Oracle Cloud Native Environment

Concepts for Release 1.7

F79926-03
December 2023

Oracle Cloud Native Environment Concepts for Release 1.7,

F79926-03

Copyright © 2022, 2023, Oracle and/or its affiliates.

Contents

Preface

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

Documentation License

The content in this document is licensed under the Creative Commons Attribution–Share Alike 4.0 (CC-BY-SA) license. In accordance with CC-BY-SA, if you distribute this content or an adaptation of it, you must provide attribution to Oracle and retain the original copyright notices.

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

Oracle Cloud Native Environment is a fully integrated suite for the development and management of cloud native applications.

Oracle Cloud Native Environment is a curated set of open source projects that are based on open standards, specifications, and APIs defined by the Open Container Initiative (OCI) and Cloud Native Computing Foundation (CNCF) that can be easily deployed, have been tested for interoperability and for which enterprise-grade support is offered. Oracle Cloud Native Environment delivers a simplified framework for installations, updates, upgrades, and configuration of key features for orchestrating microservices.

Oracle Cloud Native Environment uses Kubernetes to deploy and manage containers. When you create an environment, in addition to Kubernetes nodes, the Oracle Cloud Native Environment Platform API Server must be installed on a server, and is needed to perform a deployment and manage modules. The term *module* refers to a packaged software component that can be deployed to provide both core and optional cluster-wide functionality. The Kubernetes module for Oracle Cloud Native Environment is the core module, and automatically installs, and configures Kubernetes, CRI-O, runC, and Kata Containers on the Kubernetes nodes and brings up a Kubernetes cluster.

The Kubernetes nodes run an Oracle Cloud Native Environment Platform Agent. The Platform Agent communicates with the Platform API Server to manage the deployment of modules.

The Oracle Cloud Native Environment Platform CLI performs the validation and deployment of modules to the nodes, enabling easy deployment of modules such as the Kubernetes module. The required software for modules is configured by the Platform CLI, such as Kubernetes, CRI-O, runC, Kata Containers, CoreDNS, Flannel, and Calico. The Platform CLI also reports details about installed modules.

Optional modules can be installed into a Kubernetes cluster:

- The Calico module for Oracle Cloud Native Environment, which is used to set up Calico as the Kubernetes CNI for the pod data plane.
- The Multus module for Oracle Cloud Native Environment which is used to set up Multus, which is used to create a network bridge Calico or Flannel.
- The Oracle Cloud Infrastructure Cloud Controller Manager module for Oracle Cloud Native Environment, which is used to set up persistent storage and load balancers to provide external IP addresses for Kubernetes applications in a Kubernetes cluster running on Oracle Cloud Infrastructure instances.
- The MetalLB module for Oracle Cloud Native Environment, which is used to provide external IP addresses for Kubernetes applications running on bare metal hosts. MetalLB lets you use Kubernetes LoadBalancer services, which traditionally use a cloud provider network load balancer, in a bare metal environment.
- The Rook module for Oracle Cloud Native Environment, which is used to set up Ceph persistent storage for Kubernetes applications in a Kubernetes cluster.

- The KubeVirt module for Oracle Cloud Native Environment, which is used to set up KubeVirt to enable you to create virtual machines that are run and managed in a Kubernetes cluster.
- The Operator Lifecycle Manager module for Oracle Cloud Native Environment, which is used to deploy and manage Kubernetes operators in a Kubernetes cluster.
- The Istio module for Oracle Cloud Native Environment, which is used to deploy a service mesh on top of the Kubernetes cluster. The Istio module also installs the Prometheus module and the Grafana module.

2 Components

This chapter contains information on the components that are used to create Oracle Cloud Native Environment.

Container Runtimes

Containers are the fundamental infrastructure to deploy modern cloud applications. Oracle delivers the tools to create and provision Open Container Initiative (OCI)-compliant containers using CRI-O.

CRI-O, an implementation of the Kubernetes CRI (Container Runtime Interface) to enable using Open Container Initiative compatible runtimes, is included with Oracle Cloud Native Environment. CRI-O can run either runC or Kata Containers containers directly from Kubernetes, without any unnecessary code or tooling.

