Oracle Cloud Native Environment

Concepts for Release 2

F96190-08
August 2025

Oracle Cloud Native Environment Concepts for Release 2,

F96190-08

Contents

Preface

This document provides an overview of the different components of Oracle Cloud Native Environment (Oracle CNE) and explains key concepts that are essential to working with Oracle CNE.

1 Introduction

Oracle Cloud Native Environment (Oracle CNE) is a fully integrated suite for the development and management of cloud native applications. Oracle CNE delivers a simplified framework for installations, updates, upgrades, and configuration of key features for orchestrating microservices.

Oracle CNE uses Kubernetes to deploy and manage containers. A Kubernetes cluster automatically installs, and configures Kubernetes, CRI-O, runC, and Kata Containers on the Kubernetes nodes.

The Oracle CNE Command Line Interface (CLI) performs the deployment of a Kubernetes cluster, and applications to the cluster.

The Oracle CNE User Interface (UI) can be installed to manage the cluster and applications.

2 Components

The components that make up Oracle CNE are:

- Oracle CNE Command Line Interface (CLI).
- Oracle CNE User Interface (UI).
- Oracle CNE application catalog.
- Oracle Container Host for Kubernetes (OCK) images.
- OCK Image Builder.

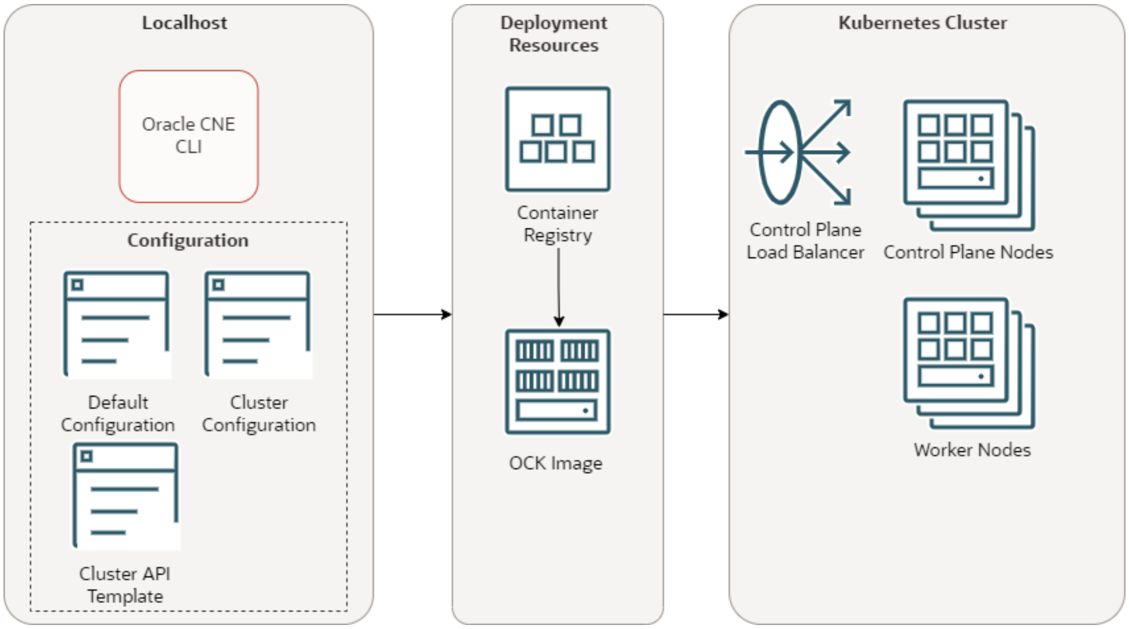
CLI

Introduces the Oracle CNE Command Line Interface (CLI).

The CLI is used to create and manage Kubernetes clusters, using the available cluster providers. The CLI is installed on Oracle Linux 8, or 9. The CLI has some in-built default configuration that can be used to create a basic Kubernetes cluster, or configuration files where you can set up the cluster with the parameters and options you want for the deployment environment.

Where a cluster provider type uses the Kubernetes Cluster API, the CLI can also use Cluster API templates to define clusters to even finer specifications where a configuration option isn't available in a cluster configuration file, but is available in the Cluster API provider.

Figure 2-1 CLI Architecture



The CLI architecture has the following components:

- **CLI**: The CLI used to create and manage Kubernetes clusters. The ocne command.
- **Default configuration**: A YAML file that contains configuration for all ocne commands.
- **Cluster configuration**: A YAML file that contains configuration for a specific Kubernetes cluster.
- **Cluster API template**: A YAML file that contains Cluster Resources for the Kubernetes Cluster API to create a cluster.
- **Container registry**: A container registry used to pull the images used to create nodes in a Kubernetes cluster. The default is the Oracle Container Registry.
- **OCK image**: The OCK image pulled from the container registry, which is used to create Kubernetes nodes.
- **Control plane load balancer**: A load balancer used for High Availability (HA) of the control plane nodes. This might be the default internal load balancer, or an external one.
- **Control plane nodes**: Control plane nodes in a Kubernetes cluster.
- **Worker nodes**: Worker nodes in a Kubernetes cluster.

UI

Introduces the Oracle CNE User Interface (UI).

The Oracle CNE UI provides a web-based interface to manage the maintenance and installation of Kubernetes cluster resources, and applications.

The UI runs in the Kubernetes cluster as a deployment named `ui`, running in the `ocne-system` namespace. A deployment named `ocne-catalog` also runs in the `ocne-system` namespace to serve the application catalog.

The UI is based on the open source Kubernetes UI Headlamp application. For more information on the Headlamp project, see the upstream Headlamp documentation.

Application Catalogs

Learn about application catalogs and cloud native applications in Oracle Cloud Native Environment (Oracle CNE).

An application catalog is a searchable collection of software that can be installed into a Kubernetes cluster. Installed catalogs can be searched using both the Oracle CNE Command Line Interface (CLI) and the User Interface (UI).

Catalogs have a straightforward life cycle. They can be added and removed, but not changed.

Two types of application catalogs can be configured within a cluster: an Oracle catalog, and an external community catalog.

An application catalog is set up in two flavors: a Helm repository, and a service that's compatible with Artifact Hub (an external catalog). The Oracle catalog is a Helm repository, while an external catalog typically points to artifacthub.io and is compatible with Artifact Hub.

