Oracle Private Cloud Appliance

Kubernetes Engine

F81950-08
May 2025

Oracle Private Cloud Appliance Kubernetes Engine,

F81950-08

Contents

9 Adding Storage for Containerized Applications

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.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`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 Kubernetes Engine

Oracle Private Cloud Appliance Kubernetes Engine (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 1.30.3, 1.29.9, and 1.28.8.

Best practice is to keep your clusters upgraded so that they are always running versions of Kubernetes that are currently supported by OKE. Viewing a cluster tells you if a newer Kubernetes version is available for that cluster. See Updating an OKE Cluster.

Supported Versions of the OCI Terraform Provider

This guide provides example Terraform scripts to configure the network resources. To use these scripts, you must install both Terraform and the Oracle Cloud Infrastructure (OCI) Terraform provider.

In your `provider` block, specify the version of the OCI Terraform provider to install as at least v4.50.0 but no greater than v6.36.0:

```
provider "oci" {
    version          = ">= 4.50.0, <= 6.36.0"
...
}
```

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.
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 Private Cloud Appliance 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.

- Private Cloud Appliance Administrator Tasks
- Cluster Administrator Tasks
- Creating the OraclePCA-OKE.cluster_id Tag
- Creating OraclePCA Tags

Private Cloud Appliance Administrator Tasks

These prerequisite tasks must be performed by a Service Enclave administrator or by a Compute Enclave user that has authorization to create resources such as groups, policies, and tag namespaces.

Task	Description	Resources
Administration network	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.	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*
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.	Providing Platform Images in the *Oracle Private Cloud Appliance Administrator Guide*
OKE users group	These users groups have a policy that authorizes members to use OKE.	Creating an OKE Users Group
OraclePCA-OKE defined tag	This tag is required to create or update an OKE cluster or node pool. This tag is used to identify instances that need to be in a dynamic group.	Creating the OraclePCA-OKE.cluster_id Tag
OKE dynamic group	The dynamic group authorizes its member instances to manage OKE resources.	Creating a Cluster Dynamic Group

Task	Description	Resources
OraclePCA tags	These tags are used when creating a cluster.	Creating OraclePCA Tags
Certificate Authority bundle	After upgrade, patching, or any other outage, or if the automated Certificate Authority bundle update fails, you might need to update the CA bundle manually on the management node.	Updating the Certificate Authority Bundle

At least three free public IP addresses are required to use OKE on Private Cloud Appliance. Verify that free public IP addresses are available for the NAT gateway, the control plane load balancer, and the worker load balancer. For more information, see Creating OKE Network Resources.

In the Service Web UI, select PCAConfig > Network Environment > Public IPs > "Free Public IPs". In the Service CLI, enter the following command:

```
PCA-ADMIN> show networkConfig
"Free Public IPs"
```

Creating an OKE Users Group

OKE users groups have a policy that authorizes their members to use OKE. You need to create separate OKE users groups to authorize different users to use OKE in different compartments.

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 user group policy. The following is an example policy for an 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.

```
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 network-load-balancers in compartment compartment-name
allow group group-name to manage virtual-network-family in compartment compartment-name
```

See Managing Policies in the *Oracle Private Cloud Appliance User Guide*.

Creating a Cluster Dynamic Group

A dynamic group authorizes its 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 cluster nodes that have this tag are members of the dynamic group.

The following is an example policy for the dynamic group. In this example. `oke_dyn_grp` is the name of the dynamic group and `oke` is the name of the compartment where resources are created. Note that all policy statements are for the same compartment. If clusters in this group require access to resources in other compartments, change the policy accordingly. See Managing Policies in the *Oracle Private Cloud Appliance User Guide*.

```
allow dynamic-group oke_dyn_grp to manage file-family in compartment oke
allow dynamic-group oke_dyn_grp to manage volume-family in compartment oke
allow dynamic-group oke_dyn_grp to manage load-balancers in compartment oke
allow dynamic-group oke_dyn_grp to manage instance-family in compartment oke
allow dynamic-group oke_dyn_grp to manage virtual-network-family in compartment oke
allow dynamic-group oke_dyn_grp to use tag-namespaces in compartment oke
```

For information about the purpose of the `use tag-namespaces` policy, see Exposing Containerized Applications.

Using Terraform to Create a Dynamic Group

The following example shows how to use Terraform to create a dynamic group.

variables.tf

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

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

variable "compartment_name" {
    description = "compartment name"
    type        = string
    nullable    = false
}

variable "oke_dyn_grp" {
    description = "Dynamic group that needs to be created for instance principal"
    default = "oke-dyn-ip-grp"
}

variable "oke_policy_name" {
    description = "Policy set name for dynamic group"
    default = "oke-instance-principal-policy"
}
```

terraform.tfvars

```
# 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 name
compartment_name = "oke"

# Dynamic Group Name
oke_dyn_grp = "oke-dyn-ip-group"
```

```
# OKE Dynamic Group Policy Name
oke_policy_name = "oke-dyn-grp-policy"
```

provider.tf

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

main.tf

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = ">= 4.50.0, <= 6.36.0"
      # If necessary, you can pin a specific version here
      # version = "6.36.0"
    }
  }
  required_version = ">= 1.1"
}
```

oke-dyn-grp.tf

```
resource "oci_identity_dynamic_group" "oke-dynamic-grp" {
    compartment_id = "${var.tenancy_ocid}"
    description    = "PCA OKE worker dynamic group for instance principal"
    matching_rule  = "tag.${oci_identity_tag_namespace.oracle-pca.name}.$
{oci_identity_tag.cluster-id.name}.value"
    name           = "${var.oke_dyn_grp}"
    depends_on = [oci_identity_tag.cluster-id]
}
```

oke-policy.tf

```
resource "oci_identity_policy" "oke-dyn-grp-policy" {
    compartment_id = "${var.tenancy_ocid}"
    description    = "Dynamic group policies for OKE Resources"
    name           = "${var.oke_policy_name}"
    statements     = [
        "allow dynamic-group ${oci_identity_dynamic_group.oke-dynamic-grp.name} to
manage load-balancers in compartment ${var.compartment_name}",
        "allow dynamic-group ${oci_identity_dynamic_group.oke-dynamic-grp.name} to
manage volume-family in compartment ${var.compartment_name}",
        "allow dynamic-group ${oci_identity_dynamic_group.oke-dynamic-grp.name} to
manage file-family in compartment ${var.compartment_name}",
        "allow dynamic-group ${oci_identity_dynamic_group.oke-dynamic-grp.name} to
manage instance-family in compartment ${var.compartment_name}",
        "allow dynamic-group ${oci_identity_dynamic_group.oke-dynamic-grp.name} to
manage virtual-network-family in compartment ${var.compartment_name}",
        "allow dynamic-group ${oci_identity_dynamic_group.oke-dynamic-grp.name} to use
tag-namespaces in compartment ${var.compartment_name}"
    ]

    depends_on = [oci_identity_dynamic_group.oke-dynamic-grp]
}
```

oke-tag-ns.tf

Create the OraclePCA-OKE.cluster_id tag, which is also described in Creating the OraclePCA-OKE.cluster_id Tag.

```
resource "oci_identity_tag" "cluster-id" {
    description      = "Default tag key definition"
    name             = "cluster_id"
    tag_namespace_id = "${oci_identity_tag_namespace.oracle-pca.id}"
    depends_on = [oci_identity_tag_namespace.oracle-pca]
}

resource "oci_identity_tag_namespace" "oracle-pca" {
    compartment_id = "${var.tenancy_ocid}"
    description    = "Default Tag namespace for Oracle PCA OKE"
    name           = "OraclePCA-OKE"
}
```

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*.

Cluster Administrator 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 work in more than one tenancy, create a profile for each tenancy as described in Using Multiple Profiles. If you already have OCI CLI installed, use `oci -v` to check the version. The minimum required version for using OKE is 3.48.0.

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 Private Cloud Appliance Compute Enclave or on your local system:

1. Create network resources: VCN, subnets, internet gateway, NAT gateway, route tables, and security lists. See Creating OKE Network Resources.
2. Create the OraclePCA-OKE.cluster_id defined tag.

   This tag is required to create or update an OKE cluster or node pool. This tag is used to identify instances that need to be in a dynamic group. See Creating the OraclePCA-OKE.cluster_id Tag.
3. Create the OraclePCA tag namespace and keys that are used when creating a cluster. See Creating OraclePCA Tags.
4. Create an OKE cluster. See Creating an OKE Cluster.
5. Create a Kubernetes configuration file for the cluster. See Creating a Kubernetes Configuration File.
6. 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.
7. Create a worker node pool. See Creating an OKE Worker Node Pool.
8. Configure any registries or repositories that the worker nodes need.
9. Create a service to expose containerized applications outside the Private Cloud Appliance. See Exposing Containerized Applications.
10. Create persistent storage for applications to use. See Adding Storage for Containerized Applications.

Creating the OraclePCA-OKE.cluster_id Tag

The OraclePCA-OKE.cluster_id tag is required to create or update an OKE cluster or node pool. 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.

The following sections describe how to create the OraclePCA-OKE tag namespace and the cluster_id tag key. You must create both the tag namespace and the tag key.

> **Important:**
>
> Do not delete this tag key definition. The tag namespace name must be exactly "OraclePCA-OKE", and the tag key name must be exactly "cluster_id".

Creating the OraclePCA-OKE Tag Namespace

Using the Compute Web UI

1. In the navigation menu, select Governance, and then select Tag Namespaces.

Ensure that the compartment that is selected in the compartment drop-down menu above the tag namespaces list is the compartment where you want to create the OraclePCA-OKE.cluster_id tag.

2. If OraclePCA-OKE is not shown in the Tag Namespaces list, select the Create Namespace Definitions button.
3. In the Create Namespace Definition dialog, scroll down to the Tagging section and click in the Tag Namespace field.
   - If the OraclePCA-OKE tag namespace is not listed, continue with Step 4 of this procedure.
   - If the OraclePCA-OKE tag namespace is already available, select the Cancel button in the Create Namespace Definition dialog. You will get an error message if you try to create this tag namespace when it is already available.
4. In the Create Namespace Definition dialog, enter the following information:
   - *Namespace Definition Name*: Enter "OraclePCA-OKE".
   - *Description*: For example, "Required to create or update an OKE cluster or node pool."
5. Select Create Namespace Definition.

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

Using theOCI CLI

Run the tag namespace create command. The `--compartment-id` value is the compartment where you want to create the OraclePCA-OKE.cluster_id tag. If this tag has already been created in a different compartment, then it is available to use in any other compartment, and you will receive an error message from this create tag namespace command.

```
$ oci iam tag-namespace create --compartment-id ocid1.compartment.unique_ID \
--name OraclePCA-OKE --description "Required to create or update an OKE cluster or node
pool."
{
  "data": {
    "compartment-id": "ocid1.compartment.unique_ID",
    "defined-tags": {
      "Oracle-Tags": {
        "CreatedBy": "okeuser",
        "CreatedOn": "2024-06-06T14:37:29.26Z"
      }
    },
    "description": "Required to create or update an OKE cluster or node pool.",
    "freeform-tags": {},
    "id": "ocid1.tag_namespace.unique_ID",
    "is-retired": false,
    "lifecycle-state": "ACTIVE",
    "locks": null,
    "name": "OraclePCA-OKE",
    "time-created": "2024-06-06T14:37:29.357678+00:00"
  },
  "etag": "a86bcf9b-f9a3-4891-b632-37d490161fe5"
}
```

Creating the cluster_id Tag Key

Using the Compute Web UI

1. Navigate to the OraclePCA-OKE tag namespace details page, scroll down to the Resources box, and select Tag Key Definitions.

2. If the cluster_id tag key is listed, select the name to display the details page. Ensure that Tag Value Type is String.
3. If the cluster_id tag key is not listed, select the Create Tag Key Definition button.
4. In the Create Tag Key Definition dialog, enter the following information:
   - *Name*: Enter "cluster_id".
   - *Description*: For example, "Required to create or update an OKE cluster or node pool."
   - *Tag Value Type*: Select "Static Value".
5. Select Create Tag Key Definition.

Using theOCI CLI

1. Check whether the cluster_id tag already exists in the OraclePCA-OKE tag namespace.

   ```
   $ oci iam tag list --tag-namespace-id ocid1.tag_namespace.unique_ID
   ```

2. If the cluster_id tag key is listed, view the details of the tag to confirm that `validator-type` is `DEFAULT`.

   ```
   $ oci iam tag get --tag-namespace-id ocid1.tag_namespace.unique_ID --tag-name
   cluster_id
   ```

3. If the cluster_id tag key is not listed, run the tag create command.

   You do not need to specify the `--validator` option because you want the default value.

   ```
   $ oci iam tag create --tag-namespace-id ocid1.tag_namespace.unique_ID \
   --name "cluster_id" --description "Required to create or update an OKE cluster or
   node pool." \
   --validator '{"validatorType": "DEFAULT"}'
   {
     "data": {
       "compartment-id": "unique_ID",
       "defined-tags": {
         "Oracle-Tags": {
           "CreatedBy": "okeuser",
           "CreatedOn": "2024-06-06T21:36:51.38Z"
         }
       },
       "description": "Required to create or update an OKE cluster or node pool.",
       "freeform-tags": {},
       "id": "ocid1.tag.unique_ID",
       "is-cost-tracking": false,
       "is-retired": false,
       "lifecycle-state": "ACTIVE",
       "name": "cluster_id",
       "tag-namespace-id": "ocid1.tag_namespace.unique_ID",
       "tag-namespace-name": "OraclePCA-OKE",
       "time-created": "2024-06-06T21:36:51.456538+00:00",
       "validator": {
         "validator-type": "DEFAULT"
       }
     },
     "etag": "5bf59d9a-5998-4857-a590-fca2a3386cc2"
   }
   ```

Creating OraclePCA Tags

Oracle Private Cloud Appliance uses the OraclePCA tag namespace to set attributes that are not available as OCI CLI options or API attributes. For OKE, some cluster attributes must be set by using OraclePCA tags.

Other attributes that can only be set by using OraclePCA tags, such as some block volume and file system attributes, are documented in the Oracle Private Cloud Appliance User Guide. You might want to set some of these for nodes in a node pool.

When you use the OCI CLI or API, you can specify the OraclePCA tag namespace, tag key, and values for the attributes that you want to set. You do not need to first create the OraclePCA tag namespace and tag keys.

To use the Compute Web UI to set these attributes, you must first create the OraclePCA tag namespace, tag keys, and value choices.

> **Caution:**
>
> Do not delete these tag keys. Do not create this tag namespace and these keys unless you need to use the Compute Web UI to create clusters. If you create this tag namespace and these keys, create them exactly as shown here, do not modify them, and do not delete them.

The following sections describe how to create the OraclePCA tag namespace, and how to create the tag key definitions for the OKE attributes.

Creating the OraclePCA Tag Namespace

1. In the navigation menu, select Governance, and then select Tag Namespaces.

   Ensure that the compartment that is selected in the compartment drop-down menu above the tag namespaces list is the compartment where you want to create the OraclePCA tag namespace.

2. If OraclePCA is not shown in the Tag Namespaces list, select the Create Namespace Definitions button.

3. In the Create Namespace Definition dialog, scroll down to the Tagging section and click in the Tag Namespace field.

   - If the OraclePCA tag namespace is not listed, continue with Step 4 of this procedure.
   - If the tag you need is already available, select the Cancel button in the Create Namespace Definition dialog. You will get an error message if you try to create this tag namespace when it is already available.

4. In the Create Namespace Definition window, enter the following information:

   - *Namespace Definition Name*: Enter "OraclePCA".
   - *Description*: For example, "Support block volume, cluster, and file system parameters that are only available on Private Cloud Appliance."

5. Select Create Namespace Definition.

   The details page for the new OraclePCA tag namespace definition is displayed.

Creating the OraclePCA Tag Key Definitions

1. Navigate to the OraclePCA tag namespace details page, scroll down to the Resources box, and select Tag Key Definitions.
2. If the tag key you need is not listed, select the Create Tag Key Definition button.
3. In the Create Tag Key Definition dialog, enter the following information for the tag key that you are creating.

   SSH Key

   - *Name*: Enter "sshkey".
   - *Description*: For example, "Your public SSH key."
   - *Tag Value Type*: Select "Static Value".

   Number of Control Plane Nodes

   - *Name*: Enter "cpNodeCount".
   - *Description*: For example, "Number of nodes in the control plane."
   - *Tag Value Type*: Select "A List of Values".
   - *Values*: Enter "1", newline, "3", newline, and "5".

   Shape of Control Plane Nodes

   - *Name*: Enter "cpNodeShape".
   - *Description*: For example, "The shape of the control plane nodes."
   - *Tag Value Type*: Select "Static Value".

   Shape Configuration of Control Plane Nodes

   - *Name*: Enter "cpNodeShapeConfig".
   - *Description*: For example, "The number of OCPUs and optionally amount of memory for a flexible node shape."
   - *Tag Value Type*: Select "Static Value".
4. Select Create Tag Key Definition.

3
OKE Best Practices

Use the best practices described in this topic to get the most from your Kubernetes Engine clusters.

Cluster Management Best Practices

Upgrade Clusters

Keep your clusters upgraded so that they are always running versions of Kubernetes that are listed as currently supported by OKE. Viewing a cluster tells you if a newer Kubernetes version is available for that cluster. See Supported Versions of Kubernetes and Updating an OKE Cluster.

Use Kubernetes labels.

Use Kubernetes labels to organize the many Kubernetes resources (such as services, pods, containers, and networks) that comprise a cluster.

Kubernetes labels are key-value pairs that help you to maintain these resources and keep track of how they interact with each other in a cluster.

Use resource tagging.

Use resource tagging to organize the many resources (such as worker nodes, VCNs, load balancers, and block volumes) used by the Kubernetes clusters you create with Kubernetes Engine.

When a large number of resources is spread across multiple compartments in a tenancy, it can be challenging to track the resources that are used for specific purposes. It can also be challenging to aggregate the resources, report on them, and take bulk actions on them.

Tagging enables you to define keys and values, and associate those tags with resources. You can then use the tags to organize and list resources based on your business needs.

For more information, see the Resource Tag Management chapter in the *Oracle Private Cloud Appliance User Guide*.

Set resource requests and limits.

- Set resource requests to specify the minimum amount of resources a container can use.
- Set resource limits to specify the maximum amount of resources a container can use.

Sometimes an application fails to deploy on a Kubernetes cluster due to limited availability of resources on that cluster. The failure of the application to deploy can be avoided by correctly setting resource requests and resource limits.

If you do not set resource requests and limits, pods in a cluster can start utilizing more resources than necessary. If a pod starts consuming more CPU or memory on a node, then the Kubernetes scheduler (`kube-scheduler`) might not be able to place new pods on the node, and the node might even crash.

For more information, see Resource Management for Pods and Containers on the `kubernetes.io` site.

Provide dedicated nodes by using taints and tolerations.

Use Kubernetes taints and tolerations to limit resource-intensive applications to specific worker nodes.

Using taints and tolerations enables you to keep node resources available for workloads that require them, and prevents the scheduling of other workloads on the nodes.

For more information, see Taints and Tolerations on the `kubernetes.io` site.

Control pod scheduling by using node selectors and affinity.

Several different methods are available to constrain a pod to run on particular nodes, or to specify a preference for a pod to run on particular nodes. The recommended approaches all use label selectors to specify the node selection.

