Oracle Private Cloud Appliance

Container Engine for Kubernetes

F81950-02
April 2024

Oracle Private Cloud Appliance Container Engine for Kubernetes,

F81950-02

Contents

Preface

This publication is part of the customer documentation set for Oracle Private Cloud Appliance Release 3.0. Note that the documentation follows the release numbering scheme of the appliance software, not the hardware on which it is installed. All Oracle Private Cloud Appliance product documentation is available at https://docs.oracle.com/en/engineered-systems/private-cloud-appliance/index.html.

Oracle Private Cloud Appliance Release 3.x is a flexible general purpose Infrastructure as a Service solution, engineered for optimal performance and compatibility with Oracle Cloud Infrastructure. It allows customers to consume the core cloud services from the safety of their own network, behind their own firewall.

Audience

This documentation is intended for owners, administrators and operators of Oracle Private Cloud Appliance. It provides architectural and technical background information about the engineered system components and services, as well as instructions for installation, administration, monitoring and usage.

Oracle Private Cloud Appliance has two strictly separated operating areas, known as enclaves. The Compute Enclave offers a practically identical experience to Oracle Cloud Infrastructure: It allows users to build, configure and manage cloud workloads using compute instances and their associated cloud resources. The Service Enclave is where privileged administrators configure and manage the appliance infrastructure that provides the foundation for the cloud environment. The target audiences of these enclaves are distinct groups of users and administrators. Each enclave also provides its own separate interfaces.

It is assumed that readers have experience with system administration, network and storage configuration, and are familiar with virtualization technologies. Depending on the types of workloads deployed on the system, it is advisable to have a general understanding of container orchestration, and UNIX and Microsoft Windows operating systems.

Feedback

Provide feedback about this documentation at https://www.oracle.com/goto/docfeedback.

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.

Convention	Meaning
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, code in examples, text that appears on the screen, or text that you enter.
`$` prompt	The dollar sign (`$`) prompt indicates a command run as a non-root user.
`#` prompt	The pound sign (`#`) prompt indicates a command run as the `root` user.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

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

Overview of Container Engine for Kubernetes

Oracle Private Cloud Appliance Container Engine for Kubernetes (OKE) is a scalable, highly available service that can be used to deploy any containerized application to the cloud.

The OKE service uses Cluster API Provider (CAPI) and Cluster API Provider for Oracle Cloud Infrastructure (CAPOCI) to orchestrate the cluster on the Private Cloud Appliance.

The OKE service uses Kubernetes, the open-source system for automating deployment, scaling, and management of containerized applications across clusters of hosts. Kubernetes groups the containers that make up an application into logical units called pods for easy management.

For more information about Kubernetes in Oracle, see What Is Kubernetes? For more general information about Kubernetes, see the Kubernetes site.

Using the OKE Service

You can access the OKE service to create OKE clusters by using the Compute Web UI, the OCI CLI, and API. For general information about using the Private Cloud Appliance Compute Web UI and OCI CLI, see the Working in the Compute Enclave chapter in the *Oracle Private Cloud Appliance User Guide*.

You can access OKE clusters by using the Kubernetes command line (`kubectl`), the Kubernetes Dashboard, and the Kubernetes API.

On Private Cloud Appliance, the OKE service manages all OKE cluster nodes, which are compute instances. An authorized user can perform tasks such as patch the instance.

Supported Versions of Kubernetes

The OKE service uses versions of Kubernetes that are certified as conformant by the Cloud Native Computing Foundation (CNCF). The OKE service is itself ISO-compliant (ISO-IEC 27001, 27017, 27018).

Supported versions of Kubernetes are 1.28.*x*, 1.27.*x*, and 1.26.*x*.

OKE Service Limits

The following table shows the service limits for the OKE service on Private Cloud Appliance.

Service	Limit
Maximum number of clusters per tenancy	10
Maximum number of worker nodes (compute instances) per cluster. These nodes can be distributed across multiple node pools.	128
Maximum number of nodes per node pool/ group	128
Maximum number of node pools/groups per cluster	No limit on number of node pools as long as total nodes per cluster does not exceed 128.

Service	Limit
Maximum number of pods per node	110. This is the Kubernetes default.

2

OKE Workflow

Most steps to configure and use the OKE service are performed by regular users in the Compute Enclave. Some steps need to be performed by a Compute Enclave user with more administrative authorizations, and some steps can only be performed by a Service Enclave administrator.

- Administrator Tasks
- User Tasks

Administrator Tasks

If you enable the appliance administration network, verify that the administration network and the data center network are configured to allow traffic to and from the cluster control plane. See the following resources:

- Editing Administration Network Information in the *Oracle Private Cloud Appliance Administrator Guide*
- Administration Network Configuration Notes in the *Oracle Private Cloud Appliance Installation Guide*
- Access Configuration With Administration Network in the *Oracle Private Cloud Appliance Security Guide*

Create the following resources in the Private Cloud Appliance Compute Enclave:

- Platform images. Platform images include images required by OKE that have Kubernetes installed on them. Platform images should be imported to all tenancies in the Compute Enclave during appliance installation, upgrade, or patching. If this was not done, a Service Enclave administrator must import images as described in Providing Platform Images in the *Oracle Private Cloud Appliance Administrator Guide*.
- A users group that has a policy that authorizes members to use OKE. See Creating and Managing User Groups in the *Oracle Private Cloud Appliance User Guide* to create a group or update an existing group. Include the `manage cluster-family` authorization in the policy. The following is an example policy for the OKE user group. Depending on your organization, for example if you have a separate team who manage network resources, some of the following "manage" authorizations could be "read" or "use" authorizations, or you might need to add authorizations. You might need to create more than one user group to authorize OKE work in different compartments.

  ```
  allow group group-name to read all-resources in tenancy
  allow group group-name to manage cluster-family in compartment compartment-name
  allow group group-name to manage instance-family in compartment compartment-name
  allow group group-name to manage virtual-network-family in compartment compartment-
  name
  allow group group-name to manage volume-family in compartment compartment-name
  ```

- The OraclePCA-OKE/cluster_id defined tag.

  This tag is required to create or update an OKE cluster or node pool. This tag also is used to identify instances that need to be in a dynamic group.

1. Create the OraclePCA-OKE tag namespace.

   In the Resource Tag Management chapter of the *Oracle Private Cloud Appliance User Guide*, follow the procedure in "Creating a Tag Namespace."

   > **Important:**
   >
   > Create the OraclePCA-OKE tag namespace in the tenancy (root) compartment, not in a child compartment of the tenancy.

   In the Create Namespace Definition dialog:

   – Enter "OraclePCA-OKE" for the Name.

   – Enter a description for the tag namespace.

   – Click the Create Namespace Definition button.

     The details page for the OraclePCA-OKE tag namespace is shown.

2. Create the cluster_id tag key definition in the OraclePCA-OKE tag namespace.

   On the details page for the OraclePCA-OKE tag namespace, click the Create Tag Key Definition button above the list of tag key definitions.

   In the Create Tag Key Definition dialog:

   – Enter "cluster_id" for the Name.

   – Enter a description for the tag key.

   – Ensure that Static Value is selected for the Tag Value Type.

   – Click the Create Tag Key Definition button.

> **Important:**
>
> The tag namespace name must be exactly `OraclePCA-OKE`, and the tag key name must be exactly `cluster_id`.

When you create a node pool, or update the node pool to add nodes, this tag is applied to every node to identify instances that need to be members of the dynamic group.

- A dynamic group to authorize member instances to manage OKE resources. See Creating and Managing Dynamic Groups in the *Oracle Private Cloud Appliance User Guide*.

  Enter the following matching rule to define the group:

  ```
  tag.OraclePCA-OKE.cluster_id.value
  ```

  All nodes that have this tag are members of the dynamic group.

- A policy for the dynamic group. See Managing Policies in the *Oracle Private Cloud Appliance User Guide*.

  Specify the following rules for the policy:

```
allow dynamic-group dynamic-group-name to use instance-family in tenancy
allow dynamic-group dynamic-group-name to use virtual-network-family in tenancy
allow dynamic-group dynamic-group-name to manage load-balancers in tenancy
allow dynamic-group dynamic-group-name to manage volume-family in tenancy
Allow dynamic-group dynamic-group-name to manage file-family in tenancy
```

After upgrade, patching, or any other outage, or if the automated Certificate Authority bundle update fails, you might want to update the CA bundle manually on the management node. See Updating the Certificate Authority Bundle.

Updating the Certificate Authority Bundle

The Certificate Authority (CA) bundle for this Private Cloud Appliance is downloaded and made available to a cluster when the cluster is created. The CA bundle includes the certificate, private and public keys, and other authorization information.

The CA bundle is automatically updated on the appliance when regular certificate rotation occurs or when the appliance is upgraded, for example.

When the CA bundle is updated on the appliance, then it must be updated on the local system, for example to enable use of `cluster-api`. This is similar to replacing the CA bundle in your `~/.oci` configuration so that you can run OCI CLI commands.

A process runs every hour to check the validity of the CA bundle and updates the CA bundle if necessary.

If you need to update the CA bundle between these hourly checks, the process can be run manually:

1. Log onto the management node of the Private Cloud Appliance as a system administrator with root privilege.
2. Get the name of an OKE pod.

   The following command lists the three OKE pods in the `oke` namespace:

   ```
   # kubectl get pod -n oke -l app=oke
   ```
3. Run the command to update the CA bundle.

   Use one of the `oke-`***`uniqueID`*** pod names from the preceding step.

   ```
   # kubectl exec -it oke-6c4d85d6f-72fxs -n oke -c oke -- /usr/bin/pca-oke-cluster-
   tool
   ```

You can check Loki logs in Grafana for any errors that might have occurred when this process ran either automatically or manually. See "Accessing System Logs" in the Status and Health Monitoring chapter of the *Oracle Private Cloud Appliance Administrator Guide*.

User Tasks

Perform the following tasks on your local system:

1. Configure OCI CLI access. See Using the OCI CLI in the *Oracle Private Cloud Appliance User Guide*. If you already have OCI CLIinstalled, use `oci -v` to check the version. The minimum required version for using OKE is 3.15.1.
2. Install the Kubernetes client command line tool, `kubectl`. See Install kubectl. If you already have `kubectl` installed, ensure the version is within one minor version of the Kubernetes version that you are using. See Supported Versions of Kubernetes.

Perform the following tasks in the Compute Enclave or on your local system:

1. Create network resources: VCN, subnets, internet gateway, NAT gateway, route tables, and security lists. See Creating Network Resources.
2. Create an OKE cluster. See Creating an OKE Cluster.
3. Create a Kubernetes configuration file for the cluster. See Creating a Kubernetes Configuration File.
4. Create a Kubernetes Dashboard to manage the cluster and to manage and troubleshoot applications running in the cluster. On the https://kubernetes.io/ site, see Deploy and Access the Kubernetes Dashboard.
5. Create a worker node pool. See Creating an OKE Worker Node Pool.
6. Configure any registries or repositories that the worker nodes need.
7. Configure any proxies that are needed on your worker nodes. See Configuring a Proxy.
8. Create a service to expose containerized applications outside the Private Cloud Appliance. See Exposing Containerized Applications.
9. Create persistent storage for applications to use. See Adding Storage for Containerized Applications.

3

Creating Network Resources

The resource definitions in the following sections in this chapter create a working example set of network resources for workload clusters. Use this configuration as a guide when you create these resources. You can change the values of properties such as CIDR blocks and IP addresses. You should not change the values of properties such as the network protocol, the stateful setting, or the private/public setting. See Workload Cluster Network Ports for specific ports that must be open for specific purposes.

If your network requires proxy settings to enable worker nodes to reach outside registries or repositories, for example, see Configuring a Proxy.

Note:

If the appliance administration network is enabled, ask your system administrator to verify that the administration network and the data center network are configured to allow traffic to and from the cluster control plane. See Administration Network Configuration Notes in the *Oracle Private Cloud Appliance Installation Guide*.

Create the following network resources. To use Terraform, see Example Terraform Scripts for Network Resources.

Note:

Create all of these network resources in the same compartment on the appliance.