A Helm-based catalog is a collection of Helm artifacts, namely an `index.yaml` file, and a set of tarballs. These can be made available and served by any URI that can be read by an Oracle CNE component (the `ocne` CLI, the UI, and Helm).

An external catalog must be compatible with the Artifact Hub API. In practice, this means the external catalogs are served by an instance of Artifact Hub.

Oracle Catalog

The Oracle catalog is a collection of cloud native application software provided by Oracle. Oracle CNE applications are delivered through the Oracle catalog. The Oracle catalog can be accessed in two ways, either using the embedded version in the CLI, or from an application deployed to a Kubernetes cluster.

The embedded catalog is built into the CLI, and is named `embedded`. This catalog can be accessed and queried without deploying a Kubernetes cluster.

The Oracle catalog can also be deployed to a cluster, and is named `Oracle Cloud Native Environment Application Catalog`. The CLI deploys the `ocne-catalog` application from the Oracle Container Registry to a cluster in the `ocne-system` namespace. The `ocne-catalog` application includes a Helm repository, and an instance of NGINX that serves the static content. While the Oracle catalog can be served anywhere that has a container runtime, it's primarily intended to be run within the Kubernetes cluster that consumes its contents.

> **Note**
>
> The list of applications included in the Oracle catalog is available in the upstream Oracle CNE documentation. To see the application list that matches the Oracle CNE release, select the release branch in the **Branches** drop down. For example:
>
> - The branch name for Release 2.0 is release/2.0.
> - The branch name for Release 2.1 is release/2.1.
> - The branch name for Release 2.2 is release/2.2.

The embedded Oracle catalog is updated when you update the CLI. The Oracle catalog running as an application in the cluster can be updated using the `ocne application update` command. This means that the catalog contents might differ.

External Catalogs

External catalogs can be added using the `ocne catalog add` CLI command. External catalogs are added using a Kubernetes external service resource. An external catalog isn't added to the cluster, instead, it's referenced. The catalog data isn't installed, or served from the cluster. An example of an external catalog is the Artifact Hub catalog. Artifact Hub is a web-based application that provides cloud native packages you can install into a Kubernetes cluster. For more information on Artifact Hub, see:

https://artifacthub.io/

Oracle Container Host for Kubernetes Image

Describes the Oracle Container Host for Kubernetes (OCK) image used to create nodes in a Kubernetes cluster.

Oracle CNE includes a CLI that can manage the life cycle of Kubernetes clusters, using OSTree based container images. The container image includes both the host Oracle Linux OS, and the Kubernetes software distribution. The image is deployed to hosts or Virtual Machines (VMs) to create nodes in a Kubernetes cluster. This image is referred to in this documentation as the Oracle Container Host for Kubernetes (OCK) image.

The OCK image is distributed on the Oracle Container Registry in the following formats:

Bootable image
This is a container image in the Qcow2 format, available at:
container-registry.oracle.com/olcne/ock
The bootable image contains a single VM image in the Qcow2 format, and is used to create boot media for virtualized platforms. This image is used as the boot media for clusters created with the libvirt and Kubernetes Cluster API providers (the OCI and Oracle Linux Virtualization Manager providers).
By default, the image is configured to work with the libvirt provider. A conversion of the boot image to the appropriate format for OCI and Oracle Linux Virtualization Manager is done when you upload the image.

OSTree image
This is an OSTree commit based container image, available at:
container-registry.oracle.com/olcne/ock-ostree
This image is used as the basis for an OSTree archive for customized installations using the Bring Your Own provider.
This image is also used for updating cluster nodes to stage patch updates, and to update to the next Kubernetes minor release.
For information on OSTree containers, see the upstream OSTree documentation.

Both images use the container label for the Kubernetes version they match, for example, 1.32.

Configuration Files

Describes the configuration files that can be used to customize the CLI command.

Kubernetes clusters and applications can be configured through a set of YAML configuration files and `ocne` command line arguments. Configuration is layered, with each layer of configuration taking precedence over the previous layer. The layered structure provides convenient reuse of parameters that would otherwise be duplicated into every deployment.

You can configure `ocne` subcommands using three hierarchical methods. The methods are (in hierarchical order):

- Global defaults in the default configuration file, set in the `$HOME/.ocne/defaults.yaml` file.
- Kubernetes cluster configuration files. These files set the options for individual clusters and can be any name.
- Options provided with the `ocne` command.

Global defaults can be overridden by a global configuration file. Those values can in turn be overridden by a cluster or application specific configuration file. Finally, that entire stack of configuration can be overridden by `ocne` command line options.

For information on cluster configuration files, see Oracle Cloud Native Environment: Kubernetes Clusters.

Kubernetes Cluster API Templates

Describes Kubernetes Cluster API template files in Oracle CNE.

Some Oracle CNE cluster providers use the Kubernetes Cluster API to provision and manage clusters. The default cluster settings create a useful cluster, but, it's likely that extra configuration might be required. To customize a deployment, you can generate, and edit a

Kubernetes Cluster API template to use as a basis for the cluster. For more information on the Kubernetes Cluster API, see the upstream documentation.

The `ocne cluster template` command is used to create a cluster template, and uses the default configuration and any cluster configuration you set to generate the template. Depending on the provider type, it might also fetch things such as compute image OCIDs from the configured OCI compartment automatically. The resulting YAML file contains the Cluster Resources for the Kubernetes Cluster API to create a cluster, using all the configuration you have on the local system. This template can be included as an option in a cluster configuration file and used to create a cluster.

3

Cluster Providers

Oracle CNE can be used to create Kubernetes clusters on:

- Kernel-based Virtual Machines (KVM) using the `libvirt` provider.
- Oracle Linux Virtualization Manager using the `olvm` provider.
- Oracle Cloud Infrastructure (OCI) using the `oci` provider.
- Custom installations for bare metal or other platforms using the `byo` provider.

The `libvirt` provider is the default cluster provider, and can be used to provision Kubernetes clusters using Kernel-based Virtual Machines (KVM). The default KVM stack includes libvirt, and is included, by default, with Oracle Linux.

Kubernetes clusters are deployed to Oracle Linux Virtualization Manager using the `olvm` provider. The `olvm` provider is an implementation of the Kubernetes Cluster API. For information about the Kubernetes Cluster API, see the upstream documentation. The `olvm` provider uses the oVirt REST API to communicate with Oracle Linux Virtualization Manager. For information on the oVirt REST API, see the oVirt REST API upstream documentation.