Often, the `kube-scheduler` makes a reasonable placement when constraints and preferences are not specified. However, there are some circumstances where you might want to control the node on which a pod runs. In these situations, best practice is to control the scheduling of pods on nodes using Kubernetes node selectors, node affinity, and inter-pod affinity.

Using node selectors, node affinity, and inter-pod affinity enables the `kube-scheduler` to logically isolate workloads, such as according to the node's hardware.

Use third-party tools for backup and disaster recovery.

Use third-party tools such as Velero with Kubernetes Engine for backup and disaster recovery.

The combined backup and disaster recovery capabilities of these tools and Kubernetes Engine can provide a reliable, robust, and scalable Kubernetes platform that is production-ready.

Networking Best Practices

Create separate compartments for each team.

If you expect multiple teams to create clusters, create a separate compartment for each team.

Size your VCN appropriately.

Allow for possible future cluster and node pool scaling requirements when sizing the VCN in which you want to create and deploy Kubernetes clusters.

Ensure that the VCN has a CIDR block that is large enough to allocate network addresses to all the resources that a cluster requires: subnets, Kubernetes API endpoint, worker nodes, pods, load balancers.

Select the pod networking CNI plugin that best suits your needs.

Consider pod networking requirements carefully, and then select the pod networking CNI plugin that best suits your needs.

- If applications require the use of base networking requirements (and not the use of IP addresses from the VCN), or require a high density of pods per worker node, best practice is to use the Flannel Overlay CNI plugin. See Creating Flannel Overlay Network Resources.
- If applications require pods to have an IP address from the VCN CIDR, or require the consistent network performance offered by virtual machines (regardless of the nodes on which the pods are running) with no additional overlay, best practice is to use the OCI VCN-Native Pod Networking CNI plugin. See Creating VCN-Native Pod Networking Resources.

Configure externalTrafficPolicy appropriately when exposing applications.

Carefully consider the most appropriate value for the `externalTrafficPolicy` setting when provisioning a network load balancer for a Kubernetes service of type LoadBalancer.

Avoid overlapping pod and service CIDR blocks with an on-premise CIDR block and when using the Flannel Overlay CNI plugin.

Avoid the situation where the CIDR block used by the Flannel Overlay network to provision pods and services with IP addresses overlaps with a CIDR block used to provision external compute instances with IP addresses.

Kubernetes clusters require a unique IP address for every pod. Therefore, IP address planning is necessary because addresses cannot overlap with the private IP address space used on-premises or in other connected VCNs.

Plan the number of nodes you will need.

Create a plan for the number of nodes in a cluster that takes into account node size, the application profile of pods, and the selected pod networking CNI plugin.

Use separate subnets and security rules.

Use separate subnets and security rules when configuring network resources. The VCN in which you want to create and deploy clusters must have at least two different subnets, and can have more:

- A Kubernetes API endpoint subnet
- A worker nodes subnet
- One regional, or two AD-specific, load balancer subnets (optional)
- A pods subnet (when using the OCI VCN-Native Pod Networking CNI plugin)
- A bastion subnet (optional)

You can choose to combine the subnets, and also to combine security rules. However, this approach makes security harder to manage and is therefore not recommended unless you are using network security groups to control access to clusters.

Security Best Practices

Plan exposure level.

Answer the following questions before implementing a security plan for the clusters you create with Kubernetes Engine:

- How much internet exposure do you want clusters to have?
- How do you plan to expose workloads internally to your VCN, and externally to the internet?
- How do you plan to scale workloads?
- Which types of Oracle services will the cluster consume?

Create private clusters.

If your cluster does not require direct access from the internet, create a private cluster. In a private cluster, the Kubernetes API server and worker nodes are assigned only private IP addresses.

Optionally use a NAT gateway for outbound internet access, a Dynamic Routing Gateway (DRG) to enable access from the on-premises network, a Local Peering Gateway (LPG) to allow access from other VCNs.

Place all applications in private subnets.

If the applications running on worker nodes do not requiredirect access to the internet, both the worker nodes subnet and the worker load balancer subnet should be private.

Restrict cluster traffic using Network Security Groups.

Define security rules in network security groups (NSGs), rather than in security lists, for the VCN in which you want to create and deploy clusters. See "Controlling Traffic with Network Security Groups" in "Configuring VCN Rules and Options" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*.

General security best practices.

- Apply security patches regularly.
- Use a combination of Kubernetes network policies and NSGs.
- Use NSGs in conjunction with infrastructure-as-code tools (such as Terraform).
- Rotate secrets and certificates regularly.
- Run all applications as a non-privileged user.
- Treat containers as immutable.

Auditing, logging, and monitoring.

- Check logs regularly.
- Enable audit logging.
- Use Kubernetes cluster-based logging.
- Monitor cluster components.
- Log network traffic metadata and analyze it regularly.
- Use small and secure container images.
- Limit credential exposure.

Storage Best Practices

- Choose the appropriate storage type.
- Create and use storage classes to define application needs.
- Create and use volumes for persistent storage.
- Limit storage resource consumption.
- Secure and back up data.

Upgrade Best Practices

- Use the latest supported version of Kubernetes.
- Set up test and production environments.
- Cordon and drain worker nodes in preparation for maintenance.
- Treat worker nodes as immutable.

4

Creating OKE 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 Flannel Overlay Networking and Workload Cluster Network Ports for VCN-Native Pod Networking for specific ports that must be open for specific purposes.

> **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*.

This chapter describes how to create network resources for two networking types:

- Creating Flannel Overlay Network Resources
- Creating VCN-Native Pod Networking Resources

Public and Private Clusters summarizes which network resources you need to create a public cluster and which network resources you need to create a private cluster.

Pod Networking

The Kubernetes networking model assumes containers (pods) have unique and routable IP addresses within a cluster. In the Kubernetes networking model, pods use those IP addresses to communicate with other pods on the same node in a cluster or on a different node, with pods on other clusters, with the cluster's control plane nodes, with other services (such as storage services), and with the internet.

By default, pods accept traffic from any source. To enhance cluster security, control access to and from pods using security rules defined as part of network security groups (recommended) or security lists. The security rules apply to all pods in all the worker nodes connected to the pod subnet specified for a node pool. See Controlling Traffic with Network Security Groups and Controlling Traffic with Security Lists in the *Oracle Private Cloud Appliance User Guide*.

Public and Private Clusters

Before you create a cluster, decide what kind of network access the cluster requires: whether you need a public cluster or a private cluster. You cannot create both public and private clusters in one VCN.

The key difference between a public cluster and a private cluster is whether you configure public or private subnets for the Kubernetes API endpoint and the worker load balancer.

> **Note:**
>
> The subnets for the worker nodes and control plane nodes are always private.

For the worker nodes and control plane nodes, you can configure route rules that allow access only within the VCN or outside the VCN. This documentation names those route tables "vcn_private" and "nat_private." You can choose either of these private subnet configurations for your worker nodes and control plane nodes whether the cluster is private or the cluster is public.

Public Clusters

A public cluster requires the following network resources:

- A public subnet for the Kubernetes API endpoint. See the instructions for creating a public "control-plane-endpoint" subnet in Creating a Flannel Overlay Control Plane Load Balancer Subnet and Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet.
- A public subnet for the worker load balancer. See the instructions for creating a public "service-lb" subnet in Creating a Flannel Overlay Worker Load Balancer Subnet and Creating a VCN-Native Pod Networking Worker Load Balancer Subnet.
- An internet gateway to connect resources on a public subnet to the internet using public IP addresses.
- A NAT gateway. Use a NAT gateway for outbound internet access. A NAT gateway connects resources on a private subnet to the internet without exposing private IP addresses.
- At least three free public IP addresses. Free public IP addresses are required for the NAT gateway, control plane load balancer, and worker load balancer.

  The worker load balancer requires a free public IP address to expose applications. The worker load balancer might require more free public IP addresses depending on the applications running on the pods. For how to display the list of free public IP addresses on the appliance, see Private Cloud Appliance Administrator Tasks.

Private Clusters

If you create multiple OKE VCNs, each CIDR must be unique. CIDRs of different VCNs for private clusters cannot overlap with any other VCN CIDRs or any on-premises CIDR. The IP addresses used must be exclusive to each VCN.

A private cluster has the following network resources:

- A private subnet for the Kubernetes API endpoint. See the instructions for creating a private "control-plane-endpoint" subnet in Creating a Flannel Overlay Control Plane Load Balancer Subnet and Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet.
- A private subnet for the worker load balancer. See the instructions for creating a private "service-lb" subnet in Creating a Flannel Overlay Worker Load Balancer Subnet and Creating a VCN-Native Pod Networking Worker Load Balancer Subnet.
- A route table with no route rules. This route table allows access only within the VCN.
- (Optional) A Local Peering Gateway (LPG). Use an LPG to allow access from other VCNs. An LPG allows access to the cluster from an instance running on a different VCN. Create an LPG on the OKE VCN, and create an LPG on a second VCN on the Private Cloud

Appliance. Use the LPG connect command to peer the two LPGs. Peered VCNs can be in different tenancies. CIDRs for the peered VCNs cannot overlap. See "Connecting VCNs through a Local Peering Gateway" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*.

Create a route rule to steer VCN subnet traffic to and from the LPGs, and security rules to allow or deny certain types of traffic. See Creating a Flannel Overlay VCN or Creating a VCN-Native Pod Networking VCN for the route table to add to the OKE VCN and similar route table to add to the second VCN. Add the same route rule on the second VCN, specifying the OKE VCN CIDR as the destination.

Install the OCI SDK and `kubectl` on the instance on the second VCN and connect to the private cluster. See Creating a Kubernetes Configuration File.

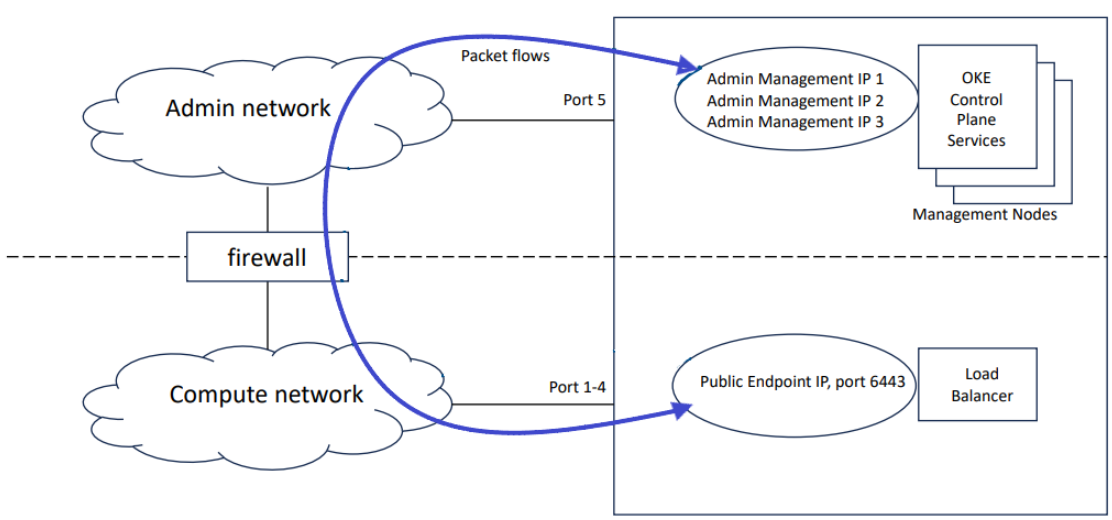
- (Optional) A Dynamic Routing Gateway (DRG). Use a DRG to enable access from the on-premises network. A DRG allows traffic between the OKE VCN and the on-premises network's IP address space. Create the DRG in the OKE VCN compartment, and then attach the OKE VCN to that DRG. See "Connecting to the On-Premises Network through a Dynamic Routing Gateway" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*.

  Create a route rule to steer traffic to the on-premises data center network's IP address space. See Creating a Flannel Overlay VCN or Creating a VCN-Native Pod Networking VCN for the route table to add to the OKE VCN.

OKE Cluster Management with Administration Network

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

Figure 4-1 Example of System Configured with a Separate Administration Network



The OKE service 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 service 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.

See Workload Cluster Network Ports for Flannel Overlay Networking and Workload Cluster Network Ports for VCN-Native Pod Networking for how to configure ports for OKE. If you are using a separate administration network, see also the table Access Configuration With Administration Network in Port Matrix in the *Oracle Private Cloud Appliance Security Guide*.

Creating Flannel Overlay Network Resources

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.

The resource definitions in the following sections in this topic create a working example set of network resources for workload clusters when you are using Flannel Overlay networking. 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 Flannel Overlay Networking for specific ports that must be open for specific purposes.

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

> **Note:**
>
> Create all of these network resources in the same compartment on the appliance.

- VCN. See Creating a Flannel Overlay VCN.
- Internet Gateway
- NAT Gateway
- Dynamic Routing Gateway
- Local Peering Gateway
- Route rules
- Security lists
- The following four subnets:
  - Worker. See Creating a Flannel Overlay Worker Subnet.
  - Worker load balancer. See Creating a Flannel Overlay Worker Load Balancer Subnet.
  - Control plane. See Creating a Flannel Overlay Control Plane Subnet.
  - Control plane load balancer. See Creating a Flannel Overlay Control Plane Load Balancer Subnet.

Workload Cluster Network CIDR Ranges for Flannel Overlay Networking

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 for use with Flannel Overlay networking.

> **Note:**
>
> These are examples only. The CIDR ranges you use depend on the number of clusters you have, the number of nodes in each cluster, and the type of networking you are using.

For Flannel Overlay networking, IP addresses are managed by the underlying Container service. Pods are not assigned IP addresses from the IP address pool that is defined in the pod subnet CIDR. This is the reason you do not need a pod subnet when you are using Flannel Overlay networking.

The primary difference between IP address requirements of Flannel Overlay networking and VCN-Native Pod Networking is that VCN-Native Pod Networking requires more IP addresses to be available. The table in Workload Cluster Network CIDR Ranges for VCN-Native Pod Networking shows larger CIDR ranges than the following table for Flannel Overlay CIDR ranges. The CIDR ranges used with Flannel Overlay networking can be much smaller than the CIDR ranges used with VCN-Native Pod Networking.

Table 4-1 Example CIDR Values to Use with Flannel Overlay Networking

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

The IP Subnet Calculator on Calculator.net is one tool for finding all available networks for a given IP address and prefix length.

Workload Cluster Network Ports for Flannel Overlay Networking

The following table lists ports that are used by workload clusters when you use Flannel Overlay networking. 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*.

If you are using a separate administration network, see OKE Cluster Management with Administration Network.

Table 4-2 Ports that Must Be Available for Use by Workload Clusters for Flannel Overlay Networking

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.

Example Terraform Scripts for Flannel Overlay Network Resources

The following Terraform scripts create the network resources that are required by OKE when you are using Flannel Overlay networking. Subsequent sections in this topic 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 for Flannel Overlay Networking), and the OKE control plane subnet must be named `control-plane`. See Workload Cluster Network CIDR Ranges for Flannel Overlay Networking 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 when you are using Flannel Overlay networking. 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 for Flannel Overlay Networking. 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"
}

# Flag to Enable private endpoint
variable "enable_private_endpoint" {
  description = "Flag to create private control plane endpoint/service-lb"
  type = bool
  default = false
  nullable = false
}
```

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.

The version of the OCI provider that you use must be at least v4.50.0 but no greater than v6.36.0.

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = ">= 4.50.0, <= 6.36.0"
```

```
      # If necessary, you can pin a specific version here
      # version = "6.36.0"
    }
  }
  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
    },
  ]])

  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
      },
    ]
  }
```

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" "default_private" {
  manage_default_resource_id = oci_core_vcn.oke_vcn.default_route_table_id
  display_name               = "Default - private"
}

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
      }
    }
  }
}
```

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 = var.enable_private_endpoint
  route_table_id             = var.enable_private_endpoint==false ?
oci_core_route_table.public[0].id : oci_core_vcn.oke_vcn.default_route_table_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
      }
    }
  }
}
```