- VCN. See Creating an OKE VCN.
- Internet gateway
- NAT gateway
- Route rules
- Security lists
- The following four subnets:
  - Worker. See Creating an OKE Worker Subnet.
  - Worker load balancer. See Creating an OKE Worker Load Balancer Subnet.
  - Control plane. See Creating an OKE Control Plane Subnet.
  - Control plane load balancer. See Creating an OKE Control Plane Load Balancer Subnet.

Workload Cluster Network Ports

The following table lists ports that are used by workload clusters. These ports must be available to configure workload cluster networking. You might need to open additional ports for other purposes.

All protocols are TCP. All port states are Stateful. Port 6443 is the port used for Kubernetes API and is also known as ***kubernetes_api_port*** in this guide.

See also the tables in Port Matrix in the *Oracle Private Cloud Appliance Security Guide*. The first table is for environments where the administration network is not enabled. The second table, Access Configuration With Administration Network, is for environments where the administration network is enabled.

Source IP Address	Destination IP Address	Port	Description
bastion host: ***vcn_cidr***	Worker nodes subnet: ***worker_cidr***	22	Outbound connections from the bastion host to the worker CIDR.
bastion host: ***vcn_cidr***	Control plane subnet: ***kmi_cidr***	22	Outbound connections from the bastion host to the control plane nodes.
Worker nodes subnet: ***worker_cidr***	yum repository	80	Outbound connections from the worker CIDR to external applications.
Worker nodes subnet: ***worker_cidr***	Secure yum repository	443	Secure outbound traffic from the worker CIDR to external applications.
Worker nodes subnet: ***worker_cidr***	Container registry	5000	Outbound connections from the worker CIDR to the container registry.
Worker nodes subnet: ***worker_cidr***	Control plane subnet: ***kmi_cidr***	6443	Outbound connections from the worker CIDR to the Kubernetes API. This is necessary to allow nodes to join through either a public IP address on one of the nodes or the load balancer public IP address.
Worker nodes subnet: ***worker_cidr***	Control plane load balancer	6443	Inbound connections from the worker CIDR to the Kubernetes API.
CIDR for clients: ***kube_client_cidr***	Control plane load balancer	6443	Inbound connections from clients to the Kubernetes API server.
Worker nodes subnet: ***worker_cidr***	Control plane subnet: ***kmi_cidr***	6443	Private outbound connections from the worker CIDR to `kubeapi` on the control plane subnet.
kube_client_cidr	Worker nodes subnet: ***worker_cidr***	30000-32767	Inbound traffic for applications from Kubernetes clients.

Workload Cluster Network CIDR Ranges

Throughout this documentation, variables are used to represent CIDR ranges for instances in different subnets. The following table lists the CIDR variables and example values. Change these example values as necessary for your environment. The IP Subnet Calculator on Calculator.net is one tool for finding all available networks for a given IP address and prefix length.

Variable Name	Description	Example Value
vcn_cidr	VCN CIDR range	172.31.252.0/23
worker_cidr	Worker subnet CIDR	172.31.253.0/24
workerlb_cidr	Worker load balancer subnet CIDR	172.31.252.0/25
kmi_cidr	OKE control plane subnet CIDR	172.31.252.224/28
kmilb_cidr	OKE control plane load balancer subnet CIDR	172.31.252.240/28
kube_client_cidr	CIDR for clients that are allowed to contact the Kubernetes API server	10.0.0.0/8

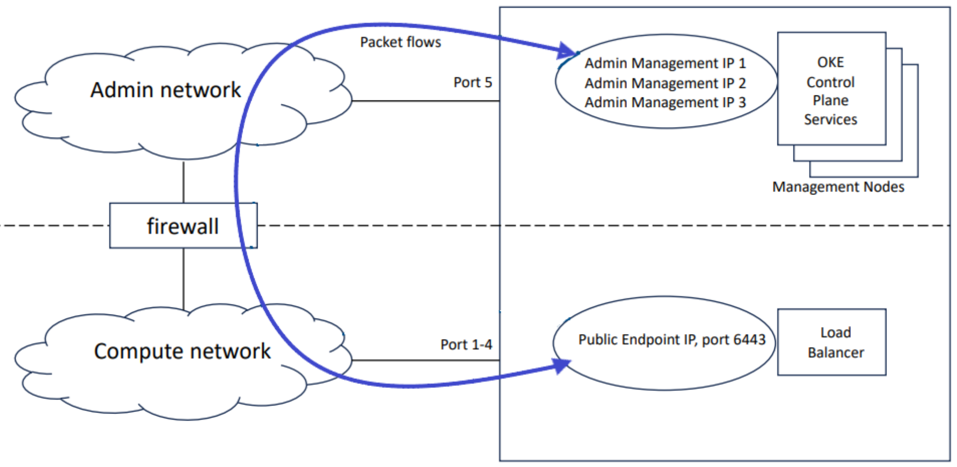
OKE Cluster Management with Administration Network

When OKE is used on a system configured with a separate administration network, the data center firewall must be configured to allow traffic between the OKE control plane and the OKE clusters deployed by Compute Enclave users.

The OKE control plane runs on the management nodes in the administration network, while the OKE clusters are deployed in the data network. The management interface of an OKE cluster is port 6443 on its load balancer public IP address. This address is assigned from the data center IP range you reserved and configured as public IPs during initial appliance setup.

Because of the network segregation, traffic from the OKE control plane must exit the appliance through the administration network, and reenter through the data network to reach the OKE cluster. The data center network infrastructure must allow traffic in both directions. Without the necessary firewall and routing rules, users cannot deploy OKE clusters.

Figure 3-1 Example of System Configured with a Separate Administration Network



Example Terraform Scripts for Network Resources

The following Terraform scripts create the network resources that are required by OKE. Subsequent topics in this chapter show other ways to define these same network resources.

Most of the values shown in these scripts, such as resource display names and CIDRs, are examples. Some ports must be specified as shown (see Workload Cluster Network Ports), and the OKE control plane subnet must be named `control-plane`. See Workload Cluster Network CIDR Ranges for comments about CIDR values.

- variables.tf
- terraform.tfvars
- provider.tf
- main.tf
- oke_vcn.tf
- oke_worker_seclist.tf
- oke_worker_subnet.tf
- oke_kmi_seclist.tf
- oke_kmi_subnet.tf

variables.tf

This file creates several variables that are used to configure OKE network resources. Many of these variables are not assigned values in this file. One port and five CIDRs are assigned values. The `kubernetes_api_port`, port 6443, is the port used to access the Kubernetes API. See also Workload Cluster Network Ports. The five CIDRs that are defined in this file are for the OKE VCN, worker subnet, worker load balancer subnet, control plane subnet, and control plane load balancer subnet.

```
variable "oci_config_file_profile" {
  type    = string
  default = "DEFAULT"
}

variable "tenancy_ocid" {
  description = "tenancy OCID"
  type        = string
  nullable    = false
}

variable "compartment_id" {
  description = "compartment OCID"
  type        = string
  nullable    = false
}

variable "vcn_name" {
  description = "VCN name"
  nullable    = false
}

variable "kube_client_cidr" {
  description = "CIDR of Kubernetes API clients"
  type        = string
  nullable    = false
}

variable "kubernetes_api_port" {
  description = "port used for kubernetes API"
  type        = string
  default     = "6443"
}

variable "worker_lb_ingress_rules" {
  description = "traffic allowed to worker load balancer"
  type = list(object({
    source   = string
    port_min = string
    port_max = string
  }))
  nullable = false
}

variable "worker_ingress_rules" {
  description = "traffic allowed directly to workers"
  type = list(object({
    source   = string
    port_min = string
    port_max = string
  }))
  nullable = true
}

#
# IP network addressing
#
variable "vcn_cidr" {
  default = "172.31.252.0/23"
}
```

```
# Subnet for KMIs where kube-apiserver and other control
# plane applications run
variable "kmi_cidr" {
  description = "K8s control plane subnet CIDR"
  default     = "172.31.252.224/28"
}

# Subnet for KMI load balancer
variable "kmilb_cidr" {
  description = "K8s control plane LB subnet CIDR"
  default     = "172.31.252.240/28"
}

# Subnet for worker nodes, max 128 nodes
variable "worker_cidr" {
  description = "K8s worker subnet CIDR"
  default     = "172.31.253.0/24"
}

# Subnet for worker load balancer (for use by CCM)
variable "workerlb_cidr" {
  description = "K8s worker LB subnet CIDR"
  default     = "172.31.252.0/25"
}
```

terraform.tfvars

This file assigns values to some of the variables that were created in `variables.tf`. It also defines security list rules for accessing the worker nodes and the worker load balancer.

```
# Name of the profile to use from $HOME/.oci/config
oci_config_file_profile = "DEFAULT"

# Tenancy OCID from the oci_config_file_profile profile.
tenancy_ocid = "ocid1.tenancy.unique_ID"

# Compartment in which to build the OKE cluster.
compartment_id = "ocid1.compartment.unique_ID"

# Display name for the OKE VCN.
vcn_name = "oketest"

# CIDR of clients that are allowed to contact Kubernetes API server.
kube_client_cidr = "10.0.0.0/8"

# Security list rules for who is allowed to contact the worker load balancer.
# Adjust these values for your applications.
worker_lb_ingress_rules = [
  {
    source   = "10.0.0.0/8"
    port_min = 80
    port_max = 80
  },
  {
    source   = "10.0.0.0/8"
    port_min = 443
    port_max = 443
  },
]
```

```
# Security list rules for who is allowed to contact worker nodes directly.
# This example allows 10.0.0.0/8 to contact the default nodeport range.
worker_ingress_rules = [
  {
    source   = "10.0.0.0/8"
    port_min = 30000
    port_max = 32767
  },
]
```

provider.tf

This file is required in order to use the OCI provider. The file initializes the OCI module using the OCI profile configuration file.

```
provider "oci" {
  config_file_profile = var.oci_config_file_profile
  tenancy_ocid        = var.tenancy_ocid
}
```

main.tf

This file specifies the provider to use (`oracle/oci`), defines several security list rules, and initializes required local variables.

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = ">= 4.50.0"
      # If necessary, you can pin a specific version here
      #version = "4.71.0"
    }
  }
  required_version = ">= 1.1"
}

locals {
  kube_internal_cidr = "253.255.0.0/16"
  worker_lb_ingress_rules = var.worker_lb_ingress_rules
  worker_ingress_rules = flatten([var.worker_ingress_rules, [
    {
      source   = var.vcn_cidr
      port_min = 22
      port_max = 22
    },
    {
      source   = var.workerlb_cidr
      port_min = 30000
      port_max = 32767
    },
    {
      source   = var.workerlb_cidr
      port_min = 10256
      port_max = 10256
    },
    {
      source   = var.kmi_cidr
      port_min = 22
      port_max = 65535
    },
  ]])
  worker_ingress_udp_rules = [
    {
      source   = var.worker_cidr
      port_min = 8285
      port_max = 8472
    },
    {
      source   = var.kmi_cidr
      port_min = 8285
      port_max = 8472
    },
  ]

  kmi_lb_ingress_rules = [
    {
      source   = local.kube_internal_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.kube_client_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.vcn_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
  ]

  kmi_ingress_rules = [
    {
      source   = var.kube_client_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.kmilb_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.worker_cidr
      port_min = 1024
      port_max = 65535
    },
    {
      source   = var.kmi_cidr
      port_min = 1024
      port_max = 65535
    },
  ]
  kmi_ingress_udp_rules = [
    {
      source   = var.worker_cidr
      port_min = 8285
      port_max = 8472
    },
    {
      source   = var.kmi_cidr
      port_min = 8285
      port_max = 8472
    },
  ]
}
```

oke_vcn.tf

This file defines a VCN, NAT gateway, internet gateway, private route table, and public route table. The private route table is the default route table for the VCN.