Kubernetes clusters are deployed to OCI using the `oci` provider. The `oci` provider uses the Kubernetes Cluster API Provider for OCI to perform the deployment. This is an implementation of the Kubernetes Cluster API. The Kubernetes Cluster API is implemented as Kubernetes Custom Resources (CRs), that are serviced by applications running in a Kubernetes cluster. The Kubernetes Cluster API has a large interface and is explained in the upstream documentation. For information on the Kubernetes Cluster API, see the Kubernetes Cluster API documentation. For information on the Cluster API implementation for OCI, see the Kubernetes Cluster API Provider for OCI documentation.

You can make custom installations of the Oracle Container Host for Kubernetes (OCK) image on arbitrary platforms. This means you can create a Kubernetes cluster using bare metal or other virtual instances, not provided explicitly by Oracle CNE. These installations are known as *Bring Your Own* (BYO) installations. You use the `byo` provider to perform these installations.

libvirt Provider

Use the `libvirt` provider to create KVM based Kubernetes clusters with libvirt.

The `libvirt` provider is the default cluster provider, and can be used to provision Kubernetes clusters using Kernel-based Virtual Machines (KVM). The default KVM stack includes libvirt, and is included, by default, with Oracle Linux.

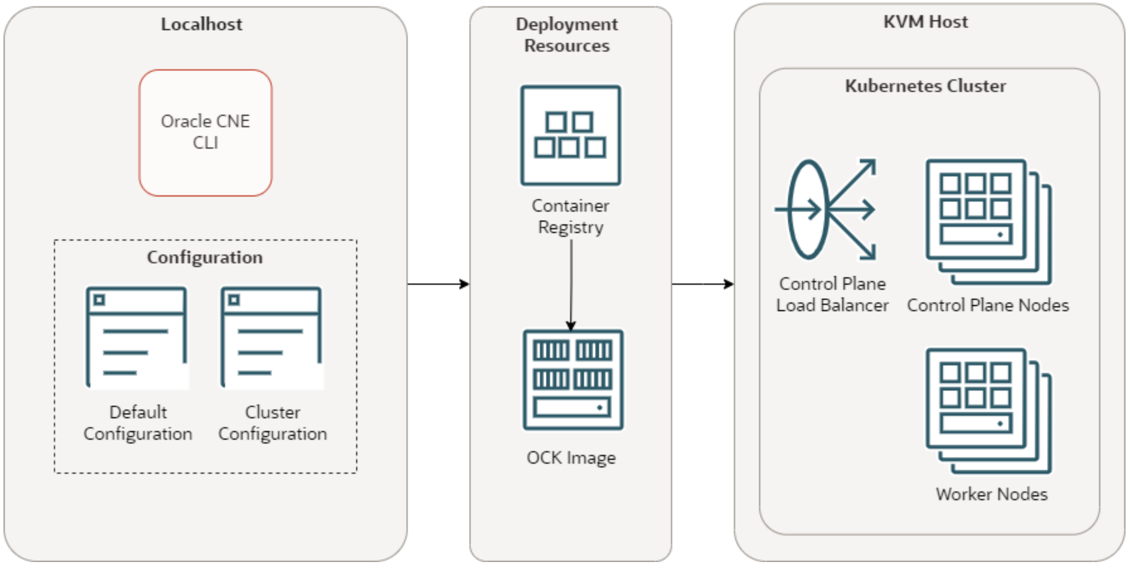
> **Note**
>
> We recommend the Oracle KVM stack as this KVM version offers many more features for Oracle Linux systems. For information on the Oracle KVM stack and libvirt, see the Oracle Linux 9: KVM User's Guide or the Oracle Linux 8: KVM User's Guide.

The system used to create libvirt clusters must be a 64-bit x86 or 64-bit ARM system running Oracle Linux 8 or 9, and include the Unbreakable Enterprise Kernel Release 7 (UEK R7).

The `libvirt` provider provisions Kubernetes clusters using libvirt on a single host, and is useful for creating and destroying Kubernetes clusters for testing and development. While the `libvirt` provider can be used for test and development clusters, it does deploy a production worthy cluster configuration.

> **Important**
>
> As all libvirt cluster nodes are running on a single host, be aware that if the host running the cluster goes down, so do all the cluster nodes.

