







## Join a community dedicated to learning open source

The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.



**Network** with tens of thousands of community members



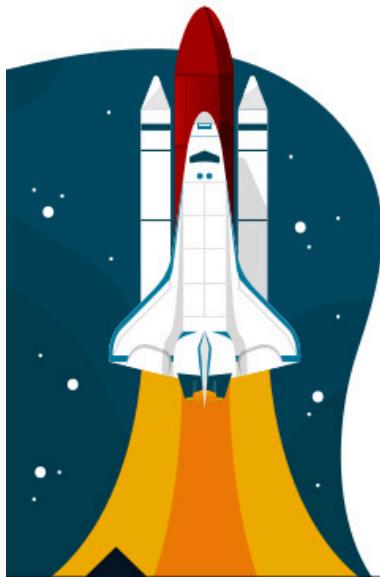
**Engage** in thousands of active conversations and posts



**Join and interact** with hundreds of certified training instructors



**Unlock** badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

**Access** free Red Hat training videos

**Discover** the latest Red Hat Training and Certification news

**Connect** with your instructor - and your classmates - before, after, and during your training course.

**Join** peers as you explore Red Hat products

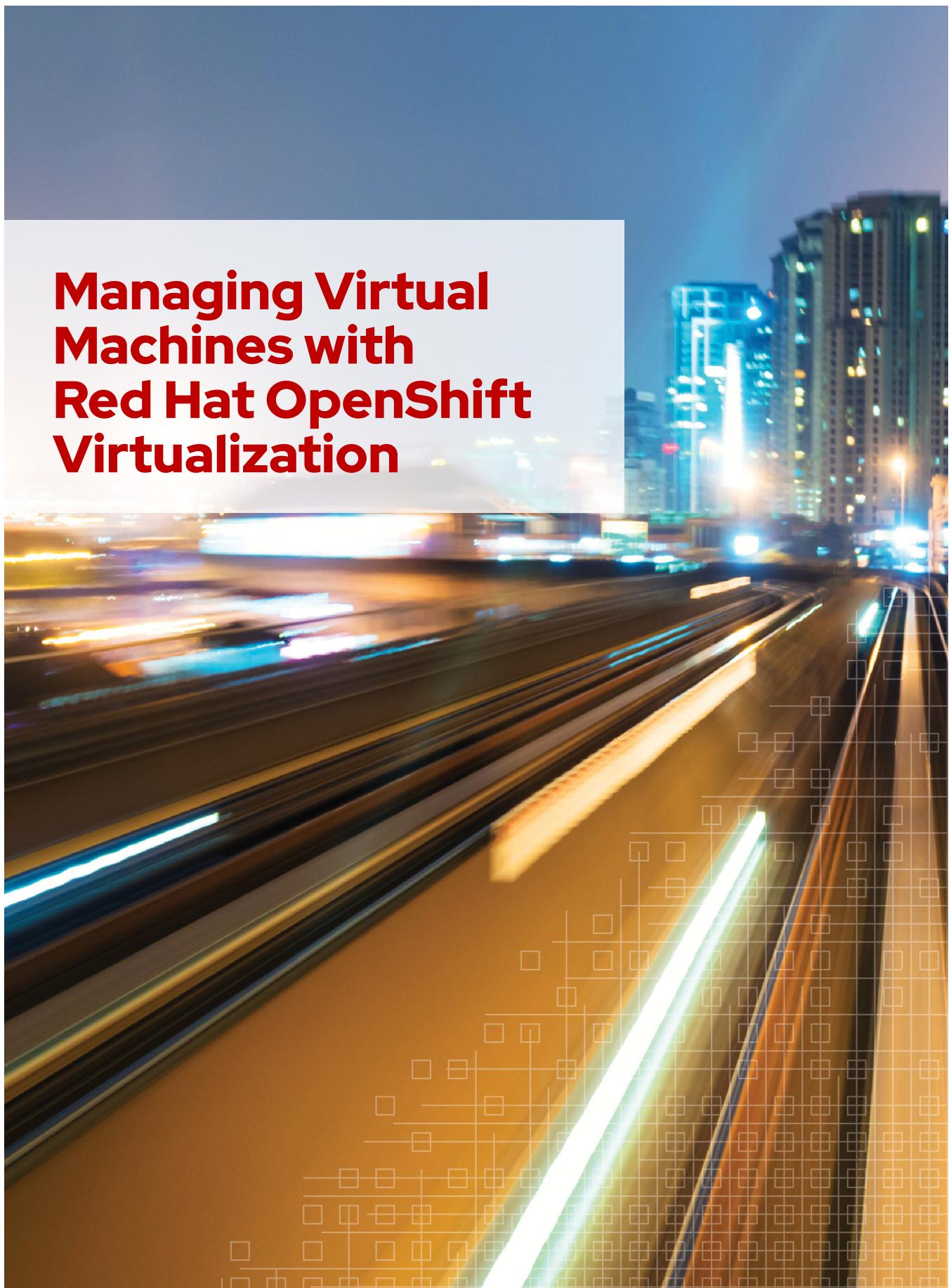
Join the conversation [learn.redhat.com](https://learn.redhat.com)



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.



# Managing Virtual Machines with Red Hat OpenShift Virtualization



# OCP 4.16 DO316

## Managing Virtual Machines with Red Hat OpenShift

### Virtualization

### Edition 3 20250110

### Publication date 20250110

Authors:

Andrés Hernández, Bernardo Gargallo Jaquotot,  
Christopher Caillouet, François Andrieu, Harpal Singh,  
Maria Ordoñez, Michael Jarrett, Natalie Watkins, Randy Thomas,  
Susan Lauber

Course Architects:

Susan Lauber, Natalie Watkins

DevOps Engineer:

Benjamin Chardi Marco

Editors:

Julian Cable, Rachel Lee

© 2025 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2025 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to [training@redhat.com](mailto:training@redhat.com) [mailto:[training@redhat.com](mailto:training@redhat.com)] or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle American, Inc. and/or its affiliates.

XFS® is a registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is a trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors:David Sacco, Steven Bonneville

<b>Document Conventions</b>	<b>xi</b>
Admonitions .....	xi
Inclusive Language .....	xii
<b>Introduction</b>	<b>xiii</b>
Managing Virtual Machines with Red Hat OpenShift Virtualization .....	xiii
Orientation to the Classroom Environment .....	xiv
Performing Lab Exercises .....	xxii
<b>1. Red Hat OpenShift Virtualization</b>	<b>1</b>
OpenShift Virtualization .....	2
Quiz: OpenShift Virtualization .....	8
Underlying Kubernetes Architecture .....	12
Guided Exercise: Access and Identify Key OpenShift Cluster Services .....	23
Deploying the OpenShift Virtualization Operator .....	31
Guided Exercise: Deploy the OpenShift Virtualization Operator .....	42
Quiz: Red Hat OpenShift Virtualization .....	53
Summary .....	55
<b>2. Running and Accessing Virtual Machines</b>	<b>57</b>
Virtual Machine Resources .....	58
Quiz: Virtual Machine Resources .....	66
Creating and Accessing Virtual Machines by Using the Web Console .....	70
Guided Exercise: Create and Access a Virtual Machine by Using the Web Console .....	77
Creating and Accessing Virtual Machines from the Command Line .....	85
Guided Exercise: Create and Access a Virtual Machine from the Command Line .....	92
Monitoring and Inspecting Virtual Machines .....	98
Guided Exercise: Monitor and Inspect Virtual Machines .....	106
Lab: Create, Configure, and Monitor Virtual Machines .....	115
Summary .....	130
<b>3. Configuring Kubernetes Networking for Virtual Machines</b>	<b>131</b>
Configuring Network Communication Between Virtual Machines Within the Cluster .....	132
Guided Exercise: Configure Networking Communication Between Virtual Machines .....	142
Restricting Access to Virtual Machines by Using a Network Policy .....	150
Guided Exercise: Restrict Access to a Virtual Machine by Using a Network Policy .....	155
Configuring External Access to Virtual Machines .....	162
Guided Exercise: Configure External Access to a Virtual Machine .....	171
Configure External Access with Load Balancer Services .....	178
Guided Exercise: Configure SSH Access for a Virtual Machine with Load Balancer Services .....	189
Lab: Configure Kubernetes Networking for Virtual Machines .....	200
Summary .....	211
<b>4. Connecting Virtual Machines to External Networks</b>	<b>213</b>
Advanced Networking with Multus CNI .....	214
Quiz: Advanced Networking with Multus CNI .....	219
Configuring Multihomed Nodes and Virtual Machines .....	223
Guided Exercise: Configure Multihomed Nodes and Virtual Machines .....	241
Lab: Connect Virtual Machines to External Networks .....	254
Summary .....	262
<b>5. Configuring Storage for Virtual Machines</b>	<b>263</b>
Attaching Persistent Storage to Virtual Machines .....	264
Guided Exercise: Attach Persistent Storage to Virtual Machines .....	277
Managing Storage Devices on Virtual Machines .....	281
Guided Exercise: Attach and Access Storage Devices on a Virtual Machine .....	288
Connecting a Virtual Machine to External Storage .....	294

Guided Exercise: Connect a Virtual Machine to External Storage .....	303
Lab: Configure Storage for Virtual Machines .....	308
Summary .....	317
<b>6. Create and Restore Backups of Virtual Machines</b>	<b>319</b>
Exporting and Importing Virtual Machine Data and Settings .....	320
Guided Exercise: Export and Import Virtual Machine Data and Settings .....	336
Managing Virtual Machine Snapshots .....	342
Guided Exercise: Manage Virtual Machine Snapshots .....	348
OADP Operator Deployment and Features .....	354
Guided Exercise: Verify the OADP Operator Installation .....	365
Back up and Restore with OADP .....	376
Guided Exercise: Back up and Restore an OpenShift Project with OADP .....	393
Lab: Back up and Restore a Virtual Machine with OADP .....	407
Summary .....	417
<b>7. Replicating Virtual Machines by Using Instance Types, Templates, and Clones</b>	<b>419</b>
Cloning Virtual Machines .....	420
Guided Exercise: Clone a Virtual Machine .....	427
Create Virtual Machines by Using Predefined Virtual Machine Templates and Instance Types .....	436
Guided Exercise: Creating Virtual Machines with Predefined Virtual Machine Templates and Instance Types .....	453
Creating Custom Templates and Instance Types .....	463
Guided Exercise: Create a Custom Template and an Instance Type .....	471
Lab: Replicating Virtual Machines by Using Instance Types, Templates, and Clones .....	484
Summary .....	495
<b>8. Advanced Virtual Machine Management</b>	<b>497</b>
Migrating Virtual Machines from Foreign Hypervisors .....	498
Guided Exercise: Migrate Virtual Machines from Foreign Hypervisors .....	517
Controlling the Scheduling of Virtual Machines .....	530
Guided Exercise: Control the Scheduling of a Virtual Machine .....	540
Configuring Health Probes for Virtual Machines .....	550
Guided Exercise: Configure Health Probes for Virtual Machines .....	555
Live Migrating a Virtual Machine .....	564
Guided Exercise: Live Migrate a Virtual Machine .....	572
Lab: Advanced Virtual Machine Management .....	579
Summary .....	588
<b>9. Configuring Kubernetes High Availability for Virtual Machines</b>	<b>589</b>
Virtual Machine Load Balancing with Kubernetes Networking Resources .....	590
Guided Exercise: Load Balance a Virtual Machine with Kubernetes Networking Resources .....	593
Preparing for Node Maintenance .....	601
Guided Exercise: Perform Node Maintenance .....	609
Configuring Virtual Machines to Survive Node Failure .....	616
Guided Exercise: Configure Virtual Machines to Survive a Node Failure .....	623
OpenShift and OpenShift Virtualization Updates .....	630
Quiz: OpenShift and OpenShift Virtualization Updates .....	635
Lab: Configure Kubernetes High Availability for Virtual Machines .....	637
Summary .....	646
<b>10. Comprehensive Review</b>	<b>647</b>
Comprehensive Review .....	648
Lab: Deploy OpenShift Virtualization and Configure External Network Access for a Virtual Machine .....	650

Lab: Use a Template to Deploy a Virtual Machine and Prepare a Node for Maintenance ...	657
Lab: Clone a Virtual Machine and Configure Load Balancing .....	668



# Document Conventions

---

This section describes various conventions and practices that are used throughout all Red Hat Training courses.

## Admonitions

Red Hat Training courses use the following admonitions:



### References

These describe where to find external documentation that is relevant to a subject.



### Note

Notes are tips, shortcuts, or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on something that makes your life easier.



### Important

Important sections provide details of information that is easily missed: configuration changes that apply only to the current session, or services that need restarting before an update applies. Ignoring these admonitions will not cause data loss, but might cause irritation and frustration.



### Warning

Do not ignore warnings. Ignoring these admonitions will most likely cause data loss.

# Inclusive Language

---

Red Hat Training is currently reviewing its use of language in various areas to help remove any potentially offensive terms. This is an ongoing process and requires alignment with the products and services that are covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

# Introduction

## Managing Virtual Machines with Red Hat OpenShift Virtualization

*Managing Virtual Machines with Red Hat OpenShift Virtualization (DO316)* is a hands-on course that teaches the essential skills to create and manage virtual machines (VM) on OpenShift by using the Red Hat OpenShift Virtualization operator. This course does not require previous knowledge of containers and Kubernetes.

### Course Objectives

- Create, access, and manage VMs on OpenShift clusters.
- Control usage and access of CPU, memory, storage, and networking resources from VMs by using the same Kubernetes features that would also control usage and access to these resources for containers.
- Manage high availability (HA) of VMs by using standard Kubernetes features and extensions from OpenShift Virtualization.
- Connect VMs on OpenShift to data center services outside their OpenShift cluster, such as storage and databases.

### Audience

- Virtual Machine Administrators who want to virtualize workloads from traditional Hypervisors to OpenShift Virtualization.
- Platform Engineers, Cloud Administrators, and System Administrators who want to support virtualized workloads, either independently from or in the same OpenShift cluster as containerized workloads.

### Prerequisites

- This course requires no previous experience with containers, Kubernetes, and OpenShift, however, learners are encouraged to attend, before taking DO316, Containers, Kubernetes and Red Hat OpenShift Technical Overview (DO080) [<https://www.redhat.com/en/services/training/do080-deploying-containerized-applications-technical-overview>].
- Although Linux skills are not required for managing OpenShift clusters and OpenShift Virtualization, operating individual Linux VMs requires Linux sysadmin skills that the Red Hat System Administration I (RH124) [<https://www.redhat.com/en/services/training/rh124-red-hat-system-administration-i>] and Red Hat System Administration II (RH134) [<https://www.redhat.com/en/services/training/rh134-red-hat-system-administration-ii>] courses provide.

# Orientation to the Classroom Environment

In this course, the main computer system that is used for hands-on learning activities (exercises) is **workstation**.

The **workstation** machine has a **student** standard user account, with **student** as the password. No exercise in this course requires that you log in as **root**, but if you must, the **root** password on the **workstation** machine is **redhat**.

From the **workstation** machine, you type the **oc** commands to manage the OpenShift cluster, which comes preinstalled as part of your classroom environment.

Also from the **workstation** machine, you run the required shell scripts and Ansible Playbooks to complete the exercises for this course.

If exercises require you to open a web browser to access any application or website, then you must use the graphical console of the **workstation** machine and use the Firefox web browser from there.



## Note

The first time that you start your classroom environment, the OpenShift cluster takes a little longer to become fully available. The **lab** command at the beginning of each exercise checks and waits as required.

If you try to access your cluster by using either the **oc** command or the web console without first running a **lab** command, then your cluster might not yet be available. In that case, wait a few minutes and try again.

## Log in to OpenShift from the Shell

To access your OpenShift cluster from the **workstation** machine, use `https://api.ocp4.example.com:6443` as the API URL, for example:

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
```

Besides the **admin** user, which has cluster administrator privileges, your OpenShift cluster also provides a **developer** user, with **developer** as the password, with no special privileges.

## Accessing the OpenShift Web Console

If you prefer to use the OpenShift web console, then open a Firefox web browser on your **workstation** machine and access the following URL:

- `https://console-openshift-console.apps.ocp4.example.com`

Click **htpasswd\_provider** and provide the login credentials for either the **admin** or the **developer** user:

## User credentials

Username	Password
admin	redhatocp
developer	developer

## The Classroom Environment

Every student gets a complete remote classroom environment. As part of that environment, every student gets a dedicated OpenShift cluster for administration tasks.

The classroom environment runs entirely as virtual machines in a large Red Hat OpenStack Platform cluster, which is shared among many students. Red Hat Training maintains many OpenStack clusters, in different data centers across the globe, to provide lower latency to students from many countries.

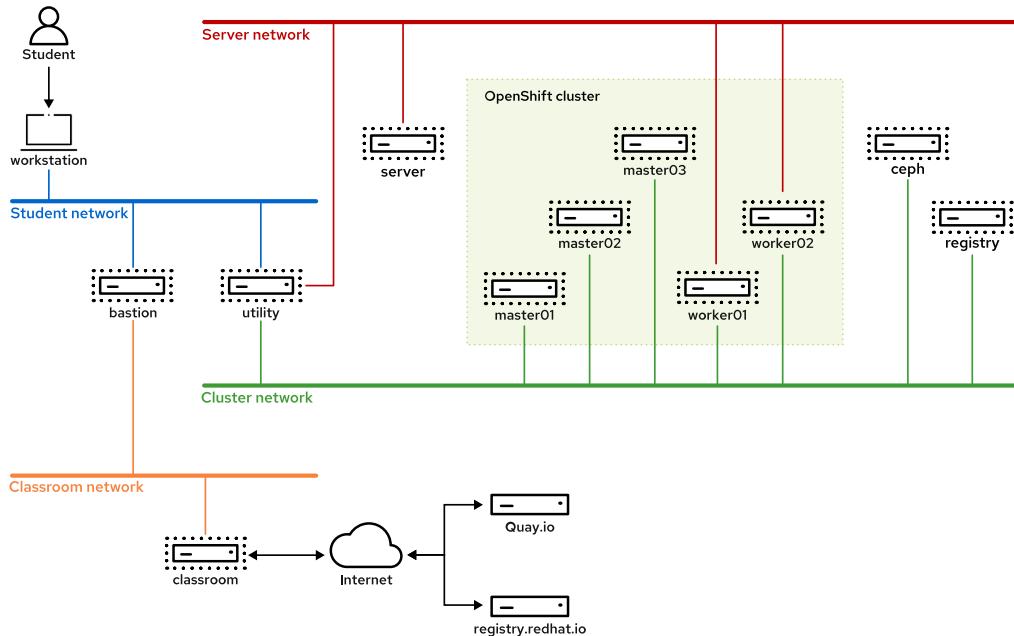


Figure 0.1: DO316 classroom architecture

All machines in the Student, Classroom, Server, and Cluster networks run Red Hat Enterprise Linux 9 (RHEL 9), except those machines that are nodes of the OpenShift cluster, which run RHEL CoreOS.

To improve consistency in SCSI device disk mapping, the classroom lab environment uses the `sd_mod.probe=sync` kernel option for synchronous device probing of SCSI devices. Additionally, disks are provided through a Ceph cluster as SCSI devices and receive drive mappings of `/dev/sd*` on classroom machines.

The `bastion`, `utility`, `ceph`, `server`, and `classroom` systems must always be running. These systems provide required infrastructure services for the classroom environment and its OpenShift cluster. For most exercises, you are not expected to interact with any of these services directly.

## Introduction

Usually, the `lab` commands from the exercises access these machines when it is required to set up your environment for the exercise, and require no further action from you.

For the few exercises that require you to access a system other than `workstation`, primarily the `utility` system, you receive explicit instructions and the necessary connection information as part of the exercise.

All systems in the Student network are in the `lab.example.com` DNS domain, and all systems in the Classroom network are in the `example.com` DNS domain.

The `masterXX` and `workerXX` systems are nodes of the OpenShift 4 cluster that is part of your classroom environment.

All systems in the Cluster network are in the `ocp4.example.com` DNS domain.

All systems in the Server network are in the `srv.example.com` DNS domain.

### Classroom Machines

Machine name	IP addresses	ROLE
<code>workstation.lab.example.com</code>	172.25.250.9	Graphical workstation for system administration
<code>classroom.example.com</code>	172.25.254.254	Router to link the Classroom network to the internet
<code>bastion.lab.example.com</code>	172.25.250.254	Router to link the Student network to the Classroom network
<code>utility.lab.example.com</code>	172.25.250.253	Router to link the Student, Classroom, and Server networks
<code>server.srv.example.com</code>	192.168.51.40	Server and router to link the Cluster network to a private DHCP network
<code>ceph.ocp4.example.com</code>	192.168.50.30	Server with a Red Hat Storage Ceph preinstalled cluster
<code>master01.ocp4.example.com</code>	192.168.50.10	Control plane node
<code>master02.ocp4.example.com</code>	192.168.50.11	Control plane node
<code>master03.ocp4.example.com</code>	192.168.50.12	Control plane node
<code>worker01.ocp4.example.com</code>	172.25.250.13	Compute node
<code>worker02.ocp4.example.com</code>	172.25.250.14	Compute node

## The Dedicated OpenShift Cluster

The Red Hat OpenShift Container Platform 4 (RHOCP) cluster inside the classroom environment is preinstalled by using the Pre-existing Infrastructure installation method. All nodes are treated as bare metal servers, despite being virtual machines in an OpenStack cluster.

OpenShift cloud-provider integration capabilities are not enabled, and some features that depend on that integration, such as machine sets and autoscaling of cluster nodes, are not available.

## Introduction

Your OpenShift cluster is in the state from running the OpenShift installer with default configurations, except for some day-2 customizations:

- The cluster provides a default storage class that is backed by a Network File System (NFS) storage provider. Thus, applications that require persistent storage volumes perform comparably to running on a cluster that is installed by using the Full-stack Automation installation method.
- An HTPasswd Identity Provider (IdP) is preconfigured with the `admin` and `developer` users.

The *Troubleshooting Access to your OpenShift Cluster* section describes how to access the **utility** machine.

## Restoring Access to Your OpenShift Cluster

If you suspect that you cannot log in to your OpenShift cluster as the `admin` user any more because you incorrectly changed your cluster authentication settings, then run the `lab finish` command from your current exercise and restart the exercise by running its `lab start` command.

If running a `lab` command is not sufficient, then you can follow instructions in the next section to use the **utility** machine to access your OpenShift cluster.

## Troubleshooting Access to Your OpenShift Cluster

The **utility** machine ran the OpenShift installer inside your classroom environment, and it is a useful resource to troubleshoot cluster issues. You can view the installer manifests and logs in the `/home/lab/ocp4` directory of the **utility** machine.

You rarely need to log in to the **utility** server to perform exercises. If your OpenShift cluster is taking too long to start, or is in a degraded state, then you can log in to the **utility** machine as the `lab` user to troubleshoot your classroom environment.

The `student` user on the **workstation** machine is already configured with SSH keys that enable logging in to the **utility** machine without a password.

```
[student@workstation ~]$ ssh lab@utility
```

On the **utility** machine, the `lab` user is preconfigured with a `.kube/config` file that grants access as `system:admin` without first requiring the `oc login` command.

You can then run troubleshooting commands, such as `oc get node`, if they fail from the **workstation** machine.

You should not require SSH access to your OpenShift cluster nodes for regular administration tasks, because OpenShift 4 provides the `oc debug` command. The `lab` user on the **utility** server is preconfigured with SSH keys to access all cluster nodes if necessary. For example:

```
[lab@utility ~]$ ssh -i ~/.ssh/lab_rsa core@master01.ocp4.example.com
```

In the preceding example, replace `master01` with the name of the chosen cluster node.

## Approving Node Certificates on Your OpenShift Cluster

Red Hat OpenShift Container Platform clusters are designed to run continuously, 24x7, until they are decommissioned. Unlike a production cluster, the classroom environment contains a cluster

## Introduction

that was stopped after installation and will be stopped and restarted several times before you finish this course. This scenario requires special handling, unlike a production cluster.

The control plane and compute nodes in an OpenShift cluster often communicate with each other. All communication between cluster nodes is protected by mutual authentication based on per-node TLS certificates.

The OpenShift installer handles creating and approving TLS certificate signing requests (CSRs) for the Full-stack Automation installation method. The system administrator manually approves these CSRs for the Pre-existing Infrastructure installation method.

All per-node TLS certificates have a short expiration life of 24 hours (the first time) and 30 days (after renewal). When the certificates are about to expire, the affected cluster nodes create new CSRs, and the control plane automatically approves them. If the control plane is offline when the TLS certificate of a node expires, then a cluster administrator must approve the pending CSR.

The **utility** machine includes a system service that approves CSRs from the cluster when you start your classroom, to ensure that your cluster is ready when you begin the exercises. If you create or start your classroom and begin an exercise too quickly, then your cluster might not be ready. If so, wait a few minutes while the **utility** machine handles CSRs, and then try again.

Sometimes, the **utility** machine fails to approve all required CSRs, for example because the cluster took too long to generate all the required CSRs requests, and the system service did not wait long enough. Also, the OpenShift cluster nodes might not have waited long enough for their CSRs to be approved, and new CSRs might have been issued that superseded previous ones.

If these issues arise, then your cluster takes too long to come up, and your `oc login` or `lab` commands fail. To resolve the problem, you can log in to the **utility** machine, as explained previously, and run the `sign.sh` script to approve any additional and pending CSRs.

```
[lab@utility ~]$ ./sign.sh
```

The `sign.sh` script loops a few times if your cluster nodes issue new CSRs that supersede the ones that were approved.

After you approve, or after the system service in the **utility** machine approves all CSRs, OpenShift must restart some cluster operators. It takes some moments before your OpenShift cluster is ready to answer requests from clients. To help you handle this scenario, the **utility** machine provides the `wait.sh` script, which waits until your OpenShift cluster is ready to accept authentication and API requests from remote clients.

```
[lab@utility ~]$ ./wait.sh
```

Although unlikely, if neither the service on the **utility** machine nor running the `sign.sh` and `wait.sh` scripts makes your OpenShift cluster available to begin exercises, then open a customer support ticket.

**Note**

You can run troubleshooting commands from the utility machine at any time, even if you have control plane nodes that are not ready. Some useful commands are as follows:

`oc get node`

Verify whether all your cluster nodes are ready.

`oc get csr`

Verify whether your cluster still has any pending, unapproved CSRs.

`oc get co`

Verify whether any of your cluster operators are unavailable, in a degraded state, or are progressing through configuration and rolling out pods.

If these verifications fail, then you can try deleting and re-creating your classroom as a final step before creating a customer support ticket.

## Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning (ROLE) classroom. Self-paced courses are accessed through a web application that is hosted at [rol.redhat.com](http://rol.redhat.com) [<http://rol.redhat.com>]. Log in to this site with your Red Hat Customer Portal user credentials.

### Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through web page interface controls. The state of each classroom virtual machine is displayed on the **Lab Environment** tab.

The screenshot shows a web-based interface for managing a lab environment. At the top, there are tabs for 'Table of Contents', 'Course', and 'Lab Environment'. Below the tabs, there is a section titled '▶ Lab Controls' with instructions: 'Click CREATE to build all of the virtual machines needed for the classroom lab environment. This may take several minutes to complete. Once created the environment can then be stopped and restarted to pause your experience.' It also notes that 'If you DELETE your lab, you will remove all of the virtual machines in your classroom and lose all of your progress.' Below this section is a table listing five virtual machines:

Name	Status	Action	Open Console
bastion	active	ACTION -	OPEN CONSOLE
classroom	active	ACTION -	OPEN CONSOLE
servera	building	ACTION -	OPEN CONSOLE
serverb	building	ACTION -	OPEN CONSOLE
workstation	active	ACTION -	OPEN CONSOLE

Figure 0.2: An example course Lab Environment management page

**Machine States**

<b>Virtual machine state</b>	<b>Description</b>
building	The virtual machine is being created.
active	The virtual machine is running and available. If it just started, it still might be starting services.
stopped	The virtual machine is shut down. On starting, the virtual machine boots into the same state it was in before shutdown. The disk state is preserved.

**Classroom Actions**

<b>Button or action</b>	<b>Description</b>
CREATE	Create the ROLE classroom. Creates and starts all the virtual machines that are needed for this classroom.
CREATING	The ROLE classroom virtual machines are being created. Creation can take several minutes to complete.
DELETE	Delete the ROLE classroom. Destroys all virtual machines in the classroom. <b>All saved work on those systems' disks is lost.</b>
START	Start all virtual machines in the classroom.
STARTING	All virtual machines in the classroom are starting.
STOP	Stop all virtual machines in the classroom.

**Machine Actions**

<b>Button or action</b>	<b>Description</b>
OPEN CONSOLE	Connect to the system console of the virtual machine in a new browser tab. You can log in directly to the virtual machine and run commands, when required. Normally, log in to the <b>workstation</b> virtual machine only, and from there, use <code>ssh</code> to connect to the other virtual machines.
ACTION > Start	Start (power on) the virtual machine.
ACTION > Shutdown	Gracefully shut down the virtual machine, preserving disk contents.
ACTION > Power Off	Forcefully shut down the virtual machine, while still preserving disk contents. This action is equivalent to removing the power from a physical machine.
ACTION > Reset	Forcefully shut down the virtual machine and reset associated storage to its initial state. <b>All saved work on that system's disks is lost.</b>

## Introduction

At the start of an exercise, if instructed to reset a single virtual machine node, click **ACTION > Reset** for only that specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click **ACTION > Reset** on every virtual machine in the list.

If you want to return the classroom environment to its original state at the start of the course, then click **DELETE** to remove the entire classroom environment. After the lab is deleted, then click **CREATE** to provision a new set of classroom systems.



### Warning

The **DELETE** operation cannot be undone. All completed work in the classroom environment is lost.

## The Auto-stop and Auto-destroy Timers

The Red Hat Online Learning enrollment entitles you to a set allotment of computer time. To help to conserve your allotted time, the ROLE classroom uses timers, which shut down or delete the classroom environment when the appropriate timer expires.

To adjust the timers, locate the two + buttons at the bottom of the course management page. Click the auto-stop + button to add another hour to the auto-stop timer. Click the auto-destroy + button to add another day to the auto-destroy timer. Auto-stop has a maximum of 11 hours, and auto-destroy has a maximum of 14 days. Be careful to keep the timers set while you are working, so that your environment is not unexpectedly shut down. Be careful not to set the timers unnecessarily high, which could waste your subscription time allotment.

# Performing Lab Exercises

Run the `lab` command from the `workstation` machine to prepare your environment before each hands-on exercise, and again to clean up after an exercise. Each hands-on exercise has a unique name within a course.

There are two types of exercises.

- The first type, a *guided exercise*, is a practice exercise that follows a course narrative. If a narrative is followed by a quiz, this usually indicates that the topic did not have an achievable practice exercise.
- The second type, an *end-of-chapter lab*, is a gradable exercise to help verify your learning. When a course includes a comprehensive review, the review exercises are structured as gradable labs. The syntax for running an exercise script is:

```
[student@workstation ~]$ lab action exercise
```

The `action` is a choice of `start`, `grade`, or `finish` commands. All exercises support `start` and `finish` commands. Only end-of-chapter labs and comprehensive review labs support the `grade` command.

## start

The start logic for a script verifies the resources required to begin an exercise. This might include configuring settings, creating resources, checking prerequisite services, and verifying necessary outcomes from previous exercises. Exercise start logic allows you to perform any exercise at any time, even if prerequisite exercises have not been performed.

## grade

End-of-chapter labs help verify what you have learned after practicing with earlier guided exercises. The grade action directs the `lab` command to display a list of grading criteria, with a `PASS` or `FAIL` status for each. To achieve a `PASS` status for all criteria, fix the failures, and then run the grade action.

## finish

The `finish` logic for a script deletes exercise resources that are no longer necessary. Clean-up logic allows you to repeatedly perform an exercise, benefiting course performance by ensuring that unneeded objects release their resources.

## Listing Available Lab Scripts

To list the available exercises for a course, use tab completion in the `Lab` command:

```
[student@workstation ~]$ lab start Tab Tab
accessing-clicreate    backup-snapshot      replicating-create
accessing-guicreate    ha-loadbalance       replicating-review
accessing-monitor      ha-maintenance     replicating-using
accessing-review       ha-node             review-cr1
advanced-import        ha-review           review-cr2
advanced-migrate       multihomed-nmstate   review-cr3
advanced-probe         multihomed-review    storage-external
```

advanced-review	network-external	storage-intro
advanced-selector	network-lb	storage-multiple
backup-export	netework-policy	storage-review
backup-oadp	network-review	virtualization-architecture
backup-restore	network-services	virtualization-deploy
backup-review	replicating-clone	

## Troubleshooting Lab Scripts

When you run the `lab` command with a valid exercise and action, it creates one log file in `/tmp/log/labs`, plus the directory if it does not exist. The file, named `exercise`, captures error messages that are normally displayed on your terminal.

```
[student@workstation ~]$ ls -l /tmp/log/labs
-rw-rw-r--. 1 student student 1024 Apr 25 12:00 comprehensive-review
```

## Interpreting the Exercise Log Files

Exercise scripts send output to the log file when the scripts fail. Thus, the exercise log is useful for troubleshooting.

Although exercise scripts are always run from the `workstation` machine, they perform tasks on other systems in the course environment. Many course environments, including OpenStack and OpenShift, use a command-line tool that is invoked from the `workstation` machine to communicate with server systems by using API calls.

Because script actions typically distribute tasks to multiple systems, additional troubleshooting is necessary to determine where a failed task occurred. Log in to those other systems and use Linux diagnostic skills to read local system log files and determine the root cause of the lab script failure.

## Upgrading Lab Package Version

The `workstation` machine has the course lab package installed. If the instructor requests it, you can upgrade the version to fix a bug in the lab package that came preinstalled.

