Oracle Cloud Infrastructure

A Guide to Building Cloud Native Applications

Jeevan Gheevarghese Joseph
Adao Oliveira Junior
Mickey Boxell

ORACLE

Oracle Cloud Infrastructure: A Guide to Building Cloud Native Applications

```
hostname=oke-c2usfphkqza-nctxoizruoq-seoda7iqkwa-3
internal_addr=10.0.10.151
kubernetes.io/arch=amd64
kubernetes.io/hostname=10.0.10.151
kubernetes.io/os=linux
last-migration-failure=get_kubesvc_failure
name=NC
node-role.kubernetes.io/node=
node.info.ds_proxymux_client=true
node.info/compartment.name=oracle-cloudnative
node.info/kubeletVersion=v1.25
node.kubernetes.io/instance-type=VM.Standard.E3.Flex
oci.oraclecloud.com/fault-domain=FAULT-DOMAIN-2
oke.oraclecloud.com/node.info.private_subnet=false
oke.oraclecloud.com/node.info.private_worker=true
topology.kubernetes.io/region=uk-london-1
topology.kubernetes.io/zone=UK-LONDON-1-AD-2
```

SSH Keys

SSH keys are another optional node pool property. Adding your public portion of an SSH key pair to the node pool enables you to access the nodes directly through SSH. The public key is added to all worker nodes in the cluster. If you don't specify a public SSH key, you will not have SSH access to the worker nodes. Figure 4–7 shows a user adding an SSH key to the node pool using the OCI Console. Note that you cannot use SSH to directly access worker nodes in private subnets because they have private IP addresses only; they are accessible only by other resources inside the VCN. You can use the Oracle Cloud Infrastructure Bastion service to enable external SSH access to worker nodes in private subnets.

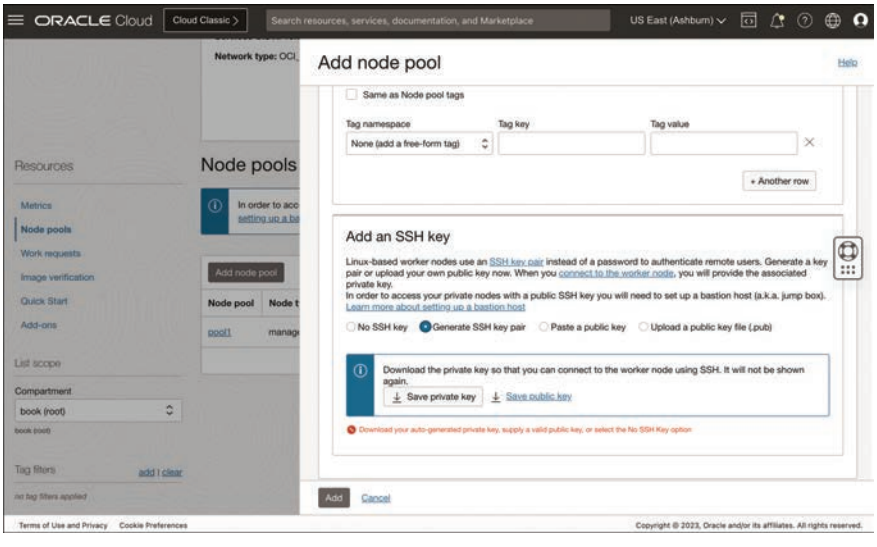

Figure 4-7 Adding an SSH Key to a Node Pool Using the OCI Console

Tagging Your Resources

Oracle Cloud Infrastructure Tagging allows you to add metadata to resources, which enables you to define keys and values and then associate them with resources. At their most basic, tags can be used to organize resources based on your business needs; however, tags opens up a lot of possibilities, from cost tracking to access control.

OCI offers tagging in two flavors: *free-form tags* and *defined tags*. Most of the advanced capabilities of tagging are applicable to defined tags, in which you create a tag namespace and then create a series of well-defined tag keys for which you can use a multitude of tag values. Although all defined tags can be used for cost analysis and usage reporting, defined tags that are designated as cost-tracking tags allow you to use them in OCI budgets. Budgets can track and forecast the cost for resources and alert you proactively when the forecast or actual consumption crosses a threshold of the budget you have set.