Container Orchestration

The version of Kubernetes used in Oracle Cloud Native Environment is based on the upstream Kubernetes project, and released under the CNCF Kubernetes Certified Conformance program. The Platform API Server simplifies the configuration and set up of the Kubernetes module to create a Kubernetes cluster and includes backup and recovery. The Kubernetes module is developed for Oracle Linux and integrates with CRI-O to provide a comprehensive container and orchestration environment for the delivery of microservices and next-generation application development.

Cloud Native Networking

The Container Network Interface (CNI) project, incubating under CNCF, seeks to simplify networking for container workloads by defining a common network interface for containers. The CNI plugin is included with Oracle Cloud Native Environment.

When you install Kubernetes, you can choose between Flannel or Calico as the CNI. You can also install Multus on top of Flannel or Calico to create a network bridge to pods.

Cloud Native Storage

Several storage projects are associated with the CNCF foundation, and the providers are included by default in Kubernetes. Storage integration is provided using plugins, referred to as the Container Storage Interface (CSI). The plugins adhere to a standard specification.

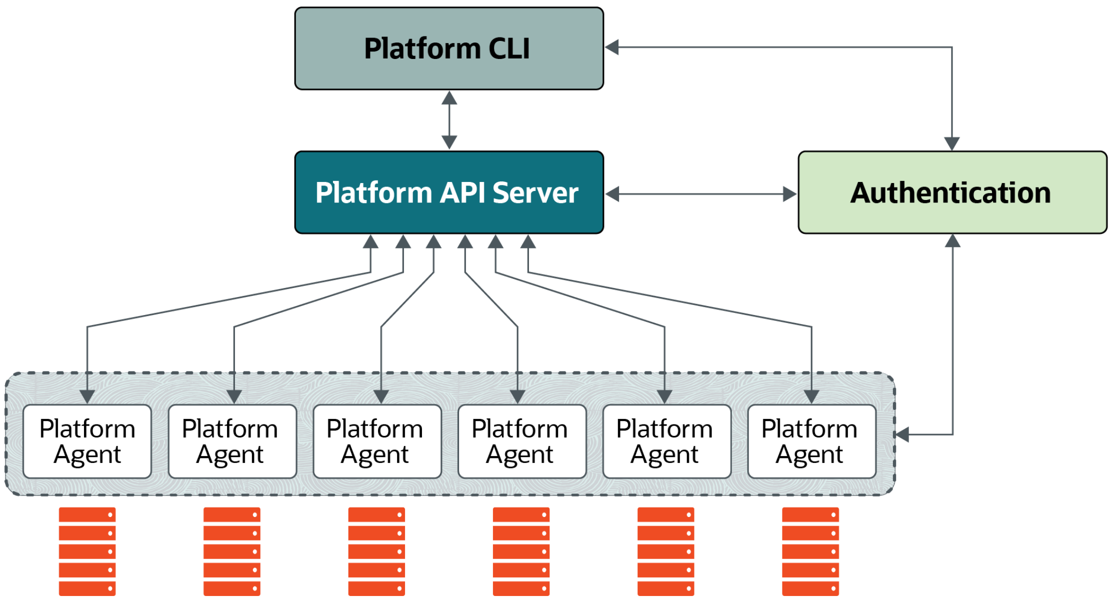
The Oracle Cloud Infrastructure Cloud Controller Manager module is used to set up dynamically provisioned persistent storage using Oracle Cloud Infrastructure.

The Rook module is used to set up dynamically provisioned persistent storage using Ceph.

3
Architecture

Oracle Cloud Native Environment is built from several discrete components. You interact with the environment directly using the Platform CLI. The Platform API Server interacts with the Platform Agent on each Kubernetes node. The Platform Agent is responsible for handling host-level operations on behalf of the Platform API Server.

Figure 3-1 Architecture



Platform API Server

The Platform API Server performs the business logic and manages all entities, from hosts to microservices. The Platform API Server is responsible for managing the state of the environment, including the deployment, and configuration of modules to one or more nodes in a cluster.