```
[student@workstation ~]$ lab upgrade DO316
```



## Chapter 1

# Red Hat OpenShift Virtualization

### Goal

Distinguish Red Hat OpenShift Virtualization from container technologies and from traditional virtual machine technologies. Describe the features and use cases of OpenShift Virtualization. Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift environment.

### Sections

- OpenShift Virtualization (and Quiz)
- Underlying Kubernetes Architecture (and Guided Exercise)
- Deploying the OpenShift Virtualization Operator (and Guided Exercise)

# OpenShift Virtualization

## Objectives

- Describe the features and use cases of OpenShift Virtualization.

## Virtualization Concepts

*System virtualization* enables a single computer to be partitioned into multiple, concurrent virtual computers, where each computer runs its own operating system. These virtual machines are fully isolated from each other, as if they are running on private hardware. Virtual machines have their own network interfaces, IP addresses, file systems, and other peripherals. Each virtual machine can run any supported operating system or version.

Virtualization enables efficient use of physical computing resources. A large system can be partitioned into many virtual systems, to use to full capacity. Virtualization enables quick provisioning of new virtual servers, which is faster than deploying new physical hardware.

The *hypervisor* component manages and supports the physical server virtualization. This hypervisor runs the virtual machines for each virtualized operating system. It provides access to virtual CPUs, memory, disks, networking, and other peripherals, and restricts virtual machines from having direct access to hardware or to other virtual machines.

The physical machine and operating system that provide the hypervisor is called a *host*. A *guest* is an operating system that runs in a *virtual machine* (VM) on a hypervisor.

A *virtualization management* solution effectively manages virtual machines and hosts that run in your infrastructure. Typically, virtualization management tools create an integrated virtualization environment to create, manage, and monitor virtual machines, hypervisors, storage, and networking resources. A virtualization manager provides centralized control for your virtualization infrastructure, rather than requiring you to individually manage and monitor hundreds of hypervisors and their virtual machines.

## Virtualization versus Cloud Computing

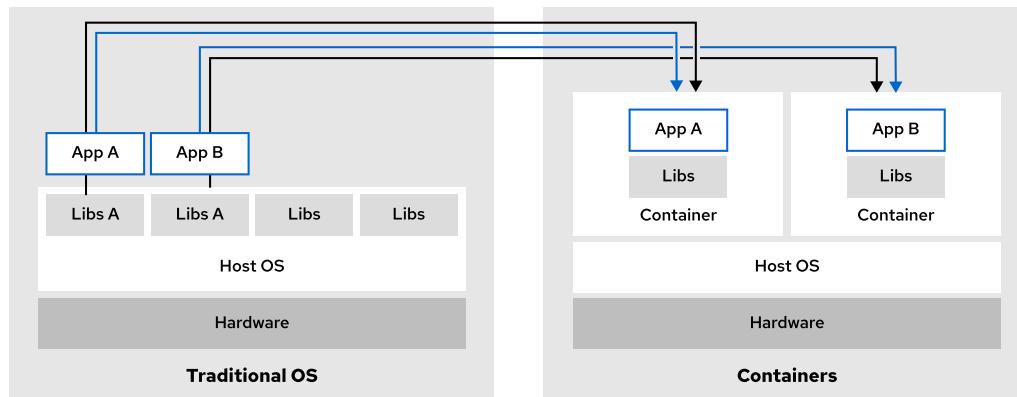
Virtualization provides a foundation for cloud computing. Cloud computing uses virtualization technology to provide shared computing resources as on-demand *Infrastructure-as-a-Service* (IaaS) offerings over the network. A managed virtualization environment is different from a public or private cloud. Both managed virtualization and cloud computing enable administering physical resources as multiple virtual resources. Cloud computing shares a pool of automatically allocated resources as a service. Cloud resources are available through a self-service model and can also offer the benefits of automated management, scaling, and elasticity, whereas virtualization environments offer enterprise-class private data center management.

Managed virtualization and cloud computing each lend themselves to different use cases. One primary determinant for selecting virtualization over cloud is the expected workload. Virtualization is typically the better platform than cloud computing for workloads that require finite resources for a specific application that is available to a defined set of users.

## Virtual Machines versus Containers

A *container* is a process that runs an application independently from other containers on the host. Containers generally serve a similar role to virtual machines, where an application resides in a self-contained environment with virtualized networking for communication. Although both use cases initially seem to be the same, containers have a smaller footprint, and start and stop faster than a virtual machine. For both memory and disk usage, VMs are often measured in gigabytes, whereas containers are measured in megabytes.

Containers are created from a *container image*, which includes all the runtime dependencies of the application. Containers can then be executed on any host, without requiring the installation of any dependency on the host, and without conflicts between the dependencies of different containers.



**Figure 1.1: Applications in containers versus on the host operating system**

Containers use Linux kernel features, such as namespaces and control groups (cgroups). For example, containers use cgroups for resource management, such as CPU time allocation and system memory. Namespaces isolate a container's processes and resources from other containers and the host system.

Containers are preferred for microservices with rapid development and individual scaling needs. A VM is useful when an organization does not want to assign resources to converting legacy or complex applications to containers. A VM is also needed when an additional full computing environment is required, such as when an application requires specific, dedicated hardware. Additionally, a VM is preferable when an application requires a non-Linux operating system or a different kernel from the host.

## Introduction to OpenShift Virtualization

Like other Red Hat virtualization products, OpenShift Virtualization uses *Kernel-based Virtual Machine* (KVM) technology as the hypervisor and the libvirt libraries for defining the virtual machines. KVM and libvirt are run in a container that can be deployed in Kubernetes and OpenShift platforms along with containerized applications. The *Quick EMUlator* (QEMU) guest agent passes information about a VM to the host.

Red Hat OpenShift is a set of modular components and services that are built on top of Kubernetes container infrastructure. The Red Hat OpenShift software product is derived from the Kubernetes open source project. Although Red Hat OpenShift and Kubernetes are container orchestration software, Red Hat OpenShift is packaged as a downstream enterprise open source platform.

The OpenShift Virtualization operator offers features for enterprise-grade virtualization, such as live migration, high availability, system scheduling, power management, image management, snapshots, thin provisioning, and monitoring.

Red Hat OpenShift, with the Virtualization operator installed, manages both containers and virtual machines from a common interface.

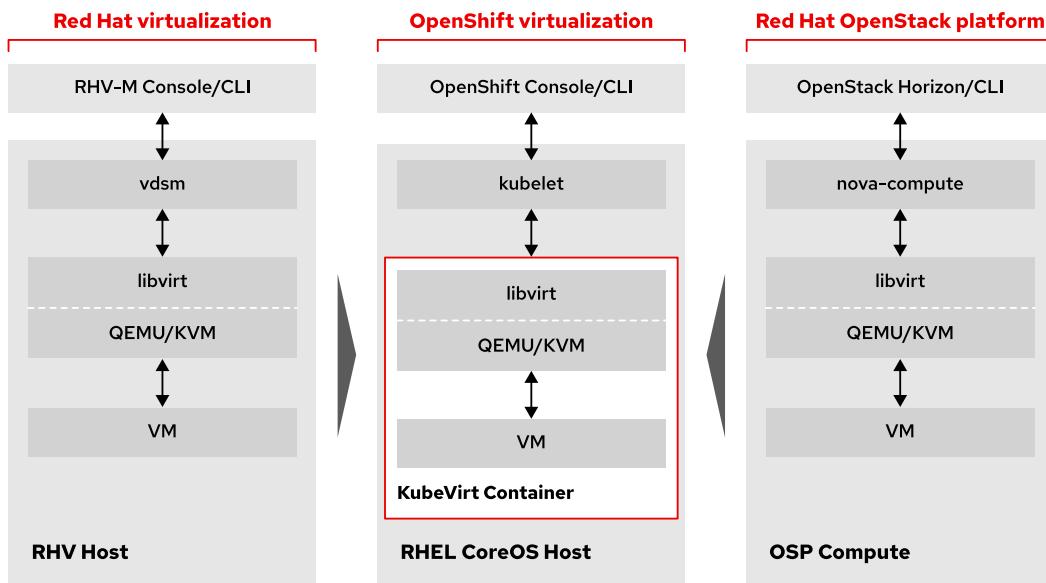


Figure 1.2: Containerizing KVM

## Kubernetes Overview

Kubernetes is a platform that simplifies the deployment, management, and scaling of containerized applications.

Kubernetes creates a cluster that runs applications across multiple nodes and manages pools of resources, such as CPU, RAM, storage, and networking. If a node fails, then Kubernetes can restart an application in a different node to provide high uptime and fault tolerance for an application.

Kubernetes uses a declarative configuration model. Kubernetes administrators write a definition of the workloads to execute in the cluster, and Kubernetes ensures that the running workloads match the definition. For example, an administrator defines several workloads. Each workload defines the amount of required memory. Kubernetes then creates the necessary containers in different nodes, to meet the memory requirements.

Kubernetes defines workloads in terms of *pods*. A pod is one or multiple containers that run in the same cluster node. Pods with multiple containers are useful in certain situations, when two containers must run in the same cluster node to share some resource. For example, a job is a workload that runs a task in a pod until completion.

Kubernetes offers the following features, which can be used with VMs:

### Service discovery and load balancing

Distributing applications over multiple nodes can complicate communication between applications.

Kubernetes automatically configures networking and provides a DNS service for pods. With these features, pods can communicate with services from other pods transparently across

nodes by using only hostnames instead of IP addresses. Multiple pods can back a service for performance and reliability. For example, Kubernetes can evenly split incoming requests to an NGINX web server by considering the availability of the NGINX pods.

### **Self-healing**

If applications declare health check procedures, then Kubernetes can monitor, restart, and reschedule failing or unavailable applications. Self-healing protects applications both from internal failure (the application stops unexpectedly) or external failure (the node that runs the application becomes unavailable).

### **Operators**

Operators are packaged Kubernetes applications that add tooling to manage the complete lifecycle of an application that is deployed in the Kubernetes cluster. The open source operator framework provides guidelines for authoring operators that use the Kubernetes API to add features to the cluster. Each cluster adds the operators with the necessary environment and functions to meet its needs.

## **Features of Red Hat OpenShift**

Red Hat OpenShift is a set of modular components and services that are built on top of Kubernetes container infrastructure. Red Hat OpenShift includes these features for cluster management:

### **Developer workflow**

Integrates a built-in container registry, *Continuous Integration/Continuous Delivery* (CI/CD) pipelines, and *Source-to-Image* (S2I), a tool to build artifacts from source repositories to container images.

### **Metrics and logging**

Includes a built-in and self-analyzing metrics service and aggregated logging based on the upstream Prometheus project.

### **Unified UI**

Provides a unified toolset and an integrated UI to manage the cluster functions.

Red Hat builds key features such as Stateful Sets, role-based access control (RBAC), and network policies. Red Hat also provides critical components of Kubernetes such as the etcd key-value store, which is the cornerstone of Kubernetes that maintains cluster configuration.

## **Red Hat OpenShift Virtualization**

Red Hat OpenShift Virtualization is a Kubernetes operator on Red Hat OpenShift, and is based on the KubeVirt Kubernetes virtualization add-on. KubeVirt delivers container-native virtualization by using the *Kernel-based Virtual Machine* (KVM) Linux Kernel hypervisor within a Kubernetes container. KubeVirt includes similar services to traditional virtualization platforms, to provide the best of mature virtualization management technology and Kubernetes container orchestration. Started by Red Hat, KubeVirt is now a Cloud Native Computing Foundation Incubation project and an open source upstream project.

The following figure shows the resources and services that form a Red Hat OpenShift cluster, which includes the self-service administrator command-line and web interfaces:

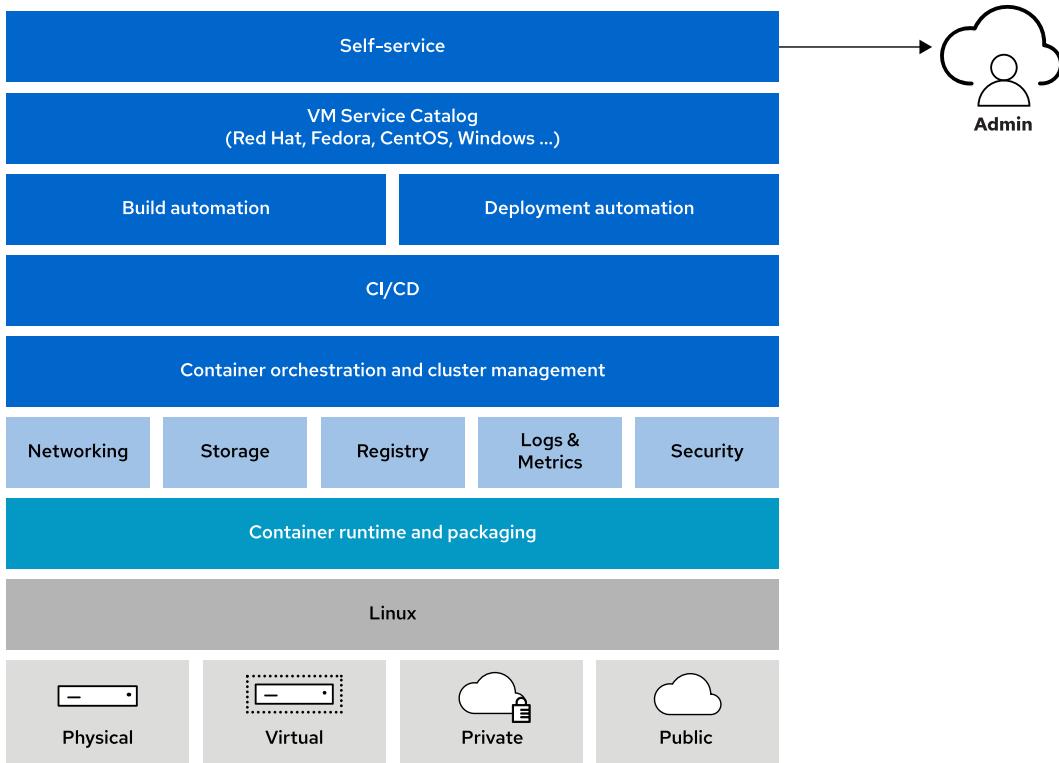


Figure 1.3: Red Hat OpenShift cluster resources

## Virtualization Meets Containerization

This course explores the features of OpenShift Virtualization that provide the management and orchestration layer for virtual machine workloads within a Red Hat OpenShift cluster. With OpenShift Virtualization, you can deliver the resilience and ease of managing both VM-based and containerized applications from a single interface and infrastructure. This capability eliminates the need for a separate and potentially expensive virtualization management solution and creates a valuable opportunity for businesses that are undergoing container adoption and still maintain a traditional deployment of virtual machines.

## Benefits of OpenShift Virtualization

Adding OpenShift Virtualization to a Red Hat OpenShift cluster enables deploying and managing virtualized workloads with the same hardware, environment, and tool set that the containerized applications use. Managing virtual machines alongside containerized deployments in a single cluster provides both administrative and economical advantages. Using OpenShift Virtualization unifies the management interfaces that system administrators use, to improve application reliability and reduce operational overhead.

The following operators and components are integrated for the OpenShift Virtualization operator:

### OpenShift Virtualization Operator

OpenShift Virtualization is an open source technology to enable running VMs in Red Hat OpenShift with the OpenShift Virtualization operator. It is based on the upstream *KubeVirt* project, and runs and manages traditional VMs as if they were containers.

**HyperConverged Operator (HCO)**

The HCO deploys and manages virtualization components and supplemental operators, such as the *Containerized Data Importer (CDI)*, for OpenShift Virtualization.

**Hostpath Provisioner Operator (HPP)**

The HPP operator is a local storage provisioner for virtual machines. Local storage or mounted volumes on nodes can be provisioned to *Virtual Machine Instances (VMIs)*.

**QEMU agent**

The *Quick EMUlator (QEMU)* guest agent is a daemon that runs inside the VM and passes information to the host about the VM, users, file systems, and secondary networks.

**Kernel-based Virtual Machine (KVM)**

KVM is an open source virtualization technology that is built into the Linux kernel that enables a host machine to run multiple, isolated virtual environments, *guest VMs*.

**Containerized Data Importer (CDI)**

An upstream Kubernetes component that delivers a hardware-independent management layer for storage volumes that attach to deployed instances.

**Virtual machine consoles**

Connect to VMs by using command-line tools, the web console, and common protocols, such as *Remote Desktop Protocol (RDP)* for Windows VMs, and SSH.

**Stand-alone Virtual Machine Instances (VMIs)**

Use the command line and the web console to manage stand-alone VMIs that were created independently from the OpenShift cluster, such as VMIs that were created by using scripts, with automation, or with other command-line methods.

**References**

For more information, refer to the *OpenShift Virtualization* documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index)

The upstream KubeVirt project is hosted at  
<https://kubevirt.io>

For more information, refer to the *OpenShift Virtualization - Reference Implementation Guide* article at  
<https://access.redhat.com/articles/7067871>

## ► Quiz

# OpenShift Virtualization

Choose the correct answers to the following questions:

► 1. **What is the primary role of a hypervisor in system virtualization?**

- a. It manages and supports the physical server virtualization, and runs virtual machines for each virtualized operating system.
- b. It manages the applications that run on the virtual machines.
- c. It provides a user interface for managing virtual networks.
- d. It acts as the operating system for the physical server.

► 2. **Which feature does Red Hat OpenShift Virtualization provide for your OpenShift cluster?**

- a. Role-based access controls to manage cluster access optimally
- b. Security audits for the cluster workloads to ensure protection from exploits
- c. Management of VMs alongside containerized applications on common infrastructure
- d. Management of VMware clusters from the OpenShift web console

► 3. **Which open source upstream project is OpenShift Virtualization based on?**

- a. Hyper V
- b. KubeVirt
- c. oVirt
- d. KVM

► 4. **Which three tasks does OpenShift Virtualization provide for an OpenShift cluster through CRDs? (Choose three.)**

- a. Creates and manages Linux and Windows virtual machines.
- b. Provides encryption and security audits for the OpenShift cluster.
- c. Supports monitoring, health checks, and self-healing of deployed virtual machines.
- d. Manages storage allocation for the OpenShift cluster.
- e. Manages a VMware cluster by using open source technology.
- f. Supports connectivity to virtual machines by using common protocols, such as RDP and SSH.

- 5. Which OpenShift Virtualization component runs inside the VM and passes information to the host about the VM, users, file systems, and secondary networks?
- a. KVM monitor
  - b. libvirt daemon
  - c. RDP service
  - d. QEMU guest agent

## ► Solution

# OpenShift Virtualization

Choose the correct answers to the following questions:

► 1. **What is the primary role of a hypervisor in system virtualization?**

- a. It manages and supports the physical server virtualization, and runs virtual machines for each virtualized operating system.
- b. It manages the applications that run on the virtual machines.
- c. It provides a user interface for managing virtual networks.
- d. It acts as the operating system for the physical server.

► 2. **Which feature does Red Hat OpenShift Virtualization provide for your OpenShift cluster?**

- a. Role-based access controls to manage cluster access optimally
- b. Security audits for the cluster workloads to ensure protection from exploits
- c. Management of VMs alongside containerized applications on common infrastructure
- d. Management of VMware clusters from the OpenShift web console

► 3. **Which open source upstream project is OpenShift Virtualization based on?**

- a. Hyper V
- b. KubeVirt
- c. oVirt
- d. KVM

► 4. **Which three tasks does OpenShift Virtualization provide for an OpenShift cluster through CRDs? (Choose three.)**

- a. Creates and manages Linux and Windows virtual machines.
- b. Provides encryption and security audits for the OpenShift cluster.
- c. Supports monitoring, health checks, and self-healing of deployed virtual machines.
- d. Manages storage allocation for the OpenShift cluster.
- e. Manages a VMware cluster by using open source technology.
- f. Supports connectivity to virtual machines by using common protocols, such as RDP and SSH.

- **5. Which OpenShift Virtualization component runs inside the VM and passes information to the host about the VM, users, file systems, and secondary networks?**
- a. KVM monitor
  - b. libvirt daemon
  - c. RDP service
  - d. QEMU guest agent

# Underlying Kubernetes Architecture

## Objectives

- Describe Kubernetes high-level architecture and operators to deploy and support OpenShift Virtualization.

## Kubernetes Overview

Kubernetes is a platform that simplifies the deployment, management, and scaling of containerized applications.

Kubernetes uses a declarative configuration model. Kubernetes administrators write a definition of the workloads to execute in the cluster, and Kubernetes ensures that the running workloads match the definition. For example, an administrator defines several workloads. Each workload defines the amount of required memory. Kubernetes then creates the necessary containers in different nodes, to meet the memory requirements.

Resources are the configurable and consumable components that the cluster manages. These resources are a part of a declarative API with a well-defined schema structure and endpoints. Because the schema and endpoints are predictable and structured, most Kubernetes tools work with any Kubernetes API even if they are not part of the core. OpenShift Container Platform and OpenShift Virtualization both extend the API and provide additional resources.

## Kubernetes Architectural Concepts

Kubernetes uses several servers (also called *nodes*) to ensure the resilience and scalability of the applications that it manages. These nodes can be physical or virtual machines. The nodes are of two types, each of which provides a different aspect of the cluster operations.

*Control plane* nodes provide cluster-wide coordination for scheduling the deployed workloads. These nodes also manage the state of the cluster configuration in response to cluster events and requests.

*Compute plane* nodes run the user workloads.

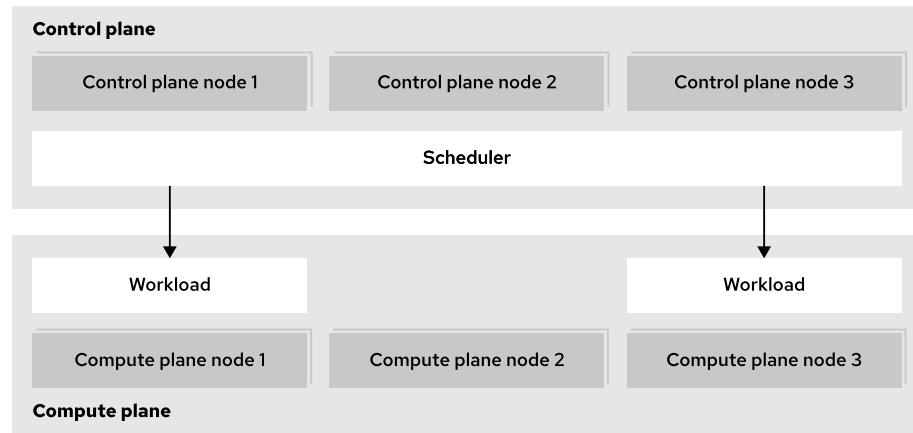


Figure 1.4: Kubernetes components

Although a server can act as both a control plane node and a compute plane node, the two roles are usually separated for increased stability, security, and manageability. Kubernetes clusters can range from small single-node clusters, to large clusters with up to 5,000 nodes.

## The Kubernetes API and Configuration Model

Kubernetes provides a model to define the workloads to run on a cluster.

Administrators define workloads in terms of resources.

All resource types work in the same way, with a uniform API. Tools such as the `kubectl` command can manage resources of all types, even custom resource types.

Kubernetes can import and export resources as text files. By working with resource definitions in text formats, administrators can describe their workloads instead of performing the right sequence of operations to create them. This approach is called *declarative resource management*.

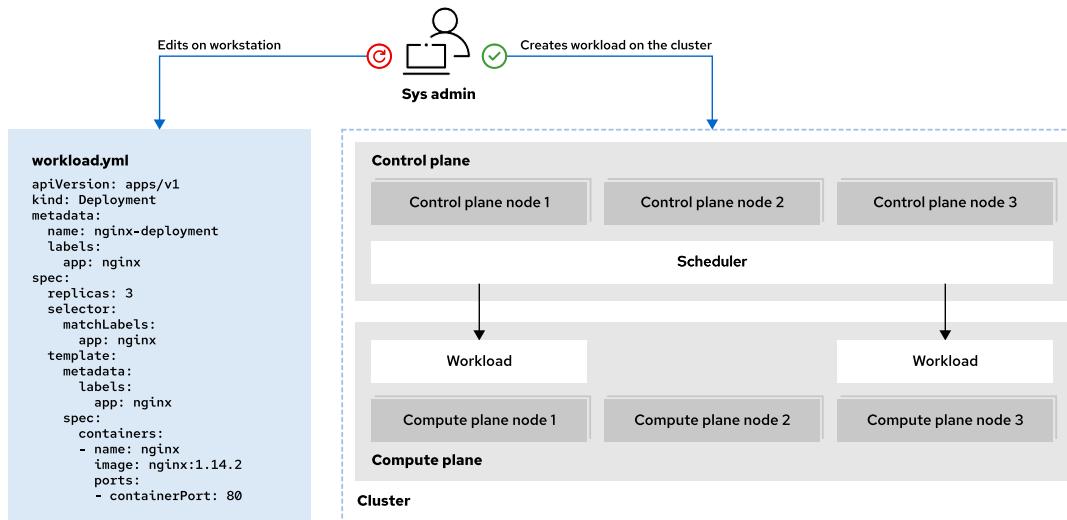


Figure 1.5: Declarative resource management

Declarative resource management can reduce the work to create workloads and improve maintainability. Additionally, when using text files to describe resources, administrators can use generic tools such as version control systems to gain further benefits such as change tracking.

To support declarative resource management, Kubernetes uses *controllers* that continuously track the state of the cluster and perform the necessary steps to keep the cluster in the intended state. This process means that changes to resources often require some time to be effective. However, Kubernetes can automatically apply complex changes. For example, if you increase the RAM requirements of an application that cannot be satisfied on the current node, then Kubernetes can move the application to a node with sufficient RAM. Kubernetes redirects traffic to the new instance only when the movement is complete.

Kubernetes also supports *namespaces*, which provide a mechanism for isolating groups of resources. Administrators can create namespaces, and most resources must be created inside a namespace. Besides helping organize large amounts of resources, namespaces provide the foundations for features such as resource access. Administrators can define permissions for namespaces, to allow specific users to view or modify resources.

## Kubernetes OperatorHub

Operators for Kubernetes are shared through the OperatorHub open source community. Red Hat, in collaboration with Amazon Web Services (AWS), Google Cloud, and Microsoft, launched the OperatorHub, which is accessible at <https://operatorhub.io>.

The OperatorHub platform is a public repository and marketplace for operators that are compatible with OpenShift and other distributions of Kubernetes. From the OperatorHub, you can view operators that provide features for your Kubernetes cluster and find the tools to author custom operators. Operators that are created by using the Operator Framework are shared with the open source community through the OperatorHub.

## Red Hat OpenShift

Kubernetes provides many features to build clusters of servers that run containerized applications. However, rather than providing a complete solution, Kubernetes provides extension points so system administrators can complete Kubernetes. Red Hat OpenShift builds on the extension points of Kubernetes to provide a complete platform.

Red Hat OpenShift adds capabilities to a production platform, such as remote management, multitenancy, increased security, monitoring and auditing, application lifecycle management, and self-service interfaces for developers.

The Red Hat OpenShift product family integrates many components:

- The Red Hat Enterprise Linux CoreOS lightweight operating system
- The CRI-O container runtime engine with a reduced attack surface
- Kubernetes open source container orchestration platform
- A self-service web console
- Several preinstalled application services, such as the logging and monitoring frameworks
- Certified container images for multiple programming language runtimes, databases, and other software packages

This technology stack within Red Hat OpenShift provides enterprise scaling and reliability for modern deployments.

## Red Hat OpenShift Editions

Red Hat offers several editions of the Red Hat OpenShift product that are tailored for different business requirements and needs. These editions include managed offerings from public cloud partners, such as Amazon Web Services, and self-managed editions.

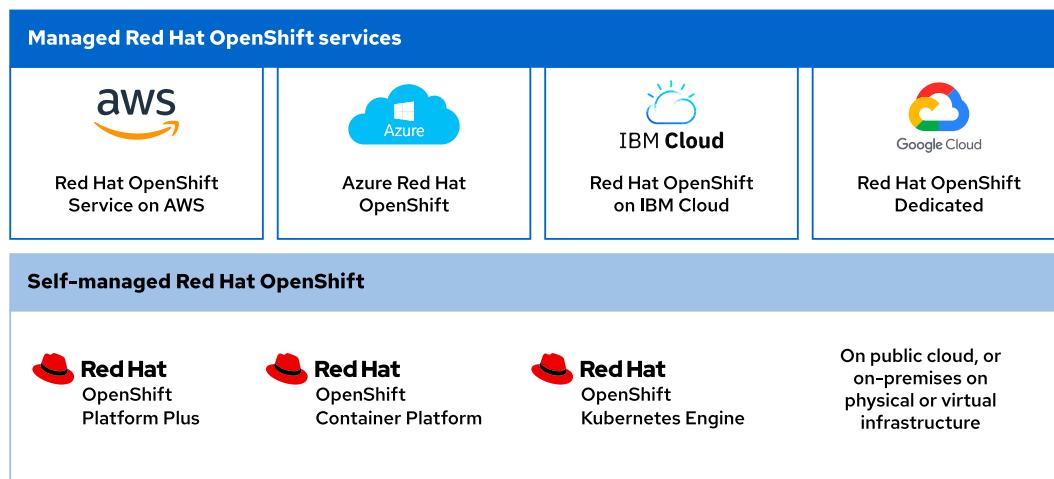


Figure 1.6: Product editions

Red Hat OpenShift Kubernetes Engine includes the latest version of the Kubernetes platform with the additional security hardening and enterprise stability that Red Hat is famous for delivering. This deployment runs on the Red Hat Enterprise Linux CoreOS immutable container operating system, by using Red Hat OpenShift Virtualization for virtual machine management, and provides an administrator console to aid in operational support.

Red Hat OpenShift Container Platform (RHOCP) builds on the features of the OpenShift Kubernetes Engine to include additional cluster manageability, security, stability, and ease of application development for businesses. Additional features of this tier include a developer console, as well as log management, cost management, and metering information. This offering adds Red Hat OpenShift Serverless (Knative), Red Hat OpenShift Service Mesh (Istio), Red Hat OpenShift Pipelines (Tekton), and Red Hat OpenShift GitOps (Argo CD) to the deployment.

Red Hat OpenShift Platform Plus expands further on the offering to deliver the most valuable and robust available features. This offering includes Red Hat Advanced Cluster Management for Kubernetes, Red Hat Advanced Cluster Security for Kubernetes, and the Red Hat Quay private registry platform. For the most complete and full-featured container experience, Red Hat OpenShift Platform Plus bundles all the necessary tools for a complete development and administrative approach to containerized application platform management.

The exercises in this course use a self-managed RHOCP cluster.

## Red Hat Marketplace

Red Hat provides a catalog of certified enterprise operators for installation on Red Hat OpenShift clusters through the Red Hat Marketplace. You can peruse, test, and adopt operators from this curated collection to provide the required features for your cluster. As you evolve your Red Hat OpenShift cluster to add functions and integrate with other business systems, such as storage, the operators in the Red Hat Marketplace can help you manage those resources.

At the onset of a container adoption journey, operators such as Red Hat OpenShift Virtualization can provide varied functions to the cluster that aid the business in a transition from virtual machines to containers.

As container adoption grows, businesses that explore container technologies often need to continue managing and developing existing or new virtual machine deployments. Red Hat OpenShift clusters where the Red Hat OpenShift Virtualization operator is installed handle both virtualized and containerized applications in a single platform to aid your container adoption journey.

## Overview of the Red Hat OpenShift Web Console

The Red Hat OpenShift web console provides a graphical user interface to perform many administrative tasks for managing a cluster. The web console uses the Kubernetes APIs and OpenShift extension APIs to deliver a robust graphical experience. The menus, tasks, and features within the web console are always available by using the CLI. The web console provides ease of access and management for the complex tasks that cluster administration requires.

## Accessing the Red Hat OpenShift Web Console

You access the web console with any modern web browser. The web console URL is generally configurable, and you can discover the address for your cluster web console by using the `oc login -u <USERNAME> -p <PASSWORD> <API_ENDPOINT>:<PORT>` command. From a command-line interface, you must first authenticate to the cluster by using the `oc login -u developer -p developer https://api.ocp4.example.com:6443` command:

```
[user@host ~]$ oc login -u developer -p developer \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

Then, you execute the `oc whoami --show-console` command to retrieve the web console URL:

```
[user@host ~]$ oc whoami --show-console
https://console.openshift-console.apps.ocp4.example.com
```

Lastly, use a web browser to go to the URL, which displays the authentication page:

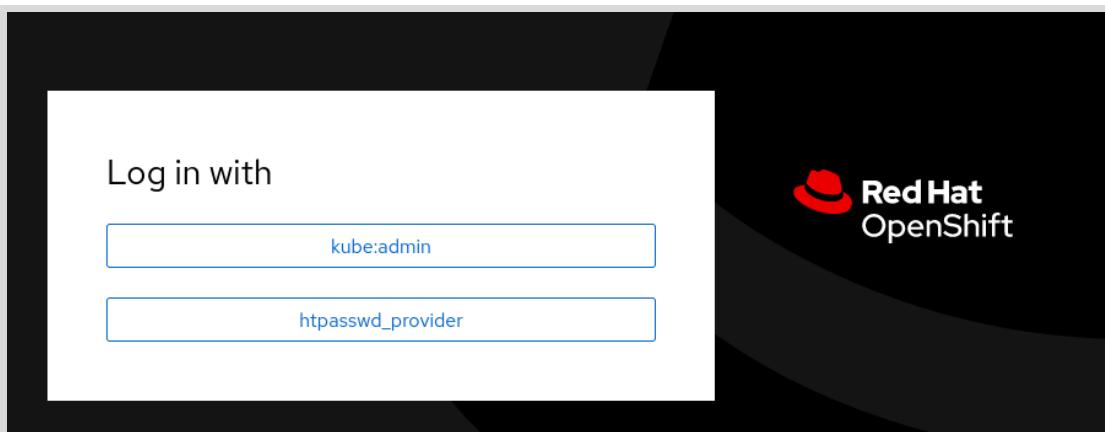


Figure 1.7: The Red Hat OpenShift authentication page

By using the credentials for your cluster access, you reach the home page for the web console.

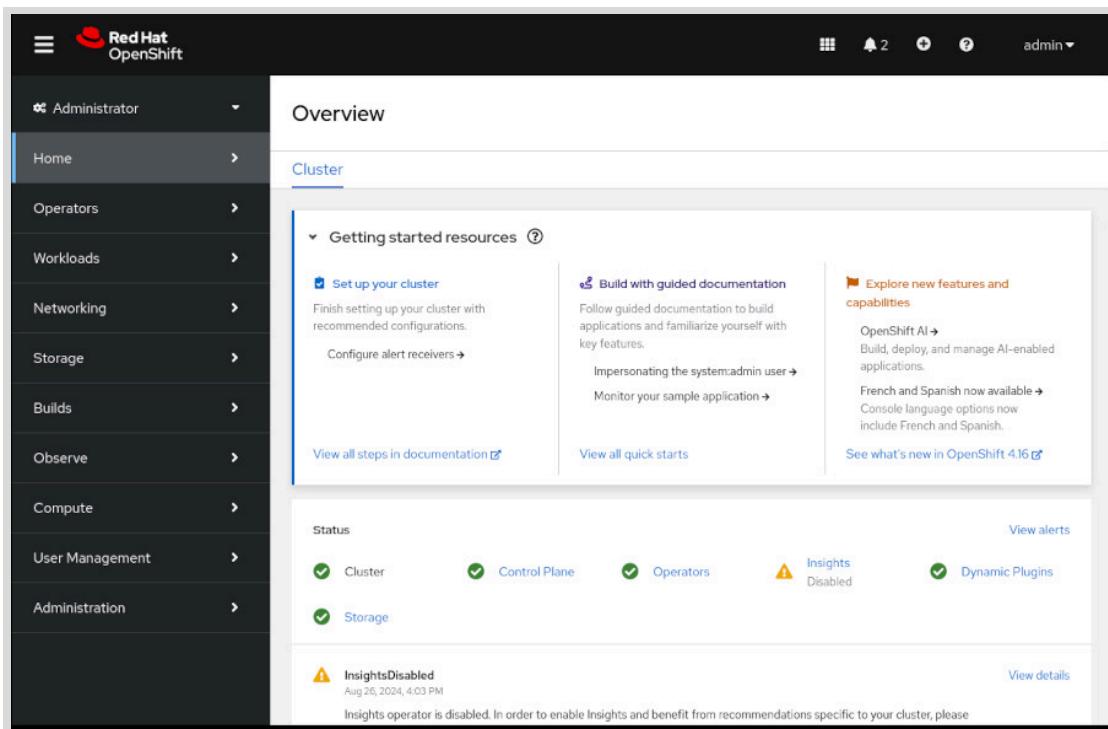


Figure 1.8: The Red Hat OpenShift home page

## Web Console Perspectives

The Red Hat OpenShift web console provides the **Administrator** and **Developer** console perspectives. The sidebar menu layout and the features that it displays differ between using the two console perspectives. The first item on the console sidebar menu is the perspective switcher, to switch between the **Administrator** and the **Developer** perspectives.

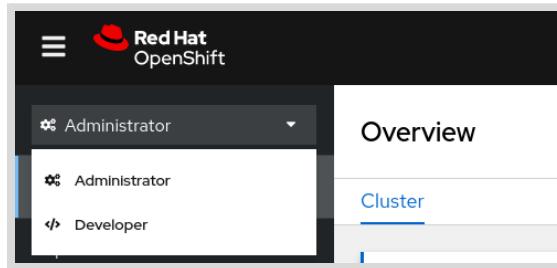
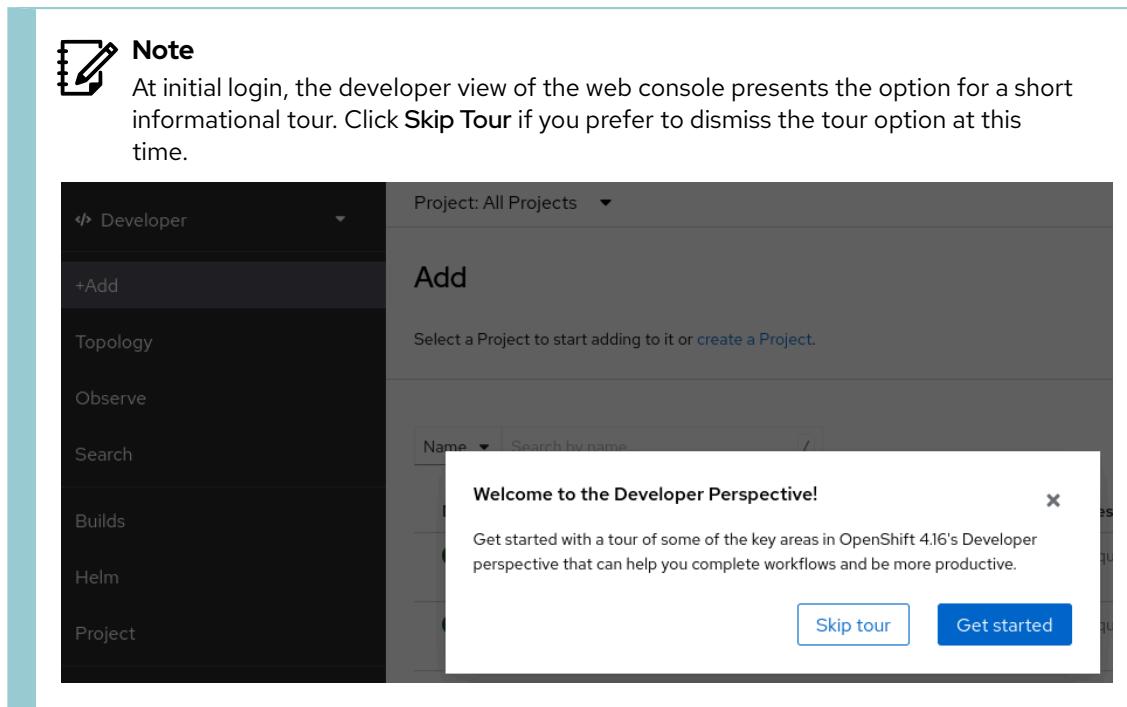


Figure 1.9: The Red Hat OpenShift web console perspective switcher

Each perspective presents the user with different menu categories and pages that cater to the needs of the two separate personas. The **Administrator** perspective focuses on cluster configuration, deployments, and operations of the cluster and running workloads. The **Developer** perspective focus on creating and running applications.

## Red Hat OpenShift Web Console Layout

From the web console home page, the primary navigation method is through the sidebar. The sidebar organizes cluster functions and administration into several major categories. By selecting a category from the sidebar, the category expands to reveal the various areas that each provide specific cluster information, configuration, or functions.



By default, the console displays the **Home > Overview** page, which provides a quick glimpse of initial cluster configurations, documentation, and general cluster status. Go to **Home > Projects** to list all projects in the cluster that are available to the credentials in use.

You might initially peruse the **Operators > OperatorHub** page, which provides access to the collection of operators that are available for your cluster.

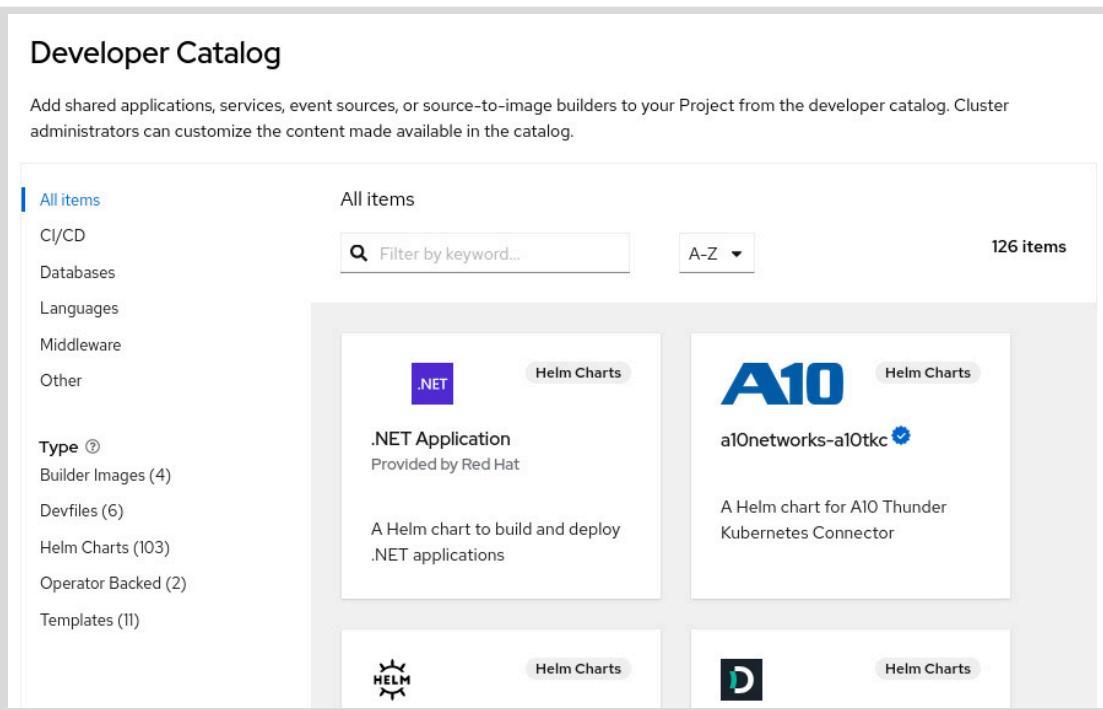
The screenshot shows the Red Hat OpenShift OperatorHub interface. At the top, there is a navigation bar with 'All Items' selected. Below this is a sidebar with categories: Cloud Provider, Developer Tools, Modernization & Migration, Networking, OpenShift Optional, Storage, and Other. Under 'Source', there is a checkbox for 'do316 Operator Catalog' which is checked. Under 'Provider', there is a checkbox for 'Red Hat' which is checked. Under 'Install state', there is a checkbox for 'Cs' which is checked. To the right of the sidebar is a search bar with 'Filter by keyword...' and a count of '11 items'. The main area displays a grid of operator cards. One card for 'Kubernetes NMState Operator' is shown with the description: 'Kubernetes NMState is a declarative means of configuring NetworkManager.' Another card for 'MetallLB Operator' is shown with the description: 'An operator for deploying...' and a note that it is 'Installed'. There are also other cards visible in the grid.

**Figure 1.11: The Red Hat OpenShift OperatorHub**

By adding operators to the cluster, you can extend the features and functions that your OpenShift cluster provides. Use the search filter to find the available operators to enhance the cluster and to supply the required OpenShift aspects.

By clicking the link on the **Operator Hub** page, you can peruse the Developer Catalog.

Select any project, or use the search filter to find a specific project, to visit the Developer Catalog for that project, where shared applications, services, event sources, or source-to-image builders are available.



**Figure 1.12: The Developer Catalog**

After finding the preferred additions for a project, a cluster administrator can further customize the content that the catalog provides. By adding the necessary features to a project with this approach, developers can customize features to provide an ideal application deployment.

## Red Hat OpenShift Key Concepts

When you navigate the Red Hat OpenShift web console, it is useful to know some introductory OpenShift, Kubernetes, and container terminology. The following list includes some basic concepts that can help you to navigate the OpenShift web console.

- Projects: A Kubernetes namespace with additional annotations that provide multitenancy scoping for applications.
- Operators: Packaged Kubernetes applications that extend cluster functions.
- Pods: The smallest unit of a Kubernetes-managed containerized application. A pod consists of one or more containers.
- Deployments: The operational unit that provides granular management of a running application.
- Routes: Networking configuration to expose your applications and services to resources outside the cluster.

- Persistent Volume (PV): Storage resource that a dynamic provisioner or an administrator provisions, to define access to physical storage.
- Persistent Volume Claims (PVC): Storage configuration that binds a PV to a namespace so pods can use it.

## Red Hat OpenShift Command-line Utilities (CLI)

The CLI utilities provide developers with a range of commands that are useful for managing the Red Hat OpenShift cluster and its applications. Each command is translated into an API call, and the response is displayed in the command line.

Developers who are familiar with Kubernetes can use the `kubectl` utility to manage a Red Hat OpenShift cluster. This course uses the `oc` command-line utility, to take advantage of additional Red Hat OpenShift features.

The following list shows the most common `oc` commands:

### **oc login**

Before you can interact with your Red Hat OpenShift cluster, you must authenticate your requests, by using the `login` command.

For example, in this course, you can use the following command:

```
[user@host ~]$ oc login https://api.ocp4.example.com:6443
Username: admin
Password: redhatocp
Login successful.
...output omitted...
```

### **oc get**

Use the `get` command to retrieve a list of selected resources in a project.

You must specify the resource type to list.

For example, the following command returns the list of the pod resources in the current project.

```
[user@host ~]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
console-85dd9766f8-455cc   1/1     Running   1          6d
...output omitted...
```

### **oc describe**

Use the `oc describe` command to print a detailed description of the selected resources.

For example, to view the details of the `console` route resource, use the following command:

```
[user@host ~]$ oc describe route console
Name:           console
Namespace:      openshift-console
Created:        7 days ago
Labels:         app=console
Annotations:    haproxy.router.openshift.io/timeout=5m
Requested Host: console-openshift-console.apps.ocp4.example.com
...output omitted...
```

### oc apply

Use the `apply` command to create or update a Red Hat OpenShift resource by file name or standard input. The input contains the JSON or YAML representation of a Red Hat OpenShift resource.

For example, to create resources from the `pod.yaml` file, use the following command:

```
[user@host ~]$ oc apply -f pod.yaml
pod/quotes-pod created
```

### oc explain

Use the `oc explain` command to describe the fields that are included in the YAML or JSON representation of a resource.

You can use the YAML path to get information about a particular field. The following example shows the `spec` attributes for a service resource:

```
[user@host ~]$ oc explain service.spec
KIND:       Service
VERSION:    v1

FIELD: spec <ServiceSpec>

DESCRIPTION:
  Spec defines the behavior of a service.
  https://git.k8s.io/community/contributors/devel/sig-architecture/api-
  conventions.md#spec-and-status
  ServiceSpec describes the attributes that a user creates on a service.

FIELDS:
  ...output omitted...
```

### oc delete

Use the `delete` command to delete an existing Red Hat OpenShift resource. You must specify the resource type and the resource name.

For example, to delete the `quotes-ui` pod, use the following command:

```
[user@host ~]$ oc delete pod quotes-ui
pod/quotes-ui deleted
```

## oc logs

Use the `logs` command to print the standard output of a pod. This command requires a pod name as an argument. You can print only logs of a container in a pod, which means that the resource type is omitted.

For example, to print the logs from the `react-ui` pod, use the following command:

```
[user@host ~]$ oc logs react-ui  
Compiled successfully!  
...output omitted...
```

The `oc` utility provides equal function to the web console. For a full list of commands, execute `oc --help`. Additionally, you can execute `oc command --help`, for example `oc logs --help` to view the `oc logs` command documentation.



## References

For more information about the web console, refer to the *Web Console* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/web\\_console/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/web_console/index)

For more information about the command-line tools, refer to the *CLI Tools* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/cli\\_tools/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/cli_tools/index)

For more information about Kubernetes concepts, refer to the upstream documentation at

<https://kubernetes.io/docs/concepts/overview>

## ► Guided Exercise

# Access and Identify Key OpenShift Cluster Services

Connect to the OpenShift cluster by using the web console and the `oc` command to view resources that are used with either containerized applications or virtual machine workloads.

## Outcomes

- Explore the features and components of Red Hat OpenShift Container Platform (RHOC) by using the web console.
- Use the web console to describe the cluster nodes, networking, storage, and authentication.
- View the default cluster operators, pods, deployments, and services.

## Before You Begin

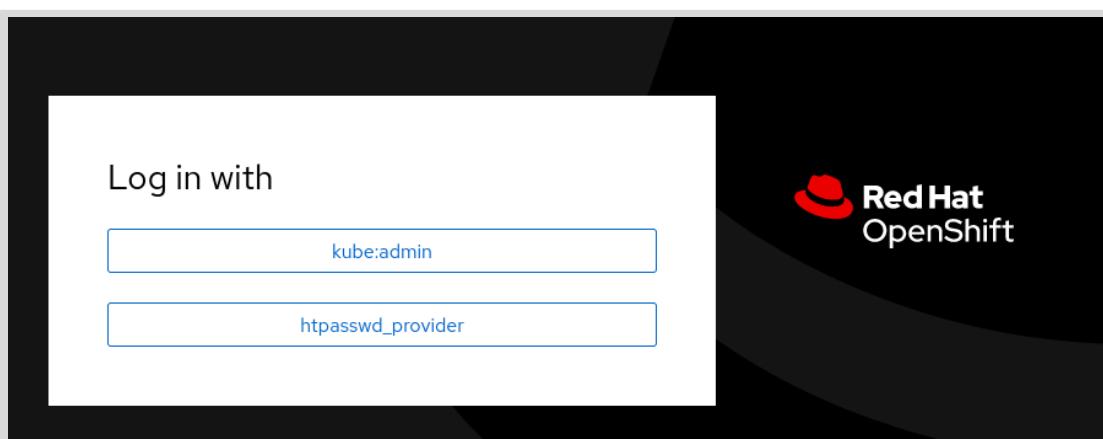
As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster is validated for the exercise.

```
[student@workstation ~]$ lab start virtualization-architecture
```

## Instructions

- 1. Log in to the web console and view the cluster information.
- 1.1. Open a web browser and log in to <https://console-openshift-console.apps.ocp4.example.com> from the `workstation` machine.
  - 1.2. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.



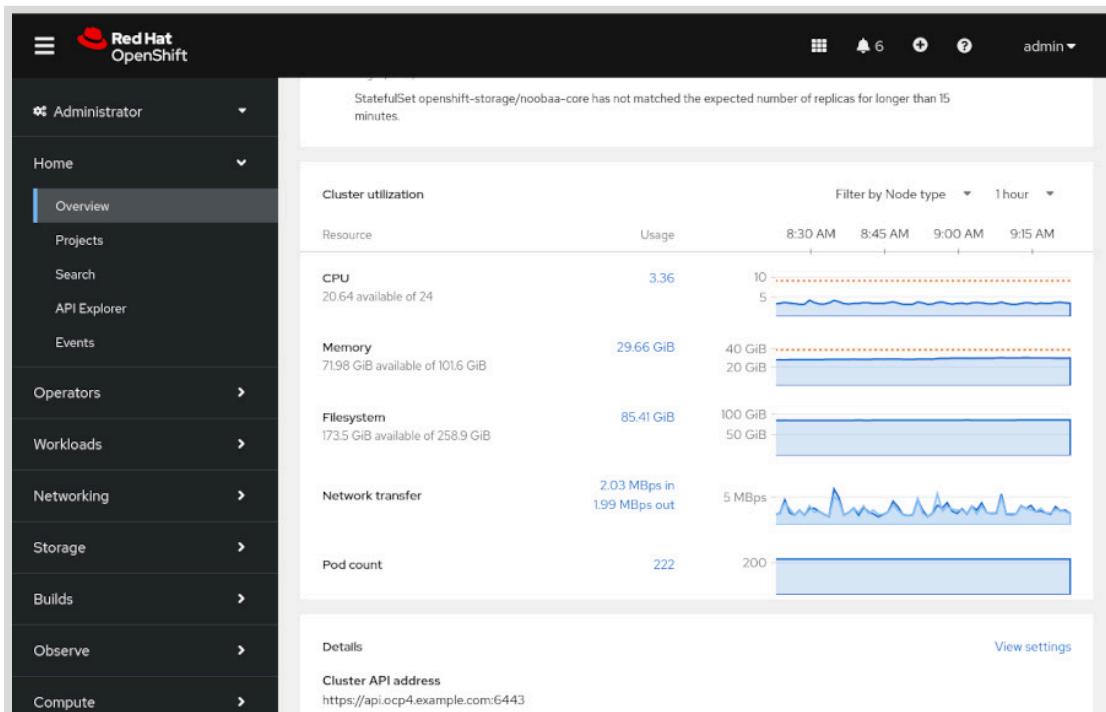
**Note**

When you use a cluster administrator account, you can browse the cluster components, but do not alter or remove any components during this exercise.

- From the Home > Overview page in the Administration view, locate the cluster utilization graphs. Click any graph to see more information about the metrics that are collected.

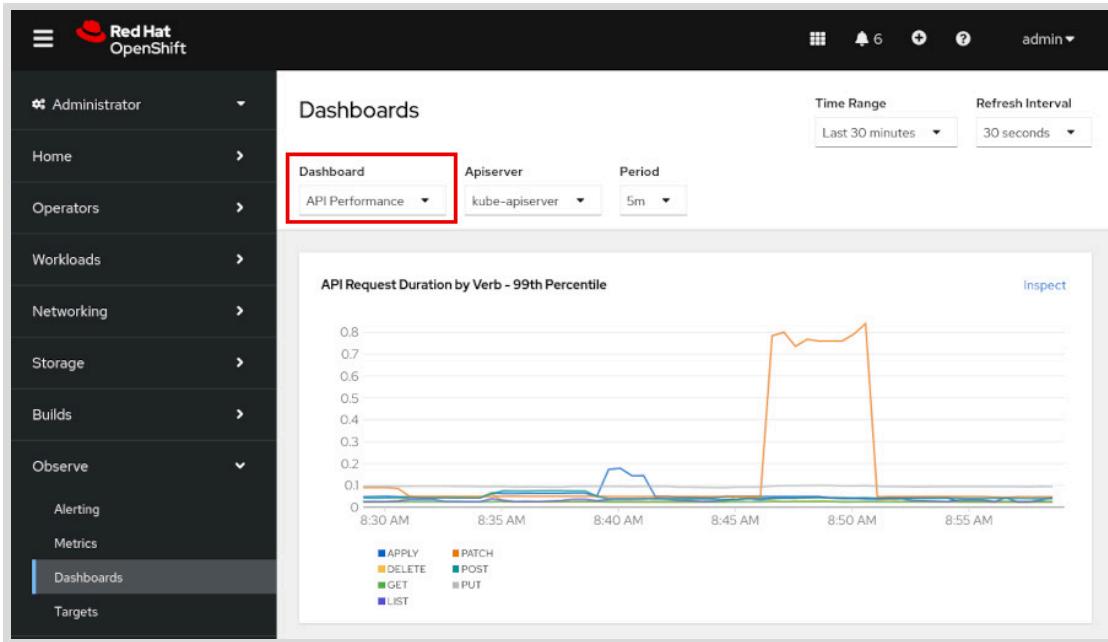
**Note**

Depending on the resolution and zoom level of the browser, you might need to scroll down the page to see the utilization graphs.



▶ 2. Gather information about the cluster.

- From the RHOC4 web console left panel, go to **Observe** > **Dashboards**. If you do not see the left panel, then click the main menu icon at the upper left of the web console. Change dashboards with the pull-down menus near the top of the page.



- 2.2. From the RHOCP web console left panel, go to **Compute > nodes**. The classroom cluster has five nodes: three with the **control-plane** role, and all five with the **worker** role.

The screenshot shows the Red Hat OpenShift web console interface. On the left, there is a navigation sidebar with various links like Home, Operators, Workloads, Networking, Storage, Builds, Observe, Compute, Nodes, Machines, and MachineSets. The 'Nodes' link is currently selected. The main content area is titled 'Nodes' and shows a table listing five nodes. The columns include Name, Status, Roles, Pods, Mem..., CPU, Filesy..., Creat..., and Last Seen. The first three nodes (master 01, master 02, master 03) are listed under the 'control-plane, master, worker' role, while the last two nodes (worker 01, worker 02) are listed under the 'worker' role. All nodes are marked as 'Ready'. A red box highlights the 'Roles' column for the first three nodes.

Name	Status	Roles	Pods	Mem...	CPU	Filesy...	Creat...
master 01	Ready	control-plane, master, worker	62	9.53 GiB / 23.47 GiB	1.092 cores / 4 cores	22.31 GiB / 59.78 GiB	Aug 19, 2024, 10:44 AM
master 02	Ready	control-plane, master, worker	79	9.89 GiB / 23.47 GiB	1.047 cores / 4 cores	23.06 GiB / 59.78 GiB	Aug 19, 2024, 10:44 AM
master 03	Ready	control-plane, master, worker	38	6.29 GiB / 23.47 GiB	0.647 cores / 4 cores	18.2 GiB / 59.78 GiB	Aug 19, 2024, 10:44 AM
worker 01	Ready	worker	17	1.73 GiB / 15.62 GiB	0.114 cores / 6 cores	10.21 GiB / 39.78 GiB	Aug 20, 2024, 10:18 AM
worker 02	Ready	worker	26	2.22 GiB / 15.62 GiB	0.213 cores / 6 cores	11.42 GiB / 39.78 GiB	Aug 20, 2024, 10:18 AM

- 2.3. From the RHOCP web console left panel, go to **Administration > Cluster Settings** and view the version of the RHOCP cluster.

The screenshot shows the 'Cluster Settings' page in the Red Hat OpenShift web console. On the left, there's a navigation sidebar with options like Networking, Storage, Builds, Observe, Compute, User Management, Administration (with 'Cluster Settings' selected), Namespaces, ResourceQuotas, LimitRanges, and CustomResourceDefinitions. The main content area is titled 'Cluster Settings' and has tabs for Details, ClusterOperators, and Configuration. Under 'Details', there's a section for 'Last completed version' (4.16.2) with a link to 'View release notes'. To the right of this, there's an 'Update status' section with a note about the update channel not being configured, and a 'Channel' section showing 'Not configured'. Other sections include 'Subscription' (OpenShift Cluster Manager), 'Cluster ID' (c99bd614-03fd-49c5-9517-5db87993ddff), 'Desired release image' (quay.io/openshift-release-dev/ocp-release@sha256:198ae5a65918351fbdcfeaf4d5c83a16716ed7734ac6cbeea4c47a32bfffad6), and 'Cluster version configuration' (CV version).

► 3. Explore the web console application.

- 3.1. Select Home > Projects, and then search for and select the `openshift-console` project.

The screenshot shows the 'Projects' page in the Red Hat OpenShift web console. The left sidebar includes 'Administrator', 'Home' (with 'Overview' selected), 'Projects' (selected), 'Search', 'API Explorer', and 'Events'. The main area is titled 'Projects' with a 'Create Project' button. A search bar at the top has 'Name' set to 'console'. Below the search bar is a table with columns: Name, Display ..., Status, Requester, Memory, CPU, and Created. The first row in the table is highlighted with a red box and shows a PR icon, the name 'PR openshift-console', 'No display name', 'Active', 'No requester', '135.8 MiB', '0.005 cores', and 'Aug 19, 2024, 10:54 AM'. There are two more rows below it, each with a PR icon and similar details.

- 3.2. To view the pods, click pods in the inventory section of the project overview page, or use the RHOCP web console left panel and go to Workloads > Pods. If you do not see the console pods, then use the Project menu at the top to go to the `openshift-console` project.

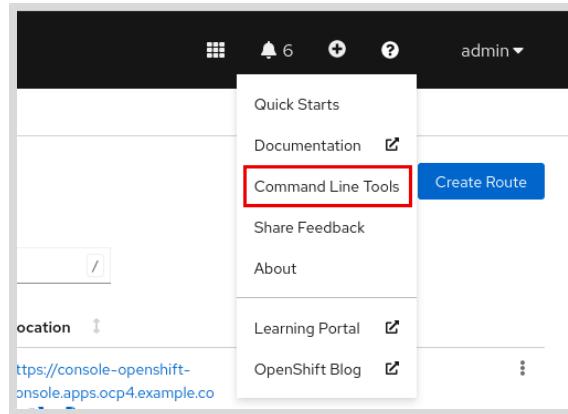
The screenshot shows the Red Hat OpenShift web console interface. The left sidebar has a dark theme with navigation links like Home, Overview, Projects, Search, API Explorer, Events, Operators, Workloads (selected), Pods (selected), Deployments, and DeploymentConfigs. The main content area is titled 'Pods' and shows a table of three pods in the 'openshift-console' project. The table columns are Name, Status, Ready, Restarts, Owner, Memory, CPU, and Created. The first pod is 'console-85dd976f8-2ft4z' (Running, 1/1 ready, 2 restarts, owner 'f8', 40.6 MiB memory, 0.002 cores CPU, created Aug 20, 2024, 11:22 AM). The second is 'console-85dd976f8-455cc' (Running, 1/1 ready, 2 restarts, owner 'f8', 31.2 MiB memory, 0.001 cores CPU, created Aug 20, 2024, 11:22 AM). The third is 'download-5-69678789-986-48mlm' (Running, 1/1 ready, 4 restarts, owner 'f8', 33.3 MiB memory, 0.001 cores CPU, created Aug 19, 2024, 11:21 AM). A red box highlights the 'Project: openshift-console' dropdown at the top of the content area.

- 3.3. Go to **Networking > Routes** to see the resources that enable external access to the application. The **console** route has the URL to access the web console.

The screenshot shows the Red Hat OpenShift web console interface. The left sidebar has a dark theme with navigation links like Home, Operators, Workloads, Networking (selected), Services, Routes (selected), Ingresses, NetworkPolicies, and Storage. The main content area is titled 'Routes' and shows a table of two routes in the 'openshift-console' project. The table columns are Name, Status, Location, and Service. The first route is 'console' (Accepted status, location 'https://console-openshift-console.apps.ocp4.example.com', service 'console'). The second route is 'downloads' (Accepted status, location 'https://downloads-openshift-console.apps.ocp4.example.com', service 'downloads'). A red box highlights the location URL for the 'console' route.

- 4. Access the cluster by using the oc command.

- 4.1. Click the question mark menu in the upper-left area of the web console page and select **Command Line Tools**.



- 4.2. The "Download oc" URLs use the downloads route from earlier. Following the "Copy login command" link steps through creating a token for login. Creating a token and a login command for the **admin** user is optional, because the exercises use a username and a password for login.

**Note**

You do not need to download the `oc` command, because it is already installed on the workstation machine.