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 = var.enable_private_endpoint
  route_table_id             = var.enable_private_endpoint==false ?
oci_core_route_table.public[0].id : oci_core_default_route_table.default_private[0].id
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmilb.id
  ]
}
```

Creating a Flannel Overlay VCN

Create the following resources in the order listed:

1. VCN
2. Route rules
   - Public clusters:
     – Internet gateway and a route table with a route rule that references that internet gateway.
     – NAT gateway and a route table with a route rule that references that NAT gateway.
   - Private clusters:
     – Route table with no route rules.
     – (Optional) Dynamic Routing Gateway (DRG) and a route table with a route rule that references that DRG. See Private Clusters.
     – (Optional) Local Peering Gateway (LPG) and a route table with a route rule that references that LPG. See Private Clusters.
3. Security list. 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 Flannel Overlay 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

- Public internet access. For traffic on a public subnet that connects to the internet using public IP addresses, create an internet gateway and a route rule that references that internet gateway.
- Private internet access. For traffic on a private subnet that needs to connect to the internet without exposing private IP addresses, create a NAT gateway and a route rule that references that NAT gateway.
- VCN-only access. To restrict communication to only other resources on the same VCN, use the default route table, which has no route rules.
- Instances in another VCN. To enable communication between the cluster and an instance running on a different VCN, create a Local Peering Gateway (LPG) and a route rule that references that LPG.
- On-premises IP address space. To enable communication between the cluster and the on-premises network IP address space, create a Dynamic Routing Gateway (DRG), attach the OKE VCN to that DRG, and create a route rule that references that DRG.

VCN Private Route Table

Edit the default route table that was created when you created the VCN. Change the name of the route table to vcn_private. This route table does not have any route rules. Do not add any route rules.

NAT Private Route Table

Create a NAT gateway and a route table with a route rule that references 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 Flannel Overlay Network Resources.

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

Private 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 Flannel Overlay Network Resources.

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

Compute Web UI property	OCI CLI property
• Name: nat_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: NAT private route rule	• `--display-name`: nat_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`: NAT private route rule

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

Local Peering Gateway

Create a Local Peering gateway (LPG) and a route table with a route rule that references the LPG.

Local Peering Gateway

To create the LPG, use the instructions in "Connecting VCNs through a Local Peering Gateway" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*.

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

Private 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 this example, use the following input to create the route table with a private route rule that references the LPG that was created in the preceding step.

Compute Web UI property	OCI CLI property
• Name: lpg_rt Route rule • Target Type: Local Peering Gateway • Local Peering Gateway: Name of the LPG that was created in the preceding step • CIDR Block: ***`CIDR_for_the_second_VCN`*** • Description: LPG private route rule	• `--display-name`: lpg_rt `--route-rules` • `networkEntityId`: OCID of the LPG that was created in the preceding step • `destinationType`: `CIDR_BLOCK` • `destination`: ***`CIDR_for_the_second_VCN`*** • `description`: LPG private route rule

Note the name and OCID of this route table for assignment to the "control-plane-endpoint" subnet (Creating a Flannel Overlay Control Plane Load Balancer Subnet).

Add the same route rule on the second VCN (the peered VCN), specifying the OKE VCN CIDR as the destination.

Dynamic Routing Gateway

Create a Dynamic Routing gateway (DRG) and a route table with a route rule that references the DRG.

Dynamic Routing Gateway

To create the DRG and attach the OKE VCN to that DRG, use the instructions in "Connecting to the On-Premises Network through a Dynamic Routing Gateway" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*. Create the DRG in the OKE VCN compartment, and then attach the OKE VCN to that DRG.

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

Private 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 this example, use the following input to create the route table with a private route rule that references the DRG that was created in the preceding step.

Compute Web UI property	OCI CLI property
• Name: drg_rt Route rule • Target Type: Dynamic Routing Gateway • Dynamic Routing: Name of the DRG that was created in the preceding step • CIDR Block: 0.0.0.0/0 • Description: DRG private route rule	• `--display-name`: drg_rt `--route-rules` • `networkEntityId`: OCID of the DRG that was created in the preceding step • `destinationType`: `CIDR_BLOCK` • `destination`: `0.0.0.0/0` • `description`: DRG private route rule

Note the name and OCID of this route table for assignment to the "control-plane-endpoint" subnet (Creating a Flannel Overlay Control Plane Load Balancer Subnet).

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 Flannel Overlay 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 Flannel Overlay 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 Flannel Overlay Network Resources.

Compute Web UI property	OCI CLI property
• Name: Default	`--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."

Compute Web UI property	OCI CLI property

Ingress Rule 2
• Stateless: uncheck the box
• Ingress CIDR: 0.0.0.0/0
• IP Protocol: ICMP
  – Parameter Type: 3: Destination Unreachable
• Description: "Blocks incoming requests from any source."
	Ingress Rule 2
	• `isStateless`: `false`
	• `source`: `0.0.0.0/0`
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `1`
	• `icmpOptions`
	  – `type`: `3`
	• `description`: "Blocks incoming requests from any source."

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

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

Creating a Flannel Overlay 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 Flannel Overlay 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`

Compute Web UI property	OCI CLI property

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`."

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."

Compute Web UI property	OCI CLI property

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."

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 Flannel Overlay 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 a Flannel Overlay VCN. Create the worker subnet in the same compartment where you created the VCN.

Create either a NAT private worker subnet or a VCN private worker subnet. Create a NAT private worker subnet to communicate outside the VCN.

Table 4-3 Create a NAT Private Worker Subnet

Compute Web UI property	OCI CLI property
• Name: worker • CIDR Block: ***worker_cidr*** • Route Table: Select "nat_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 "nat_private" route table • `--security-list-ids`: OCIDs of the "worker-seclist" security list and the "Default Security List for oketest-vcn" security list

The difference in the following private subnet is the VCN private route table is used instead of the NAT private route table.

Table 4-4 Create a VCN Private Worker Subnet

Compute Web UI property	OCI CLI property
• Name: worker • CIDR Block: ***worker_cidr*** • Route Table: Select "vcn_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 "vcn_private" route table • `--security-list-ids`: OCIDs of the "worker-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating a Flannel Overlay 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 Flannel Overlay 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`

Compute Web UI property	OCI CLI property

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."

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 Flannel Overlay 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 a Flannel Overlay VCN. Create the worker load balancer subnet in the same compartment where you created the VCN.

Create either a private or a public worker load balancer subnet. Create a public worker load balancer subnet to use with a public cluster. Create a private worker load balancer subnet to expose applications in a private cluster.

Table 4-5 Create a Public Worker Load Balancer Subnet

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

The difference in the following private subnet is the VCN private route table is used instead of the public route table.

Table 4-6 Create a Private Worker Load Balancer Subnet

Compute Web UI property	OCI CLI property
• Name: service-lb • CIDR Block: ***workerlb_cidr*** • Route Table: Select "vcn_private" from the list • Private 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`: `true` • `--route-table-id`: OCID of the "vcn_private" route table • `--security-list-ids`: OCIDs of the "workerlb-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating a Flannel Overlay 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 Flannel Overlay Network Resources.

For this example, use the following input for the control plane subnet security list. The `kubernetes_api_port` is the port used to access the Kubernetes API: port 6443. See also Workload Cluster Network Ports for VCN-Native Pod Networking. See also Workload Cluster Network Ports for Flannel Overlay Networking.

<table>
<tr><th>Compute Web UI property</th><th>OCI CLI property</th></tr>
<tr><td>
Name: kmi-seclist
</td><td>
<code>--vcn-id</code>: <code>ocid1.vcn.oke_vcn_id</code>
<code>--display-name</code>: <code>kmi-seclist</code>
</td></tr>
<tr><td>
Six ingress security rules:
</td><td>
Six ingress security rules:
<code>--ingress-security-rules</code>
</td></tr>
<tr><td>
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."
</td><td>
Ingress Rule 1
<code>isStateless</code>: <code>false</code>
<code>source</code>: kube_client_cidr
<code>sourceType</code>: <code>CIDR_BLOCK</code>
<code>protocol</code>: <code>6</code>
<code>tcpOptions</code>
<code>destinationPortRange</code>
<code>max</code>: kubernetes_api_port
<code>min</code>: kubernetes_api_port
<code>description</code>: "Allow inbound connections to the Kubernetes API server."
</td></tr>
<tr><td>
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."
</td><td>
Ingress Rule 2
<code>isStateless</code>: <code>false</code>
<code>source</code>: kmilb_cidr
<code>sourceType</code>: <code>CIDR_BLOCK</code>
<code>protocol</code>: <code>6</code>
<code>tcpOptions</code>
<code>destinationPortRange</code>
<code>max</code>: kubernetes_api_port
<code>min</code>: kubernetes_api_port
<code>description</code>: "Allow inbound connections from the control plane load balancer."
</td></tr>
<tr><td>
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."
</td><td>
Ingress Rule 3
<code>isStateless</code>: <code>false</code>
<code>source</code>: worker_cidr
<code>sourceType</code>: <code>CIDR_BLOCK</code>
<code>protocol</code>: <code>6</code>
<code>tcpOptions</code>
<code>destinationPortRange</code>
<code>max</code>: <code>65535</code>
<code>min</code>: <code>1024</code>
<code>description</code>: "Allow inbound connections from worker nodes to the control plane."
</td></tr>
</table>

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."

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 Flannel Overlay Network Resources.

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

Create either a NAT private control plane subnet or a VCN private control plane subnet. Create a NAT private control plane subnet to communicate outside the VCN.

> **Important:**
>
> The name of this subnet must be exactly "control-plane".

Table 4-7 Create a NAT Private Control Plane Subnet

Compute Web UI property	OCI CLI property

• Name: control-plane
• CIDR Block: ***kmi_cidr***
• Route Table: Select "nat_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 "nat_private" route table
	• `--security-list-ids`: OCIDs of the "kmi-seclist" security list and the "Default Security List for oketest-vcn" security list

The difference in the following private subnet is the VCN private route table is used instead of the NAT private route table.

Table 4-8 Create a VCN Private Control Plane Subnet

Compute Web UI property	OCI CLI property
• Name: control-plane • CIDR Block: ***kmi_cidr*** • Route Table: Select "vcn_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 "vcn_private" route table • `--security-list-ids`: OCIDs of the "kmi-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating a Flannel Overlay 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 Flannel Overlay 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`

<table>
<tr><th>Compute Web UI property</th><th>OCI CLI property</th></tr>
<tr><td>Ingress Rule 1:
• Stateless: uncheck the box
• Ingress CIDR: <code>253.255.0.0/16</code>
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."</td><td>Ingress Rule 1:
• <code>isStateless</code>: <code>false</code>
• <code>source</code>: <code>253.255.0.0/16</code>
This value is required. Do not change this CIDR value.
• <code>sourceType</code>: <code>CIDR_BLOCK</code>
• <code>protocol</code>: <code>6</code>
• <code>tcpOptions</code>
<code>destinationPortRange</code>
– <code>max</code>: kubernetes_api_port
– <code>min</code>: kubernetes_api_port
• <code>description</code>: "Allow inbound connections to the control plane load balancer."</td></tr>
<tr><td>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."</td><td>Ingress Rule 2:
• <code>isStateless</code>: <code>false</code>
• <code>source</code>: kube_client_cidr
• <code>sourceType</code>: <code>CIDR_BLOCK</code>
• <code>protocol</code>: <code>6</code>
• <code>tcpOptions</code>
<code>destinationPortRange</code>
– <code>max</code>: kubernetes_api_port
– <code>min</code>: kubernetes_api_port
• <code>description</code>: "Allow inbound connections to the control plane load balancer."</td></tr>
<tr><td>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."</td><td>Ingress Rule 3:
• <code>isStateless</code>: <code>false</code>
• <code>source</code>: vcn_cidr
• <code>sourceType</code>: <code>CIDR_BLOCK</code>
• <code>protocol</code>: <code>6</code>
• <code>tcpOptions</code>
<code>destinationPortRange</code>
– <code>max</code>: kubernetes_api_port
– <code>min</code>: kubernetes_api_port
• <code>description</code>: "Allow inbound connections to the control plane load balancer."</td></tr>
</table>

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 Flannel Overlay 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 a Flannel Overlay VCN. Create the control plane load balancer subnet in the same compartment where you created the VCN.

Create either a private or a public control plane load balancer subnet. Create a public control plane load balancer subnet to use with a public cluster. Create a private control plane load balancer subnet to use with a private cluster.

See Private Clusters for information about using Local Peering Gateways to connect a private cluster to other instances on the Private Cloud Appliance and using Dynamic Routing Gateways to connect a private cluster to the on-premises IP address space. To create a private control plane load balancer subnet, specify one of the following route tables (see Creating a Flannel Overlay VCN):

- vcn_private
- lpg_rt
- drg_rt

Table 4-9 Create a Public Control Plane Load Balancer Subnet

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

The difference in the following private subnet is the VCN private route table is used instead of the public route table. Depending on your needs, you could specify the LPG route table or the DRG route table instead.

Table 4-10 Create a Private Control Plane Load Balancer Subnet

Compute Web UI property	OCI CLI property
• Name: control-plane-endpoint • CIDR Block: ***kmilb_cidr*** • Route Table: Select "vcn_private" from the list • Private 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`: `true` • `--route-table-id`: OCID of the "vcn_private" route table • `--security-list-ids`: OCIDs of the "kmilb-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating VCN-Native Pod Networking Resources

VCN-Native Pod Networking enables you to directly manage the traffic from pods because pod IP addresses come directly from the VCN CIDR block and not from a network overlay such as Flannel Overlay. VCN-Native Pod Networking offers more flexibility and control over the traffic and allows you to use different security rules.

VCN-Native Pod Networking connects nodes in a Kubernetes cluster to pod subnets in the OKE VCN. Pod IP addresses within the OKE VCN are directly routable from other VCNs that are connected (peered) to the OKE VCN, and from on-premises networks.

When you create a cluster that uses VCN-Native Pod Networking, the VCN that you specify must have a subnet named "pod". You must provide a subnet named "pod" so that the system can find that subnet. The pod subnet has security rules that enable pods on control plane nodes to communicate directly with pods on worker nodes and with other pods and other resources. See Creating a VCN-Native Pod Networking Pod Subnet. If you select VCN-Native Pod Networking and do not have a subnet named "pod", the cluster creation will fail.

When you create a node pool for a cluster that is using VCN-Native Pod Networking, the pod subnet that you specify (Pod Communication > Pod Communication Subnet or `--pod-subnet-ids`) serves the function of a pod subnet for pods on worker nodes. That pod subnet should have security rules that enable pods on worker nodes to communicate directly with other pods on worker nodes and control plane nodes. You can optionally specify the worker node subnet as the pod subnet. The CIDR of the pod subnet that you specify must be larger than /25. The pod subnet should be larger than the worker node subnet.

In general, when you use VCN-Native Pod Networking, security rules can enable pods to communicate directly with other pods on the same node or on other nodes in the cluster, with other clusters, with other services, and with the internet.

Node Shapes and Number of Pods

When using the OCI VCN-Native Pod Networking CNI plugin, each pod needs a private IP address. By default, 31 IP addresses are assigned to a VNIC for use by pods running on the worker node.

You can specify the maximum number of pods that you want to run on a worker node. The default maximum is 31 pods per worker node. You can specify up to 110.

A node shape, and therefore a worker node, has a minimum of two VNICs. The first VNIC is connected to the worker subnet. The second VNIC is connected to the pod subnet. Therefore a worker node can support at least 31 pods. If you want more than 31 pods on a single worker node, specify a shape for the node pool that supports three or more VNICs: one VNIC to connect to the worker node subnet, and at least two VNICs to connect to the pod subnet.

A VM.PCAStandard1.4 standard node shape can have a maximum of four VNICs, and the worker node can support up to 93 pods. A VM.PCAStandard.E5.Flex node shape with five OCPUs can have a maximum of five VNICs, and the worker node can support up to 110 pods. A node cannot have more than 110 pods (see OKE Service Limits).

The following formula summarizes the maximum number of pods supported per node:

```
MIN( (Number of VNICs - 1) * 31 ), 110)
```

For information about all node shapes, see "Compute Shapes" in the Compute Instance Concepts chapter in the *Oracle Private Cloud Appliance Concepts Guide*.

VCN-Native Pod Networking Resources

The resource definitions in the following sections in this topic create a working example set of network resources for workload clusters when you are using VCN-Native Pod Networking. 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 VCN-Native Pod Networking for specific ports that must be open for specific purposes.

Create the following network resources. To use Terraform, see Example Terraform Scripts for VCN-Native Pod Networking Resources.

> **Note:**
>
> Create all of these network resources in the same compartment on the appliance.

- VCN. See Creating a VCN-Native Pod Networking VCN.
- Internet Gateway
- NAT Gateway
- Dynamic Routing Gateway
- Local Peering Gateway
- Route rules
- Security lists
- The following five subnets:
  - Pod. See Creating a VCN-Native Pod Networking Pod Subnet.
  - Worker. See Creating a VCN-Native Pod Networking Worker Subnet.
  - Worker load balancer. See Creating a VCN-Native Pod Networking Worker Load Balancer Subnet.
  - Control plane. See Creating a VCN-Native Pod Networking Control Plane Subnet.
  - Control plane load balancer. See Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet.

Workload Cluster Network CIDR Ranges for VCN-Native Pod Networking

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 for use with VCN-Native Pod Networking.

> **Note:**
>
> These are examples only. The CIDR ranges you use depend on the number of clusters you have, the number of nodes in each cluster, the shape you select for the worker nodes, and the type of networking you are using.

For VCN-Native Pod Networking, every pod gets an IP address assigned from the IP address pool that is defined in the pod subnet CIDR. The shape you specify for the node pool determines the maximum number of VNICs (pods) for each worker node, as described in Node Shapes and Number of Pods.

The primary difference between IP address requirements of VCN-Native Pod Networking and Flannel Overlay networking is that VCN-Native Pod Networking requires more IP addresses to be available. The table in Workload Cluster Network CIDR Ranges for Flannel Overlay Networking shows smaller CIDR ranges than the following table for VCN-Native Pod Networking CIDR ranges.

Note:

The pod subnet CIDR must be larger than /25. The pod subnet should be larger than the worker node subnet.

Table 4-11 Example CIDR Values to Use with VCN-Native Pod Networking

Variable Name	Description	Example Value
vcn_cidr	VCN CIDR range This Is a small VCN with 8192 IP's for creating OKE infrastructure.	172.31.0.0/19
worker_cidr	Worker subnet CIDR	172.31.8.0/21
workerlb_cidr	Worker load balancer subnet CIDR	172.31.0.0/23
kmi_cidr	OKE control plane subnet CIDR	172.31.4.0/22
kmilb_cidr	OKE control plane load balancer subnet CIDR	172.31.2.0/23
pod_cidr	Pod subnet CIDR	172.31.16.0/20
kube_client_cidr	CIDR for clients that are allowed to contact the Kubernetes API server	10.0.0.0/8

The IP Subnet Calculator on Calculator.net is one tool for finding all available networks for a given IP address and prefix length.

Workload Cluster Network Ports for VCN-Native Pod Networking

The following table lists ports that are used by workload clusters when you use VCN-Native Pod Networking. 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*.

If you are using a separate administration network, see OKE Cluster Management with Administration Network.

Table 4-12 Ports that Must Be Available for Use by Workload Clusters for VCN-Native Pod Networking

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***	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.
kmi_cidr	***worker_cidr***, ***pod_cidr***	10250	Kubernetes API endpoint to worker node communication.
kmi_cidr	***worker_cidr***, ***pod_cidr***	10256	Allow load balancer or network load balancer to communicate with `kube-proxy` on worker nodes or pod subnet.
pod_cidr	***kmilb_cidr***	12250	Pod to Kubernetes API endpoint communication.
kmi_cidr	***kmi_cidr***	2379-2381	Communication between the `etcd` server and metrics services. Ports 2379 and 2380 are used by Kubernetes to communicate with the `etcd` server. Port 2381 is used by Kubernetes to collect metrics from `etcd`.
kmi_cidr	***kmi_cidr***	10257-10260	Inbound connection for Kubernetes components.

Example Terraform Scripts for VCN-Native Pod Networking Resources

The following Terraform scripts create the network resources that are required by OKE when you are using VCN-Native Pod Networking. Other sections 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 for VCN-Native Pod Networking), the OKE pod subnet must be named `pod`, and the OKE control plane subnet must be named `control-plane`. See Workload Cluster Network CIDR Ranges for VCN-Native Pod Networking for comments about CIDR values.

- variables.tf
- terraform.tfvars
- provider.tf
- main.tf
- oke_vcn.tf
- oke_pod_seclist.tf
- oke_pod_subnet.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 when you are using VCN-Native Pod Networking. 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 for VCN-Native Pod Networking. The six CIDRs that are defined in this file are for the OKE VCN, pod subnet, 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 = "172.31.0.0/19"
}

# Subnet for KMIs where kube-apiserver and other control
# plane applications run, maximum 9 nodes.
variable "kmi_cidr" {
  description = "Kubernetes control plane subnet CIDR"
  default     = "172.31.4.0/22"
}

# Subnet for KMI load balancer.
variable "kmilb_cidr" {
  description = "Kubernetes control plane LB subnet CIDR"
  default     = "172.31.2.0/23"
}

# Subnet for worker nodes, maximum 128 nodes.
variable "worker_cidr" {
  description = "K8s worker subnet CIDR"
  default     = "172.31.8.0/21"
}

# Subnet for worker load balancer (for use by CCM).
variable "workerlb_cidr" {
  description = "K8s worker LB subnet CIDR"
  default     = "172.31.0.0/23"
}

# Subnet for pod communication
variable "pod_cidr" {
  description = "K8s pod communication subnet CIDR"
```

```
  default     = "172.31.16.0/20"
}

# Flag to Enable private endpoint
variable "enable_private_endpoint" {
  description = "Flag to create private control plane endpoint/service-lb"
  type = bool
  default = false
  nullable = false
}
```

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 above 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 the Kubernetes apiserver.
kube_client_cidr = "10.0.0.0/8"

# Security list rules for who is allowed to contact the worker load balancer
# (adjust 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 10/8 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.

The version of the OCI provider that you use must be at least v4.50.0 but no greater than v6.36.0.

