

```
olcnectl provision
--api-server host
--master-nodes hosts
--worker-nodes hosts
--environment-name name
--name name
--load-balancer location
```

Use the `--api-server` option to set the FQDN of the node on which the Platform API Server should be set up.

Use the `--master-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes control plane nodes. This is a comma separated list.

Use the `--worker-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes worker nodes. This is a comma separated list.

Use the `--environment-name` option to set the name to identify the environment.

Use the `--name` option to set the name to identify the Kubernetes module.

Use the `--load-balancer` option to set the URL to the external load balancer.

There are a number of other command options that you may need, such as the SSH log in credentials, proxy server information, and the option to automatically accept any prompts using the `--yes` option. For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

For example:

```
olcnectl provision \
--api-server operator.example.com \
--master-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--load-balancer lb.example.com:6443
```

3. A list of the steps to be performed on each node is displayed and a prompt is displayed to proceed. For example, on a control plane node, the changes may look similar to:

```
? Apply control-plane configuration on control1.example.com:
* Install oracle-olcne-release
...
* Install and enable olcne-agent

Proceed? yes/no(default) yes
```

Enter `yes` to continue. The node is set up.

Information about the changes on each node is displayed. You need to confirm the set up steps for each node.

> **Tip:**
>
> If you want to avoid accepting the changes on each node, use the `--yes` command option with the `olcnectl provision` command.

4. The nodes are set up with the Oracle Cloud Native Environment platform and a Kubernetes module is installed to set up a Kubernetes cluster. You can show information about the environment using the syntax:

```
olcnectl module instances
--api-server host_name:8091
--environment-name name
```

> **Tip:**
>
> To avoid having to enter the `--api-server` option in future `olcnectl` commands, add the `--update-config` option.

For example:

```
olcnectl module instances \
--api-server operator.example.com:8091 \
--environment-name myenvironment \
--update-config
```

The output looks similar to:

```
INFO[...] Global flag configuration for myenvironment has been written to
the
local Platform config and you don't need to specify them for any future
calls
INSTANCE                    MODULE      STATE
control1.example.com:8090   node        installed
...
mycluster                   kubernetes  installed
```

If you want to see more information about the deployment, use the `olcnectl module report` command. For example:

```
olcnectl module report \
--environment-name myenvironment \
--name mycluster \
--children
```

5. Set up the Kubernetes CLI (`kubectl`) on a control plane node. The `kubectl` command is installed on each control plane node in the cluster. To use it to access the cluster, you need to configure it using the Kubernetes configuration file.

   Log into a control plane node and copy and paste these commands to a terminal in your home directory:

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   export KUBECONFIG=$HOME/.kube/config
   echo 'export KUBECONFIG=$HOME/.kube/config' >> $HOME/.bashrc
   ```

   Verify that you can use the `kubectl` command using any `kubectl` command such as:

   ```
   kubectl get pods --all-namespaces
   ```

   The output looks similar to:

   ```
   NAMESPACE              NAME                           READY
   STATUS    RESTARTS   AGE
   externalip-validat...  externalip-validation-...      1/1
   Running   0          1h
   kube-system            coredns-...                    1/1
   Running   0          1h
   kube-system            coredns-...                    1/1
   Running   0          1h
   kube-system            etcd-...                       1/1
   Running   2          1h
   kube-system            etcd-...                       1/1
   Running   2          1h
   kube-system            kube-apiserver-...             1/1
   Running   2          1h
   kube-system            kube-apiserver-...             1/1
   Running   2          1h
   kube-system            kube-controller-manager-...    1/1
   Running   5 (1h ago) 1h
   kube-system            kube-controller-manager-...    1/1
   Running   2          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   kube-system            kube-proxy-...                 1/1
   Running   0          1h
   kube-system            kube-proxy-...                 1/1
   Running   0          1h
   kube-system            kube-proxy-...                 1/1
   Running   0          1h
   kube-system            kube-scheduler-...             1/1
   Running   5 (1h ago) 1h
   ```