Platform Agent

The Platform Agent runs on each host to proxy requests from the Platform API Server to small worker applications. The primary reason for this is to ensure the Platform Agent process uses as little memory as possible. The Platform Agent refers to the union of the Platform Agent process and associated worker applications.

The Platform Agent knows how to gather the state of resources on its host and to change the state of those resources. The Platform Agent knows if a firewall port is open or closed, or if a package is installed and at which version. It also knows how to close that port if it's open, upgrade the package if it's old, or install the package if it's not installed.

Platform CLI

The Platform CLI is used to communicate with the Platform API Server. The Platform CLI is an application (the `olcnectl` command) that converts the input to Platform API Server calls. No business logic takes place in the Platform CLI. Parsing of the commands entered into the Platform CLI takes place in the Platform API Server.

The Platform CLI must be installed on an *operator* node.

Authentication

Standard X.509 certificates are used to establish node identity and authentication. The Platform API Server, Platform CLI, and the Platform Agent on Kubernetes nodes require a valid certificate chain for each component to mutually authenticate. Without these certificates, connections between the components and nodes are rejected.

X.509 certificates can be created and distributed manually, or using an authentication server such as Vault by HashiCorp.

4

Environments and Modules

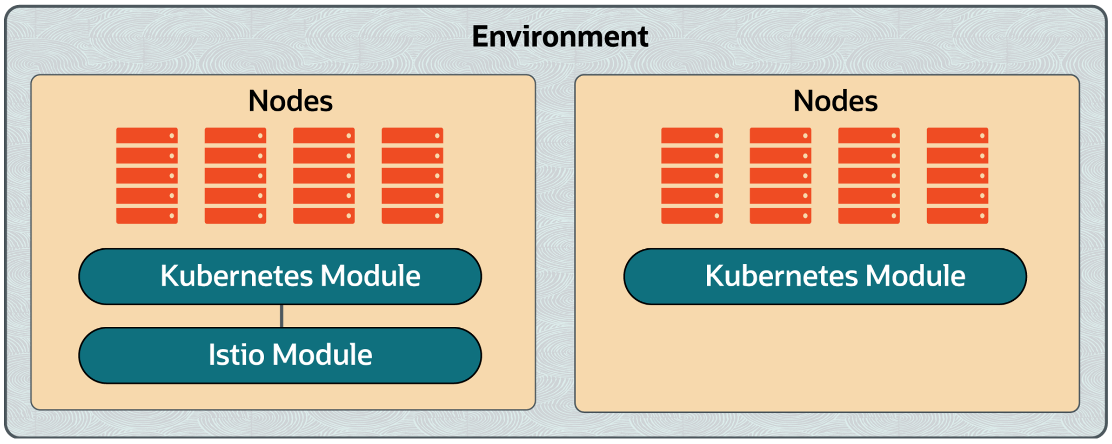
This chapter introduces the concepts of environments and modules in Oracle Cloud Native Environment.

Environments

An *environment* is a namespace that encapsulates the software installed and managed by Oracle Cloud Native Environment. Each environment contains at least the Kubernetes module.

The Platform CLI lets you create and manage many deployments. Each deployment contains an environment, and each environment might contain many modules. This lets you create many Kubernetes clusters using the same Oracle Cloud Native Environment installation. Each Kubernetes cluster must have dedicated nodes, a server can't be used in two clusters, or environments.

Figure 4-1 Environments



Modules

A *module* is a curated unit of software that can be installed and managed by Oracle Cloud Native Environment. A module fulfills at least one specific role in a deployment. Modules that fulfill the same roles can be swapped out in a managed way. Modules might encapsulate other modules.