Figure 3-1 libvirt Cluster Architecture

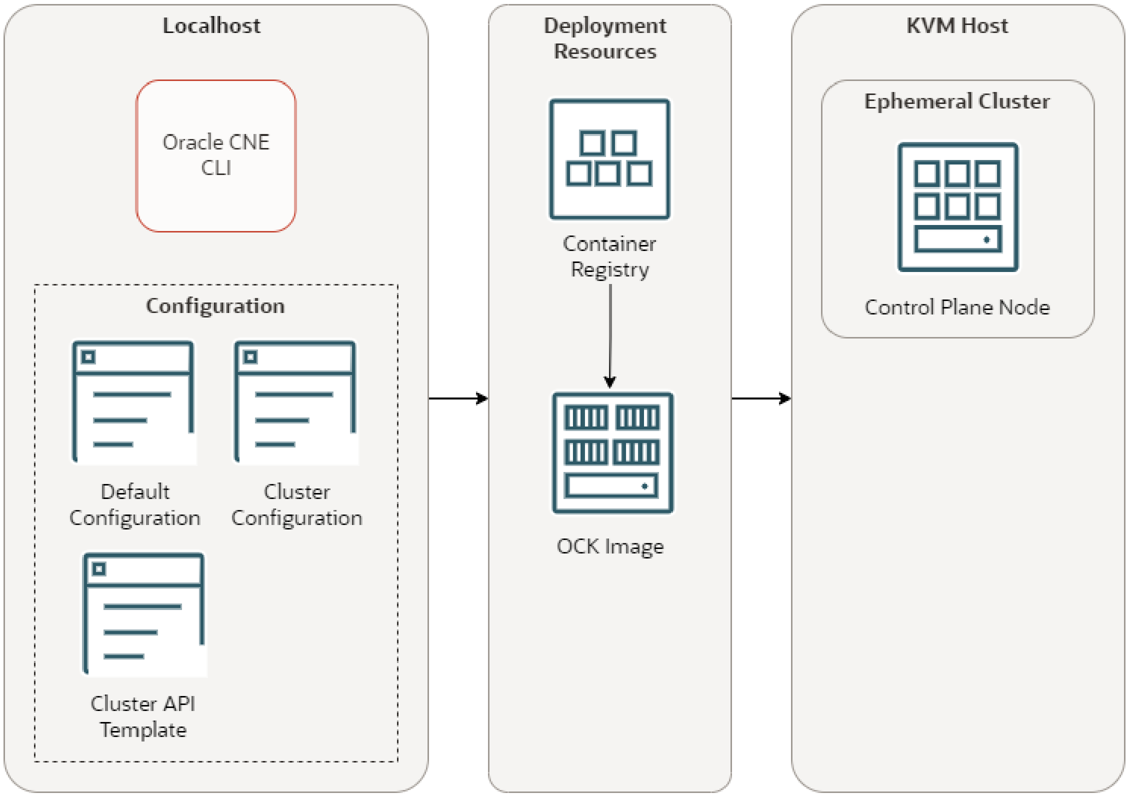


The libvirt cluster architecture has the following components:

- **CLI**: The CLI used to create and manage Kubernetes clusters. The `ocne` command.
- **Default configuration**: A YAML file that contains configuration for all `ocne` commands.
- **Cluster configuration**: A YAML file that contains configuration for a specific Kubernetes cluster.
- **Container registry**: A container registry used to pull the images used to create nodes in a Kubernetes cluster. The default is the Oracle Container Registry.
- **OCK image**: The OCK image pulled from the container registry, which is used to create Kubernetes nodes.
- **Control plane load balancer**: A load balancer used for High Availability (HA) of the control plane nodes.
- **Control plane nodes**: Control plane nodes in a Kubernetes cluster.
- **Worker nodes**: Worker nodes in a Kubernetes cluster.

The `libvirt` provider is also used to provision Kubernetes clusters when using some CLI commands. This cluster type is often referred to as an *ephemeral cluster*. An ephemeral cluster is a single node cluster that lives for a short time and is created and destroyed as needed by the CLI. An existing cluster can also be used as an ephemeral cluster by including the location of a `kubeconfig` file as an option with CLI commands.

Figure 3-2 libvirt Ephemeral Cluster



The ephemeral cluster architecture has the following components:

- **CLI**: The CLI used to create and manage Kubernetes clusters. The `ocne` command.
- **Default configuration**: A YAML file that contains configuration for all `ocne` commands.
- **Cluster configuration**: A YAML file that contains configuration for a specific Kubernetes cluster.
- **Cluster API template**: A YAML file that contains Cluster Resources for the Kubernetes Cluster API to create a cluster.
- **Container registry**: A container registry used to pull the images used to create nodes in a Kubernetes cluster. The default is the Oracle Container Registry.
- **OCK image**: The OCK image pulled from the container registry, which is used to create Kubernetes nodes.
- **Ephemeral cluster**: A temporary Kubernetes cluster used to perform a CLI command. The default for this is a single node cluster created with the libvirt provider on the localhost. This might also be an external cluster.

Single and multi node clusters can be created on Oracle Linux 8 and 9, on both 64-bit x86 and 64-bit ARM systems. Because all cluster nodes run on a single host, it's not possible to create hybrid clusters. However, it's possible to use an ARM system to create a remote cluster on x86 hardware and, conversely, x86 hardware can be used to create a remote cluster on ARM.

The libvirt provider requires the target system to be running libvirt and requires that the user be configured to have access to libvirt. Oracle CNE implements a libvirt connection using the legacy single-socket client. If local libvirt clusters are created, the UNIX domain socket is used.

To create Kubernetes clusters on a remote system, enable a remote transport mechanism for libvirt. We recommend you set up SSH key-based authentication to the remote system as a normal user, and that you configure the user with the privilege to run libvirt. You can, however, use any of the libvirt remote transport options. For more information on libvirt remote transports, see the upstream libvirt documentation.

Most remote cluster deployments leverage the qemu+ssh transport, which uses SSH to tunnel the UNIX domain socket back to the CLI. Oracle CNE doesn't configure the libvirt transports or system services. This must be set up correctly, according to the documentation for the OS.

Clusters created with the libvirt provider create a tunnel so the cluster can be accessed through a port on the host where the cluster is deployed. The port range starts at 6443 and increments from there. As clusters are deleted, the ports are freed. If a cluster is created on a remote system, ensure a range of ports are accessible through the system firewall, starting at 6443.

> **Important**
>
> You can disable the firewall in a testing environment, however we don't recommend this for production systems.

Use the ocne cluster start command to create a Kubernetes cluster using the libvirt provider. As this provider is the default, you don't need to specify the provider type. For example:

```
ocne cluster start
```

This command creates a single node cluster using all the default options, and installs the UI and application catalog.

You can add extra command line options to the ocne cluster start command to set up the cluster with non default settings, such as the number of control plane and worker nodes. For information on these command options, see Oracle Cloud Native Environment: CLI.

You can also customize the default settings by adding options to the default configuration file, or a configuration file specific to the cluster you want to create. For information on these configuration files, see Oracle Cloud Native Environment: Kubernetes Clusters and Oracle Cloud Native Environment: CLI.

For clusters started on systems with access to privileged libvirt instances, two kubeconfig files are created when you create a cluster, one for access to the local cluster, and one that can be used on the remote cluster host.

Oracle Linux Virtualization Manager Provider

Use the olvm provider to create Kubernetes clusters on Oracle Linux Virtualization Manager.

Kubernetes clusters are deployed to Oracle Linux Virtualization Manager using the olvm provider. The olvm provider is an implementation of the Kubernetes Cluster API. For information about the Kubernetes Cluster API, see the upstream documentation. The olvm provider uses the oVirt REST API to communicate with Oracle Linux Virtualization Manager. For information on the oVirt REST API, see the oVirt REST API upstream documentation.

Creating a cluster on Oracle Linux Virtualization Manager requires you to provide the OAuth 2.0 credentials to an existing instance of Oracle Linux Virtualization Manager. For information on OAuth 2.0, see the oVirt upstream documentation.

The `olvm` provider implements an infrastructure Cluster controller (OLVMCluster) Custom Resource Definition (CRD), and an infrastructure Machine controller (OLVMMachine) CRD, as Cluster API controllers. For each cluster, one OLVMCluster Custom Resource (CR) is needed, and an OLVMMachine CR for control plane nodes, and another for worker nodes. The configuration to create these CRs is set in a cluster configuration file.