```
kube-system             kube-scheduler-...            1/1     Running
2          1h
kubernetes-dashboard    kubernetes-dashboard-...      1/1     Running
0          1h
```

> **Note:**
>
> After the deployment, a Kubernetes configuration file is created in the local directory of the operator node. The file is named `kubeconfig.environment_name.cluster_name` and contains information about the Kubernetes cluster. This file is created for your convenience and is not required to set up `kubectl` on the control plane nodes.
>
> You may want to use this file to add to a larger Kubernetes configuration file if you have multiple clusters. See the upstream Kubernetes documentation for more information on configuring access to multiple clusters.

Related Topics

- Creating a MetalLB Module
- Creating a Gluster Container Storage Interface Module
- Creating an Operator Lifecycle Manager Module
- Creating an Istio Module

7

Quick HA Install on Oracle Cloud Infrastructure

Install a Highly Available Oracle Cloud Native Environment on Oracle Cloud Infrastructure, on bare metal hosts or virtual machines, including a Kubernetes cluster. This example uses the Oracle Cloud Infrastructure load balancer.

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This is the fastest method to set up a basic Highly Available deployment of Oracle Cloud Native Environment on Oracle Cloud Infrastructure, on bare metal hosts or virtual machines. This method sets up the nodes, installs the Oracle Cloud Native Environment platform and installs a Kubernetes cluster. An Oracle Cloud Infrastructure load balancer is used for the Kubernetes cluster.

Security Considerations: You should consider the following security settings when you use this installation example:

- Private CA Certificates are used to secure network communication between the Kubernetes nodes.
- SELinux is set to `permissive` mode on the host operating system on each node.
- The Kubernetes `externalIPs` service is not deployed.

If you want to perform a more complex deployment and change these security settings, use a configuration file as shown in Quick Install using Configuration File.

Nodes Required: As many nodes as required for High Availability. (See Kubernetes High Availability Requirements). They are:

- **Operator Node**: One node to use as the operator node, which is used to perform the installation using the Platform CLI (`olcnectl`), and to host the Platform API Server.
- **Kubernetes control plane**: At least three nodes to use as Kubernetes control plane nodes.
- **Kubernetes worker**: At least two nodes to use as Kubernetes worker nodes.

Before you begin: Complete the prerequisite set up. See Prerequisites.

To do a quick install with an Oracle Cloud Infrastructure load balancer:

1. Set up the Oracle Cloud Infrastructure load balancer.

a. Log into the Oracle Cloud Infrastructure User Interface.

b. Create a load balancer.

c. Add a backend set to the load balancer using weighted round robin. Set the health check to be TCP port 6443.

d. Add the control plane nodes to the backend set. Set the port for the control plane nodes to port 6443.

e. Create a listener for the backend set using TCP port 6443.

2. On the operator node, use the `olcnectl provision` command to start the installation. The mandatory syntax is:

```
olcnectl provision
--api-server host
--master-nodes hosts
--worker-nodes hosts
--environment-name name
--name name
--load-balancer location
```

Use the `--api-server` option to set the FQDN of the node on which the Platform API Server should be set up.

Use the `--master-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes control plane nodes. This is a comma separated list.

Use the `--worker-nodes` option to set the FQDN of the nodes that should be set up with the Platform Agent and assigned the role of Kubernetes worker nodes. This is a comma separated list.

Use the `--environment-name` option to set the name to identify the environment.

Use the `--name` option to set the name to identify the Kubernetes module.

Use the `--load-balancer` option to set the URL to the Oracle Cloud Infrastructure load balancer.

There are a number of other command options that you may need, such as the SSH log in credentials, proxy server information, and the option to automatically accept any prompts using the `--yes` option. For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

For example:

```
olcnectl provision \
--api-server operator.example.com \
--master-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--load-balancer lb.example.com:6443
```

3. A list of the steps to be performed on each node is displayed and a prompt is displayed to proceed. For example, on a control plane node, the changes may look similar to:

```
? Apply control-plane configuration on control1.example.com:
* Install oracle-olcne-release
...
* Install and enable olcne-agent