The available modules are:

- Kubernetes module
- Calico module
- Multus module

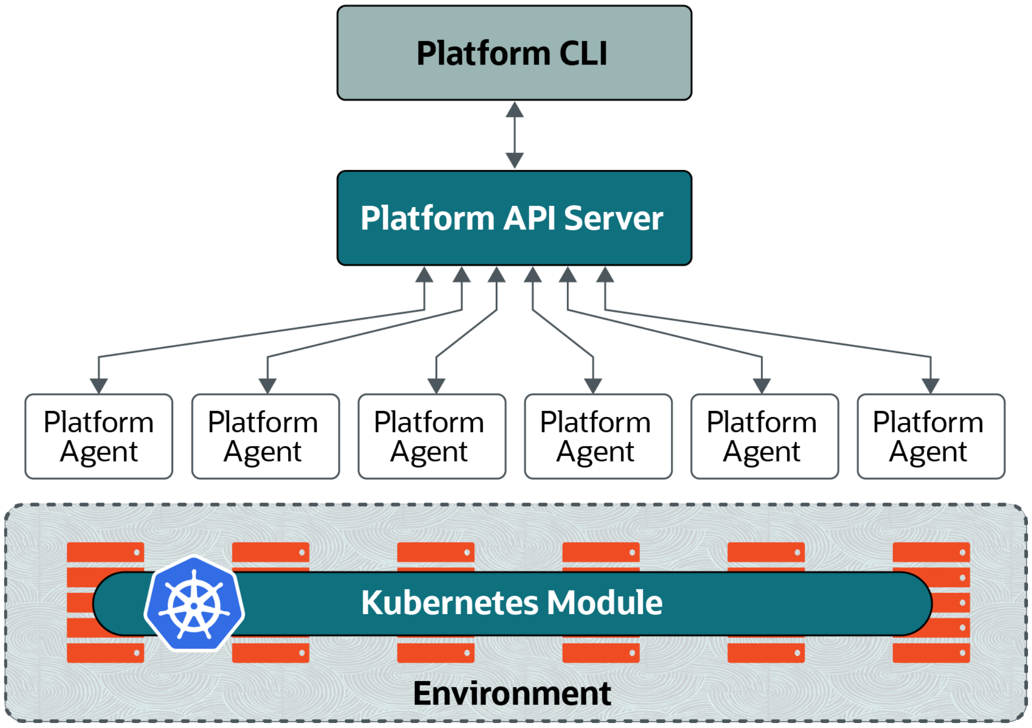
- Oracle Cloud Infrastructure Cloud Controller Manager module
- MetalLB module
- Rook module
- KubeVirt module
- Operator Lifecycle Manager module
- Istio module
- Prometheus module
- Grafana module

Helm is used by the Platform API Server to install optional modules. Helm is a package manager for Kubernetes. Helm simplifies the task of deploying and managing software inside Kubernetes clusters. Helm uses *charts* to manage the packages that it can deploy. A chart is a collection of files that describe a related set of Kubernetes resources.

Kubernetes Module

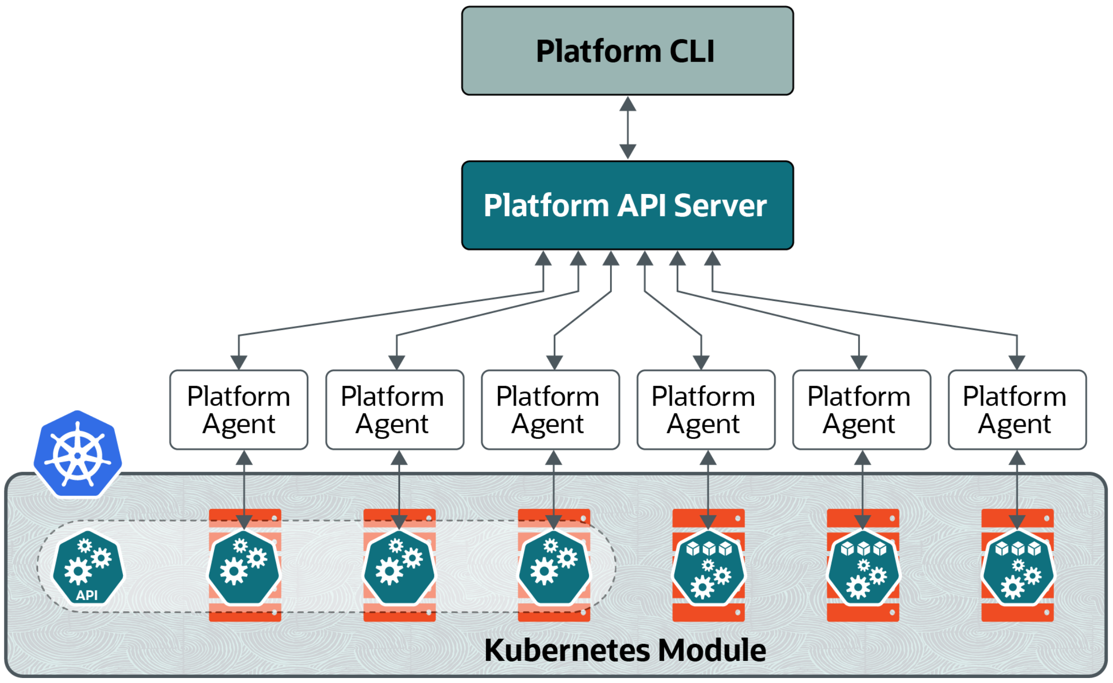
The core module in Oracle Cloud Native Environment is the Kubernetes module. The Kubernetes module is used to deploy a Kubernetes cluster in an environment.