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = ">= 4.50.0, <= 6.36.0"
      # If necessary, you can pin a specific version here
      # version = "6.36.0"
    }
  }
  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.kmi_cidr
      port_min = 22
      port_max = 22
    },
    {
      source   = var.worker_cidr
      port_min = 22
      port_max = 22
    },
    {
      source   = var.worker_cidr
      port_min = 10250
      port_max = 10250
    },
    {
      source   = var.worker_cidr
      port_min = 10256
      port_max = 10256
    },
    {
      source   = var.worker_cidr
      port_min = 30000
      port_max = 32767
    },
    {
      source   = var.workerlb_cidr
      port_min = 10256
      port_max = 10256
    },
    {
      source   = var.workerlb_cidr
      port_min = 30000
```

```
      port_max = 32767
    },
    {
      source   = var.kmi_cidr
      port_min = 10250
      port_max = 10250
    },
    {
      source   = var.kmi_cidr
      port_min = 10256
      port_max = 10256
    },
    {
      source   = var.pod_cidr
      port_min = 30000
      port_max = 32767
    },
  ]])
  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.kmi_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.worker_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.worker_cidr
      port_min = 12250
      port_max = 12250
    },
    {
      source   = var.pod_cidr
      port_min = 12250
      port_max = 12250
    },
  ]
  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.kmilb_cidr
      port_min = 12250
      port_max = 12250
    },
    {
      source   = var.worker_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.worker_cidr
      port_min = 12250
      port_max = 12250
    },
    {
      source   = var.kmi_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.kmi_cidr
      port_min = 2379
      port_max = 2381
    },
    {
      source   = var.kmi_cidr
      port_min = 8044
      port_max = 8045
    },
    {
      source   = var.kmi_cidr
      port_min = 10250
      port_max = 10250
    },
    {
      source   = var.kmi_cidr
      port_min = 10257
      port_max = 10260
    },
    {
      source   = var.pod_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.pod_cidr
      port_min = 12250
      port_max = 12250
    },
  ]
  pod_ingress_rules = [
    {
      source   = var.vcn_cidr
      port_min = 22
      port_max = 22
    },
    {
      source   = var.workerlb_cidr
      port_min = 10256
      port_max = 10256
    },
    {
```

```
      source   = var.worker_cidr
      port_min = 10250
      port_max = 10250
    },
    {
      source   = var.worker_cidr
      port_min = 10256
      port_max = 10256
    },
    {
      source   = var.worker_cidr
      port_min = 80
      port_max = 80
    },
  ]
}
```

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" "default_private" {
  manage_default_resource_id = oci_core_vcn.oke_vcn.default_route_table_id
  display_name               = "Default - private"
}

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_pod_seclist.tf

This file defines the security list for the pod subnet. The rules for this security list were defined in other Terraform files in this set.

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

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

  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.pod_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 = icmp_type
    for_each = [0, 8]

    content {
      # ping from VCN; unreachable/TTL from anywhere
      source      = var.kmi_cidr
      source_type = "CIDR_BLOCK"
      protocol    = "1"
      icmp_options {
        type = icmp_type.value
      }
    }
  }

  dynamic "ingress_security_rules" {
    for_each = var.pod_cidr != null ? [var.pod_cidr] : []

    content {
      source      = ingress_security_rules.value
      source_type = "CIDR_BLOCK"
      protocol    = "all"
    }
  }
}
```

oke_pod_subnet.tf

This file defines the pod subnet.

> **Important:**
>
> The name of the `pod` subnet must be exactly `pod`.

```
resource "oci_core_subnet" "pod" {
  cidr_block     = var.pod_cidr
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name               = "pod"
  dns_label                  = "pod"
  prohibit_public_ip_on_vnic = true

  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.pod.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 = icmp_type
    for_each = [0, 8]

    content {
      # ping from VCN; unreachable/TTL from anywhere
      source      = var.kmi_cidr
      source_type = "CIDR_BLOCK"
      protocol    = "1"
      icmp_options {
        type = icmp_type.value
      }
    }
  }
}
```

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 = var.enable_private_endpoint
  route_table_id             = var.enable_private_endpoint==false ?
oci_core_route_table.public[0].id : oci_core_vcn.oke_vcn.default_route_table_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
      }
    }
  }

  dynamic "ingress_security_rules" {
    for_each = var.enable_private_endpoint ? [1] : []
    content {
      source   = var.kmilb_cidr
      source_type = "CIDR_BLOCK"
      protocol = "6"
      tcp_options {
        min = var.kubernetes_api_port
```

```
        max = var.kubernetes_api_port
      }
    }
  }
}

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
      }
    }
  }
}
```

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 = var.enable_private_endpoint
  route_table_id             = var.enable_private_endpoint==false ?
oci_core_route_table.public[0].id : oci_core_default_route_table.default_private[0].id
  security_list_ids = [
```

```
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmilb.id
  ]
}
```

Creating a VCN-Native Pod Networking VCN

Create the following resources in the order listed:

1. VCN
2. Route rules
   - Public clusters:
     – Internet gateway and a route table with a route rule that references that internet gateway.
     – NAT gateway and a route table with a route rule that references that NAT gateway.
   - Private clusters:
     – Route table with no route rules.
     – (Optional) Dynamic Routing Gateway (DRG), attach the OKE VCN to that DRG, and create a route table with a route rule that references that DRG. See Private Clusters.
     – (Optional) Local Peering Gateway (LPG) and a route table with a route rule that references that LPG. See Private Clusters.
3. Security list. 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 VCN-Native Pod Networking 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

- Public internet access. For traffic on a public subnet that connects to the internet using public IP addresses, create an internet gateway and a route rule that references that internet gateway.

- Private internet access. For traffic on a private subnet that needs to connect to the internet without exposing private IP addresses, create a NAT gateway and a route rule that references that NAT gateway.
- VCN-only access. To restrict communication to only other resources on the same VCN, use the default route table, which has no route rules.
- Instances in another VCN. To enable communication between the cluster and an instance running on a different VCN, create a Local Peering Gateway (LPG) and a route rule that references that LPG.
- Data center IP address space. To enable communication between the cluster and the on-premises network IP address space, create a Dynamic Routing Gateway (DRG) and a route rule that references that DRG.

VCN Private Route Table

Edit the default route table that was created when you created the VCN. Change the name of the route table to vcn_private. This route table does not have any route rules. Do not add any route rules.

NAT Private Route Table

Create a NAT gateway and a route table with a route rule that references 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 VCN-Native Pod Networking Resources.

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

Private 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 Flannel Overlay Network Resources.

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

Compute Web UI property	OCI CLI property
• Name: nat_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: NAT private route rule	• `--display-name`: nat_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`: NAT private route rule

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

Local Peering Gateway

Create a Local Peering gateway (LPG) and a route table with a route rule that references the LPG.

Local Peering Gateway

To create the LPG, use the instructions in "Connecting VCNs through a Local Peering Gateway" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*.

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

Private 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 this example, use the following input to create the route table with a private route rule that references the LPG that was created in the preceding step.

Compute Web UI property	OCI CLI property
• Name: lpg_rt Route rule • Target Type: Local Peering Gateway • Local Peering Gateway: Name of the LPG that was created in the preceding step • CIDR Block: ***CIDR_for_the_second_VCN*** • Description: LPG private route rule	• `--display-name`: lpg_rt `--route-rules` • `networkEntityId`: OCID of the LPG that was created in the preceding step • `destinationType`: `CIDR_BLOCK` • `destination`: ***CIDR_for_the_second_VCN*** • `description`: LPG private route rule

Note the name and OCID of this route table for assignment to the "control-plane-endpoint" subnet (Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet).

Add the same route rule on the second VCN (the peered VCN), specifying the OKE VCN CIDR as the destination.

Dynamic Routing Gateway

Create a Dynamic Routing gateway (DRG) and a route table with a route rule that references the DRG.

Dynamic Routing Gateway

To create the DRG and attach the OKE VCN to that DRG, use the instructions in "Connecting to the On-Premises Network through a Dynamic Routing Gateway" in the Networking chapter of the *Oracle Private Cloud Appliance User Guide*. Create the DRG in the OKE VCN compartment, and then attach the OKE VCN to that DRG.

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

Private 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 this example, use the following input to create the route table with a private route rule that references the DRG that was created in the preceding step.

Compute Web UI property	OCI CLI property
• Name: drg_rt Route rule • Target Type: Dynamic Routing Gateway • Dynamic Routing: Name of the DRG that was created in the preceding step • CIDR Block: 0.0.0.0/0 • Description: DRG private route rule	• `--display-name`: drg_rt `--route-rules` • `networkEntityId`: OCID of the DRG that was created in the preceding step • `destinationType`: `CIDR_BLOCK` • `destination`: `0.0.0.0/0` • `description`: DRG private route rule

Note the name and OCID of this route table for assignment to the "control-plane-endpoint" subnet (Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet).

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 VCN-Native Pod Networking 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 VCN-Native Pod Networking 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 VCN-Native Pod Networking Resources.

Compute Web UI property	OCI CLI property
• Name: Default Security List for oketest-vcn	`--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: "Blocks incoming requests from any source."	**Ingress Rule 2** • `isStateless`: `false` • `source`: `0.0.0.0/0` • `sourceType`: `CIDR_BLOCK` • `protocol`: `1` • `icmpOptions` – `type`: `3` • `description`: "Blocks incoming requests from any source."
Ingress Rule 3 • Stateless: uncheck the box • Ingress CIDR: 0.0.0.0/0 • IP Protocol: ICMP – Parameter Type: 11: Time Exceeded • Description: "Time exceeded."	**Ingress Rule 3** • `isStateless`: `false` • `source`: `0.0.0.0/0` • `sourceType`: `CIDR_BLOCK` • `protocol`: `1` • `icmpOptions` – `type`: `11` • `description`: "Time exceeded."

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

Creating a VCN-Native Pod Networking Pod Subnet

The instructions in this topic create a pod subnet named "pod" in the VCN that provides the private IP addresses for pods running on the control plane nodes. The number of IP addresses in this subnet should be equal to or greater than the number of IP addresses in the control plane subnet. The pod subnet must be a private subnet.

The pod subnet supports communication between pods and direct access to individual pods using private pod IP addresses. The pod subnet must be private. The pod subnet enables pods to communicate with other pods on the same worker node, with pods on other worker nodes, with OCI services (through a service gateway) and with the internet (through a NAT gateway).

Create the following resources in the order listed:

1. Pod security list
2. Pod subnet

Create a Pod 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 VCN-Native Pod Networking Resources.

The security rules shown in the following table define traffic that is allowed to contact pods directly. Use these security rules as part of network security groups (NSGs) or in security lists. Oracle recommends using NSGs. See Security Best Practices.

The security rules apply to all pods in all the worker nodes connected to the pod subnet specified for a node pool.

Route incoming requests to pods based on routing policies specified by routing rules and route tables. See the route tables defined in Creating a VCN-Native Pod Networking VCN.

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

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

Compute Web UI property	OCI CLI property

Ingress Rule 1
• Stateless: uncheck the box
• Ingress CIDR: ***vcn_cidr***
• IP Protocol: TCP
  – Destination Port Range: 22
• Description: "Allow SSH connection to the pod subnet from all subnets in the VCN."
	Ingress Rule 1
	• `isStateless`: `false`
	• `source`: ***vcn_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `22`
	  – `min`: `22`
	• `description`: "Allow SSH connection to the pod subnet from all subnets in the VCN."

Ingress Rule 2
• Stateless: uncheck the box
• Ingress CIDR: ***workerlb_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10256
• Description: "Allow the worker load balancer to contact the pods."
	Ingress Rule 2
	• `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 pods."

Ingress Rule 3
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10250
• Description: "Allow Kubernetes API endpoint to pod (via worker node) communication."
	Ingress Rule 3
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `10250`
	  – `min`: `10250`
	• `description`: "Allow Kubernetes API endpoint to pod (via worker node) communication."

Compute Web UI property	OCI CLI property

Ingress Rule 4
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10256
• Description: "Allow Load Balancer or Network Load Balancer to communicate with the `kube-proxy` pod (via the worker subnet)."
	Ingress Rule 4
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `10256`
	  – `min`: `10256`
	• `description`: "Allow Load Balancer or Network Load Balancer to communicate with the `kube-proxy` pod (via the worker subnet)."

Ingress Rule 5
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: 80
• Description: "Allow the worker node to contact the pods."
This ingress is optional. This port is open for an end user application. This rule could be different based on what applications are deployed.
	Ingress Rule 5
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `80`
	  – `min`: `80`
	• `description`: "Allow the worker node to contact the pods."
	This ingress is optional. This port is open for an end user application. This rule could be different based on what applications are deployed.

Ingress Rule 6 • Stateless: uncheck the box • Ingress CIDR: ***pod_cidr*** • IP Protocol: All protocols • Description: "Allow the pod CIDR to communicate with itself."	**Ingress Rule 6** • `isStateless`: `false` • `source`: ***pod_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `all` • `description`: "Allow the pod CIDR to communicate with itself."

Create the Pod 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 VCN-Native Pod Networking Resources.

For this example, use the following input to create the pod subnet. Use the OCID of the VCN that was created in Creating a VCN-Native Pod Networking VCN. Create the pod subnet in the same compartment where you created the VCN.

> **Important:**
>
> The name of this subnet must be exactly "pod".

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

Creating a VCN-Native Pod Networking 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 VCN-Native Pod Networking 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`
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."
Ten ingress security rules:	**Ten ingress security rules:** `--ingress-security-rules`

Compute Web UI property	OCI CLI property

Ingress Rule 1
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: 22
• Description: "Allow SSH connection from the control plane subnet."
	Ingress Rule 1
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `22`
	  – `min`: `22`
	• `description`: "Allow SSH connection from the control plane subnet."

Ingress Rule 2
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: 22
• Description: "Allow SSH connection from the worker subnet."
	Ingress Rule 2
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `22`
	  – `min`: `22`
	• `description`: "Allow SSH connection from the worker subnet."

Ingress Rule 3
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10250
• Description: "Allow Kubernetes API endpoint to worker node communication."
	Ingress Rule 3
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `10250`
	  – `min`: `10250`
	• `description`: "Allow Kubernetes API endpoint to worker node communication."

Ingress Rule 4
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10256
• Description: "Allow Load Balancer or Network Load Balancer to communicate with `kube-proxy` on worker nodes."
	Ingress Rule 4
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `10256`
	  – `min`: `10256`
	• `description`: "Allow Load Balancer or Network Load Balancer to communicate with `kube-proxy` on worker nodes."

Compute Web UI property	OCI CLI property
Ingress Rule 5 • Stateless: uncheck the box • Ingress CIDR: ***worker_cidr*** • IP Protocol: TCP – Destination Port Range: 30000-32767 • Description: "Allow traffic to worker nodes."	**Ingress Rule 5** • `isStateless`: `false` • `source`: ***worker_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `32767` – `min`: `30000` • `description`: "Allow traffic to worker nodes."
Ingress Rule 6 • Stateless: uncheck the box • Ingress CIDR: ***workerlb_cidr*** • IP Protocol: TCP – Destination Port Range: 10256 • Description: "Allow Load Balancer or Network Load Balancer to communicate with `kube-proxy` on worker nodes."	**Ingress Rule 6** • `isStateless`: `false` • `source`: ***workerlb_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `10256` – `min`: `10256` • `description`: "Allow Load Balancer or Network Load Balancer to communicate with `kube-proxy` on worker nodes."
Ingress Rule 7 • Stateless: uncheck the box • Ingress CIDR: ***workerlb_cidr*** • IP Protocol: TCP – Destination Port Range: 30000-32767 • Description: "Allow worker nodes to receive connections through Network Load Balancer."	**Ingress Rule 7** • `isStateless`: `false` • `source`: ***workerlb_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `32767` – `min`: `30000` • `description`: "Allow worker nodes to receive connections through Network Load Balancer."

Compute Web UI property	OCI CLI property

Ingress Rule 8
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10250
• Description: "Allow Kubernetes API endpoint to worker node communication."
	Ingress Rule 8
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `10250`
	  – `min`: `10250`
	• `description`: "Allow Kubernetes API endpoint to worker node communication."

Ingress Rule 9
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10256
• Description: "Allow Load Balancer or Network Load Balancer to communicate with `kube-proxy` on worker nodes."
	Ingress Rule9
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `10256`
	  – `min`: `10256`
	• `description`: "Allow Load Balancer or Network Load Balancer to communicate with `kube-proxy` on worker nodes."

Ingress Rule 10
• Stateless: uncheck the box
• Ingress CIDR: ***pod_cidr***
• IP Protocol: TCP
  – Destination Port Range: 30000-32767
• Description: "Allow worker nodes to receive connections through the pod subnet."
	Ingress Rule 10
	• `isStateless`: `false`
	• `source`: ***pod_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `32767`
	  – `min`: `30000`
	• `description`: "Allow worker nodes to receive connections through the pod subnet."

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 VCN-Native Pod Networking Resources.

For this example, use the following input to create the worker subnet. Use the OCID of the VCN that was created in Creating a VCN-Native Pod Networking VCN. Create the worker subnet in the same compartment where you created the VCN.

Create either a NAT private worker subnet or a VCN private worker subnet. Create a NAT private worker subnet to communicate outside the VCN.

Table 4-13 Create a NAT Private Worker Subnet

Compute Web UI property	OCI CLI property
• Name: worker • CIDR Block: ***worker_cidr*** • Route Table: Select "nat_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 "nat_private" route table • `--security-list-ids`: OCIDs of the "worker-seclist" security list and the "Default Security List for oketest-vcn" security list

The difference in the following private subnet is the VCN private route table is used instead of the NAT private route table.

Table 4-14 Create a VCN Private Worker Subnet

Compute Web UI property	OCI CLI property
• Name: worker • CIDR Block: ***worker_cidr*** • Route Table: Select "vcn_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 "vcn_private" route table • `--security-list-ids`: OCIDs of the "worker-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating a VCN-Native Pod Networking 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 VCN-Native Pod Networking 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`
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."
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."

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 VCN-Native Pod Networking 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 a VCN-Native Pod Networking VCN. Create the worker load balancer subnet in the same compartment where you created the VCN.

Create either a private or a public worker load balancer subnet. Create a public worker load balancer subnet to use with a public cluster. Create a private worker load balancer subnet to expose applications in a private cluster.

Table 4-15 Create a Public Worker Load Balancer Subnet

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

The difference in the following private subnet is the VCN private route table is used instead of the public route table.

Table 4-16 Create a VCN Private Worker Load Balancer Subnet

Compute Web UI property	OCI CLI property
• Name: service-lb • CIDR Block: ***workerlb_cidr*** • Route Table: Select "vcn_private" from the list • Private 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`: `true` • `--route-table-id`: OCID of the "vcn_private" route table • `--security-list-ids`: OCIDs of the "workerlb-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating a VCN-Native Pod Networking 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 VCN-Native Pod Networking Resources.

For this example, use the following input for the control plane subnet security list. The `kubernetes_api_port` is the port used to access the Kubernetes API: port 6443. See also Workload Cluster Network Ports for VCN-Native Pod Networking.

Compute Web UI property	OCI CLI property
• Name: kmi-seclist	• `--vcn-id`: `ocid1.vcn.`***`oke_vcn_id`*** • `--display-name`: `kmi-seclist`
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."
Twelve ingress security rules:	**Twelve 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 clients to communicate with Kubernetes API."
	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 clients to communicate with Kubernetes API."

Ingress Rule 2
• Stateless: uncheck the box
• Ingress CIDR: ***`kmilb_cidr`***
• IP Protocol: TCP
  – Destination Port Range: ***`kubernetes_api_port`***
• Description: "Allow the load balancer to communicate with Kubernetes control plane APIs."
	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 the load balancer to communicate with Kubernetes control plane APIs."