The controllers that implement the Kubernetes Cluster API run inside a Kubernetes cluster. These clusters are known as *management clusters*. Management clusters control the life cycle of other clusters, known as *workload clusters*. A workload cluster can be its own management cluster.

Using the Kubernetes Cluster API to deploy a cluster on Oracle Linux Virtualization Manager requires that a Kubernetes cluster is available to use as the management cluster. Any running cluster can be used. To set the management cluster, set the `KUBECONFIG` environment variable, or use the `--kubeconfig` option of `ocne` commands. You could also set this cluster using a configuration file. If no management cluster is available, a cluster is created automatically using the `libvirt` provider, with the default configuration.

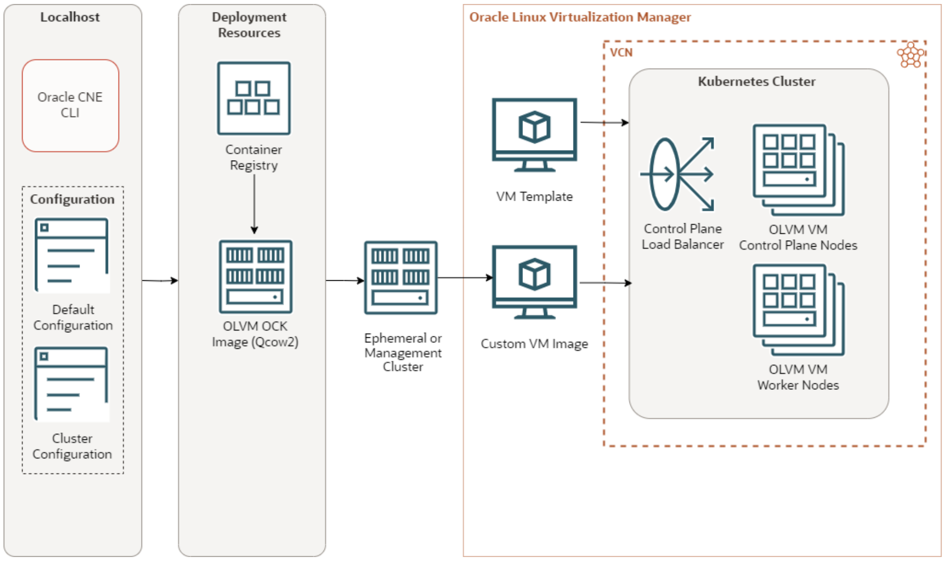
When a workload cluster has been deployed, it's managed using the Cluster API resources in the management cluster.

A workload cluster can be its own management cluster. This is known as a *self-managed cluster*. In a self-managed cluster, when the workload cluster has been deployed, the Cluster API resources are migrated into the workload cluster.

The `olvm` provider can be deployed using IPv4 IP addresses, or as a dual stack configuration using both IPv4 and IPv6 IP addresses. IPv6 on its own can't be used.

> **ⓘ Note**
>
> Dynamic Host Configuration Protocol (DHCP) can't be used to assign IP addresses to the VMs used for Kubernetes nodes. Instead, provide a range of IP addresses for VMs to use as Kubernetes nodes, and set an IP address for the built-in Keepalived and NGINX load balancer for control plane nodes (the virtual IP).

Figure 3-3 Oracle Linux Virtualization Manager Cluster



The Oracle Linux Virtualization Manager cluster architecture has the following components:

- **CLI**: The CLI used to create and manage Kubernetes clusters. The `ocne` command.
- **Default configuration**: A YAML file that contains configuration for all `ocne` commands.
- **Cluster configuration**: A YAML file that contains configuration for a specific Kubernetes cluster.
- **Container registry**: A container registry used to pull the images used to create nodes in a Kubernetes cluster. The default is the Oracle Container Registry.
- **Oracle Linux Virtualization Manager OCK image**: The CLI is used to create this image, based on the OCK image, pulled from the container registry. The CLI is then used to upload this image to Oracle Linux Virtualization Manager.
- **Ephemeral or management cluster**: A Kubernetes cluster used to perform a CLI command. This cluster might also be used to start the cluster services, or to manage the cluster.
- **VM template**: An Oracle Linux Virtualization Manager VM template, used to create VMs in the Kubernetes cluster.
- **Custom VM image**: The OCK image is loaded into Oracle Linux Virtualization Manager as a custom VM image.
- **Control plane load balancer**: A network load balancer used for High Availability (HA) of the control plane nodes.
- **Control plane nodes**: VMs running control plane nodes in a Kubernetes cluster.
- **Worker nodes**: VMs running worker nodes in a Kubernetes cluster.

OCI Provider

Use the `oci` provider to create Kubernetes clusters on Oracle Cloud Infrastructure (OCI).

Kubernetes clusters are deployed to OCI using the `oci` provider. The `oci` provider uses the Kubernetes Cluster API Provider for OCI to perform the deployment. This is an implementation of the Kubernetes Cluster API. The Kubernetes Cluster API is implemented as Kubernetes Custom Resources (CRs), that are serviced by applications running in a Kubernetes cluster. The Kubernetes Cluster API has a large interface and is explained in the upstream documentation. For information on the Kubernetes Cluster API, see the Kubernetes Cluster API documentation. For information on the Cluster API implementation for OCI, see the Kubernetes Cluster API Provider for OCI documentation.

Creating a cluster on OCI requires you to provide the credentials to an existing tenancy. The required privileges depend on the configuration of the cluster that's created. For some deployments, it might be enough to have the privileges to create and destroy compute instances. For other deployments, more privilege might be required.

Clusters are deployed into specific compartments. The `oci` provider requires that a compartment is available. Compartments can be specified either by the Oracle Cloud Identifier (OCID), or by its path in the compartment hierarchy, for example, `parentcompartment/mycompartment`.