Figure 4-2 Kubernetes module



The Kubernetes module installs and configures Kubernetes on the nodes and sets up the cluster.

Figure 4-3 Kubernetes cluster



The Kubernetes module includes:

- **Flannel**: The default overlay network for a Kubernetes cluster.
- **CoreDNS**: The DNS server for a Kubernetes cluster.
- **CRI-O**: Manages the container runtime for a Kubernetes cluster.
- **runC**: The default lightweight, portable container runtime for a Kubernetes cluster.
- **Kata Containers**: An optional lightweight virtual machine runtime for a Kubernetes cluster.

For more information about installing and using the Kubernetes module, see Kubernetes Module.

Calico Module

The Calico module installs Calico into a Kubernetes cluster. This lets you use Calico as the CNI for the Kubernetes data plane.

For information about installing and using the Calico module, see Calico Module.

Multus Module

The Multus module installs Multus into a Kubernetes cluster. This lets you use Multus to create a network bridge to pods. Multus can be used with either Calico or Flannel as the CNI for the Kubernetes data plane.

For information about installing and using the Multus module, see Multus Module.

Oracle Cloud Infrastructure Cloud Controller Manager Module

The Oracle Cloud Infrastructure Cloud Controller Manager module is used to set up dynamically provisioned persistent storage and application load balancers using Oracle Cloud Infrastructure.

Oracle Cloud Infrastructure block volumes and file storage provide reliable, high-performance storage designed to work with a range of Oracle Cloud Infrastructure virtual machines and bare metal instances. With built-in redundancy, storage is persistent, and durable beyond the lifespan of a virtual machine. The Oracle Cloud Infrastructure Cloud Controller Manager module creates a Kubernetes StorageClass provisioner to access Oracle Cloud Infrastructure storage.

The Oracle Cloud Infrastructure Flexible Network Load Balancing service (Oracle Cloud Infrastructure load balancer) provides automated traffic distribution from one entry point to many backend servers in a Virtual Cloud Network (VCN). It operates at the connection level and load balances incoming client connections to healthy backend servers based on Layer 3/Layer 4 (IP protocol) data.

The Oracle Cloud Infrastructure load balancer provides network load balancers for Kubernetes applications running on Oracle Cloud Infrastructure.

For information about installing and using the Oracle Cloud Infrastructure Cloud Controller Manager module, see Oracle Cloud Infrastructure Cloud Controller Manager Module.

MetalLB Module

MetalLB is a network load balancer for Kubernetes applications running on bare metal hosts. MetalLB lets you use Kubernetes LoadBalancer services, which traditionally use a cloud provider network load balancer, in a bare metal environment.

The MetalLB module is used to set up network load balancers for Kubernetes applications using MetalLB.

For information about installing and using the MetalLB module, see MetalLB Module.

Rook Module

The Rook module is used to set up dynamically provisioned persistent storage using Ceph. Rook is a container storage platform built on Ceph. Rook is deployed as a Kubernetes operator inside a Kubernetes cluster and automates the work required to provision Ceph-backed persistent storage using the Kubernetes Container Storage Interface.