```
resource "oci_core_vcn" "oke_vcn" {
  cidr_block     = var.vcn_cidr
  dns_label      = var.vcn_name
  compartment_id = var.compartment_id
  display_name   = "${var.vcn_name}-vcn"
}

resource "oci_core_nat_gateway" "vcn_ngs" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "VCN nat g6s"
}

resource "oci_core_internet_gateway" "vcn_igs" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "VCN i6t g6s"
  enabled      = true
}

resource "oci_core_default_route_table" "private" {
  manage_default_resource_id = oci_core_vcn.oke_vcn.default_route_table_id
  display_name               = "Default - private"

  route_rules {
    destination       = "0.0.0.0/0"
    destination_type  = "CIDR_BLOCK"
    network_entity_id = oci_core_nat_gateway.vcn_ngs.id
  }
}

resource "oci_core_route_table" "public" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "public"

  route_rules {
    destination       = "0.0.0.0/0"
    destination_type  = "CIDR_BLOCK"
    network_entity_id = oci_core_internet_gateway.vcn_igs.id
  }
}
```

oke_worker_seclist.tf

This file defines the security lists for both the worker subnet and the worker load balancer subnet. The rules for these security lists were defined in other Terraform files in this set.

```
resource "oci_core_security_list" "workerlb" {
  display_name   = "${var.vcn_name}-workerlb"
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.worker_lb_ingress_rules

    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}

resource "oci_core_security_list" "worker" {
  display_name   = "${var.vcn_name}-worker"
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.worker_ingress_rules

    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.worker_ingress_udp_rules

    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "17"
      udp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}
```

oke_worker_subnet.tf

This file defines the worker and worker load balancer subnets. The worker load balancer subnet is named `service-lb`.

```
resource "oci_core_subnet" "worker" {
  cidr_block     = var.worker_cidr
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name               = "worker"
  dns_label                  = "worker"
  prohibit_public_ip_on_vnic = true

  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.worker.id
  ]
}

resource "oci_core_subnet" "worker_lb" {
  cidr_block     = var.workerlb_cidr
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name               = "service-lb"
  dns_label                  = "servicelb"
  prohibit_public_ip_on_vnic = false
  route_table_id             = oci_core_route_table.public.id

  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.workerlb.id
  ]
}
```

oke_kmi_seclist.tf

This file defines the security lists for the control plane and control plane load balancer subnets. This file also defines updates to make to the default security list for the VCN.

```
resource "oci_core_default_security_list" "oke_vcn" {
  manage_default_resource_id = oci_core_vcn.oke_vcn.default_security_list_id

  egress_security_rules {
    destination      = "0.0.0.0/0"
    destination_type = "CIDR_BLOCK"
    protocol         = "all"
  }

  dynamic "ingress_security_rules" {
    iterator = icmp_type
    for_each = [3, 8, 11]

    content {
      # ping from VCN; unreachable/TTL from anywhere
      source      = (icmp_type.value == "8" ? var.vcn_cidr : "0.0.0.0/0")
      source_type = "CIDR_BLOCK"
      protocol    = "1"
      icmp_options {
        type = icmp_type.value
      }
    }
  }
}
```

```
resource "oci_core_security_list" "kmilb" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name = "${var.vcn_name}-kmilb"

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.kmi_lb_ingress_rules

    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}

resource "oci_core_security_list" "kmi" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name = "${var.vcn_name}-kmi"

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.kmi_ingress_rules

    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.kmi_ingress_udp_rules

    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "17"
      udp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}
```

oke_kmi_subnet.tf

This file defines the control plane and control plane load balancer subnets.

Important:

The name of the `kmi` subnet must be exactly `control-plane`.

```
resource "oci_core_subnet" "kmi" {
  cidr_block                 = var.kmi_cidr
  compartment_id             = var.compartment_id
  display_name               = "control-plane"
  dns_label                  = "kmi"
  vcn_id                     = oci_core_vcn.oke_vcn.id
  prohibit_public_ip_on_vnic = true
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmi.id
  ]
}