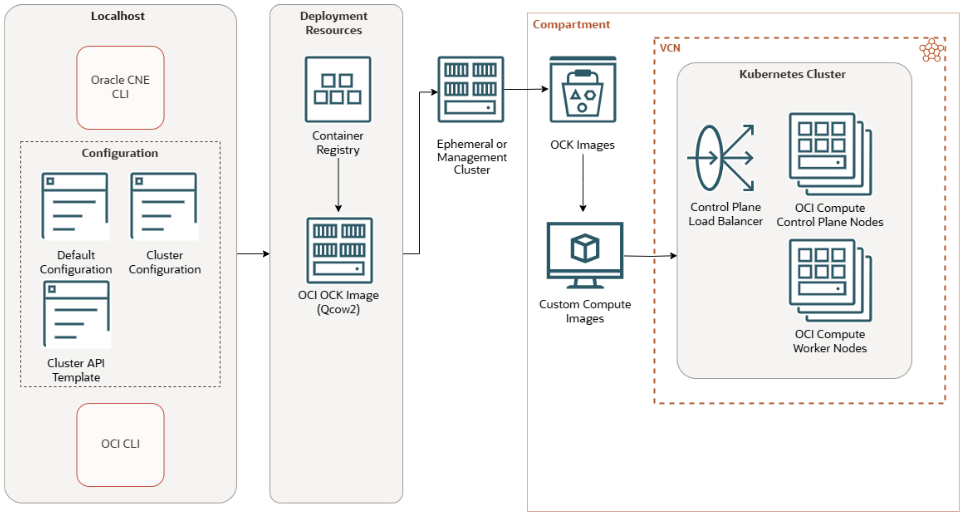
The controllers that implement the Kubernetes Cluster API run inside a Kubernetes cluster. These clusters are known as *management clusters*. Management clusters control the life cycle of other clusters, known as *workload clusters*. A workload cluster can be its own management cluster.

Using the Kubernetes Cluster API to deploy a cluster on OCI requires that a Kubernetes cluster is available to use as the management cluster. Any running cluster can be used. To set the management cluster, set the `KUBECONFIG` environment variable, or use the `--kubeconfig` option of `ocne` commands. You could also set this cluster using a configuration file. If no management cluster is available, a cluster is created automatically using the `libvirt` provider, with the default configuration.

When a workload cluster has been deployed, it's managed using the Kubernetes Cluster API resources in the management cluster.

A workload cluster can be its own management cluster. This is known as a *self-managed cluster*. In a self-managed cluster, when the workload cluster has been deployed, the Cluster API resources are migrated into the workload cluster.

Figure 3-4 OCI Cluster



The OCI cluster architecture has the following components:

- **CLI**: The CLI used to create and manage Kubernetes clusters. The `ocne` command.
- **Default configuration**: A YAML file that contains configuration for all `ocne` commands.
- **Cluster configuration**: A YAML file that contains configuration for a specific Kubernetes cluster.
- **Cluster API template**: A YAML file that contains Cluster Resources for the Kubernetes Cluster API to create a cluster.
- **OCI CLI**: The OCI CLI is installed on the localhost, including the configuration to read and write to the tenancy and compartment.
- **Container registry**: A container registry used to pull the images used to create nodes in a Kubernetes cluster. The default is the Oracle Container Registry.
- **OCI OCK image**: The CLI is used to create this image, based on the OCK image, pulled from the container registry. The CLI is then used to upload this image to OCI.
- **Ephemeral or management cluster**: A Kubernetes cluster used to perform a CLI command. This cluster might also be used to start the cluster services, or to manage the cluster.
- **Compartment**: An OCI compartment in which the cluster is created.
- **OCK images**: The OCK image is loaded into an Object Storage bucket. When the upload is complete, a custom compute image is created from the OCK image.
- **Custom compute images**: The OCK image is available as a custom compute image and can be used to create compute nodes in a Kubernetes cluster.
- **Control plane load balancer**: A network load balancer used for High Availability (HA) of the control plane nodes.
- **Control plane nodes**: Compute instances running control plane nodes in a Kubernetes cluster.
- **Worker nodes**: Compute instances running worker nodes in a Kubernetes cluster.

Bring Your Own Provider

Use the `byo` provider to create Kubernetes clusters using bare metal or other virtual instances not provided explicitly by Oracle CNE.

You can make custom installations of the Oracle Container Host for Kubernetes (OCK) image on arbitrary platforms. This means you can create a Kubernetes cluster using bare metal or other virtual instances, not provided explicitly by Oracle CNE. These installations are known as *Bring Your Own* (BYO) installations. You use the `byo` provider to perform these installations.

You can install the OCK image into environments that require manual installation of individual hosts. A common case is a bare metal deployment. Another is a case where a standardized *golden image* for an OS is required. This install type is intended to cover all cases where deploying the standard OS boot image isn't possible.

This installation process is used to create new Kubernetes clusters or expand existing ones. This installation type leverages the Anaconda and Kickstart installation options of Oracle Linux to deploy OSTree content onto a host.

The BYO installation consists of a handful of components, spread across several Oracle CNE CLI commands.

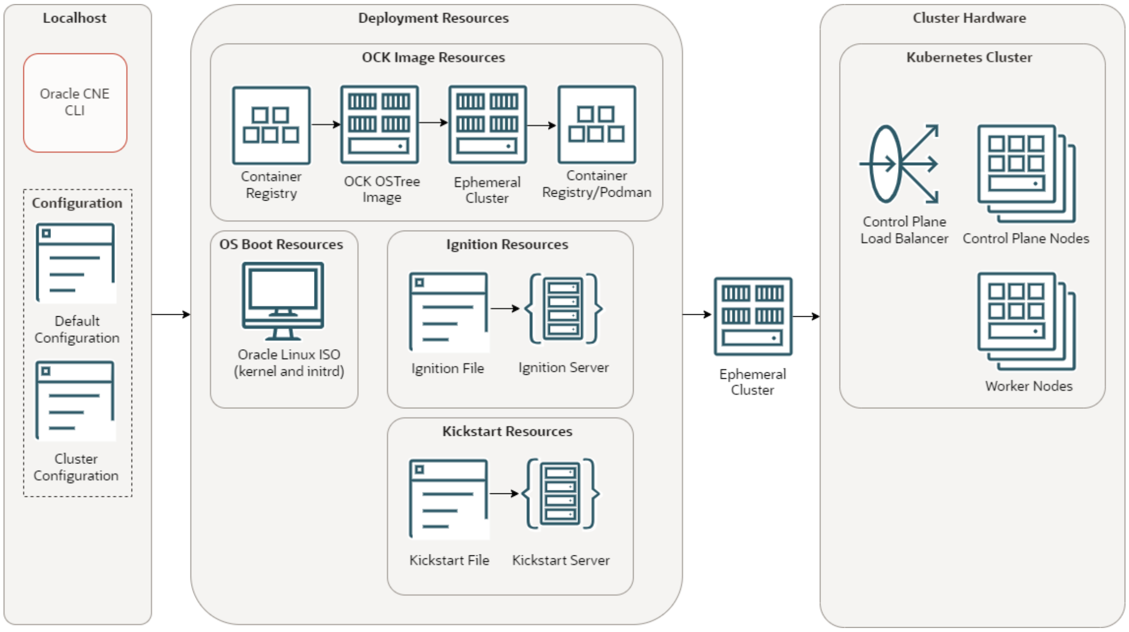
- The `ocne image create` command is used to download OSTree content from official Oracle CNE sources, and convert them into a format that can be used for a custom installation. It also creates an OSTree archive server.
- The `ocne image upload` command is used to copy the OSTree archive server to a container registry. You can also use Podman to serve the OSTree archive locally if you don't want to use a container registry. You can load the image into any target available with the Open Container Initiative transports and formats. See `containers-transports(5)` for available options.
- The `ocne cluster start` command generates Ignition content that's consumed by the newly installed host during boot. This Ignition information is used to start a new Kubernetes cluster. You specify what you want to include in the Ignition configuration.
- The `ocne cluster join` command generates Ignition content that's used to add nodes to an existing Kubernetes cluster.