For information about installing and using the Rook module, see Rook Module.

KubeVirt Module

The KubeVirt module installs KubeVirt. KubeVirt is a virtualization technology that can create and manage virtual machines in a Kubernetes cluster. KubeVirt leverages the benefits of Kubernetes orchestration and management to virtual machines, enabling you to run virtual machines and containers in a unified infrastructure. It simplifies the management of mixed workloads, provides better resource usage, and enhances the overall flexibility and scalability of Kubernetes clusters.

For information about installing and using the KubeVirt module, see KubeVirt Module.

Operator Lifecycle Manager Module

The Operator Lifecycle Manager module manages the installation and lifecycle management of Kubernetes operators in a Kubernetes cluster.

A Kubernetes operator is a design pattern that lets you write code to automate tasks and extend Kubernetes. An operator is a set of concepts you can use to define a service for Kubernetes and helps to automate administrative services in Kubernetes.

For information about installing and using the Operator Lifecycle Manager module, see Operator Lifecycle Manager Module.

Istio Module

Istio is a fully featured service mesh for microservices in Kubernetes clusters. Istio can handle most aspects of microservice management, for example, identity, authentication, transport security, metric scraping, and so on.

The Istio module for Oracle Cloud Native Environment installs Istio into a Kubernetes module (cluster).

The Istio module installs components that are used solely by Istio:

- Egress gateway
- Ingress gateway
- Istiod
- Prometheus (installed by the Prometheus module)
- Grafana (installed by the Grafana module)

When you deploy the Istio module, Prometheus is also deployed as a supporting module. Prometheus is used to monitor and gather metrics about the Kubernetes cluster. Another supporting module that's deployed with Istio is Grafana. Grafana is a monitoring and visualization tool for time-series data stored in a database, which in this case, is Prometheus. Grafana lets you to visually query and monitor the network traffic and services in a Kubernetes cluster. Grafana includes browser-based dashboards to visualize the cluster metrics gathered from Prometheus.

For information about installing and using the Istio module, see Istio Module.

Prometheus Module

Prometheus is a systems monitoring and alerting toolkit that collects and stores metrics and other time series data from various sources and presents it in an easily retrievable manner.

The Prometheus module for Oracle Cloud Native Environment is configured with rich monitoring of important systems inside a Kubernetes cluster.

The Prometheus module is required by the Istio module and is used to create an embedded instance of Prometheus for use by Istio.

Note:

In this release, the Prometheus module is only used in the context of an Istio module deployment.

Grafana Module

Grafana is a monitoring and visualization tool that lets you query the time-series data in Prometheus and create dashboards to visualize that data. You can visually monitor a Kubernetes cluster, the services that are running, and network traffic.

The Grafana module for Oracle Cloud Native Environment is configured to connect to and read data from Prometheus.

The Grafana module is required by the Istio module and is used to create an instance of Grafana for use by Istio.

Note:

In this release, the Grafana module is only used in the context of an Istio module deployment.

5

Network Planes

This chapter contains information about the Oracle Cloud Native Environment management, control, and data planes.

Management Plane

The management plane consists of the components that make up the Oracle Cloud Native Environment platform: the Platform API Server, the Platform Agent, and the Platform CLI.

Communication between the components is secured using Transport Layer Security (TLS). You can configure the cipher suites to use for TLS for the management plane.

You can set up the X.509 certificates used for TLS before you create environment, or have a certificate management application, such as Vault, manage these for you.

Control Plane

The control plane contains the Kubernetes components and any load balancer.

Kubernetes has a sophisticated networking model with many options that lets users finely tune the networking configuration. Oracle Cloud Native Environment simplifies the Kubernetes networking by setting network defaults that align with community best practices.

By default, all Kubernetes services are bound to the network interface that handles the default route for the system. The default route is set to the network interface used by the Platform Agent, and is used for both the Kubernetes control plane and the data plane.