The screenshot shows the 'Command Line Tools' page. At the top, there is a 'Copy login command' button. Below it, the heading 'oc - OpenShift Command Line Interface (CLI)' is displayed. A paragraph explains that the OpenShift command line interface allows creating applications and managing OpenShift projects from a terminal. It notes that the `oc` binary offers the same capabilities as the `kubectl` binary but is further extended to support OpenShift Container Platform features. A bulleted list provides download links for various platforms:

- [Download oc for Linux for x86\\_64](#)
- [Download oc for Mac for x86\\_64](#)
- [Download oc for Windows for x86\\_64](#)
- [Download oc for Linux for ARM 64](#)
- [Download oc for Mac for ARM 64](#)
- [Download oc for Linux for IBM Power, little endian](#)
- [Download oc for Linux for IBM Z](#)
- [LICENSE](#)

At the bottom of the page, there is a section for 'helm - Helm 3 CLI'.

- 4.3. Use the command line to log in to your Red Hat OpenShift cluster as the **admin** user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 5. Use the oc command-line interface to verify the status of applications.

- 5.1. Change to the openshift-console project.

```
[student@workstation ~]$ oc project openshift-console
Now using project "openshift-console" on server "https://
api.ocp4.example.com:6443".
```

- 5.2. List the pods with their IP addresses.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE	READINESS	GATES				
console-6748d84b5f-6v6h9	1/1	Running	1	3d1h	10.8.0.27	master01
master01	<none>	<none>				
console-6748d84b5f-c24f9	1/1	Running	1	3d1h	10.9.0.20	master02
master02	<none>	<none>				
downloads-6967878986-78dcv	1/1	Running	1	3d1h	10.8.0.26	master01
master01	<none>	<none>				
downloads-6967878986-chqnx	1/1	Running	1	3d1h	10.11.0.16	worker01
worker01	<none>	<none>				

- 5.3. Query the API for a description of the service resource type.

```
[student@workstation ~]$ oc explain service
KIND:       Service
VERSION:    v1

DESCRIPTION:
  Service is a named abstraction of software service (for example, mysql)
  consisting of local port (for example 3306) that the proxy listens on, and
  the selector that determines which pods will answer requests sent through
  the proxy.
...output omitted...
```

- 5.4. Describe the details of the console route resource. The endpoint IP addresses match the console pod IP addresses.

```
[student@workstation ~]$ oc describe route console
...output omitted...
Service:     console
Weight:      100 (100%)
Endpoints:   10.8.0.27:8443, 10.9.0.20:8443
```

- 5.5. View the persistent volumes in the cluster.

```
[student@workstation ~]$ oc get pv
NAME                                     CAPACITY   ACCESS MODES  RECLAIM
POLICY      STATUS     CLAIM                                     STORAGECLASS
VOLUMEATTRIBUTESCLASS  REASON    AGE
pvc-2fe17e71-4be8-49c8-a02e-3a8d22c309ac  50Gi     RWO          Delete
                                         Bound     openshift-storage/db-noobaa-db-pg-0
                                         <unset>           ocs-external-
                                         storagecluster-ceph-rbd
                                         6d16h
pvc-520e6df1-bef3-4796-ad47-d95035031d2b  20Gi     RWX          Delete
                                         Bound     openshift-image-registry/registry-claim
                                         <unset>           nfs-storage
                                         8d
```

5.6. Display the persistent volume claims across all namespaces.

```
[student@workstation ~]$ oc get pvc -A
NAMESPACE        NAME          STATUS   VOLUME
CAPACITY   ACCESS MODES  STORAGECLASS
VOLUMEATTRIBUTESCLASS  AGE
openshift-image-registry  registry-claim  Bound   pvc-520e6df1-bef3-4796-
ad47-d95035031d2b  20Gi     RWX          nfs-storage
                     <unset>           8d
openshift-storage       db-noobaa-db-pg-0  Bound   pvc-2fe17e71-4be8-49c8-
a02e-3a8d22c309ac  50Gi     RWO          ocs-external-storagecluster-ceph-rbd
                     <unset>           6d16h
```

5.7. List the status and role for each node in the cluster.

```
[student@workstation ~]$ oc get nodes
NAME      STATUS   ROLES          AGE   VERSION
master01  Ready    control-plane, master, worker  8d    v1.29.6+aba1e8d
master02  Ready    control-plane, master, worker  8d    v1.29.6+aba1e8d
master03  Ready    control-plane, master, worker  8d    v1.29.6+aba1e8d
worker01  Ready    worker          6d17h  v1.29.6+aba1e8d
worker02  Ready    worker          6d17h  v1.29.6+aba1e8d
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish virtualization-architecture
```

# Deploying the OpenShift Virtualization Operator

## Objectives

- Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift environment.

## OpenShift Virtualization Components

OpenShift Virtualization supports managing and orchestrating VMs with similar tools and methods as for container management. With these shared tools and methods, administrators and developers can centralize infrastructure management.

## The OpenShift Virtualization Operator

The following diagram shows some of the components of the OpenShift Virtualization operator.

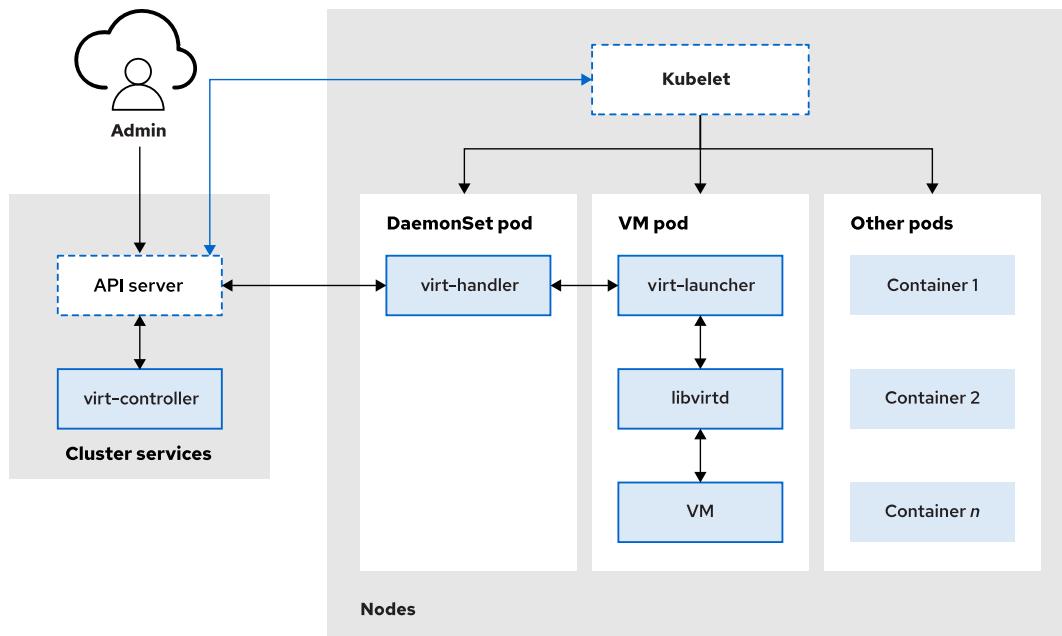


Figure 1.23: OpenShift Virtualization Operator components

The components of the OpenShift Virtualization operator are described as follows:

### `virt-api`

A cluster-level component that provides an HTTP RESTful entry point to manage VM and VM-related workflows within the cluster. It updates virtualization Custom Resource Definitions (CRD) and handles defaulting and validating Virtual Machine Instance (VMI) CRDs.

### `virt-controller`

A cluster-level component that handles cluster-wide virtualization functions and manages the lifecycle of pods that are associated with VMIs. The operator creates the pod where the VM objects execute.

**virt-handler**

A host-level DaemonSet resource, which runs on each node, and monitors for changes to a VM object and runs the necessary operations to meet the required state.

**virt-launcher**

The primary container in a VMI-associated pod runs the `virt-launcher` component to provide control groups (cgroups) and namespaces for hosting the VMI process. This processing occurs when the `virt-handler` component passes the VM CRD object to the `virt-launcher` component. The `virt-launcher` component uses a container-local `libvirtd` instance to start the VMI. The `virt-launcher` component then monitors the VMI process until the process exits at VMI termination.

**libvirtd**

Every VMI pod has a `libvirtd` instance for the `virt-launcher` component to manage the lifecycle of a VMI process.

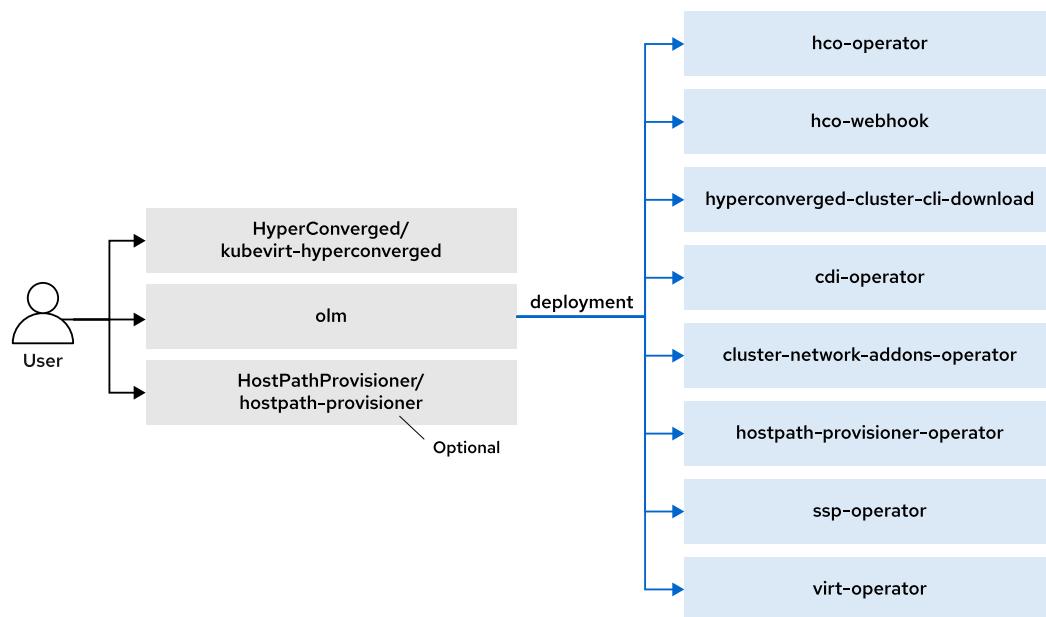
## The HyperConverged Operator

A required component of OpenShift Virtualization is the HyperConverged Operator (HCO), which includes the following resources:

Resource name	Description
deployment/hco-webhook	Validates the HyperConverged custom resource contents.
deployment/hyperconverged-cluster-cli-download	Provides the <code>virtctl</code> tool binaries to the cluster so that you can download them directly from the cluster.
kubevirt/kubevirt-kubevirt-hyperconverged	Contains all needed operators, CRs, and objects for OpenShift Virtualization.
ssp/ssp-kubevirt-hyperconverged	A Scheduling, Scale, and Performance (SSP) CR.
cdi/cdi-kubevirt-hyperconverged	A Containerized Data Importer (CDI) CR.
networkaddonsconfig/cluster	A CR that instructs and is managed by the Cluster Network Addons operator.

## Operator CRs

The chosen options during the HCO installation determine the operators that are deployed.

**Figure 1.24: Deployed operators**

The default settings include the following operator CRs:

#### **Containerized Data Importer (CDI)**

The CDI operator components manage the authorization to upload VM disks into PVCs. The CDI also directs external disk upload traffic to the appropriate upload server pod so that it can be written to the correct PVC and provides a helper pod that imports a virtual machine image into a PVC when creating a data volume.

#### **Hostpath Provisioner (HPP)**

In an environment without available network storage, you can deploy the hostpath provisioner to provision VMs on a node's local storage. The HPP operator includes a deployment resource to provide a worker pod for each node where the HPP is designated to run. The pods mount the backing storage on the node. The operator also deploys two daemonsets with driver interfaces of the HPP.

#### **Cluster Network Addons**

The Cluster Network Addons operator deploys networking components on a cluster and manages the related resources for extended network functions.

#### **Scheduling, Scale, and Performance (SSP)**

The SSP operator deploys common templates, related default boot sources, pipeline tasks, and the template validator.

## **Preparing an OpenShift Cluster for Virtualization**

You can deploy Red Hat OpenShift clusters in many environments. Adding the virtualization operator to the cluster requires extra specifications. Red Hat OpenShift clusters on bare metal instances that are configured on-premise or that IBM Cloud or Amazon Web Services provide are suitable for using the virtualization operator within a cluster. The virtualization operator is also included with the OpenShift Kubernetes Engine and Red Hat OpenShift Service on AWS (ROSA) offerings.

**Note**

If you install OpenShift Virtualization on single-node OpenShift, the following features are not supported:

- High availability
- Pod disruption
- Live migration
- Virtual machines or templates where an eviction strategy is configured

Before installation, verify that a Red Hat OpenShift cluster meets the minimum requirements to support the features of the OpenShift Virtualization operator.

- All CPUs must be supported by Red Hat Enterprise Linux 9, support Intel 64 or AMD64 CPU extensions, have Intel VT or AMD-V hardware virtualization enabled, and have the no-execute (NX) flag enabled.
- Compute nodes must exceed the requirements of the virtual machines in the cluster.
- For high availability, either use installer-provisioned infrastructure with machine health checks or manually monitor node availability.
- For live migration, shared storage is required.
- You must enable Operator Lifecycle Manager on restricted networks for deployment in a disconnected environment.
- You must create a default OpenShift Virtualization or OpenShift Container Platform storage class by setting the `storageclass.kubevirt.io/is-default-virt-class=true` or `storageclass.kubernetes.io/is-default-class=true` annotation on the storage class.
- If you deploy OpenShift Virtualization with Red Hat OpenShift Data Foundation, then you must create a dedicated storage class for Windows virtual machine disks. For more information, refer to the following article: <https://access.redhat.com/articles/6978371>

**Note**

Red Hat recommends not exceeding tested maximums to maintain performance. Review the tested maximums for Red Hat OpenShift Virtualization in the *Supported Limits for OpenShift Virtualization 4.x* article at <https://access.redhat.com/articles/6571671>

## Installing the Red Hat OpenShift Virtualization Operator

As with other operators, you can install the OpenShift Virtualization operator by using either the web console or the command line. To install the operator by using the web console, go to Operators > Operator Hub in the left panel. Type **OpenShift Virtualization** in the search bar, and select **OpenShift Virtualization**.

Configure your preferred channel and version options, and click **Install** to open the **Install Operator** page.

**Update channel \***

stable

**Version \***

4.16.2

**Installation mode \***

All namespaces on the cluster (default)  
This mode is not supported by this Operator

A specific namespace on the cluster  
Operator will be available in a single Namespace only.

**Installed Namespace \***

Operator recommended Namespace: `openshift-cnv`

Select a Namespace

**OpenShift Virtualization**  
provided by Red Hat

**Provided APIs**

**HC** OpenShift Virtualization Deployment Required

Represents the deployment of OpenShift Virtualization

**HPP** HostPathProvisioner Deployment

Represents the deployment of HostPathProvisioner

**Figure 1.25: OpenShift Virtualization Operator installation**

Review or augment the installation options and click **Install** to deploy the operator to the `openshift-cnv` namespace.

OpenShift Virtualization requires that the KubeVirt HyperConverged Operator is installed to deploy and manage OpenShift Virtualization and its components. The HCO also creates Custom Resources to deploy and manage supplemental operators for OpenShift Virtualization, which include the CDI, SSP, and Cluster Network Addons operators.

When the OpenShift Virtualization operator installs successfully, the web console displays a prompt to create a HyperConverged instance.

The screenshot shows the OpenShift Virtualization operator page. At the top, it displays the operator name "OpenShift Virtualization" and version "kubevirt-hyperconverged-operator.v4.16.2 provided by Red Hat". A green checkmark icon indicates the operator is installed successfully. Below this, a section titled "Installed operator: custom resource required" contains a brief description: "The Operator has installed successfully. Create the required custom resource to be able to use this Operator." A card for the "HyperConverged" operator is shown, marked as "Required". It describes the operator as "Creates and maintains an OpenShift Virtualization Deployment". At the bottom of the card are two buttons: "Create HyperConverged" (which is highlighted with a red border) and "View installed Operators in Namespace openshift-cnv".

**Figure 1.26: Post-installation creation of a HyperConverged instance**

When you create the HyperConverged instance, you can set configuration parameters. For example, node placement for virtualization components, a global eviction strategy, tuning policies, and live migration policies are available during installation, and you can also update these settings after installation.

**infra**

infra HyperConvergedConfig influences the pod configuration (currently only placement) for all the infra components needed on the virtualization enabled cluster but not necessarily directly on each node running VMs/VMIs.

**nodePlacement**

NodePlacement describes node scheduling configuration.

- affinity** >
 

affinity enables pod affinity/anti-affinity placement expanding the types of constraints that can be expressed with nodeSelector. affinity is going to be applied to the relevant kind of pods in parallel with nodeSelector. See <https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#affinity-and-anti-affinity>
- tolerations** >
 

tolerations is a list of tolerations applied to the relevant kind of pods. See <https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/> for more info. These are additional tolerations other than default ones.

**workloads** >

workloads HyperConvergedConfig influences the pod configuration (currently only placement) of components

**Figure 1.27: HyperConverged node placement settings**

Click **Create HyperConverged** to create a HyperConverged instance in your cluster.



### Note

After the first HyperConverged instance creation completes, you must refresh the web console browser page before the virtualization menu items appear. Sometimes, you must log out and log back in to the web console before seeing the new features.

OpenShift Virtualization and HyperConverged instance parameters are discussed in greater detail throughout the course.

## Installation from the Command Line

To use the command-line installation method, create a YAML file to define the needed components for the subscription, and apply it by using the `oc apply -f file.yaml` command.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-cnv
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: kubevirt-hyperconverged-group
```

```

namespace: openshift-cnv
spec:
  targetNamespaces:
    - openshift-cnv ①
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.16.2 ②
  channel: stable ③

```

- ① The project namespace for the virtualization operator installation (default: `openshift-cnv`)
- ② The operator version to install
- ③ The source channel for the operator installation

After applying the subscription, create a YAML file to define the HCO operator, and apply it by using the `oc apply -f file.yaml` command.

```

apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged ①
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv ②
spec: {} ③

```

- ① The required object type
- ② The project namespace that is specified during operator installation
- ③ The desired state of HyperConverged resource

Use the `oc explain` command to describe the `spec` fields of the HyperConverged resource.

```

[student@workstation ~]$ oc explain hyperconverged.spec
GROUP:      hco.kubevirt.io
KIND:       HyperConverged
VERSION:    v1beta1

FIELD: spec <Object>

DESCRIPTION:
  HyperConvergedSpec defines the desired state of HyperConverged

FIELDS:
  applicationAwareConfig      <Object>
    ApplicationAwareConfig set the AAQ configurations

```

```

certConfig      <Object>
  certConfig holds the rotation policy for internal, self-signed certificates

commonBootImageNamespace      <string>
  CommonBootImageNamespace override the default namespace of the common boot
  images, in order to hide them.

...output omitted...

```

## HyperConverged Configuration

You can configure some HyperConverged settings in the form view during installation with the web console. You can configure any settings with edits to the YAML during or after installation, from the web console or the command line.

### Deployment Strategy

You can specify nodes by using node selectors, affinity rules, and tolerations for different virtualization workloads. With these tools, you can specify which nodes virtualization components are placed on. You can specify node placement rules for both virtual machines and operators. To set node placement rules, specify `Node Placement` under `Workloads` or `Infra` during deployment, or edit the YAML file for the chosen object after deployment.

### Live Migration Configuration

Live migration limits and timeouts are applied so that migration processes do not overwhelm the cluster.

### Eviction Strategy

`EvictionStrategy` defines at the cluster level whether to migrate the VMI instead of shutting it off when nodes are placed in maintenance mode and drained of all workloads. If the VMI-specific eviction strategy field is set, then it overrides the cluster-level field.

### Feature Gates

KubeVirt has a set of features that are not mature enough to be enabled by default. These features are protected by *feature gates*, where Kubernetes uses key=value pair settings to enable or disable features. Feature gates for OpenShift Virtualization 4.16 are enabled to import common boot images and to run as non-root.

## Creating Virtual Machines

You can create virtual machines by using the OpenShift Virtualization catalog, which provides virtual machine templates and instance types for various operating systems, including Red Hat Enterprise Linux and Microsoft Windows. To access the catalog, use the OpenShift web console and go to **Virtualization > Catalog**.

You can also create virtual machines by using volumes, the Quick Start guided tour, or by applying a YAML file in the web console or from the command line. Different methods are used through this course.



### Note

Only the default quick start guides are available within the classroom cluster.

Virtualization quick start guides can be added alongside the operator installation in clusters with access to the internet.

To begin the Quick Start guided tour, select **Help > Quick Starts**. The Help menu is represented by a ? icon. A search field is available to find quick start guides that relate to virtualization, if available in the cluster. Several virtualization-related quick start guides exist, to introduce various OpenShift Virtualization features. These guides assist new administrators with such tasks as creating VMs from templates or volumes, and uploading a boot source.

The screenshot shows the 'Quick Starts' section of the OpenShift Virtualization interface. At the top, there is a search bar with the text 'virtual' typed into it. Below the search bar, there are four items listed:

- Create a virtual machine from a template**: Duration 5 minutes. Description: Quick create a virtual machine from a template that already has a boot source.
- Create a virtual machine from a volume**: Duration 5 minutes. Description: Quick create a virtual machine from a volume with InstanceType.
- Creating a Windows boot source to use with OpenShift Pipelines**: Duration 10 minutes. Description: Creating a Windows boot source to use with OpenShift Pipelines.
- Upload a boot source**: Duration 10 minutes. Description: Upload a boot source.

After selecting a Quick Start guide, click **Start** to begin the guided tour. After completing the Quick Start guide, you can access the new virtual machine from **Virtualization > VirtualMachines**.

## The `virtctl` Tool

The `virtctl` client is an extra command-line utility for managing virtualization resources from the command line. This utility can start, stop, and connect to a virtual machine console. To view all the subcommands, use the `virtctl help` command.

You can download the `virtctl` client from the cluster after OpenShift Virtualization is installed, by selecting **Help > Command Line Tools**. The Help menu is represented by a ? icon.



## References

For more information about installing from the web console, refer to *Installing OpenShift Virtualization by Using the Web Console* at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-installing-virt-operator\\_installing-virt](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-installing-virt-operator_installing-virt)

For more information about installing from the command line, refer to *Installing OpenShift Virtualization by Using the CLI* at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#installing-virt-operator-cli\\_installing-virt](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#installing-virt-operator-cli_installing-virt)

For more information about the hostpath provisioner, refer to the *Configuring Local Storage for Virtual Machines* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-configuring-local-storage-with-hpp](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-configuring-local-storage-with-hpp)

For more information about specifying nodes, refer to *Specifying Nodes for OpenShift Virtualization Components* at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-node-placement-virt-components](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-node-placement-virt-components)

For more information about creating virtual machines, refer to the *Creating VMs from Red Hat Images* section at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#creating-vms-from-red-hat-images](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#creating-vms-from-red-hat-images)

For more information about features that are included with OpenShift Kubernetes Engine, refer to the *About OpenShift Container Engine* at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/about/index#oke-about](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/about/index#oke-about)

For more information about feature gates, refer to the *Activating Feature Gates* section of the KubeVirt user guide at  
[https://kubevirt.io/user-guide/cluster\\_admin/activating\\_feature\\_gates](https://kubevirt.io/user-guide/cluster_admin/activating_feature_gates)

For more information about virtualization on ROSA, refer to the *OpenShift Virtualization on Red Hat OpenShift Service on AWS (ROSA)* article from Amazon:  
<https://aws.amazon.com/blogs/ibm-redhat/openshift-virtualization-on-red-hat-openshift-service-on-aws-rosa>

## ► Guided Exercise

# Deploy the OpenShift Virtualization Operator

Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift environment and create a virtual machine to verify that the operator functions.

### Outcomes

- Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift Container Platform (RHOCP) cluster by using the web console.
- Review the installed operators and confirm that the OpenShift Virtualization operator is installed.
- Create a CentOS virtual machine from a preconfigured instance.

### Before You Begin

As the student user on the **workstation** machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable and creates the required resources for this exercise.

```
[student@workstation ~]$ lab start virtualization-deploy
```

### Instructions

- 1. Log in to the web console.
  - 1.1. Open a web browser and log in to <https://console-openshift-console.apps.ocp4.example.com> from the **workstation** machine.
  - 1.2. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- 2. From the web console left panel, go to **Operators > OperatorHub** and verify that **All Projects** is selected in the **Project** field. If you do not see the left panel, then click the main menu icon at the upper left of the web console.

The screenshot shows the Red Hat OpenShift web interface. On the left, there's a sidebar with a user dropdown set to 'Administrator'. Below it are 'Home', 'Operators' (which has 'OperatorHub' and 'Installed Operators' listed), and 'Workloads'. At the top right, there's a dropdown for 'Project: All Projects'. The main content area is titled 'OperatorHub' and contains a brief description about discovering operators from the Kubernetes community and Red Hat partners. It includes tabs for 'All Items' (selected) and 'Cloud Provider' and 'Developer Tools', and a search bar with a placeholder 'Filter by keyword...'. The 'OperatorHub' link in the sidebar is also highlighted with a red box.

- 3. Type **virtualization** in the OperatorHub search bar. Scroll down and select **OpenShift Virtualization** from the list.

The screenshot shows the Red Hat OpenShift OperatorHub search results for 'virtualization'. On the left, there are filters for 'All Items', 'Cloud Provider', 'Developer Tools', 'Modernization & Migration', 'Networking', 'OpenShift Optional', 'Storage', and 'Other'. Under 'Source', there are checkboxes for 'do316 Operator Catalog' (Cs 2) and 'Red Hat' (2). The search bar at the top has 'virtualization' typed into it. Below the search bar, there are two search results. The first result is 'Migration Toolkit for Virtualization Operator' provided by Red Hat, which facilitates migration of VM workloads to OpenShift Virtualization. The second result is 'OpenShift Virtualization' provided by Red Hat, which creates and maintains an OpenShift Virtualization Deployment. The 'OpenShift Virtualization' result is highlighted with a red box.

- 4. From the Operator page, click **Install** to open the **Install Operator** page.

Review the available installation options. For this exercise, keep the default options:

- Update channel is set to **stable**.
- Version is set to **4.16.2**.
- Installation mode is set to **A specific namespace on the cluster**.
- Installed Namespace is set to **Operator recommended Namespace: openshift-cnv**.
- Update approval is set to **Automatic**.

Click **Install** to start the operator installation. The installation might take several minutes to complete.

The screenshot shows the 'OpenShift Virtualization' operator page. At the top, it displays the operator name 'kubevirt-hyperconverged-operator.v4.16.2 provided by Red Hat'. Below this, the 'Installing Operator' section indicates 'AllRequirementsMet: all requirements found, attempting install'. It states that the operator is being installed and may take a few minutes. Once installed, required custom resources will be available for creation. A 'HyperConverged' operator card is shown, marked as 'Required' and described as creating and maintaining an OpenShift Virtualization Deployment. At the bottom, there are two buttons: 'Create HyperConverged' (grayed out) and 'View installed Operators in Namespace openshift-cnv' (blue).

- 5. Create a HyperConverged instance.

- 5.1. After the OpenShift Virtualization operator installation completes, you are prompted to create a HyperConverged instance. Click **Create HyperConverged** to configure the options for the HyperConverged instance.

The screenshot shows the OpenShift Virtualization Operator page. At the top, there's a header with the operator name and a green checkmark icon. Below the header, a section titled "Installed operator: custom resource required" contains a message about creating a custom resource. A specific operator card for "HyperConverged" is shown, indicating it is required and creates and maintains an OpenShift Virtualization Deployment. A red box highlights the "Create HyperConverged" button. To the right of the button is a link to view installed operators in the "openshift-cnv" namespace.

**Note**  
You can also create a HyperConverged instance in the OpenShift web console by navigating to Operators > Installed Operators. Select the OpenShift Virtualization operator and then click the OpenShift Virtualization Deployment tab. Click Create HyperConverged.

- 5.2. Locate and expand the **featureGates** section and clear the `enableCommonBootImageImport` option.

## featureGates

featureGates is a map of feature gate flags. Setting a flag to `true` will enable the feature. Setting `false` or removing the feature gate, disables the feature.

### deployTektonTaskResources

#### deployTektonTaskResources

deploy resources (kubevirt tekton tasks and example pipelines) in SSP operator

Deprecated: This field is ignored.

### enableCommonBootImageImport

#### enableCommonBootImageImport

Opt-in to automatic delivery/updates of the common data import cron templates.

There are two sources for the data import cron templates: hard coded list of common templates, and custom templates that can be added to the dataImportCronTemplates field. This feature gates only control the common templates. It is possible to use custom templates by adding them to the dataImportCronTemplates field.



### Note

In a later step, you update the HyperConverged configuration to import the common boot images from a local registry.

- 5.3. Scroll to the bottom of the page and click **Create** in the lower left of the window.
- 5.4. Wait for the operation to finish, and then refresh the browser. The kubevirt-hyperconverged instance displays a status of **Conditions: Reconcile, Complete, Available, Upgradeable** when the operation completes.

The screenshot shows the Red Hat OpenShift web console interface. In the top right, there's a notification about a web console update. Below it, the 'Installed Operators' section shows 'OpenShift Virtualization' is installed. The main area displays a table of 'HyperConverged' entries. One entry, 'HC kubevirt-hyperconverged', has its status conditions highlighted with a red box, showing 'Reconcile Complete', 'Available', and 'Upgradeable'. A 'Create HyperConverged' button is visible at the top right of the list table.

**Note**

If the notification to refresh the browser closed, then either refresh the browser or log out and then log in again to ensure that the **Virtualization** entry appears in the side panel.

- ▶ 6. Update the HyperConverged configuration to import common boot images from a local registry.

- 6.1. Use the command line to log in to your Red Hat OpenShift cluster as the **admin** user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 6.2. Run the following command to update the configuration. You can ignore any missing annotation warnings.

```
[student@workstation ~]$ oc apply -f \
~/D0316/labs/virtualization-deploy/disconnected-hco.yaml
...output omitted...
hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged configured
```

- ▶ 7. From the OpenShift web console left panel, go to **Operators > Installed Operators** to confirm that the OpenShift Virtualization installation succeeded.

Project: All Projects ▾

## Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the [Understanding Operators documentation](#). Or create an Operator and ClusterServiceVersion using the [Operator SDK](#).

Name	Namespace	Managed Namespaces	Status	Provided APIs
 <b>OpenShift Virtualization</b>	NS openshift-cnv	NS openshift-cnv	<span style="color: green;">Succeeded</span> Up to date	OpenShift Virtualization Deployment HostPathProvisioner Deployment

► 8. Verify the configuration from the command line.

- 8.1. Use the `oc describe` command to retrieve information about the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class and verify that it shows the `is-default-virt-class` annotation.

```
[student@workstation ~]$ oc describe storageclass \
  ocs-external-storagecluster-ceph-rbd-virtualization
Name:           ocs-external-storagecluster-ceph-rbd-virtualization
IsDefaultClass: No
Annotations:    description=Provides RWO and RWX Block volumes suitable for
                 Virtual Machine disks,storageclass.kubevirt.io/is-default-virt-class=true
...output omitted...
```

- 8.2. List the `operatorgroup`, `subscription`, and `hyperconverged` resources from the `openshift-cnv` project.

```
[student@workstation ~]$ oc get operatorgroup,subscription,hyperconverged \
  -n openshift-cnv
NAME                                     AGE
operatorgroup.coreos.com/openshift-cnv-86hfg   115m

NAME                               PACKAGE
  SOURCE          CHANNEL
subscription.coreos.com/kubevirt-hyperconverged   kubevirt-
hyperconverged      do316-catalog-cs    stable

NAME                                     AGE
hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged  37m
```

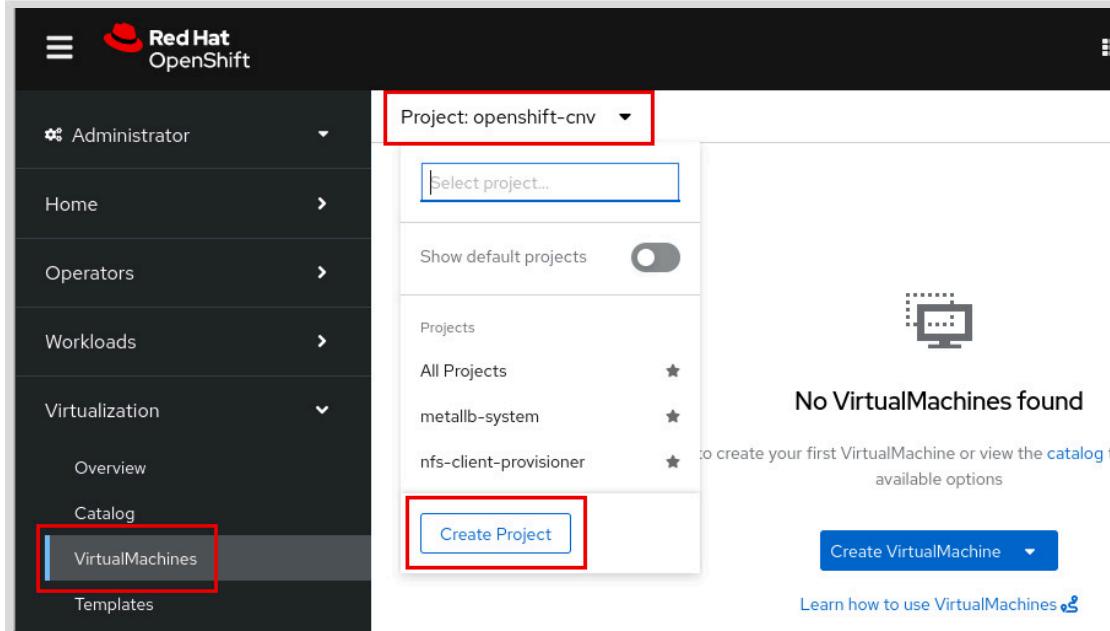
- 8.3. List the daemon sets in the `openshift-cnv` project.

```
[student@workstation ~]$ oc get daemonsets -n openshift-cnv
NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   ...
bridge-marker              5         5         5       5           5           ...
kube-cni-linux-bridge-plugin 5         5         5       5           5           ...
virt-handler               5         5         5       5           5           ...
```

- 9. Create a project and a simple VM to confirm that OpenShift Virtualization is functioning.

9.1. From the OpenShift web console left panel, go to **Virtualization > VirtualMachines**.

9.2. From the **Project** menu, click **Create Project**.



9.3. Name the project **test-vms**, and then click **Create**.

9.4. Click **Create virtual machine** and select the **From Instance Type** option.

9.5. In the first section, search for **rhel** images, and then click **rhel9** to select the boot volume.

**1 Select volume to boot from**

Volumes project  
PR All project ▾ Fil... / rhel ▾ Show

Name rhel Clear all filters

Volume name	Operating s...	Storage class	Size	Description
rhel8	Red Hat Enterprise Linux 8	ocs-external-storagecluster-rbdplugin-snapclass	30.00 GiB	-
<b>rhel9</b>	Red Hat Enterprise Linux 9	ocs-external-storagecluster-rbdplugin-snapclass	30.00 GiB	-

- 9.6. Scroll down to the **Select InstanceType** section and select the **U series > small** instance type.

**2 Select InstanceType**

Red Hat provided User provided

n1	cxl	u1	gn1	m1
N series ▾	CX series ▾	U series ▾	GN series ▾	M series ▾

nano: 1 CPUs, 512 MiB Memory  
micro: 1 CPUs, 1 GiB Memory  
**small: 1 CPUs, 2 GiB Memory** ▾  
medium: 1 CPUs, 4 GiB Memory  
large: 2 CPUs, 8 GiB Memory  
xlarge: 4 CPUs, 16 GiB Memory  
2xlarge: 8 CPUs, 32 GiB Memory  
4xlarge: 16 CPUs, 64 GiB Memory  
8xlarge: 32 CPUs, 128 GiB Memory

- 9.7. In the **VirtualMachine Details** section, change the **Name** field to **rhel9-test** and click **Create VirtualMachine**.

**3 VirtualMachine details**

Name	rhel9-test	Project	test-vms
Operating system	Red Hat Enterprise Linux 9	Boot disk size	30.00 GiB
InstanceType	u1.small	Storage class	ocs-external-storage...
CPU   Memory	1CPU   2 GiB Memory	Public SSH key	Not configured
		Dynamic SSH key injection	Off

Start this VirtualMachine after creation

**Create VirtualMachine** Customize VirtualMachine Cancel View YAML & CLI

- 10. From the resulting details page for the VM, monitor the deployment of the VM until the status is Running.

Project: test-vms ▾

VirtualMachines > VirtualMachine details

**VM rhel9-test** Running Actions ▾

**Overview** Metrics YAML Configuration Events Console Snapshots Diagnostics

<b>Details</b>	Alerts (0)
<b>Name</b> rhel9-test	<b>VNC console</b>
<b>Status</b> <span style="border: 1px solid red; padding: 2px;">Running</span>	
<b>Created</b> Oct 11, 2024, 2:12 PM (0 minutes ago)	
<b>Operating system</b> Red Hat Enterprise Linux 9.4 (Plow)	<span style="background-color: black; color: black;">[REDACTED]</span>
<b>CPU   Memory</b> 1CPU   2 GiB Memory	

**General**

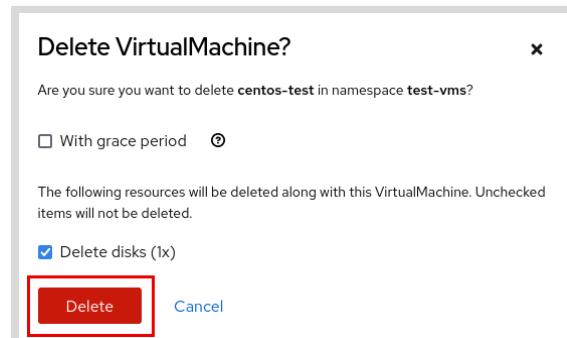
- Namespace (NS) test-vms
- Node (N) worker02
- VirtualMachineInstance (VMI) rhel9-t...
- Pod (P) virt-launch...
- Owner No owner

- 11. Remove the deployed test VM.

11.1. Go to Actions > Delete.

The screenshot shows the 'VirtualMachines' details page for a VM named 'rhel9-test'. The 'Actions' dropdown menu is open, and the 'Delete' option under the 'VirtualMachine' section is highlighted with a red box.

11.2. Click **Delete** to confirm the action.



## Finish

As the student user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish virtualization-deploy
```

## ► Quiz

# Red Hat OpenShift Virtualization

Match the following items to their counterparts in the table.

A catalog of certified enterprise operators for installation on OpenShift clusters.

A host-level DaemonSet resource that monitors changes to a VM object.

Handles cluster-wide virtualization functions and manages the lifecycle of pods that are associated with VMIs.

Includes resources that deploy and manage OpenShift Virtualization.

Technology that is used as the hypervisor in OpenShift Virtualization.

The container in a VMI-associated pod that starts and monitors the VMI.

Used to isolate groups of resources and to provide foundations for resource access control.

Component	Purpose
KVM	
Red Hat Marketplace	
HCO	
Namespaces	
virt-handler	
virt-launcher	
virt-controller	

## ► Solution

# Red Hat OpenShift Virtualization

Match the following items to their counterparts in the table.

Component	Purpose
KVM	Technology that is used as the hypervisor in OpenShift Virtualization.
Red Hat Marketplace	A catalog of certified enterprise operators for installation on OpenShift clusters.
HCO	Includes resources that deploy and manage OpenShift Virtualization.
Namespaces	Used to isolate groups of resources and to provide foundations for resource access control.
virt-handler	A host-level DaemonSet resource that monitors changes to a VM object.
virt-launcher	The container in a VMI-associated pod that starts and monitors the VMI.
virt-controller	Handles cluster-wide virtualization functions and manages the lifecycle of pods that are associated with VMIs.

# Summary

---

- Kubernetes uses declarative management of cluster resources to simplify the deployment, management, and scaling of containerized applications.
- Red Hat OpenShift is a set of modular components that are built on top of a Kubernetes container infrastructure, which provides a production platform through remote management.
- Red Hat OpenShift offers the Red Hat OpenShift Virtualization operator through the Red Hat Marketplace.
- OpenShift Virtualization uses Kernel-based Virtual Machine (KVM) technology as the hypervisor and the libvirt libraries for defining the virtual machines.
- Virtual machines that are created outside the OpenShift cluster can be imported and then managed with the OpenShift Virtualization operator.
- OpenShift Virtualization provides custom resource definitions and controllers to extend Kubernetes functions.
- IBM Cloud and Amazon Web Services have cloud offerings for Red Hat OpenShift clusters on bare metal instances that are suitable for installing and using Red Hat OpenShift Virtualization.
- A Red Hat OpenShift cluster must meet specific minimum requirements to support different features of OpenShift Virtualization:
  - The cluster must be installed on premise, or an appropriate cloud platform, on bare metal with Red Hat Enterprise Linux CoreOS workers.
  - All CPUs must be supported by Red Hat Enterprise Linux 9; must support Intel 64 or AMD64 CPU extensions; must have Intel VT or AMD-V hardware virtualization enabled; and must have the no-execute (NX) flag enabled.
  - You must create a default OpenShift Virtualization or Red Hat OpenShift storage class.
- The OpenShift Virtualization operator requires the HyperConverged Operator to be installed to create virtual machines.



## Chapter 2

# Running and Accessing Virtual Machines

### Goal

Create, manage, inspect, and monitor virtual machines in Red Hat OpenShift Virtualization.

### Sections

- Virtual Machine Resources (and Quiz)
- Creating and Accessing Virtual Machines by Using the Web Console (and Guided Exercise)
- Creating and Accessing Virtual Machines from the Command Line (and Guided Exercise)
- Monitoring and Inspecting Virtual Machines (and Guided Exercise)

### Lab

- Create, Configure, and Monitor Virtual Machines

# Virtual Machine Resources

## Objectives

- Describe and compare components of virtual machine instances.

## Workload Controllers

Red Hat OpenShift offers a set of resources to help you run applications inside your cluster. The pods execute containers within an OpenShift cluster.

The pod resource can execute one or more containers. These resources consist of your running application instances. You can use pod management tools to interact with containers.

In Kubernetes, workload resources manage applications to reduce the need to interact with your application's pod directly.

The most common workload resources are explained next.

## Deployments

A *deployment* object is a resource that uses pod templates to describe the intended state of your application and its components. Deployments use this template to create a running instance of your application. A deployment interacts with a *replica set* resource to ensure that your application has the intended number of pods.

## Replica Sets

A *replica set* resource, whether used independently or with a deployment, ensures that the specified quantity of pod replicas are available in a healthy state. Replica sets use set-based selectors to identify pods in the cluster. Replica sets create, update, or delete pods automatically to maintain the specified state from the deployment.

## Daemon Sets

A *daemon set* resource ensures that a pod runs on every selected node. Unlike a replica set resource, a daemon set resource is not bound to a specific number of replicas.

The daemon set automatically creates a pod on the node when a new node, with a matching node selector, is added to the cluster. If a daemon set resource is deleted, then the cluster automatically cleans up the associated pods.

## Stateful Sets

A *stateful set* is a set of pods that execute stateful applications with consistent, unique identities (network and storage). The stateful set workload resource is useful to point to the same storage identity to maintain the application's persistent data.

## Difference Between Workload Controllers, VMs, and VMIs

In general, a VM is a virtual environment that simulates the resources of a physical machine, which includes CPU, memory, network interfaces, and storage. Typically, VMs maintain the application state and persistent data, similar to `StatefulSet` pods.

In Red Hat OpenShift Virtualization, a VM object specifies a template to create a running instance of the VM inside your cluster. The running instance of a VM is a VMI. A container inside a pod executes and manages the VMI.

The relationship between a VM and a VMI is similar to the relationship between a deployment and a replica set. A VM object specifies the template for a VMI pod. If a VMI is deleted, then Kubernetes generates another instance that is based on the VM object configuration.

When a VM is created and started, a `virt-controller` pod signals to a `virt-handler` pod on a cluster node to create a `virt-launcher` pod for the VMI. The `virt-launcher` pod consists of a running `libvirtd` container instance to execute the VMI as an isolated process.

## Components of an OpenShift Virtual Machine

Because VMIs exist in pods, they have components to execute and maintain the health of the workload. These components include console access, network interfaces, `virt-launcher` pods, persistent volumes, and persistent volume claims.

### Virtual Machine Consoles

Red Hat OpenShift Virtualization provides three consoles to access and manage VMIs:

#### VNC Console

The *Virtual Network Computing* (VNC) console connects you to the VMI console through VNC. The VNC console provides both text-based and graphical consoles, depending on the configuration of the VMI's operating system. VNC is the default selection when you access the `Console` tab of a VMI through the OpenShift web console.

#### Serial Console

The *serial console* connects you to the VMI's text-based console through its serial port.

#### RDP Console

The *Remote Desktop Protocol* (RDP) provides a graphical tool through the RDP console for Windows-based VMIs. The *Quick Emulator* (QEMU) guest agent must be installed and running on the VMI to use the RDP console.

When using the OpenShift web console, you can switch between the various console types.



#### Note

The RDP console is not shown in this course. For more information, refer to the *Connecting to the Desktop Viewer* section at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-connecting-to-vm-console-web\\_desktop-viewer](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-connecting-to-vm-console-web_desktop-viewer)

## Network Interfaces

When you attach a network interface to a VM, you have several options, such as connecting to the pod network or to external networks.

## Default Pod Network

The *default pod network* assigns an IP address to the VMI from the cluster's pod network. To use the default pod network, the network interface must use the masquerade binding method. A masquerade binding uses NAT to enable other pods in the cluster to communicate with the VMI. The VMI is accessible only within the cluster by default.

## Multus

You can connect a VM to multiple interfaces and to external networks with the *Container Network Interface* (CNI) plug-in, Multus. Multus provides a *Custom Resource Definition* (CRD), which is called a *Network Attachment Definition* (NAD), to enable you to connect pods, VMs, or VMIs to one or more networks.

## Single Root I/O Virtualization

A VMI can also use a *Single Root I/O Virtualization* (SR-IOV) device that enables a VMI to connect to a virtual function network for high performance.



### Note

VM networking is explained more fully later in the course.

## Persistent Volumes and Persistent Volume Claims

A *Persistent Volume* (PV) is a storage resource that the persistent volume API object provisions. A PV represents existing storage within the cluster. You can use a *Persistent Volume Claim* (PVC) to attach a PV to a VMI, which represents a request for a specific storage resource.



### Note

Persistent volumes and other storage management concepts are explained later in the course.

## Boot Sources

The cluster uses a boot source to instantiate a VM. A boot source is a fully configured disk image that includes the operating system (OS), drivers, and other initial data for a VM. A boot source can be a cloud image in either RAW or QCOW2 format, an installation disk image in ISO format, a container disk from a container image registry, or a PVC. You can upload a boot source from your local system, use an existing volume, use an existing snapshot, or download the volume from a registry. The download from a registry creates a `dataimportcron` resource that defines a cron job to pull and import the disk image.

You can also use base operating system images from container registries, such as Quay.io. To view, add, or manage available boot sources from the web console, go to **Virtualization > Bootable Volumes**. For some operating systems, Red Hat provides a boot source.

A *golden image* is a preconfigured snapshot of a VM that you can use as a boot source to create a VM. A golden image is created by installing and configuring an OS and applications on a VM. After a golden image is created, it is saved as a template or as an image file. Boot sources from Red Hat are golden images that are published as container disks in the registry for versions of Red Hat Enterprise Linux (RHEL). Red Hat golden images are automatically updated to the latest version of the OS and are synchronized to connected OpenShift clusters after installing OpenShift Virtualization. The *Containerized Data Importer* (CDI) imports the golden images from Red Hat to the `openshift-virtualization-os-images` project as snapshots or PVCs. You can use the golden images to create VMs. You can edit labels and annotations, or delete the configured volume, but you cannot manage other properties.

**Note**

Importing default images is based on the `enableCommonBootImageImport` option of the HCO operator. This option is enabled by default during installation, and can be changed with the manifest file.

## Configuration Maps and Secrets

A *configuration map* is a resource type that contains a set of keys and values that other resources can use. Configuration maps and secrets centralize configuration values that several resources use. The values of secrets are always encoded (not encrypted), and their access is restricted to authorized users.

To add a configuration map and a secret to a VM, you must create a disk to contain the configuration map or the secret. The configuration map and the secret must be created before you attach them to the VM. After adding the disk with the configuration map or secret, you must start the VM and mount the disk manually from the VM console.

## Components of Virtual Machine Creation

For the Red Hat OpenShift components that provide virtualization, the cluster structures components into a hierarchy, where higher components (such as `virt-controller`) orchestrate lower components (such as the `virt-handler` daemon set and the VM's pod).

