Oracle Cloud Native Environment

KubeVirt Module for Release 1.9

F93861-01
May 2024

Oracle Cloud Native Environment KubeVirt Module for Release 1.9,

F93861-01

Contents

Preface

This document contains information about setting up and using the KubeVirt module in Oracle Cloud Native Environment. It describes how to prepare a virtual machine image and use it to create a virtual machine instance using KubeVirt and deploy it into a Kubernetes cluster.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

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

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Introduction to the KubeVirt Module

KubeVirt is a virtualization technology that can create and manage virtual machines in a Kubernetes cluster. The virtual machines are created using Kubernetes custom resource definitions (CRDs) and can be managed using the `kubectl` command. A VirtualMachine CRD can be used to create VirtualMachineInstances, and each instance runs as both a virtual machine and a pod. You can create CRDs for VirtualMachineInstanceReplicaSets (which are similar to a ReplicaSet) which can be used to create and manage many VirtualMachineInstances with the same configuration, defined in a template.

For upstream KubeVirt documentation, see https://kubevirt.io/user-guide/.

For more information about KubeVirt, see https://kubevirt.io/.

The KubeVirt module is used to install KubeVirt. The default namespace for KubeVirt is `kubevirt`.

We recommend that you provide persistent storage to KubeVirt virtual machines. Every meaningful workload in the computing industry requires some sort of data storage. Persistent storage is essential when working with stateful applications as it's important that you can retain data beyond the lifecycle of the container or virtual machine. As container images are read-only, any writes to the file system by the virtual machine aren't persisted between boots or live migration. For virtual machines to be restarted, live migrated, and to maintain state, persistent storage is required so that the state can be written somewhere. We recommend that you use a CephFilesystem to provide this storage. You can install and configure a CephFilesystem using the Rook module. For information on setting up CephFilesystem with Rook, see Rook Module.

2
Installing the KubeVirt Module

This chapter discusses how to install the KubeVirt module on Oracle Cloud Native Environment.

Prerequisites

This section contains the prerequisites for installing the KubeVirt module.

Setting up Persistent Storage

We recommend that you set up storage for KubeVirt virtual machines to maintain state over reboots and live migration. You might want to use a CephFilesystem that's made available using a StorageClass. CephFilesystem is a ReadWriteMany file system, which is required to maintain state. The Rook module can be used to set up CephFilesystem storage. For information on setting up a CephFilesystem and StorageClass using the Rook module, see Rook Module.

Setting up a Container Registry

KubeVirt pulls containerized virtual machine images from a container registry. The Kubernetes nodes need to have access to a container registry with the appropriately configured container images to create virtual machines with KubeVirt.

You can set up a container registry using Podman, Oracle Container Runtime for Docker, or any other container registry software. For information creating a Podman local container registry, see Oracle Linux: Podman User's Guide. For information on creating an Oracle Container Runtime for Docker container registry, see Oracle Linux: Oracle Container Runtime for Docker User's Guide.

Creating a KubeVirt Configuration File

To perform any advanced configuration for KubeVirt, you can optionally provide a KubeVirt configuration file. This lets you set up or override any KubeVirt settings such as debug logging or emulation options, when you deploy the KubeVirt module. The KubeVirt objects that you can override are listed in the upstream KubeVirt documentation.

Provide a KubeVirt configuration file on the operator node in YAML format. The high level structure for this file would look similar to:

```
apiVersion: kubevirt.io/v1
kind: KubeVirt
metadata:
  annotations: {}
  labels: {}
  name: {}
  namespace: {}
spec: {}
```

The Platform API Server uses the information contained in the configuration file when creating the KubeVirt module. KubeVirt performs all the set up and configuration for KubeVirt, using the information you provide in this file.

Deploying the KubeVirt Module

You can deploy all the modules required to set up KubeVirt using a single `olcnectl module create` command. This method might be useful if you want to deploy the KubeVirt module at the same time as deploying a Kubernetes cluster.

If you have an existing deployment of the Kubernetes module, you can specify that instance when deploying the KubeVirt module.

This section guides you through installing each component required to deploy the KubeVirt module.

For the full list of the Platform CLI command options available when creating modules, see the `olcnectl module create` command in Platform Command-Line Interface.

To deploy the KubeVirt module:

1. If you don't already have an environment set up, create one into which the modules can be deployed. For information on setting up an environment, see Installation. The name of the environment in this example is `myenvironment`.
2. If you don't already have a Kubernetes module set up or deployed, set one up. For information on adding a Kubernetes module to an environment, see Kubernetes Module. The name of the Kubernetes module in this example is `mycluster`.
3. Create a KubeVirt module and associate it with the Kubernetes module named `mycluster` using the `--kubevirt-kubernetes-module` option. In this example, the KubeVirt module is named `mykubevirt`.