Consider a scenario in which you have a single cluster used by several applications or teams. You might want to implement both cost tracking and access control on a per-application (or per-team) basis on this shared infrastructure. Assume that each application is deployed on its own dedicated node pool. The applications can also create and use other resources, such as load balancers and storage, dynamically. To get an accurate estimate of the cost for each application, you can use tagging to tag the resources this application uses. Similarly, you can write access policies that restrict access for each team to only the resources that are used by their application, using tags.

Tags can be set on clusters, node pools, load balancers, and storage attachments. When creating a cluster, you can set tag defaults for the various types of resources, which you can also override with resource annotations when needed. With the resources tagged, you can implement features such as cost tracking, setting access controls, or setting budgets for the various applications. Similarly, you can use tags to keep track of the resources used by the dev, test, and prod environments for an individual application.

Tags present a flexible way of attaching additional metadata to resources: How you use these tags is up to you. This affords a tremendous amount of flexibility in the types of tags you can create and how you can leverage them.

Creating a Cluster

You can use OKE to create new Kubernetes clusters in many ways, including using the console (web UI), the CLI, automation tools such as Terraform, or the APIs directly. The console, in particular, offers two workflows to get you started: the Quick Create and Custom Create workflows (see Figure 4-8). The Quick Create workflow is the fastest way to create a new cluster. This approach automatically creates new network resources, including regional subnets for the Kubernetes API endpoint, for worker nodes, and for load balancers. This workflow is ideally suited if you are new to Kubernetes and want to get started quickly.

Note

To create a cluster, you must belong to either the tenancy's Administrators group or a group to which a policy grants the CLUSTER_MANAGE permission.

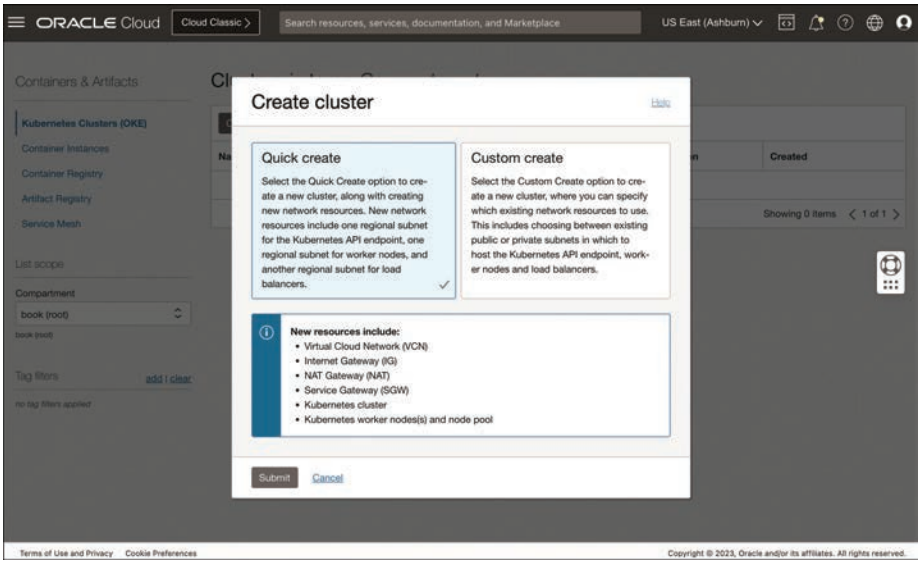

Figure 4-8 Choosing Between the Quick Create and Customer Create Workflows

Quick Create Cluster Workflow

In the Console, open the navigation menu and click Developer Services. Under Containers & Artifacts, click Kubernetes Clusters (OKE).

Step 1. Choose a compartment, and click **Create Cluster**.

Step 2. Select **Quick Create** and then click **Submit**.