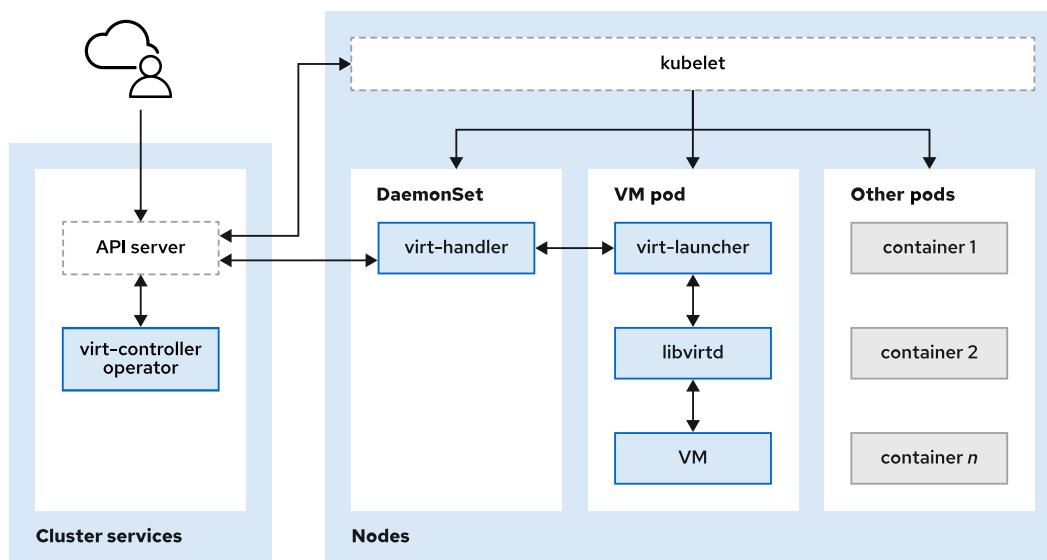


Figure 2.1: Components of virtual machine creation

The `virt-controller` operator monitors for new VMI objects. The `virt-handler` daemon set runs on each node to execute any necessary actions to meet a VM object's defined state. The `virt-handler` daemon set monitors for changes in a VM, and creates the `virt-launcher` container.

The `virt-launcher` container runs within each VM's pod and instantiates the VM with the use of a local `libvirt` instance, which provides a low-level virtualization architecture and interfaces with the kernel to manage the lifecycle of the VM process. When the VMI is provisioned, the `virt-launcher` pod routes IPv4 traffic to the Dynamic Host Configuration Protocol

## Chapter 2 | Running and Accessing Virtual Machines

(DHCP) address of the VMI. This routing enables you to connect to a VMI with a port-forwarding connection.

The `libvirtd` instance also includes the `virsh` command, to manage VMs. This utility provides several commands to list, start, stop, or reboot VMs:

1. `virsh list`: Lists all running VMs.
2. `virsh start`: Starts a VM.
3. `virsh shutdown`: Shuts down a VM.
4. `virsh dumpxml`: Dumps the XML configuration of a VM to STDOUT. You can use it to update an existing VM or use it as a template to create another VM.
5. `virsh reset`: Resets a VM without shutting it down previously.
6. `virsh restore`: Restores a VM from a saved file.
7. `virsh migrate`: Migrates a VM to another host.

To manage the VMI with `virsh` commands, access the `virt-launcher` pod of the VM. You can access the `virt-launcher` terminal from the web console. Go to **Workloads > Pods**, select the `virt-launcher` pod for the VMI, and then go to **Terminal**.

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
<code>virt-launcher-mariadb-client-hx9fh</code>	Running	1/1	0	VMI mariadb-client	759.7 MiB	0.140 cores	Sep 3, 2024, 10:34 PM
<code>virt-launcher-mariadb-server-qld4</code>	Running	1/1	0	VMI mariadb-server	891.0 MiB	0.008 cores	Sep 3, 2024, 10:32 PM

Figure 2.2: Accessing the `virt-launcher` pod

To list all VMs inside the `virt-launcher` pod, use the following command:

```
sh-5.1$ virsh list
Authorization not available. Check if polkit service is running or see debug
message for more information.
Id      Name                           State
-----
1      network-services_mariadb-client  running ❶
```

- ❶ The name is composed of the namespace and the name of the VM.

To shut down a VM, use the following command:

```
sh-5.1$ virsh shutdown network-services_mariadb-client
Authorization not available. Check if polkit service is running or see debug
message for more information.
Domain network-services_mariadb-client is being shutdown
```

To list all events from the `network-services_mariadb-client` VM, use the following command:

```
sh-5.1$ virsh event network-services_mariadb-client --all
Authorization not available. Check if polkit service is running or see debug
message for more information.
event graphics for domain network-services_mariadb-client: connect local[unix /
var/run/kubevirt-private/eef4df71-793c-474f.../virt/vnc] remote[unix ]
none
events received: 1
```

**Note**

For more information about the `virsh` CLI tool, refer to the `virsh` documentation at <https://libvirt.org/manpages/virsh.html>

The `virtctl` CLI tool provides several commands to manage VMs and VMIs. With the `virtctl` CLI tool, you can start, stop, pause, restart, or migrate a VM. You can use the `virtctl` CLI tool to connect to the serial console of a VMI, or you can create a service that forwards a designated port of a VM.

To stop a VM, use the following command:

```
[user@host ~]$ virtctl stop vm-name
VM vm-name was scheduled to stop
```

**Note**

For more information about the `virtctl` CLI tool, refer to the `virtctl` Commands documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#installing-virtctl\\_virt-using-the-cli-tools](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#installing-virtctl_virt-using-the-cli-tools)

## Stand-alone VMIs

A *stand-alone* VMI is created independently from the OpenShift Virtualization environment. You can use the CLI and an automation script to create a stand-alone VMI. You can use the web console or `oc` commands to manage stand-alone VMIs, similarly to how a VM resource manages a VMI.

To list all the VMIs in the cluster, including stand-alone VMIs, use the `oc get vmis` command:

```
[user@host ~]$ oc get vmis -A
NAMESPACE      NAME        AGE     PHASE      ...
network-services mariadb-vm-1  5m      Running    ...
network-database postresql-vm-1 59m      Running    ...
```

**Note**

The `oc` commands for VM and VMI management are explained later in the course.

## Use Cases for Creating VMs

In Red Hat OpenShift Virtualization, you can create a VM in many ways: for example, from predefined templates and instance types from Red Hat, from custom templates, from declarative manifest files, by cloning an existing VM object, or from migration operators and imports.

The most common ways to create VMs are explained next.

### Templates

You can use predefined templates to create VMs with default parameters that are provided in the configuration of the available boot source from Red Hat. You can change the parameters to customize the VM according to the needs of the applications that are configured inside the VM, such as the boot source, compute resources, or SSH keys. You can also create templates that define these custom settings and boot sources. After you create a VM from a template, you can override any parameters that are defined in the template for the VM.

### Instance types

An *instance type* is a reusable object where you can define resources and characteristics to apply to new VMs. You can define custom instance types or use the types that are included by default in OpenShift Virtualization. Instance types must define CPU and memory resources. All other attributes are optional.

When you create a VM from a predefined instance type, you cannot override any parameters that are defined in the instance type. Because an instance type defines the required CPU and memory resources, OpenShift Virtualization always rejects additional requests for these resources when creating a VM from an instance type. Predefined instance types are known as `common-instancetype`s.

### Declarative files

You can create VMs from manifest files that declare the configuration of needed parameters to customize and create the VM. Declarative files are useful for automation, such as for managing the GitOps workflow. VM manifests must be in YAML format. You can use an instance inside the VM manifest to simplify the configuration. After creating a VM manifest, you can use the `oc create -f <filename>` command to create a VM from the file.

You can use the `virtctl` command to create VM base manifests where some parameters are configured, such as the name, the run strategy, or the memory. The command dumps the configuration in YAML format of a VM to `STDOUT`, so you can copy and paste it to a YAML file.

```
[user@host ~]$ virtctl create vm --name mariadb-prod
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...output omitted...
spec:
  domain:
    devices: {}
    memory:
      guest: 512Mi
    resources: {}
  terminationGracePeriodSeconds: 180
status: {}
```



## References

For more information, refer to the *OpenShift Container Platform Virtualization* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index)

## ► Quiz

# Virtual Machine Resources

Choose the correct answers to the following questions:

- ▶ 1. Which console type is the default when accessing the Console tab of a VMI through the OpenShift web console?
  - a. Serial
  - b. SSH
  - c. RDP
  - d. VNC
  
- ▶ 2. You are configuring a virtual machine in Red Hat OpenShift Virtualization and need to ensure that the VM has persistent storage for its data. Which resource must you request to attach storage to the VM?
  - a. Daemon Set
  - b. Persistent Volume
  - c. Network Attachment Definition
  - d. Persistent Volume Claim
  
- ▶ 3. You are setting up a network configuration for a VM in a Red Hat OpenShift cluster that requires high-performance networking. Which technology must you use to achieve high performance?
  - a. Multus
  - b. Single Root I/O Virtualization (SR-IOV)
  - c. Default pod network
  - d. Masquerade binding
  
- ▶ 4. True or False: Red Hat golden images are automatically updated to the latest version of the operating system and are synchronized to connected OpenShift clusters after OpenShift Virtualization is installed.
  - a. True
  - b. False
  
- ▶ 5. What is the role of the `virt-controller` operator?
  - a. Instantiates the VM with the use of a local `libvirtd` instance.
  - b. Monitors for new VMI objects.
  - c. Monitors for changes in a VM and creates the `virt-launcher` container.

► **6. Which command lists all stand-alone VMIs in the cluster?**

- a. oc get pods
- b. virsh list
- c. oc get vmis -A
- d. virtctl fslist

## ► Solution

# Virtual Machine Resources

Choose the correct answers to the following questions:

- ▶ 1. Which console type is the default when accessing the Console tab of a VMI through the OpenShift web console?
  - a. Serial
  - b. SSH
  - c. RDP
  - d. VNC
  
- ▶ 2. You are configuring a virtual machine in Red Hat OpenShift Virtualization and need to ensure that the VM has persistent storage for its data. Which resource must you request to attach storage to the VM?
  - a. Daemon Set
  - b. Persistent Volume
  - c. Network Attachment Definition
  - d. Persistent Volume Claim
  
- ▶ 3. You are setting up a network configuration for a VM in a Red Hat OpenShift cluster that requires high-performance networking. Which technology must you use to achieve high performance?
  - a. Multus
  - b. Single Root I/O Virtualization (SR-IOV)
  - c. Default pod network
  - d. Masquerade binding
  
- ▶ 4. True or False: Red Hat golden images are automatically updated to the latest version of the operating system and are synchronized to connected OpenShift clusters after OpenShift Virtualization is installed.
  - a. True
  - b. False
  
- ▶ 5. What is the role of the virt-controller operator?
  - a. Instantiates the VM with the use of a local libvirtd instance.
  - b. Monitors for new VMI objects.
  - c. Monitors for changes in a VM and creates the virt-launcher container.

► **6. Which command lists all stand-alone VMIs in the cluster?**

- a. `oc get pods`
- b. `virsh list`
- c. `oc get vmis -A`
- d. `virtctl fslist`

# Creating and Accessing Virtual Machines by Using the Web Console

## Objectives

- Create and access a new virtual machine by using the Red Hat OpenShift web console.

## Virtual Machine Creation

You can create a VM from a template by using the Virtual Machine Creation Catalog in the Red Hat OpenShift web console or with YAML definitions through the web console or from the CLI.

### Using the Catalog to Create a VM

The fastest way to create VMs is by using the templates and instance types from the catalog in the OpenShift web console. OpenShift provides default templates, including Red Hat Enterprise Linux, CentOS, Fedora, and Microsoft Windows Server. When using a default template, you must define a boot source if the template has the `Boot source required` label. When you use a template from the catalog to create a VM, you can customize the VM parameters such as data sources, cloud-init, or SSH keys, and you can add disks.



#### Note

Some templates are incompatible with single-node OpenShift clusters.

#### Project

The namespace where the VM resides.

#### Virtual Machine Name

The name of the VM to use in your Red Hat OpenShift cluster.

#### Authorized Key and SSH Access

Determines whether the VM is configured for SSH access, and which public SSH keys are copied to `~/ .ssh/authorized_keys` on the VM. Enabling SSH access creates a node port service within the VM's namespace. Enabling SSH access from outside the cluster requires additional configuration.

#### Start this VM after creation

Determines whether to power on the VM after provisioning, or leave it powered off.

You can also create a VM from an instance type by using the web console. You can customize VM parameters such as the SSH keys and cloud-init or use a new storage volume. However, you cannot customize the bootable volume or the resources that are defined in the instance type.

## Virtual Machine Templates

Red Hat provides the default templates. These templates include settings to create generic systems with preconfigured networking, users, and storage. Default templates create a basic installation with settings that suit various needs. You can modify these templates to create templates with more specific settings. A modified template can include a specific boot source, additional storage, additional network interfaces, and other configuration differences.

## Chapter 2 | Running and Accessing Virtual Machines

You can create templates from the web console or from the CLI. New templates start with no preconfigured content. You can add your configuration content to create more specialized templates.

Initially, you can create templates only for the existing default operating systems. However, if you install the KubeVirt common-templates package in your OpenShift cluster, then you can create templates for other operating systems, such as Ubuntu or openSUSE.

## YAML Definition

As an alternative to using the OpenShift catalog for VM creation, you can use YAML definition. You can define YAML from either the web console or from the CLI with a text editor. Use YAML also to create VM templates. For both VMs and templates, the web console provides an editable default YAML configuration. The YAML file includes all the settings from a default template or an instance type in one structured document. You can adjust labels, modify system resources, define a base image, and change advanced options in one location by using the YAML editor.

You can also use the CLI to edit the YAML configuration by using the `oc edit` command. This command opens an object's YAML configuration in your default text editor. To apply your changes, save the YAML configuration. The cluster applies changes to the resource when the editor's buffer is written.

## Virtual Machine Management

Each VM has a management page that contains information about the VM, an interface to the VM's console, and controls for VM operations.

The screenshot shows the 'VirtualMachines' section of the OpenShift web console. A specific VM named 'mariadb-client' is selected, which is currently 'Running'. The 'Overview' tab is active, displaying basic details such as Name (mariadb-client), Status (Running), Created (Aug 27, 2024, 12:11 AM), Operating system (Red Hat Enterprise Linux 8.5 (Ootpa)), CPU/Memory (1CPU | 2GiB Memory), Time zone (EDT), Template (rhel8-server-small), and Hostname (mariadb-client). To the right of the details, there is a VNC console preview showing a black screen. On the far right, there are several action buttons: a square (Stop), a circle (Start), a double arrow (Restart), and a downward arrow (Actions). Below the main table, there are sections for Alerts (0), General, Namespace (network-services), Node (worker01), VirtualMachineInstance (mariadb-client), Pod (virt-launcher-mariadb-...), and Owner (No owner). At the bottom right, there is a 'Take snapshot' button.

From a VM's management page, click **Actions** for administrative operations such as starting and stopping a VM.

The following table lists the available elements on the VM's page:

Element	Description
Overview	Contains a log of cluster-level VM events, general details about the VM, resource utilization, and status information.

Element	Description
Overview > Details	Contains detailed information about the components of the VM.
Metrics	Displays CPU and memory utilization, storage metrics, network traffic metrics, and VM migration metrics.
YAML	Provides an interface to edit the VM's YAML configuration.
Configuration	Provides access to configuration subsections.
Configuration > Details	Lists the VM details. From this tab, you can edit the description of the VM, the workload profile, the CPU and memory, the hostname, the GPU and host devices, the boot mode, and the boot order of the disks.
Configuration > Storage	Lists the disks and file systems that a VM uses. From this tab, you can edit, add, or delete a VM's disks and observe details of a VM's file.
Configuration > Network	Contains the network interfaces that are connected to a VM. You can also use this tab to manage the connected network interfaces.
Configuration > Scheduling	Enables managing the node scheduling and resource requirements such as node selector rules, toleration rules, and affinity rules. Also use this section to manage the node migration strategy for the VM.
Configuration > SSH	Enables you to configure SSH access to the VM. From this tab, you can edit the SSH service type and the public SSH key.
Configuration > Initial run	Manages the cloud-init username and password. The cloud-init feature adds the user to the VM's sudoers file to grant sudo privileges to the user. Use this section to configure the Sysprep tool for Windows VMs, to automate setup and custom software provisioning.
Configuration > Metadata	From this tab, you can configure labels and annotations.
Events	Contains a detailed and customizable event log.
Console	Provides a serial and VNC console to access a VM directly.
Snapshots	Lists and manages a VM's snapshots.
Diagnostics	Displays the status condition of the VM and the volume snapshot status.

You can use the various VM pages to make and confirm changes to the VM. For example, to increase or decrease the CPU and memory of a VM, go to Configuration > Details to open the **VirtualMachine details** page. Click the defined values in the CPU | Memory section.

The screenshot shows the 'VirtualMachine details' page with the 'Configuration' tab selected. The 'CPU | Memory' section is highlighted with a red box, showing '2 CPU | 4 GiB Memory'. Other sections include 'Storage' (None), 'Network' (Workload profile: Server), 'Scheduling', and 'SSH'.

Figure 2.4: VirtualMachine details page

On the **Edit CPU | Memory** page, adjust the CPU and memory values as needed and then click **Save**.

The dialog box 'Edit CPU | Memory' shows two sliders: 'CPU sockets' set to 4 and 'Memory' set to 12 GiB. Below the sliders, it says 'Total CPU is 4'. At the bottom are 'Save', 'Restore template settings', and 'Cancel' buttons.

Figure 2.5: Edit CPU | Memory

You can confirm the new CPU and memory values from the **Overview > Details** section.

The screenshot shows the 'Overview' tab selected. In the 'Details' section, the 'CPU | Memory' row is highlighted with a red box, showing '4 CPU | 12 GiB Memory'. Other details include 'Name: rhel9-magenta-tapir-27', 'Status: Running', 'Created: Oct 6, 2024, 9:44 PM (5 hours ago)', 'Operating system: Red Hat Enterprise Linux 9.3 (Plow)', and 'Time zone: EDT'. A VNC console screenshot is also visible.

Figure 2.6: Overview page with the updated values

**Note**

Some changes to a running VM require that you either restart or live migrate the VM to apply the changes.

## Accessing Virtual Machines

You can access your VMs in several ways:

### Web Console virtual machine interface

To access the VNC and serial interface for a VM from the OpenShift web console, navigate to **Virtualization > VirtualMachines**, select the VM's project, click the VM name, and then click the **Console** tab.

- The VNC console enables video access to the VM and enables graphical operations.
- The serial console provides a direct serial connection and CLI access to a VM. When using the serial console, you might need to press **Enter** a few times for a prompt to appear. Click **Open web console** from the **Overview > Details** tab to open the VNC and the serial consoles in a new window.

### CLI virtual machine interface

You can also access the serial console or the VNC console of a VM through the **virtctl** CLI tool. The **virtctl** tool is a client application for managing and connecting to VMs that are hosted in OpenShift or KubeVirt environments.

To access OpenShift VMs with the **virtctl** tool, you must first log in to the OpenShift cluster with the **oc login** command. The **virtctl** tool refers to the Kubernetes configuration that the **oc** client creates. Select the VM's project with the **oc project *vm-project-name*** command.

- To access the serial console of a VM, use the following command. The escape characters for the serial console are **Ctrl-]**.

```
[user@host ~]$ virtctl console vm-name
Successfully connected to vm-name console. The escape sequence is ^]
...output omitted...
```

- To access the VNC or the graphical interface of a VM, use the following command:

```
[user@host ~]$ virtctl vnc vm-name
...output omitted...
```

### Other VNC clients

The **virtctl** tool calls the **remote-viewer** application. The **virtctl** tool requires a separate installation by default on Linux for VNC. You can use other VNC clients by passing the **--proxy-only** flag to the **virtctl vnc** command.

Use the port that is provided from your chosen VNC application to connect to your VM. Some other popular VNC applications are TigerVNC and TightVNC. For Windows VMs, you can use the desktop viewer and the Remote Desktop Protocol (RDP) by using the web console. An RDP client must be installed, and the QEMU guest agent must be installed on the Windows VM.

**Note**

You must disconnect the VNC console session to use the desktop viewer. Otherwise, the VNC console session continues to run in the background.

## Role-based Access Control

Openshift includes several default cluster roles that you can bind to users. The following table lists some default cluster roles:

<b>Cluster role</b>	<b>What users or groups with this role level can do</b>
admin	View any resource in a project, and modify any project resource except for quotas.
cluster-admin	Perform any action in any project.
edit	Modify most objects in a project, but cannot view or modify roles and bindings.
view	View most objects in a project, but cannot modify them. Users and groups cannot view or modify roles and bindings.

OpenShift Virtualization extends the default cluster roles for virtualization resources. You can use these roles to grant permissions to users for virtualization resources in your cluster.

<b>Cluster role</b>	<b>OpenShift Virtualization cluster role</b>	<b>What users and groups with this role level can do</b>
view	kubevirt.io:view	View all OpenShift Virtualization resources, but cannot create, delete, modify, or access the resources.
edit	kubevirt.io:edit	Modify all OpenShift Virtualization resources, but cannot modify the OpenShift Virtualization runtime configuration.
admin	kubevirt.io:admin	View and modify all OpenShift Virtualization resources, and can modify the OpenShift Virtualization runtime configuration.



## References

For more information about working with YAML templates, see the *Using Templates* guide in the Red Hat OpenShift Container Platform 4.16 documentation at [https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/images/index#using-templates](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/images/index#using-templates)

For more information about installing the `virtctl` client, see the *Installing virtctl* guide in the Red Hat OpenShift Container Platform 4.16 documentation at [https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/virtualization/index#installing-virtctl\\_virt-using-the-cli-tools](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/virtualization/index#installing-virtctl_virt-using-the-cli-tools)

For more information about using `virtctl`, see the *Using the virtctl and libguestfs CLI Tools* guide in the Red Hat OpenShift Container Platform 4.16 documentation at [https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-using-the-cli-tools](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/virtualization/index#virt-using-the-cli-tools)

For more information, refer to the *KubeVirt User Guide - Templates* documentation at [https://kubevirt.io/user-guide/virtual\\_machines/templates/#common-template-customization](https://kubevirt.io/user-guide/virtual_machines/templates/#common-template-customization)

## ► Guided Exercise

# Create and Access a Virtual Machine by Using the Web Console

Create a virtual machine by using a predefined template in the web console.

### Outcomes

- Create a VM.
- Access the VNC console from the web console.
- Manage the VM from the web console.
- Access the `virt-launcher` pod terminal.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start accessing-guicreate
```

This command ensures that all resources are available for this exercise.

### Instructions

- 1. As the `admin` user, locate and go to the Red Hat OpenShift Container Platform web console.

- 1.1. Use the command line to log in to your RHOCP cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Identify the URL for the RHOCP web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`.
- 1.4. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.

- 2. Create a RHEL 9 VM from the catalog.

- 2.1. From the OpenShift web console, locate the left panel. If you do not see the left panel, then click the main menu icon at the upper left of the web console.
- 2.2. Go to **Virtualization > Catalog**. From the **Project** search field, select the **accessing-gui** project from the list.
- 2.3. Go to **Template catalog**.

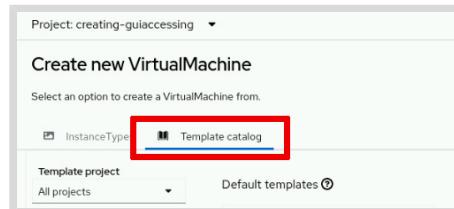


Figure 2.7: Create the VM from a template

- 2.4. Select the Red Hat Enterprise Linux 9 VM template.

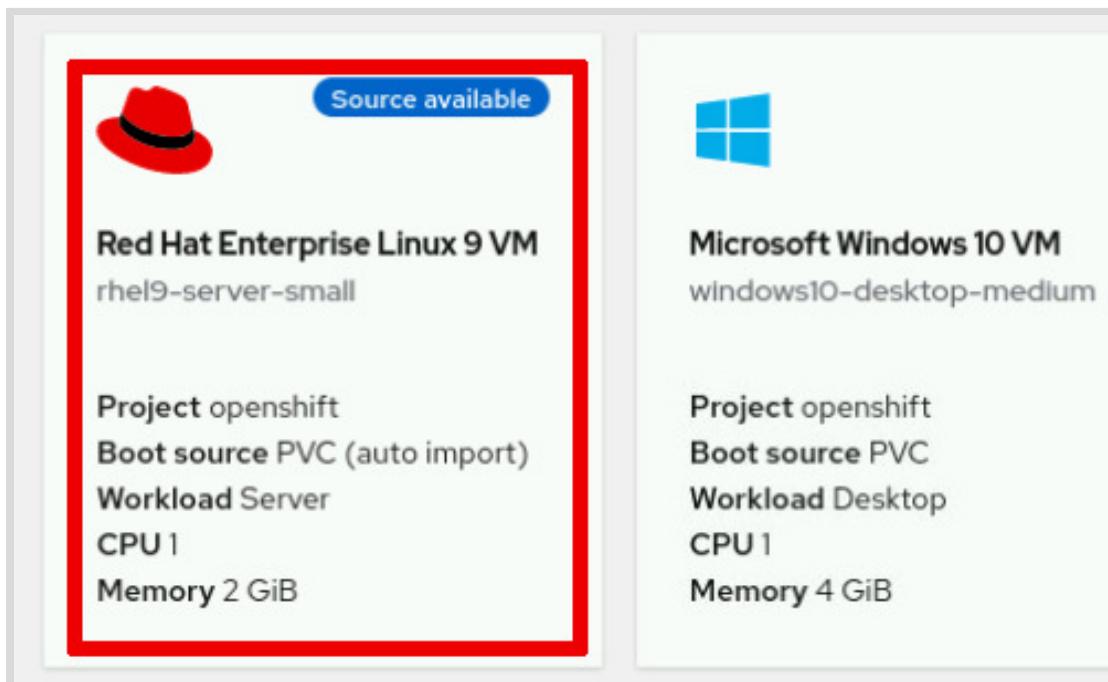


Figure 2.8: Select the Red Hat Enterprise Linux 9 template

- 2.5. From the **Optional parameters** section, enter **redhat123** for the **CLOUD\_USER\_PASSWORD** field.

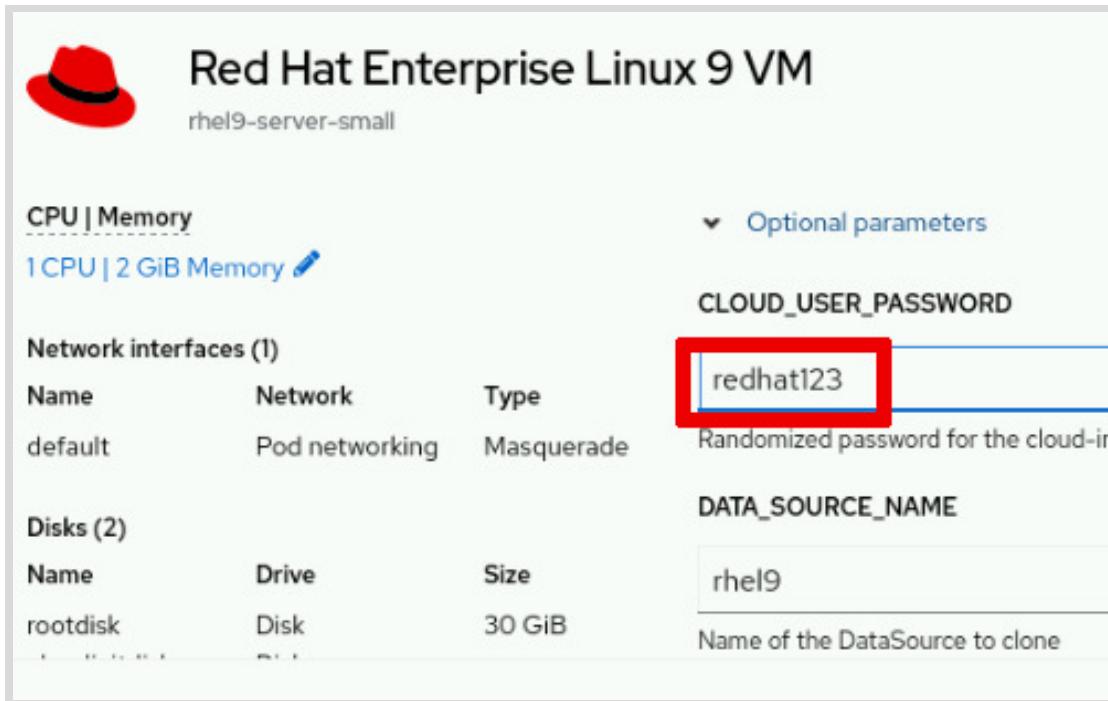


Figure 2.9: Add the password

- 2.6. Enter `accessing-vm` for the virtual machine name.
  - 2.7. Ensure that the `Start this VirtualMachine after creation` option is selected, and then click `Quick create VirtualMachine` to create the VM.
- 3. Review the details of your VM.
- 3.1. From the VM Overview tab, wait until the status changes to `Running` before continuing this exercise.

Figure 2.10: Review the VM

- 4. Access the VNC console from the web console and retrieve the status of the guest agent.
- 4.1. From the Overview tab, select the `Open web console` link under the VNC console preview to open the VM's web console in a new tab.

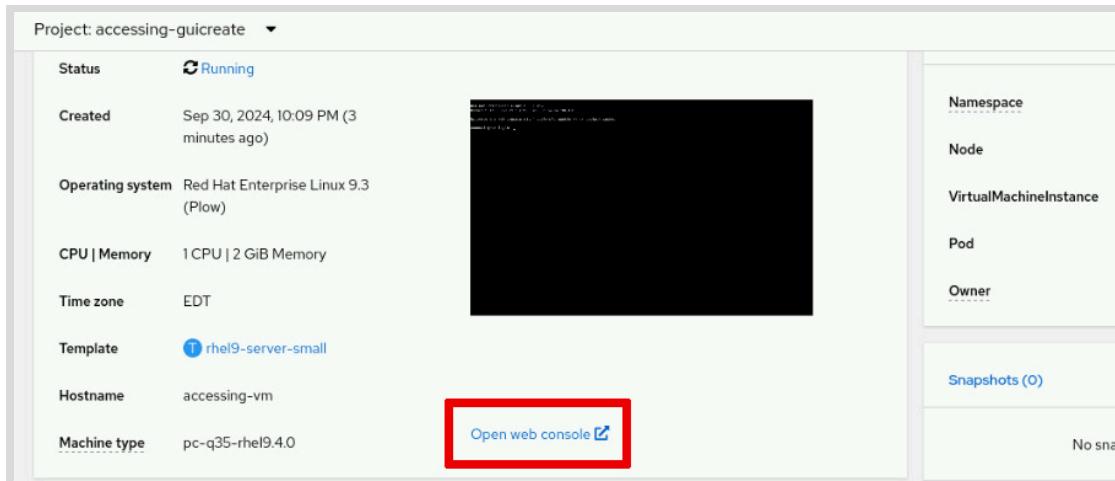


Figure 2.11: Open the VNC console

- 4.2. Use the `cloud-user` user with `redhat123` as the password to log in to the VM's VNC console. Click the **Guest login credentials** option to list the login credentials of the VM.

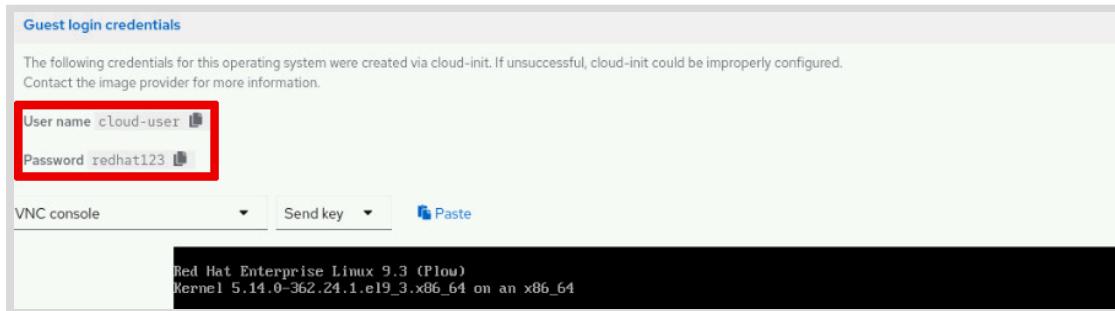


Figure 2.12: View the guest credentials

- 4.3. Retrieve the status of the QEMU agent.

```
[cloud-user@accessing-vm ~]$ systemctl status qemu-guest-agent
qemu-guest-agent.service - QEMU Guest Agent
   Loaded: loaded (/usr/lib/systemd/system/qemu-guest-agent.service; enabled;
   preset: enabled)
     Active: active (running) since Mon 2024-09-30 12:22:35 EDT; 29min ago
       ...output omitted...
```

- 4.4. Use the `ip addr` command to list the network interfaces.

```
[cloud-user@accessing-vm ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
  qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope lo
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UP group
  default qlen 1000
    link/ether 02:d0:25:00:00:00 brd ff:ff:ff:ff:ff:ff
    altname enp1s0
```

```
inet 10.0.2.2/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
    valid_lft 86311399sec preferred_lft 86311399sec
inet6 fe00::d0::25ff:fe00:0/64 scope link
    valid_lft forever preferred_lft forever
```

- 4.5. Use the `lsblk` command to list the available disks.

```
[cloud-user@accessing-vm ~]$ lsblk
NAME   ...   SIZE   ...   MOUNTPOINTS
vda   ...   30G   ...
|-vda1  ...   1M   ...
|-vda2  ...   200M  ...   /boot/efi
|-vda3  ...   600M  ...   /boot
|-vda4  ...   29.2G ...   /
vdb   ...   1M   ...
```

You can safely close the VNC console window.

- ▶ 5. Update the CPU cores of the VM to 2, and the memory to 5 GiB.

- 5.1. Go to the **Configuration** tab. On the **Details** tab, click the **CPU|Memory** values. Update the CPU to 2, and the memory to 5 GiB, and click **Save**. Wait until the **Status** changes to **Running** before continuing this exercise.



### Note

If the VM does not restart automatically, then click the restart icon to restart the VM.

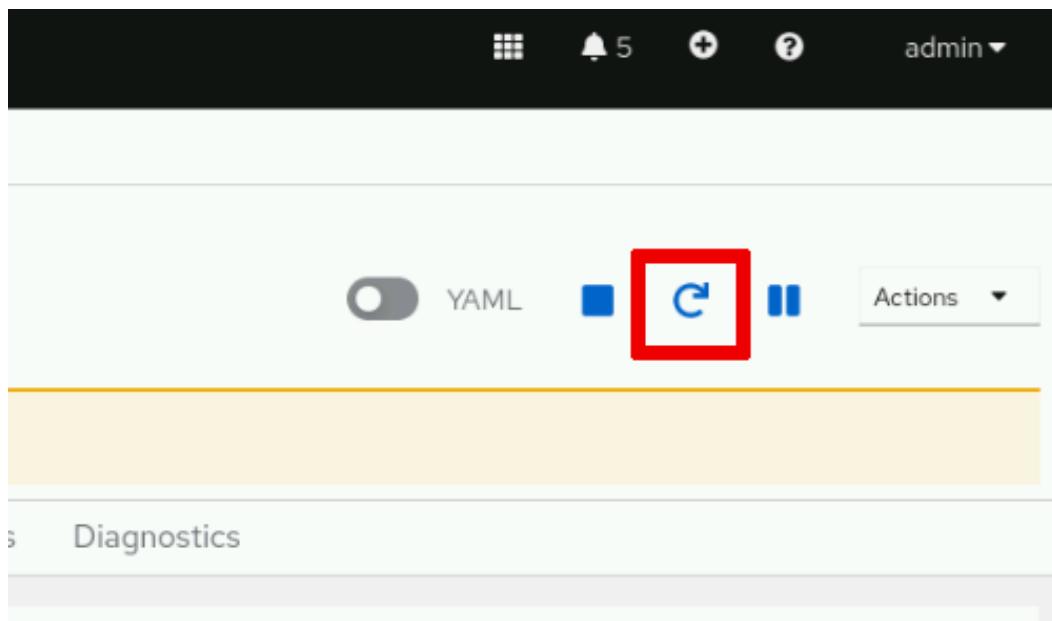


Figure 2.13: Restart the VM

- 5.2. Go to **Overview** and confirm that the CPU and memory are correct in the **Details** section.

Name	accessing-vm	VNC console
Status	 Running	
Created	Sep 30, 2024, 10:09 PM (9 minutes ago)	
Operating system	Guest agent is required	
CPU   Memory	2 CPU   5 GiB Memory	
Time zone	-	
Template	 rhel9-server-small	

Figure 2.14: Confirm the CPU and memory details

► 6. Manage the VM instance from the `virt-launcher` pod terminal.

- 6.1. From the `Overview` tab, locate the `General` section, and select the `virt-launcher` pod of the VM.

General	
Namespace	NS accessing-guicreate
Node	N master03
VirtualMachineInstance	VMI accessing-vm
Pod	P virt-launcher-accessing-...
Owner	No owner

Figure 2.15: Select the virt-launcher pod

- 6.2. Go to Terminal to open the terminal of the pod, and use the `virsh list` command to list all running VMs inside the `virt-launcher` pod.

```
sh-5.1$ virsh list
Id  Name                           State
-----
1   accessing-guicreate_accessing-vm  running
```

- 6.3. Use the `virsh shutdown` command to stop the running RHEL 9 VMI. The VM starts automatically after some time.

```
sh-5.1$ virsh shutdown accessing-guicreate_accessing-vm
Authorization not available. Check if polkit service is running or see debug
message for more information.
Domain 'accessing-guicreate_accessing-vm' is being shutdown
```



**Note**

After the VMI is deleted, Kubernetes creates another instance.

- 6.4. Go to Virtualization > VirtualMachines and select the `accessing-vm` VM. Go to Overview and confirm that the status is Running. A new `virt-launcher` pod is available in the General section.

- 6.5. Go to **Events** and review the entries to confirm that the **virtualmachine-controller** created an instance after deleting the previous one.

The screenshot shows the 'Events' interface for a project named 'accessing-guicreate'. It displays a timeline of events. Two specific events are highlighted with red boxes:

- VM accessing-vm** (Generated from virtualmachine-controller) - Started the virtual machine by creating the new virtual machine instance accessing-vm.
- VM accessing-vm** (Generated from virtualmachine-controller) - Stopped the virtual machine by deleting the virtual machine instance 28203353-4305-4668-b416-7b3ed5302916.

Other events listed include:  
VM accessing-vm (Generated from virtualmachine-controller) - Stopped the virtual machine by deleting the virtual machine instance 07fe7f77-2760-4841-95c3-7b392f305ee4.  
VM accessing-vm (Generated from virtualmachine-controller) - Created DataVolume accessing-vm.  
Older events are not stored.

**Figure 2.16: Review the events**

## Finish

As the student user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish accessing-guicreate
```

# Creating and Accessing Virtual Machines from the Command Line

## Objectives

- Provision and access new virtual machines by using command-line tools.

## Creating Virtual Machines From the Command Line

To create a VM automatically, use declarative files and Ansible Playbooks from the command line. You can edit the YAML configuration of a VM with a text editor or by using the `oc edit` command. Usually, a VM YAML file includes all the settings from a default template or an instance type in one structured document.

You can also use declarative files to back up the configuration of a VM to restore the settings to a new VMI.

### Creating a VM with Command-line Tools

You can use a YAML file to create a VM from the command line by using the `oc apply -f vm-manifest.yaml` command. You must provide the manifest file that contains the configuration of the VM.

The following example shows the configuration file of a VM that uses a PostgreSQL RHEL 9-based image:

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: postgresql-rhel9
spec:
  runStrategy: Always
  template:
    metadata:
      creationTimestamp: null
    spec:
      dataVolumeTemplates:
        - metadata:
            creationTimestamp: null
            name: postgresql-rhel9
          spec:
            source:
              registry:
                url: "registry.redhat.io/rhel9/postgresql-15:latest" ①
                secretRef: data-source-secret ②
              certConfigMap: tls-certs
        domain:
          devices: {}
          memory:
            guest: 512Mi
          resources: {}
```

```
terminationGracePeriodSeconds: 180
volumes: ③
  - dataVolume:
      name: rhel-9-minimal-volume
      name: rootdisk
status: {}
```

- ① URL for the image pull location
- ② Available secret to pull the image
- ③ A persistent volume that the VMI uses as the root disk for the OS

Use the `oc apply -f vm-manifest.yaml` command to create a VM from the manifest file:

```
[user@host ~]$ oc apply -f vm-manifest.yaml
virtualmachine.kubevirt.io/postgresql-rhel9 created
```

You can alternatively use the command line to edit a YAML configuration by using the `oc edit` command. You can update the attributes that are not marked as read-only in the manifest. To apply your changes, save the YAML configuration. The cluster applies changes to the resource when the editor buffer is written.

Use the `oc edit vm VM-name` command to edit the settings of a VM:

```
[user@host ~]$ oc edit vm postgresql-rhel9
virtualmachine.kubevirt.io/postgresql-rhel9 edited
```

After editing the configuration of the VM, save the changes and exit the text editor. After closing the text editor, restart the VM to create an instance with the latest changes by using the `virtctl restart VM-name` command:

```
[user@host ~]$ virtctl restart postgresql-rhel9
VM postgresql-rhel9 was scheduled to restart
```

With the `oc patch` command, you can update the fields of a resource by using a *strategic* merge patch, a JSON merge patch, or a JSON patch. Both JSON and YAML formats are accepted.

To add a label to a VM, use the `oc patch vm VM-name --type merge -p '{"spec": {"template": {"metadata": {"labels": {"key": "value"}}}}}'` command:

```
[user@host ~]$ oc patch vm postgresql-rhel9 --type merge -p '{"spec": {"template": {"metadata": {"labels": {"servertype": "production"}}}}}'
virtualmachine.kubevirt.io/postgresql-rhel9 patched
```

You can use the `virtctl` command-line tool to create a VM manifest file that contains general settings, such as the name, memory and CPU settings, and the namespace. The command dumps the configuration of a VM, in YAML format, to STDOUT, so you can copy and paste it to a YAML file.

To create the manifest file for a VM named `postgresql-rhel9`, in the `production` namespace, with 5 Gi of memory, use the `virtctl create vm --name postgresql-rhel9 --namespace production --memory=5Gi` command:

```
[user@host ~]$ virtctl create vm --name postgresql-rhel9 --namespace production  
--memory=5Gi  
apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
  creationTimestamp: null  
  name: postgresql-rhel9 ①  
  namespace: production ②  
spec:  
  runStrategy: Always  
  template:  
    metadata:  
      creationTimestamp: null  
    spec:  
      domain:  
        devices: {}  
        memory:  
          guest: 5Gi ③  
        resources: {}  
      terminationGracePeriodSeconds: 180  
status: {}
```

- ① Name of the VM.
- ② Namespace for creating the VM.
- ③ The VM is configured to use 5 Gi of memory.

You can use the `--volume-datasource` option to specify a data volume that the VM can use as the boot source. You must define the name of the data volume and the namespace where the resource is located. To create the manifest file of a VM named `postgresql-rhel9` that uses the `postgresql-dv` data volume, use the `virtctl create vm --name postgresql-rhel9 --volume-datasource=src:dv-namespace/dv-name` command:

```
[user@host ~]$ virtctl create vm --name postgresql-rhel9  
--volume-datasource=src:postgresql-project/postgresql-dv  
apiVersion: kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
  creationTimestamp: null  
  name: postgresql-rhel9  
  namespace: production  
spec:  
  dataVolumeTemplates:  
  - metadata:  
    creationTimestamp: null  
    name: postgresql-rhel9  
  spec:  
    sourceRef:  
      kind: DataSource  
      name: postgresql-dv ①  
      namespace: postgresql-project ②  
...output omitted...
```

- ① Name of the data volume
- ② Namespace where the data volume is

## Managing a VM with Command-line Tools

You can manage your VMs from the command line by using the `virtctl` tool. The `virtctl` command-line tool provides several options to manage the machine states, such as starting, stopping, restarting, and deleting the VM. You can download the `virtctl` binary from the Red Hat OpenShift console by going to **Virtualization > Overview** and clicking the **Download virtctl** link. Choose the corresponding binary for your operating system.

To list all the options that you can use to manage your VMs, use the `virtctl --help` command.

To stop a VM, use the `virtctl stop VM-name` command:

```
[user@host ~]$ virtctl stop postgresql-rhel9
VM postgresql-rhel9 was scheduled to stop
```

Use the `oc get vms` command to confirm that the VM is not running:

```
[user@host ~]$ oc get vms
NAME      AGE     STATUS    READY
postgresql-rhel9   40m   Stopped   False
```

You can start a stopped VM by using the `virtctl start VM-name` command:

```
[user@host ~]$ virtctl start postgresql-rhel9
VM postgresql-rhel9 was scheduled to start
```

To restart a VM, use the `virtctl restart VM-name` command:

```
[user@host ~]$ virtctl restart postgresql-rhel9
VM postgresql-rhel9 was scheduled to restart
```

Use the `oc get vms` command to confirm that the VM is starting. The status changes from **Stopping** to **Stopped**, and then to **Starting**:

```
[user@host ~]$ oc get vms
NAME      AGE     STATUS    READY
postgresql-rhel9   40m   Starting   False
```

You can use the `oc delete vm VM-name` command to delete a VM:

```
[user@host ~]$ oc delete vm postgresql-rhel9
virtualmachine.kubevirt.io "postgresql-rhel9" deleted
```

Confirm that both the VM and the associated VMI were deleted, by using the `oc get vms` and `oc get vmis` commands.

## Accessing the Console of a VM

You can access the serial or VNC consoles of a VM through the `virtctl` command-line tool. To access the serial console, use the `virtctl console VM-name` command. At the resulting prompt, you must provide the user and password that were configured during the VM creation:

```
[user@host ~]$ virtctl console postgresql-rhel9
Successfully connected to postgresql-rhel9 console.
The escape sequence is ^]

postgresql-rhel9 login: developer
Password:

[developer@postgresql-rhel9 ~]$
```

You can access the VNC console of a VM by using the `virtctl vnc VM-name` command. You must provide the user and password that were configured during the VM creation:

```
[user@host ~]$ virtctl vnc postgresql-rhel9
{"component": "", "level": "info", "msg": "--proxy-only is set to false, listening on
127.0.0.1\\n", "pos": "vnc.go:117", "timestamp": "2024-10-24T02:33:00.889615Z"}
...output omitted...
```

You can close the window to stop running the command.

## Accessing a VM with SSH

You can access a VM by using the SSH protocol from the command line. The `virtctl` tool provides a subcommand to connect to a VM by using the SSH protocol. The command requires the private key of an SSH key pair, which can be configured with a secret and injected into the VM during creation or after the VM is available.

The following example shows an example of how to use the `virtctl ssh` command to connect to a VM with the user that was configured during creation of the VM:

```
[user@host ~]$ virtctl ssh -i .ssh/lab_rsa --username developer postgresql-rhel9
...output omitted...
```

After you are done working with the VM, end the session by using the `exit` command.

## Accessing a VM with Port-Forward

Another method to access a VM from the command line is by using the `virtctl` tool to forward a VM port on the local machine.

The `virtctl port-forward vm/VM-name local-port:VM-port` command forwards a local port on the local machine to the specified port of the VM:

```
[user@host ~]$ virtctl port-forward vm/postgresql-rhel9 22080:80
{"component": "", "level": "info", "msg": "forwarding tcp 127.0.0.1:22080 to
80", "pos": "portforwarder.go:23", "timestamp": "2024-10-24T23:16:29.316910Z"}
```

After the connection of the VM is open on the specified port, you can access the VM by using a command-line client or a GUI client.

You can also use port-forwarding with the `oc` command-line client, by using the `oc port-forward` command. To connect to the VM, you must use the `virt-launcher` pod that is associated with the VM, the local port to forward the connection, and the port of the VM:

```
[user@host ~]$ oc port-forward pod/virt-launcher-postgresql-rhel9-gbxws 22080:80
Forwarding from 127.0.0.1:22080 -> 80
Forwarding from [::1]:22080 -> 80
```

**Note**

You can identify the `virt-launcher` pod that is associated with the VM by using the `oc get pods` command.

After the `virt-launcher` pod connection is open on the specified port, you can access the VM by using the command line or the GUI client. You must specify the localhost address and local port.

Press `Ctrl+C` to exit the `oc port-forward` command.

## Exposing a VM with Command-line Tools

You can create a service with the `virtctl` command-line tool that forwards a designated port of a VM and exposes the service on the specified port.

Use the `virtctl expose vm VM-name --name service-name --type ClusterIP|NodePort|LoadBalancer --port port` command to expose your VMs. If `--node-port` is not set when using the `NodePort` service type, then its value is allocated dynamically:

```
[user@host ~]$ virtctl expose vm postgresql-rhel9 --name postgresql-service --type
  ClusterIP --port 5432
Service postgresql-service successfully exposed for vm postgresql-rhel9
```

Confirm that the service was created by using the `oc get services` command:

```
[user@host ~]$ oc get services
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE
postgresql-service   ClusterIP  172.30.50.130  <none>        5432/TCP    12s
```

## Accessing GUI Consoles

The `remote-viewer` application is a remote desktop client to connect to a remote guest that supports the SPICE and VNC protocols, such as Windows, Linux, and MacOS. When you start a SPICE session, the `remote-viewer` application is called automatically. For Windows and MacOS distributions, you must install the `virt-viewer` application.

TigerVNC is a tool for graphical desktop sharing that is available for Linux, Windows, and MacOS. You must install the VNC desktop `vncserver` application on your machine to enable connections to remote machines. To enable external connections on your machine, you must install the `vncviewer` application.

Another VNC viewer that you can use in Windows and MacOS distributions is the TightVNC tool. TightVNC uses the VNC protocol to establish a connection between the server host and the client desktop, similar to TigerVNC.



## References

For more information about creating VMs with the `virtctl` tool, refer to the *Creating Virtual Machines* section at

[https://kubevirt.io/user-guide/user\\_workloads/creating\\_vms/](https://kubevirt.io/user-guide/user_workloads/creating_vms/)

For more information about managing VMs with the `virtctl` tool, refer to the *Lifecycle* section at

[https://kubevirt.io/user-guide/user\\_workloads/lifecycle/](https://kubevirt.io/user-guide/user_workloads/lifecycle/)

For more information about managing VMs with the `oc` command-line client, refer to the *Managing virtual machine instances* section at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/virtual-machines#virt-manage-vmis](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/virtual-machines#virt-manage-vmis)

For more information about accessing VMs with the `virtctl` tool, refer to the *VM Connection Commands* section at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/getting-started#virt-using-the-cli-tools](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/getting-started#virt-using-the-cli-tools)

For more information about port forwarding with the `oc` command-line client, refer to the *oc port-forward* section at

[https://docs.redhat.com/en/documentation/red\\_hat\\_build\\_of\\_microshift/4.16/html-single/cli\\_tools/microshift-oc-cli-commands#oc-policy-scc-subject-review](https://docs.redhat.com/en/documentation/red_hat_build_of_microshift/4.16/html-single/cli_tools/microshift-oc-cli-commands#oc-policy-scc-subject-review)

## ► Guided Exercise

# Create and Access a Virtual Machine from the Command Line

Create a Red Hat Enterprise Linux virtual machine from an existing MariaDB image and use it to manage a database server.

### Outcomes

- Create a RHEL 9 VM that contains a MariaDB server.
- Edit a VM to update the description.
- Access the VNC console of a VM.
- Access the MariaDB server by using port forwarding.
- Manage the status of a VM.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start accessing-clicreate
```

### Instructions

The `lab` command creates the `accessing-clicreate` namespace and configures a RHEL 9 VM boot source that contains a MariaDB server.