Two motivations are behind this choice. The first is that the Platform API Server always must always communicate with the Kubernetes API server. By making sure the Kubernetes API server is bound to the same interface as the Platform Agent, this condition is always met. Also, if nodes have many network interfaces, the sensitive networks aren't the networks that Oracle Cloud Native Environment uses to communicate.

When deploying a highly available cluster having many control plane nodes with an internal load balancer, the Platform API Server uses the same network interface as was set to host the Kubernetes control plane services to host the virtual IP address.

Data Plane

The data plane is the network used by the pods running on Kubernetes.

The same algorithm for to decide the default control plane interface is used when instantiating the Kubernetes pod network. The network interface used by the Platform Agent is used for both the Kubernetes control plane and the data plane. In environments with many networks, this might not be the best choice. Oracle Cloud Native Environment lets you customize the network interface used for pod networking when you create the Kubernetes module. When the CNI is brought up, it uses the network interface you specify for the pod network.

When you install Kubernetes, you can choose between Flannel or Calico as the CNI. You can also install Multus on top of Flannel or Calico to create a network bridge to pods.

6

Storage

Every meaningful workload in the computing industry requires some sort of data storage. Persistent storage is essential when working with stateful applications such as databases, as it's important that you can retain data beyond the lifecycle of the container, or even of the pod itself.

Persistent storage in Kubernetes is handled in the form of PersistentVolume objects and are bound to pods using a PersistentVolumeClaim. You can host a PersistentVolume locally or on networked storage devices or services.

A typical Kubernetes environment involves many hosts and includes some type of networked storage. Using networked storage helps to ensure resilience and lets you take full advantage of a clustered environment. In the case where the node where a pod is running fails, a new pod can be started on another node and storage access can be resumed. This is important for database environments where replica setup has been configured.

Persistent Storage

Persistent storage is provided in Kubernetes using the PersistentVolume subsystem. To configure persistent storage, you must be familiar with the following terms:

- **PersistentVolume**

  A PersistentVolume defines the type of storage that's being used and the method used to connect to it. This is the real disk or networked storage service that's used to store data.

- **PersistentVolumeClaim**

  A PersistentVolumeClaim defines the parameters that a consumer, such as a pod, uses to bind the PersistentVolume. The claim might specify quota and access modes to be applied to the resource for a consumer. A pod can use a PersistentVolumeClaim to gain access to the volume and mount it.

- **StorageClass**

  A StorageClass is an object that specifies a volume plugin, known as a provisioner, that lets users to define PersistentVolumeClaims without needing to preconfigure the storage for a PersistentVolume. This can be used to provide access to similar volume types as a pooled resource that can be dynamically provisioned for the lifecycle of a PersistentVolumeClaim.

PersistentVolumes can be provisioned either statically or dynamically.

Static PersistentVolumes are manually created and contain the details required to access real storage and can be consumed directly by any pod that has an associated PersistentVolumeClaim.

Dynamic PersistentVolumes can be automatically generated if a PersistentVolumeClaim doesn't match an existing static PersistentVolume and an existing StorageClass is requested in the claim. A StorageClass can be defined to host a pool of storage that can be accessed dynamically. Creating a StorageClass is an optional step that's only required if you intend to use dynamic provisioning.

The process to provision persistent storage is as follows:

1. Create a PersistentVolume or StorageClass.
2. Create PersistentVolumeClaims.
3. Configure a pod to use the PersistentVolumeClaim.

The process for adding and configuring NFS and iSCSI volumes is described in detail in the upstream documentation.

Container Storage Interface Plugins

The Container Storage Interface (CSI) is an Open Container Initiative standard for controlling storage workloads from container engines. Kubernetes implements this interface to provide automated control for storage workloads inside Kubernetes clusters. For a list of the Kubernetes storage provisioners, see the upstream documentation.

You can install CSI plugins into a Kubernetes cluster in Oracle Cloud Native Environment. To make it easier to perform the CSI plugin installation, Oracle provides several storage related modules.

The Oracle Cloud Infrastructure Cloud Controller Manager module for Oracle Cloud Native Environment can be used to set up the CSI plugin for Oracle Cloud Infrastructure.