   ```
   olcnectl module create \
   --environment-name myenvironment \
   --module kubevirt \
   --name mykubevirt \
   --kubevirt-kubernetes-module mycluster
   ```

   The `--module` option sets the module type to create, which is `kubevirt`. You define the name of the KubeVirt module using the `--name` option, which in this case is `mykubevirt`.

   The `--kubevirt-kubernetes-module` option sets the name of the Kubernetes module.

   If you don't include all the required options when adding the module, you're prompted to provide them.
4. Use the `olcnectl module install` command to install the KubeVirt module. For example:

   ```
   olcnectl module install \
   --environment-name myenvironment \
   --name mykubevirt
   ```

You can optionally use the `--log-level` option to set the level of logging displayed in the command output. By default, error messages are displayed. For example, you can set the logging level to show all messages when you include:

```
--log-level debug
```

The log messages are also saved as an operation log. You can view operation logs as commands are running, or when they've completed. For more information using operation logs, see Platform Command-Line Interface.

The KubeVirt module is deployed into the Kubernetes cluster.

5. You might want to optionally install the KubeVirt CLI tools `virtctl` and `virt-viewer`. These tools make managing many operations involving KubeVirt virtual machines easier. These tools are useful when accessing the virtual machine console and remote desktop. On a control plane node, install these tools:

```
sudo dnf install virtctl virt-viewer
```

Verifying the KubeVirt Module Deployment

You can verify the KubeVirt module is deployed using the `olcnectl module instances` command on the operator node. For example:

```
olcnectl module instances \
--environment-name myenvironment
```

The output looks similar to:

```
INSTANCE                       MODULE          STATE
mycluster                      kubernetes      installed
mykubevirt                     kubevirt        installed
myrook                         rook            installed
...
```

Note the entry for `kubevirt` in the `MODULE` column is in the `installed` state.

In addition, use the `olcnectl module report` command to review information about the module. For example, use the following command to review the KubeVirt module named `mykubevirt` in `myenvironment`:

```
olcnectl module report \
--environment-name myenvironment \
--name mykubevirt \
--children
```

For more information on the syntax for the `olcnectl module report` command, see Platform Command-Line Interface.

On a control plane node, verify the `virt-*` deployments are running in the `kubevirt` namespace:

```
kubectl get deployments --namespace kubevirt
```

The output looks similar to:

```
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
virt-api          2/2     2            2           117m
virt-controller   2/2     2            2           116m
virt-operator     2/2     2            2           117m
```

3
Using KubeVirt

This chapter discusses how to use the KubeVirt module to create virtual machines in Oracle Cloud Native Environment.

Creating a Virtual Machine Image

KubeVirt pulls a containerized image from a container registry to create virtual machine instances. You need to prepare these images before you can use them with KubeVirt. This section contains an example of how to create a container image of Oracle Linux that can be used to create a KubeVirt virtual machine.

This example uses an Oracle Linux cloud image in QCOW format from:

https://yum.oracle.com/oracle-linux-templates.html

For example, on any host that can connect to your local container registry, download the Oracle Linux 9 QCOW image:

```
wget https://yum.oracle.com/templates/OracleLinux/OL9/u2/x86_64/OL9U2_x86_64-kvm-b197.qcow
```

Create a Dockerfile so you can load the image into your local container registry. For example, in the same directory as the QCOW image is located, create a file named `Dockerfile` that contains:

```
FROM scratch
ADD --chown=107:107 OL9U2_x86_64-kvm-b197.qcow  /disk/
```

Build a containerized image from the QCOW image. For example:

```
podman build . -t myregistry.example.com/kubevirt/oraclelinux:ol9.2
```

You should see output similar to:

```
STEP 1: FROM scratch
STEP 2: ADD --chown=107:107 OL9U2_x86_64-kvm-b197.qcow  /disk/
STEP 3: COMMIT myregistry.example.com/kubevirt/oraclelinux:ol9.2
--> 09b0b23bb66
Successfully tagged myregistry.example.com/kubevirt/oraclelinux:ol9.2
09b0b23bb6673bdfd1481d81dace0b6008c6ab25e1d156c525b9abaaf8d9f30e
```

The containerized image is stored locally. You must now upload it to your container registry. If required, log into your container registry:

```
podman login myregistry.example.com
```

Push the containerized image to your local registry. For example:

```
podman push myregistry.example.com/kubevirt/oraclelinux:ol9.2
```

You should see output similar to:

```
Getting image source signatures
Copying blob 8a8b1918f588 done
Copying config 09b0b23bb6 done
Writing manifest to image destination
Storing signatures
```

You now have a containerized image for Oracle Linux that you can use to create a KubeVirt virtual machine instance. The container image location in this example is:

```
myregistry.example.com/kubevirt/oraclelinux:ol9.2
```

Creating a KubeVirt Instance

This section contains a basic test to verify you can use KubeVirt to create a virtual machine.