► 1. Log in as the `admin` user and use the `accessing-clicreate` namespace.

- 1.1. Use the command line to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Change to the `accessing-clicreate` namespace.

```
[student@workstation ~]$ oc project accessing-clicreate
Now using project "accessing-clicreate" on server "https://
api.ocp4.example.com:6443".
```

- ▶ 2. Create a RHEL 9 VM manifest file, configure it to use the `rhel9-do316` data source in the `openshift-virtualization-os-images` namespace, and create the VM from the command line.
- 2.1. Create a RHEL 9 VM YAML manifest in the `accessing-clicreate` namespace by using the `virtctl create vm` command. Use `rhel9-database` for the name of the VM, configure the VM with 5 Gi of memory, and use the `rhel9-do316` data source in the `openshift-virtualization-os-images` namespace.

```
[student@workstation ~]$ virtctl create vm \
--name rhel9-database \
--namespace accessing-clicreate --memory=5Gi \
--volume-datasource=src:openshift-virtualization-os-images/rhel9-do316
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  creationTimestamp: null
  name: rhel9-database
  namespace: accessing-clicreate
spec:
  dataVolumeTemplates:
  - metadata:
      creationTimestamp: null
      name: rhel9-database-ds-rhel9-do316
    spec:
      sourceRef:
        kind: DataSource
        name: rhel9-do316
        namespace: openshift-virtualization-os-images
      storage:
        resources: {}
  preference:
    inferFromVolume: rhel9-database-ds-rhel9-do316
    inferFromVolumeFailurePolicy: Ignore
  runStrategy: Always
  template:
    metadata:
      creationTimestamp: null
    spec:
      domain:
        devices: {}
        memory:
          guest: 5Gi
        resources: {}
      terminationGracePeriodSeconds: 180
      volumes:
      - dataVolume:
          name: rhel9-database-ds-rhel9-do316
          name: rhel9-database-ds-rhel9-do316
  status: {}
```

- 2.2. Create the `rhel9-database.yaml` file and copy the result of the previous command into the file.

- 2.3. Use the `oc apply` command to create the resource. The `~/D0316/labs/` `accessing-clicreate/rhel9-database.yaml` file contains the output of the `virtctl create` command.

```
[student@workstation ~]$ oc apply -f rhel9-database.yaml
virtualmachine.kubevirt.io/rhel9-database created
```

- 2.4. Confirm that the `rhel9-database` VM is running.

```
[student@workstation ~]$ oc get vms
NAME    AGE    STATUS
rhel9-database  53s   Running   True
```

- ▶ 3. Update the description of the VM from the command line, and re-create the VMI to contain the latest changes.

- 3.1. Edit the VM configuration to add the A RHEL 9 VM that contains a MariaDB server description by using the `oc patch vm rhel9-database --type merge -p '{"spec":{"template":{"metadata":{"annotations":{"description":"A RHEL 9 VM that contains a MariaDB server"}}}}}'` command.

```
[student@workstation ~]$ oc patch vm rhel9-database --type json -p='[
  {"op":"add","path":"/spec/template/metadata/annotations",
   "value":{"description":"A RHEL 9 VM that contains a MariaDB server"}}
]'

virtualmachine.kubevirt.io/rhel9-database patched
```



#### Note

Alternatively, you can update the `rhel9-database.yaml` manifest with the annotation and then apply the changes with the `oc apply -f rhel9-database.yaml` command.

- 3.2. Restart the `rhel9-database` VM.

```
[student@workstation ~]$ virtctl restart rhel9-database
VM rhel9-database was scheduled to restart
```

- 3.3. Confirm that the VMI is ready before moving to the next step.

```
[student@workstation ~]$ oc get vmis
NAME    AGE    PHASE
rhel9-database  16s   Running
```

- 3.4. Use the `oc describe vmi rhel9-database` command to confirm that the added description is present in the metadata section.

```
[student@workstation ~]$ oc describe vmi rhel9-database
Name:          rhel9-database
Namespace:     accessing-clicreate
...output omitted...
Annotations:   description: A RHEL 9 VM that contains a MariaDB server
...output omitted...
```

▶ 4. Access the VNC console of the VM.

- 4.1. Use the `virtctl vnc rhel9-database` command to access the VNC console of the VM. Log in as the `root` user with `redhat` as the password.

```
[student@workstation ~]$ virtctl vnc rhel9-database
{"component": "", "level": "info", "msg": "--proxy-only is set to false, listening on
127.0.0.1\n", "pos": "vnc.go:117", "timestamp": "2024-11-01T01:00:16.669580Z"}
...output omitted...
```

- 4.2. Retrieve the status of the MariaDB server.

```
[root@localhost ~]$ systemctl status mariadb.service
mariadb.service - MariaDB 10.5 database server
  Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset:
disabled)
  Active: active (running) since Tue 2024-11-05 21:05:30 EST 1h 34min ago
...output omitted...
```

- 4.3. Close the window to finish the connection.

▶ 5. Redirect the network traffic from port 13306 of the `workstation` machine to port 3306 on the VM. The MariaDB database that is running inside the `rhel9-database` VM listens on TCP port 3306.

- 5.1. Use the `virtctl port-forward vm/rhel9-database` command to open a connection to the MariaDB server inside the `rhel9-database` VM. Redirect the VM network traffic that comes from the 3306 port to the port 13306 of the `workstation` machine.

```
[student@workstation ~]$ virtctl port-forward vm/rhel9-database 13306:3306
>{"component": "", "level": "info", "msg": "forwarding tcp 127.0.0.1:13306 to
3306", "pos": "portforwarder.go:23", "timestamp": "2024-11-05T22:49:14.722058Z"}
```

▶ 6. Open a new terminal and access the MariaDB server in the `rhel9-database` VM. Confirm that you can use the `sakila` database and list the tables in the database.

- 6.1. Use the MySQL client in the `workstation` machine to connect to the MariaDB server. Log in as `devuser` with `developer` as the password, and use the `sakila` database.

```
[student@workstation ~]$ mysql -h 127.0.0.1 \
-u devuser -p'developer' --port 13306 sakila
...output omitted...

mysql>
```

6.2. List the tables of the `sakila` database.

```
mysql> SHOW TABLES ;
+-----+
| Tables in sakila      |
+-----+
actor
actor_info
address
category
...output omitted...
+-----+
23 rows in set (0.00 sec)
```

6.3. Close the connection to the database.

```
mysql> quit
Bye
[student@workstation]$
```

You can safely close the terminal window. Press `Ctrl+C` to close the port forwarding connection in the first window.

## ► 7. Stop the VM.

7.1. Stop the `rhel9-database` VM by using the `virtctl stop` command.

```
[student@workstation ~]$ virtctl stop rhel9-database
VM rhel9-database was scheduled to stop
```

7.2. Confirm that the `rhel9-database` VM is stopped.

```
[student@workstation ~]$ oc get vms
NAME     AGE     STATUS     READY
rhel9-database 45m   Stopped   True
```

## ► 8. Delete the VM by using the `oc` command.

8.1. Delete the `rhel9-database` VM.

```
[student@workstation ~]$ oc delete vm rhel9-database
virtualmachine.kubevirt.io "rhel9-database" deleted
```

8.2. Confirm that the `rhel9-database` VM is not present.

```
[student@workstation ~]$ oc get vms  
No resources found in accessing-clicreate namespace.
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish accessing-clicreate
```

# Monitoring and Inspecting Virtual Machines

## Objectives

- Monitor virtual machine resources and inspect the configuration of a virtual machine.

## Virtual Machine Monitoring

Red Hat OpenShift provides a monitoring stack for cluster resources that is based on the open source Prometheus project.

Users can view details by using logging and monitoring dashboards for each VM in the OpenShift web console, similar to VMware's vCenter and Aria.

To view a VM's metrics, such as resource utilization and cluster-level events, use the OpenShift web console to go to **Virtualization > VirtualMachines**. Select the VM's project from the **Projects** menu and then click the VM's name to open the **Overview** tab for the VM.

The **Details** card on the **Overview** tab provides identifying information about the VM, including its name, status, date of creation, hostname, and operating system. The QEMU guest agent that is installed in the VM provides information about the VM, users, file systems, and secondary networks. If the QEMU guest agent is not installed on the VM, then the overview and the details tabs display information about the operating system that was specified when the VM was created.

The **Utilization** card includes CPU, Memory, Storage, and Network Transfer usage charts. From this card, cluster administrators can select a duration for the utilization data.

The **Overview** tab provides additional cards to summarize hardware devices, alerts, snapshots, network interfaces, and disks.

From the developer perspective, cluster administrators can access the **Observe** menu to review dashboards with information about each project's resource utilization, such as usage metrics for CPU, memory, network, and storage I/O.

From the administrator perspective, cluster administrators can go to **Virtualization > Overview** and select the **Top consumers** tab. This tab lists the VMs that are the top consumers of resources, such as CPU, memory, and storage throughput.

## Command-line Monitoring

The `oc` client is a command-line utility for managing Red Hat OpenShift resources. Administrators and developers can use the `oc` command to develop, build, deploy, and run applications or VM workloads in Red Hat OpenShift. You can review and diagnose events and conditions for VM and storage resources. For example, cluster administrators can review a VM's logs from the `virt-launcher` pod to debug issues by using the `oc logs` command:

```
[user@host ~]$ oc logs virt-launcher-name
```

Cluster administrators can also use the `oc get` and `oc describe` commands to retrieve the status of all VMs in the cluster or a specific VM in a namespace.

## Chapter 2 | Running and Accessing Virtual Machines

As a cluster administrator, you can use the `oc get` command with the `-A` flag to list the status of all VMs in the cluster:

```
[user@host ~]$ oc get vm -A
```

Use the following `oc get` command to retrieve the status of all VMs in a specific namespace:

```
[user@host ~]$ oc get vm -n namespace
```

You can also use the `oc describe` command to retrieve detailed information for a VM, such as recent events and specifications:

```
[user@host ~]$ oc describe vm vm-name -n namespace
```



### Note

Use the `oc help` command for more information about using the `oc` command.

As a cluster administrator, you can manage and inspect a VM by using the `virsh` command tool within the VM's `virt-launcher` pod and its `libvirt` container.

Execute the `virsh` command tool to manage and monitor a VM from the console of the VM's `virt-launcher` pod by using either the OpenShift web console or the `oc` client. From the OpenShift web console, go to **Workloads > Pods** and select the VM's `virt-launcher` pod. Click the **Console** tab to open the pod's console. Use the following command to access the `virt-launcher` pod's console with the `oc` client:

```
[user@host ~]$ oc exec -it virt-launcher-pod-name -n namespace -- /bin/bash
```

You can use the `virsh event vm-name --all` command to list all the types of events of the VM.

```
[user@host ~]$ virsh event vm-name --all
Authorization not available. Check if polkit service is running or see debug
message for more information.
event 'rtc-change' for domain 'accessing-guicreate_vm-name': 0
events received: 1
```

Use the `virsh dommemstat vm-name` command to display the available memory of the VM:

```
[user@host ~]$ virsh dommemstat vm-name
Authorization not available. Check if polkit service is running or see debug
message for more information.
actual 2097152
swap_in 0
swap_out 0
major_fault 999
minor_fault 815298
unused 3207612
available 3829592
```

```
usable 3246248
last_update 1728277931
disk_caches 241872
hugegetlb_pgalloc 0
hugegetlb_pgfail 0
rss 949396
```

Use the `virsh domstate vm-name` command to get the state of the VM:

```
[user@host ~]$ virsh domstate vm-name
Authorization not available. Check if polkit service is running or see debug
message for more information.
running
```

## Managing Virtual Machine Instances

Red Hat OpenShift 4.16 provides functions with the `virtctl` command tool to reduce the need to access the `virt-launcher` pod's console. For example, as a cluster administrator, you can view a list of file systems that are available to the VMI with the `virtctl fslist vmi-name` command.

```
[user@host ~]$ virtctl fslist production-mariadb -n production-vms
{
  "metadata": {},
  "items": [
    {
      "diskName": "vda2",
      "mountPoint": "/boot/efi",
      "fileSystemType": "vfat",
      "usedBytes": 6006784,
      "totalBytes": 104634368
    },
    {
      "diskName": "vda3",
      "mountPoint": "/",
      "fileSystemType": "xfs",
      "usedBytes": 2419097600,
      "totalBytes": 10619924480
    }
  ]
}
```

Additionally, you can view guest agent information about the VMI's operating system with the `virtctl guestosinfo vmi-name` command.



### Note

Use the `virtctl help` command for a descriptive list of `virtctl` commands.

## Role-based Access Control

Red Hat OpenShift uses *role-based access control* (RBAC) to grant permissions to cluster resources, including project monitoring.

RBAC is a technique for managing access to computer system resources. In Red Hat OpenShift, RBAC determines whether a user can perform certain actions within the cluster or project. You can choose between two role types, cluster or local, depending on the user's level of responsibility.



### Note

Cluster role bindings take precedence over local role bindings.

Authorization is a separate step from authentication. The authorization process is managed by rules, roles, and bindings.

RBAC object	Description
Rule	Allowed actions for objects or groups of objects.
Role	Sets of rules. Users and groups can be associated with multiple roles.
Binding	Assignment of users or groups to a role.

Red Hat OpenShift defines two groups of roles and bindings according to the user's scope and responsibility: cluster roles and local roles.

Role level	Description
Cluster role	Users or groups with this role level can manage the OpenShift cluster.
Local role	Users or groups with this role level can manage only elements at a project level.

As a cluster administrator, you can monitor and query metrics on all projects in the cluster.

By default, developer users can monitor only projects where they have the `view` role permissions. Developer users can view monitoring information in the OpenShift web console developer perspective from the **Observe > Metrics** menu. From there, a developer can run performance metric queries, one at a time.

Cluster administrators can grant permissions for additional projects to developer users.

As a cluster administrator, use the following command to grant `view` role permissions to a user:

```
[user@host ~]$ oc create rolebinding view \
--clusterrole=view --user=user -n project
```

To grant the `view` role permissions to a group, run the following command:

```
[user@host ~]$ oc create rolebinding view \
--clusterrole=view --group=group -n project
```

Alternatively, use the web console to add a role binding to a user. Log in to the web console as an administrator, select the **Administrator** perspective, and go to **User Management > Users**.

Name	Full name
U admin	-
U dbadmin1	-
U developer	-

Figure 2.17: Manage users from the web console

Select a user to go to the user's details page. From the user details page, select the **RoleBindings** tab, and then click **Create binding** to access the **Create RoleBinding** page.

Figure 2.18: Create a role binding

## Chapter 2 | Running and Accessing Virtual Machines

The fields on the **Create RoleBinding** page are identical to the fields in the `oc create rolebinding` command.

Enable monitoring for user-defined projects by setting the `enableUserWorkload: true` field in the `cluster-monitoring-config` configuration map in the `openshift-monitoring` namespace.

Cluster administrators can make this change:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true
```

## Querying Virtual Resource Metrics

Only cluster administrators have access to all namespaces concurrently in the **Observe > Metrics** interface. From the Developer perspective, you can go to **Observe** and then click the **Dashboard** tab to review various metrics, including CPU and memory usage charts, current network usage, bandwidth, and storage I/O for individual projects.

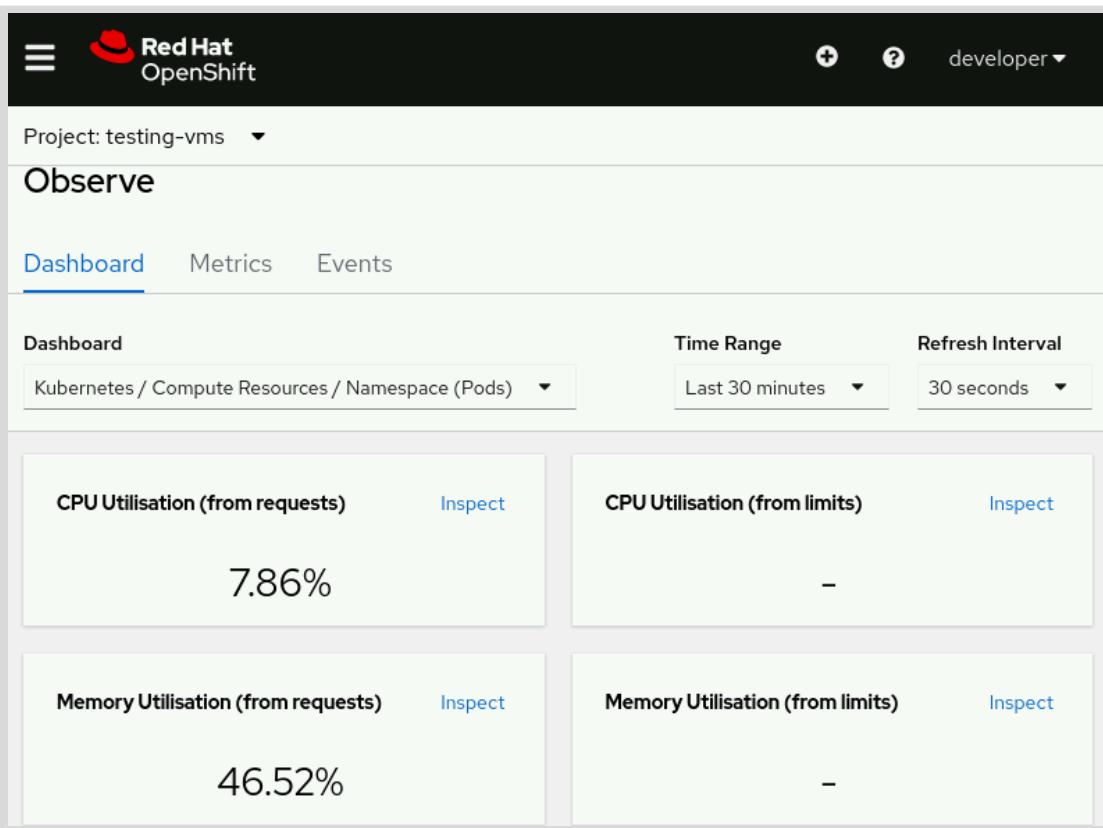
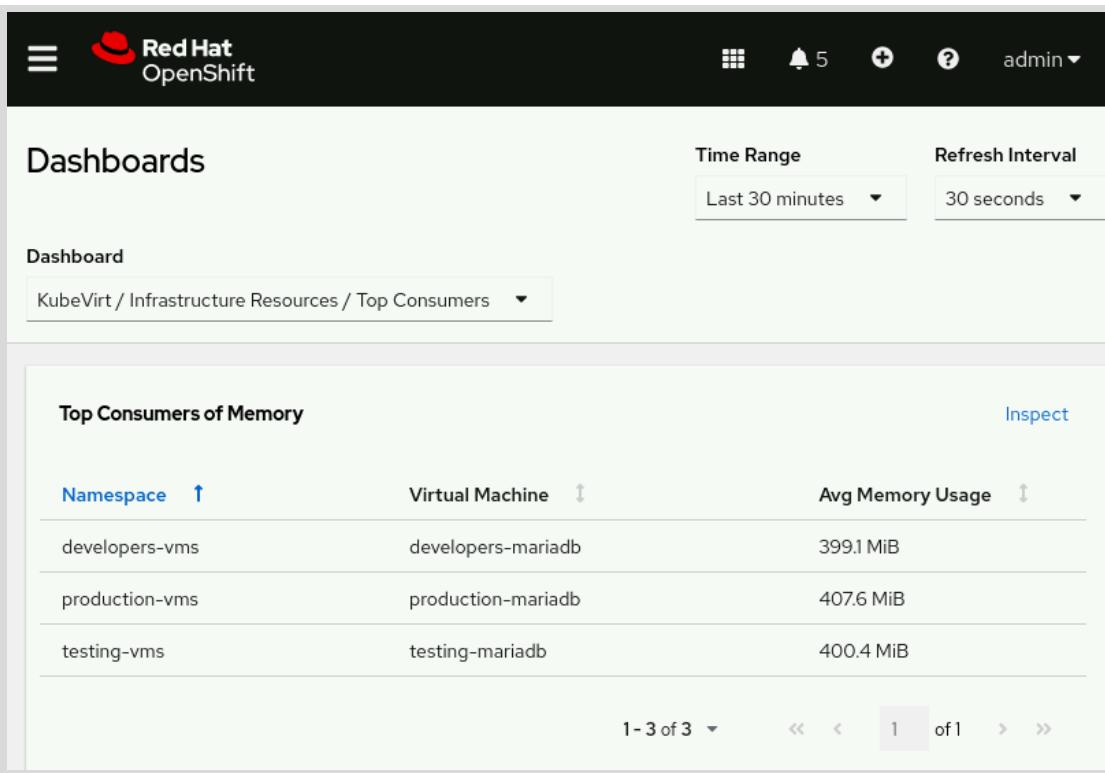


Figure 2.19: Developer view of the monitoring dashboard

As a cluster administrator, you can go to **Observe > Dashboards** and then select the **KubeVirt / Infrastructure Resources / Top Consumers** dashboard from the **Dashboard** list. You can use this

dashboard to review metrics for all VMs in the cluster and to identify the VMs that are consuming the most resources.



The screenshot shows the Red Hat OpenShift monitoring interface. At the top, there's a navigation bar with the Red Hat OpenShift logo and a user dropdown for 'admin'. Below the header, a 'Dashboards' section is visible with 'Time Range' set to 'Last 30 minutes' and 'Refresh Interval' set to '30 seconds'. A 'Dashboard' card is selected, showing the path 'KubeVirt / Infrastructure Resources / Top Consumers'. The main content area displays a table titled 'Top Consumers of Memory' with three columns: 'Namespace', 'Virtual Machine', and 'Avg Memory Usage'. The data shows three entries: 'developers-vms' with 'developers-mariadb' and 399.1 MiB; 'production-vms' with 'production-mariadb' and 407.6 MiB; and 'testing-vms' with 'testing-mariadb' and 400.4 MiB. There are also navigation controls at the bottom of the table.

Namespace ↑	Virtual Machine ↑	Avg Memory Usage ↑
developers-vms	developers-mariadb	399.1 MiB
production-vms	production-mariadb	407.6 MiB
testing-vms	testing-mariadb	400.4 MiB

**Figure 2.20: Administrator view of monitoring dashboard**

You can also access the **Observe > Metrics** interface to execute *Prometheus Query Language (PromQL)* queries against an entire Red Hat OpenShift Virtualization environment. Use this tool to examine information about the cluster and user-defined workloads as a dashboard or a graph.

Various metrics can help administrators and developers when writing PromQL queries for virtualization monitoring:

#### `kubevirt_vmi_vcpu_wait_seconds`

Returns the wait time (in seconds) for a VM's vCPU.

#### `kubevirt_vmi_network_receive_bytes_total`

Returns the total amount of received traffic (in bytes) on the VM's network.

#### `kubevirt_vmi_network_transmit_bytes_total`

Returns the total amount of transmitted traffic (in bytes) on the VM's network.

#### `kubevirt_vmi_storage_read_traffic_bytes_total`

Returns the total amount of storage that is read (in bytes) by the VM.

#### `kubevirt_vmi_storage_write_traffic_bytes_total`

Returns the total amount of storage that is written (in bytes) for the VM's storage-related traffic.

#### `kubevirt_vmi_storage_iops_read_total`

Returns the amount of read I/O operations that the VM is performing per second.

#### `kubevirt_vmi_storage_iops_write_total`

Returns the amount of write I/O operations that the VM is performing per second.

**kubevirt\_vmi\_memory\_swap\_in\_traffic\_bytes\_total**

Returns the total amount (in bytes) of memory that the virtual guest is swapping in.

**kubevirt\_vmi\_memory\_swap\_out\_traffic\_bytes\_total**

Returns the total bytes of memory that the virtual guest is swapping out.



## References

For more information about this topic, refer to the *Using the CLI Tools, and Logging, Events, and Monitoring* chapters in the *Red Hat OpenShift Container Platform Virtualization* guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index)

For more information about RBAC, refer to the *Using RBAC to Define and Apply Permissions* chapter in the Red Hat OpenShift Container Platform 4.16 *Authentication and Authorization* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/authentication\\_and\\_authorization/index#authorization-overview\\_using-rbac](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/authentication_and_authorization/index#authorization-overview_using-rbac)

This topic is covered in more detail in the *Red Hat OpenShift Administration II: Configuring a Production Cluster* training course at

<https://rol.redhat.com/rol/app/courses/do280-4.14/pages/ch03>

For more information about enabling monitoring for user-defined projects, refer to the *Enabling Monitoring for User-defined Projects* chapter in the Red Hat OpenShift Container Platform 4.16 *Observability* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/monitoring/index#enabling-monitoring-for-user-defined-projects](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/monitoring/index#enabling-monitoring-for-user-defined-projects)

For more information about Prometheus queries, refer to the *Querying Prometheus* guide at

<https://prometheus.io/docs/prometheus/latest/querying/basics/>

## ► Guided Exercise

# Monitor and Inspect Virtual Machines

Retrieve the performance metrics of virtual machines in different projects.

## Outcomes

- Inspect the current RBAC policies of different projects.
- Use RBAC policies to permit a non-cluster administrator to view the performance metrics of virtual machines in a project.
- Retrieve performance metrics from virtual machines by using the Red Hat OpenShift web console and PromQL syntax.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start accessing-monitor
```

This command ensures that all resources are available for this exercise.

## Instructions

- 1. Grant permissions to the `developer` user to retrieve metrics of the VMs in different projects. Create a role binding to assign the `view` cluster role to the `developer` user in each of the three projects. Use `view` as the name for each new role binding.

- 1.1. Use the command line to log in to the cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. List the current role bindings of the `developers-vms`, `production-vms`, and `testing-vms` projects.

```
[student@workstation ~]$ oc get rolebindings -n developers-vms
NAME          ROLE           AGE
system:deployers   ClusterRole/system:deployer   2h
system:image-builders   ClusterRole/system:image-builder   2h
system:image-pullers   ClusterRole/system:image-puller   2h
```

```
[student@workstation ~]$ oc get rolebindings -n production-vms
NAME          ROLE           AGE
system:deployers   ClusterRole/system:deployer   2h
system:image-builders   ClusterRole/system:image-builder   2h
```

```
system:image-pullers    ClusterRole/system:image-puller    2h

[student@workstation ~]$ oc get rolebindings -n testing-vms
NAME          ROLE           AGE
system:deployers   ClusterRole/system:deployer   2h
system:image-builders ClusterRole/system:image-builder 2h
system:image-pullers   ClusterRole/system:image-puller  2h
```

- 1.3. Use the `oc create` command to create the `view` role binding in the `developers-vms` project.

```
[student@workstation ~]$ oc create rolebinding view \
--clusterrole=view --user=developer -n developers-vms
rolebinding.rbac.authorization.k8s.io/view created
```

- 1.4. Confirm that the `developer` user has the assigned permissions on the `developers-vms` projects.

```
[student@workstation ~]$ oc get rolebindings view -n developers-vms -o wide
NAME  ROLE           AGE  USERS      GROUPS      SERVICEACCOUNTS
view   ClusterRole/view  2m   developer
```

- 1.5. Assign the `view` cluster role binding to the other two projects.

```
[student@workstation ~]$ oc create rolebinding view \
--clusterrole=view --user=developer -n production-vms
rolebinding.rbac.authorization.k8s.io/view created

[student@workstation ~]$ oc create rolebinding view \
--clusterrole=view --user=developer -n testing-vms
rolebinding.rbac.authorization.k8s.io/view created
```

- 1.6. Confirm that the `developer` user has the assigned permissions on the other two projects.

```
[student@workstation ~]$ oc get rolebindings view -n production-vms -o wide
NAME  ROLE           AGE  USERS      GROUPS      SERVICEACCOUNTS
view   ClusterRole/view  2m   developer

[student@workstation ~]$ oc get rolebindings view -n testing-vms -o wide
NAME  ROLE           AGE  USERS      GROUPS      SERVICEACCOUNTS
view   ClusterRole/view  2m   developer
```

- 2. As the `admin` user, log in to the OpenShift web console and view the top consumers of resources for all VMs.

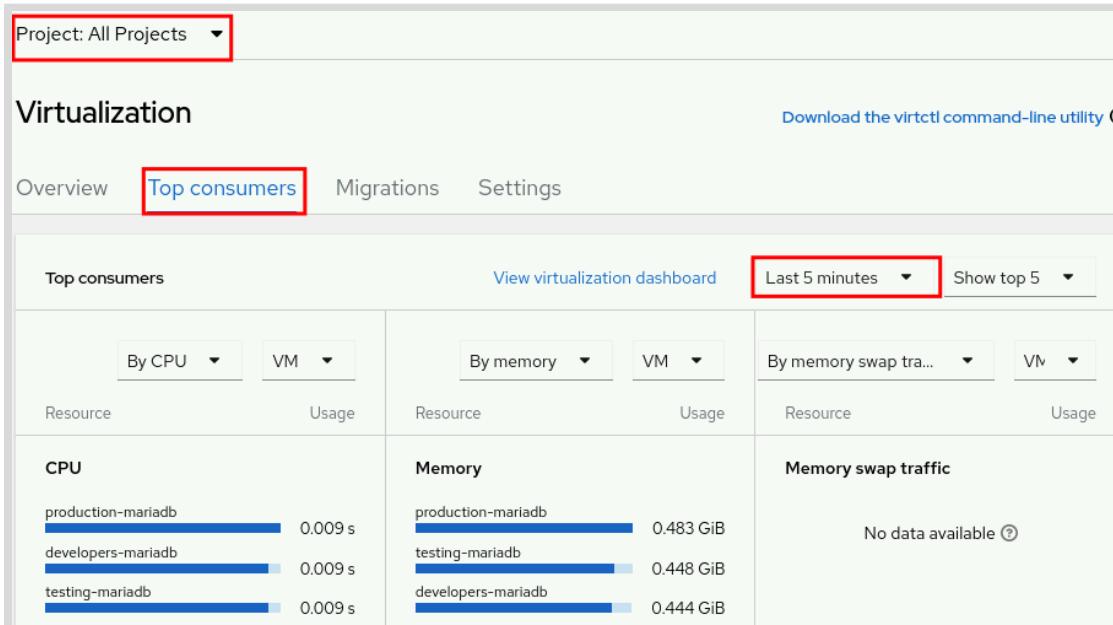
- 2.1. Open a web browser and go to the web console URL.

<https://console-openshift-console.apps.ocp4.example.com>

Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.

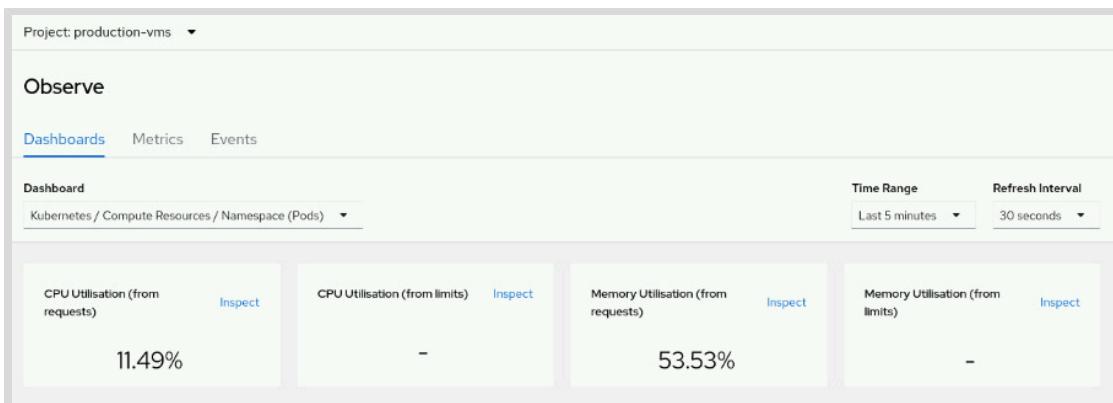
- 2.2. Use the perspective switcher to select the `Administrator` view.

- 2.3. From the OpenShift web console left panel, go to **Virtualization > Overview** and select **Top consumers**. Set the time range to **Last 5 minutes** for a more granular view, and select all projects from the **Projects** menu. Notice the CPU and memory usage.



**Figure 2.21: Overview of top consumers**

- 3. Observe the metrics dashboard for the **production-vms** project.
- 3.1. Use the perspective switcher to change to the **Developer** view.
  - 3.2. From the OpenShift web console left panel, go to **Observe**.
  - 3.3. From the **Project** drop-down menu, select the **production-vms** project.
  - 3.4. Read through the performance metrics. Set the time range to **Last 5 minutes** for a more granular view.



**Figure 2.22: Developer performance metrics dashboard**

The performance metrics are for all pods in the project, including the **virt-launcher** pod. Scroll down to review additional metrics, such as the **CPU Usage** graph.

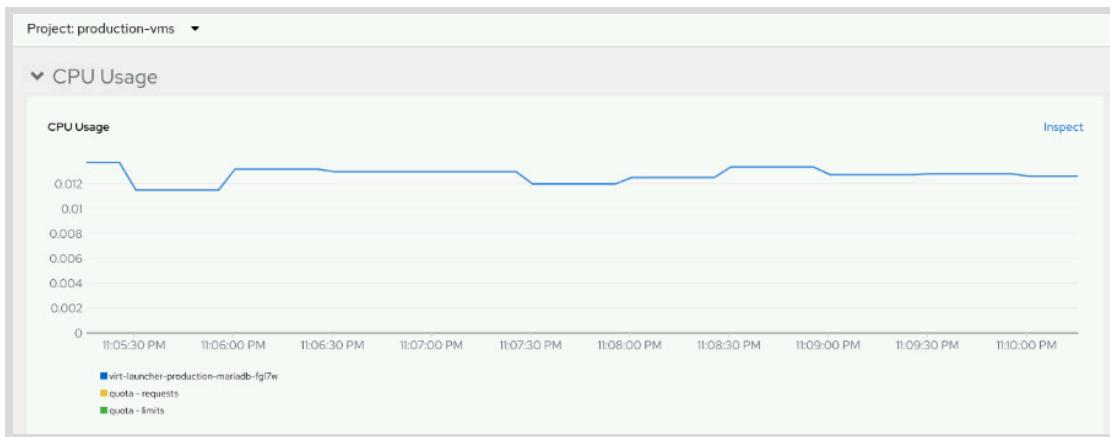


Figure 2.23: CPU usage metrics production

- ▶ 4. View the performance metrics for the VM that is running in the **production-vms** project.
  - 4.1. Use the perspective switcher to change to the **Administrator** view.
  - 4.2. Go to **Virtualization > Virtual Machines** and select the **production-vms** project.
  - 4.3. Select the **production-mariadb** VM and go to the **Metrics** tab.
  - 4.4. Set the time range to **Last 5 minutes** and review the performance metrics of the **production-mariadb** VM.

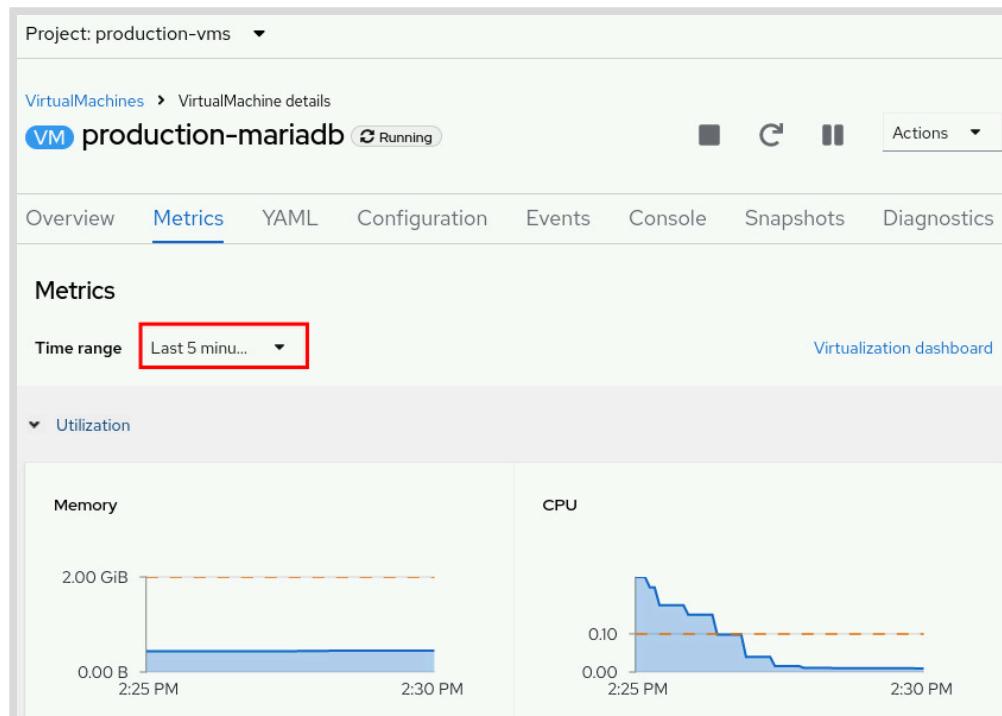


Figure 2.24: VM performance metrics

- ▶ 5. Generate resource loads on the **production-mariadb** VM and monitor the resource consumption.

- 5.1. Open the VNC console of the production-mariadb virtual machine. Click the **Console** tab, then select **VNC console** and log in as the **developer** user with **developer** as the password.

```
...output omitted...
production-mariadb login: developer
Password: developer
[developer@production-mariadb ~]$
```

- 5.2. Execute the **stress-load.sh** script in the home directory of the **developer** user to generate a stress load on the virtual machine. The script takes approximately two minutes to complete. You can proceed to the next step while the script executes.

```
[developer@production-mariadb ~]$ ./stress-load.sh
stress-ng: info: [16243] setting to a 2 mins, 0 secs run per stressor
stress: info [16243] dispatching hogs: 8 cpu, 2 hdd, 6 io, 1 vm
stress: info [16254] io: this is a legacy I/O sync stressor, consider using iomix
instead
...output omitted...
```

- 5.3. Go to the **Metrics** tab and set the time range to **Last 5 minutes** for a more granular view. Notice the gradual increase in CPU and memory resource consumption.

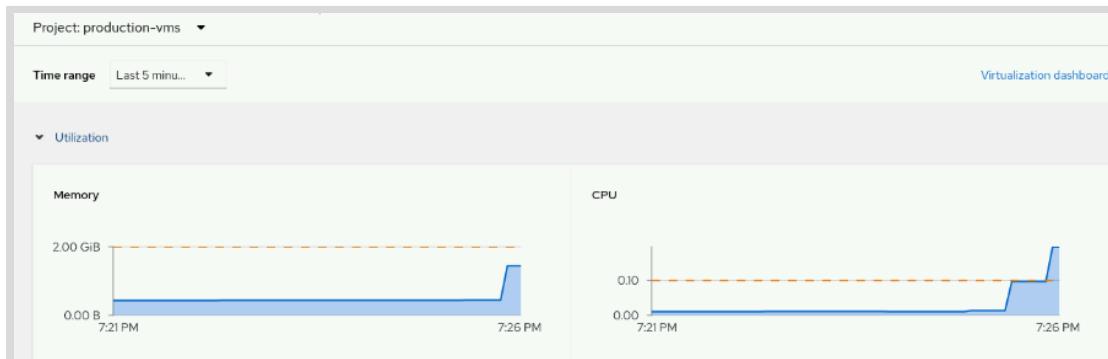
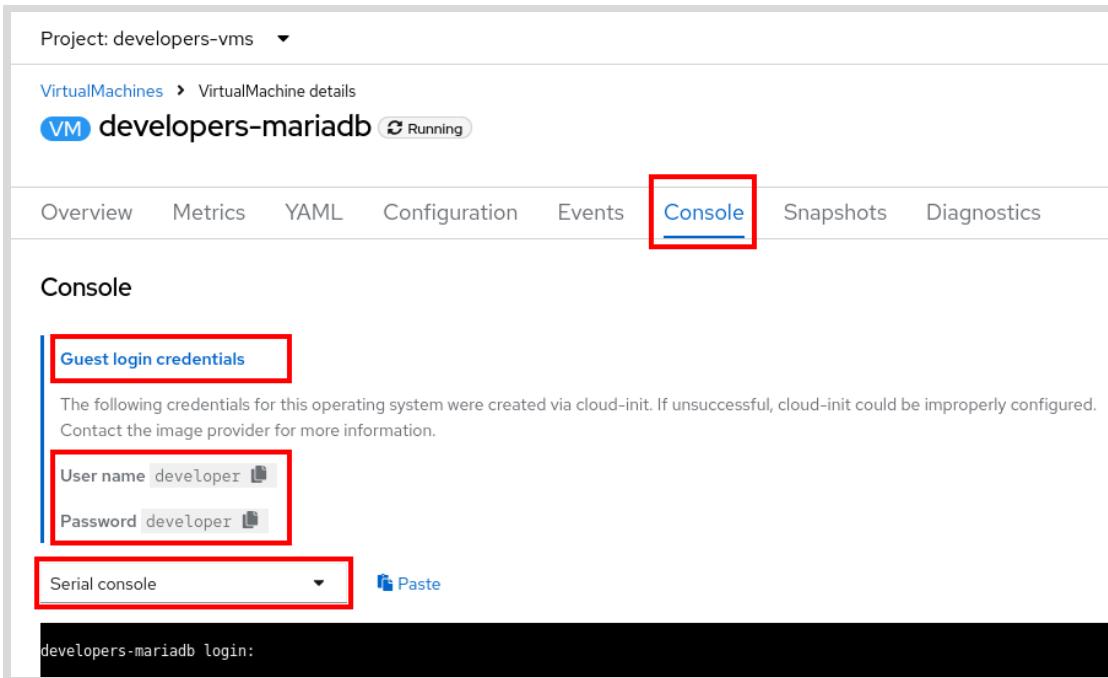


Figure 2.25: Metrics of the 'production-mariadb' virtual machine

- 6. Generate resource loads on the **developers-mariadb** VM and monitor the resource consumption.

- 6.1. Select the **developers-vms** project from the **Project** list.
- 6.2. Click **developers-mariadb**, then click the **Console** tab and click **Guest login credentials** to view the VM username and password. Select **Serial console** and log in with the guest credentials above the console prompt.

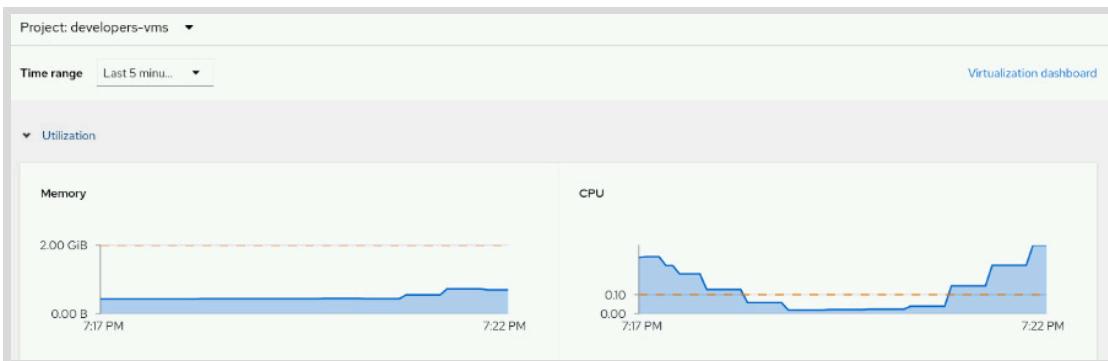


**Figure 2.26: VNC console of the 'developers-mariadb' virtual machine**

- 6.3. Execute the `stress-load.sh` script in the home directory of the `developer` user to generate a stress load on the virtual machine. The script takes approximately two minutes to complete. You can proceed to the next step while the script executes.

```
[developer@developers-mariadb ~]$ ./stress-load.sh
stress-ng: info: [16243] setting to a 2 mins, 0 secs run per stressor
stress: info [16243] dispatching hogs: 2 cpu, 6 hdd, 2 io, 4 vm
stress: info [16252] io: this is a legacy I/O sync stressor, consider using iomix
instead
...output omitted...
```

- 6.4. Go to the **Metrics** tab and set the time range to **Last 5 minutes** for a more granular view. Notice the increase in CPU and memory resource consumption.



**Figure 2.27: Metrics of the 'developers-mariadb' virtual machine**

7. Generate resource loads on the `testing-mariadb` VM and monitor the resource consumption.

- 7.1. Use the `virtctl console` command to connect to the VM serial console. Log in as the `developer` user with `developer` as the password.

```
[student@workstation ~]$ virtctl console testing-mariadb -n testing-vms  
...output omitted...  
  
testing-mariadb login: developer  
Password: developer  
...output omitted...  
[developer@testing-mariadb ~]$
```

- 7.2. Execute the `stress-load.sh` script in the home directory of the `developer` user to generate a stress load on the virtual machine. The script takes approximately two minutes to complete. You can proceed to the next step while the script executes.

```
[developer@testing-mariadb ~]$ ./stress-load.sh  
stress-ng: info: [16249] setting to a 2 mins, 0 secs run per stressor  
stress: info [16249] dispatching hogs: 6 cpu, 4 hdd, 8 io, 1 vm  
stress: info [16260] io: this is a legacy I/O sync stressor, consider using iomix  
instead  
...output omitted...
```

- 7.3. Log out of the console session, and then press `Ctrl+]` to exit the serial console.

```
[developer@testing-mariadb ~]$  
[student@workstation ~]$
```

- 7.4. Go to the **Metrics** tab and notice the increase in CPU and memory resource. Set the time range to `Last 5 minutes` for a more granular view.