Proceed? yes/no(default) yes
```

Enter `yes` to continue. The node is set up.

Information about the changes on each node is displayed. You need to confirm the set up steps for each node.

> **Tip:**
>
> If you want to avoid accepting the changes on each node, use the `--yes` command option with the `olcnectl provision` command.

4. The nodes are set up with the Oracle Cloud Native Environment platform and a Kubernetes module is installed to set up a Kubernetes cluster. You can show information about the environment using the syntax:

```
olcnectl module instances
--api-server host_name:8091
--environment-name name
```

> **Tip:**
>
> To avoid having to enter the `--api-server` option in future `olcnectl` commands, add the `--update-config` option.

For example:

```
olcnectl module instances \
--api-server operator.example.com:8091 \
--environment-name myenvironment \
--update-config
```

The output looks similar to:

```
INFO[...] Global flag configuration for myenvironment has been written to
the
local Platform config and you don't need to specify them for any future
calls
INSTANCE                    MODULE      STATE
control1.example.com:8090   node        installed
```

```
...
mycluster                  kubernetes  installed
```

If you want to see more information about the deployment, use the `olcnectl module report` command. For example:

```
olcnectl module report \
--environment-name myenvironment \
--name mycluster \
--children
```

5. Set up the Kubernetes CLI (`kubectl`) on a control plane node. The `kubectl` command is installed on each control plane node in the cluster. To use it to access the cluster, you need to configure it using the Kubernetes configuration file.

   Log into a control plane node and copy and paste these commands to a terminal in your home directory:

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   export KUBECONFIG=$HOME/.kube/config
   echo 'export KUBECONFIG=$HOME/.kube/config' >> $HOME/.bashrc
   ```

   Verify that you can use the `kubectl` command using any `kubectl` command such as:

   ```
   kubectl get pods --all-namespaces
   ```

   The output looks similar to:

   ```
   NAMESPACE              NAME                           READY
   STATUS    RESTARTS   AGE
   externalip-validat...  externalip-validation-...      1/1
   Running   0          1h
   kube-system            coredns-...                    1/1
   Running   0          1h
   kube-system            coredns-...                    1/1
   Running   0          1h
   kube-system            etcd-...                       1/1
   Running   2          1h
   kube-system            etcd-...                       1/1
   Running   2          1h
   kube-system            kube-apiserver-...             1/1
   Running   2          1h
   kube-system            kube-apiserver-...             1/1
   Running   2          1h
   kube-system            kube-controller-manager-...    1/1
   Running   5 (1h ago) 1h
   kube-system            kube-controller-manager-...    1/1
   Running   2          1h
   kube-system            kube-flannel-...               1/1
   Running   0          1h
   ```

```
kube-system             kube-flannel-...              1/1     Running
0          1h
kube-system             kube-flannel-...              1/1     Running
0          1h
kube-system             kube-proxy-...                1/1     Running
0          1h
kube-system             kube-proxy-...                1/1     Running
0          1h
kube-system             kube-proxy-...                1/1     Running
0          1h
kube-system             kube-scheduler-...            1/1     Running   5
(1h ago)  1h
kube-system             kube-scheduler-...            1/1     Running
2          1h
kubernetes-dashboard    kubernetes-dashboard-...      1/1     Running
0          1h
```

> **Note:**
>
> After the deployment, a Kubernetes configuration file is created in the local directory of the operator node. The file is named `kubeconfig.`*`environment_name`*`.`*`cluster_name`* and contains information about the Kubernetes cluster. This file is created for your convenience and is not required to set up `kubectl` on the control plane nodes.
>
> You may want to use this file to add to a larger Kubernetes configuration file if you have multiple clusters. See the upstream Kubernetes documentation for more information on configuring access to multiple clusters.

Related Topics

- Creating an Oracle Cloud Infrastructure Cloud Controller Manager Module
- Creating an Operator Lifecycle Manager Module
- Creating an Istio Module

8
Quick Install using Configuration File

Install Oracle Cloud Native Environment on bare metal hosts or virtual machines, including a Kubernetes cluster, using a configuration file.

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This sets up a basic deployment of Oracle Cloud Native Environment on bare metal hosts, including a Kubernetes cluster.