This example uses the Oracle Linux 9 container image created in Creating a Virtual Machine Image.

To create a virtual machine with KubeVirt:

1. Create a VirtualMachine file for your virtual machine. The containerDisk option is where you set the location of the container image to create the virtual machine. This should be the location of a container image in either your local container registry, or a public container registry such as DockerHub. Save the file as `vm.yaml`.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: ol9-no-pvc
spec:
  running: true
  template:
    spec:
      networks:
      - name: foo
        pod: {}
      domain:
        resources:
          requests:
            memory: 1024M
        firmware:
          bootloader:
            efi:
              secureBoot: false
        features:
          smm:
```

```
        enabled: true
    devices:
      interfaces:
      - name: foo
        masquerade: {}
        ports:
        - port: 80
      disks:
      - name: containerdisk
        disk:
          bus: virtio
  volumes:
  - name: containerdisk
    containerDisk:
      image: myregistry.example.com/kubevirt/oraclelinux:ol9.2
```

2. Create the VirtualMachine:

```
kubectl apply -f vm.yaml
virtualmachine.kubevirt.io/ol9-no-pvc created
```

3. You can see the VirtualMachine is created using the `kubectl get vm` command:

```
kubectl get vm
NAME                              AGE     STATUS    READY
ol9-no-pvc                        28s     Running   True
```

You can see information on the VirtualMachineInstance using the `kubectl get vmi` command:

```
kubectl get vmi
NAME                    AGE   PHASE     IP
NODENAME              READY
ol9-no-pvc              48s   Running   10.244.3.19
worker1.example.com   True
```

You can get detailed information about the VirtualMachineInstance using the `kubectl describe vmi` command:

```
kubectl describe vmi ol9-no-pvc
Name:         ol9-no-pvc
Namespace:    default
Labels:       kubevirt.io/nodeName=worker1.example.com
Annotations:  kubevirt.io/latest-observed-api-version: v1
              kubevirt.io/storage-observed-api-version: v1alpha3
API Version:  kubevirt.io/v1
Kind:         VirtualMachineInstance
...
  Volumes:
    Container Disk:
      Image:              myregistry.example.com/kubevirt/
oraclelinux:ol9.2
      Image Pull Policy:  IfNotPresent
```

```
    Name:                 containerdisk
...
  Virtual Machine Revision Name:  revision-start-
vm-032efda3-040f-4dc0-941d-382daa5f926b-1
  Volume Status:
    Name:    containerdisk
    Target:  sda
Events:
  Type    Reason            Age   From                       Message
  ----    ------            ----  ----                       -------
  Normal  SuccessfulCreate  94s   virtualmachine-controller
Created virtual machine pod virt-launcher-ol9-no-pvc-sz6hb
  Normal  Created           89s   virt-handler
VirtualMachineInstance defined.
  Normal  Started           89s   virt-handler
VirtualMachineInstance started.
```

You can see the pod that is running that maps to the VirtualMachineInstance using:

```
kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
virt-launcher-ol9-no-pvc-....   2/2     Running   0          1h
```

4. You can delete the VirtualMachine using:

```
kubectl delete vm ol9-no-pvc
virtualmachine.kubevirt.io "ol9-no-pvc" deleted
```

Creating a KubeVirt Instance with Persistent Storage

This section contains a basic test to verify you can use KubeVirt to create a virtual machine using persistent storage. This type of virtual machine can be live migrated and the state is maintained.

This example uses the Oracle Linux 9 container image created in Creating a Virtual Machine Image.

This example also uses a CephFilesystem that is available via a StorageClass named `rook-cephfs`. CephFilesystem is a ReadWriteMany filesystem and is useful to allow writing to the virtual machine filesystem to enable state to be maintained and persist over reboots and live migration. Information on setting up a CephFilesystem using the Rook module is available in Rook Module.

To create a virtual machine that uses persistent storage with KubeVirt:

1. Create a Kubernetes PersistentVolumeClaim file for your StorageClass. In this example, this is a CephFilesystem StorageClass named `rook-cephfs`. On a control plane node, create a file named `pvc-vm.yaml`. Copy the following into the file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
  name: ol9-migratable
spec:
  storageClassName: rook-cephfs
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 30Gi
```