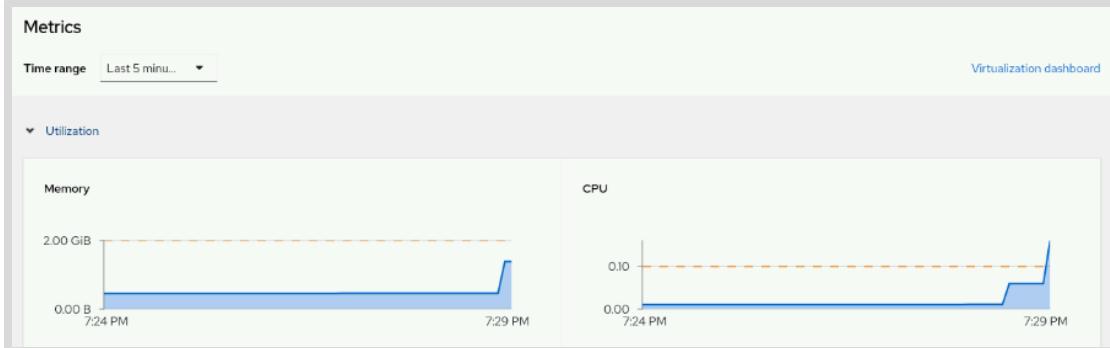


Figure 2.28: Metrics of the 'testing-mariadb' virtual machine

- 8. As the `admin` user, log in to the OpenShift web console and view the top consuming resources of all VMs.
- 8.1. If the `Administrator` view is not already active, then use the perspective switcher to select this view.
  - 8.2. Go to `Virtualization > Overview` and select `Top consumers`. Set the time range to `Last 5 minutes` for a more granular view, and select all projects from the `Projects` menu. Notice the increase in the various performance metrics.

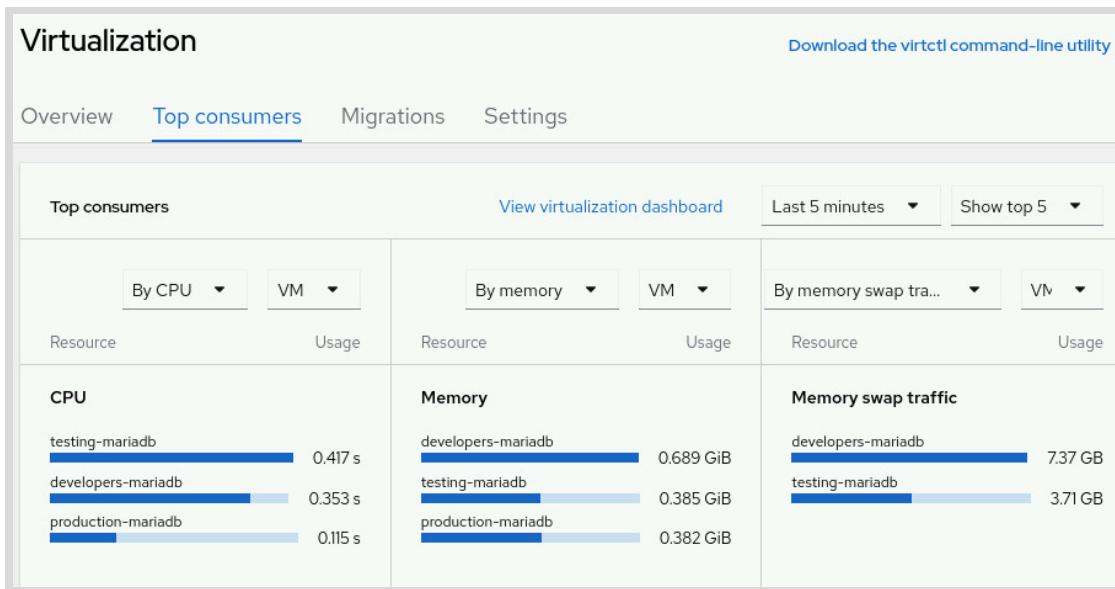


Figure 2.29: Metrics of top consumers

- ▶ 9. Use the **Observe > Dashboards** interface to retrieve updated performance metrics for all VMs in the `production-vms`, `developers-vms`, and `testing-vms` projects.
- 9.1. From the Administrator perspective, go to **Observe > Dashboards**.
  - 9.2. Select the **KubeVirt / Infrastructure Resources / Top Consumers** option from the **Dashboard** menu. You can use this dashboard to identify the top consumers of memory, of CPU by `virt-launcher` pods, of storage IOPs, and so on.
  - 9.3. Examine the reported metrics for the VMs. Each panel includes an **Inspect** button that opens a Metrics page with the PromQL query that the panel executes. Click the **Inspect** button to view the executed PromQL query for the **Top Consumers of Memory** panel.

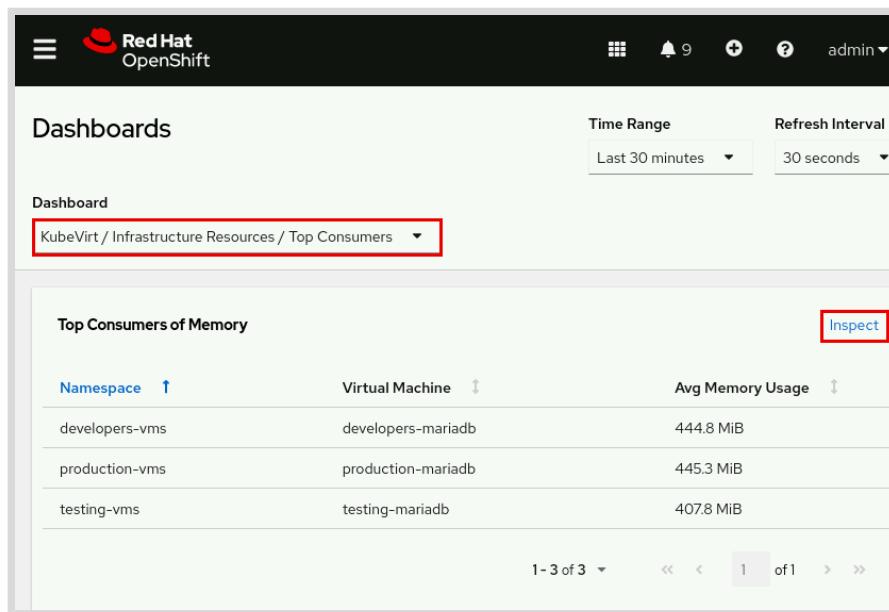
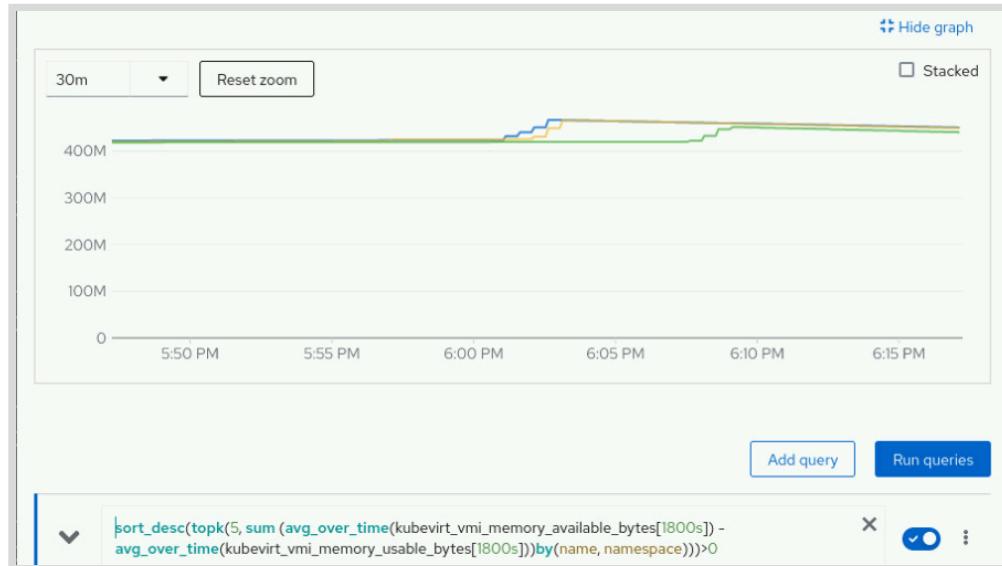


Figure 2.30: Top consumers of memory



### Note

A file with more queries is available at `~/DO316/labs/accessing-monitor/queries.txt`. Use this file to copy, paste, and observe additional metrics from the PromQL queries as needed.



**Figure 2.31: Input expression query**

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish accessing-monitor
```

## ▶ Lab

# Create, Configure, and Monitor Virtual Machines

Create virtual machines in different projects and retrieve metrics to monitor the resource consumption of the virtual machines.

## Outcomes

- Grant permissions to users and groups to create, manage, and monitor VMs.
- Create VMs from the OpenShift web console and the command line.
- Inspect and update the configuration of a VM.
- Access the console of a VM with the `virtctl` client.
- Monitor the resource consumption of a VM.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start accessing-review
```

## Instructions

The `database-admins` group must create and manage the development and staging versions of a MariaDB VM to gather performance metrics. The image for the MariaDB VMs exists within the `vm-images` namespace. Members of the `database-admins` group must be able to view all resources and edit any virtualization resources in the `vm-images` namespace.

As a member of the `database-admins` group, create the `dev-mariadb` and `staging-mariadb` VMs in their respective namespaces by using the `mariadb` PVC in the `vm-images` namespace. Grant the `developer` user access to view the performance metrics, and log in to the `dev-mariadb` VM. Finally, use the metrics to determine whether the VMs are experiencing resource contention.



### Note

If you cannot see the needed templates in the catalog, then log out and access the console as the `admin` user, and then log out again and access the console as the `dbadmin1` user.

1. Log in to the OpenShift web console as the `admin` user with `redhatocp` as the password. Add the `admin` role binding for all resources in the `development-db` and `staging-db` namespaces, and the `view` role binding for all resources in the `vm-images` namespace to the `database-admins` group. Then, add a `vm-admin` role binding to grant the

- `kubevirt.io:edit` cluster role to the `database-admins` group in the `vm-images` namespace.
2. Log in to the OpenShift web console as the `dbadmin1` user with `redhatocp` as the password. Create a VM with the **Red Hat Enterprise Linux 8 VM** template in the `development-db` namespace. Select the `mariadb` PVC in the `vm-images` namespace as the boot source. Name the VM `dev-mariadb` and assign 1 CPU and 2 GiB memory to the VM. Set the `rootdisk` to use the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class. Customize the VM to create the `developer` user with `developer` as the password.

**Note**

If you cannot see the **Red Hat Enterprise Linux 8 VM** template, then log out and access the console as the `admin` user, and then select the **Red Hat Enterprise Linux 8 VM** template. Log out again and access the console as the `dbadmin1` user.

3. Add the `view` role binding to the `developer` user in the `development-db` namespace.
4. Log in as the `developer` user with `developer` as the password, and use the `virtctl` client to open the console of the `dev-mariadb` VM.

In the `dev-mariadb` VM console, use the `watch vmstat -S M` command to retrieve the performance metrics that the VM's operating system reports. Recall that the `dev-mariadb` VM is configured with 1 CPU and 2 GiB (or 2048 MiB). The output of the `watch vmstat -S M` command includes the `swpd` column to report the used memory in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

5. As the `dbadmin2` user, use the `oc` client to create the `staging-mariadb` VM manifest in the `staging-db` namespace. Use the resource file at `~/D0316/labs/accessing-review/staging-mariadb.yaml` to create the VM.

Before deploying the VM, change the file as needed. Specify the `mariadb` PVC in the `vm-images` namespace as the source PVC. Modify the `cloudInitNoCloud` object to create a local account for the `dbadmin2` user with `redhatocp` as the password.

After creating the VM, wait a few moments and confirm that the VM is running.

6. Confirm that the `dbadmin2` user can access and log in to the `staging-mariadb` VM. Use the `oc` client to log in to the OpenShift cluster as the `dbadmin2` user with `redhatocp` as the password. Connect to the `staging-mariadb` VNC console with the `virtctl` client and log in as the `dbadmin2` user.

In the `staging-mariadb` console, use the `watch vmstat -S M` command to retrieve the performance metrics that the VM's operating system reports. Recall that the output of the `watch vmstat -S M` command includes the `swpd` column to report the used memory in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

7. The `developer` and `dbadmin2` users both reported performance issues with the `dev-mariadb` and `staging-mariadb` VMs. As the `admin` user, confirm that the VMs are experiencing performance issues. Use the **KubeVirt / Infrastructure Resources / Top Consumers** dashboard and PromQL queries to retrieve the current performance metrics of

both VMs for the last 5 minutes. Verify whether the VMs have sufficient available memory with the `memory_available_bytes` metric.

**Note**

The `~/D0316/labs/accessing-review/queries.txt` file contains the necessary queries. Use this file to copy and paste the PromQL queries as needed.

8. Both the `dev-mariadb` and the `staging-mariadb` VMs require an increase in their CPU and memory resources. As the `admin` user, use the OpenShift web console to increase the available CPU by 1. Increase the available memory by 2 GiB. Restart the VM to apply the changes.
9. As the `admin` user, modify the CPU and memory settings of the `staging-mariadb` VM. Increase the available CPU by 1. Increase the available memory by 2 GiB. Restart the VM to apply the changes.
10. As the `dbadmin1` user, confirm, with the `watch vmstat -S M` command, that the performance metrics improved for the `dev-mariadb` and `staging-mariadb` VMs. Use the `virtctl` client to connect to the VNC console of the VMs. In the console window of each VM, log in as the local user accounts to run the `watch vmstat -S M` command.
11. Now that performance testing is finished, as the `dbadmin1` user, stop the `dev-mariadb` and `staging-mariadb` VMs with the `virtctl` client.
12. Now that the `dbadmin1` user no longer needs the `dev-mariadb` VM, use the `oc delete` command to delete the VM. You can close any remaining command-line windows.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade accessing-review
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish accessing-review
```

## ► Solution

# Create, Configure, and Monitor Virtual Machines

Create virtual machines in different projects and retrieve metrics to monitor the resource consumption of the virtual machines.

### Outcomes

- Grant permissions to users and groups to create, manage, and monitor VMs.
- Create VMs from the OpenShift web console and the command line.
- Inspect and update the configuration of a VM.
- Access the console of a VM with the `virtctl` client.
- Monitor the resource consumption of a VM.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start accessing-review
```

### Instructions

The `database-admins` group must create and manage the development and staging versions of a MariaDB VM to gather performance metrics. The image for the MariaDB VMs exists within the `vm-images` namespace. Members of the `database-admins` group must be able to view all resources and edit any virtualization resources in the `vm-images` namespace.

As a member of the `database-admins` group, create the `dev-mariadb` and `staging-mariadb` VMs in their respective namespaces by using the `mariadb` PVC in the `vm-images` namespace. Grant the `developer` user access to view the performance metrics, and log in to the `dev-mariadb` VM. Finally, use the metrics to determine whether the VMs are experiencing resource contention.



#### Note

If you cannot see the needed templates in the catalog, then log out and access the console as the `admin` user, and then log out again and access the console as the `dbadmin1` user.

- Log in to the OpenShift web console as the `admin` user with `redhatocp` as the password. Add the `admin` role binding for all resources in the `development-db` and `staging-db` namespaces, and the `view` role binding for all resources in the `vm-images` namespace to the `database-admins` group. Then, add a `vm-admin` role binding to grant the

kubevirt.io:edit cluster role to the database-admins group in the vm-images namespace.

- 1.1. Use the command line to log in to your Red Hat OpenShift cluster as the **admin** user with **redhatocp** as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

- 1.2. Add the **admin** role binding on all resources in the **development-db** namespace. Confirm that the role binding is successfully applied to the namespace.

```
[student@workstation ~]$ oc create rolebinding admin \
--clusterrole=admin --group=database-admins -n development-db
rolebinding.rbac.authorization.k8s.io/edit created

[student@workstation ~]$ oc get rolebinding -n development-db
NAME          ROLE           AGE
admin         ClusterRole/admin   26s
system:deployers ClusterRole/system:deployer 4m
system:image-builders ClusterRole/system:image-builder 4m
system:image-pullers ClusterRole/system:image-puller 4m
```

- 1.3. Add the **admin** role binding on all resources in the **staging-db** namespace. Confirm that the role binding is successfully applied to the namespace.

```
[student@workstation ~]$ oc create rolebinding admin \
--clusterrole=admin --group=database-admins -n staging-db
rolebinding.rbac.authorization.k8s.io/edit created

[student@workstation ~]$ oc get rolebinding -n staging-db
NAME          ROLE           AGE
admin         ClusterRole/admin   20s
system:deployers ClusterRole/system:deployer 5m
system:image-builders ClusterRole/system:image-builder 5m
system:image-pullers ClusterRole/system:image-puller 5m
```

- 1.4. Add the **view** role binding on all resources in the **vm-images** namespace. Confirm that the role binding is successfully applied to the namespace.

```
[student@workstation ~]$ oc create rolebinding view \
--clusterrole=view --group=database-admins -n vm-images
rolebinding.rbac.authorization.k8s.io/admin created

[student@workstation ~]$ oc get rolebinding -n vm-images
NAME          ROLE           AGE
allow-clone-to-user ClusterRole/datavolume-cloner 6m
system:deployers ClusterRole/system:deployer 6m
system:image-builders ClusterRole/system:image-builder 6m
system:image-pullers ClusterRole/system:image-puller 6m
view          ClusterRole/view    25s
```

- 1.5. Add the `vm-admin` role binding for the `kubevirt.io:edit` cluster role to the `database-admins` group in the `vm-images` namespace. Confirm that the role binding is successfully applied to the namespace.

```
[student@workstation ~] oc create rolebinding vm-admin \
--clusterrole=kubevirt.io:edit --group=database-admins -n vm-images
rolebinding.rbac.authorization.k8s.io/vm-admin created
```

```
[student@workstation ~] oc get rolebinding -n vm-images
NAME          ROLE           AGE
system:deployers   ClusterRole/system:deployer   7m
system:image-builders   ClusterRole/system:image-builder   7m
system:image-pullers   ClusterRole/system:image-puller   7m
view            ClusterRole/view        47s
vm-admin        ClusterRole/kubevirt.io:edit    10s
```

2. Log in to the OpenShift web console as the `dbadmin1` user with `redhatocp` as the password. Create a VM with the **Red Hat Enterprise Linux 8** VM template in the `development-db` namespace. Select the `mariadb` PVC in the `vm-images` namespace as the boot source. Name the VM `dev-mariadb` and assign 1CPU and 2 GiB memory to the VM. Set the `rootdisk` to use the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class. Customize the VM to create the `developer` user with `developer` as the password.

**Note**

If you cannot see the **Red Hat Enterprise Linux 8** VM template, then log out and access the console as the `admin` user, and then select the **Red Hat Enterprise Linux 8** VM template. Log out again and access the console as the `dbadmin1` user.

- 2.1. Open a web browser and go to the web console URL.  
<https://console-openshift-console.apps.ocp4.example.com>  
Click `htpasswd_provider` and log in as the `dbadmin1` user with `redhatocp` as the password.
- 2.2. Change to the **Administrator** perspective and go to **Virtualization > Catalog**. Select the `development-db` namespace from the **Projects** list.
- 2.3. Click **Template catalog** and select the **Red Hat Enterprise Linux 8** VM template. Specify `dev-mariadb` as the VM name.
- 2.4. Select **PVC (clone PVC)** from the **Disk source** list. Select the `vm-images` namespace from the **PVC project** list. Select the `mariadb` PVC from the **PVC name** list.
- 2.5. Confirm that the network interface type is `masquerade` in the network interfaces section.
- 2.6. Click the **Customize VirtualMachine** button and then click the **Disks** tab. Confirm that the `cloudinitdisk` and `rootdisk` disks are attached.
- 2.7. At the end of the `rootdisk` row, click the vertical ellipsis icon `:` and select **Edit** from the menu. Change the **StorageClass** field to `ocs-external-storagecluster-ceph-rbd-virtualization`. Click **Save**.

- 2.8. Click the **Scripts** tab and then click **edit** next to **Cloud-init**. Update the cloud-init configuration to specify the **developer** user with **developer** as the password, and then click **Apply**.
  - 2.9. Review the settings and click **Create Virtual Machine**. Wait a few moments and confirm that the VM's status is **Running**. Log out of the OpenShift web console.
3. Add the **view** role binding to the **developer** user in the **development-db** namespace.
    - 3.1. Log in as the **dbadmin1** user with **redhatocp** as the password from the command line.

```
[student@workstation ~]$ oc login -u dbadmin1 -p redhatocp  
Login successful.
```

```
You have access to the following namespaces and can switch between them with 'oc  
project <projectname>':
```

```
* development-db  
  openshift-virtualization-os-images  
  staging-db  
  vm-images
```

```
Using project "development-db".
```

- 3.2. If necessary, switch to the **development-db** namespace.

```
[student@workstation ~]$ oc project development-db  
Now using project "development-db" on server "https://api.ocp4.example.com:6443"
```

- 3.3. Add the **view** role binding to the **developer** user.

```
[student@workstation ~]$ oc create rolebinding view \  
  --clusterrole=view --user=developer  
rolebinding.rbac.authorization.k8s.io/view created
```

4. Log in as the **developer** user with **developer** as the password, and use the **virtctl** client to open the console of the **dev-mariadb** VM.  
In the **dev-mariadb** VM console, use the **watch vmstat -S M** command to retrieve the performance metrics that the VM's operating system reports. Recall that the **dev-mariadb** VM is configured with 1 CPU and 2 GiB (or 2048 MiB). The output of the **watch vmstat -S M** command includes the **swpd** column to report the used memory in swap spaces. The output also includes the **free** column to report available free memory and the **wa** column to report CPU wait.

- 4.1. Log in as the **developer** user with **developer** as the password.

```
[student@workstation ~]$ oc login -u developer -p developer  
Login successful.
```

```
You have access to the following namespaces and can switch between them with 'oc  
project <projectname>':
```

**Chapter 2 |** Running and Accessing Virtual Machines

```
* development-db  
openshift-virtualization-os-images
```

Using project "development-db".

- 4.2. Use the `virtctl` client to open the VNC console of the `dev-mariadb` VM in the `development-db` namespace.

```
[student@workstation ~]$ virtctl vnc dev-mariadb -n development-db  
...output omitted...
```

- 4.3. In the new VNC console window, log in as the `developer` user with `developer` as the password. Within the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics that the operating system reports.

```
Red Hat Enterprise Linux 8.5 (ootpa)  
...output omitted...
```

```
dev-mariadb login: developer  
Password: developer  
[developer@dev-mariadb ~]$ watch vmstat -S M  
procs -----memory----- swap-- io--- system-- cpu----  
r b swpd free buff cache si so bi bo in cs us sy id wa st  
13 2 0 229 0 508 0 0 5447 170256 1190 550 20 34 40 4 0
```

The `wa` column reports a value above 0. The `free` column reports a low value of free memory. The results indicate potential issues with CPU wait and available free memory.

- 4.4. Press `Ctrl+C` to exit the `watch vmstat` command and use the `logout` command to log out of the `dev-mariadb` console.

```
[developer@dev-mariadb ~]$ logout
```

5. As the `dbadmin2` user, use the `oc` client to create the `staging-mariadb` VM manifest in the `staging-db` namespace. Use the resource file at `~/D0316/labs/accessing-review/staging-mariadb.yml` to create the VM.

Before deploying the VM, change the file as needed. Specify the `mariadb` PVC in the `vm-images` namespace as the source PVC. Modify the `cloudInitNoCloud` object to create a local account for the `dbadmin2` user with `redhatocp` as the password.

After creating the VM, wait a few moments and confirm that the VM is running.

- 5.1. From the command line, log in as the `dbadmin2` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u dbadmin2 -p redhatocp  
Login successful.  
...output omitted...
```

- 5.2. Open the `~/D0316/labs/accessing-review/staging-mariadb.yml` file in a text editor.

```
[student@workstation ~]$ vim ~/D0316/labs/accessing-review/staging-mariadb.yml
```

- 5.3. Locate the `.spec.dataVolumeTemplates[0].spec.source.pvc` object. Modify the file to specify `mariadb` as the PVC name and `vm-images` as the PVC namespace.

```
...output omitted...
name: staging-mariadb
namespace: staging-db
spec:
  dataVolumeTemplates:
    - apiVersion: cdi.kubevirt.io/v1beta1
      kind: DataVolume
      metadata:
        name: staging-mariadb
      spec:
        pvc:
          accessModes:
            - ReadWriteMany
          resources:
            requests:
              storage: 10Gi
          storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
          volumeMode: Block
        source:
          pvc:
            name: mariadb
            namespace: vm-images
...output omitted...
```

- 5.4. Locate the `spec.template.spec.volumes[1].cloudInitNoCloud.userData` object. Modify the `#cloud-config` object to create the `dbadmin2` user.

```
...output omitted...
volumes:
  - dataVolume:
      name: staging-mariadb
    name: staging-mariadb
  - cloudInitNoCloud:
      userData: |-  
        #cloud-config  
        user: dbadmin2  
        password: redhatocp  
        chpasswd: { expire: False }
...output omitted...
```

Confirm that all the `CHANGE_ME` entries are updated before saving and closing the file.

- 5.5. Use the `oc apply -f` command to create the `staging-mariadb` VM. Wait a few moments and confirm that the VM's status is `Running`.

```
[student@workstation ~]$ oc apply -f \
~/DO316/labs/accessing-review/staging-mariadb.yml
virtualmachine.kubevirt.io/staging-mariadb created

[student@workstation ~]$ oc get vm -n staging-db
NAME          AGE     STATUS    READY
staging-mariadb 47s    Running   True
```

6. Confirm that the dbadmin2 user can access and log in to the `staging-mariadb` VM. Use the `oc` client to log in to the OpenShift cluster as the `dbadmin2` user with `redhatocp` as the password. Connect to the `staging-mariadb` VNC console with the `virtctl` client and log in as the `dbadmin2` user.

In the `staging-mariadb` console, use the `watch vmstat -S M` command to retrieve the performance metrics that the VM's operating system reports. Recall that the output of the `watch vmstat -S M` command includes the `swpd` column to report the used memory in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

- 6.1. Log in as the `dbadmin2` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u dbadmin2 -p redhatocp
Login successful.

You have access to the following namespaces and can switch between them with 'oc
project <projectname>':
* development-db
  openshift-virtualization-os-images
  staging-db
  vm-images

Using project "development-db".
```

- 6.2. Switch to the `staging-db` namespace.

```
[student@workstation ~]$ oc project staging-db
Now using project "staging-db" on server "https://api.ocp4.example.com:6443".
```

- 6.3. Connect to the VNC console of the `staging-mariadb` VM with the `virtctl` client.

```
[student@workstation ~]$ virtctl vnc staging-mariadb
{"component": "", "level": "info", "msg": "--proxy-only is set to false, listening on\n127.0.0.1\\n", "pos": "vnc.go:117", "timestamp": "2024-11-01T018:39:01.195390Z"}\n...output omitted...
```



### Note

The VM takes several minutes to be completely available. Wait for it before moving to the next step.

- 6.4. Log in as the dbadmin2 user with redhatocp as the password in the staging-mariadb VNC console window.

```
Red Hat Enterprise Linux 8.5 (ootpa)
...output omitted...
```

```
staging-mariadb login: dbadmin2
Password: redhatocp
[dbadmin2@staging-mariadb ~]$
```

- 6.5. Within the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics that the VM's operating system report.

```
[dbadmin2@staging-mariadb ~]$ watch vmstat -S M
procs -----memory----- swap-- -----io---- -system- -----cpu-----
r b    swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
14 2      0     250     0     422     0     4  1481  52022   750  490 43 13 43  2  0
```

The `wa` column reports a value above 0. The `free` column reports a low value of free memory. The results show potential issues with CPU wait and available free memory.

- 6.6. Press `Ctrl+C` to exit the `watch vmstat` command and use the `logout` command to log out of the `staging-mariadb` console. Close the VNC window.

```
[dbadmin2@staging-mariadb ~]$ logout
```

7. The developer and dbadmin2 users both reported performance issues with the `dev-mariadb` and `staging-mariadb` VMs. As the `admin` user, confirm that the VMs are experiencing performance issues. Use the `KubeVirt / Infrastructure Resources / Top Consumers` dashboard and PromQL queries to retrieve the current performance metrics of both VMs for the last 5 minutes. Verify whether the VMs have sufficient available memory with the `memory_available_bytes` metric.



#### Note

The `~/D0316/labs/accessing-review/queries.txt` file contains the necessary queries. Use this file to copy and paste the PromQL queries as needed.

- 7.1. From the OpenShift web console, log in as the `admin` user with `redhatocp` as the password.
- 7.2. If necessary, switch to the `Administrator` perspective in the web console.
- 7.3. Go to `Observe > Dashboards`.
- 7.4. Select the `KubeVirt / Infrastructure Resources / Top Consumers` dashboard from the `Dashboard` menu. Select `Last 5 minutes` from the `Time Range` menu.

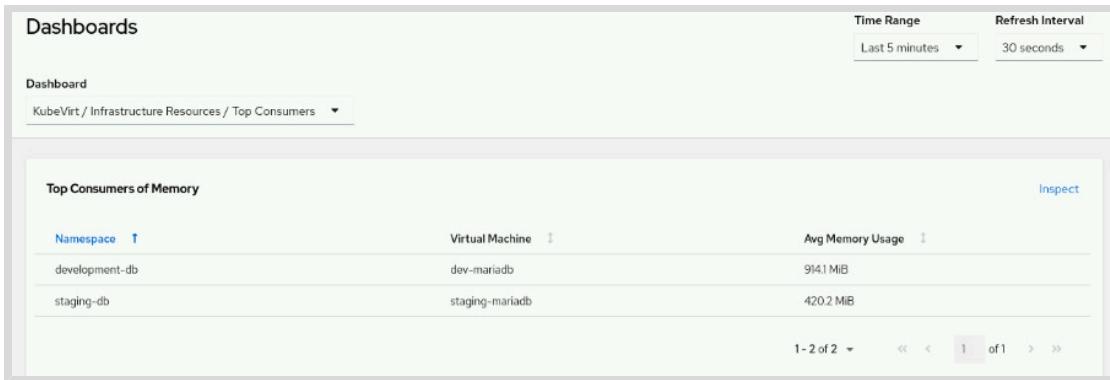


Figure 2.32: KubeVirt top consumers

The charts indicate that the dev-mariadb and staging-mariadb VMs are experiencing issues with memory and CPU.

- 7.5. Identify whether the VMs have available free memory with the `memory_available_bytes` metric. Go to **Observe > Metrics**. Enter the following query into the Expression field, and then click **Run queries** to execute the query:

```
topk(3, sum by (name, namespace)
      (rate(kubevirt_vmi_memory_available_bytes{namespace=~".*-db"}[5m])))
```

A returned value of 0 indicates that the VMs need more memory.

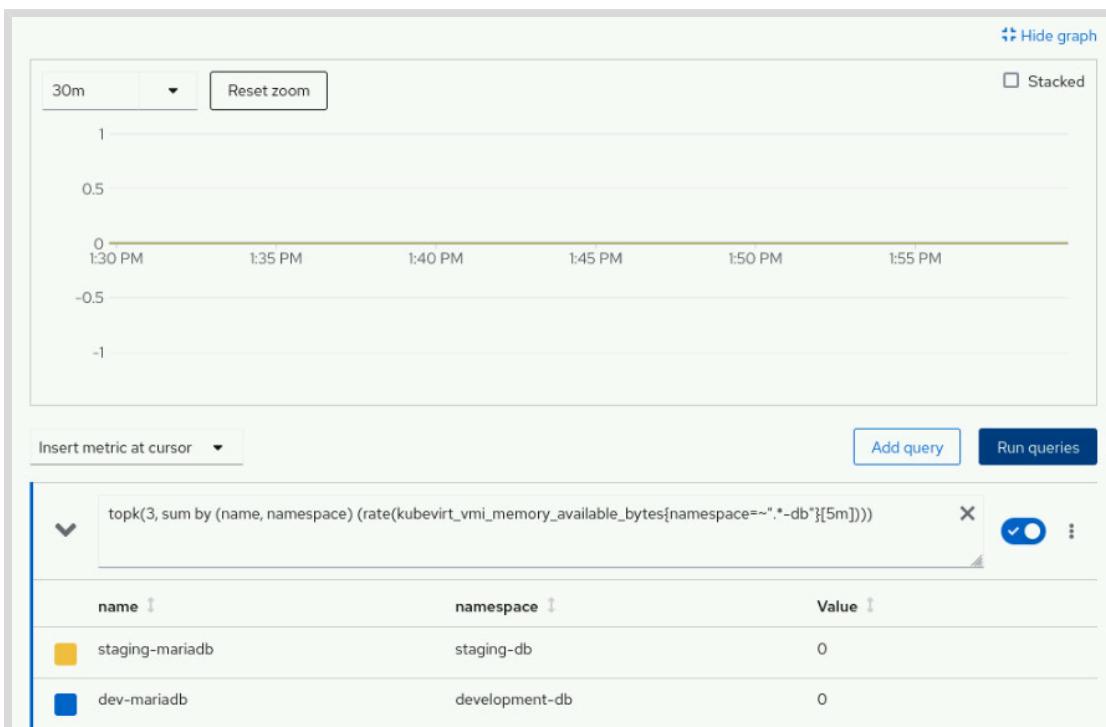


Figure 2.33: Available memory metrics

8. Both the dev-mariadb and the staging-mariadb VMs require an increase in their CPU and memory resources. As the `admin` user, use the OpenShift web console to increase the available CPU by 1. Increase the available memory by 2 GiB. Restart the VM to apply the changes.

- 8.1. In the OpenShift web console, go to **Virtualization > VirtualMachines**. Select the **development -db** namespace from the **Project** list.
  - 8.2. Click the **dev-mariadb** VM to open the **Virtual Machine Overview** window.
  - 8.3. Click the **Configuration** tab and scroll down to the **CPU | Memory** section.
  - 8.4. Click the **1CPU|2 GiB Memory** link to open the edit window.
  - 8.5. Enter 2 in the **CPU** field. Enter **4 GiB** in the **Memory** field and click **Save**.
  - 8.6. Click the **Action > Restart** option to restart the VM. Wait for the VM to reach the **Running** status.
9. As the **admin** user, modify the CPU and memory settings of the **staging-mariadb** VM. Increase the available CPU by 1. Increase the available memory by 2 GiB. Restart the VM to apply the changes.
    - 9.1. In the OpenShift web console, select the **staging-db** namespace from the **Project** list.
    - 9.2. Click the **staging-mariadb** VM to open the **Virtual Machine Overview** window.
    - 9.3. Click the **Configuration** tab and scroll down to the **CPU | Memory** section.
    - 9.4. Click the **1CPU|2 GiB Memory** link to open the edit window.
    - 9.5. Enter 2 in the **CPU** field. Enter **4 GiB** in the **Memory** field. Click **Save**.
    - 9.6. Click the **Action > Restart** option. Wait for the VM to reach the **Running** status.
  10. As the **dbadmin1** user, confirm, with the **watch vmstat -S M** command, that the performance metrics improved for the **dev-mariadb** and **staging-mariadb** VMs. Use the **virtctl** client to connect to the VNC console of the VMs. In the console window of each VM, log in as the local user accounts to run the **watch vmstat -S M** command.
    - 10.1. Open a new command-line window and log in to the cluster as the **dbadmin1** user with **redhatocp** as the password.

```
[student@workstation ~]$ oc login -u dbadmin1 -p redhatocp
Login successful.
...output omitted...
```

- 10.2. Use the **virtctl** client to open the VNC console of the **dev-mariadb** VM in the **development -db** namespace.

```
[student@workstation ~]$ virtctl vnc dev-mariadb -n development-db
...output omitted...
```

- 10.3. In the new VNC console window, log in as the **developer** user with **developer** as the password.
- 10.4. Within the VNC console window, use the **watch vmstat -S M** command to retrieve the VM's performance metrics that the operating system reports.

```
[developer@dev-mariadb ~]$ watch vmstat -S M
procs -----memory----- --swap-- -----io---- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
2 0 0 1315 3 2120 0 0 109 25068 540 22 17 35 48 0 0
```

The results reflect the impact of increasing the available memory and CPU resources.

- 10.5. Press **Ctrl+C** to exit the `watch vmstat` command and use the `logout` command to log out of the `dev-mariadb` console. Close the VNC window.

```
[developer@dev-mariadb ~]$ logout
```

- 10.6. In the command-line window, use the `virtctl` client to open the VNC console of the `staging-mariadb` VM in the `staging-db` namespace.

```
[student@workstation ~]$ virtctl vnc staging-mariadb -n staging-db
...output omitted...
```

- 10.7. In the new VNC console window, log in as the `dbadmin2` user with `redhatocp` as the password.
- 10.8. In the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics that the operating system reports.

```
[dbadmin2@staging-mariadb ~]$ watch vmstat -S M
procs -----memory----- --swap-- -----io---- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
1 0 0 1756 4 2482 0 0 423 23940 586 146 42 9 48 0 0
```

The results reflect the impact of increasing the available memory and CPU resources.

- 10.9. Press **Ctrl+C** to exit the `watch vmstat` command and use the `logout` command to log out of the `staging-mariadb` console. Close the VNC window.

```
[dbadmin2@staging-mariadb ~]$ logout
```

11. Now that performance testing is finished, as the `dbadmin1` user, stop the `dev-mariadb` and `staging-mariadb` VMs with the `virtctl` client.
  - 11.1. In the command-line window, use the `virtctl` client to stop the `dev-mariadb` VM in the `development-db` namespace. Wait a few moments and confirm that the VM is stopped.

```
[student@workstation ~]$ virtctl stop dev-mariadb -n development-db
VM dev-mariadb was scheduled to stop
```

```
[student@workstation ~]$ oc get vm -n development-db
NAME        AGE     STATUS      READY
dev-mariadb  25m    Stopped    False
```

- 11.2. Use the `virtctl` client to stop the `staging-mariadb` VM in the `staging-db` namespace. Wait a few moments and confirm that the VM is stopped.

```
[student@workstation ~]$ virtctl stop staging-mariadb -n staging-db
VM staging-mariadb was scheduled to stop

[student@workstation ~]$ oc get vm -n staging-db
NAME          AGE   STATUS    READY
stagingmariadb 27m   Stopped   False
```

12. Now that the dbadmin1 user no longer needs the dev-mariadb VM, use the `oc delete` command to delete the VM. You can close any remaining command-line windows.

- 12.1. Delete the dev-mariadb VM.

```
[student@workstation ~]$ oc delete vm/dev-mariadb -n development-db
virtualmachine.kubevirt.io "dev-mariadb" deleted
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade accessing-review
```

## Finish

As the student user on the workstation machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish accessing-review
```

# Summary

---

- In Red Hat OpenShift Virtualization, a VM object defines the template to create a VMI, which is the running instance of the VM inside your cluster.
- You can attach a persistent volume to a VMI by defining a PVC, to represent a request for a specific storage resource.
- To list all VMIs in a cluster, use the `oc get vmis` command.
- You can troubleshoot a VM by using the `virsh` command from within the VM's `virt-launcher` pod and its `libvirtd` container.
- Cluster administrators access the **Observe > Dashboard** interface to analyze VM metrics such as CPU utilization, memory utilization, and top VM resource consumers.
- Cluster administrators access the **Observe > Metrics** interface to run Prometheus Query Language (PromQL) queries that examine information about the cluster and user-defined workloads as a dashboard or graph.
- Cluster administrators can access all projects for monitoring, and developer users can access projects where they have at least `view` role permissions.
- Default templates enable a basic installation with useful settings, and you can make changes according to your VM's requirements.
- From a VM's management page, you can perform administrative operations, such as starting and stopping a VM, getting information about your VM, accessing the VM through a console, and adding new disks and network interfaces.
- Default templates include Red Hat Enterprise Linux, CentOS, Fedora, and Microsoft Windows Server. You can also install other operating systems, such as Ubuntu or openSUSE, by installing the KubeVirt common templates package in your OpenShift cluster.
- The `virtctl` tool is a client application for managing and connecting to VMs that are hosted in OpenShift or Kubernetes.
- You can access the VNC console of a VM by using the `virtctl vnc VM-name` command.
- The `remote-viewer` application is a remote desktop client to connect to a remote guest that supports the SPICE and VNC protocols.
- TigerVNC is a tool for graphical desktop sharing, for Linux, Windows, and MacOS.

## Chapter 3

# Configuring Kubernetes Networking for Virtual Machines

### Goal

Configure standard Kubernetes network objects and external access for virtual machines and virtual machine-backed applications.

### Sections

- Configuring Network Communication Between Virtual Machines Within the Cluster (and Guided Exercise)
- Restricting Access to Virtual Machines by Using a Network Policy (and Guided Exercise)
- Configuring External Access to Virtual Machines (and Guided Exercise)
- Configure External Access with Load Balancer Services (and Guided Exercise)
- Configure Kubernetes Networking for Virtual Machines

### Lab

# Configuring Network Communication Between Virtual Machines Within the Cluster

---

## Objectives

- Describe how virtual machines communicate on a Kubernetes software-defined network (SDN) and create a cluster-scoped virtual IP address for a virtual machine.

## Review the Kubernetes SDN

Kubernetes automatically assigns an IP address on the same subnet to every pod. Pods can communicate with each other even if they run on different cluster nodes or belong to different Kubernetes namespaces.

Kubernetes implements this infrastructure with the use of a *Software-Defined Network (SDN)*, which enables Kubernetes to control the network traffic and the network resources programmatically, similar to the VMware vSwitch software program.

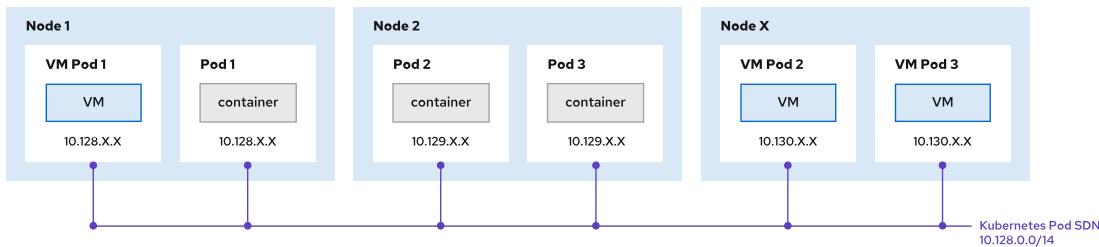
## Introduction to the Cluster Network Operator

Red Hat OpenShift uses the Cluster Network Operator (CNO) for managing the SDN. The operator calls a plug-in that adheres to the Container Network Interface (CNI) specification to configure container network interfaces.

Red Hat OpenShift includes some plug-in providers such as OVN-Kubernetes and Kuryr. The OVN-Kubernetes provider, which runs the Open vSwitch (OVS) plug-in on each node, is selected by default during installation.

## Expose Pods and Virtual Machines by Using Services

The following schema shows that the Kubernetes SDN connects all the pods to a shared network.



**Figure 3.1: Kubernetes pod SDN**

Pods are constantly created and deleted across the nodes in the cluster. For example, when you deploy a new version of your application, Kubernetes destroys the old pods and deploys new ones. When a node goes into maintenance, Kubernetes destroys its pods, and starts new ones on the remaining nodes. Because Kubernetes allocates a unique IP address for each pod that it creates, addressing pods can be challenging.

You can use Kubernetes services to provide a single and unique IP address for other pods to use, independently of where the pods are running. Because VMs run within a pod, VMs can also use services.

### Chapter 3 | Configuring Kubernetes Networking for Virtual Machines

Services use labels to enable selectors to indicate which pods receive the traffic through the service. Kubernetes adds each pod that matches these selectors to the service resource as an endpoint. When pods are created and destroyed, the service automatically updates the endpoints and load balances client requests across member pods.

Kubernetes uses one subnet for pods and one subnet for services, and when you address the service IP, Kubernetes forwards the traffic transparently to the pods.

The following diagram shows three pods that run the API of an application. The pods are not all running on the same node. The `service1` service balances the load between the pods, and the `service2` service forwards the requests to a VM.

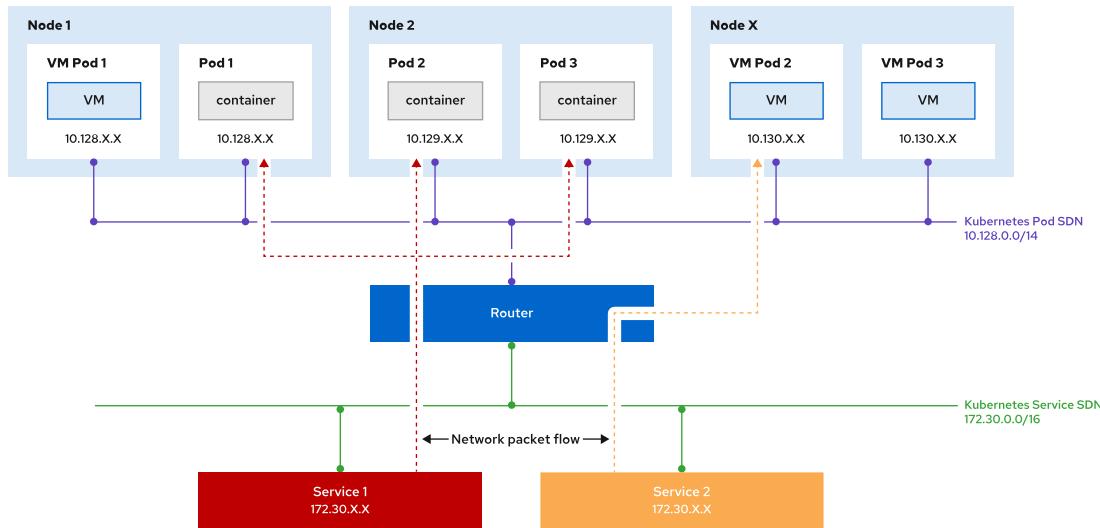


Figure 3.2: Pod and service subnets

During installation, you can configure the address range of each network. Run the `oc get network/cluster -o yaml` command to list the ranges that your cluster is using.

```
[user@host ~]$ oc get network/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: Network
metadata:
...output omitted...
spec:
  clusterNetwork:
    - cidr: 10.8.0.0/14 ①
      hostPrefix: 23
    externalIP:
      policy: {}
    networkType: OVNKubernetes
    serviceNetwork:
      - 172.30.0.0/16 ②
...output omitted...
```

① The pod network CIDR

② The service network CIDR

Kubernetes provides several types of services to expose the VM outside the cluster, such as the `ClusterIP` type, which assigns an internal IP address to the service but is accessible only from

inside the cluster. Red Hat OpenShift provides a route mechanism to expose the service outside the cluster. This course explains routes elsewhere.

## Address a Service by its DNS Record

The internal DNS service responds to queries from applications that are deployed in Kubernetes, to find the IP address of a service.

The DNS Operator deploys and runs a DNS server, which monitors the services to automatically create and update the DNS records. The DNS Operator manages the `svc.cluster.local` domain name for services, and creates records in the `servicename.namespace.svc.cluster.local` format. The operator automatically creates the `/etc/resolv.conf` configuration file inside the pods so that name resolution is immediately available without further configuration.

The following example retrieves the IP address of the backend service in `prod2` namespace by querying the `backend.prod2.svc.cluster.local` record:

```
[user@host ~]# getent hosts backend.prod2.svc.cluster.local
172.30.123.204 backend.prod2.svc.cluster.local
```

## Create Services for Virtual Machines

VMs in Kubernetes run inside `virt-launcher` pods. These pods get an IP address on the pod SDN, which you can use to create a Kubernetes service to access them through a fixed IP address on the service SDN, which is different from the IP address of the VM.

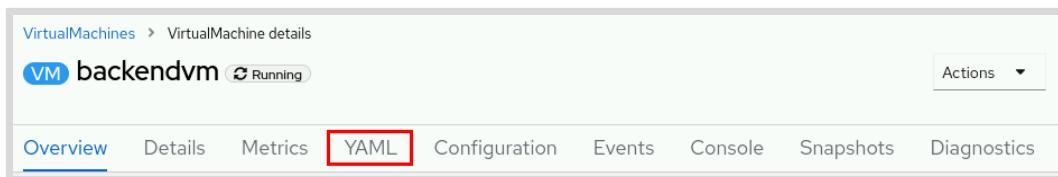
The `virt-launcher` process that runs inside the pod runs a Dynamic Host Configuration Protocol (DHCP) server, which provides an IP address and the DNS configuration to the VM. The `virt-launcher` pod redirects the inbound traffic to the VM and routes the outbound traffic to its destination.

You can create a Kubernetes service to expose a VM inside the cluster. After you expose the VM, pods and other VMs inside the cluster can access the application that the VM hosts.

## Prepare a Label for a Service

To expose a VM, you must add a label to the `VirtualMachine` resource, and create a service with a selector that specifies the same label that is added to the VM.

To add a label to a VM from the web console, go to `Virtualization > VirtualMachines`, select the VM, and then go to the `YAML` tab.



**Figure 3.3: YAML manifest of a VM**

The following `VirtualMachine` resource definition shows the `tier: backend` label that a Kubernetes service uses to select the `virt-launcher` pod.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
labels:
  app: backendvm
...output omitted...
  name: backendvm
spec:
  dataVolumeTemplates:
...output omitted...
  template:
    metadata:
      creationTimestamp: null
      labels:
        tier: backend
...output omitted...
```



### Note

Add the label to the `labels` section in the `.spec.template.metadata.labels` path. This change ensures that the label is set to the `virt-launcher` pod.