Nodes Required: At least three nodes. They are:

- **Operator Node**: One node to use as the operator node, which is used to perform the installation using the Platform CLI (`olcnectl`), and to host the Platform API Server.
- **Kubernetes control plane**: At least one node to use as a Kubernetes control plane node.
- **Kubernetes worker**: At least one node to use as a Kubernetes worker node.

Before you begin: Complete the prerequisite set up. See Prerequisites.

To do a quick install using a configuration file:

1. On the operator node, create an Oracle Cloud Native Environment configuration file for your deployment. For information on creating an Oracle Cloud Native Environment configuration file, see Platform Command-Line Interface. This example uses the file name `myenvironment.yaml` for the configuration file.

   A basic example configuration file is:

   ```
   environments:
     - environment-name: myenvironment
       globals:
         api-server: operator.example.com:8091
         selinux: enforcing
       modules:
         - module: kubernetes
           name: mycluster
           args:
             container-registry: container-registry.oracle.com/olcne
             master-nodes:
               - control1.example.com:8090
   ```

```
    worker-nodes:
      - worker1.example.com:8090
      - worker2.example.com:8090
```

This example configuration file uses the default settings to create a Kubernetes cluster with a single control plane node, and two worker nodes. You should change the nodes listed to those of your own hosts.

Private CA Certificates, using default settings, are automatically created and distributed to each node to secure the communication. If you want to use your own pre-generated certificates, specify the location of the certificates using the `olcne-ca-path`, `olcne-node-cert-path` and `olcne-node-key-path` options. The certificates must be in place on the nodes before you provision them using the configuration file. For example, the `globals` section would look similar to:

```
globals:
  api-server: operator.example.com:8091
  selinux: enforcing
  olcne-ca-path: /etc/olcne/certificates/ca.cert
  olcne-node-cert-path: /etc/olcne/certificates/node.cert
  olcne-node-key-path: /etc/olcne/certificates/node.key
```

> **Tip:**
>
> You can use the `olcnectl certificates distribute` command to generate your own certificates using your own key material, and copy them to the nodes. See also the `olcnectl certificates generate` and `olcnectl certificates copy` commands.

By default, a Kubernetes service is deployed that controls access to `externalIPs` in Kubernetes services. Private CA Certificates are also automatically generated for this purpose, using default values. If you want to use your own certificates, include the location using the `restrict-service-externalip-ca-cert`, `restrict-service-externalip-tls-cert` and `restrict-service-externalip-tls-key` options in the `args` section for the Kubernetes module. You can also set the IP addresses that can be accessed by Kubernetes services using the `restrict-service-externalip-cidrs` option. For example, the `args` section would look similar to:

```
    args:
      container-registry: container-registry.oracle.com/olcne
      master-nodes:
        - control1.example.com:8090
      worker-nodes:
        - worker1.example.com:8090
        - worker2.example.com:8090
      restrict-service-externalip-ca-cert: /etc/olcne/
certificates/restrict_external_ip/ca.cert
      restrict-service-externalip-tls-cert: /etc/olcne/
certificates/restrict_external_ip/node.cert
      restrict-service-externalip-tls-key: /etc/olcne/
```

```
certificates/restrict_external_ip/node.key
          restrict-service-externalip-cidrs:
192.0.2.0/24,198.51.100.0/24
```

If you do not want to deploy this service, use the `restrict-service-externalip: false` option in the configuration file. For example, the `args` section would look similar to:

```
        args:
          container-registry: container-registry.oracle.com/olcne
          master-nodes:
            - control1.example.com:8090
          worker-nodes:
            - worker1.example.com:8090
            - worker2.example.com:8090
          restrict-service-externalip: false
```

For more information on setting access to `externalIPs` in Kubernetes services, see Container Orchestration.