Compute Web UI property	OCI CLI property

Ingress Rule 3
• Stateless: uncheck the box
• Ingress CIDR: ***kmilb_cidr***
• IP Protocol: TCP
  – Destination Port Range: `12250`
• Description: "Allow Kubernetes worker to Kubernetes API endpoint communication via the control plane load balancer."
	Ingress Rule 3
	• `isStateless`: `false`
	• `source`: ***kmilb_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `12250`
	  – `min`: `12250`
	• `description`: "Allow Kubernetes worker to Kubernetes API endpoint communication via the control plane load balancer."

Ingress Rule 4
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: ***kubernetes_api_port***
• Description: "Allow worker nodes to access the Kubernetes API."
	Ingress Rule 4
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: ***kubernetes_api_port***
	  – `min`: ***kubernetes_api_port***
	• `description`: "Allow worker nodes to access the Kubernetes API."

Ingress Rule 5
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: `12250`
• Description: "Allow Kubernetes worker to Kubernetes API endpoint communication."
	Ingress Rule 5
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: `12250`
	  – `min`: `12250`
	• `description`: "Allow Kubernetes worker to Kubernetes API endpoint communication."

Ingress Rule 6
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: ***kubernetes_api_port***
• Description: "Allow the control plane to reach itself."
	Ingress Rule 6
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	`destinationPortRange`
	  – `max`: ***kubernetes_api_port***
	  – `min`: ***kubernetes_api_port***
	• `description`: "Allow the control plane to reach itself."

<table>
<tr><th>Compute Web UI property</th><th>OCI CLI property</th></tr>
<tr><td>Ingress Rule 7
Stateless: uncheck the box
Ingress CIDR: kmi_cidr
IP Protocol: TCP
Destination Port Range: 2379-2381
Description: "Allow the control plane to reach etcd services and metrics. Ports 2379 and 2380 are used by Kubernetes to communicate with the etcd server. Port 2381 is used by Kubernetes to collect metrics from etcd."</td><td>Ingress Rule 7
isStateless: false
source: kmi_cidr
sourceType: CIDR_BLOCK
protocol: 6
tcpOptions
destinationPortRange
max: 2381
min: 2379
description: "Allow the control plane to reach etcd services and metrics. Ports 2379 and 2380 are used by Kubernetes to communicate with the etcd server. Port 2381 is used by Kubernetes to collect metrics from etcd."</td></tr>
<tr><td>Ingress Rule 8
Stateless: uncheck the box
Ingress CIDR: kmi_cidr
IP Protocol: TCP
Destination Port Range: 8044-8045
Description: "Allow the control plane to reach etcd service discovery."</td><td>Ingress Rule 8
isStateless: false
source: kmi_cidr
sourceType: CIDR_BLOCK
protocol: 6
tcpOptions
destinationPortRange
max: 8045
min: 8044
description: "Allow the control plane to reach etcd service discovery."</td></tr>
<tr><td>Ingress Rule 9
Stateless: uncheck the box
Ingress CIDR: kmi_cidr
IP Protocol: TCP
Destination Port Range: 10250
Description: "Allow Kubernetes API endpoint to control plane node communication."</td><td>Ingress Rule 9
isStateless: false
source: kmi_cidr
sourceType: CIDR_BLOCK
protocol: 6
tcpOptions
destinationPortRange
max: 10250
min: 10250
description: "Allow Kubernetes API endpoint to control plane node communication."</td></tr>
</table>

Compute Web UI property	OCI CLI property

Ingress Rule 10
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: 10257-10260
• Description: "Allow inbound connection for Kubernetes components."
	Ingress Rule 10
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `10260`
	  – `min`: `10257`
	• `description`: "Allow inbound connection for Kubernetes components."

Ingress Rule 11
• Stateless: uncheck the box
• Ingress CIDR: ***pod_cidr***
• IP Protocol: TCP
  – Destination Port Range: ***kubernetes_api_port***
• Description: "Allow pods to communicate with Kubernetes APIs."
	Ingress Rule 11
	• `isStateless`: `false`
	• `source`: ***pod_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: ***kubernetes_api_port***
	  – `min`: ***kubernetes_api_port***
	• `description`: "Allow pods to communicate with Kubernetes APIs."

Ingress Rule 12
• Stateless: uncheck the box
• Ingress CIDR: ***pod_cidr***
• IP Protocol: TCP
  – Destination Port Range: 12250
• Description: "Allow Kubernetes pods to Kubernetes API endpoint communication."
	Ingress Rule 12
	• `isStateless`: `false`
	• `source`: ***pod_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `12250`
	  – `min`: `12250`
	• `description`: "Allow Kubernetes pods to Kubernetes API endpoint communication."

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 VCN-Native Pod Networking Resources.

Use the following input to create the control plane subnet. Use the OCID of the VCN that was created in Creating a VCN-Native Pod Networking VCN. Create the control plane subnet in the same compartment where you created the VCN.

Create either a NAT private control plane subnet or a VCN private control plane subnet. Create a NAT private control plane subnet to communicate outside the VCN.

> **Important:**
>
> The name of this subnet must be exactly "control-plane".

Table 4-17 Create a Data Center Private Control Plane Subnet

Compute Web UI property	OCI CLI property
• Name: control-plane • CIDR Block: ***kmi_cidr*** • Route Table: Select "nat_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 "nat_private" route table • `--security-list-ids`: OCIDs of the "kmi-seclist" security list and the "Default Security List for oketest-vcn" security list

The difference in the following private subnet is the VCN private route table is used instead of the NAT private route table.

Table 4-18 Create a VCN Private Control Plane Subnet

Compute Web UI property	OCI CLI property
• Name: control-plane • CIDR Block: ***kmi_cidr*** • Route Table: Select "vcn_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 "vcn_private" route table • `--security-list-ids`: OCIDs of the "kmi-seclist" security list and the "Default Security List for oketest-vcn" security list

Creating a VCN-Native Pod Networking 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 VCN-Native Pod Networking 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
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."
Six ingress security rules:	**Six ingress security rules:** `--ingress-security-rules`
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 a Kubernetes container to communicate with Kubernetes APIs."	**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 a Kubernetes container to communicate with Kubernetes APIs."

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: ***kubernetes_api_port***
• Description: "Allow clients to connect with the Kubernetes cluster."
	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 clients to connect with the Kubernetes cluster."

Ingress Rule 3:
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: ***kubernetes_api_port***
• Description: "Allow the control plane to reach itself via the load balancer."
	Ingress Rule 3:
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: ***kubernetes_api_port***
	  – `min`: ***kubernetes_api_port***
	• `description`: "Allow the control plane to reach itself via the load balancer."

Ingress Rule 4:
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: TCP
  – Destination Port Range: ***kubernetes_api_port***
• Description: "Allow worker nodes to connect with the cluster via the control plane load balancer."
	Ingress Rule 4:
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: ***kubernetes_api_port***
	  – `min`: ***kubernetes_api_port***
	• `description`: "Allow worker nodes to connect with the cluster via the control plane load balancer."

Compute Web UI property	OCI CLI property
Ingress Rule 5: • Stateless: uncheck the box • Ingress CIDR: ***`worker_cidr`*** • IP Protocol: TCP – Destination Port Range: `12250` • Description: "Allow Kubernetes worker to Kubernetes API endpoint communication via the load balancer."	**Ingress Rule 5:** • `isStateless`: `false` • `source`: ***`worker_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `12250` – `min`: `12250` • `description`: "Allow Kubernetes worker to Kubernetes API endpoint communication via the load balancer."
Ingress Rule 6: • Stateless: uncheck the box • Ingress CIDR: ***`pod_cidr`*** • IP Protocol: TCP – Destination Port Range: `12250` • Description: "Allow Kubernetes pods to Kubernetes API endpoint communication via the load balancer."	**Ingress Rule 6:** • `isStateless`: `false` • `source`: ***`pod_cidr`*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `12250` – `min`: `12250` • `description`: "Allow Kubernetes pods to Kubernetes API endpoint communication via the 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 VCN-Native Pod Networking 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 a VCN-Native Pod Networking VCN. Create the control plane load balancer subnet in the same compartment where you created the VCN.

Create either a private or a public control plane load balancer subnet. Create a public control plane load balancer subnet to use with a public cluster. Create a private control plane load balancer subnet to use with a private cluster.

See Private Clusters for information about using Local Peering Gateways to connect a private cluster to other instances on the Private Cloud Appliance and using Dynamic Routing Gateways to connect a private cluster to the on-premises IP address space. To create a private control plane load balancer subnet, specify one of the following route tables (see Creating a Flannel Overlay VCN):

- vcn_private
- lpg_rt
- drg_rt

Table 4-19 Create a Public Control Plane Load Balancer Subnet

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

The difference in the following private subnet is the VCN private route table is used instead of the public route table. Depending on your needs, you could specify the LPG route table or the DRG route table instead.

Table 4-20 Create a Private Control Plane Load Balancer Subnet

Compute Web UI property	OCI CLI property
• Name: control-plane-endpoint • CIDR Block: ***kmilb_cidr*** • Route Table: Select "vcn_private" from the list • Private 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`: `true` • `--route-table-id`: OCID of the "vcn_private" route table • `--security-list-ids`: OCIDs of the "kmilb-seclist" security list and the "Default Security List for oketest-vcn" security list

5 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.

You can create either a public cluster or a private cluster. See Public and Private Clusters for the resources required for each.

> **Note:**
>
> You cannot create both public and private clusters in one VCN.

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

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.

If you create a public 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. See Creating the OraclePCA-OKE.cluster_id Tag.
> - 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.

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

To create a node pool at the same time that you create the cluster, you must use the Compute Web UI.

To specify tags to be applied to all load balancers created by Kubernetes services, you must use the OCI CLI.

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

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. On the clusters list page, select 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.

     If the Kubernetes version that you want to use is not listed, use the OCI CLI or the OCI API to create the cluster and specify the Kubernetes version.
   - **Tagging**: Add defined or free-form tags for the cluster resource.

     > **Note:**
     >
     > Do not specify values for the OraclePCA-OKE.cluster_id 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 OraclePCA defined tags to provide the following information for control plane nodes. If these tags are not listed in the Compute Web UI Tagging menus, you must create them. See Creating OraclePCA Tags.

     > **Important:**
     >
     > If you are using Private Cloud Appliance Release 3.0.2-b1081557, these defined tags are not recognized. You must use free-form tags to specify these values as described in the workaround in Create Cluster Does Not Support Extension Parameters. In Private Cloud Appliance Release 3.0.2-b1185392 and later, the free-form tags are deprecated; use the defined tags described below for SSH key, number of control plane nodes, node shape, and node configuration in Private Cloud Appliance Release 3.0.2-b1185392 and later.

     > **Note:**
     >
     > None of these values - SSH key, number of nodes, node shape, or node shape configuration - can be set or changed after the cluster is created. If you set these tags when you update the cluster, the new values are ignored.

     – Your public SSH key.

Specify sshkey for the tag key (OraclePCA.sshkey). 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 cpNodeCount for the tag key (OraclePCA.cpNodeCount), and select 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 cpNodeShape for the tag key (OraclePCA.cpNodeShape), 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 cpNodeShapeConfig for the tag key (OraclePCA.cpNodeShapeConfig). 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.

> **Note:**
>
> If the cluster will have 1-10 worker nodes, specify at least 16 GB memory. If the cluster will have 11-128 worker nodes, specify at least 2 OCPUs and 32 GB memory. Note that you cannot change the number of OCPUs or amount of memory when you update the cluster.

  In the Value field for the tag, enter the node shape configuration value as shown in the following examples.

  In the following example, the default amount of memory will be configured:

```
{"ocpus":1}
```

  In the following example, the amount of memory is specified:

```
{"ocpus":2, "memoryInGBs":48}
```

> **Note:**
>
> If you use Terraform to specify a complex value (a value that is a key/value pair), then you must escape the double quotation marks in the value as shown in the following example:
>
> ```
> "OraclePCA.cpNodeShapeConfig"="{\"ocpus\":2,\"memoryInGBs\":48}"
> ```

- **Add-ons**: This section shows a tile for each add-on that is available for this cluster. In the Create Cluster dialog, all add-ons are Disabled. See Installing the WebLogic Kubernetes Operator Add-on.

4. Select 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. For more description, see Creating Flannel Overlay Network Resources.

     VCN-Native Pod Networking connects nodes in a Kubernetes cluster to pod subnets in the OKE VCN. As a result, pod IP addresses within the OKE VCN are directly routable from other VCNs that are connected (peered) to the OKE VCN, and from on-premises networks. For more description, see Creating VCN-Native Pod Networking Resources.

     > **Note:**
     >
     > If you specify VCN-Native Pod Networking, then the VCN you specify must have a subnet named "pod". See Creating VCN-Native Pod Networking Resources.

   - **VCN**. Select the VCN that has the configuration of the "oke_vcn" VCN described in Creating a Flannel Overlay VCN or Creating a VCN-Native Pod Networking VCN.
   - **Kubernetes Service LB Subnet**. The subnet that is configured to host the load balancer in an OKE cluster. To create a public cluster, create and specify here the public version of the "service-lb" subnet described in Creating a Flannel Overlay Worker Load Balancer Subnet or Creating a VCN-Native Pod Networking Worker Load Balancer Subnet. To create a private cluster, create and specify here the private version of the "service-lb" subnet.
   - **Kubernetes API Endpoint Subnet**. The regional subnet in which to place the cluster endpoint. To create a public cluster, create and specify here the public version of the "control-plane-endpoint" subnet described in Creating a Flannel Overlay Control Plane Load Balancer Subnet or Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet. To create a private cluster, create and specify here the private version of the "control-plane-endpoint" 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, select 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. Select Next.
7. On the **Node Pool** page, select the Add Node Pool button to optionally add a node pool as part of creating this cluster. See Creating an OKE Worker Node Pool to add node pools after the cluster is created.

   If you select the Add Node Pool button, enter the following information in the Add Node Pool section:

   - **Name**: The name of the new node pool. Avoid using confidential information.
   - **Compartment**: The compartment in which to create the new node pool.
   - **Node Count**: 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.
   - See Creating an OKE Worker Node Pool for information about Network Security Groups, Placement Configuration, Source Image, Shape, and Pod Communication.

8. Review your entries.

   If you created a node pool, you have the opportunity to edit or delete the node pool in this review.

9. Select Submit.

   The details page for the cluster is displayed. Scroll to the Resources section and select Work Requests to see the progress of the cluster creation. If you created a node pool, the NODEPOOL_CREATE work request might still be In Progress for a time after the cluster is Active and the CLUSTER_CREATE work request is Succeeded.

   The cluster details page does not list OraclePCA tags on the Tags tab (and you cannot filter a list of clusters by the values of OraclePCA tags). To review the settings of the OraclePCA tags, use the CLI.

   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

To install a cluster add-on, use the `cluster install-addon` command after you have created the cluster. See Installing the WebLogic Kubernetes Operator Add-on.

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.

- The Kubernetes version that you want to use. Use the following command to show a list of available Kubernetes versions:

  ```
  oci ce cluster-options get --cluster-option-id all
  ```

  You might be able to list more Kubernetes versions by using the `compute image list` command and looking in the display name. In the following example, the Kubernetes version in the image is 1.29.9:

  ```
  "display-name": "uln-pca-Oracle-Linux8-OKE-1.29.9-20250325.oci"
  ```

  Another way to specify a version that is not listed is to use the OCID of an older cluster instead of the keyword `all` as the argument of the `--cluster-option-id` option to list the Kubernetes version used for that specified cluster:

  ```
  oci ce cluster-options get --cluster-option-id cluster_OCID
  ```

  If you are using Private Cloud Appliance Release 3.0.2-b1081557, the `cluster-options get` command is not available. Use the `compute image list` command to get the Kubernetes version from the image display name.
- 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 a Flannel Overlay VCN or Creating a VCN-Native Pod Networking VCN.
- OCID of the OKE service LB subnet. Specify the subnet that has configuration like the "service-lb" subnet described in Creating a Flannel Overlay Worker Load Balancer Subnet or Creating a VCN-Native Pod Networking Worker Load Balancer Subnet. For a public cluster, follow the instructions to create the public version of the "service-lb" subnet. For a private cluster, create the private version of the "service-lb" 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 a Flannel Overlay Control Plane Load Balancer Subnet or Creating a VCN-Native Pod Networking Control Plane Load Balancer Subnet. For a public cluster, follow the instructions to create the public version of the "control-plane-endpoint" subnet. For a private cluster, create the private version of the "control-plane-endpoint" 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. (Optional) Specify either `OCI_VCN_IP_NATIVE` or `FLANNEL_OVERLAY` for the value of the `cniType` parameter in the argument for the `--cluster-pod-network-options` option. See the descriptions of Flannel Overlay and VCN-Native Pod Networking in the Compute Web UI procedure. If you do not specify the `--cluster-pod-network-options` option, `FLANNEL_OVERLAY` is used.

  ```
  --cluster-pod-network-options '[{"cniType": "OCI_VCN_IP_NATIVE"}]'
  ```

> **Note:**
>
> If you specify `OCI_VCN_IP_NATIVE`, then the VCN you specify must have a subnet named `pod`. See Creating VCN-Native Pod Networking Resources.

2. (Optional) Add defined or free-form tags for the cluster resource by using the `--defined-tags` or `--freeform-tags` options.

   > **Note:**
   >
   > Do not specify values for the OraclePCA-OKE.cluster_id 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 `--defined-tags` option to provide the following information for control plane nodes. Specify OraclePCA as the tag namespace.

   > **Important:**
   >
   > If you are using Private Cloud Appliance Release 3.0.2-b1081557, these defined tags are not recognized. You must use free-form tags to specify these values as described in the workaround in Create Cluster Does Not Support Extension Parameters. In Private Cloud Appliance Release 3.0.2-b1185392 and later, the free-form tags are deprecated; use the defined tags described below for SSH key, number of control plane nodes, node shape, and node configuration in Private Cloud Appliance Release 3.0.2-b1185392 and later.

   > **Note:**
   >
   > None of these values - SSH key, number of nodes, node shape, or node shape configuration - can be set or changed after the cluster is created. If you set these tags when you update the cluster, the new values are ignored.