The Rook module for Oracle Cloud Native Environment can be used to set up the CSI plugin for Ceph.

7

Highly Available Clusters

This chapter contains high level information about the types of highly available (HA) Kubernetes clusters you can deploy using Oracle Cloud Native Environment.

Kubernetes can be deployed with more than one replica of the control plane node. Automated failover to those replicas provides a more scalable and resilient service. This type of cluster deployment is referred to in this document as an HA cluster.

Important:

To create an HA cluster you need at least three control plane nodes and two worker nodes.

Creating an HA cluster with three control plane nodes ensures replication of configuration data between them through the distributed key store, `etcd`, so an HA cluster is resilient to a single control plane node failing without any loss of data or uptime. If more than one control plane node fails, you can restore the control plane nodes in the cluster from a backup file to avoid data loss.

Oracle Cloud Native Environment implements the Kubernetes stacked `etcd` topology, where `etcd` runs on the control plane nodes. For more information on this topology, see the upstream documentation.

Load Balancer

An HA cluster needs a load balancer to provide high availability of the control plane nodes. A load balancer communicates with the Kubernetes API Server to maintain high availability of the control plane nodes.

You can use an external load balancer instance, a load balancer provided by a cloud infrastructure, or have the Platform CLI install a load balancer on the control plane nodes.

For more information about the configuration requirements of the load balancer, see Getting Started.

Highly Available Cluster with External Load Balancer

When you set up an HA cluster to use an external load balancer, the load balancer is used to manage the resource availability and efficiency of control plane nodes to ensure instances of the Kubernetes API Server on control plane nodes can fail without impacting cluster availability.

To use an external load balancer, it must be set up and ready to use before you perform an HA cluster deployment. The load balancer hostname and port is entered as an option when you create the Kubernetes module.

A load balancer provided by a cloud infrastructure can also be used, for example, an Oracle Cloud Infrastructure load balancer. This must also be set up and ready to use before you create the Kubernetes module.

Highly Available Cluster with Internal Load Balancer

When you set up an HA cluster to use an internal load balancer, the Platform CLI installs NGINX and Keepalived on the control plane nodes to enable the container-based deployment of the load balancer. The internal load balancer configures the native active-active high availability solution for the Kubernetes API server.

NGINX improves the resource availability and efficiency of control plane nodes to ensure instances of the Kubernetes API server on control plane nodes can fail without impacting cluster availability.

If you use the internal load balancer, you must set aside an IP address on the control plane network to use as a virtual IP address. The Keepalived instance makes sure the virtual IP address is always reachable by monitoring the health of other control plane nodes and appropriating the IP address if a node fails. Keepalived is used to failover automatically to a standby control plane node if problems occur.

As part of deploying the internal load balancer, the `olcne-nginx` and `keepalived` services are enabled and started on the control plane nodes.

8

Terminology

Environment

A namespace that encapsulates the software installed and managed by Oracle Cloud Native Environment. Each environment contains at least the Kubernetes module.

Module

A module represents some installable object in an environment. The most common module is the Kubernetes module, as it's required for most other modules. A module might encapsulate other modules.

Node

A host system that's a member of an Oracle Cloud Native Environment, including hosts used as Kubernetes control plane and worker nodes, and hosts that run the Platform API Server and Platform CLI.

Kubernetes Node

A host in an Oracle Cloud Native Environment that contains the Platform Agent and other required software to run as a Kubernetes cluster member, either as a control plane or worker node.

Operator Node

A host in an Oracle Cloud Native Environment that contains the Platform CLI. This node might also be a Platform API Server node.

Platform Agent

The Oracle Cloud Native Environment Platform Agent is a software agent installed on all nodes which is used by the Platform API Server to report and change state as directed by the Platform API Server.

Platform API Server

The Oracle Cloud Native Environment Platform API Server manages the state of one or more environments, including the deployment, and configuration of modules to one or more nodes in a cluster.

Platform API Server Node

A host in an Oracle Cloud Native Environment that contains the Platform API Server. This node might also be an operator node.

Platform CLI

The Oracle Cloud Native Environment Platform CLI used to create and manage deployments. The `olcnectl` command.