For more information on OSTree, see the upstream OSTree documentation.

For more information on Ignition, see the upstream Ignition documentation.

BYO installations of the OCK image use an OSTree archive with Anaconda and Kickstart to create bootable media. When the base OS installation is complete, Ignition is used to complete the first-boot configuration and provision Kubernetes services on the host.

Figure 3-5 BYO Cluster



The BYO cluster architecture has the following components:

- **CLI**: The CLI used to create and manage Kubernetes clusters. The `ocne` command.
- **Default configuration**: A YAML file that contains configuration for all `ocne` commands.
- **Cluster configuration**: A YAML file that contains configuration for a specific Kubernetes cluster.
- **Container registry**: A container registry used to pull the OCK OSTree images. The default is the Oracle Container Registry.

- **OCK OSTree Image**: The OCK OSTree image pulled from the container registry.
- **Ephemeral cluster**: A temporary Kubernetes cluster used to perform a CLI command.
- **Container registry/Podman**: A container registry or container server, such as Podman, used to serve the OCK OSTree images.
- **Oracle Linux ISO**: An ISO file to serve the kernel and initrd to use for the OS on nodes.
- **Ignition file**: An Ignition file, generated by the CLI, used to join nodes to a cluster.
- **Ignition server**: The Ignition file, loaded into a method that serves Ignition files.
- **Kickstart file**: A Kickstart file that provides the location of the OCK OSTree image, Ignition file, and the OS kernel and initrd.
- **Kickstart server**: The Kickstart file, loaded into a method that servers Kickstart files.
- **Ephemeral cluster**: A temporary Kubernetes cluster used to perform a CLI command.
- **Control plane load balancer**: A load balancer used for High Availability (HA) of the control plane nodes. This might be the default internal load balancer, or an external one.
- **Control plane nodes**: Control plane nodes in a Kubernetes cluster.
- **Worker nodes**: Worker nodes in a Kubernetes cluster.

4

Cluster Administration

Describes administration of Kubernetes clusters using the CLI.

This chapter contains information on using the CLI to administer Kubernetes clusters.

Cluster Updates
The `ocne node update` command is used to update the Oracle Container Host for Kubernetes (OCK) image on nodes in the cluster. Updating the OCK image is used for patch updates, and for minor Kubernetes updates.

Cluster Backups
Backups of a cluster can be done with the `ocne cluster backup` command.

Cluster Analysis
The `ocne cluster dump` and `ocne cluster analyze` commands are used to create and analyze a dump of cluster and node data from a Kubernetes cluster. Analyzing a cluster is useful for debugging and getting detailed information about a cluster.

OS Console
The `ocne cluster console` command is used to connect to the OS console of a node in a cluster. The console provides a method to connect to the host in a `chroot`ed environment to perform debugging or inspection of the host's OS.

Cluster Updates

Learn how to update a Kubernetes cluster by updating the Oracle Container Host for Kubernetes (OCK) image on each Kubernetes node.

This section shows you how to update nodes to the latest Kubernetes patch release, or to update them to the next Kubernetes minor release.

Patch releases include errata updates and might include Common Vulnerabilities and Exposures (CVE) fixes, Kubernetes updates, OS updates, and so on. An update to the next Kubernetes minor version is performed in the same way as patch updates, with one extra step to set the Kubernetes version number.

Oracle CNE delivers all updates through updated Oracle Container Host for Kubernetes (OCK) images. Updates are delivered through an OCK image that's specific to the Kubernetes minor version, for example for Kubernetes Release 1.32.

Each node periodically polls the container registry to check for updates to the OCK image it's running, or for an image for the target Kubernetes version if you're upgrading Kubernetes. When you set the Kubernetes version for an upgrade, the image for that version is pulled and staged on the nodes in the cluster. Patch updates are downloaded to each node automatically and don't need to be staged before a node update.

When an update is available, use the `ocne node update` command to reboot a node to use the new image. Running the `ocne node update` command for a node completes the following actions:

1. The node is drained (using the `kubectl drain` command) from the cluster. This evicts the pods from the node.
2. The host OCK image is installed on the node, and the node is restarted.
3. The node is returned to the cluster (using the `kubectl uncordon` command) and is made available to run pods.

Update nodes sequentially, starting with the control plane nodes.

> **Tip**
>
> To save time, you can start the update process as soon as one of the control plane nodes has been annotated as having an update available.

You can update a Highly Available cluster without bringing the cluster down. As one control plane node is taken offline, another control plane node takes control of the cluster. In a cluster with a single control plane node, the control plane node is offline for a short time while the update is performed.

If applications are running on more than one worker node, they remain up, and available, during an update.

Cluster Backups

Learn about backing up a Kubernetes cluster using the CLI.

Adopting a back up strategy to protect a Kubernetes cluster against control plane node failures is important, especially for clusters with only one control plane node. High availability clusters with many control plane nodes also need a fallback plan if the resilience provided by the replication and fail over functionality has been exceeded.