If you want to include other modules to deploy with the Kubernetes module, add them to the configuration file. A more complex example of a configuration file, which includes an external load balancer, and installs other modules, is:

```
environments:
  - environment-name: myenvironment
    globals:
      api-server: operator.example.com:8091
      selinux: enforcing
    modules:
      - module: kubernetes
        name: mycluster
        args:
          container-registry: container-registry.oracle.com/olcne
          load-balancer: lb.example.com:6443
          master-nodes:
            - control1.example.com:8090
            - control2.example.com:8090
            - control3.example.com:8090
          worker-nodes:
            - worker1.example.com:8090
            - worker2.example.com:8090
            - worker3.example.com:8090
        - module: helm
          name: myhelm
          args:
            helm-kubernetes-module: mycluster
        - module: operator-lifecycle-manager
          name: myolm
          args:
            olm-helm-module: myhelm
        - module: istio
          name: myistio
```

```
    args:
      istio-helm-module: myhelm
```

2. On the operator node, use the `olcnectl provision` command with the `--config-file` option to start the installation. For example:

   ```
   olcnectl provision --config-file myenvironment.yaml
   ```

   There are a number of other command options that you may need, such as the SSH log in credentials, proxy server information, and the option to automatically accept any prompts using the `--yes` option. For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

3. A list of the steps to be performed on each node is displayed and a prompt is displayed to proceed. For example, on a control plane node, the changes may look similar to:

   ```
   ? Apply control-plane configuration on control1.example.com:
   * Install oracle-olcne-release
   ...
   * Install and enable olcne-agent

   Proceed? yes/no(default) yes
   ```

   Enter `yes` to continue. The node is set up.

   Information about the changes on each node is displayed. You need to confirm the set up steps for each node.

   > **Tip:**
   >
   > If you want to avoid accepting the changes on each node, use the `--yes` command option with the `olcnectl provision` command.

4. The nodes are set up with the Oracle Cloud Native Environment platform and the module(s) are installed. You can show information about the environment using the syntax:

   ```
   olcnectl module instances
   --api-server host_name:8091
   --environment-name name
   ```

   > **Tip:**
   >
   > To avoid having to enter the `--api-server` option in future `olcnectl` commands, add the `--update-config` option.

For example:

```
olcnectl module instances \
--api-server operator.example.com:8091 \
--environment-name myenvironment \
--update-config
```

The output looks similar to:

```
INFO[...] Global flag configuration for myenvironment has been written to
the
local Platform config and you don't need to specify them for any future
calls
INSTANCE                    MODULE      STATE
control1.example.com:8090   node        installed
...
mycluster                   kubernetes  installed
```

If you want to see more information about the deployment, use the `olcnectl module report` command. For example:

```
olcnectl module report \
--environment-name myenvironment \
--name mycluster \
--children
```

5. Set up the Kubernetes CLI (`kubectl`) on a control plane node. The `kubectl` command is installed on each control plane node in the cluster. To use it to access the cluster, you need to configure it using the Kubernetes configuration file.

   Log into a control plane node and copy and paste these commands to a terminal in your home directory:

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   export KUBECONFIG=$HOME/.kube/config
   echo 'export KUBECONFIG=$HOME/.kube/config' >> $HOME/.bashrc
   ```

   Verify that you can use the `kubectl` command using any `kubectl` command such as:

   ```
   kubectl get pods --all-namespaces
   ```

   The output looks similar to:

   ```
   NAMESPACE              NAME                          READY   STATUS
   RESTARTS   AGE
   externalip-validat...  externalip-validation-...     1/1     Running
   0          1h
   kube-system            coredns-...                   1/1     Running
   0          1h
   kube-system            coredns-...                   1/1     Running
   ```