Step 3. Either accept the default configurations (see Figure 4-9) or choose alternatives:

a. Give a name to your cluster.

b. Choose the compartment where you want your cluster control plane and related networking resources created.

c. Choose a Kubernetes version for your cluster control plane.

d. Specify whether you want a private or public Kubernetes API endpoint. In the case of a private subnet, the Kubernetes API endpoint will be hosted on a private subnet and assigned a private IP address. In the case of a public subnet, the Kubernetes API endpoint will be hosted on a public subnet with a public IP address automatically assigned.

e. Choose between managed and virtual nodes. Managed Kubernetes worker nodes are compute instances in your tenancy. Managed nodes come with the flexibility to configure them to meet your specific requirements, but you are responsible for upgrading Kubernetes and host OS versions and for ensuring that capacity is properly scaled. In the case of virtual nodes, the resources to execute your Kubernetes pods are provisioned dynamically, as needed, and exist in the OKE service tenancy. Virtual nodes remove the operational overhead of upgrading your data plane infrastructure and managing the capacity of clusters.

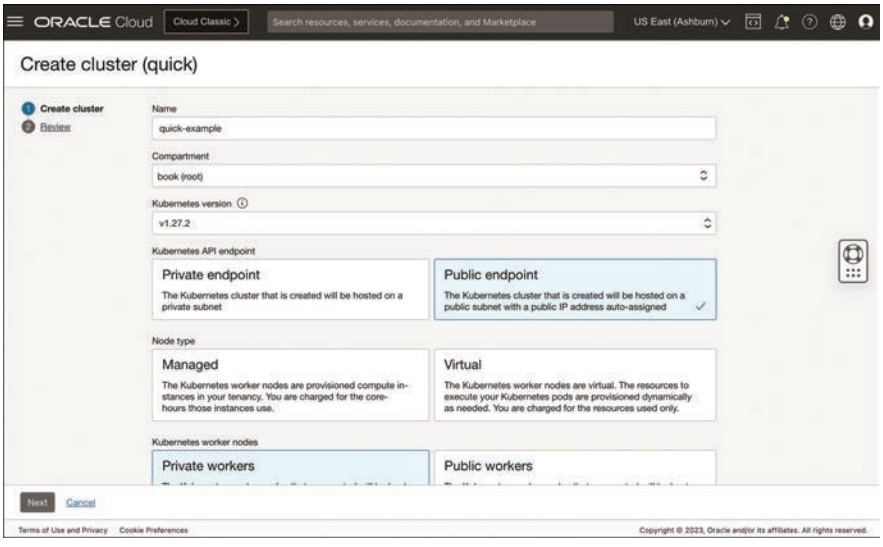

Figure 4-9 The First Step in the Quick Cluster Creation Workflow

Step 4. Depending on your chosen node type, the following steps will differ:

a. Choosing managed nodes gives you a choice between creating a private subnet or public subnet to host your Kubernetes worker nodes. It also give you a choice of image to use for your worker node hosts. These images determine the operating system and other software used for managed nodes. Selecting managed nodes also gives you expanded options for the shape of your nodes, compared to virtual nodes. Additionally, the choice of managed nodes enables you to customize the size and encryption options for the boot volumes of nodes in the node pool. Select the **Specify a Custom Boot Volume Size** check box, and enter a custom size from 50GB to 32TB to specify a custom size for the boot volume. The specified size must be larger than the default boot volume size for the selected image. If you increase the boot volume size, you must also extend the partition for the boot volume, to take advantage of

the larger size using the `oci-growfs` utility. Nodes with the VM instance chosen as the shape allow you to optionally select the **Use In-Transit encryption** check box. This is not configurable for bare metal instances. Bare metal instances that support in-transit encryption have it enabled by default. Boot volumes are encrypted by default, but you can optionally use your own Vault service encryption key to encrypt the data in this volume. To use the Vault service for your encryption needs, select the **Encrypt This Volume with a Key That You Manage** check box. Then select the Vault compartment and Vault that contain the master encryption key you want to use. Also select the master encryption key compartment and master encryption key. If you enable this option, this key is used for both data-at-rest encryption and in-transit encryption.

b. Choosing virtual nodes gives you a choice of pod shape, which determines the processor type on which to run the pod. Note that you explicitly specify the CPU and memory resource requirements for virtual nodes in the pod spec. Choosing virtual nodes also gives you the option to apply taints to nodes in the virtual node pool. Taints allow virtual nodes to repel pods, thereby ensuring that pods do not run on virtual nodes in a particular virtual node pool.

c. Both options enable you to choose the number of nodes created in the default node pool. Both options also allow you to optionally specify Kubernetes labels. These labels are added to the set of default labels already on the node and are used to target workloads at specific node pools.