2. Create the Kubernetes PersistentVolumeClaim.

```
kubectl apply -f pvc-vm.yaml
persistentvolumeclaim/ol9-migratable created
```

3. Create a VirtualMachine file for your virtual machine. This example uses the PersistentVolumeClaim to enable writing to the virtual machine to maintain state. The containerDisk option is where you set the location of the container image to create the virtual machine. This should be the location of a container image in either your local container registry, or a public container registry such as DockerHub. Save the file as `vm.yaml`.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: ol9
spec:
  running: true
  template:
    spec:
      evictionStrategy: LiveMigrate
      terminationGracePeriodSeconds: 30
      networks:
      - name: foo
        pod: {}
      domain:
        resources:
          requests:
            memory: 1024M
        firmware:
          bootloader:
            efi:
              secureBoot: false
        features:
          smm:
            enabled: true
        devices:
          interfaces:
          - name: foo
            masquerade: {}
            ports:
            - port: 80
          disks:
          - name: containerdisk
            disk:
              bus: virtio
```

```
        - name: installdisk
          disk:
            bus: virtio
      volumes:
      - name: containerdisk
        containerDisk:
          image: myregistry.example.com/kubevirt/oraclelinux:ol9.2
      - name: installdisk
        persistentVolumeClaim:
          claimName: ol9-migratable
```

4. Create the VirtualMachine:

```
kubectl apply -f vm.yaml
virtualmachine.kubevirt.io/ol9 created
```

5. You can see the VirtualMachine is created using the `kubectl get vm` command:

```
kubectl get vm
NAME                    AGE     STATUS    READY
ol9                     41m     Running   True
```

You can see information on the VirtualMachineInstance using the `kubectl get vmi` command:

```
kubectl get vmi
NAME                    AGE     PHASE     IP
NODENAME              READY
ol9                     42m     Running   10.244.3.29
worker1.example.com   True
```

You can get detailed information about the VirtualMachineInstance using the `kubectl describe vmi` command:

```
kubectl describe vmi ol9
Name:         ol9
Namespace:    default
Labels:       kubevirt.io/nodeName=worker1.example.com
Annotations:  kubevirt.io/latest-observed-api-version: v1
              kubevirt.io/storage-observed-api-version: v1alpha3
API Version:  kubevirt.io/v1
Kind:         VirtualMachineInstance
...
  Volumes:
    Container Disk:
      Image:              myregistry.example.com/kubevirt/
oraclelinux:ol9.2
      Image Pull Policy:  IfNotPresent
    Name:                 containerdisk
    Name:                 installdisk
    Persistent Volume Claim:
      Claim Name:  ol9-migratable
...
```

```
  Virtual Machine Revision Name:  revision-start-vm-05252d81-
e403-4e7b-9834-b9e03850ac7c-1
  Volume Status:
    Name:    containerdisk
    Target:  sda
    Name:    installdisk
    Persistent Volume Claim Info:
      Access Modes:
        ReadWriteMany
      Capacity:
        Storage:            30Gi
      Filesystem Overhead:  0.055
      Requests:
        Storage:    30Gi
      Volume Mode:  Filesystem
    Target:         vda
Events:
  Type    Reason            Age   From                          Message
  ----    ------            ----  ----                          -------
  Normal  SuccessfulCreate  42m   disruptionbudget-controller  Created
PodDisruptionBudget kubevirt-disruption-budget-rgh5f
  Normal  SuccessfulCreate  42m   virtualmachine-controller    Created
virtual machine pod virt-launcher-ol9-6qvsh
  Normal  Created           42m   virt-handler
VirtualMachineInstance defined.
  Normal  Started           42m   virt-handler
VirtualMachineInstance started.
```

You can see the pod that is running that maps to the VirtualMachineInstance using:

```
kubectl get pod
NAME                       READY   STATUS    RESTARTS   AGE
virt-launcher-ol9-....     2/2     Running   0          1h
```

6. You can delete the VirtualMachine using:

```
kubectl delete vm ol9
virtualmachine.kubevirt.io "ol9" deleted
```

4
Removing the KubeVirt Module

You can remove a deployment of the KubeVirt module and leave the Kubernetes cluster in place. To do this, you remove the KubeVirt module from the environment.

Use the `olcnectl module uninstall` command to remove the KubeVirt module. For example, to uninstall the KubeVirt module named `mykubevirt` in the environment named `myenvironment`:

```
olcnectl module uninstall \
--environment-name myenvironment \
--name mykubevirt
```

The KubeVirt module is removed from the environment.