```
0          1h
kube-system            etcd-...                        1/1
Running   2          1h
kube-system            etcd-...                        1/1
Running   2          1h
kube-system            kube-apiserver-...              1/1
Running   2          1h
kube-system            kube-apiserver-...              1/1
Running   2          1h
kube-system            kube-controller-manager-...     1/1
Running   5 (1h ago) 1h
kube-system            kube-controller-manager-...     1/1
Running   2          1h
kube-system            kube-flannel-...                1/1
Running   0          1h
kube-system            kube-flannel-...                1/1
Running   0          1h
kube-system            kube-flannel-...                1/1
Running   0          1h
kube-system            kube-proxy-...                  1/1
Running   0          1h
kube-system            kube-proxy-...                  1/1
Running   0          1h
kube-system            kube-proxy-...                  1/1
Running   0          1h
kube-system            kube-scheduler-...              1/1
Running   5 (1h ago) 1h
kube-system            kube-scheduler-...              1/1
Running   2          1h
kubernetes-dashboard   kubernetes-dashboard-...        1/1
Running   0          1h
```

Note:

After the deployment, a Kubernetes configuration file is created in the local directory of the operator node. The file is named `kubeconfig.`*`environment_name`*`.`*`cluster_name`* and contains information about the Kubernetes cluster. This file is created for your convenience and is not required to set up `kubectl` on the control plane nodes.

You may want to use this file to add to a larger Kubernetes configuration file if you have multiple clusters. See the upstream Kubernetes documentation for more information on configuring access to multiple clusters.

9

Quick HA Install using Configuration File on Oracle Cloud Infrastructure

Install a basic deployment of Oracle Cloud Native Environment on Oracle Cloud Infrastructure, including a Kubernetes cluster. Any extra modules you want to install can be added to a configuration file. The example in this topic installs all modules available for Oracle Cloud Infrastructure.

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

This sets up a deployment of Oracle Cloud Native Environment on Oracle Cloud Infrastructure, including a Kubernetes cluster and the Oracle Cloud Infrastructure Cloud Controller Manager module.

Nodes Required: As many nodes as required for High Availability. (See Kubernetes High Availability Requirements). They are:

- **Operator Node**: One node to use as the operator node, which is used to perform the installation using the Platform CLI (`olcnectl`), and to host the Platform API Server.
- **Kubernetes control plane**: At least three nodes to use as Kubernetes control plane nodes.
- **Kubernetes worker**: At least two nodes to use as Kubernetes worker nodes.

Before you begin: Complete the prerequisite set up. See Prerequisites.

To do a quick HA install on Oracle Cloud Infrastructure using a configuration file:

1. Set up the Oracle Cloud Infrastructure load balancer.
    a. Log into the Oracle Cloud Infrastructure User Interface.
    b. Create a load balancer.
    c. Add a backend set to the load balancer using weighted round robin. Set the health check to be TCP port 6443.
    d. Add the control plane nodes to the backend set. Set the port for the control plane nodes to port 6443.
    e. Create a listener for the backend set using TCP port 6443.
2. On the operator node, create an Oracle Cloud Native Environment configuration file for your deployment. For information on creating an Oracle Cloud Native Environment

configuration file, see Platform Command-Line Interface. This example uses the file name `myenvironment.yaml` for the configuration file.

A basic example configuration file that installs the Kubernetes module, and the Oracle Cloud Infrastructure Cloud Controller Manager module is:

```
environments:
  - environment-name: myenvironment
    globals:
      api-server: operator.example.com:8091
      selinux: enforcing
    modules:
      - module: kubernetes
        name: mycluster
        args:
          container-registry: container-registry.oracle.com/olcne
          load-balancer: lb.example.com:6443
          master-nodes:
            - control1.example.com:8090
            - control2.example.com:8090
            - control3.example.com:8090
          worker-nodes:
            - worker1.example.com:8090
            - worker2.example.com:8090
            - worker3.example.com:8090
      - module: helm
        name: myhelm
        args:
          helm-kubernetes-module: mycluster
      - module: oci-ccm
        name: myoci
        args:
          oci-ccm-helm-module: myhelm
          oci-region: us-ashburn-1
          oci-tenancy: ocid1.tenancy.oc1..unique_ID
          oci-compartment: ocid1.compartment.oc1..unique_ID
          oci-user: ocid1.user.oc1..unique_ID
          oci-fingerprint: b5:52:...
          oci-private-key: /home/opc/.oci/oci_api_key.pem
          oci-vcn: ocid1.vcn.oc1..unique_ID
          oci-lb-subnet1: ocid1.subnet.oc1..unique_ID
```

This example configuration file uses the default settings to create a Kubernetes cluster with a three control plane nodes, three worker nodes and uses an external load balancer that is already set up on Oracle Cloud Infrastructure. You should change the nodes listed to those of your own hosts. You should also change the URL to that of your own Oracle Cloud Infrastructure load balancer. A number of values are required to set up the Oracle Cloud Infrastructure Cloud Controller Manager module, and you can find information about what to provide for this module in Storage and Application Load Balancers.

> **Tip:**
>
> Private CA Certificates are automatically generated for communication between the Kubernetes nodes and for the Kubernetes `externalIPs` service. If you want to use your own CA Certificates, or to add additional modules to the configuration file, see the information about these options in Quick Install using Configuration File.

3. On the operator node, use the `olcnectl provision` command with the `--config-file` option to start the installation. For example:

   ```
   olcnectl provision --config-file myenvironment.yaml
   ```

   There are a number of other command options that you may need, such as the SSH log in credentials, proxy server information, and the option to automatically accept any prompts using the `--yes` option. For information on the syntax options for the `olcnectl provision` command, see Platform Command-Line Interface.

4. A list of the steps to be performed on each node is displayed and a prompt is displayed to proceed. For example, on a control plane node, the changes may look similar to:

   ```
   ? Apply control-plane configuration on control1.example.com:
   * Install oracle-olcne-release
   ...
   * Install and enable olcne-agent