   - 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 `cpNodeCount` 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 `cpNodeShape` 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.

  ```
  $ oci compute shape list --compartment-id compartment_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 `cpNodeShapeConfig` 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.

  > **Note:**
  >
  > If the cluster will have 1-10 worker nodes, specify at least 16 GB memory. If the cluster will have 11-128 worker nodes, specify at least 2 OCPUs and 32 GB memory. Note that you cannot change the number of OCPUs or amount of memory when you update the cluster.

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

```
{
  "OraclePCA": {
    "sshkey": "ssh-rsa remainder_of_key_text",
    "cpNodeCount": 1,
    "cpNodeShape": "VM.PCAStandard1.Flex",
    "cpNodeShapeConfig": {
      "ocpus": 2,
      "memoryInGBs": 48
    }
  }
}
```

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.

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

3. (Optional) You can use the `--service-lb-defined-tags` or `--service-lb-freeform-tags` options to specify tags to be applied to all load balancers created by Kubernetes services. Ensure that the applicable dynamic group includes the `use tag-namespaces` policy. See Exposing Containerized Applications.

4. Run the create cluster command.

If the `--endpoint-subnet-id` that you specify is a public subnet, then a public endpoint is created, and the `--endpoint-public-ip-enabled` option must be set to `true`.

If the `--endpoint-subnet-id` that you specify is a private subnet, then a private endpoint is created, and the `--endpoint-public-ip-enabled` option must be set to `false`.

Example:

```
$ oci ce cluster create \
--compartment-id ocid1.compartment.unique_ID --kubernetes-version version \
--name "Native Cluster" --vcn-id ocid1.vcn.unique_ID \
--cluster-pod-network-options '{"cniType":"OCI_VCN_IP_NATIVE"}' \
--endpoint-subnet-id control-plane-endpoint_subnet_OCID \
--endpoint-public-ip-enabled false \
--service-lb-subnet-ids '["service-lb_subnet_OCID"]' \
--defined-tags '{"OraclePCA":{"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
```

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 required.

   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 you do not specify the `--profile` option, the current value of your `OCI_CLI_PROFILE` environment variable is used. Best practice is to specify this value.

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

   Example:

   Use the following command to configure a Kubernetes configuration 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 --profile profile-name
   New config written to the Kubeconfig file /home/username/.kube/config
   ```

   A Kubernetes configuration file includes an OCI CLI command that dynamically generates an authentication token and inserts it when you run a `kubectl` command. By default, the OCI CLI command in the Kubernetes configuration file uses your current OCI CLI profile when generating an authentication token. If you have defined multiple profiles in your OCI

CLI configuration file, use one of the following methods to specify which profile to use when generating the authentication token. The value of ***profile-name*** is the name of the profile in your OCI CLI configuration file.

- Ensure that your `OCI_CLI_PROFILE` environment variable is set to the profile for the tenancy where the `ocid1.cluster.`***unique_ID*** resides. This setting is ignored if one of the following methods was used to specify the profile for this cluster in the Kubernetes configuration file.
- Specify the `--profile` option on the `create-kubeconfig` command line as shown in the preceding example command.
- Edit the generated configuration file as shown in the following example.

```
user:
  exec:
    apiVersion: client.authentication.k8s.io/v1beta1
    args:
    - ce
    - cluster
    - generate-token
    - --cluster-id
    - cluster ocid
    - --profile
    - profile-name
    command: oci
    env: []
```

Use the following command to set your `KUBECONFIG` environment variable to the Kubernetes configuration 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

Before you run `kubectl` commands, enure that your `OCI_CLI_PROFILE` environment variable is set to the name of the profile that is defined in your OCI CLI configuration file:

```
$ export OCI_CLI_PROFILE=profile-name
```

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.

Best practice is to keep your clusters upgraded so that they are always running versions of Kubernetes that are currently supported by OKE. See the instructions in the following procedures to determine whether a newer supported version of Kubernetes is available.

> **Note:**
>
> If you set or modify any of the following tags, the new values are ignored: SSH key (OraclePCA.sshkey), number of nodes (OraclePCA.cpNodeCount), node shape (OraclePCA.cpNodeShape), or node shape configuration (OraclePCA.cpNodeShapeConfig). These values can be set only when you create the cluster.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. In the clusters list, if a Kubernetes version update is available, an exclamation point icon is displayed next to the Kubernetes Version number. To upgrade to a newer version, select the Actions menu and select Upgrade Available. Select a new version from the drop-down menu.

   Alternatively, on the cluster details page, select Upgrade Available next to the Kubernetes Version number, or select the Upgrade button at the top of the page.
3. On the clusters list page, select the name of the cluster that you want to update.
4. At the top of the cluster details page, select the Edit button.

   Do not specify values for the OraclePCA-OKE.cluster_id 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.
5. When you are finished making changes, select Save Changes.

Using the OCI CLI

1. Get the OCID of the cluster that you want to update: `oci ce cluster list`
2. Check whether a newer version of Kubernetes is available.

   Run the get cluster command: `oci ce cluster get`

   If the value of `available-kubernetes-upgrades` is not the empty set, specify one of the listed versions as the `--kubernetes-version` in the update cluster command.
3. Run the update cluster command.

   If you specify the `--defined-tags` or `--freeform-tags` options, do not specify values for the OraclePCA-OKE.cluster_id 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
```

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

   Enter the cluster name, and select 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
   ```

6 Managing OKE Cluster Add-ons

Cluster add-ons are components that you can choose to deploy on a Kubernetes cluster. Cluster add-ons extend core Kubernetes functionality and improve cluster manageability and performance.

This chapter describes how to install the WebLogic Kubernetes Operator add-on, which supports running WebLogic Server and Fusion Middleware Infrastructure domains on Kubernetes. For detailed information about the WebLogic Kubernetes Operator, refer to the public operator documentation at https://github.com/oracle/weblogic-kubernetes-operator.

Installing the WebLogic Kubernetes Operator Add-on

You can enable the WebLogic Kubernetes Operator add-on when you create a cluster or for an existing cluster.

> **Note:**
>
> To bring the WebLogic Server to the running state, create additional rules in separate WebLogic Server security lists for the control plane and worker subnets, and for the pod subnet if you are using VCN-Native Pod Networking. See Ports Required by WebLogic Server.

Add-on installation remains in Accepted state and waits until the cluster is in the Active state.

After the cluster is in the Active state, the WebLogic Kubernetes Operator is in Needs Attention state until a node pool is created for the cluster.

When a node-pool has been created for the cluster, the add-on is reconciled, and the add-on is in Ready state unless some other problem exists. See Add-on Reconciliation.

> **Note:**
>
> Enabling the WebLogic Kubernetes Operator add-on on a VCN-Native Pod Networking cluster requires an entry for 169.254.169.254 in `crio-noproxy` node metadata for the nodepools where the add-on pods might be scheduled. See "Proxy settings" in the OCI CLI procedure in Creating an OKE Worker Node Pool.

Install the Add-on When You Create a Cluster

To install an add-on when you create a cluster, you must use the Compute Web UI.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. Above the clusters list, select the Create Cluster button.

3. On the bottom of the first page of the Create Cluster dialog, the Add-ons section shows the available cluster add-ons. In the Create Cluster dialog, all add-ons are Disabled.
4. Select the WLS Operator (WebLogic Kubernetes Operator) add-on.
   a. **Enable**: Select the checkbox for "Enable Add-On WLS Operator" to deploy and enable the add-on for this cluster.
   b. **Add-on version updates**: Select how you want the version of the add-on to be updated as newer versions of the add-on become available and as newer versions of Kubernetes are supported for OKE. Select either Automatic Updates or Choose a Version. See descriptions of these options in Version Updates for Add-ons.

      If you select Choose a Version, then you must select a version from the list.
   c. **Configurations**: Select the Add configuration button to select a configuration option and specify a value. See the descriptions in Configuration Parameters for the WebLogic Kubernetes Operator Add-on.

      Select the Add configuration button to set another configuration parameter.

Install the Add-on for an Existing Cluster

Outside Certificates

If you want to install the WebLogic Kubernetes Operator add-on on an existing cluster that is using a certificate that is not the certificate that is specific to the Private Cloud Appliance, perform the following steps on the cluster where you want to install the add-on:

1. Perform certificate rotation as described in Updating the Certificate Authority Bundle.
2. Perform any updates to node pool configuration that are required, such as boot volume size change or shape changes, for example.
3. Cycle worker nodes as described in Node Cycling an OKE Node Pool.
4. Enable or install the WebLogic Kubernetes Operator add-on as described in this procedure.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. In the clusters list, select the name of the cluster in which you want to install the add-on.
3. On the cluster details page, scroll to the Resources section, and select Add-ons.
4. In the add-ons list, for the WLS Operator add-on, select the Actions menu, and select Edit. On the WLS Operator dialog, select the Enable Add-on WLS Operator checkbox to do one of the following:
   - Deploy and enable the WebLogic Kubernetes Operator add-on if the add-on has not been enabled on this cluster before.
   - Enable the WebLogic Kubernetes Operator add-on if the add-on was previously deployed for this cluster but is currently disabled.
5. Configure the add-on.
   a. **Add-on version updates**: Select the method you want to use to update the version of the add-on as newer versions of the add-on become available and as newer versions of Kubernetes are supported for OKE: either Automatic Updates or Choose a Version. See descriptions of these options in Version Updates for Add-ons.

      If you select Choose a Version, then you must select a version from the list.

b. **Add-on configuration**: Select the Add configuration button to select a configuration option and specify a value. See the descriptions in Configuration Parameters for the WebLogic Kubernetes Operator Add-on.

   To set another configuration parameter, select the Add configuration button.

6. Select the Save Changes button in the dialog.

Using the OCI CLI

1. Get the OCID of the cluster for which you want to install an add-on: `oci ce cluster list`
2. Construct an argument for the `--configurations` option.

   Use the `--configurations` option to specify one or more key/value pairs in JSON format to pass as arguments to the cluster add-on.

   For descriptions of the configuration parameters, see Configuration Parameters for the WebLogic Kubernetes Operator Add-on.

   The inline syntax is shown in the example in the next step of this procedure. You might find it easier to use a file:

```
--configurations file://./weblogic-cfg.json
```

   The format and content of the configuration file is given by the following command:

```
$ oci ce cluster install-addon --generate-param-json-input configurations
[
  {
    "key": "string",
    "value": "string"
  },
  {
    "key": "string",
    "value": "string"
  }
]
```

   In the following example, both `requests` and `limits` are specified because the memory limit is lower than the default memory request. If a limit is less than the corresponding request, the deployment will fail.

   Double quotation marks within a value must be escaped with a single backslash.

```
[
    {
        "key": "weblogic-operator.ContainerResources",
        "value": "{
            \"requests\": {
                \"cpu\": \"250m\",
                \"memory\": \"150Mi\"
            },
            \"limits\": {
                \"cpu\": \"500m\",
                \"memory\": \"200Mi\"
            }
        }"
    },
    {
        "key": "weblogic-operator-webhook.ContainerResources",
        "value": "{
            \"limits\": {
                \"cpu\": \"150m\",
                \"memory\": \"200Mi\"
```

```
            }
        }"
    },
    {
        "key": "numOfReplicas",
        "value": "1"
    }
]
```

3. Run the install add-on command.

   Syntax:

   ```
   $ oci ce cluster install-addon --cluster-id cluster_OCID \
   --addon-name addon_name
   ```

   Example:

   If you specify a version, you are selecting the "Stay on the specific version" option for updating the add-on version, described in Version Updates for Add-ons. If you set the version to null, or you omit the `--version-parameterconflict` option, you are selecting the default behavior "Automatically update the add-on."

   Note that the version string must begin with a "v".

   Enclose the configurations argument in single quotation marks so that you do not need to escape every double quotation mark in the argument value.

   ```
   $ oci ce cluster install-addon --cluster-id ocid1.cluster.unique_ID \
   --addon-name WeblogicKubernetesOperator --version-parameterconflict "v4.2.13" \
   --configurations '[{"key": "weblogic-operator.ContainerResources", "value":
   "{\"limits\": {\"cpu\": \"500m\", \"memory\": \"512Mi\"}}"}, \
   {"key": "weblogic-operator-webhook.ContainerResources", "value": "{\"limits\":
   {\"cpu\": \"150m\", \"memory\": \"200Mi\"}}"}]'

   {
    "opc-work-request-id": "ocid1.cccworkrequest.unique_ID"
   }
   ```

Version Updates for Add-ons

When you enable a cluster add-on, you can choose one of the following options for updating the add-on version:

- (Default) Automatically update the add-on when new versions become available.

  The newest version of the add-on that supports the Kubernetes version that is specified for the cluster is deployed when you install the add-on. When a newer version of the add-on is released, the add-on is automatically updated if the new add-on version is compatible with the versions of Kubernetes that are supported by OKE at that time and the version of Kubernetes that the cluster is running.

  Best practice is to keep your clusters upgraded so that they are always running versions of Kubernetes that are listed as currently supported by OKE. See Supported Versions of Kubernetes and Updating an OKE Cluster.

- Stay on the specific version of the add-on that you select until you change it.

  If you specify that you want to choose the version of the add-on to deploy, the version you choose is enabled. Ensure that the add-on version is compatible with the Kubernetes version that you have selected for the cluster or that is already running on the cluster.

  When you use the Compute Web UI, you select the version from a list. All versions on the list are compatible with the Kubernetes version that you have selected for the cluster or that is already running on the cluster.

When you use the OCI CLI, use the following `addon-option list` commands to get the information you need before you run the `cluster install-addon` command.

List available versions of all cluster add-ons that are supported on the specified Kubernetes version.

```
$ oci ce addon-option list --kubernetes-version v1.29.9
```

List available versions of the specified cluster add-on that are supported on the specified Kubernetes version.

```
$ oci ce addon-option list --kubernetes-version v1.29.9 --addon-name 
WeblogicKubernetesOperator
```

Configuration Parameters for the WebLogic Kubernetes Operator Add-on

The following configuration parameters are available for the WebLogic Kubernetes Operator cluster add-on.

> **Note:**
>
> For weblogic-operator container resources and weblogic-operator-webhook container resources, if you set a limit without specifying a request, and the limit is less than the default request, the deployment will fail.

Use the values of the weblogic-operator container resources parameter and the weblogic-operator-webhook container resources parameter to determine the maximum number of replicas you can specify. The values of these parameters could be the default values shown in the following table or different values that you requested when you enabled the WebLogic Kubernetes Operator.

Example:

A worker node with 6 Gb RAM and 2 OCPUs could accommodate 12 pods if you don't count other cluster-related or custom pods. Each WebLogic Server pod could require 250m/768Mi (cpu/memory), and Flannel Overlay pods could require 100m/50Mi. Best practice is not to exceed 70% of CPU/memory usage per node. Considering only weblogic-operator pods, 8 would be ideal to leave room for system daemons or fluctuating workloads.

Since you also must allocate weblogic-operator-webhook pods, it would be better to schedule a maximum of 6 replicas per weblogic-operator pod per node, leaving room for a maximum of 8 weblogic-operator-webhook pods,

In this example (a worker node with 6 Gb RAM and 2 OCPUs), 6 replicas per node (12 for a 2 worker node cluster) is the best configuration.

This calculation of maximum number of replicas varies for each case, depending on the WebLogic requirements, the size of the node pools, and the shape configuration (CPU and memory) for each node pool.

Parameter Name Compute Web UI OCI CLI	Description
numOfReplicas `numOfReplicas`	(Required) The integer number of replicas of the add-on deployment.

Parameter Name Compute Web UI OCI CLI	Description
weblogic-operator container resources `weblogic-operator.ContainerResources`	(Optional) These are resource values for the main WebLogic Operator container. The resource quantities that the add-on containers request, and the resource usage limits that the add-on containers cannot exceed. See Resource Management for Pods and Containers in the Kubernetes documentation. If you do not specify a request, the default request values are: • cpu: 250m • memory: 512Mi If you set a usage limit, you must set a limit equal to or greater than these default resource request values or the request values that you specified.
weblogic-operator-webhook container resources `weblogic-operator-webhook.ContainerResources`	(Optional) These are resource values for the webhook container used by the operator. The resource quantities that the add-on containers request, and the resource usage limits that the add-on containers cannot exceed. If you do not specify a request, the default request values are: • cpu: 100m • memory: 100Mi If you set a usage limit, you must set a limit equal to or greater than these default resource request values or the request values that you specified.

Ports Required by WebLogic Server

This section describes additional security rules needed to specify ports that are required to bring the WebLogic Server to the running state. Create additional rules in separate WebLogic Server security lists for the control plane and worker subnets, and for the pod subnet if you are using VCN-Native Pod Networking.

The following rules are for the control plane subnet. These rules are used for both Flannel Overlay networking and VCN-Native Pod Networking.

Table 6-1 WebLogic Server Security Rules for the Control Plane Subnet

Compute Web UI property	OCI CLI property

Ingress Rule 1
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: 8084
• Description: "This service port is the default for the WebLogic Server Console and is used to manage WebLogic Server domains."
	Ingress Rule 1
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `8084`
	  – `min`: `8084`
	• `description`: "This service port is the default for the WebLogic Server Console and is used to manage WebLogic Server domains."

Ingress Rule 2
• Stateless: uncheck the box
• Ingress CIDR: ***worker_cidr***
• IP Protocol: UDP
  – Destination Port Range: 8472
• Description: "WebLogic Server administration."
	Ingress Rule 2
	• `isStateless`: `false`
	• `source`: ***worker_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `17`
	• `udpOptions`
	  `destinationPortRange`
	  – `max`: `8472`
	  – `min`: `8472`
	• `description`: "WebLogic Server administration."

The following rules are for the worker subnet. These rules are used for both Flannel Overlay networking and VCN-Native Pod Networking.

Table 6-2 WebLogic Server Security Rules for the Worker Subnet

Compute Web UI property	OCI CLI property

Ingress Rule 1
• Stateless: uncheck the box
• Ingress CIDR: ***kmi_cidr***
• IP Protocol: TCP
  – Destination Port Range: 8084
• Description: "This service port is the default for the WebLogic Server Console and is used to manage WebLogic Server domains."
	Ingress Rule 1
	• `isStateless`: `false`
	• `source`: ***kmi_cidr***
	• `sourceType`: `CIDR_BLOCK`
	• `protocol`: `6`
	• `tcpOptions`
	  `destinationPortRange`
	  – `max`: `8084`
	  – `min`: `8084`
	• `description`: "This service port is the default for the WebLogic Server Console and is used to manage WebLogic Server domains."

Table 6-2 (Cont.) WebLogic Server Security Rules for the Worker Subnet

Compute Web UI property	OCI CLI property
Ingress Rule 2 • Stateless: uncheck the box • Ingress CIDR: ***kmi_cidr*** • IP Protocol: UDP – Destination Port Range: 8472 • Description: "WebLogic Server administration."	**Ingress Rule 2** • `isStateless`: `false` • `source`: ***kmi_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `8472` – `min`: `8472` • `description`: "WebLogic Server administration."
Ingress Rule 3 • Stateless: uncheck the box • Ingress CIDR: ***worker_cidr*** • IP Protocol: UDP – Destination Port Range: 7001-9000 • Description: "These ports are used by WebLogic Server."	**Ingress Rule 3** • `isStateless`: `false` • `source`: ***worker_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `9000` – `min`: `7001` • `description`: "These ports are used by WebLogic Server."

The following rules are for the pod subnet. These rules are used for VCN-Native Pod Networking.

Table 6-3 WebLogic Server Security Rules for the Pod Subnet

Compute Web UI property	OCI CLI property
Ingress Rule 1 • Stateless: uncheck the box • Ingress CIDR: ***kmi_cidr*** • IP Protocol: TCP – Destination Port Range: 8084 • Description: "This service port is the default for the WebLogic Server Console and is used to manage WebLogic Server domains."	**Ingress Rule 1** • `isStateless`: `false` • `source`: ***kmi_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `6` • `tcpOptions` `destinationPortRange` – `max`: `8084` – `min`: `8084` • `description`: "This service port is the default for the WebLogic Server Console and is used to manage WebLogic Server domains."

Table 6-3 (Cont.) WebLogic Server Security Rules for the Pod Subnet

Compute Web UI property	OCI CLI property
Ingress Rule 2 • Stateless: uncheck the box • Ingress CIDR: ***worker_cidr*** • IP Protocol: UDP – Destination Port Range: 8472 • Description: "WebLogic Server administration."	**Ingress Rule 2** • `isStateless`: `false` • `source`: ***worker_cidr*** • `sourceType`: `CIDR_BLOCK` • `protocol`: `17` • `udpOptions` `destinationPortRange` – `max`: `8472` – `min`: `8472` • `description`: "WebLogic Server administration."