resource "oci_core_subnet" "kmi_lb" {
  cidr_block                 = var.kmilb_cidr
  compartment_id             = var.compartment_id
  dns_label                  = "kmilb"
  vcn_id                     = oci_core_vcn.oke_vcn.id
  display_name               = "control-plane-endpoint"
  prohibit_public_ip_on_vnic = false
  route_table_id             = oci_core_route_table.public.id
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmilb.id
  ]
}
```

Creating an OKE VCN

Create the following resources in the order listed:

1. VCN
2. Internet gateway
3. Route table with public route rule
4. NAT gateway
5. Route table with private route rule
6. Modify the VCN default security list

Resource names and CIDR blocks are example values.

VCN

To create the VCN, use the instructions in Creating a VCN in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

For this example, use the following input to create the VCN. The VCN covers one contiguous CIDR block. The CIDR block cannot be changed after the VCN is created.

Compute Web UI property	OCI CLI property
• Name: oketest-vcn • CIDR Block: ***vcn_cidr*** • DNS Label: oketest This label must be unique across all VCNs in the tenancy.	• `--display-name`: `oketest-vcn` • `--cidr-blocks`: `'["`***vcn_cidr***`"]'` • `--dns-label`: `oketest` This label must be unique across all VCNs in the tenancy.

Note the OCID of the new VCN. In the examples in this guide, this VCN OCID is `ocid1.vcn.`***oke_vcn_id***.

Next steps: To enable internet access for OKE nodes, add an internet gateway and a route rule that references that internet gateway. For traffic that needs to go outside the VCN but not to the internet (for example, to your data center), add a NAT gateway and edit the default route table to add a route rule that references that NAT gateway.

Private Route Table

Create a NAT gateway, and edit the default route table to reference the NAT gateway.

NAT Gateway

To create the NAT gateway, use the instructions in Enabling Public Connections through a NAT Gateway in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

Note the name and OCID of the NAT gateway for assignment to the private route rule.

Private Route Rule

Modify the default route table, using the following input to create a private route rule that references the NAT gateway that was created in the preceding step. See "Updating Rules in a Route Table" in Working with Route Tables in the *Oracle Private Cloud Appliance User Guide*.

Compute Web UI property	OCI CLI property
• Display name: Default - private Route rule • Target Type: NAT Gateway • NAT Gateway: Name of the NAT gateway that was created in the preceding step • CIDR Block: 0.0.0.0/0 • Description: OKE private route rule	• `--rt-id`: `ocid1.routetable.`***`default_routetable_id`*** • `--display-name`: Default - private `--route-rules` • `networkEntityId`: OCID of the NAT gateway that was created in the preceding step • `destinationType`: `CIDR_BLOCK` • `destination`: `0.0.0.0/0` • `description`: OKE private route rule

Note the name and OCID of this route table for assignment to private subnets.

Public Route Table

Create an Internet gateway and a route table with a route rule that references the Internet gateway.

Internet Gateway

To create the internet gateway, use the instructions in Providing Public Access through an Internet Gateway in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

Note the name and OCID of the internet gateway for assignment to the public route rule.

Public Route Rule

To create a route table, use the instructions in "Creating a Route Table" in Working with Route Tables in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

For this example, use the following input to create the route table with a public route rule that references the internet gateway that was created in the preceding step.

Compute Web UI property	OCI CLI property
• Name: public Route rule • Target Type: Internet Gateway • Internet Gateway: Name of the internet gateway that was created in the preceding step • CIDR Block: 0.0.0.0/0 • Description: OKE public route rule	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: public `--route-rules` • `networkEntityId`: OCID of the internet gateway that was created in the preceding step • `destinationType`: `CIDR_BLOCK` • `destination`: `0.0.0.0/0` • `description`: OKE public route rule

Note the name and OCID of this route table for assignment to public subnets.

VCN Default Security List

Modify the default security list, using the input shown in the following table. Delete all of the default rules and create the rules shown in the following table.

To modify a security list, use the instructions in "Updating a Security List" in Controlling Traffic with Security Lists in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

Compute Web UI property	OCI CLI property
	`--security-list-id`: `ocid1.securitylist.`***`default_securitylist_id`***
One egress security rule: • Stateless: uncheck the box • Egress CIDR: 0.0.0.0/0 • IP Protocol: All protocols • Description: "Allow all outgoing traffic."	**One egress security rule:** `--egress-security-rules` • `isStateless`: `false` • `destination`: `0.0.0.0/0` • `destinationType`: `CIDR_BLOCK` • `protocol`: `all` • `description`: "Allow all outgoing traffic."
Three ingress security rules:	**Three ingress security rules:** `--ingress-security-rules`
Ingress Rule 1 • Stateless: uncheck the box • Ingress CIDR: ***`vcn_cidr`*** • IP Protocol: ICMP – Parameter Type: 8: Echo • Description: "Allow ping from VCN."	**Ingress Rule 1** • `isStateless`: `false` • `source`: ***`vcn_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `1` • `icmpOptions` – `type`: `8` • `description`: "Allow ping from VCN."
Ingress Rule 2 • Stateless: uncheck the box • Ingress CIDR: 0.0.0.0/0 • IP Protocol: ICMP – Parameter Type: 3: Destination Unreachable • Description: "Allow unreachables."	**Ingress Rule 2** • `isStateless`: `false` • `source`: `0.0.0.0/0` • `sourceType`: `CIDR_BLOCK` • `protocol`: `1` • `icmpOptions` – `type`: `3` • `description`: "Allow unreachables."

Compute Web UI property	OCI CLI property
Ingress Rule 3 • Stateless: uncheck the box • Ingress CIDR: 0.0.0.0/0 • IP Protocol: ICMP – Parameter Type: 11: Time Exceeded • Description: "Allow time exceeded."	**Ingress Rule 3** • `isStateless`: `false` • `source`: `0.0.0.0/0` • `sourceType`: `CIDR_BLOCK` • `protocol`: `1` • `icmpOptions` – `type`: `11` • `description`: "Allow time exceeded."

Note the name and OCID of this default security list for assignment to subnets.

Creating an OKE Worker Subnet

Create the following resources in the order listed:

1. Worker security list
2. Worker subnet

Create a Worker Security List

To create a security list, use the instructions in "Creating a Security List" in Controlling Traffic with Security Lists in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

This security list defines traffic that is allowed to contact worker nodes directly.

For this example, use the following input for the worker subnet security list.

Compute Web UI property	OCI CLI property
• Name: worker-seclist	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `worker-seclist`
Seven ingress security rules:	**Seven ingress security rules:** `--ingress-security-rules`
Ingress Rule 1 • Stateless: uncheck the box • Ingress CIDR: ***`vcn_cidr`*** • IP Protocol: TCP – Destination Port Range: 22 • Description: "Allow intra-VCN `ssh`."	**Ingress Rule 1** • `isStateless`: `false` • `source`: ***`vcn_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `22` – `min`: `22` • `description`: "Allow intra-VCN `ssh`."

Compute Web UI property	OCI CLI property
Ingress Rule 2 • Stateless: uncheck the box • Ingress CIDR: ***`kube_client_cidr`*** • IP Protocol: TCP – Destination Port Range: 30000-32767 • Description: "Allow clients to contact the node port range."	**Ingress Rule 2** • `isStateless`: `false` • `source`: ***`kube_client_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `32767` – `min`: `30000` • `description`: "Allow clients to contact the node port range."
Ingress Rule 3 • Stateless: uncheck the box • Ingress CIDR: ***`workerlb_cidr`*** • IP Protocol: TCP – Destination Port Range: 30000-32767 • Description: "Allow the worker load balancer to contact the worker nodes."	**Ingress Rule 3** • `isStateless`: `false` • `source`: ***`workerlb_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `32767` – `min`: `30000` • `description`: "Allow the worker load balancer to contact the worker nodes."
Ingress Rule 4 • Stateless: uncheck the box • Ingress CIDR: ***`workerlb_cidr`*** • IP Protocol: TCP – Destination Port Range: 10256 • Description: "Allow the worker load balancer to contact the worker nodes."	**Ingress Rule 4** • `isStateless`: `false` • `source`: ***`workerlb_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `10256` – `min`: `10256` • `description`: "Allow the worker load balancer to contact the worker nodes."
Ingress Rule 5 • Stateless: uncheck the box • Ingress CIDR: ***`kmi_cidr`*** • IP Protocol: TCP – Destination Port Range: 22-65535 • Description: "Allow the control plane to contact the worker nodes."	**Ingress Rule 5** • `isStateless`: `false` • `source`: ***`kmi_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `65535` – `min`: `22` • `description`: "Allow the control plane to contact the worker nodes."

Compute Web UI property	OCI CLI property
Ingress Rule 6 • Stateless: uncheck the box • Ingress CIDR: ***worker_cidr*** • IP Protocol: UDP – Destination Port Range: 8285-8472 • Description: "Allow flannel traffic."	**Ingress Rule 6** • `isStateless`: `false` • `source`: ***`worker_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `8472` – `min`: `8285` • `description`: "Allow flannel traffic."
Ingress Rule 7 • Stateless: uncheck the box • Ingress CIDR: ***kmi_cidr*** • IP Protocol: UDP – Destination Port Range: 8285-8472 • Description: "Allow flannel traffic."	**Ingress Rule 7** • `isStateless`: `false` • `source`: ***`kmi_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `8472` – `min`: `8285` • `description`: "Allow flannel traffic."

Create the Worker Subnet

To create a subnet, use the instructions in Creating a Subnet in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

For this example, use the following input to create the worker subnet. Use the OCID of the VCN that was created in Creating an OKE VCN. Create the worker subnet in the same compartment where you created the VCN.

Compute Web UI property	OCI CLI property
• Name: worker • CIDR Block: ***worker_cidr*** • Route Table: Select "Default - private" from the list • Private Subnet: check the box • DNS Hostnames: Use DNS Hostnames in this Subnet: check the box – DNS Label: worker • Security Lists: Select "worker-seclist" and "Default Security List for oketest-vcn" from the list	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `worker` • `--cidr-block`: ***`worker_cidr`*** • `--dns-label`: `worker` • `--prohibit-public-ip-on-vnic`: `true` • `--route-table-id`: OCID of the "Default - private" route table • `--security-list-ids`: OCIDs of the "worker-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating an OKE Worker Load Balancer Subnet

Create the following resources in the order listed:

1. Worker load balancer security list
2. Worker load balancer subnet

Create a Worker Load Balancer Security List

To create a security list, use the instructions in "Creating a Security List" in Controlling Traffic with Security Lists in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

This security list defines traffic, such as applications, that is allowed to contact the worker load balancer.

For this example, use the following input for the worker load balancer subnet security list. These sources and destinations are examples; adjust these for your applications.

Note:

When you create an external load balancer for your containerized applications (see Exposing Containerized Applications), remember to add that load balancer service front-end port to this security list.

Compute Web UI property	OCI CLI property
• Name: workerlb-seclist	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `workerlb-seclist`
Two ingress security rules:	**Two ingress security rules:** `--ingress-security-rules`
Ingress Rule 1 • Stateless: uncheck the box • Ingress CIDR: ***`kube_client_cidr`*** • IP Protocol: TCP – Destination Port Range: 80 • Description: "Allow inbound traffic for applications."	**Ingress Rule 1** • `isStateless`: `false` • `source`: ***`kube_client_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `80` – `min`: `80` • `description`: "Allow inbound traffic for applications."

Compute Web UI property	OCI CLI property
Ingress Rule 2 • Stateless: uncheck the box • Ingress CIDR: ***`kube_client_cidr`*** • IP Protocol: TCP – Destination Port Range: 443 • Description: "Allow inbound traffic for applications."	**Ingress Rule 2** • `isStateless`: `false` • `source`: ***`kube_client_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `443` – `min`: `443` • `description`: "Allow inbound traffic for applications."

Create the Worker Load Balancer Subnet

To create a subnet, use the instructions in Creating a Subnet in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

For this example, use the following input to create the worker load balancer subnet. Use the OCID of the VCN that was created in Creating an OKE VCN. Create the worker load balancer subnet in the same compartment where you created the VCN.

Compute Web UI property	OCI CLI property
• Name: service-lb • CIDR Block: ***`workerlb_cidr`*** • Route Table: Select "public" from the list • Public Subnet: check the box • DNS Hostnames: Use DNS Hostnames in this Subnet: check the box – DNS Label: servicelb • Security Lists: Select "workerlb-seclist" and "Default Security List for oketest-vcn" from the list	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `service-lb` • `--cidr-block`: ***`workerlb_cidr`*** • `--dns-label`: `servicelb` • `--prohibit-public-ip-on-vnic`: `false` • `--route-table-id`: OCID of the "public" route table • `--security-list-ids`: OCIDs of the "workerlb-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating an OKE Control Plane Subnet

Create the following resources in the order listed:

1. Control plane security list
2. Control plane subnet

Create a Control Plane Security List

To create a security list, use the instructions in "Creating a Security List" in Controlling Traffic with Security Lists in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

For this example, use the following input for the control plane subnet security list.

Compute Web UI property	OCI CLI property
• Name: kmi-seclist	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `kmi-seclist`
Six ingress security rules:	**Six ingress security rules:** `--ingress-security-rules`
Ingress Rule 1 • Stateless: uncheck the box • Ingress CIDR: ***`kube_client_cidr`*** • IP Protocol: TCP – Destination Port Range: ***`kubernetes_api_port`*** • Description: "Allow inbound connections to the Kubernetes API server."	**Ingress Rule 1** • `isStateless`: `false` • `source`: ***`kube_client_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: ***`kubernetes_api_port`*** – `min`: ***`kubernetes_api_port`*** • `description`: "Allow inbound connections to the Kubernetes API server."
Ingress Rule 2 • Stateless: uncheck the box • Ingress CIDR: ***`kmilb_cidr`*** • IP Protocol: TCP – Destination Port Range: ***`kubernetes_api_port`*** • Description: "Allow inbound connections from the control plane load balancer."	**Ingress Rule 2** • `isStateless`: `false` • `source`: ***`kmilb_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: ***`kubernetes_api_port`*** – `min`: ***`kubernetes_api_port`*** • `description`: "Allow inbound connections from the control plane load balancer."
Ingress Rule 3 • Stateless: uncheck the box • Ingress CIDR: ***`worker_cidr`*** • IP Protocol: TCP – Destination Port Range: 1024-65535 • Description: "Allow inbound connections from worker nodes to the control plane."	**Ingress Rule 3** • `isStateless`: `false` • `source`: ***`worker_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `65535` – `min`: `1024` • `description`: "Allow inbound connections from worker nodes to the control plane."

Compute Web UI property	OCI CLI property
Ingress Rule 4 • Stateless: uncheck the box • Ingress CIDR: ***kmi_cidr*** • IP Protocol: TCP – Destination Port Range: 1024-65535 • Description: "Allow inbound connections within the control plane."	**Ingress Rule 4** • `isStateless`: `false` • `source`: ***`kmi_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `65535` – `min`: `1024` • `description`: "Allow inbound connections within the control plane."
Ingress Rule 5 • Stateless: uncheck the box • Ingress CIDR: ***worker_cidr*** • IP Protocol: UDP – Destination Port Range: 8285-8472 • Description: "Allow flannel traffic."	**Ingress Rule 5** • `isStateless`: `false` • `source`: ***`worker_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `8472` – `min`: `8285` • `description`: "Allow flannel traffic."
Ingress Rule 6 • Stateless: uncheck the box • Ingress CIDR: ***kmi_cidr*** • IP Protocol: UDP – Destination Port Range: 8285-8472 • Description: "Allow flannel traffic."	**Ingress Rule 6** • `isStateless`: `false` • `source`: ***`kmi_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `8472` – `min`: `8285` • `description`: "Allow flannel traffic."

Create the Control Plane Subnet

To create a subnet, use the instructions in Creating a Subnet in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

Use the following input to create the control plane subnet. Use the OCID of the VCN that was created in Creating an OKE VCN. Create the control plane subnet in the same compartment where you created the VCN.

Important:

The name of this subnet must be exactly "control-plane".

Compute Web UI property	OCI CLI property
• Name: control-plane • CIDR Block: ***kmi_cidr*** • Route Table: Select "Default - private" from the list • Private Subnet: check the box • DNS Hostnames: Use DNS Hostnames in this Subnet: check the box – DNS Label: kmi • Security Lists: Select "kmi-seclist" and "Default Security List for oketest-vcn" from the list	• `--vcn-id`: `ocid1.vcn.`***oke_vcn_id*** • `--display-name`: `control-plane` • `--cidr-block`: ***kmi_cidr*** • `--dns-label`: `kmi` • `--prohibit-public-ip-on-vnic`: `true` • `--route-table-id`: OCID of the "Default - private" route table • `--security-list-ids`: OCIDs of the "kmi-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating an OKE Control Plane Load Balancer Subnet

Create the following resources in the order listed:

1. Control plane load balancer security list
2. Control plane load balancer subnet

Create a Control Plane Load Balancer Security List

To create a security list, use the instructions in "Creating a Security List" in Controlling Traffic with Security Lists in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

The control plane load balancer accepts traffic on port 6443, which is also called ***kubernetes_api_port*** in this guide. Adjust this security list to only accept connections from where you expect the network to run. Port 6443 must accept connections from the cluster control plane instances and worker instances.

For this example, use the following input for the control plane load balancer subnet security list.

Compute Web UI property	OCI CLI property
• Name: kmilb-seclist	• `--vcn-id`: `ocid1.vcn.`***oke_vcn_id*** • `--display-name`: kmilb-seclist
Three ingress security rules:	**Three ingress security rules:** `--ingress-security-rules`

Compute Web UI property	OCI CLI property
Ingress Rule 1: • Stateless: uncheck the box • Ingress CIDR: `253.255.0.0/16` This value is required. Do not change this CIDR value. • IP Protocol: TCP – Destination Port Range: ***`kubernetes_api_port`*** • Description: "Allow inbound connections to the control plane load balancer."	**Ingress Rule 1:** • `isStateless`: `false` • `source`: `253.255.0.0/16` This value is required. Do not change this CIDR value. • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: ***`kubernetes_api_port`*** – `min`: ***`kubernetes_api_port`*** • `description`: "Allow inbound connections to the control plane load balancer."
Ingress Rule 2: • Stateless: uncheck the box • Ingress CIDR: ***`kube_client_cidr`*** • IP Protocol: TCP – Destination Port Range: ***`kubernetes_api_port`*** • Description: "Allow inbound connections to the control plane load balancer."	**Ingress Rule 2:** • `isStateless`: `false` • `source`: ***`kube_client_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: ***`kubernetes_api_port`*** – `min`: ***`kubernetes_api_port`*** • `description`: "Allow inbound connections to the control plane load balancer."
Ingress Rule 3: • Stateless: uncheck the box • Ingress CIDR: ***`vcn_cidr`*** • IP Protocol: TCP – Destination Port Range: ***`kubernetes_api_port`*** • Description: "Allow inbound connections to the control plane load balancer."	**Ingress Rule 3:** • `isStateless`: `false` • `source`: ***`vcn_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: ***`kubernetes_api_port`*** – `min`: ***`kubernetes_api_port`*** • `description`: "Allow inbound connections to the control plane load balancer."

Create the Control Plane Load Balancer Subnet

To create a subnet, use the instructions in Creating a Subnet in the *Oracle Private Cloud Appliance User Guide*. For Terraform input, see Example Terraform Scripts for Network Resources.

For this example, use the following input to create the control plane load balancer subnet. Use the OCID of the VCN that was created in Creating an OKE VCN. Create the control plane load balancer subnet in the same compartment where you created the VCN.

Compute Web UI property	OCI CLI property
• Name: control-plane-endpoint • CIDR Block: ***kmilb_cidr*** • Route Table: Select "public" from the list • Public Subnet: check the box • DNS Hostnames: Use DNS Hostnames in this Subnet: check the box – DNS Label: kmilb • Security Lists: Select "kmilb-seclist" and "Default Security List for oketest-vcn" from the list	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `control-plane-endpoint` • `--cidr-block`: ***`kmilb_cidr`*** • `--dns-label`: `kmilb` • `--prohibit-public-ip-on-vnic`: `false` • `--route-table-id`: OCID of the "public" route table • `--security-list-ids`: OCIDs of the "kmilb-seclist" security list and the "Default Security List for oketest-vcn" security list

4

Creating and Managing OKE Clusters

This chapter describes how to create, update, and delete an OKE cluster. Be sure to carefully read the descriptions of the cluster parameters before you create the cluster.

A cluster includes cluster management nodes. This chapter describes how to recognize those management nodes in a list of all instances in a tenancy.

This chapter also describes how to create a Kubernetes configuration file. You need a Kubernetes configuration file for each OKE cluster that you work with. The Kubernetes configuration file enables you to access OKE clusters using the `kubectl` command and the Kubernetes Dashboard.

Creating an OKE Cluster

These procedures describe how to create an OKE cluster.

The Network Load Balancer and public IP address are created and assigned as part of cluster creation.

> **Important:**
>
> Before you can create a cluster, the following conditions must be met:
>
> - The OraclePCA-OKE/cluster_id defined tag must exist in the tenancy.
> - All fault domains must be healthy.
> - Each fault domain must have at least one healthy compute instance.
> - Sufficient resources must be available to create a cluster.
> - Ensure that no appliance upgrade is scheduled during the cluster create.

The OraclePCA-OKE/cluster_id defined tag is required to create or update an OKE cluster or node pool. This tag also is used to identify instances that need to be in a dynamic group. To verify the tag exists, in the Compute Web UI select Governance > Tag Namespaces and make sure the tenancy (root compartment) is selected on the compartment menu above the list. In the OCI CLI, use the following command:

```
$ oci iam tag-namespace list --compartment-id $OCI_CLI_TENANCY
```

If notifications are configured for operations such as system upgrade, ensure you are on the list to be notified of such planned outages.

After you create a cluster, see the Cluster Next Steps section.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).

2. On the clusters list page, click the Create Cluster button.
3. On the Cluster page in the Create Cluster dialog, provide the following information:
   - **Name**: The name of the new cluster. Avoid entering confidential information.
   - **Compartment**: The compartment in which to create the new cluster.
   - **Kubernetes Version**: The version of Kubernetes to run on the control plane nodes. Accept the default version or select a different version.
   - **Tagging**: Add defined or free-form tags for the cluster resource.

     Note:

     Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the cluster resource.

     Use free-form tags to provide the following information for control plane nodes:

     – Your public SSH key.

       Specify `sshkey` for the tag key. Paste your public SSH key into the Value field.

       Important:

       You cannot add an SSH key after the cluster is created.

     – Number of nodes.

       By default, the number of nodes in the control plane is 3. You can specify 1, 3, or 5 nodes. To specify the number of control plane nodes, specify `cp_node_count` for the tag key, and enter 1, 3, or 5 in the Value field.

     – Node shape.

       For Private Cloud Appliance X10 systems, the shape of the control plane nodes is VM.PCAStandard.E5.Flex and you cannot change it. For all other Private Cloud Appliance systems, the default shape is VM.PCAStandard1.1, and you can specify a different shape.

       To use a different shape, specify `cp_node_shape` for the tag key, and enter the name of the shape in the Value field. For a description of each shape, see Compute Shapes in the *Oracle Private Cloud Appliance Concepts Guide*.

     – Node shape configuration.

       If you specify a shape that is not a flexible shape, do not specify a shape configuration. The number of OCPUs and amount of memory are set to the values shown for this shape in "Standard Shapes" in Compute Shapes in the *Oracle Private Cloud Appliance Concepts Guide*.

       If you specify a flexible shape, you can change the default shape configuration.

To provide shape configuration information, specify `cp_node_shape_config` for the tag key. You must specify the number of OCPUs (`ocpus`) you want. You can optionally specify the total amount of memory you want (`memoryInGBs`). The default value for gigabytes of memory is 16 times the number you specify for OCPUs.

The following are examples of node shape configuration values. Enter everything, including the surrounding single quotation marks, in the Value field for the tag. In the first example, the default amount of memory will be configured.

```
'{"ocpus":1}'
'{"ocpus":2, "memoryInGBs":24}'
```

4. Click Next.
5. On the Network page in the Create Cluster dialog, provide the following information:
   - **Network Type**. Specifies how pods running on nodes in the cluster communicate with each other, with the cluster's control plane nodes, with pods on other clusters, with other services (such as storage services), and with the internet.

     The **Flannel overlay** network type encapsulates communication between pods in the flannel overlay network. The flannel overlay network is a simple private overlay virtual network that satisfies the requirements of the OKE networking model by attaching IP addresses to containers. The pods in the private overlay network are only accessible from other pods in the same cluster.
   - **VCN**. Select the VCN that has the configuration of the "oke_vcn" VCN described in Creating an OKE VCN.
   - **Kubernetes Service LB Subnet**. The subnet that is configured to host the load balancer in an OKE cluster. Select the subnet that has configuration like the "service-lb" subnet described in Creating an OKE Worker Load Balancer Subnet.
   - **Kubernetes API Endpoint Subnet**. The regional subnet in which to place the cluster endpoint. Select the subnet that has configuration like the "control-plane-endpoint" subnet described in Creating an OKE Control Plane Load Balancer Subnet.
   - **Kubernetes Service CIDR Block**. (Optional) The default value is 10.96.0.0/16.
   - **Pods CIDR Block**. (Optional) The default value is 10.244.0.0/16.
   - **Network Security Group**. If you check the box to enable network security groups, click the Add Network Security Group button and select an NSG from the drop-down list. You might need to change the compartment to find the NSG you want.
6. Click Next.
7. Review your entries and click Submit.

   The details page for the cluster is displayed. Scroll to the Resources section and click Work Requests to see the progress of the cluster creation. When the cluster is in the Active state, click Node Pools to add a node pool. See the Cluster Next Steps section.

   The cluster details page does not list the cluster control plane nodes. To view the control plane nodes, view the list of instances in the compartment where you created this cluster. Names of control plane nodes are in the following format:

   ```
   oke-ID1-control-plane-ID2
   ```

   - ***`ID1`*** - The first 32 characters after the ***`pca_name`*** in the cluster OCID.
   - ***`ID2`*** - A unique identifier added when the cluster has more than one control plane node.

Search for the instances in the list whose names contain the ***ID1*** string from this cluster OCID.

Using the OCI CLI

1. Get the information you need to run the command.
   - The OCID of the compartment where you want to create the cluster: `oci iam compartment list`
   - The name of the cluster. Avoid using confidential information.
   - OCID of the virtual cloud network (VCN) in which you want to create the cluster. Specify the VCN that has the configuration of the "oke_vcn" VCN described in Creating an OKE VCN.
   - OCID of the OKE service LB subnet. Specify the subnet that has configuration like the "service-lb" subnet described in Creating an OKE Worker Load Balancer Subnet. Specify only one OKE Service LB subnet.
   - OCID of the Kubernetes API endpoint subnet. Specify the subnet that has configuration like the "control-plane-endpoint" subnet described in Creating an OKE Control Plane Load Balancer Subnet.
   - OKE service CIDR block. (Optional) The default value is 10.96.0.0/16.
   - Pods CIDR block. (Optional) The default value is 10.244.0.0/16.
   - (Optional) The OCID of the Network Security Group to apply to the cluster endpoint. Do not specify more than one NSG. If you specify an NSG, use the following syntax:

     ```
     --endpoint-nsg-ids '["ocid1.networksecuritygroup.unique_ID"]'
     ```
   - (Optional) Your public SSH key in RSA format. You cannot add or update an SSH key after the cluster is created.
   - The network type. You do not need to specify the network type because `FLANNEL_OVERLAY` is used by default. See the descriptions in the Compute Web UI procedure. If you specify the network type, you must specify the following:

     ```
     --cluster-pod-network-options '{"cniType":"FLANNEL_OVERLAY"}'
     ```
2. (Optional) Add defined or free-form tags for the cluster resource by using the `--defined-tags` and `--freeform-tags` options.

   > **Note:**
   >
   > Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the cluster resource.

   Define an argument for the `--freeform-tags` option to provide the following information for control plane nodes:

   - Your public SSH key.

     Specify `sshkey` for the tag key, and paste your public SSH key as the value.

> **Important:**
>
> You cannot add an SSH key after the cluster is created.

- Number of nodes.

  By default, the number of nodes in the control plane is 3. You can specify 1, 3, or 5 nodes. To specify the number of control plane nodes, specify `cp_node_count` for the tag key, and enter 1, 3, or 5 in the Value field.

- Node shape.

  For Private Cloud Appliance X10 systems, the shape of the control plane nodes is VM.PCAStandard.E5.Flex and you cannot change it. For all other Private Cloud Appliance systems, the default shape is VM.PCAStandard1.1, and you can specify a different shape.

  To use a different shape, specify `cp_node_shape` for the tag key, and enter the name of the shape as the value. Use the following command to list the available shapes and their characteristics. To list only shapes that are compatible with the image that you plan to use, specify the image OCID. Use the `compute image list` command to find the image OCID.

  ```
  $ oci compute shape list --compartment-id compartment_OCID --image-id
  image_OCID
  ```

- Node shape configuration.

  If you specify a shape that is not a flexible shape, do not specify a shape configuration. The number of OCPUs and amount of memory are set to the values shown for this shape in "Standard Shapes" in Compute Shapes in the *Oracle Private Cloud Appliance Concepts Guide*.

  If you specify a flexible shape, you can change the default shape configuration. To provide shape configuration information, specify `cp_node_shape_config` for the tag key. You must specify the number of OCPUs (`ocpus`) you want. You can optionally specify the total amount of memory you want (`memoryInGBs`). The default value for gigabytes of memory is 16 times the number you specify for OCPUs.

  Specify free-form tags either inline or in a file in JSON format, such as the following example file:

  ```
  {
    "sshkey": "ssh-rsa remainder_of_key_text",
    "cp_node_count": 1,
    "cp_node_shape": "VM.PCAStandard1.Flex",
    "cp_node_shape_config": {
      "ocpus": 2,
      "memoryInGBs": 24
    }
  }
  ```

  Use the following syntax to specify a file of tags. Specify the full path to the `.json` file unless the file is in the same directory where you are running the command.

  ```
  --freeform-tags file://cluster_tags.json
  ```

3. Run the create cluster command.

   Example:

The `--endpoint-public-ip-enabled true` option is required when `--endpoint-subnet-id` or `--endpoint-nsg-ids` is specified.

```
$ oci ce cluster create \
--compartment-id ocid1.compartment.unique_ID --kubernetes-version version \
--name "Cluster One" --vcn-id ocid1.vcn.unique_ID \
--endpoint-public-ip-enabled true \
--endpoint-subnet-id control-plane-endpoint_subnet_OCID \
--service-lb-subnet-ids '["service-lb_subnet_OCID"]' \
--freeform-tags '{"sshkey":"ssh-rsa remainder_of_key_text"}'
```

The output from this `cluster create` command is the same as the output from the `cluster get` command.

Use the `work-request get` command to check the status of the create operation. The work request OCID is in `created-by-work-request-id` in the `metadata` section of the `cluster create` output.