Click **Next** to review the details you entered for the new cluster. If you have not selected any features restricted to enhanced clusters, you can choose to create a basic cluster. To do so, check the **Create a Basic Cluster** check box on the Review page. Otherwise, leave the box unchecked to create an enhanced cluster. Click **Create Cluster** to create the new network resources and the new cluster. Click **Close** to return to the Console.

Custom Create Cluster Workflow

The Custom Create workflow gives you the most control over creating a new cluster. It allows you to explicitly define the new cluster's properties and specify which existing network resources to use, including the existing public or private subnets in which to create the Kubernetes API endpoint, worker nodes, and load balancers. Because the Custom Create workflow opens up more features and configuration options, it is better suited for more advanced scenarios, such as when you want to bring your own networking resources or when you need to configure advanced capabilities.

Step 1. In the Console, open the navigation menu and click **Developer Services**. Under Containers & Artifacts, click **Kubernetes Clusters (OKE)**.

Step 2. Choose a compartment and click **Create Cluster**.

Step 3. Select **Custom Create** and then click **Submit**.

Step 4. You can accept the default configurations or choose alternatives (see Figure 4-10):

a. Give a name to your cluster.

b. Choose the compartment where you want your cluster control plane and related networking resources created.

c. Choose a Kubernetes version for your cluster control plane.

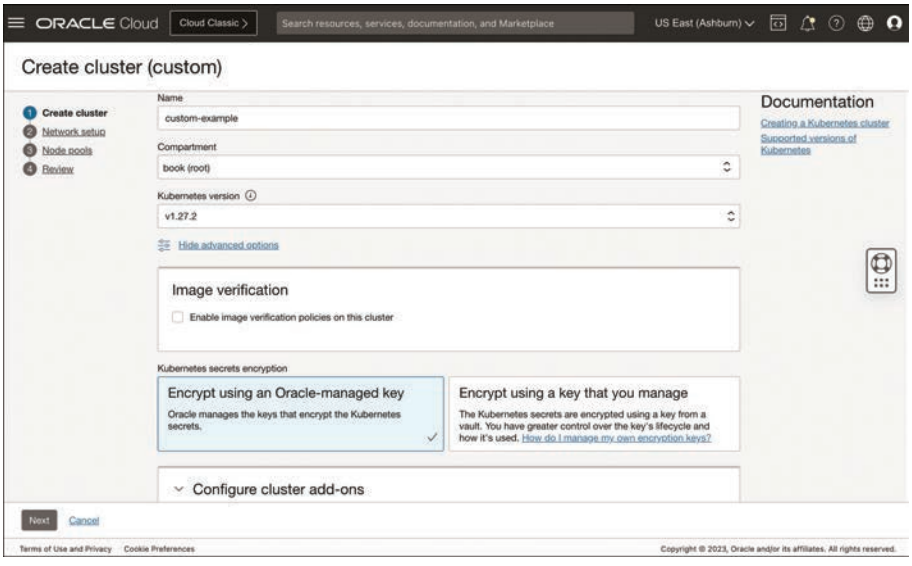

Figure 4-10 The First Step in the Custom Cluster Creation Workflow

Step 5. Click **Show Advanced Options** to view other options available for cluster configuration.

a. Specify whether to allow the deployment of images from Oracle Cloud Infrastructure Registry only if they have been signed by particular master encryption keys. To enforce the use of signed images, select **Enable Image Verification Policies on This Cluster**, and then specify the encryption key and the vault that contains it.

b. **Encrypt using an Oracle-managed key:** Encrypt Kubernetes secrets in the etcd key-value store using a master encryption key that is managed by Oracle.

c. **Encrypt using a key that you manage:** Encrypt Kubernetes secrets in the etcd key-value store using a master encryption key (stored in the Vault service) that you manage.

d. Specify how to manage cluster add-ons. Select **Configure Cluster Add-ons** to enable or disable specific add-ons, select add-on versions,