   Proceed? yes/no(default) yes
   ```

   Enter `yes` to continue. The node is set up.

   Information about the changes on each node is displayed. You need to confirm the set up steps for each node.

   > **Tip:**
   >
   > If you want to avoid accepting the changes on each node, use the `--yes` command option with the `olcnectl provision` command.

5. The nodes are set up with the Oracle Cloud Native Environment platform and the module(s) are installed. You can show information about the environment using the syntax:

   ```
   olcnectl module instances
   --api-server host_name:8091
   --environment-name name
   ```

> **Tip:**
>
> To avoid having to enter the `--api-server` option in future `olcnectl` commands, add the `--update-config` option.

For example:

```
olcnectl module instances \
--api-server operator.example.com:8091 \
--environment-name myenvironment \
--update-config
```

The output looks similar to:

```
INFO[...] Global flag configuration for myenvironment has been
written to the
local Platform config and you don't need to specify them for any
future calls
INSTANCE                    MODULE      STATE
control1.example.com:8090   node        installed
...
mycluster                   kubernetes  installed
```

If you want to see more information about the deployment, use the `olcnectl module report` command. For example:

```
olcnectl module report \
--environment-name myenvironment \
--name mycluster \
--children
```

6. Set up the Kubernetes CLI (`kubectl`) on a control plane node. The `kubectl` command is installed on each control plane node in the cluster. To use it to access the cluster, you need to configure it using the Kubernetes configuration file.

   Log into a control plane node and copy and paste these commands to a terminal in your home directory:

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   export KUBECONFIG=$HOME/.kube/config
   echo 'export KUBECONFIG=$HOME/.kube/config' >> $HOME/.bashrc
   ```

   Verify that you can use the `kubectl` command using any `kubectl` command such as:

   ```
   kubectl get pods --all-namespaces
   ```

The output looks similar to:

```
NAMESPACE              NAME                           READY   STATUS
RESTARTS   AGE
externalip-validat...  externalip-validation-...      1/1     Running
0          1h
kube-system            coredns-...                    1/1     Running
0          1h
kube-system            coredns-...                    1/1     Running
0          1h
kube-system            etcd-...                       1/1     Running
2          1h
kube-system            etcd-...                       1/1     Running
2          1h
kube-system            kube-apiserver-...             1/1     Running
2          1h
kube-system            kube-apiserver-...             1/1     Running
2          1h
kube-system            kube-controller-manager-...    1/1     Running   5
(1h ago)   1h
kube-system            kube-controller-manager-...    1/1     Running
2          1h
kube-system            kube-flannel-...               1/1     Running
0          1h
kube-system            kube-flannel-...               1/1     Running
0          1h
kube-system            kube-flannel-...               1/1     Running
0          1h
kube-system            kube-proxy-...                 1/1     Running
0          1h
kube-system            kube-proxy-...                 1/1     Running
0          1h
kube-system            kube-proxy-...                 1/1     Running
0          1h
kube-system            kube-scheduler-...             1/1     Running   5
(1h ago)   1h
kube-system            kube-scheduler-...             1/1     Running
2          1h
kubernetes-dashboard   kubernetes-dashboard-...       1/1     Running
0          1h
```