```
$ oci ce work-request get --work-request-id workrequest_OCID
```

When the cluster is in the `ACTIVE` state, you can add a node pool. See the Cluster Next Steps section.

To identify the control plane nodes for this cluster, list instances in the compartment where you created the cluster. Names of control plane nodes are in the following format:

```
oke-ID1-control-plane-ID2
```

- ***`ID1`*** - The first 32 characters after the ***`pca_name`*** in the cluster OCID.
- ***`ID2`*** - A unique identifier added when the cluster has more than one control plane node.

Search for the instances in the list whose names contain the ***`ID1`*** string from this cluster OCID.

Cluster Next Steps

1. Create a Kubernetes configuration file for the cluster. See Creating a Kubernetes Configuration File.
2. Deploy a Kubernetes Dashboard to manage the cluster and to manage and troubleshoot applications running in the cluster. On the https://kubernetes.io/ site, see Deploy and Access the Kubernetes Dashboard.
3. Create a node pool for the cluster. See Creating an OKE Worker Node Pool.
4. Create a backup for the workload cluster. For example, see Backing up an etcd cluster and Restoring up an etcd cluster in Operating etcd clusters for Kubernetes. Use the etcd backup to recover OKE clusters under disaster scenarios such as losing all control plane nodes. An etcd backup contains all OKE states and critical information. An etcd backup does not back up applications or other content on cluster nodes.

Creating a Kubernetes Configuration File

Set up a Kubernetes configuration file for each OKE cluster that you work with. Your Kubernetes configuration file enables you to access OKE clusters using the `kubectl` command and the Kubernetes Dashboard.

Kubernetes configuration files organize information about clusters, users, namespaces, and authentication mechanisms. You can define contexts to easily switch between clusters and namespaces. The `kubectl` tool uses Kubernetes configuration files to find the information it needs to choose a cluster and communicate with the API server of a cluster.

Installing the Kubernetes Command Line Tool

Install and configure the Kubernetes command line tool `kubectl`. The `kubectl` tool enables you to perform operations on OKE clusters such as deploy applications, inspect and manage cluster resources, and view logs.

To install `kubectl`, see https://kubernetes.io/docs/tasks/tools/. The `kubectl` version must be within one minor version of the OKE cluster Kubernetes version. For example, a v1.29 client can communicate with v1.28, v1.29, and v1.30 control planes. See Supported Versions of Kubernetes.

For more information, including a complete list of `kubectl` operations, see the Command line tool (kubectl) reference page.

Creating a Kubernetes Configuration File

Use the OCI CLI to create your Kubernetes configuration file.

Tip:

The Quick Start button on a cluster details page in the Compute Web UI shows how to create a Kubernetes configuration file, and provides the OCID of the cluster.

1. Get the OCID of the cluster: `oci ce cluster list`
2. Run the command to create the configuration file.

   The `--cluster-id` option is the only required option.

   The default value of the `--file` option is `~/.kube/config`. If you already have a file at the specified location and you want to replace it, use the `--overwrite` option. To maintain more than one configuration file, select a different file by using the `KUBECONFIG` environment variable or the `--kubeconfig` option.

   The value of the `--kube-endpoint` option must be `PUBLIC_ENDPOINT`.

   If provided, the value of the `--token-version` option must be 2.0.0.

   Example:

   Use the following command to configure a Kubeconfig file for the specified cluster using the public endpoint:

   ```
   $ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.unique_ID \
   --file $HOME/.kube/config --kube-endpoint PUBLIC_ENDPOINT
   New config written to the Kubeconfig file /home/username/.kube/config
   ```

   Use the following command to set your `KUBECONFIG` environment variable to the Kubeconfig file that you created or updated in the preceding command:

   ```
   $ export KUBECONFIG=$HOME/.kube/config
   ```

   The following command shows the content of your new YAML configuration file:

```
$ kubectl config view
```

If you run the command again with a different cluster OCID, the new information is merged with the existing information. The following message is displayed:

```
Existing Kubeconfig file found at /home/username/.kube/config and new config
merged into it
```

Verify Your Cluster Access

Run the following command to confirm that you can access your cluster:

```
$ kubectl cluster-info
```

Every Kubernetes namespace contains at least one ServiceAccount: the default ServiceAccount for that namespace, which is named `default`. If you do not specify a ServiceAccount when you create a Pod, the OKE service automatically assigns the ServiceAccount named `default` in that namespace.

An application running inside a Pod can access the Kubernetes API using automatically mounted service account credentials.

Updating an OKE Cluster

When you update a cluster, you can change the cluster name, Kubernetes version, and tags.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).
2. Click the name of the cluster that you want to update.
3. At the top of the cluster details page, click the Edit button.

   Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the cluster resource.
4. When you are finished making changes, click Save Changes.
5. If a Kubernetes version update is available, a link labeled "Upgrade Available" is displayed next to the Kubernetes Version number on the cluster details page. Click that link to display a drop-down menu of versions that you can select.

Using the OCI CLI

1. Get the OCID of the cluster that you want to update: `oci ce cluster list`
2. Run the update cluster command.

   If you specify the `--defined-tags` or `--freeform-tags` options, do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the cluster resource.

   Example:

   ```
   $ oci ce cluster update --cluster-id ocid1.cluster.unique_ID \
   --kubernetes-version newer_kubernetes_version --name new_cluster_name
   ```

For the value of the `--kubernetes-version` option, check Supported Versions of Kubernetes.

Deleting an OKE Cluster

Deleting a cluster deletes the cluster control plane nodes, worker nodes, and node pools. Other cluster resources such as VCNs, internet gateways, NAT gateways, route tables, security lists, load balancers, and block volumes are not deleted when you delete the cluster. Those resources must be deleted separately.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).
2. For the cluster that you want to delete, click the Actions menu, and click Delete.
3. Confirm that you want to delete the cluster.

   Enter the cluster name, and click the Delete button.

Using the OCI CLI

1. Get the OCID of the cluster that you want to delete: `oci ce cluster list`
2. Run the delete cluster command.

   Example:

   ```
   $ oci ce cluster delete --cluster-id ocid1.cluster.unique_ID --force
   ```

5

Creating and Managing OKE Worker Node Pools

This chapter describes how to create, update, and delete node pools for an OKE cluster. Be sure to carefully read the descriptions of the node pool parameters before you create the node pool.

This chapter also describes how to recognize node pool nodes in a list of all instances in a tenancy, and how to delete a single node from a node pool.

If a proxy is required to reach outside registries or repositories, for example, follow the instructions in Configuring a Proxy for each node in the node pool.

Creating an OKE Worker Node Pool

These procedures describe how to create a pool of worker nodes for an OKE workload cluster. Nodes are Private Cloud Appliance compute instances.

You cannot customize the OKE cloud-init scripts.

To add defined or free-form tags to all nodes in the node pool, use the OCI CLI.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).

   If the cluster to which you want to attach a node pool is not listed, select a different compartment from the compartment menu above the list.

2. Click the name of the cluster to which you want to add a node pool.
3. On the cluster details page, scroll to the Resources section, and click Node Pools.
4. On the Node Pools list, click the Add Node Pool button.
5. In the Add Node Pool dialog, provide the following information:
   - **Name**: The name of the new node pool. Avoid using confidential information.
   - **Compartment**: The compartment in which to create the new node pool.
   - **Node pool options**: In the Node Count field, enter the number of nodes you want in this node pool. The default is 0. The maximum number is 128 per cluster, which can be distributed across multiple node pools.
   - **Network Security Group**: If you check the box to enable network security groups, click the Add Network Security Group button and select an NSG from the drop-down list. You might need to change the compartment to find the NSG you want.
   - **Placement configuration**
     – **Subnet**: Select a subnet that has configuration like the "worker" subnet described in Creating an OKE Worker Subnet. Select only one subnet. The subnet must have rules set to communicate with the control plane endpoint. The

subnet must use the private route table and must have a security list like the worker-seclist security list described in Creating an OKE Worker Subnet.

  – **Fault domain**: Select a fault domain or select "Automatically select the best fault domain," which is the default option.

- **Source Image**: Select an image.

  a. Select the Platform Image Source Type.

  b. Select an image from the list.

     The image list has columns Operating System, OS Version, and Kubernetes Version. You can use the drop-down menu arrow to the right of the OS Version or Kubernetes Version to select a different version.

     > **Note:**
     >
     > The image that you specify must not have a Kubernetes version that is newer than the Kubernetes version that you specified when you created the cluster. The Kubernetes Version for the cluster is in a column of the cluster list table.

- **Shape**: Select a shape for the worker nodes. For a description of each shape, see Compute Shapes in the *Oracle Private Cloud Appliance Concepts Guide*. For Private Cloud Appliance X10 systems, the shape is VM.PCAStandard.E5.Flex and you cannot change it.

  If you select a shape that is not a flexible shape, the amount of memory and number of OCPUs are displayed. These numbers match the numbers shown for this shape in the table in the *Oracle Private Cloud Appliance Concepts Guide*.

  If you select a flexible shape, then you must specify the number of OCPUs you want. You can optionally specify the total amount of memory you want. The default value for gigabytes of memory is 16 times the number you specify for OCPUs. Click inside each value field to see the minimum and maximum allowed values.

- **Boot Volume**: (Optional) Check the box to specify a custom boot volume size.

  Boot volume size (GB): The default boot volume size for the selected image is shown. To specify a larger size, enter a value from 50 to 16384 in gigabytes (50 GB to 16 TB) or use the increment and decrement arrows.

  If you specify a custom boot volume size, you need to extend the partition to take advantage of the larger size. Oracle Linux platform images include the `oci-utils` package. Use the `oci-growfs` command from that package to extend the root partition and then grow the file system. See oci-growfs.

- **SSH Key**: The public SSH key for the worker nodes. Either upload the public key file or copy and paste the content of the file.

- **Tagging**: Add defined or free-form tags for the node pool resource.

> **Note:**
>
> Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the node pool resource.

6. Click the Add Node Pool button.

   The details page for the node pool is displayed. Scroll to the Resources section and click Work Requests to see the progress of the node pool creation and see nodes being added to the Nodes list.

   To identify these nodes in a list of instances, note that the names of these nodes are in the format `oke-`***`ID`***, where ***`ID`*** is the first 32 characters after the ***`pca_name`*** in the node pool OCID. Search for the instances in the list whose names contain the ***`ID`*** string from this node pool OCID.

Using the OCI CLI

1. Get the information you need to run the command.

   - The OCID of the compartment where you want to create the node pool: `oci iam compartment list`
   - The OCID of the cluster for this node pool: `oci ce cluster list`
   - The name of the node pool. Avoid using confidential information.
   - The placement configuration for the nodes, including the subnet and fault domain. See the "Placement configuration" description in the Compute Web UI procedure. Use the following command to show the content and format of this option:

     ```
     $ oci ce node-pool create --generate-param-json-input placement-configs
     ```

     Use the following command to list fault domains: `oci iam fault-domain list`. Do not specify more than one fault domain or more than one subnet in the placement configuration. To allow the system to select the best fault domains, do not specify any fault domain.
   - The OCID of the image to use for the nodes in this node pool.

     Use the following command to get the OCID of the image that you want to use:

     ```
     $ oci compute image list --compartment-id compartment_OCID
     ```

     > **Note:**
     >
     > The image that you specify must not have a Kubernetes version that is newer than the Kubernetes version that you specified when you created the cluster.

     The Kubernetes version for the cluster is shown in `cluster list` output. The Kubernetes version for the image is shown in the `display-name` property in `image list` output. The Kubernetes version of the following image is 1.28.3.

```
"display-name": "uln-pca-Oracle-Linux8-OKE-1.28.3-20240210.oci"
```

Do not specify the `--kubernetes-version` option in the `node-pool create` command.

You can specify a custom boot volume size in gigabytes. The default boot volume size is 50 GB. To specify a custom boot volume size, use the `--node-source-details` option to specify both the boot volume size and the image. You cannot specify both `--node-image-id` and `--node-source-details`. Use the following command to show the content and format of the node source details option.

```
$ oci ce node-pool create --generate-param-json-input node-source-details
```

If you specify a custom boot volume size, you need to extend the partition to take advantage of the larger size. Oracle Linux platform images include the `oci-utils` package. Use the `oci-growfs` command from that package to extend the root partition and then grow the file system. See oci-growfs.

- The name of the shape of the worker nodes in this node pool. For Private Cloud Appliance X10 systems, the shape of the control plane nodes is VM.PCAStandard.E5.Flex and you cannot change it. For all other Private Cloud Appliance systems, the default shape is VM.PCAStandard1.1, and you can specify a different shape.

  If you specify a flexible shape, then you must also specify the shape configuration, as shown in the following example. You must provide a value for `ocpus`. The `memoryInGBs` property is optional; the default value in gigabytes is 16 times the number of `ocpus`.

  ```
  --node-shape-config '{"ocpus": 32, "memoryInGBs": 512}'
  ```

  If you specify a shape that is not a flexible shape, do not specify `--node-shape-config`. The number of OCPUs and amount of memory are set to the values shown for this shape in "Standard Shapes" in Compute Shapes in the *Oracle Private Cloud Appliance Concepts Guide*.

- (Optional) The OCID of the Network Security Group to use for the nodes in this node pool. Do not specify more than one NSG.

- (Optional) Tags. Add defined or free-form tags for the node pool resource by using the `--defined-tags` and `--freeform-tags` options. Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the node pool resource.

  To add defined or free-form tags to all nodes in the node pool, use the `--node-defined-tags` and `--node-freeform-tags` options.

  > **Important:**
  >
  > Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated.

2. Run the create node pool command.

Example:

See the preceding Compute Web UI procedure for information about the options shown in this example and other options such as `--node-boot-volume-size-in-gbs` and `nsg-ids`.