The state for Kubernetes clusters is maintained in an `etcd` database. Access to the database is shared between all Kubernetes API Server instances. Taking regular backups of the `etcd` database is a critical part of a Kubernetes disaster recovery plan.

Typically, the backup contains sensitive data, such as Kubernetes Secret objects, so care must be taken to store the backups in a secure location.

If restoring from an `etcd` backup is part of a disaster recovery strategy, the integrity of the backup file is important. Backups must therefore be stored in a location with integrity safeguards.

> **Important**
>
> Only the key containers required for the Kubernetes control plane node are backed up. No application containers are backed up.

You don't need to bring down the cluster to perform a back up as part of a disaster recovery plan. Use the `ocne cluster backup` command to back up the key containers and manifests for all the control plane nodes in the cluster (the `etcd` database).

Important

The CLI doesn't provide a command to restore a cluster from an `etcd` database backup. For information on restoring a cluster using the `etcd` backup, see the upstream Kubernetes documentation.

OS Console

Learn how to access a Kubernetes node's OS console using the `ocne cluster console` command.

Oracle CNE systems are administered through Kubernetes. If you need to directly access a node's OS for debugging and testing purposes, use the `ocne cluster console` command to start an administration console.

The console can be started with extra debugging tools that can be used for investigation and diagnosis purposes, by including the `--toolbox` option.

The `ocne cluster console` command can also be used with the `-- command` option to run commands on a node, without directly interacting with the shell. This might be helpful to return information about a node, without connecting directly to the console.

By default, the console session starts with the initial working directory set to root (`/`). If you need to access services that run on the node itself, for example the `ocne-update.service`, you can run the `chroot /hostroot` command, and `chroot` to the local file system of the node. Or, you can start the console session already `chroot`ed to the node file system, using the `--direct` option.

The `ocne cluster console` command is the method you use to access a node's OS in the cluster. The only reason to access a node using some other method, such as SSH, or a serial console, is when the node can't be accessed using this method.

For information on the credentials to use for SSH, see Oracle Cloud Native Environment: Kubernetes Clusters.

5 Glossary

The following terms are used in this documentation.

application
Cloud native software that can be installed into a Kubernetes cluster. Cloud native applications are designed to take advantage of cluster resources and scaling capabilities.

application catalog
A searchable collection of cloud native software applications that can be installed into a Kubernetes cluster.

application template
An application template contains the configuration options for a specific application. These are a set of Helm values. They can be extracted from an application catalog and viewed, saved to a file, or edited directly. The template can then be used to install the application using the provided configuration.

boot image
An Oracle Container Host for Kubernetes (OCK) Virtual Machine image in Qcow2 format. The boot image is used to create cluster nodes.
See also: Oracle Container Host for Kubernetes (OCK) image.

Bring Your Own provider
The Bring Your Own (`byo`) provider is used to create Kubernetes clusters using bare metal or other virtual instances not provided explicitly by Oracle CNE.

CLI
The Oracle CNE Command Line Interface (CLI). This is the command line tool used to create and manage Kubernetes clusters in Oracle CNE. The `ocne` command.

cluster configuration file
A set of configuration options in a YAML file that are applied to a specific Kubernetes cluster.

cluster API template
A file that contains all the information required to create a Kubernetes cluster using the Kubernetes Cluster API. The `ocne cluster template` command is used to create this file. Save and edit this template to create clusters using the Kubernetes Cluster API.

container registry
A repository of container images. For example, the Oracle Container Registry.

default configuration file
A YAML file used to set common, environment-specific values for CLI commands.

ephemeral cluster
A single node cluster created with the libvirt provider that's short-lived and is created and destroyed as needed by the CLI.

Kubernetes Cluster API
An API to provision, run, and maintain Kubernetes clusters. The API an be extended by providing Custom Resource Definitions (CRDs) to manage the infrastructure in a declarative way. The Oracle Cloud Infrastructure (OCI) provider is an implementation of the Kubernetes Cluster API.

libvirt provider
The libvirt provider can be used to provision Kubernetes clusters using Kernel-based Virtual Machines (KVM). The default KVM stack includes libvirt, and is included by default, with Oracle Linux. Libvirt is the default provider.

management cluster
A Kubernetes cluster that controls the life cycle of workload clusters. The controllers that implement the Kubernetes Cluster API run inside a management cluster. A workload cluster can be its own management cluster. Management clusters can also be ephemeral clusters, depending on the context.
See also: ephemeral cluster, and workload cluster.

node
A single virtual or physical machine within a Kubernetes cluster.

Oracle Container Host for Kubernetes (OCK) image
A container image that contains the Oracle Linux OS and the Kubernetes software distribution. The image is used to create nodes in a Kubernetes cluster. The container image is available as a bootable Qcow2 image, and an OSTree based image. The image can be manipulated for custom installations.

OCI provider
The OCI (`oci`) provider is used to create Kubernetes clusters on OCI. The `oci` provider uses the Kubernetes Cluster API Provider for OCI to perform the deployment, which is an implementation of the Kubernetes Cluster API.

Oracle Cloud Infrastructure provider
The long name for the OCI provider. See OCI provider.

Oracle Linux Virtualization Manager provider
The Oracle Linux Virtualization Manager (`olvm`) provider is used to create Kubernetes clusters on Oracle Linux Virtualization Manager. The `olvm` provider implements the Kubernetes Cluster API and uses the oVirt REST API to communicate with Oracle Linux Virtualization Manager.

OS console
An administration console to directly access a node's OS, for testing, and debugging purposes.

OSTree archive image
A version of the Oracle Container Host for Kubernetes (OCK) image that has been converted to an OSTree archive and served over HTTP. This image type is used to update cluster nodes and generate customized installations using the Bring Your Own provider.
See also: Oracle Container Host for Kubernetes (OCK) image.

self-managed cluster
A Kubernetes workload cluster that also acts as a management cluster. When the cluster has been deployed by a management or ephemeral cluster, the Kubernetes Cluster API resources are migrated into the new self-managed cluster.
See also: management cluster.

UI
The Oracle CNE User Interface (UI) provides a web-based interface to manage the maintenance and installation of Kubernetes cluster resources, and cloud native applications. The UI runs in the Kubernetes cluster as a deployment named `ui`, in the `ocne-system` namespace.

workload cluster
A Kubernetes cluster whose life cycle is controlled by a management cluster. A workload cluster might also be a management cluster.
See also: management cluster.