The resource definition includes several `labels` sections for different parts of the VM configuration. Kubernetes does not use these other labels to identify pods for services.

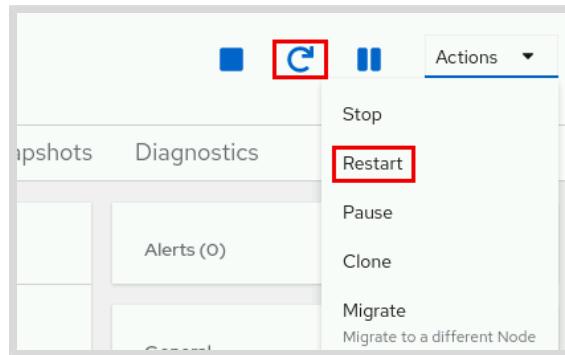
To add a label from the command line, use the `oc edit vm vmname` command to edit the virtual machine, and save the changes.

```
[user@host ~]$ oc edit vm backendvm
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
labels:
  app: backendvm
...output omitted...
  name: backendvm
spec:
  dataVolumeTemplates:
...output omitted...
  template:
    metadata:
      creationTimestamp: null
      labels:
        tier: backend
...output omitted...
```

When you edit a `VirtualMachine` resource manifest, Kubernetes does not automatically propagate your changes to the `VirtualMachineInstance` and `virt-launcher` pod resources. You can restart the VM to re-create the resources and apply the latest changes.

Kubernetes re-creates the virtual machine instance (VMI) and the `virt-launcher` pod resources with the latest details from the `VirtualMachine` resource.

To restart a VM from the OpenShift web console, go to **Virtualization > VirtualMachines**, select the VM, and then click **Actions > Restart**. Alternatively, use the restart icon.



**Figure 3.4: Restart a VM**

You can use the `virtctl` client to restart a VM from the command line. The following example uses the `virtctl restart vmname` command to restart the backup VM.

```
[user@host ~]$ virtctl restart backendvm
VM backendvm was scheduled to restart
```

Use the `oc get vm` command to confirm that the VM is running:

```
[user@host ~]$ oc get vm
NAME      AGE     PHASE     IP          NODENAME    READY
backendvm  23s     Running   10.11.0.22  worker02   True
```

You can also manually set the tier: `backend` label to the `virt-launcher` pod. From the command line, use the `oc label pod` command:

```
[user@host ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
virt-launcher-backendvm-frxbj  1/1     Running   0          4m34s

[user@host ~]$ oc label pod virt-launcher-backendvm-frxbj tier=backend
pod/virt-launcher-backendvm-frxbj labeled
```

To manually set the tier: `backend` label from the web console, go to **Workloads > Pods**, select the `virt-launcher` pod, and then go to the **Details** tab.

The screenshot shows the 'Pods' section with a link to 'Pod details'. Below is the pod name 'virt-launcher-backendvm-frxbj' with a 'Running' status. A red box highlights the 'Edit' button next to the 'Labels' section, which contains the label 'kubevirt.io=virt-launcher'.

Name	Status
virt-launcher-backendvm-frxbj	Running

Namespace	Restart policy
test-vms	Never restart

Labels	Active deadline seconds
kubevirt.io=virt-launcher	Not configured

Figure 3.5: Edit the labels of the virt-launcher pod

In the Labels section, click Edit and then add the label.

The dialog title is 'Edit labels'. It explains that labels help organize and select resources. Below is a list of existing labels for the pod, followed by an input field where a new label can be added. A red box highlights the input field containing 'tier=backend'.

Labels for virt-launcher-backendvm-frxbj

- kubevirt.io=virt-launcher
- kubevirt.io/created-by=6169ef24-aa6a-4148-afc9-ddad56ee6e89
- kubevirt.io/domain=centos-test
- kubevirt.io/nodeName=worker01
- kubevirt.io/size=small
- vm.kubevirt.io/name=backendvm

Cancel Save

Figure 3.6: Add a label to the virt-launcher pod

**Warning**

Even though you set the label of the `virt-launcher` pod, you must still define the same label at the `VirtualMachine` resource level.

When you restart a VM, Kubernetes destroys the VMI and the `virt-launcher` pod resources and then re-creates them from the `VirtualMachine` resource. If you do not also associate the label with the `VirtualMachine` resource, then the label that you set to the `virt-launcher` pod is lost.

## Configure a Service for a Virtual Machine

To create a service from the web console, go to **Networking > Services**, click **Create Service**, and then use the YAML editor to declare the service.

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: example
5    namespace: creating-accessing-vms
6  spec:
7    selector:
8      app: MyApp
9    ports:
10      - protocol: TCP
11        port: 80
12        targetPort: 9376
13

```

**Figure 3.7: YAML manifest of a service**

After you create the service, the web console displays the IP address.

You can use the following manifest to create a service from the command line:

```

apiVersion: v1
kind: Service
metadata:
  name: backend ①
  namespace: prod2 ②
spec:
  type: ClusterIP
  selector:

```

```
tier: backend ③  
ports:  
  - protocol: TCP ④  
    port: 80  
    targetPort: 8080
```

- ① The name of the service that you create. The DNS Operator creates a record for that service name.
- ② The namespace that hosts the VM.
- ③ The label that matches the label that you define in the `VirtualMachine` resource.
- ④ The service listens on port 80/TCP and forwards the requests to the back-end VM on port 8080.

Use the `oc create -f service_file.yaml` command to create a service from a YAML manifest:

```
[user@host ~]$ oc create -f service_file.yaml  
service/backend created
```

Confirm the creation of the service and notice the IP address that is assigned:

```
[user@host ~]$ oc get svc  
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE  
backend   ClusterIP  172.30.123.204  <none>          80/TCP       2m57s
```

The `virtctl` tool also provides a way to create a service from the command line. Use the `virtctl expose vmi vmi-name` command to create the service. The `virtctl expose vmi vmi-name` command uses all the VM labels to create a selector for the service.

```
[user@host ~]$ virtctl expose vmi backendvm --name backend \  
--type ClusterIP --port 80 --target-port 8080
```

For each pod that matches the selector, Kubernetes automatically creates an endpoint resource.

A VMI must match all the labels that a selector uses for the service to include the VMI. Although using all the labels of a VM to create a service is convenient for creating new services, it might not be ideal. If a label on the VMI changes, then the VMI is no longer included in the services.

To confirm that the service points to the VM, you can compare the IP address in the endpoint resource with the IP address in the `VirtualMachineInstance` resource:

```
[user@host ~]$ oc get endpoints  
NAME      ENDPOINTS      AGE  
backend   10.128.3.135:8080  3m13s  
  
[user@host ~]$ oc get vmi  
NAME      AGE      PHASE      IP      NODENAME      READY  
backendvm  13m     Running    10.128.3.135  prodnode1  True
```

After you create the service, the DNS Operator adds the `backend.prod2.svc.cluster.local` record with the `172.30.123.204` address. You can then access the application that is running on the VM from another pod or another VM by using the `backend.prod2.svc.cluster.local` DNS name. Because the `/etc/resolv.conf` file that the DNS Operator deploys on the pods defines the `svc.cluster.local` and `prod2.svc.cluster.local` search domains, you can also use the `backend.prod2` or `backend` short names to access the application.

The following example starts a temporary test pod and performs DNS queries for the service name:

```
[user@host ~]$ oc run mytestnet -it --rm --image=rhel8/toolbox
If you don't see a command prompt, try pressing enter.

[user@mytestnet /]# getent hosts backend.prod2.svc.cluster.local
172.30.123.204  backend.prod2.svc.cluster.local

[user@mytestnet /]# getent hosts backend.prod2
172.30.123.204  backend.prod2.svc.cluster.local

[user@mytestnet /]# getent hosts backend
172.30.123.204  backend.prod2.svc.cluster.local
```

## Configure a Headless Service for a Virtual Machine

A service provides a DNS name to a stable IP address. A service also provides load balancing to all the pods that are connected to the service. However, for a client to connect directly to a single pod, use a **Headless Service**. A headless service creates a DNS entry that ties a DNS name to the IP address of a pod. A headless service enables a client to connect directly to whichever pod it prefers. A cluster IP is not allocated; kube-proxy does not handle the service; and no load balancing or proxying occurs from the platform.

To create a headless service, set the `spec.clusterIP` field of the manifest to a value of `None`.

```
apiVersion: v1
kind: Service
metadata:
  name: mysubdomain ①
  namespace: prod2 ②
spec:
  type: ClusterIP
  selector:
    tier: backend ③
  clusterIP: None ④
  ports:
    - protocol: TCP
      port: 1234
      targetPort: 1234
```

- ① The name of the service must match the `spec.subdomain` attribute in the `VirtualMachine` resource. The DNS Operator creates a record for that service name.
- ② The namespace that hosts the VM.

- ③ The selector attribute that matches the label that you define in the `VirtualMachine` resource.
- ④ The value of `None` specifies a headless service.
- ⑤ The list of ports that the service exposes must define at least one port.



## References

For more information, refer to the *Cluster Network Operator in OpenShift Container Platform* and *DNS Operator in OpenShift Container Platform* in the *Red Hat OpenShift Container Platform Networking* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index)

For more information about VM networking, refer to the *Virtual Machine Networking* section in the *Red Hat OpenShift Container Platform Virtualization* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virtual-machine-networking](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virtual-machine-networking)

For more information about headless services, refer to the *Accessing a Virtual Machine by Its Internal FQDN* section in the *Red Hat OpenShift Container Platform Virtualization* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-accessing-vm-internal-fqdn](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-accessing-vm-internal-fqdn)

For more information about service resources, refer to the *Service* section in the *Kubernetes* documentation at  
<https://kubernetes.io/docs/concepts/services-networking/service/>

For more information about Headless Services, refer to the *Headless Services* section in the *Kubernetes* documentation at  
<https://kubernetes.io/docs/concepts/services-networking/service/#headless-services>

For more information about DNS records, refer to the *DNS Records* section in the *KubeVirt* user guide at  
<https://kubevirt.io/user-guide/network/dns/>

## ► Guided Exercise

# Configure Networking Communication Between Virtual Machines

Configure a Kuberentes service that establishes a connection between a virtual machine that is running a MariaDB server and a virtual machine that is running a database client.

## Outcomes

- Create a service of the ClusterIP type.
- Link the service to a virtual machine.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. This command also creates the `network-services` namespace and starts two virtual machines in that namespace: the `mariadb-server` VM, which hosts a MariaDB database, and the `mariadb-client` VM, which provides the client `mysql` command-line tool.

```
[student@workstation ~]$ lab start network-services
```

## Instructions

- 1. As the `admin` user, go to the Red Hat OpenShift web console, and confirm that the `mariadb-server` VM and the `mariadb-client` VM are running.
- 1.1. From a command line, log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

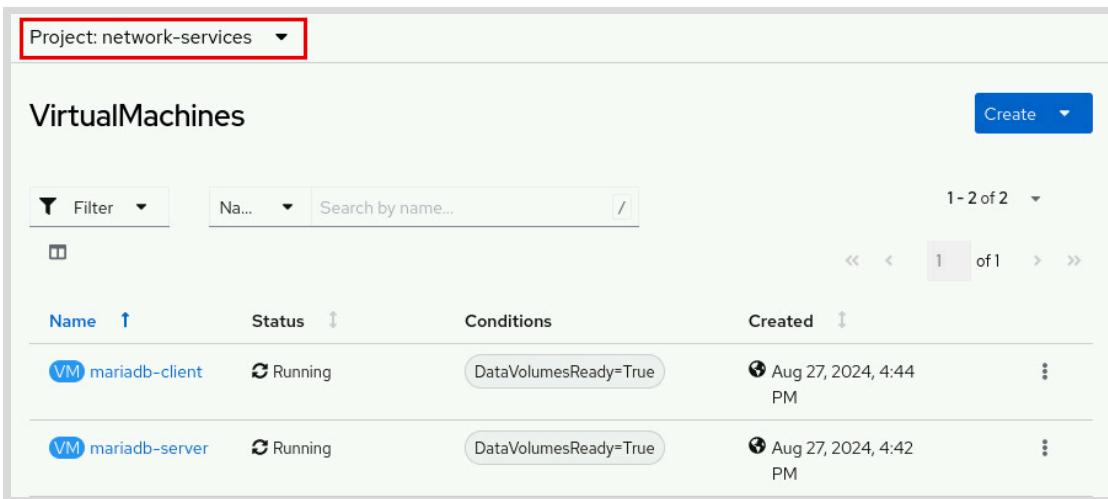
- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Set the `network-services` project as the active project.

```
[student@workstation ~]$ oc project network-services
Now using project "network-services" on server ...
```

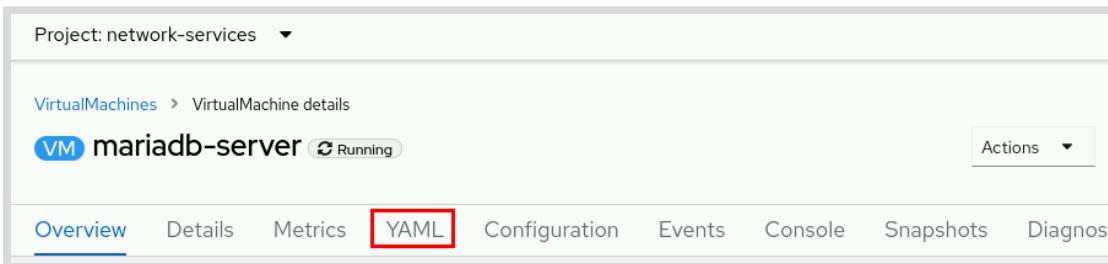
14. Open a web browser and go to the web console URL.  
<https://console-openshift-console.apps.ocp4.example.com>.  
Select **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.
15. Select the **Administrator** perspective. Go to **Virtualization > VirtualMachines** and then select the **network-services** project. Confirm that the **mariadb-client** and **mariadb-server** VMs are running.



The screenshot shows the OpenShift web console interface. At the top, a dropdown menu is set to "Project: network-services". Below it, the title "VirtualMachines" is displayed, along with a "Create" button. A search bar and filter options are available. The main area lists two virtual machines:

Name	Status	Conditions	Created	Actions
VM mariadb-client	Running	DataVolumesReady=True	Aug 27, 2024, 4:44 PM	...
VM mariadb-server	Running	DataVolumesReady=True	Aug 27, 2024, 4:42 PM	...

- ▶ 2. Add the `app=mariadb-server` label to the `mariadb-server` VM. You use this label later in the exercise.
- 2.1. From the web console, go to **Virtualization > VirtualMachines**, select the `mariadb-server` VM, and then go to the **YAML** tab to access the YAML editor.



The screenshot shows the "VirtualMachine details" page for the "mariadb-server" VM. The top navigation bar shows "VirtualMachines > VirtualMachine details". The VM name "mariadb-server" is listed with its status as "Running". Below the navigation bar, there are tabs: Overview (selected), Details, Metrics, **YAML** (highlighted with a red box), Configuration, Events, Console, Snapshots, and Diagnose.

- 2.2. Add the `app=mariadb-server` label in the `.spec.template.metadata.labels` path. Click **Save**

```
...output omitted...
spec:
  dataVolumeTemplates:
    ...output omitted...
  runStrategy: RerunOnFailure
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: mariadb-server
        flavor.template.kubevirt.io/small: "true"
```

```
kubevirt.io/domain: mariadb-server
kubevirt.io/size: small
...output omitted...
```

- 2.3. Restart the VM so that Kubernetes re-creates an instance that includes the new label. From the web console, click **Actions** > **Restart**.

The screenshot shows the OpenShift web console interface for a VM named 'mariadb-server'. The 'Actions' dropdown menu is open, and the 'Restart' option is highlighted with a red box. Other options in the menu include Stop, Pause, Clone, Migrate, Copy SSH command, Edit labels, and Edit annotations.

- 2.4. From the command line, use the `watch oc get vmi,pods` command to confirm that Kubernetes re-creates the VMI resource and the virt-launcher pod. Use the `-l` option to filter the results and confirm that the label is now available on the VMI. Press `Ctrl+C` to exit the `watch oc get` command.

```
[student@workstation ~]$ watch oc get vmi,pods -l app=mariadb-server
...output omitted...

NAME                                     AGE   PHASE
virtualmachineinstance.kubevirt.io/mariadb-server  35s  Running ...
...output omitted...

NAME          READY   STATUS    RESTARTS   AGE
pod/virt-launcher-mariadb-server-gtkln  1/1     Running   0          35s
```

- 3. Create a service named `mariadb` with the `ClusterIP` type. Use the `app=mariadb-server` label for the selector. The service must listen on TCP port 3306 and forward the traffic to the VM on that port.

- 3.1. From the command line, use the `virtctl expose` command to create the service.

```
[student@workstation ~]$ virtctl expose vmi mariadb-server --name mariadb \
--type ClusterIP --port 3306 --target-port 3306
Service mariadb successfully exposed for vmi mariadb-server
```

- 3.2. View the resource manifest for the `mariadb` service to verify that the selector uses the `app=mariadb-server` label. Save the change.

```
[student@workstation ~]$ oc get service mariadb -o yaml
...output omitted...
selector:
  app: mariadb-server ①
  flavor.template.kubevirt.io/small: "true"
  kubevirt.io/domain: mariadb-server
  kubevirt.io/size: small
  os.template.kubevirt.io/rhel8.4: "true"
  workload.template.kubevirt.io/server: "true"
  sessionAffinity: None
  type: ClusterIP
...output omitted...
```

- ① The `virtctl expose` command adds all the labels of the VM to the selector. You can create a service with this single selector to achieve the same results.

- 3.3. Confirm that the `mariadb` server has the `mariadb-server` VMI as an active endpoint.

```
[student@workstation ~]$ oc get service,vmi,endpoints
NAME          TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)     AGE
service/mariadb  ClusterIP  172.30.88.34 <none>        3306/TCP   3m49s

NAME                           ... PHASE      IP           ...
virtualmachineinstance.kubevirt.io/mariadb-client ... Running   10.8.2.45   ...
virtualmachineinstance.kubevirt.io/mariadb-server ... Running   10.8.2.66   ...

NAME          ENDPOINTS      AGE
endpoints/mariadb  10.8.2.66:3306  3m49s
```

By using this configuration, the DNS operator creates the `mariadb.network-services.svc.cluster.local` record.

- ▶ 4. Log in to the `mariadb-client` VM, and then verify database connectivity.
- 4.1. From the command line, use the `virtctl console` command to access the console of the `mariadb-client` VM. Press `Enter` to access the login prompt. If the serial console disconnects before you finish, then repeat this command to log in again.

```
[student@workstation ~]$ virtctl console mariadb-client
Successfully connected to mariadb-client console. The escape sequence is ^]

mariadb-client login:
```

**Note**

You can also access the VM console from the web console. From the web console, go to **Virtualization > VirtualMachines**, select the **mariadb-client** VM, and then select the **Console** tab.

Project: network-services ▾

VirtualMachines > VirtualMachine details

**VM mariadb-client** ⚡ Running Actions ▾

Overview Details Metrics YAML Configuration Events **Console** Snapshots Diagnostics

**Console**

Guest login credentials ▾

VNC console ▾ Send key ▾ Paste

```
Red Hat Enterprise Linux 8.5 (Ootpa)
Kernel 4.18.0-348.el8.x86_64 on an x86_64
Activate the web console with: systemctl enable --now cockpit.socket
mariadb-client login:
```

4.2. Log in as the developer user with developer as the password.

```
mariadb-client login: developer
Password:
[developer@mariadb-client ~]$
```

4.3. Connect to the sakila database that is running on the mariadb-server VM. Use the DNS service name to access the database and log in as the devuser user with developer as the password.

```
[developer@mariadb-client ~]$ mysql -u devuser -p'developer' \
-h mariadb.network-services.svc.cluster.local sakila
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(sakila)]>
```

4.4. List the tables of the sakila database, and log out.

```
MariaDB [sakila]> SHOW TABLES;
+-----+
| Tables_in_sakila |
+-----+
| actor           |
```

```
...output omitted...
```

```
MariaDB [sakila]> quit
Bye
[developer@mariadb-client ~]$
```

- 4.5. Inspect and confirm that the DHCP server that is running inside the `virt-launcher` pod configures the VM system to use the short service name.

Review the search line in the `/etc/resolv.conf` file.

With that configuration, you can refer to the `mariadb` service as `mariadb`, `mariadb.network-services`, or `mariadb.network-services.svc`.

```
[developer@mariadb-client ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
search network-services.svc.cluster.local svc.cluster.local cluster.local
  ocp4.example.com srv.example.com
nameserver 172.30.0.10
[developer@mariadb-client ~]$
```

- 4.6. Access the `sakila` database by using the `mariadb` host. Log out of the database after confirming that you can use the short name.

```
[developer@mariadb-client ~]$ mysql -u devuser -p'developer' \
-h mariadb sakila
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...

MariaDB [sakila]> quit
Bye

[developer@mariadb-client ~]$
```

Press `Crtl+]` to escape the console.

- 5. Verify that you can access the database that is running on the `mariadb-server` VM from any namespace. Start an interactive pod by using the `registry.ocp4.example.com:8443/redhattraining/mariadb:10.5` container image, which provides the `mysql` client command.

- 5.1. From the command line, create a project.

```
[student@workstation ~]$ oc new-project test
Now using project "test" on server "https://api.ocp4.example.com:6443".
...
```

- 5.2. Run a temporary interactive pod in the `test` namespace. The pod uses the `mariadb:10.5` container image, which provides the `mysql` command.

**Note**

It is safe to ignore pod security warnings when using a cluster `admin` user that creates unmanaged pods. The `admin` user can create privileged pods that the Security Context Constraints controller does not manage.

```
[student@workstation ~]$ oc run testdb -it --rm \
--image=registry.ocp4.example.com:8443/redhattraining/mariadb:10.5 \
--command -- mysql --connect-timeout=5 -u devuser -p'developer' \
-h mariadb.network-services.svc.cluster.local sakila
...output omitted...
If you don't see a command prompt, try pressing enter.

MariaDB [(sakila)]>
```

**Note**

You can copy and paste the command from the `~/D0316/labs/network-services/commands.txt` file that the `lab` command prepared for you.

- 5.3. Execute an SQL query to verify connectivity. When done, log out of the database.

The test confirms that you can access the service from any namespace.

```
MariaDB [sakila]> SHOW TABLES;
+-----+
| Tables_in_sakila |
+-----+
| actor           |
...output omitted...

MariaDB [sakila]> quit
Bye
Session ended, resume using 'oc attach testdb -c testdb -i -t' command when the
pod is running
pod "testdb" deleted
[student@workstation ~]$
```

- 5.4. Delete the test project.

```
[student@workstation ~]$ oc delete project test
project.project.openshift.io "test" deleted
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-services
```



# Restricting Access to Virtual Machines by Using a Network Policy

---

## Objectives

- Control network access for virtual machines by using a network policy.

## Configure Network Policies for Services

By default, all pods in a namespace are accessible from any pods in any other namespace. For example, a pod or a VM that runs in a namespace can connect to a pod or a VM that runs in another namespace. However, to restrict connections between pods, you must establish stricter rules.

With Kubernetes network policies, you can configure isolation policies for pods or VMs. You can use labels on pods and namespaces to identify the targeted resources of the network policies.

Network policies use labels on namespace resources to specify a target namespace. Administrators might use the command line or the web console to add labels to a namespace to prepare the namespace to use a network policy. From the OpenShift web console, go to **Administration > Namespaces**, select the namespace, and then go to the **Details** tab.

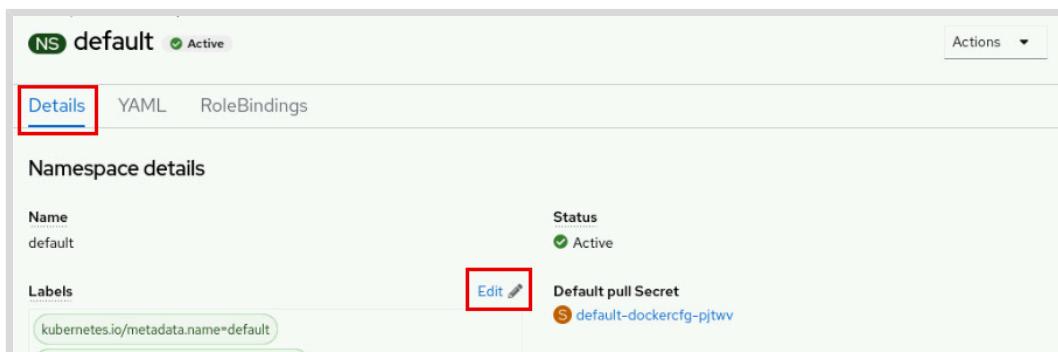
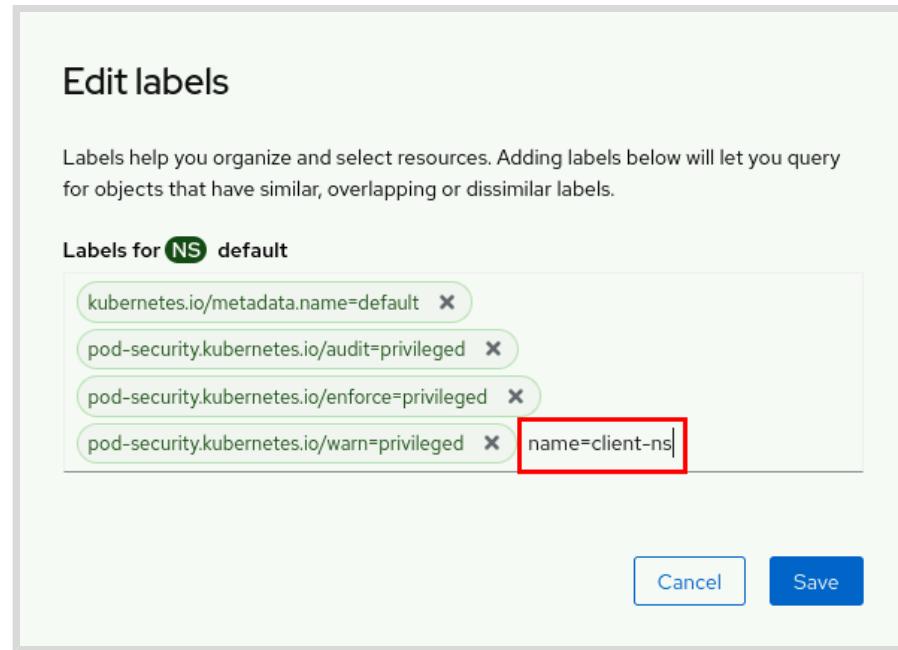


Figure 3.12: Edit the labels of a namespace

In the **Labels** section, click **Edit** and then add the label.



**Figure 3.13: Add a label to a namespace**

To add a label to a namespace from the command line, use the `oc label namespace` command. The following command adds the `name=client-ns` label to the `prod-front` namespace:

```
[user@host ~]$ oc label namespace prod-front name=client-ns
```

Network policies are namespaced resources. The following example defines a network policy for all pods with the `tier=backend` label in the current namespace. The policy enables ingress traffic to port 8080 from pods and VMs whose label is `tier=front` in the namespace with the `name=client-ns` label. The second rule in the list enables any pod or VM in the namespace with the `name=server-ns` label to connect on port 8080.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: network-1-policy
spec:
  podSelector: ①
  matchLabels:
    tier: backend

  ingress: ②
  - from: ③
    - namespaceSelector: ④
      matchLabels:
        name: client-ns
    podSelector:
      matchLabels:
        tier: front
  - namespaceSelector: ⑤
    matchLabels:
```

```
name: server-ns
ports: ⑥
- port: 8080
  protocol: TCP
```

- ① The top-level `podSelector` field is required and defines which pods and VMs in the current namespace the network policy applies to. If the `podSelector` field is empty, then the policy applies to all the pods and VMs in the current namespace.
- ② The `ingress` field lists the ingress traffic rules.
- ③ The `from` field lists the allowed sources. The incoming traffic is allowed if any of the rules match (logical OR).
- ④ The first rule allows traffic from namespaces with the `name=client-ns` label and from pods and VMs with the `tier=front` label.
- ⑤ The second rule enables incoming traffic for all pods and VMs in the namespaces with the `name=server-ns` label.
- ⑥ The `ports` field lists the destination ports that allow traffic to reach the selected pods.

To create a network policy from the web console, go to **Networking > NetworkPolicies**, click **Create Network Policy**, and complete the form. The form provides links for adding ingress rules, egress rules, namespace selectors, pod selectors, and ports.



### Important

In clusters with Red Hat OpenShift 4.16.2, the form does not pick up the currently selected namespace. However, the form is still useful for constructing the network policy. To mitigate the namespace issue, switch to the YAML view to add the correct namespace to the resource manifest. Then, click **Create**.

**Create NetworkPolicy**

Create by completing the form.

Configure via:  Form view  YAML view

**Policy name \***  
my-policy

**Pod selector**  
If no pod selector is provided, the policy will apply to all pods in the namespace.  
[+ Add pod selector](#)  
Show a preview of the affected pods that this policy will apply to

**Policy type**

Figure 3.14: Form view of a network policy

Use the YAML view to look at the resource manifest that the Form view constructed. You can also click YAML view to use the YAML editor to create the resource.

The screenshot shows a 'Create NetworkPolicy' interface. On the left, there's a code editor with the following YAML manifest:

```
1 kind: NetworkPolicy
2 apiVersion: networking.k8s.io/v1
3 metadata:
4   name: ''
5   namespace: default
6 spec:
7   podSelector: {}
8   policyTypes: []
9 
```

On the right, there's a sidebar titled 'NetworkPolicy' with tabs for 'Schema' and 'Samples'. The 'Schema' tab contains the following text:

NetworkPolicy describes what network traffic is allowed for a set of Pods

- **apiVersion** string  
APIVersion defines the versioned schema of this representation of an object.

Figure 3.15: Manifest file of a network policy

## Set a Cluster-scoped Network Policy

Red Hat OpenShift also provides the AdminNetworkPolicy (ANP) resource for managing network policies. The ANP is a cluster-scoped network policy that can restrict or enable access before creating the targeted namespace.

The ANP resource is similar to a namespace-scoped network policy, with three additional attributes: the **subject**, the **priority**, and the **action** attributes. The **subject** attribute identifies the namespaces or pods that the ANP applies to.

The **priority** attribute determines the order of evaluation. The lower the value, the higher the precedence. The priority attribute takes a value from 0-99. If two ANP policies have conflicting rules, then the policy with the lower-priority attribute value takes precedence over the other rule.

The **action** attribute takes a value of Allow, Deny, or Pass.

Action	Meaning
Allow	Allow ingress or egress
Deny	Deny ingress or egress
Pass	Defer access control to the namespaced network policies

You cannot use namespaced network policies to override an ANP that uses Allow or Deny actions. For example, the following ANP gives the monitoring namespace access to any namespace that uses the security=unrestricted label. The targeted namespace can use a network policy to block monitoring only after first removing the identifying label from the namespace.

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: sample-anp-allow-monitoring ①
spec:
```

```

priority: 50 2
subject:
  namespaces:
    matchLabels:
      security: unrestricted 3
ingress: 4
- name: "allow-monitoring-unrestricted" 5
  action: "Allow" 6
  from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring

```

- 1** Specify a name for the ANP.
- 2** The **priority** field determines the precedence of the traffic rules.
- 3** Use labels to select the namespaces to which the rules apply.
- 4** ANPs have both ingress and egress rules.
- 5** Specify a name for the ANP rule.
- 6** The **action** value can be **Allow**, **Deny**, or **Pass**.



## References

For more information, refer to the *Network Security* section in the *Red Hat OpenShift Container Platform Networking* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#network-security](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#network-security)

For more information, refer to the *Admin Network Policy* section in the *Red Hat OpenShift Container Platform Networking* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#ovn-k-anp](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#ovn-k-anp)

## ► Guided Exercise

# Restrict Access to a Virtual Machine by Using a Network Policy

Define a network policy to limit access to a database server to only specific pods and projects.

## Outcomes

- Create a network policy to protect a virtual machine.
- Add a label to a namespace.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. This command also creates the `network-policy` namespace and starts a virtual machine in that namespace: the `mariadb-server` VM, which hosts a MariaDB database.

```
[student@workstation ~]$ lab start network-policy
```

## Instructions

- 1. As the `admin` user, go to the Red Hat OpenShift web console, and confirm that the `mariadb-server` VM is running.
- 1.1. From the command line, log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Set the `network-policy` project as the active project, if it is not already set.

```
[student@workstation ~]$ oc project network-policy
Now using project "network-policy" on server "https://api.ocp4.example.com:6443".
```

- 1.3. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.4. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- 1.5. Select the **Developer** perspective. Go to **Topology** and select the **network-policy** project. Click the `mariadb-server` icon. Confirm that the `mariadb-server` VM is running.

The screenshot shows the OpenShift Topology interface. On the left, there is a list of VMs with icons. One VM is selected, showing its details on the right. The VM is named 'mariadb-server'. The 'Status' field is highlighted with a red box and shows 'Running'. Other fields include 'Namespace' (NS network-policy), 'Pod' (virt-launcher-mariadb-server-b2r7x), 'Labels' (flavor.template.kubevirt.io/small=true, kubevirt.io/vm=mariadb-server), and 'Boot order' (1. mariadb-server (Disk)).

- 2. Protect the database server so that only clients that are running in selected namespaces can access it.

- 2.1. Add the `allowing=db-clients` label to the `network-policy` namespace.

From the command line, use the `oc label namespace` command to add this label to the `network-policy` namespace.

```
[student@workstation ~]$ oc label namespace network-policy allowing=db-clients  
namespace/network-policy labeled
```

The cluster administrator adds such labels to help with using network policies. The label on the namespace enables the network policy to identify namespaces that are permitted to access the database.

- 2.2. Confirm that the `allowing=db-clients` label is set on the `network-policy` namespace.

From the command line, use the `oc get namespace` command to view the labels for the `network-policy` namespace.

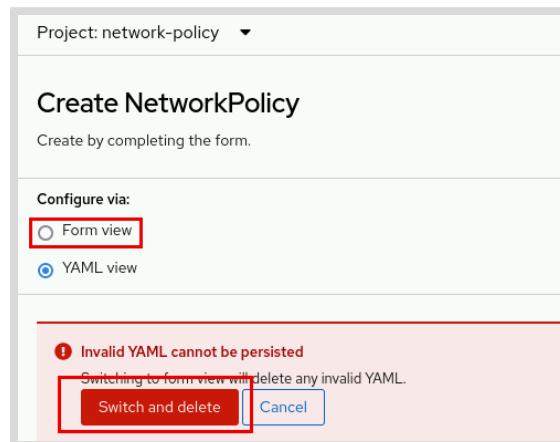
```
[student@workstation ~]$ oc get namespace network-policy --show-labels  
NAME      STATUS   AGE     LABELS  
network-policy  Active  3m3s  allowing=db-clients, kubernetes.io/metadata=true
```

The `--show-labels` subcommand output confirms the addition of the `allowing=db-clients` label.

- 2.3. Confirm that the `app=mariadb-server` label is applied to the `mariadb-server` VMI. The label on the VMI enables the network policy to identify the pod that the network traffic rule applies to.

```
[student@workstation ~]$ oc get vmi mariadb-server --show-labels
NAME          AGE   PHASE     IP           ...   LABELS
mariadb-server  82m  Running   10.11.0.22  ...   app=mariadb-server, ...
```

- 2.4. Prepare the network policy resource. From the web console, select the Administrator perspective. Go to **Networking > NetworkPolicies**, select the **network-policy** project, and click **Create NetworkPolicy**. If the page starts with **YAML view**, then select **Form view** and click **Switch** and delete to confirm the change.



### Note

The **Switch and delete** warning is absent if the page starts with the form view.

- 2.5. In the **Policy Name** field, enter **allow-db-clients**.
- 2.6. Select **Add pod selector**. Enter the form values from the following table.

Field	Value
Pod Selector > Label	app
Pod Selector > Selector	mariadb-server

- 2.7. Select **Add ingress rule**. Select **Add allowed source > Allow pods from inside the cluster**.

Ingress rule

Remove Add allowed source ▾

Sources added to this rule will allow traffic to the pod. This list are combined using a logical OR operation.

Allow pods from the same namespace  
Allow pods from inside the cluster  
Allow peers by IP block

Ports

Add ports to restrict traffic through them. If no ports are provided, your policy will make all ports accessible to traffic.

+ Add port

2.8. Select **Add namespace selector** and enter the values from the following table.

Field	Value
Add namespace selector > Label	allowing
Add namespace selector > Selector	db-clients

2.9. Go to the **Ports** section of the form. Select **Add port**. Enter the values from the following table.

Field	Value
Protocol	TCP
Port	3306



### Warning

Do not click **Create** yet. The form view does not retrieve the selected project in use. The project has the same name as the namespace. Clicking **Create** would try to add a network policy to the default namespace. The web console shows an error if you attempt to create the network policy from this point. To resolve this problem, in the next instruction, you change to the YAML view to add the **network-policy** namespace to the resource manifest.

2.10. Scroll to the top of the form and select **YAML view**. In the YAML view, add **network-policy** for the namespace. The resulting YAML follows.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-db-clients
  namespace: network-policy
```

```

spec:
  podSelector: ①
    matchLabels:
      app: mariadb-server
  policyTypes:
    - Ingress
  ingress: ②
    - from:
      - podSelector: {}
        namespaceSelector:
          matchLabels:
            allowing: db-clients
  ports:
    - port: 3306
      protocol: TCP

```

- ① The `podSelector` section indicates that the policy applies to pods with the `app: mariadb-server` label. You previously set that label to the `virt-launcher` pod, which is associated with the `mariadb-server` VM.
- ② The `ingress` section enables inbound connections to port 3306 from all pods in namespaces with the `allowing: db-clients` label.

Click **Create**.



### Note

The `~/D0316/labs/network-policy/policy.yaml` file contains the correct configuration, and you can use it for comparison.

- 3. Create a service named `mariadb` with the `ClusterIP` type. The service must listen on TCP port 3306 and forward traffic to the VM on that port.

- 3.1. Use the `virtctl expose` command to create the service.

```
[student@workstation ~]$ virtctl expose vmi mariadb-server --name mariadb \
--type ClusterIP --port 3306 --target-port 3306
Service mariadb successfully exposed for vmi mariadb-server
```

- 3.2. Confirm that the endpoints of the service include the `mariadb-server` VMI.

```
[student@workstation ~]$ oc get service,vmi,endpoints
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP     PORT(S)      AGE
service/mariadb   ClusterIP  172.30.197.46   <none>        3304/TCP   55s

NAME                           ...      IP      ...
virtualmachineinstance.kubevirt.io/mariadb-server  ...  10.11.0.22  ...

NAME           ENDPOINTS      AGE
endpoints/mariadb  10.11.0.22:3306  55s
```

- 4. Confirm that a pod in the `network-policy` namespace can access the database. Confirm that a pod that is running in a different namespace can no longer access the database.

For both pods, use the `registry.ocp4.example.com:8443/redhattraining/mariadb:10.5` container image, which provides the `mysql` client command.

- 4.1. From the command line, run a temporary interactive pod in the `network-policy` namespace. The pod uses the `registry.ocp4.example.com:8443/redhattraining/mariadb:10.5` container image, which provides the `mysql` client command. At the MariaDB prompt, run the `SHOW TABLES;` command. Use the `exit` command to log out of the MySQL session.

**Note**

You can copy and paste the command from the `~/D0316/labs/network-policy/commands.txt` file that the `lab` command prepared for you.

```
[student@workstation ~]$ oc run testdb -it --rm \
--image=registry.ocp4.example.com:8443/redhattraining/mariadb:10.5 \
--command -- mysql --connect-timeout=5 -u devuser -p'developer' \
-h mariadb.network-policy.svc.cluster.local sakila
...output omitted...
MariaDB [sakila]> SHOW TABLES;
+-----+
| Tables_in_sakila |
+-----+
| actor           |
...output omitted...

MariaDB [sakila]> exit
Bye
Session ended, resume using 'oc attach testdb -c testdb -i -t' command when the
pod is running
pod "testdb" deleted
```

- 4.2. Create a `test` project.

```
[student@workstation ~]$ oc new-project test-project
Now using project "test-project" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 4.3. Confirm that the `test` project does not use the `allowing=db-clients` label.

```
[student@workstation ~]$ oc get project test-project --show-labels
NAME          DISPLAY NAME   STATUS    LABELS
test-project            Active    kubernetes.io/metadata.name=test-proj...
```

A cluster administrator must explicitly add a custom label to a namespace. Therefore, the `test-project` namespace does not have the `allowing=db-clients` label.

- 4.4. From the command line, run a temporary interactive pod in the `test-project` namespace. The pod uses the `registry.ocp4.example.com:8443/redhattraining/mariadb:10.5` container image, which provides the `mysql` client command.

The command does not display the MariaDB prompt, and times out after five seconds. The timeout confirms that you can access the database only from namespaces with the `allowing: db-clients` label.

```
[student@workstation ~]$ oc run testdb -it --rm \
--image=registry.ocp4.example.com:8443/redhattraining/mariadb:10.5 \
--command -- mysql --connect-timeout=5 -u devuser -p'developer' \
-h mariadb.network-policy.svc.cluster.local sakila
If you don't see a command prompt, try pressing enter.
```

**Enter**

```
ERROR 2002 (HY000): Can't connect to MySQL server on 'mariadb.network-
policy.svc.cluster.local' (110)
...output omitted...
```

4.5. Delete the test project.

```
[student@workstation ~]$ oc delete project test-project
project.project.openshift.io "test-project" deleted
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-policy
```

# Configuring External Access to Virtual Machines

## Objectives

- Create route and ingress resources to provide external access to virtual machines.

## Services Objects with External Access

Red Hat OpenShift provides several mechanisms to make your applications available from outside the cluster.

OpenShift provides different service resource types to configure external access:

### **NodePort**

The NodePort resource type exposes a network port, on all your cluster nodes, that redirects the incoming traffic to the pods or VMs of the service. A NodePort service requires allowing direct network connections to a cluster node, which is a security risk. Configuring NodePort services is covered later in the course.

### **LoadBalancer**

The LoadBalancer resource type instructs Red Hat OpenShift to activate a load balancer in a cloud environment. A load balancer instructs OpenShift to interact with the cloud provider that the cluster is running in, to provision a load balancer. The load balancer then provides an externally accessible IP address to the application.

- For clusters that do not run on a cloud provider, the MetallB operator can provide the load balancing services. Configuring a MetallB load balancer service is covered later in the course.
- Non-cloud environments can also configure an External IP to send traffic to services by using the LoadBalancer service type. The External IP feature is disabled by default and can be a security risk, because sensitive in-cluster traffic that is destined for external resources might be intercepted.

### **Ingress Controller**

The Ingress Operator manages the Ingress Controller, and provides the typical way to enable external access to cluster resources. The Ingress Controller is limited to HTTP, HTTPS, and TLS by using SNI, and is configured to accept external requests and to proxy them based on configured routes.

- Red Hat OpenShift provides route resources to expose your applications to networks outside the cluster. Routes provide ingress to HTTP and HTTPS traffic, TCP applications, and also to non-TCP traffic. With routes, you can access your application with a unique hostname that is publicly accessible.
- Kubernetes provides ingress resources that are similar to route resources, although routes provide more features, such as TLS reencryption, TLS passthrough, and split traffic for blue-green deployments.

When you migrate a VM to OpenShift Virtualization from another hypervisor technology, you can configure external access so that clients can still access the application. You can also make a database server available to services that are running outside your OpenShift cluster. You

can expose legacy applications that you plan later to convert to containers. After converting an application, you reconfigure external access so that clients can continue to access the application with the same DNS name.

## Configure Red Hat OpenShift Routes

Red Hat OpenShift routes provide ingress HTTP, HTTPS, and TLS traffic to services in the cluster. A route connects a public-facing DNS hostname to an internal-facing service IP.

Red Hat OpenShift implements routes by deploying a cluster-wide router service, which runs an HAProxy load balancer as a containerized application in the OpenShift cluster. Red Hat OpenShift scales and replicates router pods like any other application.

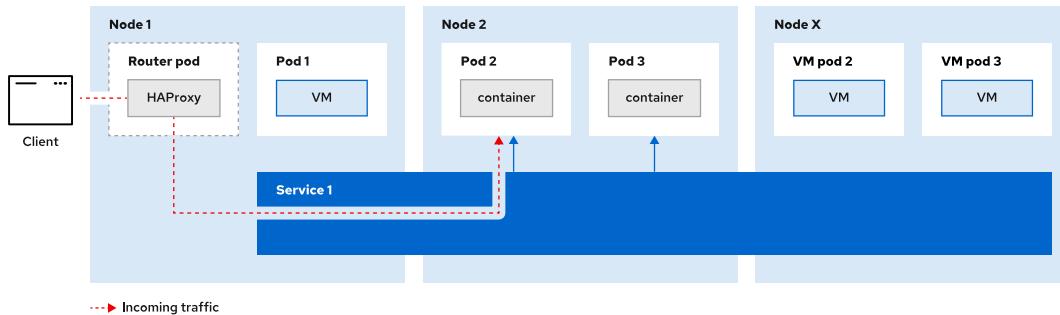


Figure 3.19: Incoming traffic through the Red Hat OpenShift router pod

Red Hat OpenShift can automatically assign a DNS entry to your application when creating a route. Red Hat OpenShift assigns a name in the `routename-namespace.default_domain` format. The Red Hat OpenShift administrators configure the `default_domain` wildcard domain name during the cluster installation. For example, if your default domain is `mycompany.com` and you create a route named `intranet` in the `prod` project, then the DNS name is `intranet-prod.mycompany.com`.

Red Hat OpenShift administrators must configure the company's DNS system so that the `default_domain` wildcard DNS record points to the public-facing IP addresses of the nodes that are running the router.



### Note

The DNS server that hosts the wildcard domain knows nothing about route hostnames. The server merely resolves any name to the configured IP addresses. Only the Red Hat OpenShift router knows about route hostnames, and treats each one as an HTTP virtual host. The Red Hat OpenShift router blocks invalid wildcard domain hostnames that do not correspond to any route and returns an HTTP error.

Routes work only with certain types of traffic: HTTP, HTTPS with Server Name Indication (SNI), and TLS with SNI. With the SNI, the client can send the name of the host that it tries to reach in clear text during the handshake process. Red Hat OpenShift uses that feature to identify the service to target when it receives encrypted traffic.

For other traffic types, such as UDP traffic or non-web TCP traffic, Red Hat recommends that you use services of the LoadBalancer or the NodePort type.

## Create Routes

After creating a service of the ClusterIP type, which points to your pods or virtual machines, you can create a route for your application. You can create a resource file in YAML or JSON format to declare your route. However, Red Hat OpenShift provides the `oc expose service` command to simplify the creation process:

```
[user@host ~]$ oc expose service/web  
route.route.openshift.io/web exposed
```

That command creates a `route` resource with the same name as the service. Use the `--name` option to define a different name.

Red Hat OpenShift automatically assigns a DNS name to the route. You can use the `--hostname` option followed by the DNS name to define it manually. The following command creates a `web` route that defines the `web-production.apps.mycompany.com` hostname.