```
$ oci ce node-pool create \
--cluster-id ocid1.cluster.unique_ID --compartment-id ocid1.compartment.unique_ID \
--name node_pool_name --node-shape shape_name --node-image-id
ocid1.image.unique_ID \
--placement-configs
'[{"availabilityDomain":"AD-1","subnetId":"ocid1.subnet.unique_ID"}]' \
--size 10 --ssh-public-key "public_key_text"
```

The output from this `node-pool create` command is the same as the output from the `node-pool get` command. The cluster OCID is shown, and a brief summary of each node is shown. For more information about a node, use the `compute instance get` command with the OCID of the node.

Use the `work-request get` command to check the status of the node pool create operation. The work request OCID is in `created-by-work-request-id` in the `metadata` section of the `cluster get` output.

```
$ oci ce work-request get --work-request-id workrequest_OCID
```

To identify these nodes in a list of instances, note that the names of these nodes are in the format `oke-ID`, where ***`ID`*** is the first 32 characters after the ***`pca_name`*** in the node pool OCID. Search for the instances in the list whose names contain the ***`ID`*** string from this node pool OCID.

Node Pool Next Steps

1. Configure any registries or repositories that the worker nodes need. Ensure you have access to a self-managed public or intranet container registry to use with the OKE service and your application images.
2. Configure any proxies that are needed. See Configuring a Proxy.
3. Create a service to expose containerized applications outside the Private Cloud Appliance. See Exposing Containerized Applications.
4. Create persistent storage for applications to use. See Adding Storage for Containerized Applications.

Configuring a Proxy

If your network requires proxy settings to enable worker nodes to reach outside registries or repositories, for example, log in to each worker node in the cluster (see Creating an OKE Worker Node Pool) and perform the following procedure.

1. Create a `crio.service.d` directory under `/etc/systemd/system`.

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

2. Create an `http-proxy.conf` file in `/etc/systemd/system/crio.service.d` and add proxy details.

   ```
   sudo cat /etc/systemd/system/crio.service.d/http-proxy.conf
   [Service]
   Environment="HTTP_PROXY=http://your_proxy.your_domain_name:your_port"
   ```

```
Environment="HTTPS_PROXY=http://your_proxy.your_domain_name:your_port"
Environment="no_proxy=localhost,127.0.0.1,your_domain_name,ocir.io,Kubernetes
_cidr,pods_cidr"
```

In the `no_proxy` entry, ***`Kubernetes_cidr`*** is the Kubernetes Service CIDR block that you enter when you create a cluster, and ***`pods_cidr`*** is the pods CIDR block that you enter when you create a cluster. See Creating an OKE Cluster.

3. Reload the `systemd` manager configuration and `crio`.

```
sudo systemctl daemon-reload
sudo systemctl restart crio
```

Updating an OKE Node Pool

When you update a node pool, existing nodes are not updated. The updated configuration only applies to new nodes that are created.

To change the properties of existing nodes, you could instead create a new node pool with the new settings and move the work to the new nodes.

You can update any configuration that you can set when you create a node pool except for the compartment where nodes will be created. See Creating an OKE Worker Node Pool for property descriptions.

The updated configuration only applies to new nodes that are created. New nodes are created when you increase the node count, change the fault domain, or change the subnet.

Important:

If you change the fault domain or subnet of a node pool, existing worker nodes are terminated and new worker nodes are created using the updated configuration.

If you make changes that add new worker nodes, see Node Pool Next Steps.

To add defined or free-form tags to all nodes in the node pool, use the OCI CLI.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).
2. Click the name of the cluster that contains the node pool that you want to update.
3. On the cluster details page, scroll to the Resources section, and click Node Pools.
4. For the node pool that you want to update in the Node Pools list, click the Actions menu and click Edit.

   The Edit Node Pool dialog opens. You can change any configuration except the compartment where new nodes will be created. See Creating an OKE Worker Node Pool for property descriptions. The updated configuration only applies to new nodes that are created, as described at the beginning of this topic.

> **Note:**
>
> Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the node pool resource.

5. When you are finished making changes, click Save Changes.

   The details page for the node pool is displayed. In addition to Node Pool Information and Tags tabs, the node pool details page has a Placement Configuration tab.

Using the OCI CLI

1. Get the OCID of the node pool that you want to update: `oci ce node-pool list`
2. (Optional) Tags. Add, change, or delete defined or free-form tags for the node pool resource by using the `--defined-tags` and `--freeform-tags` options. Do not specify values for the OraclePCA-OKE defined tag or for the ClusterResourceIdentifier free-form tag. These tag values are system-generated and only applied to nodes (instances), not to the node pool resource.

   To add, change, or delete defined or free-form tags for all nodes in the node pool, use the `--node-defined-tags` and `--node-freeform-tags` options.

   > **Important:**
   >
   > The argument you specify for the `--node-defined-tags` or `--node-freeform-tags` option replaces any existing tag definitions. Ensure that you copy and include the OraclePCA-OKE defined tag information in any `--node-defined-tags` argument, and copy and include the ClusterResourceIdentifier free-form tag information in any `--node-freeform-tags` argument. These tag values are system-generated and must not be changed or deleted.

3. Run the update node pool command.

   Syntax:

   ```
   $ oci ce node-pool update -node-pool-id ocid1.nodepool.unique_ID \
   new_configuration_settings
   ```

   The updated configuration only applies to new nodes that are created, as described at the beginning of this topic.

Deleting an OKE Node Pool Node

These procedures describe how to explicitly delete a worker node. Worker nodes are also deleted when you update a node pool to scale down the node pool or change the subnet or fault domains of the node pool. See Updating an OKE Node Pool.

Deleting a worker node permanently deletes the node. You cannot recover a deleted worker node.

When you delete a node, by default a new node is created to satisfy the node count set for the pool. To override this behavior, select the option to decrease node pool size.

Do not use the `kubectl delete node` command to terminate worker nodes in an OKE cluster. The `kubectl delete node` command removes the worker node from the cluster's etcd key-value store, but the command does not terminate the underlying compute instance.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).
2. Click the name of the cluster that contains the node that you want to delete.
3. On the cluster details page, scroll to the Resources section, and click Node Pools.
4. Click the name of the node pool that contains the node that you want to delete.
5. On the node pool details page, scroll to the Resources section, and click Nodes.
6. For the node that you want to delete, click the Actions menu, and click Delete.
7. Confirm the deletion.
    a. If you do not want a new node to be automatically created to replace the deleted node, click Decrease node pool size.
    b. Click the Delete button on the dialog.

Using the OCI CLI

1. Get the information you need to run the command.
    - OCID of the node pool: `oci ce node-pool list`
    - OCID of the node: `oci ce node-pool list`
2. Run the delete node pool node command.

    If you do not want a new node to be automatically created to replace the deleted node, specify the `--is-decrement-size` option.

    Example:

    ```
    $ oci ce node-pool delete-node --node-pool-id ocid1.nodepool.unique_ID \
    --node-id ocid1.instance.unique_ID --is-decrement-size true --force
    ```

Deleting an OKE Node Pool

Deleting a node pool permanently deletes the node pool. You cannot recover a deleted node pool.

Using the Compute Web UI

1. On the dashboard, click Containers / View Kubernetes Clusters (OKE).
2. Click the name of the cluster that contains the node pool that you want to delete.
3. On the cluster details page, scroll to the Resources section, and click Node Pools.
4. For the node pool that you want to delete, click the Actions menu, and click Delete.
5. Confirm the deletion.
    a. Enter the name of the node pool to confirm that you want to delete the node pool.
    b. Click the Delete button on the dialog.

Using the OCI CLI

1. Get the OCID of the node pool that you want to delete: `oci ce node-pool list`
2. Run the delete node pool command.

   Example:

   ```
   $ oci ce node-pool delete --node-pool-id ocid1.nodepool.unique_ID --force
   ```

6

Exposing Containerized Applications

To expose an application deployment so that worker node applications can be reached from outside the Private Cloud Appliance, create an external load balancer. An external load balancer is a Service of type LoadBalancer. The service provides load balancing for an application that has multiple running instances.

Ensure that the load balancer shape parameter has one of the following values: either `400Mbps` or `flexible`. If you specify `flexible` then you must also provide `flex-min` and `flex-max` annotations. You might need to edit the application deployment file to modify the load balancer shape value. See Specifying Alternative Load Balancer Shapes and Specifying Flexible Load Balancer Shapes for more information and examples of how to set these values.

Use the following command to create the external load balancer:

```
# kubectl create -f expose_lb
```

The following is the content of the `expose_lb` file:

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx-svc
  labels:
    app: nginx
  annotations:
    oci.oraclecloud.com/load-balancer-type: "lb"
    service.beta.kubernetes.io/oci-load-balancer-shape: "400Mbps"
spec:
  type: LoadBalancer
  ports:
   - port: 80
  selector:
    app: nginx
```

The following command shows more information about this external load balancer. The LoadBalancer Ingress IP address is the IP address that is used to reach node applications from outside the Private Cloud Appliance. In the Compute Web UI, the LoadBalancer Ingress IP address is shown under the heading "IP Address" at the bottom of the first column on load balancer details page, followed by the label "(Public)."

```
# kubectl describe svc my-nginx-svc
Name:                     my-nginx-svc
Namespace:                default
Labels:                   app=nginx
Annotations:              oci.oraclecloud.com/load-balancer-type: lb
                          service.beta.kubernetes.io/oci-load-balancer-shape: 400Mbps
Selector:                 app=nginx
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       IP_address
IPs:                      IP_address
```

```
LoadBalancer Ingress:     Load_Balancer_IP_address
Port:                     <unset> 80/TCP
TargetPort:               80/TCP
NodePort:                 <unset> 32145/TCP
Endpoints:                IP_address:port, IP_address+1:port, IP_address+2:port
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                 Age    From                Message
  ----
  Normal  EnsuringLoadBalancer  7m48s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   6m40s  service-controller  Ensured load balancer
```

Use the following command to list IP addresses and ports for the external load balancer:

```
# kubectl get svc
NAME          TYPE          CLUSTER-IP  EXTERNAL-IP               PORT(S)
AGE
kubernetes    ClusterIP     IP_address  <none>                    443/TCP
6h17m
my-nginx-svc  LoadBalancer  IP_address  Load_Balancer_IP_address  80:32145/TCP
5h5m
```

7

Adding Storage for Containerized Applications

You can add persistent storage for use by applications on an OKE cluster node. Storage created in a container's root file system will be deleted when you delete the container. For more durable storage for containerized applications, configure persistent volumes to store data outside of containers.

A persistent volume (PV) is storage that enables your data to remain intact when the containers to which the storage is connected are terminated.

A PV is a resource in the cluster. A persistent volume claim (PVC) is a request for a PV resource. A PVC is a storage request that is met by binding the PVC to a PV. A PVC provides an abstraction layer to the underlying storage.

You can provision PVCs using either of the following methods:

- Attach volumes from the Private Cloud Appliance Block Volume service. The volumes are connected to clusters created by OKE using a CSI (Container Storage Interface) volume plugin deployed on the clusters. See Creating Persistent Block Volume Storage.
- Mount file systems in the Private Cloud Appliance File Storage service. The File Storage service file systems are mounted inside containers running on clusters created by OKE using a CSI volume plugin deployed on the clusters. See Creating Persistent File System Storage.

Creating Persistent Block Volume Storage

The Private Cloud Appliance Block Volume service provides persistent, durable, and high-performance block storage for your data. See the Block Volume Storage Overview chapter in the *Oracle Private Cloud Appliance Concepts Guide* and the Block Volume Storage chapter in the *Oracle Private Cloud Appliance User Guide* for information about block volumes on the Private Cloud Appliance.

You can dynamically provision a block volume using the CSI plugin specified by the `oci-bv` storage class definition (`provisioner: blockvolume.csi.oraclecloud.com`). Then use the `kubectl` command to create the persistent volume claim.