Note:

After the deployment, a Kubernetes configuration file is created in the local directory of the operator node. The file is named `kubeconfig.`*`environment_name`*`.`*`cluster_name`* and contains information about the Kubernetes cluster. This file is created for your convenience and is not required to set up `kubectl` on the control plane nodes.

You may want to use this file to add to a larger Kubernetes configuration file if you have multiple clusters. See the upstream Kubernetes documentation for more information on configuring access to multiple clusters.

10
Retry an Install

If you need to clean up after a failed installation attempt, you should perform the following tasks. This allows you to retry the installation using the

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

`olcnectl provision` command on the same nodes.

To clean up after a failed installation attempt:

1. On the operator node, remove any modules using the syntax:

   ```
   olcnectl module uninstall
   --api-server host:8091
   --environment-name name
   --name name
   ```

   For example:

   ```
   olcnectl module uninstall \
   --api-server operator.example.com:8091 \
   --environment-name myenvironment \
   --name mycluster
   ```

2. On the operator node, remove the environment using the syntax:

   ```
   olcnectl environment delete
   --api-server host:8091
   --environment-name name
   ```

   For example:

   ```
   olcnectl environment delete \
   --api-server operator.example.com:8091 \
   --environment-name myenvironment
   ```

3. Retry the `olcnectl provision` command to create an environment and Kubernetes module.

11

Install Optional Modules

> **Important:**
>
> The software described in this documentation is either in Extended Support or Sustaining Support. See Oracle Open Source Support Policies for more information.
>
> We recommend that you upgrade the software described by this documentation as soon as possible.

After you install the Oracle Cloud Native Environment platform and install a Kubernetes module, you can add any optional modules you want to the environment using the `olcnectl` command on the operator node.

Creating an Oracle Cloud Infrastructure Cloud Controller Manager Module

When you have created and installed a Kubernetes module, you can optionally install the Oracle Cloud Infrastructure Cloud Controller Manager module to set up access to Oracle Cloud Infrastructure storage and application load balancers. This allows you to use Oracle Cloud Infrastructure block volumes to provide persistent storage for Kubernetes applications. This also allows you to create load balancers for Kubernetes applications so they can be accessed externally, from outside the cluster.

For information on installing the Oracle Cloud Infrastructure Cloud Controller Manager module to provide Oracle Cloud Infrastructure storage, see Storage.

For information on installing the Oracle Cloud Infrastructure Cloud Controller Manager module to provide Oracle Cloud Infrastructure application load balancers, see Application Load Balancers.

> **Note:**
>
> The same Oracle Cloud Infrastructure Cloud Controller Manager module is used to set up access to both Oracle Cloud Infrastructure storage and application load balancers. You do not need to deploy separate modules.

Creating a MetalLB Module

When you have created and installed a Kubernetes module, you can optionally install the MetalLB module. MetalLB is a network load balancer for Kubernetes applications running on bare metal hosts. MetalLB allows you to use Kubernetes LoadBalancer services, which

traditionally make use of a cloud provider network load balancer, in a bare metal environment. For information on installing the MetalLB module, see Application Load Balancers.

Creating a Gluster Container Storage Interface Module

When you have created and installed a Kubernetes module, you can optionally install the Gluster Container Storage Interface module to set up access to Gluster storage. This allows you to use a Gluster cluster to provide persistent storage for Kubernetes applications. For information on installing the Gluster Container Storage Interface module, see Storage.

Creating an Operator Lifecycle Manager Module

When you have created and installed a Kubernetes module, you can optionally install the Operator Lifecycle Manager module to manage the installation and lifecycle management of operators in a Kubernetes cluster. For information on installing the Operator Lifecycle Manager module, see Container Orchestration.

Creating an Istio Module

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.