Viewing OKE Cluster Add-ons

This topic describes how to list all add-ons for a cluster and how to view more information about a specified cluster add-on.

To list versions of an add-on that are supported on a specific version of Kubernetes, use the OCI CLI, or use the Edit Add-on option in the Compute Web UI.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. In the clusters list, select the name of the cluster for which you want to view add-ons.
3. On the cluster details page, scroll to the Resources section, and select Add-ons.

   The list table shows the name of the cluster add-on, whether automatic updates are Enabled, the lifecycle state of the add-on (for example, Active, Updating, Disabled), and the current version of the add-on.
4. To show more information, select the Actions menu, and select the Edit Add-on option.

   On the dialog, you can view the current add-on configuration and change the configuration. You can enable the add-on, select either Automatically Update or Choose a version, and specify configuration parameters.

Using the OCI CLI

1. Get the OCID of the cluster for which you want to view add-ons: `oci ce cluster list`
2. List all add-ons in the specified cluster.

   ```
   $ oci ce cluster list-addons --cluster-id ocid1.cluster.unique_ID
   ```

   For each add-on that is available to the specified cluster, the output shows the name of the add-on, the current installed version, and the lifecycle state.
3. For more information about a specific add-on, run the get add-on command.

   The output is the same as for the `list-addons` command except that `get-addon` also shows the configuration values.

If the add-on lifecycle state is NEEDS_ATTENTION, see Add-on Reconciliation.

Example:

```
$ oci ce cluster get-addon --cluster-id ocid1.cluster.unique_ID \
--addon-name WeblogicKubernetesOperator
{
  "data": {
    "addon-error": {
        "code": null,
        "message": null,
        "status": null
    },
    "configurations": [
      {
        "key": "numOfReplicas",
        "value": "0"
      },
      {
        "key": "weblogic-operator.ContainerResources",
        "value": "{'limits': {'cpu': '500m', 'memory': '200Mi'}}"
      },
      {
        "key": "weblogic-operator-webhook.ContainerResources",
        "value": "{'limits': {'cpu': '200m', 'memory': '300Mi'}}"
      }
    ],
    "current-installed-version": "v4.2.13",
    "lifecycle-state": "ACTIVE",
    "name": "WeblogicKubernetesOperator",
    "time-created": "2025-02-26T01:41:52.020696+00:00",
    "version": null
  },
  "etag": "5f3eef22-eb32-5c2c-774c-c7a98836a13a"
}
```

Add-on Reconciliation

The OKE service includes a reconciliation process that periodically evaluates the state of the add-on and updates the add-on if necessary.

> **Note:**
>
> You should not install, configure, update, or delete add-ons manually. Use the OKE service installation, configuration, update, and delete interfaces.

The reconciliation process behaves differently depending on the state of the add-on.

Active State

For add-ons in ACTIVE state, the reconciliation process runs every twelve hours. If resources have been manually deleted, the process detects the change and attempts recovery. Full recovery is not guaranteed.

If full recovery is successful, the add-on state returns to ACTIVE.

If recovery is partial or fails, the add-on state changes to NEEDS_ATTENTION.

Needs Attention State

For add-ons in NEEDS_ATTENTION state, the reconciliation process runs every few minutes, not every twelve hours, while the add-on remains in the NEEDS_ATTENTION state. The process checks whether all deployments associated with the add-on are ready, whether all pods are healthy. The interval between reconciliation process runs is a little longer each time.

Some issues, such as unschedulable nodes, might resolve during the reconciliation process. Other issues, such as configuration problems, require user intervention to fix. If the add-on remains in NEEDS_ATTENTION state after the reconciliation process has run, try to identify the issues. Check the K8s_app application in Grafana or check the state of the add-on operator manually.

Recovery actions that might be appropriate for a user to take include the following:

- Ensure that node pools in the cluster have at least one node available. If no nodes are available, the add-on pods cannot be scheduled, the add-on cannot be deployed.
- Ensure the configuration values and other settings are correct.
- Update the add-on as needed. Updating the add-on will trigger another reconciliation process.
- Disable and reinstall the add-on.

If the add-on recovers, the add-on is transitioned back to ACTIVE state.

If the add-on remains unhealthy, the system schedules the next check.

The reconciliation process continues to run for approximately 12.5 hours if the add-on remains in NEEDS_ATTENTION state. After the reconciliation process stops running, the add-on remains in NEEDS_ATTENTION state indefinitely. After approximately 30 minutes, the work request moves to FAILED state.

The following example shows an add-on in NEEDS_ATTENTION state.

```
$ oci ce cluster get-addon --cluster-id ocid1.cluster.unique_ID \
--addon-name WeblogicKubernetesOperator
{
  "data": {
    "addon-error": {
      "code": "409",
      "message": "Incorrect state for CR",
      "status": "IncorrectState"
    },
    "configurations": [
      {
        "key": "numOfReplicas",
        "value": "0"
      },
      {
        "key": "weblogic-operator.ContainerResources",
        "value": "{'limits': {'cpu': '500m', 'memory': '200Mi'}}"
      },
      {
        "key": "weblogic-operator-webhook.ContainerResources",
        "value": "{'limits': {'cpu': '200m', 'memory': '300Mi'}}"
      }
    ],
    "current-installed-version": "",
    "lifecycle-state": "NEEDS_ATTENTION",
    "name": "WeblogicKubernetesOperator",
```

```
    "time-created": "2025-02-26T01:41:52.020696+00:00",
    "version": null
  },
  "etag": "5f3eef22-eb32-5c2c-774c-c7a98836a13a"
}
```

Updating the WebLogic Kubernetes Operator Add-on

You can update how you want the version of the WebLogic Kubernetes Operator add-on to be updated, and you can update the number of replicas and the resource usage limits as described in Installing the WebLogic Kubernetes Operator Add-on.

Add-on update remains in Accepted state and waits until the cluster is in the Active state.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. In the clusters list, select the name of the cluster for which you want to update an add-on.
3. On the cluster details page, scroll to the Resources section, and select Add-ons.
4. In the add-ons list, for the WLS Operator add-on, select the Actions menu, and select the Edit Add-on option.
5. In the dialog that opens, configure the add-on.
   a. **Add-on version updates**: Select how you want the version of the add-on to be updated as newer versions of the add-on become available and as newer versions of Kubernetes are supported for OKE: either Automatic Updates or Choose a Version. See descriptions of these options in Version Updates for Add-ons.

      If you select Choose a Version, then you must select a version from the list.
   b. **Add-on configuration**: Optionally select a configuration parameter and a value for that parameter. See the descriptions in Configuration Parameters for the WebLogic Kubernetes Operator Add-on.

      To set another configuration parameter, select the Add configuration button.
6. Select the Save Changes button in the dialog.

Using the OCI CLI

1. Get the OCID of the cluster for which you want to update an add-on: `oci ce cluster list`
2. Run the update add-on command.

   Syntax:

   ```
   $ oci ce cluster update-addon --cluster-id cluster_OCID \
   --addon-name addon_name
   ```

   Example:

   ```
   $ oci ce cluster update-addon --cluster-id ocid1.cluster.unique_ID \
   --addon-name WeblogicKubernetesOperator \
   --configurations file://./weblogic-cfg.json --force

   {
    "opc-work-request-id": "ocid1.cccworkrequest.unique_ID"
   }
   ```

Disabling and Removing OKE Cluster Add-ons

When you disable an add-on, it is disabled and deleted from the cluster. If you subsequently enable the add-on, the add-on is reinstalled.

Add-on delete remains in Accepted state and waits until the cluster is in the Active state.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. In the clusters list, select the name of the cluster for which you want to delete the add-on.
3. On the cluster details page, scroll to the Resources section, and select Add-ons.
4. In the add-ons list, for the WLS Operator add-on, select the Actions menu, and select the Edit option.
5. In the edit dialog, deselect (uncheck) the Enable Add-on option to disable and remove the WebLogic Kubernetes Operator add-on for this cluster.
6. Select the Save Changes button in the dialog.

Using the OCI CLI

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

   Example:

   ```
   $ oci ce cluster disable-addon --cluster-id ocid1.cluster.unique_ID \
   --addon-name WeblogicKubernetesOperator
   ```

   The add-on is disabled and removed from the cluster: The `--is-remove-existing-add-on` option is ignored.

7

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.

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 configure proxy settings, use the OCI CLI or OCI API to set the proxy in node metadata. If the cluster is using VCN-Native Pod Networking, add 169.254.169.254 to the noproxy setting.

Using the Compute Web UI

1. On the dashboard, select 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. Select 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 select Node Pools.
4. On the Node Pools list, select 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, select 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. The primary VNIC from the worker subnet will be attached to this NSG.
   - **Placement configuration**
     – **Worker Node Subnet**: Select a subnet that has configuration like the "worker" subnet described in Creating a Flannel Overlay Worker Subnet or Creating a VCN-Native Pod Networking Worker Subnet. For a public cluster, create the NAT private version of the "worker" subnet. For a private cluster, create the VCN-only private version of the "worker" subnet. Select only one subnet. The subnet must have rules set to communicate with the control plane endpoint. The subnet must use a

private route table and must have a security list like the worker-seclist security list described in Creating a Flannel Overlay Worker Subnet or Creating a VCN-Native Pod Networking 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. If more than one image has the exact same Kubernetes version, select the newest image, according to the date in the image name.

     If the image that you want to use is not listed, use the OCI CLI or OCI API and specify the OCID of the image. To get the OCID of the image you want, use the `ce node-pool get` command for a node pool where you used this image before.

     > **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.

  > **Note:**
  >
  > Allocate at least 2 OCPUs and 32 GB memory for every 10 running pods. You might need to allocate more resources, depending on the workloads that are planned. See Resource Management for Pods and Containers.

- **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.

- **Pod Communication** (VCN-Native Pod Networking clusters only)

  Pod Communication Subnet: Select a subnet that has configuration like the "pod" subnet described in Creating a VCN-Native Pod Networking Pod Subnet.

  Number of Pods per node: The maximum number of pods that you want to run on a single worker node in a node pool. The default value is 31. You can enter a number from 1 to 110. The number of VNICs allowed by the shape you specify (see "Shape" above) limits this maximum pods number. See Node Shapes and Number of Pods. To conserve the pod subnet's address space, reduce the maximum number of pods you want to run on a single worker node. This reduces the number of IP addresses that are pre-allocated in the pod subnet.

  If you check the box to Use Security Rules in Network Security Group (NSG), select 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. Secondary VNICs from the pod subnet will be attached to this NSG.

- **Cordon and Drain**: (Optional) Enter the number of minutes of eviction grace duration, or use the arrows to decrease or increase the number of minutes of eviction grace duration. The maximum value and default value is 60 minutes.

  – Private Cloud Appliance Release 3.0.2-b1261765. Specify an integer from 0 to 60. If you enter 0, the value will be converted to 0.333 because 20 seconds is the minimum eviction grace duration. If you then select the up arrow, the value will change to 1.

  – Private Cloud Appliance Release 3.0.2-b1185392. Specify an integer from 1 to 60.

  You cannot deselect "Force terminate after grace period." Nodes are deleted after their pods are evicted or at the end of the eviction grace duration, even if not all pods are evicted.

  For descriptions of cordon and drain and eviction grace duration, see Node and node pool deletion settings in "Using the OCI CLI".

- **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.

- **Kubernetes Labels**: Select the Add Kubernetes Label button and enter a key name and value. You can use these labels to target pods for scheduling on specific nodes or groups of nodes. See the description and example in the OCI CLI procedure.

- **Node Pool Tags**: Defined or free-form tags for the node pool resource.

  > **Note:**
  >
  > Do not specify values for the OraclePCA-OKE.cluster_id 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.

- **Node Tags**: Defined or free-form tags that are applied to every node in the node pool.

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

6. Select the Add Node Pool button.

   The details page for the node pool is displayed. Scroll to the Resources section and select Work Requests to see the progress of the node pool creation and see nodes being added to the Nodes list. The work request status will be Accepted until the cluster is in either Active state or Failed state.

   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 worker subnet OCID 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.
   - (VCN-Native Pod Networking clusters only) The OCID of the pod subnet. See Creating a VCN-Native Pod Networking Pod Subnet. See also the description in Pod Communication in the preceding Compute Web UI procedure. Use the `--pod-subnet-ids` option. Although the `--pod-subnet-ids` option value is an array, you can specify only one pod subnet OCID.

     The maximum number of pods that you want to run on a single worker node in a node pool. Use the `--max-pods-per-node` option. The default value is 31. You can enter a number from 1 to 110. The number of VNICs allowed by the shape you specify (see "The name of the shape" below) limits this maximum pods number. See Node Shapes and Number of Pods. To conserve the pod subnet's address space, reduce the maximum number of pods you want to run on a single worker node. This reduces the number of IP addresses that are pre-allocated in the pod subnet.

     (Optional) The OCID of the Network Security Group to use for the pods in this node pool. Use the `--pod-nsg-ids` option. You can specify up to five NSGs.
   - 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
```

If the image that you want to use is not listed, you can get the OCID of the image from the output of the `ce node-pool get` command for a node pool where you used this image before.

> **Note:**
>
> The image that you specify must have "`-OKE-`" in its `display-name` and 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 1.29.9.

```
"display-name": "uln-pca-Oracle-Linux8-OKE-1.29.9-20250325.oci"
```

If more than one image has the exact same Kubernetes version, select the newest image, according to the date in the image name.

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}'
  ```

  > **Note:**
  >
  > Allocate at least 2 OCPUs and 32 GB memory for every 10 running pods. You might need to allocate more resources, depending on the workloads that are planned. See Resource Management for Pods and Containers.

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. Use the `--nsg-ids` option. Do not specify more than one NSG.
- (Optional) Labels. Setting labels on nodes enables you to target pods for scheduling on specific nodes or groups of nodes. Use this functionality to ensure that specific pods only run on nodes with certain isolation, security, or regulatory properties.

  Use the `--initial-node-labels` option to add labels to the nodes. Labels are a list of key/value pairs to add to nodes after they join the Kubernetes cluster. See "Metadata Key Limits" in the Compute Instance Concepts chapter of the *Oracle Private Cloud Appliance Concepts Guide* for information about metadata limits.

  The following is an example label to apply to the nodes in the node pool:

  ```
  --initial-node-labels '[{"key":"disktype","value":"ssd"}]
  ```

  An easy way to select nodes based on their labels is to use `nodeSelector` in the pod configuration. Kubernetes only schedules the pod onto nodes that have each of the labels that are specified in the `nodeSelector` section.

  The following example excerpt from a pod configuration specifies that pods that use this configuration must be run on nodes that have the `ssd` disk type label:

  ```
  nodeSelector:
    disktype: ssd
  ```

- (Optional) Node metadata. Use the `--node-metadata` option to attach custom user data to nodes. See the following proxy settings item for a specific example.

  See "Metadata Key Limits" in the Compute Instance Concepts chapter of the *Oracle Private Cloud Appliance Concepts Guide* for information about metadata limits. The maximum size of node metadata is 32,000 bytes.

- (Optional) Proxy settings. If your network requires proxy settings to enable worker nodes to reach outside registries or repositories, for example, create an argument for the `--node-metadata` option.

  In the `--node-metadata` option argument, provide values for `crio-proxy` and `crio-noproxy` as shown in the following example file argument:

  ```
  {
    "crio-proxy": "http://your_proxy.your_domain_name:your_port",
    "crio-noproxy":
  "localhost,127.0.0.1,your_domain_name,ocir.io,Kubernetes_cidr,pods_cidr"
  }
  ```

  If the cluster is using VCN-Native Pod Networking, add 169.254.169.254 to the noproxy setting, as in the following example:

  ```
  "crio-noproxy":
  "localhost,127.0.0.1,your_domain_name,ocir.io,Kubernetes_cidr,pods_cidr,169.254.1
  69.254"
  ```

- (Optional) Node and node pool deletion settings. You can specify how to handle node deletion when you delete a node pool, delete a specified node, decrement the size of the node pool, or change the node pool nodes placement configuration. These node deletion parameters can also be set or changed when you update the node pool, delete a specified node, or delete the node pool.

To specify node pool deletion settings, create an argument for the `--node-eviction-node-pool-settings` option. You can specify the eviction grace duration (`evictionGraceDuration`) for nodes. Nodes are always deleted after their pods are evicted or at the end of the eviction grace duration.

  – Eviction grace duration. This value specifies the amount of time to allow to cordon and drain worker nodes.

    A node that is cordoned cannot have new pods placed on it. Existing pods on that node are not affected.

    When a node is drained, each pod's containers terminate gracefully and perform any necessary cleanup.

    The eviction grace duration value is expressed in ISO 8601 format: for example, PT45S, PT20M, or PT39M21S. The default value and the maximum value are 60 minutes (PT60M). The minimum value is 20 seconds (PT20S). OKE always attempts to drain nodes for at least 20 seconds.

  – Force delete. Nodes are always deleted after their pods are evicted or at the end of the eviction grace duration. After the default or specified eviction grace duration, the node is deleted, even if one or more pod containers are not completely drained.

The following shows an example argument for the `--node-eviction-node-pool-settings` option. If you include the `isForceDeleteAfterGraceDuration` property, then its value must be `true`. Nodes are always deleted after their pods are evicted or at the end of the eviction grace duration.

```
--node-eviction-node-pool-settings '{"evictionGraceDuration": "PT30M",
"isForceDeleteAfterGraceDuration": true}'
```

> **Note:**
>
> If you use Terraform and you specify `node_eviction_node_pool_settings`, then you must explicitly set `is_force_delete_after_grace_duration` to `true`, even though true is the default value. The `is_force_delete_after_grace_duration` property setting is not optional if you are using Terraform.

- (Optional) Tags. Add defined or free-form tags for the node pool resource by using the `--defined-tags` or `--freeform-tags` options. Do not specify values for the OraclePCA-OKE.cluster_id 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.cluster_id 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`. The `--pod-subnet-ids` option is only applicable if the cluster uses VCN-Native Pod Networking.

```
$ 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"}]' \
--pod-subnet-ids '["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
```

The work request status will be `ACCEPTED` until the cluster is in either Active state or Failed state.

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. Create a service to expose containerized applications outside the Private Cloud Appliance. See Exposing Containerized Applications.
3. Create persistent storage for applications to use. See Adding Storage for Containerized Applications.

Updating an OKE Node Pool

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.

When you update node properties, by default existing nodes are not updated. The updated values only apply 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 replace existing nodes with new nodes that use these updated settings, see Node Cycling an OKE Node Pool.

Using the Compute Web UI

1. On the dashboard, select Containers / View Kubernetes Clusters (OKE).
2. Select 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 select Node Pools.
4. For the node pool that you want to update in the Node Pools list, select the Actions menu and select 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.

   Cordon and Drain settings

   Enter the number of minutes of eviction grace duration, or use the arrows to decrease or increase the number of minutes of eviction grace duration. The maximum value and default value is 60 minutes.