1. Create a persistent volume claim, specifying the storage class name `oci-bv`.

   ```
   $ kubectl create -f csi-bvs-pvc.yaml
   ```

   The following is the content of the `csi-bvs-pvc.yaml` file:

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: mynginxclaim
   spec:
     storageClassName: "oci-bv"
     accessModes:
       - ReadWriteOnce
     resources:
   ```

```
    requests:
      storage: 50Gi
```

The requested `oci-bv` storage class is automatically created; you do not need to create it.

The persistent volume claim name in the `metadata` section is user-specified. You can have more than one persistent volume claim on a persistent volume.

For the value of `accessModes`, specify `ReadWriteOnce`; do not use `ReadWriteMany`.

The value of the `storage` property must be at least 50 gigabytes.

2. Run the following command to verify that the PVC has been created:

```
$ kubectl get pvc
NAME           STATUS   VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
mynginxclaim   Pending                                    oci-bv         4m
```

The PVC has a status of Pending because the `oci-bv` storage class definition includes the following:

```
volumeBindingMode: WaitForFirstConsumer
```

3. Use the PVC when creating other objects, such as pods.

For example, you could create a new pod from the following pod definition, which instructs the system to use the `mynginxclaim` PVC as the `nginx` volume, which is mounted by the pod at `/data`:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:latest
      ports:
        - name: http
          containerPort: 80
      volumeMounts:
        - name: data
          mountPath: /usr/share/nginx/html
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: mynginxclaim
```

Run the following command to verify that the PVC has been bound to a new PV:

```
$ kubectl get pvc
NAME           STATUS   VOLUME           CAPACITY   ACCESSMODES
STORAGECLASS   AGE
mynginxclaim   Bound    csi-unique_ID    50Gi       RWO            oci-bv
```

Run the following command to verify that the pod is using the new PVC:

```
$ kubectl describe pod nginx
```

Creating Persistent File System Storage

The Private Cloud Appliance File Storage service provides a durable, scalable, distributed, enterprise-grade network file system. See "Creating a File System, Mount Target, and Export" in the File System Storage chapter in the *Oracle Private Cloud Appliance User Guide* to create a mount target, file system, and file system export on the Private Cloud Appliance. Then use the `kubectl` command to create the storage class, persistent volume, and persistent volume claim.

1. Create a mount target.

   Important:

   To ensure that the mount target can be reached from worker nodes, create the mount target on the subnet that has configuration like the "worker" subnet described in Creating an OKE Worker Subnet. Ensure that TCP port 2049 to the NFS server is open on that subnet.

   If you do not create the mount target on the worker subnet, you might need to set security rules to ensure that the worker nodes can reach the mount target.

   See "Creating a Mount Target" in the File System Storage chapter in the *Oracle Private Cloud Appliance User Guide*.

   Note the export set OCID and mount target OCID. The export set OCID is required to create the file system export, and the mount target OCID is required to create the storage class. See Steps 3 and 4.

   You can have only one mount target per VCN.

2. Create a file system.

   See "Creating a File System" in the File System Storage chapter in the *Oracle Private Cloud Appliance User Guide*.

   You can create only one file system per VCN. You can have multiple storage classes, persistent volumes, and persistent volume claims per cluster, and they all share one NFS.

3. Create a file system export to associate the mount target with the file system.

   See "Creating an Export for a File System" in the File System Storage chapter in the *Oracle Private Cloud Appliance User Guide*.

   - Specify the export set OCID from the output from creating the mount target.
   - Specify the longest CIDR (smallest network) in the CIDR range that you specified when you created the "worker" subnet as described in Creating an OKE Worker Subnet.

   Note the export path and the mount target IP address.

4. Create a storage class, specifying the mount target OCID from the output of the create mount target step.

   ```
   $ kubectl create -f sc.yaml
   ```

   The following is the content of the `sc.yaml` file:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: pca-fss
provisioner: fss.csi.oraclecloud.com
parameters:
  mntTargetId: ocid1.mounttarget.unique_ID
```

The values of the `apiVersion` and `provisioner` properties are standard. The value of the storage class name in the metadata section is user-specified. You can create more than one storage class per mount target, and the storage class name is used in the following steps to create a persistent volume and persistent volume claim.

Use the `get sc` subcommand to view information about the new storage class:

```
$ kubectl get sc
```

5. Create a persistent volume, specifying the storage class name, the export path, and the mount target IP address.

   The storage class name is in the metadata in the `sc.yaml` file in the preceding step. The export path and the mount target IP address are output from the create file system export step. See Step 3 above.

```
$ kubectl create -f pv.yaml
```

   The following is the content of the `pv.yaml` file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: fss-pv
spec:
  storageClassName: pca-fss
  capacity:
    storage: 200Gi
  accessModes:
    - ReadWriteMany
  mountOptions:
    - nosuid
  nfs:
    server: mount_target_IP_address
    path: "/export/unique_ID"
    readOnly: false
```

   The persistent volume name in the `metadata` section is user-specified. You can have more than one persistent volume in a storage class.

   In the `nfs` section, the `server` value is the mount target IP address, and the `path` value is the export path.

   Use the `get pv` subcommand to view information about the new persistent volume:

```
$ kubectl get pv
NAME    CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON  AGE
fss-pv  200Gi     RWX           Retain          Bound   default/fss-pvc  pca-
fss                 20h
```

6. Create a persistent volume claim, specifying the persistent volume name and the storage class name.

   The persistent volume name and storage class name are in the output of the `get pv` command.

   Wait for the PVC status to be Bound before using this storage.

   ```
   kubectl create -f pvc.yaml
   ```

   The following is the content of the `pvc.yaml` file:

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: fss-pvc
   spec:
     storageClassName: pca-fss
     accessModes:
       - ReadWriteMany
     resources:
       requests:
         storage: 200Gi
     volumeName: fss-pv
   ```

   The persistent volume claim name in the `metadata` section is user-specified. You can have more than one persistent volume claim on a persistent volume.

   The value of the `accessModes` property must be `ReadWriteMany`.

   The value of the `storage` property must be at least 50 gigabytes.

   Run the following command to view information about the new persistent volume claim:

   ```
   $ kubectl get pvc
   NAME      STATUS   VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
   fss-pvc   Bound    fss-pv   200Gi      RWX           pca-fss        2h
   ```

7. Use the PVC when creating other objects, such as pods.

   For example, you could create a new pod from the following pod definition, which instructs the system to use the `fss-pvc` PVC as the `nginx` volume, which is mounted by the pod at `/persistent-storage`:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     name: fss-dynamic-app
   spec:
     containers:
       - name: nginx
         image: nginx:latest
         ports:
           - name: http
             containerPort: 80
         volumeMounts:
           - name: persistent-storage
             mountPath: /usr/share/nginx/html
     volumes:
     - name: persistent-storage
       persistentVolumeClaim:
         claimName: fss-pvc
   ```

Run the following command to verify that the pod is using the new PVC:

```
$ kubectl describe pod nginx
```

Using a Persistent Volume

To use this persistent storage, create a Kubernetes Deployment and assign a persistent volume claim.

Using File System Storage

The following example uses file system storage:

```
$ kubectl create -f nginx-deploy.yaml
```

The following is the content of the `nginx-deploy.yaml` file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-fss-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-fss
  template:
    metadata:
      labels:
        app: nginx-fss
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        volumeMounts:
        - mountPath: /usr/share/nginx/
          name: data
        ports:
        - containerPort: 80
          name: http
          protocol: TCP
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: fss-pvc
```

Using Block Volume Storage

The following example uses block volume storage:

```
$ kubectl create -f nginx-deploy.yaml
```

The following is the content of the `nginx-deploy.yaml` file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-bv-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-bv
  template:
    metadata:
      labels:
        app: nginx-bv
    spec:
      containers:
      - name: nginx
        image: available_internal_registry/nginx:latest
        volumeMounts:
        - mountPath: /usr/share/nginx/
          name: data
        ports:
        - containerPort: 80
          name: http
          protocol: TCP
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: mynginxclaim
```

Verify the New Storage Asset

Use the `get pod` subcommand to show the names of the replicas in the pod:

```
$ kubectl get pod
```

Log in to the pod and use the Linux `df` command to show that the application replicas are using the `persistentVolumeClaim` storage. The Filesystem column in the `df` output shows the mount target IP address and the file system export path.

Deleting a Persistent Volume

This topic describes how to delete a PV, or retain a PV after all associated PVCs are deleted. To delete PVCs, see Deleting a Persistent Volume Claim.

For file system storage, the default behavior is to retain the PV when all associated PVCs are deleted.

For block volume storage, the default behavior is to delete the PV when all associated PVCs are deleted. You might prefer to retain the PV after all associated PVCs are deleted, for example if the volume contains critical data. See Retaining a Persistent Volume.

If a PV is retained, you can optionally delete the PV later.

Deleting a Persistent Volume Claim

To delete a PVC, first delete all pods that are using that PVC. If you attempt to delete the PVC while a pod is still using the PVC, the PVC will be stuck in Terminating state and will not be deleted. When all the pods that are using that PVC are deleted, the PVC will be deleted.

1. List all pods that are using the PVC.

   Ensure that you have JQ command line utilities installed to query JSON objects.

   Use the following command to list pods across all the namespaces that are associated with the PVC that you want to delete.

```
$ kubectl get pods --all-namespaces -o=json | jq -c '.items[] |
{name: .metadata.name, namespace: .metadata.namespace, claimName: .spec |
select(has("volumes")).volumes[] |
select(has("persistentVolumeClaim")).persistentVolumeClaim.claimName} |
select(.claimName != null)'

{"name":"pod1_name","namespace":"namespace1_name","claimName":"claim1_name"}
{"name":"pod2_name","namespace":"namespace1_name","claimName":"claim1_name"}
{"name":"pod3_name","namespace":"namespace2_name","claimName":"claim2_name"}
```

To list pods only in the current namespace, use the same command as the preceding command except omit the `--all-namespaces` option.

2. Delete all pods that are using the PVC.

   Use the pod names reported by the `kubectl get pods` command that are associated with the `claimName` that you want to delete.

   ```
   $ kubectl delete pod pod1_name pod2_name
   ```

3. Delete the PVC.

   ```
   $ kubectl delete pvc claim1_name
   ```

4. (Optional) Delete the PV.

   If the Persistent Volume Reclaim Policy is Delete, the PV is automatically deleted when all PVCs that are associated with this PV are deleted.

   To list all PVCs, use the `kubectl get pvc` command.

   If the Persistent Volume Reclaim Policy is Retain, you can use the following command to delete the PV:

   ```
   $ kubectl delete pv pv_name
   ```

Retaining a Persistent Volume

Rather than delete a PV, you might prefer to retain the PV after all associated PVCs are deleted, for example if the volume contains critical data. See Changing the Reclaim Policy of a Persistent Volume for instructions to change the reclaim policy of the PV so that the PV will be retained after all associated PVCs are deleted.

If the Persistent Volume Reclaim Policy is Delete, the PV is automatically deleted when all PVCs that are associated with this PV are deleted. To prevent this behavior, specify the Retain policy. With the Retain policy, the PV is not deleted but is released of its claim. See Recovering the Data from a Released Persistent Volume for instructions to recover the data.

If you decide you want to delete the PV even though it was retained, or you want to delete the PV after you have recovered the data, use the following command:

```
$ kubectl delete pv pv_name
```

Changing the Reclaim Policy of a Persistent Volume

1. List the PVs in the cluster.

   ```
   $ kubectl get pv
   NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
   STORAGECLASS   REASON   AGE
   fss-pv   200Gi      RWX            Delete           Bound    default/fss-pvc   pca-
   fss                 20h
   ```

2. Change the reclaim policy of the PV.

   ```
   $ kubectl patch pv fss-pv -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
   ```

3. Verify the reclaim policy change.

   The `RECLAIM POLICY` column should now say `Retain`.

   ```
   $ kubectl get pv
   ```

Recovering the Data from a Released Persistent Volume

The PV is not available for another claim after the PV has been released of its previous claim because the previous claimant's data is still on the volume. Recover the data and then re-create the PV using the same storage to make a new claim on that storage.

1. Delete the PV.

   ```
   $ kubectl delete pv pv_name
   ```

   The associated block volume or file system still exists after the PV is deleted.

2. Manually recover and clean up the data on the block volume or file system.
3. (Optional) Manually delete the block volume or file system.

   To reuse the same block volume or file system, create a new PV with the same storage asset definition.