   - Private Cloud Appliance Release 3.0.2-b1261765. You can specify an integer from 0 to 60. If you enter 0, the value will be converted to 0.333 because 20 seconds is the minimum eviction grace duration. The field will show a decimal value if the existing value was set in ISO 8601 format and included a seconds value. For example, an existing value of PT45S will show as 0.45, PT20M will show as 20, and PT39M21S will show as 39.35.
   - Private Cloud Appliance Release 3.0.2-b1185392. Specify an integer from 1 to 60. If the existing value was set in ISO 8601 format and includes a seconds value, that value will display as the next higher integer number of minutes. The seconds value will still be used, even though it does not display.

   You cannot deselect "Force terminate after grace period." For descriptions of cordon and drain and eviction grace duration, see Node and node pool deletion settings in "Using the OCI CLI" in Creating an OKE Worker Node Pool.

   > **Note:**
   >
   > Do not specify values for the OraclePCA-OKE.cluster_id 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, select 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.

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

To replace existing nodes with new nodes that use these updated settings, see Node Cycling an OKE Node Pool.

Using the OCI CLI

1. Get the information you need to run the command.
   - The OCID of the node pool that you want to update: `oci ce node-pool list`
   - (Optional) Node and node pool deletion settings. Use the `--node-eviction-node-pool-settings` option or the `--override-eviction-grace-duration` option to set the eviction grace duration for nodes. Nodes are always deleted after their pods are evicted or at the end of the eviction grace duration. See the description in Creating an OKE Worker Node Pool.
   - (Optional) Labels. To add labels to new nodes, use the `--initial-node-labels` option. Labels on existing nodes cannot be changed by using the `--initial-node-labels` option. Labels on existing nodes can be modified using `kubectl`. For more information about node labels, see Creating an OKE Worker Node Pool.
   - (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.cluster_id 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 tags to nodes that are newly added to the node pool, use the `--node-defined-tags` and `--node-freeform-tags` options.
2. (Optional) Create an argument for the `--node-pool-cycling-details` option, and use that option to apply these updates to all of the nodes in the node pool.

   Without the `--node-pool-cycling-details` option, the updated configuration specified in this `node-pool update` command only applies to new nodes that are created by this command or in the future, as described at the beginning of this topic.

   To replace existing nodes with new nodes that use these updated settings, specify the `--node-pool-cycling-details` option as described in Node Cycling an OKE Node Pool.
3. Run the update node pool command.

   Syntax:

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

Node Cycling an OKE Node Pool

By default when you update a node pool, only new nodes that are added during this update or that are added later receive the updates. To replace existing nodes with new nodes that use updated settings, enable the node cycling option.

Node cycling performs an in-place update of all existing nodes in the node pool to the latest specified configuration. New nodes are created, workloads moved onto them from existing nodes, current node pool updates applied, and the original nodes terminated.

You can set the maximum number of nodes that are starting or terminating at any particular time.

- Maximum surge. The maximum number of new nodes that can be starting at any time during this update operation. Set this value to avoid adding too many new nodes before existing nodes are terminated, which could incur excessive cost. The default value is 1. The maximum value is 5.
- Maximum unavailable. The maximum number of existing nodes that can be terminating at any time during this update operation. Set this value to ensure that enough nodes remain to handle the workload. The default value is 0. The maximum value is 7.

One of these values must be greater than 0.

Both of these values can be set to either a number (from 0 to the configured number of nodes in the node pool, but not greater than the maximum cited above) or a percentage (from 0% to 100%, but not a percentage that would result in a number greater than the maximum cited above). These values can be a maximum of four characters.

If you set either of these properties to a percent value that exceeds the maximum allowed number of nodes, the error message tells you the maximum allowed percent value for this node pool.

> **Note:**
>
> If the node cycling operation fails (for example, the operation times out), try re-running the operation. You might need to run the node cycling operation multiple times if the system is loaded and running at scale.

Using the Compute Web UI

Follow the Compute Web UI procedure in Updating an OKE Node Pool to update the node pool configuration.

1. On the node pool details page, click the Cycle Nodes button.
2. In the Cycle Nodes dialog, enter values for the Maximum Surge and Maximum Unavailable properties.

   See the rules at the beginning of this topic.
3. Click the Cycle Nodes button in the dialog to start the node pool update operation.

   To monitor the progress of the update operation, view the status of the associated work request.

Using the OCI CLI

1. Construct a command to update the node pool configuration as described in the Compute Web UI procedure in Updating an OKE Node Pool.
2. In that same command (not later as with the Compute Web UI procedure) include the `--node-pool-cycling-details` option.

   In addition to setting `maximumUnavailable` and `maximumSurge`, enable node cycling by setting `isNodeCyclingEnabled` to `true`. By default, `isNodeCyclingEnabled` is `false`, and node cycling will not be performed if you set only `maximumUnavailable` or `maximumSurge` and do not set `isNodeCyclingEnabled` to `true`.

```
$ oci ce node-pool update --node-pool-id ocid1.nodepool.unique_ID \
new_configuration_settings \
--node-pool-cycling-details
'{"isNodeCyclingEnabled":true,"maximumUnavailable":"value","maximumSurge":"value"}'
```

See the beginning of this topic for the possible values.

In the following example, the image is updated for all nodes in the node pool:

```
$ oci ce node-pool update --node-pool-id ocid1.nodepool.unique_ID \
--node-source-details '{"imageId":"ocid1.image.unique_ID","sourceType":"IMAGE"}' \
--node-pool-cycling-details
'{"isNodeCyclingEnabled":true,"maximumUnavailable":"5%","maximumSurge":"5%"}'
```

To monitor the progress of the update operation, view the status of the associated work request.

Find the work request OCID:

```
oci ce work-request list --compartment-id ocid1.compartment.unique_ID \
--resource-id ocid1.nodepool.unique_ID
```

Show the current state of the work request:

```
oci ce work-request get --work-request-id ocid1.workrequest.unique_ID
```

Using Node Doctor to Troubleshoot Worker Node Issues

If a cluster has a worker node that is in a state other than Active or Running, use the Node Doctor utility to troubleshoot the issues.

Node Doctor scans a worker node and reports the health status of the node. Node Doctor can do the following tasks:

- Identify potential problem areas and provide references to information to help you address those problem areas. See Print Troubleshooting Information.
- Collect node system information into a support bundle if you need help from Oracle Support. See Create a Support Bundle.

Use Node Doctor only on worker nodes. Because Node Doctor is installed on OKE images, Node Doctor is also available on cluster control plane nodes. Do not use Node Doctor on control plane nodes.

Check the Oracle Private Cloud Appliance Release Notes for the release in which Node Doctor was first delivered. If your node pools were created on that release or later, then you can proceed with the instructions in this topic. If your worker node image is from an earlier release, then that node does not have access to Node Doctor. Note that if your Private Cloud Appliance is running a release that includes Node Doctor, then you could use node cycling to update older worker node images. See Node Cycling an OKE Node Pool.

Connect to the Worker Node Using SSH

Perform the following steps to connect to the worker node that you want to troubleshoot.

1. Ensure that you have a private and public SSH key pair.

   You must have the private key that goes with the public key that was added to the node when the node was created.

2. Get the node user name. OKE images have the initial user name `opc` configured.
3. Get the IP address of the worker node that you need to troubleshoot.

The IP address is on the Networking tab of the node details page in the Compute Web UI.

- If the node has a public IP address, use the public IP address.
- If the node is on a private IP, then connect to the node via the bastion host.

  If a bastion host is not available, see Creating a Bastion.

4. Enter the following command at a shell prompt on your local system (public IP address) or on the bastion host (private IP address):

   ```
   ssh -i private_key_file username@ip-address
   ```

   - ***private_key_file***. The full path and name of the file that contains the private SSH key that goes with the public key that was added to the node when the node was created.
   - ***username***. The default user name for the node. This value probably is `opc`.
   - ***ip-address***. The node IP address that you got in Step 3.

5. Ensure that you have permission to execute the following file:

   ```
   /usr/local/bin/node-doctor.sh
   ```

Print Troubleshooting Information

While logged in to the worker node as described in Connect to the Worker Node Using SSH, enter the following command to print information that identifies potential problem areas:

```
$ sudo /usr/local/bin/node-doctor.sh --check
```

Use the following command to see more options:

```
$ sudo /usr/local/bin/node-doctor.sh --help
```

Create a Support Bundle

If you are not able to resolve the issue, use the following command to create a support bundle with relevant information for Oracle Support:

```
$ sudo /usr/local/bin/node-doctor.sh --generate
```

The support bundle is in the `/tmp` directory as `oke-support-bundle-`***`date`***`T`***`time`***`.tar`.

> **Note:**
>
> Monitor the `/tmp` directory to ensure that it does not fill up. Remove old files by using the `rm` command, for example.

See the following resources for information about uploading a bundle to a support ticket:

- Quick User Guide to Upload Files to My Oracle Support - MOS (Doc ID 1588459.1)
- How to Upload Files to Oracle Support (Doc ID 1547088.2)
- Using Support Bundles in the *Oracle Private Cloud Appliance Administrator Guide*

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, select Containers / View Kubernetes Clusters (OKE).
2. Select 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 select Node Pools.
4. Select 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 select Nodes.
6. For the node that you want to delete, select the Actions menu, and select Delete.
7. Confirm the deletion.
    a. If you do not want a new node to be automatically created to replace the deleted node, select Decrease node pool size.
    b. Check the box if you want to override the eviction grace duration in the cordon and drain settings for the node.

       Use the arrows to decrease or increase the number of minutes of eviction grace duration. See the description of this field in Updating an OKE Node Pool.

       You cannot deselect "Force terminate after grace period." The node is deleted after its pods are evicted or at the end of the eviction grace duration, even if not all pods are evicted.

       For descriptions of cordon and drain and eviction grace duration, see "Node and node pool deletion settings" in "Using the OCI CLI" in Creating an OKE Worker Node Pool.
    c. Select 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
```

You can use the `--override-eviction-grace-duration` option to set a new value for `evictionGraceDuration` for this node deletion. See the description of `--node-eviction-node-pool-settings` in Creating an OKE Worker Node Pool. For `node-pool delete-node`, this new eviction grace duration value only applies to the node being deleted.

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, select Containers / View Kubernetes Clusters (OKE).
2. Select 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 select Node Pools.
4. For the node pool that you want to delete, select the Actions menu, and select Delete.
5. Confirm the deletion.
   a. Enter the name of the node pool to confirm that you want to delete the node pool.
   b. Check the box if you want to override the eviction grace duration in the cordon and drain settings for the nodes in the pool.

      Use the arrows to decrease or increase the number of minutes of eviction grace duration. See the description of this field in Updating an OKE Node Pool.

      You cannot deselect "Force terminate after grace period." Nodes are deleted after their pods are evicted or at the end of the eviction grace duration, even if not all pods are evicted.

      For descriptions of cordon and drain and eviction grace duration, see "Node and node pool deletion settings" in "Using the OCI CLI" in Creating an OKE Worker Node Pool.
   c. Select 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
   ```

   You can use the `--override-eviction-grace-duration` option to set a new value for `evictionGraceDuration` for this node pool deletion. See the description of `--node-eviction-node-pool-settings` in Creating an OKE Worker Node Pool.

8 Exposing Containerized Applications

Do the following to expose an application deployment so that worker node applications can be reached from outside the Private Cloud Appliance:

- Create an external load balancer.
- Update ingress and egress rules as necessary to support the port requirements of your containerized applications. For example, if any application uses TCP port 3000, then an ingress rule needs to be added with port 3000.

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.

If you use the `--service-lb-defined-tags` or `--service-lb-flexible-tags` options to specify tags to be applied to external load balancers. then ensure that the applicable dynamic group includes the following policy. See Creating a Cluster Dynamic Group.

```
allow dynamic-group dynamic-group-name to use tag-namespaces in compartment compartment-name
```

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.

If you want to create a service load balancer on a private cluster (a cluster with a private worker load balancer subnet), then use the following annotation in your external load balancer template:

```
service.beta.kubernetes.io/oci-load-balancer-internal: "true"
```

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
```

9 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 the following methods:

- Block volumes. 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.
  - To provision a regular block volume, see Creating Persistent Block Volume Storage.
  - To provision a high performance block volume, see Creating Persistent High Performance Block Volume Storage.

  For information about block volumes on the Private Cloud Appliance, 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*.
- File systems. Mount file systems from 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.
  - Provision a PVC on a new file system using the CSI volume plugin. Create a storage class and a PVC. The CSI volume plugin dynamically creates both a new File Storage service file system and a new persistent volume backed by the new file system. See Creating Persistent File System Storage Using the CSI Volume Plugin.
  - Provision a PVC on an existing file system. Create a file system, mount target, PV, and PVC. See Creating Persistent File System Storage Using an Existing File System.

  For information about file systems on the Private Cloud Appliance, see the File Storage Overview chapter in the *Oracle Private Cloud Appliance Concepts Guide* and "Creating a File System, Mount Target, and Export" in the File System Storage chapter in the *Oracle Private Cloud Appliance User Guide*.

Creating Persistent Block Volume Storage

This procedure automatically creates the requested `oci-bv` storage class; you do not need to create it. This procedure starts with using the `kubectl` command to create the persistent volume claim.

If you need to provision a high performance block volume, see Creating Persistent High Performance Block Volume Storage.

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 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 High Performance Block Volume Storage

This procedure creates a high performance block volume as persistent storage. If you do not need a high performance block volume, use the instructions in Creating Persistent Block Volume Storage.

1. Create a high performance block volume using the CSI plugin specified by the `oci-bv-high` storage class definition (`provisioner: blockvolume.csi.oraclecloud.com`).

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

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

   ```
   apiVersion: storage.k8s.io/v1
   kind: StorageClass
   metadata:
     name: oci-bv-high
   provisioner: blockvolume.csi.oraclecloud.com
   parameters:
     vpusPerGB: "20"
     attachment-type: "paravirtualized"
   volumeBindingMode: WaitForFirstConsumer
   allowVolumeExpansion: true
   reclaimPolicy: Delete
   ```

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

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

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

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: mynginxclaim-high
   spec:
     storageClassName: "oci-bv-high"
     accessModes:
       - ReadWriteOnce
     resources:
       requests:
         storage: 50Gi
   ```

   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.

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

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

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

```
volumeBindingMode: WaitForFirstConsumer
```

4. 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-high` PVC as the `nginx` volume, which is mounted by the pod at `/data`:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-high
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-high
```

   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-high   Bound    csi-unique_ID   50Gi       RWO           oci-bv-high
```

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

```
$ kubectl describe pod nginx-high
```

Creating Persistent File System Storage Using the CSI Volume Plugin

This procedure provisions a PVC on a new file system using the CSI volume plugin. Use the `kubectl` command to create the storage class and persistent volume claim. The CSI volume plugin provisions the PVC on a new file system.

You can have only one mount target and one file system per VCN. You can have multiple storage classes, persistent volumes, and persistent volume claims per cluster. All storage classes, persistent volumes, and persistent volume claims in a cluster share one NFS.

1. Create a new storage class that uses the `fss.csi.oraclecloud.com` provisioner.

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

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

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
```

```
  name: fss-dyn-storage
provisioner: fss.csi.oraclecloud.com
parameters:
  availabilityDomain: AD-1
  compartmentOcid: ocid1.compartment.unique_ID
  mountTargetSubnetOcid: ocid1.subnet.unique_ID
  exportPath: AUTOSELECT
  exportOptions:
"[{\"source\":\"0.0.0.0/0\",\"requirePrivilegedSourcePort\":false,\"access\":\"READ_W
RITE\",\"identitySquash\":\"NONE\"}]"
  encryptInTransit: "false"
```

- The name for the new storage class is `fss-dyn-storage`.
- Either `mountTargetSubnetOcid` or `mountTargetOcid` is required. The value of `mountTargetSubnetOcid` is the OCID of the subnet where you want the CSI plugin to create a mount target. The value of `mountTargetOcid` is the OCID of an existing mount target. If you specify both `mountTargetSubnetOcid` and `mountTargetOcid`, `mountTargetOcid` is used and `mountTargetSubnetOcid` is ignored.

  To ensure that the mount target can be reached from worker nodes, specify the subnet that has configuration like the "worker" subnet described in Creating OKE Network Resources or create the mount target on the subnet that has configuration like the worker subnet. Ensure that TCP port 2049 to the NFS server is open on that subnet.
- The `compartmentOcid` is optional. This value is the OCID of the compartment where the new file system (and the new mount target, if `mountTargetSubnetOcid` is specified) will be created. The default value is the same compartment as the cluster.
- You must specify `AUTOSELECT` as the value for `exportPath`.
- The `exportOptions` value is the NFS export options entry within the file system export that defines the access granted to NFS clients when they connect to a mount target. The `source` can be a single IP address or CIDR block range. This value is a set of parameters in JSON format.
- The value of `encryptInTransit` specifies whether to encrypt data in transit.

2. Create a PVC to be provisioned by the new file system in the File Storage service.

   ```
   $ kubectl create -f fss-dyn-claim.yaml
   ```

   The following is the content of the `fss-dyn-claim.yaml` manifest file:

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: fss-dynamic-claim
   spec:
     accessModes:
     - ReadWriteMany
     storageClassName: "fss-dyn-storage"
     resources:
       requests:
         storage: 50Gi
   ```

3. Verify that the PVC has been bound to the new persistent volume.

   ```
   $ kubectl get pvc
   NAME              STATUS VOLUME                                        CAPACITY
   ACCESS MODES STORAGECLASS    AGE
   fss-dynamic-claim Bound  csi-fss-f6823a66-8b6f-4c42-9d1f-d25723e69257 50Gi
   RWX          fss-dyn-storage 6m47s
   ```

4. Use the new PVC when you create objects such as pods.

   The following is an example object creation:

   ```
   $ kubectl create nginx.yaml
   ```

   The following is the content of the `nginx.yaml` file. See the `claimName` on the last line:

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nginx-deployment
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         labels:
           app: nginx
       spec:
         containers:
         - name: nginx
           image: nginx_image_url
           ports:
           - name: http
             containerPort: 80
           volumeMounts:
           - name: persistent-storage
             mountPath: /usr/share/nginx/html
         volumes:
         - name: persistent-storage
           persistentVolumeClaim:
             claimName: fss-dynamic-claim
   ```

   Verify that the object is created and deployed:

   ```
   $ kubectl get deploy
   NAME             READY UP-TO-DATE AVAILABLE AGE
   nginx-deployment 3/3   3          0         104s
   ```

Creating Persistent File System Storage Using an Existing File System

This procedure provisions a PVC on an existing file system. 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 OKE Network Resources. Ensure that TCP port 2049 to the NFS server is open on that subnet.

See "Creating a Mount Target" in the File System Storage chapter in the *Oracle Private Cloud Appliance User Guide* and see "File Storage Network Ports" in the File Storage Overview chapter in the *Oracle Private Cloud Appliance Concepts 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 OKE Network Resources.

   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 fss-dynamic-app
```

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
nginx-deployment-55ff88b668-2k8rt 1/1 Running 0 4m54s
nginx-deployment-55ff88b668-79c2t 1/1 Running 0 4m54s
nginx-deployment-55ff88b668-qpdfd 1/1 Running 0 4m54s
```

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.

```
kubectl exec -it nginx-deployment-55ff88b668-2k8rt -- df -h /usr/share/nginx/html
Filesystem                                                                          Size
Used Avail Use% Mounted on
xxx.xx.xxx.xxx:/export/4fsderwh09ufyf84ei1lh3q2x8ou86pq5vcbx3aeeo060xxxxxxxxxxxxxx 67T
0     67T    0%    /usr/share/nginx/html
```

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.