```
[user@host ~]$ oc expose service/web --name web \  
--hostname web-production.apps.mycompany.com  
route.route.openshift.io/web exposed
```

Use the `oc get route` command to retrieve the DNS name that is assigned to the route:

```
[user@host ~]$ oc get route web  
NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD  
web web-production.apps.mycompany.com web 8080 None
```

From the web console, go to **Networking > Routes**, select a namespace, click **Create Route**, and then complete the form with the route details. You must name the route and select the associated service and its ports.

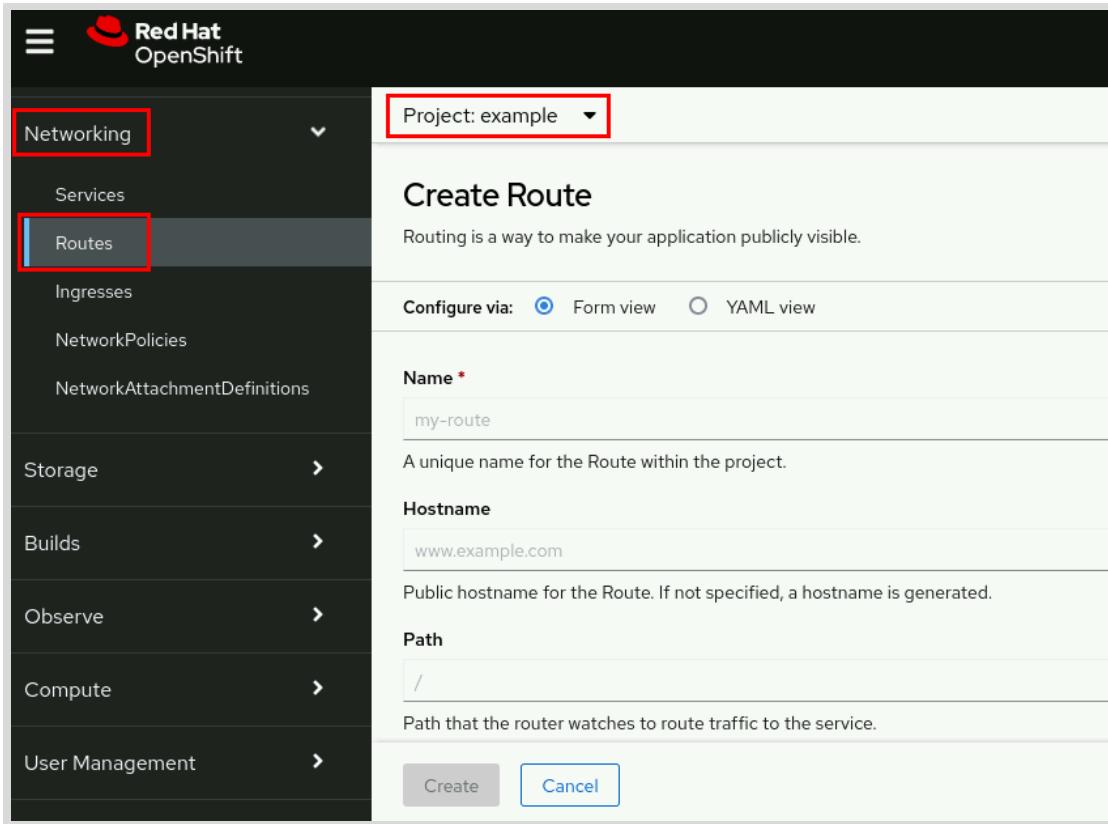


Figure 3.20: Create a route by using the web console

## Create Path-based Routes

The route resource definition accepts the path variable to specify the path component to the URL. With that configuration, you can use the same DNS name for several services and redirect the traffic based on the path component. For example, you could use the `http://intranet-prod.apps.mycompany.com/static` URL to send traffic to pods or to virtual machines that serve static web content. For REST API requests, you could use the `http://intranet-prod.apps.mycompany.com/api` URL to forward the traffic to the application back end.

Add the `--path` option to the `oc expose service` command to configure path-based routes:

```
[user@host ~]$ oc expose service/static --path=/static \
  --hostname=intranet-prod.apps.mycompany.com
route.route.openshift.io/static exposed

[user@host ~]$ oc expose service/restapi --path=/api \
  --hostname=intranet-prod.apps.mycompany.com
route.route.openshift.io/restapi exposed

[user@host ~]$ oc get routes
NAME      HOST/PORT                      PATH      SERVICES    PORT ...
static   intranet-prod.apps.mycompany.com /static   static      8080 ...
restapi  intranet-prod.apps.mycompany.com /api     restapi    80      ...
```

## Encrypted Routes

Routes can be encrypted (TLS or HTTPS) or unencrypted (HTTP). Encrypted routes use several types of TLS termination to serve certificates to the client.

An encrypted route specifies the TLS termination of the route. The following list describes the available termination types:

### Edge

TLS termination occurs at the router before Red Hat OpenShift routes the traffic to the pods or to the virtual machines. The router serves the TLS certificates to the clients. You can provide your certificate when you create the routes, or Red Hat OpenShift can use its certificate. Because TLS is terminated at the router, connections from the router to the endpoints over the internal network are not encrypted.

### Pass through

The router sends the traffic straight to the destination pod or virtual machine. In this mode, the application is responsible for serving certificates for the traffic.

### Re-encryption

Re-encryption is a variation of edge termination. The router terminates TLS with a certificate and then re-encrypts its connection to the endpoint, which might have a different certificate. Therefore, the full path of the connection is encrypted, even over the internal network.

The references section at the end provides links to documents that explain how to configure secured routes.

## Configuring Encrypted Routes

When you create a route by using the web console, the **Create Route** form includes a **Security** section for configuring encrypted routes. The subsequent fields on the form change based on the selected **TLS termination** type. When you select **Edge** termination, the following form is displayed:

The screenshot shows the 'Create Route' form with the 'Security' section expanded. The 'Secure Route' checkbox is checked. Below it, a note states: 'Routes can be secured using several TLS termination types for serving certificates.' A dropdown menu labeled 'TLS termination \*' is set to 'Edge'. Another dropdown menu labeled 'Insecure traffic' has 'Select insecure traffic type' and a note below it: 'Policy for traffic on insecure schemes like HTTP.' A 'Certificates' section follows, with a note: 'TLS certificates for edge and re-encrypt termination. If not specified, the router's default certificate is used.' A 'Certificate' input field with a 'Browse...' button is shown, with a note: 'The PEM format certificate. Upload file by dragging & dropping, selecting it, or pasting from the clipboard.'

**Figure 3.21: Create a route with TLS Edge termination**

The **Browse...** buttons enable you to select a certificate and a private key. OpenShift uses a certificate and a private key from the internal **Certificate Authority** (CA) if a certificate and a private key are not provided. The **Insecure traffic** field sets the policy for how OpenShift

manages insecure traffic such as HTTP. The following table gives possible values to select for this field:

Value	Meaning
None	Unencrypted traffic is disabled
Allow	Unencrypted traffic is sent over an insecure scheme, such as HTTP
Redirect	Redirects an unencrypted scheme (HTTP) to an encrypted scheme (HTTPS)

The form uses minor variations of these fields for TLS passthrough and TLS reencryption.

From the command line, the `oc expose` command can create only unencrypted routes. Use the `oc create route` command instead to create encrypted routes.

The following example uses the `web` service to create a route with the same name that uses Edge encryption:

```
[user@host ~]$ oc create route edge \ ①
--service web \ ②
--hostname api.apps.acme.com ③
```

- ① The `edge` keyword creates a route that uses Edge TLS termination. Use the `passthrough` flag to create a route that uses passthrough TLS termination. Use the `reencrypt` flag to create a route that uses reencryption TLS termination.
- ② A service is required to create a route.
- ③ The host name is automatically generated if not specified.

In this example, the OpenShift ingress controller provides a certificate from the internal CA. If you prefer to provide a custom private key and a signed certificate, then use the `--key` and `--cert` options to provide those items.

## Configure Routes for Virtual Machines

To configure a route for a VM, you must first create a service for the `virt-launcher` pod of the VM:

1. Add a label to the `VirtualMachine` resource in the `.spec.template.metadata.labels` path.
2. Create the service resource from a definition file in YAML format. Use the label to select the `virt-launcher` pod. The following example expects the `virt-launcher` pod to have the `tier: front` label.

```
apiVersion: v1
kind: Service
metadata:
  name: web
  namespace: prod2
spec:
  type: ClusterIP
```

```

selector:
  tier: front
ports:
  - protocol: TCP
    port: 80
    targetPort: 8080

```

After you create the service, you can then create the route.

From the command line, use the `oc expose service/web` command to create the route:

```
[user@host ~]$ oc expose service/web
route.route.openshift.io/web exposed
```

## Enable Traffic from the Router Pods

When you use network policies in your project, you must add a rule to enable the traffic that comes from the router pods. Otherwise, the `route` resources do not work, because the router pods cannot reach the destination pods and virtual machines.

The router pods run in the `openshift-ingress` namespace. That namespace already has a label that you can use with your network policies. The following command shows that the label is `network.openshift.io/policy-group: ingress`:

```
[user@host ~]$ oc get namespace openshift-ingress -o yaml
kind: Namespace
apiVersion: v1
metadata:
...output omitted...
labels:
  kubernetes.io/metadata.name: openshift-ingress
  name: openshift-ingress
  network.openshift.io/policy-group: ingress
  olm.operatorgroup.uid/44814614-edc3-4e46-9491-5f0f97af9c84: ""
  openshift.io/cluster-monitoring: "true"
  policy-group.network.openshift.io/ingress: ""
...output omitted...
```

Add the following `NetworkPolicy` resource to your project to enable the traffic from the router pods:

```

kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-from-openshift-ingress
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          network.openshift.io/policy-group: ingress ①
  podSelector: {} ②

```

- ➊ The network policy enables the traffic that comes from the `openshift-ingress` namespace, because that namespace has the `network.openshift.io/policy-group: ingress` label.
- ➋ The network policy targets all the pods in the current namespace when the `podSelector` parameter is empty.

## Configure Kubernetes Ingress Resources

The `route` resources are specific to Red Hat OpenShift. Kubernetes provides `ingress` resources, with some of the same features as routes. Some route features, such as TLS reencryption, are not available with `ingress` resources.

The following example defines an `ingress` resource named `web`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web
spec:
  rules:
    - host: web-production.apps.mycompany.com ①
      http:
        paths:
          - path: / ②
            pathType: Prefix ③
            backend:
              service: ④
              name: web
              port:
                number: 8080
```

- ➊ The `host` parameter provides the URL to assign to the application.
- ➋ The `path` parameter provides the path component of the URL.
- ➌ The `pathType` parameter specifies how to match the path component of the URL. If set to `Exact`, then the path must be identical to the `path` parameter. If set to `Prefix`, then any path that starts with the `path` component matches.
- ➍ The `service` section provides the service to associate with the `ingress` resource.

Red Hat OpenShift implements `ingress` resources with `route` resources. The following example shows the `route` resource that Red Hat OpenShift creates:

```
[user@host ~]$ oc get ingress
NAMESPACE  NAME CLASS      HOSTS          ADDRESS ...
production  web  <none>   web-production...com  router-default...com

[user@host ~]$ oc get routes
NAME      HOST/PORT          PATH  SERVICES  PORT ...
web-djh2b web-production.apps.mycompany.com  /      web        8080-tcp ...
```

To create an `ingress` resource from the web console, go to **Networking > Ingresses**, click **Create Ingress**, and then complete the YAML manifest with the `ingress` parameters.

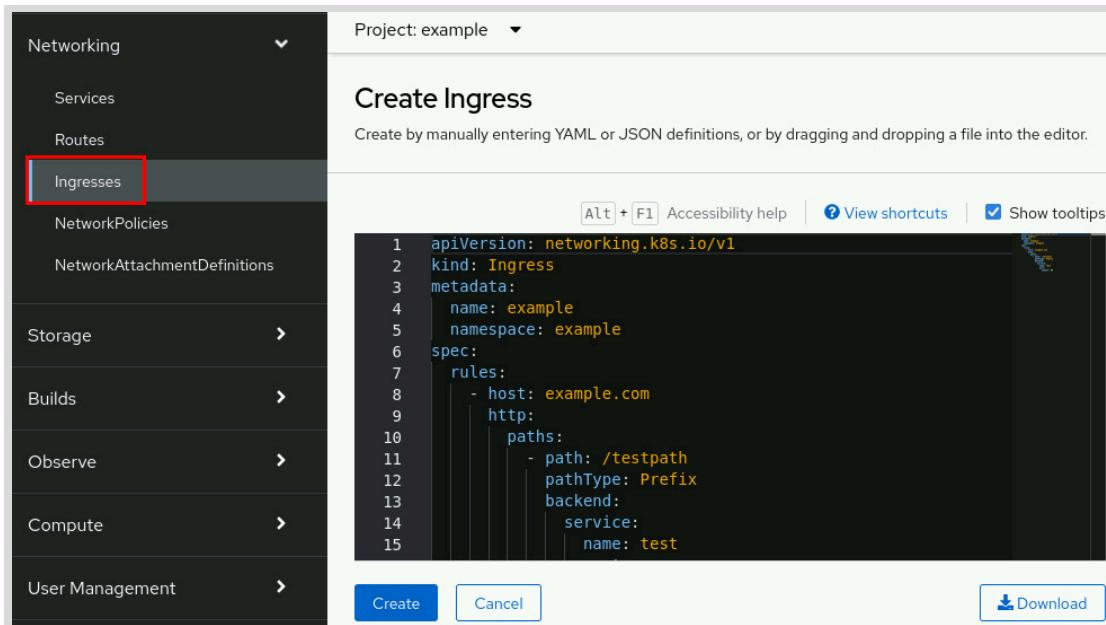


Figure 3.22: Create an ingress by using the web console



## References

For more information, refer to the *Configuring Ingress Cluster Traffic* chapter in the *Red Hat OpenShift Container Platform Networking* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#overview-traffic](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#overview-traffic)

For more information on secured routes, refer to the *Secured Routes* section in the *Red Hat OpenShift Container Platform Networking* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#configuring-default-certificate](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#configuring-default-certificate)

For more information about OpenShift routes, refer to the *OpenShift Ingress versus OpenShift Route* article at  
<https://www.redhat.com/en/blog/kubernetes-ingress-vs-openshift-route>

## ► Guided Exercise

# Configure External Access to a Virtual Machine

Configure a service and a route to establish external access for a web server that is hosted on a virtual machine.

## Outcomes

- Create a service of the ClusterIP type.
- Confirm that the service endpoint resolves to the virtual machine instance.
- Create a route to expose the virtual machine for external access.
- Update the network policy to allow traffic from the route.
- Create an encrypted route.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the `web-servers` namespace and starts a virtual machine named `hello-web` in that namespace.

```
[student@workstation ~]$ lab start network-external
```

## Instructions

- 1. As the `admin` user, locate and go to the Red Hat OpenShift web console. Confirm that the VM is running.
- 1.1. From a command line, log in to your Red Hat OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Set the `web-servers` project as the active project.

```
[student@workstation ~]$ oc project web-servers
Now using project "web-servers" on server "https://api.ocp4.example.com:6443".
```

- 1.3. Open a web browser and go to the web console URL.  
`https://console-openshift-console.apps.ocp4.example.com`

Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.

- 1.4. Go to **Virtualization > VirtualMachines** and then select the `web-servers` project. Confirm that the `hello-web` VM is running.

Name	Status	Conditions	Node	IP address
VM hello-web	Running		N worker01	10.9.2.59

- ▶ 2. Add the app: `helloworld` label to the `hello-web` VM.

- 2.1. From the command line, edit the `hello-web` VM.

```
[student@workstation ~]$ oc edit vm/hello-web
```

- 2.2. Add the app: `helloworld` label in the `spec.template.metadata.labels` path. As a reminder, Kubernetes uses this `labels` section to assign the label to the `virt-launcher` pod. Do not change the other `labels` sections.

```
...output omitted...
spec:
  dataVolumeTemplates:
  ...output omitted...
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: helloworld
        flavor.template.kubevirt.io/small: "true"
        kubevirt.io/domain: hello-web
        kubevirt.io/size: small
    ...output omitted...
```

Save the edits.

- 2.3. Restart the VM so that Kubernetes re-creates an instance that includes the new label. From the web console, click the vertical ellipsis icon `⋮` to the right of the `hello-web` VM and then click **Restart**.

Name	Status	Conditions	Node
VM hello-web	Running	DataVolumesReady=True	N master03

- 2.4. From the command line, use the `watch oc get vmi,pods` command to confirm that Kubernetes re-creates the VMI resource and the `virt-launcher` pod. Use the `-l` option to filter the results and confirm that the label is now available on the VMI. Press `Ctrl+C` to exit the `watch oc get` command.

```
[student@workstation ~]$ watch oc get vmi,pods -l app=helloworld
...output omitted...

NAME                                     AGE PHASE   IP           NODENAME READY
virtualmachineinstance.kubevirt.io/hello-web 11s Running 10.9.2.61 worker01 True

NAME                           READY   STATUS    RESTARTS   AGE
pod/virt-launcher-hello-web-6vvsr  1/1    Running   0          11s
```

- 3. Create a service named `hello-web` with the `ClusterIP` type. Use the `app: helloworld` label for the selector. The `hello-web` service must accept traffic on TCP port 80 and forward it to the VM on port 80.
- 3.1. From the command line, use the `oc create` command to create the manifest for the service.

```
[student@workstation ~]$ oc create service clusterip hello-web \
--tcp=80:80 --dry-run=client -o yaml > hello-web-service.yaml
```

- 3.2. Open the `hello-web-service.yaml` manifest in an editor and modify the selector to `app: helloworld`. Save the change to the file.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: hello-web
  name: hello-web
spec:
```

```

ports:
- name: 80-80
  port: 80
  protocol: TCP
  targetPort: 80
selector:
  app: helloworld
type: ClusterIP
status:
  loadBalancer: {}

```

- 3.3. Use the `oc create` command to create the service. The service makes the VM available on a stable IP address.

```
[student@workstation ~]$ oc create -f hello-web-service.yaml
service/hello-web created
```

- 3.4. Confirm that the `hello-web` service uses the VMI as an active endpoint.

```
[student@workstation ~]$ oc get vmi,endpoints
NAME                                     AGE   PHASE     IP
NODENAME      READY
virtualmachineinstance.kubevirt.io/hello-web 38m  Running  10.9.0.28
master03      True

NAME           ENDPOINTS          AGE
endpoints/hello-web  10.9.0.28:80  4m42s
```

► 4. Use the OpenShift web console to create a route for the `hello-web` service.

- 4.1. From the web console, go to **Networking > Routes**.
- 4.2. Select the `web-servers` project and then click **Create Route**.
- 4.3. Complete the **Create Route** form by using the following information:

Field	Value
Name	<code>hello-web</code>
Hostname	<i>Leave the default value</i>
Path	<i>Leave the default value</i>
Service	<code>hello-web</code>
Target port	<code>80 → 80(TCP)</code>

- 4.4. Click **Create** to create the route. The **Route details** page is displayed.
- 4.5. Click the link in **Location** section to confirm that you can access the web application that is running on the VM.



The screenshot shows the 'Route details' page for a route named 'hello-web'. The 'Name' is 'hello-web'. The 'Location' field contains the URL 'http://hello-web-web-servers.apps.ocp4.example.com', which is highlighted with a red box.

The web application displays a `Hello world` page in a new web browser tab.

► 5. Observe the network access from other pods.

5.1. From the command line, create a temporary project for a test pod.

```
[student@workstation ~]$ oc new-project test-project
Now using project "test-project" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

5.2. Deploy a pod in the `test-project` namespace.

```
[student@workstation ~]$ oc new-app --name test \
--image registry.ocp4.example.com:8443/ubi9/httpd-24:latest
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "test" created
  deployment.apps "test" created
  service "test" created
--> Success
...output omitted...
```

5.3. Capture the pod name in a variable.

```
[student@workstation ~]$ PODNAME=$(oc get pod -o name)
[student@workstation ~]$ echo $PODNAME
pod/test-7dd486c6dc-pmqh5
```

Your `PODNAME` value might differ from the example.

5.4. View the `web-server` namespace resources.

```
[student@workstation ~]$ oc get svc,vmi,route -n web-servers
NAME          TYPE      CLUSTER-IP        EXTERNAL-IP      PORT(S)...
service/hello-web   ClusterIP  172.30.148.136    <none>           80/TCP ...
NAME                           AGE   PHASE     IP      ...
virtualmachineinstance.kubevirt.io/hello-web  44m   Running   10.8.0.15 ...
```

NAME	HOST/PORT	...
route.route.openshift.io/hello-web	hello-web-web-servers.apps.ocp4.example.com	

- 5.5. Test access to each of the three resources. Your IP values differ from the following examples. Use the service and VMI IP values from the previous step instead of the IP values from the examples.

```
[student@workstation ~]$ oc rsh $PODNAME curl --max-time 5 \
  hello-web-web-servers.apps.ocp4.example.com | grep Welcome
  <p>Hello, World! Welcome to Red Hat Training.</p>
[student@workstation ~]$ oc rsh $PODNAME curl --max-time 5 \
  172.30.148.136 | grep Welcome
  <p>Hello, World! Welcome to Red Hat Training.</p>
[student@workstation ~]$ oc rsh $PODNAME curl --max-time 5 \
  10.8.0.15 | grep Welcome
  <p>Hello, World! Welcome to Red Hat Training.</p>
```

Without a network policy, any pod in the cluster can connect to the `hello-web` resources.

► 6. Add a network policy to allow traffic from the route.

- 6.1. From the command line on the `workstation` machine, review the `~/D0316/labs/network-external/allow-hello-web-np.yaml` resource file that the `lab` command prepared.

```
[student@workstation ~]$ cat ~/D0316/labs/network-external/allow-hello-web-np.yaml
apiVersion: v1
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-hello-web-route
  namespace: web-servers
spec:
  podSelector:
    matchLabels:
      app: helloworld
  ingress:
    - ports:
        - protocol: TCP
          port: 80
        - protocol: TCP
          port: 443
      from:
        - namespaceSelector:
            matchLabels:
              network.openshift.io/policy-group: ingress
  policyTypes:
    - Ingress
```

- 6.2. Use the `oc create` command to create the network policy.

```
[student@workstation ~]$ oc create -f \
~/DO316/labs/network-external/allow-hello-web-np.yaml
networkpolicy.networking.k8s.io/allow-hello-web-route created
```

► 7. Verify the network policy change.

- 7.1. Test access to each of the three resources. Use service and VMI IP values from your environment instead of the IP values from the examples. You can pull these commands from your command history.

```
[student@workstation ~]$ oc rsh $PODNAME curl --max-time 5 \
hello-web-web-servers.apps.ocp4.example.com | grep Welcome
command terminated with exit code 28
[student@workstation ~]$ oc rsh $PODNAME curl --max-time 5 \
10.8.0.15 | grep Welcome
command terminated with exit code 28
[student@workstation ~]$ oc rsh $PODNAME curl --max-time 5 \
172.30.148.136 | grep Welcome
command terminated with exit code 28
```

The test-project namespace does not have access, because network policy requires access via the pods in the ingress controller namespace.

► 8. Add TLS edge termination to encrypt traffic between the client and the router.

- 8.1. Use the OpenShift web console to update the hello-web route. From the web console, go to **Networking > Routes**.
- 8.2. Select the web-servers project and then select the hello-web route.
- 8.3. Select **Action > Edit route** and scroll to the **Security** section of the form.
- 8.4. Click the **Secure Route** checkbox. **TLS termination** appears under the checkbox. Set the **TLS termination** value to **Edge**. Click **Save**.
- 8.5. Click the link in **Location** to confirm that you can access the web application that is running on the VM from the **https** URL.

The web application displays a **Hello world** page in a new web browser tab.

## Finish

As the student user on the workstation machine, confirm that you are in the /home/student directory and then use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-external
```

# Configure External Access with Load Balancer Services

## Objectives

- Configure external access to virtual machines by using load balancer services from MetalLB.

## Exposing Non-HTTP Services

You can expose HTTP-based applications by using routes and ingress resources. Inbound HTTP traffic is forwarded to the application without exposing additional ports or IP addresses in the cluster.

However, in some scenarios, ingresses and routes are not sufficient to expose the service that a VM provides. Other text-based protocols, such as SMTP, POP3, and IMAP, cannot be exposed with routes and ingress resources. Binary protocols, such as SSH, LDAP, or a database such as MySQL, cannot be exposed with routes and ingress resources.

## Kubernetes Services

The Kubernetes services have a selector that matches one or more pods. The communication is forwarded to one of the matching pods when a connection is initiated to the service IP address and port. A service resource contains the following information:

- A selector that describes the pods that run the back-end service.
- A list of the ports that provide the service on the pods.

You can expose the non-http services that a VM provides by using a node port or a load balancer service. The node port or load balancer service gives you direct access to the VM service port from outside the OpenShift cluster.

### Node port service

- The node port service exposes a service port on all nodes in the cluster.
- If not specified, the node port is assigned from the default range of 30000 to 32767.
- The port of the NodePort must be within the configured range, and is unlikely to match the port of the targeted VM.

### Load balancer service

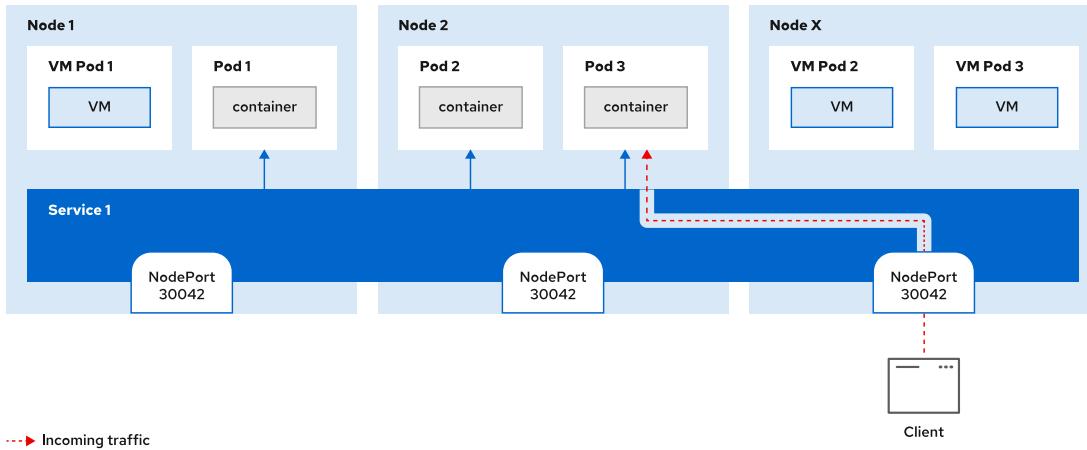
- The load balancer service provides a fixed IP address and a service port that forward the communication to the pods that match the selector.
- The IP address of the load balancer service does not change if the `virt-launcher` pod is moved to another node.
- The IP address that is assigned to the load balancer service depends on the underlying infrastructure, either from the cloud provider or from the MetalLB operator.

## Configure Node Port Service Resources

Service resources use the `ClusterIP` resource type by default. With that type, the service is accessible only from inside the cluster.

By setting the `type` parameter to `NodePort`, OpenShift opens the same network port on all the cluster nodes and then redirects the incoming traffic to your pods or VMs. By default, OpenShift allocates a port in the 30000 to 32767 range.

With this configuration, an external client can reach your service by targeting the IP address of one of your cluster nodes and the allocated port.



**Figure 3.26: Incoming traffic through a node port service**

The following service definition declares a service resource of the `NodePort` type:

```
apiVersion: v1
kind: Service
metadata:
  name: database
  namespace: prod2
spec:
  type: NodePort ①
  selector:
    vmtype: linux
  ports:
    - protocol: TCP
      port: 3306 ②
      targetPort: 3306
      nodePort: 30336 ③
```

- ① The `type` parameter declares the service type: `ClusterIP`, `NodePort`, or `LoadBalancer`.
- ② Inside the cluster, the service makes the database application available on port 3306. The service behaves similarly to a `ClusterIP` service.
- ③ The `nodePort` parameter provides the port number that OpenShift must open on all the cluster nodes. The port that you select must be available on all the nodes and in the range 30000 to 32767. If you do not provide the `nodePort` parameter, then OpenShift selects an available port.

Although the **NodePort** service type can be used for TCP and UDP traffic, Red Hat recommends avoiding doing so, for the following reasons:

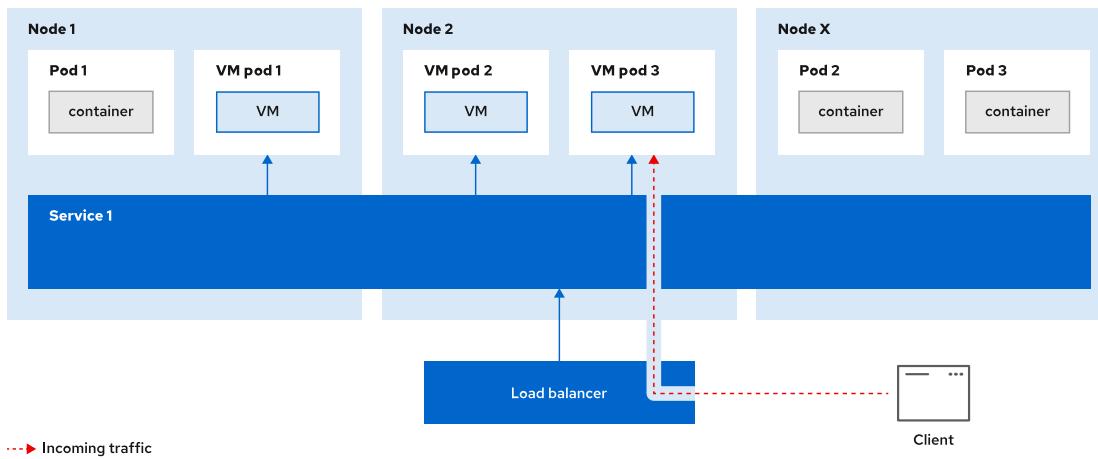
- You must expose the IP addresses of your cluster nodes from outside the cluster. The node IP exposure is a security risk.
- You cannot select a port outside the default range (30000 to 32767).
- In cloud environments, the IP address of cluster nodes might not be permanent. For example, the cloud infrastructure might add or remove nodes to adapt the cluster to the load.
- A NodePort service might require additional network configuration to enable external access to the port.

## Load Balancer Services

You can use a load balancer service to expose the VM network services on a fixed IP address. Unlike using a node port service, the load balancer service exposes a single IP address that does not change when the VM is moved to another node.

Load balancer services require the use of network features that are not available in all environments. For example, cloud providers typically provide their own load balancer services. These services use features that are specific to the cloud provider that hosts the infrastructure. The load balancer resources might require allocating finite resources such as IP addresses.

If you run a Kubernetes cluster on a cloud provider, then the cloud controller manager uses the cloud provider's APIs to configure the required cloud resources for a load balancer service. The cloud features are not available in self-hosted clusters. In self-hosted clusters, you can use a load balancer operator such as MetalLB to configure load balancer services according to the specific settings of your network.



**Figure 3.27: Incoming traffic through a load balancer service**

## Configure Load Balancer Service Resources for Cloud Environments

You can create load balancer services to provide external access to your applications on clusters that are deployed on a cloud provider's infrastructure. When you create a load balancer service, Kubernetes automatically configures an external load balancer on the cloud provider infrastructure.

- When you create a load balancer service on a cluster that is deployed on Amazon Web Services, Kubernetes creates a load balancer on Amazon Elastic Load Balancing.

- When you deploy a cluster on Microsoft Azure, the Kubernetes load balancer services are created on the Azure Load Balancer service.
- When you deploy a cluster on IBM Cloud, the Kubernetes load balancer services are created on the IBM Cloud load balancer service.

**Note**

Each Kubernetes load balancer service creates a load balancer in the cloud provider infrastructure. You can review the cost of each load balancer in the cloud provider infrastructure to estimate the total cost for this type of network services.

## Configure Load Balancer Service Resources for Non-cloud Environments

For clusters that are deployed on bare metal, services of the `LoadBalancer` type behave differently. Kubernetes assigns an IP address to the service from a pool of external addresses that the cluster administrators prepare.

Cluster administrators must configure these external IP addresses so that the network can route them to a cluster node. When the cluster nodes receive packets where the destination IP is the external IP of the service, they forward the traffic to the pods or the virtual machines that are associated with the service.

## The MetalLB Operator

MetalLB is a load balancer implementation for Kubernetes that provides a load balancing service for clusters that do not run on a cloud provider, such as a bare metal clusters or virtualized clusters that are deployed in private environments. MetalLB operates in two modes: layer 2 by using Address Resolution Protocol (ARP) and layer 3 by using Border Gateway Protocol (BGP). These two modes have different properties and requirements:

### Layer 2 (ARP)

The layer 2 mode is set by default when MetalLB is installed and configured with the MetalLB Operator. Layer 2 uses gratuitous ARP packets to advertise the IP address of the load balancer service. The incoming traffic for an IP address that MetalLB manages is forwarded to a node that advertised it via ARP.

### Layer 3 (BGP)

The layer 3 mode uses BGP to advertise the routes for the load balancer services to other BGP peers. MetalLB requires the administrator to configure the Autonomous System (AS) number. When a router receives traffic for the load balancer service, the router picks one of the nodes that advertised the load balancer IP address. The router sends the traffic to that node, and the service proxy for the CNI network plug-in distributes the traffic to all the pods that match the service selector.

MetalLB is an operator that you can install with the Operator Lifecycle Manager. After installing the operator, you must configure MetalLB through its custom resource definitions. In most situations, you must provide MetalLB with an IP address range to control the IP addresses that can be assigned for load balancer services. You must consider the network requirements before installing MetalLB in your OpenShift cluster.

You can install the MetalLB Operator by using either the web console or the command line. After you install the operator, you must create an IP address pool to use it to assign IP address to services. The MetalLB operator creates the custom resources in the `metallb-system` namespace.

## MetallB

This custom resource represents the main instance that the MetallB operator deploys.

```
[user@host ~]$ oc get metallb -n metallb-system
NAME      AGE
metallb   7d ①
```

- ① The main instance of MetallB that the MetallB operator deploys.

## IPAddressPool

The IPAddressPool custom resource contains the range of IP addresses that MetallB can allocate to the load balancer services. You must configure this range according to the network configuration in the environment where the cluster is deployed.

```
[user@host ~]$ oc get ipaddresspool -n metallb-system
NAME          AUTO ASSIGN ... ADDRESSES
gls-metallb-ipaddresspool  true     ... ["192.168.50.20-192.168.50.40"] ①
```

- ① The IP address pool for MetallB contains addresses in the range from 192.168.50.20 to 192.168.50.40.

## L2Advertisement

The L2Advertisement custom resource enables advertising the load balancer IP addresses that the selected pools provide with gratuitous ARP packets on layer 2.

```
[user@host ~]$ oc get l2advertisement -n metallb-system
NAME           IPADDRESSPOOLS ...
gls-metallb-l2advertisement  ["gls-metallb-ipaddresspool"] ... ①
```

- ① MetallB creates layer 2 advertisements for the load balancer IP addresses in the gls-metallb-ipaddresspool IP address pool.

## Using Load Balancer Services with VMs

The VM resource contains the template for the VMI and the virt-launcher pod. The kubevirt.io/domain=vm1 label represents the VM name, and it is passed to the VMI and the virt-launcher pod when the VM is started.

```
[user@host ~]$ oc get vm/vm1 -o yaml | \
grep -C 5 'kubevirt.io/domain'
template:
  metadata:
    creationTimestamp: null
  labels:
    flavor.template.kubevirt.io/small: "true"
    kubevirt.io/domain: vm1 ①
    kubevirt.io/size: small
    os.template.kubevirt.io/rhel8.4: "true"
    workload.template.kubevirt.io/server: "true"
  spec:
    architecture: amd64
```

- ① The kubevirt.io/domain label represents the VM name.

**Note**

If you create a VM by using an instance type, then the `kubevirt.io/domain` label is not passed to the VMI and `virt-launcher` pod. For VMs that are created by using instance types, the `vm.kubevirt.io/name` label is applied to the `virt-launcher` pod.

When the VM is started, the VMI and `virt-launcher` pod resources are created.

```
[user@host ~]$ oc get vm
NAME    AGE   STATUS    READY
vm1     2m    Running   True

[user@host ~]$ oc get vmi
NAME    AGE   PHASE     IP          NODENAME   READY
vm1     1m    Running   10.8.2.92  worker02   True ①

[user@host ~]$ oc get pod
NAME                READY   STATUS    RESTARTS   AGE
virt-launcher-vm1-4htsh   1/1    Running   0          1m ②
```

- ① The VMI resource exists because the VM is started.
- ② The `virt-launcher` pod resource exists because the VM is started.

You can filter the pods in a namespace to get the `virt-launcher` pod that is associated with a VM by using the `kubevirt.io/domain` label.

```
[user@host ~]$ POD=$(oc get pod -l kubevirt.io/domain=vm1 -o name)
[user@host ~]$ echo ${POD}
pod/virt-launcher-vm1-4htsh
```

You can then use the pod name to create a load balancer service resource and expose the port of the VM. You then have direct access to the service port from outside the OpenShift cluster.

The following `oc expose` command uses the `-o yaml` and `--dry-run=client` parameters to create a YAML representation of the load balancer service resource. The output is redirected to the `service.yaml` file.

```
[user@host ~]$ oc expose ${POD} --name=vm1 \
--type=LoadBalancer --selector kubevirt.io/domain=vm1 --port=22 \
-o yaml --dry-run=client | tee service.yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels: ①
    flavor.template.kubevirt.io/small: "true"
    kubevirt.io: virt-launcher
    kubevirt.io/created-by: 7aaaf118c-f174-3eba-9ec5-680cd791a020
    kubevirt.io/domain: vm1 ②
    kubevirt.io/nodeName: worker02
    kubevirt.io/size: small
```

```
os.template.kubevirt.io/rhel8.4: "true"
vm.kubevirt.io/name: vm1
workload.template.kubevirt.io/server: "true"
name: vm1
spec:
  ports:
    - port: 22 3
      protocol: TCP
      targetPort: 22 4
  selector:
    kubevirt.io/domain: vm1 5
  type: LoadBalancer 6
status:
  loadBalancer: {}
```

- 1** The labels are copied from the exposed `virt-launcher` pod.
- 2** The `kubevirt.io/domain` label represents the VM name.
- 3** The connection port on the service IP address.
- 4** The target port is copied from the `port` parameter because it was not specified in the `oc expose` command.
- 5** The service selector that matches the pod labels.
- 6** The service type is set to load balancer.

**Note**

To create a Kubernetes service, you must use the `oc expose` command with a `virt-launcher` pod. You cannot create a Kubernetes service by using the VM or VMI resources.

You can use the YAML resource manifest to create the load balancer service resource.

```
[student@workstation network-lb]$ oc apply -f service.yaml
service/vm1 created
```

MetalLB assigns an external IP address to the load balancer service. The service selector matches the running `virt-launcher` pod that is associated with a VM by using the `kubevirt.io/domain` label. The endpoint has the internal IP address and port of the `virt-launcher` pod.

```
[user@host ~]$ oc get services
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
vm1      LoadBalancer  172.30.157.246  192.168.50.22   22:30826/TCP  10m 1

[user@host ~]$ oc get endpoints
NAME      ENDPOINTS      AGE
vm1      10.8.2.94:22   10m 2
```

- 1** The exposed port 22 is for the SSH service.

- ❷ The endpoint matches the IP address of the `virt-launcher` pod, and the target port 22 forwards the SSH connections.

You can verify that the service is exposed to traffic outside the OpenShift cluster by using the external IP address and port number. The following command verifies that a TCP connection can be established to the load balancer IP address on port 22.

```
[user@host ~]$ nc -vz 192.168.50.22 22
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.50.22:22. ❶
Ncat: 0 bytes sent, 0 bytes received in 0.04 seconds.
```

- ❶ This message indicates that the TCP connection to port 22 was successful.

You can verify that the SSH service of the VM is accessible from a machine outside the OpenShift cluster by using the external IP address of the load balancer service.

```
[user@host ~]$ ssh developer@192.168.50.22
Warning: Permanently added '192.168.50.22' (ED25519) to the list of known hosts.
...output omitted...
[developer@vm1 ~]$
```

## Using Load Balancer Services with Multiple Ports

You can pass only one `--port` parameter when creating the service with the `oc expose` command. You can edit the Kubernetes load balancer service resource manifest to add more ports, if needed. Because the `--port` parameter adds a single port, it does not add a name parameter to the port. However, the manifest requires the name parameter when multiple ports are configured. The following load balancer service exposes multiple ports on an email server VM.

```
apiVersion: v1
kind: Service
metadata:
  labels:
    kubevirt.io/domain: email-server ❶
    ...output omitted...
  name: email-server
spec:
  ports:
    - port: 25 ❷
      name: smtp ❸
      protocol: TCP
      targetPort: 25
    - port: 587 ❹
      name: submission
      protocol: TCP
      targetPort: 587
    - port: 993 ❺
      name: imaps
      protocol: TCP
      targetPort: 993
    - port: 995 ❻
      name: pop3s
      protocol: TCP
      targetPort: 995
```

```
protocol: TCP
targetPort: 995
selector:
  kubevirt.io/domain: email-server 7
type: LoadBalancer 8
status:
loadBalancer: {}
```

- 1** The `kubevirt.io/domain` label represents the VM name.
- 2** Exposes port 25 for SMTP traffic
- 3** The name is optional when only one port is present in the service definition. However, the name is required when adding multiple ports to a service.
- 4** Exposes port 587 for SMTP traffic over TLS.
- 5** Exposes port 993 for IMAP traffic over SSL.
- 6** Exposes port 995 for POP3 traffic over SSL.
- 7** The service selector that matches the pod labels.
- 8** The service type is set to load balancer.

The new service ports are listed after the changes to the load balancer service are applied.

```
[user@host ~]$ oc get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP
email-server   LoadBalancer  172.30.77.133  192.168.50.30
PORT(S)           AGE
25:30935/TCP, 465:32038/TCP, 587:30373/TCP, 993:31431/TCP, 995:31115/TCP ...
```

The service selector matches the VM pod and the endpoints represent the exposed network ports on the VM.

```
[user@host ~]$ oc get endpoints
NAME          ENDPOINTS
email-server   10.9.2.18:25, 10.9.2.18:465, 10.9.2.18:587 + 2 more...    AGE
```

## Enabling SSH over LoadBalancer for a VM in the Web Console

The OpenShift web console provides a feature for adding an SSH service over a load balancer. To verify that this feature is available, a cluster administrator can go to [Virtualization > Overview](#). From the [Settings](#) tab, expand [General Settings](#) and [SSH configuration](#). Ensure that the [SSH over LoadBalancer](#) service is set to on.

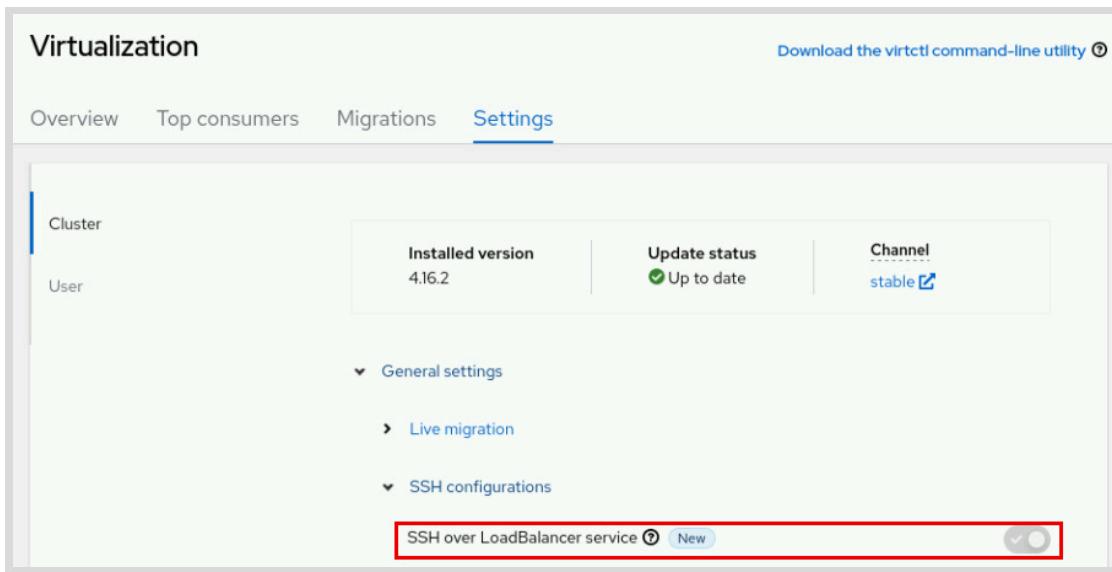


Figure 3.28: Enabling SSH over LoadBalancer

To configure the SSH over LoadBalancer service for a VM, go to **Virtualization > VirtualMachines** and select the VM. From the **VirtualMachine** details page, select the Configuration tab.

Select the SSH tab on the left. Set **SSH service type** to the **SSH over LoadBalancer** option. The dialog provides the command to log in to the VM with SSH.

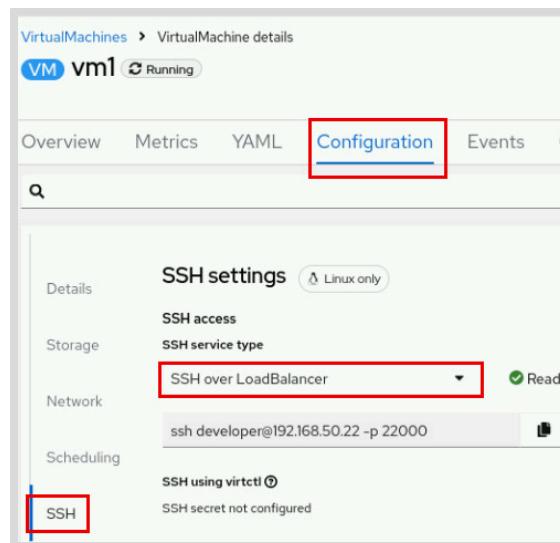


Figure 3.29: VM SSH settings



## References

For more information, refer to the *Load Balancing with MetalLB* chapter in the Red Hat OpenShift Container Platform 4.16 *Networking* documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#load-balancing-with-metallb](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#load-balancing-with-metallb)

For more information, refer to the *Enabling Load Balancer Service Creation by Using the Web Console* chapter in the Red Hat OpenShift Container Platform 4.16 *Virtualization* documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-enabling-load-balancer-service-web\\_virt-post-install-network-config](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-enabling-load-balancer-service-web_virt-post-install-network-config)

For more information about services, refer to the *Kubernetes Services* article at <https://kubernetes.io/docs/concepts/services-networking/service/>

For more information about MetalLB, refer to the *MetalLB Configuration* article at <https://metallb.universe.tf/configuration/>

For more information about MetalLB installation, refer to the *MetalLB on OpenShift* article at <https://metallb.universe.tf/installation/clouds/#metallb-on-openshift-ocp>

## ► Guided Exercise

# Configure SSH Access for a Virtual Machine with Load Balancer Services

Inspect the MetalLB configuration in the classroom environment and configure a load balancer service to establish external SSH access to the virtual machine.

## Outcomes

- Inspect the MetalLB configuration in the classroom environment.
- Create a load balancer service to expose the SSH port of the VM.
- Modify the load balancer service to expose the MySQL port of the VM.
- Confirm that the exposed services are accessible from the workstation machine.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the `network-lb` namespace and starts a virtual machine named `vm1` in that namespace.

```
[student@workstation ~]$ lab start network-lb
```

## Instructions

- 1. Log in to the OpenShift cluster and confirm that the VM is running.
- 1.1. Log in to the OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Get the URL of the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open the OpenShift web console URL in the web browser and select `htpasswd_provider`. Log in as the `admin` user with `redhatocp` as the password.
- 1.4. Go to **Virtualization > VirtualMachines** and select the `network-lb` project. Confirm that the `vm1` machine is running.

Name	Status	Conditions	Node
VM vm1	Running		N worker02

► 2. Inspect the MetalLB configuration in the classroom environment.

- 2.1. Click Operators > Installed Operators and select the network-lb project. Verify that the MetalLB operator is installed.

Name	Managed Namespaces	Status	Provided APIs
MetalLB Operator	All Namespaces	Succeeded Up to date	BGPPeer AddressPool BFDProfile BGPAdvertisement View 4 more...

- 2.2. Switch to the metallb-system project.

```
[student@workstation ~]$ oc project metallb-system
Now using project "metallb-system" on server ...
```

- 2.3. List the MetalLB resources in the metallb-system namespace.

```
[student@workstation ~]$ oc get metallb
NAME      AGE
metallb   7d ①

[student@workstation ~]$ oc get ipaddresspool
NAME          AUTO ASSIGN   ...   ADDRESSES
gls-metallb-ipaddresspool  true       ...  ["192.168.50.20-192.168.50.40"] ②

[student@workstation ~]$ oc get l2advertisement
NAME          IPADDRESSPOOLS
gls-metallb-l2advertisement  ["gls-metallb-ipaddresspool"]  ... ③
```

① The main instance of MetalLB that the MetalLB operator deploys.

- ❷ The IP address pool for MetalLB contains addresses in the range from 192.168.50.20 to 192.168.50.40.
  - ❸ MetalLB creates layer 2 advertisements for the load balancer IP addresses in the `gls-metallb-ipaddresspool` IP address pool.
- ▶ 3. Create a service named `vm1` with the `LoadBalancer` type. The service accepts traffic on TCP port 22 and forwards it to port 22 of the VM. Use the `kubevirt.io/domain=vm1` label for the selector.
- 3.1. Change to the `~/D0316/labs/network-lb` directory.

```
[student@workstation ~]$ cd ~/D0316/labs/network-lb  
[student@workstation network-lb]$
```

- 3.2. Switch to the `network-lb` project.

```
[student@workstation network-lb]$ oc project network-lb  
Now using project "network-lb" on server "https://api.ocp4.example.com:6443".
```

- 3.3. Inspect the VM resource and search for the `kubevirt.io/domain=vm1` label. This label represents the VM name, and it is passed to the VMI and to the `virt-launcher` pod when the VM is started.

```
[student@workstation network-lb]$ oc get vm/vm1 -o yaml | \  
grep -C 5 'kubevirt.io/domain'  
template:  
metadata:  
  creationTimestamp: null  
labels:  
  flavor.template.kubevirt.io/small: "true"  
  kubevirt.io/domain: vm1 ❶  
  kubevirt.io/size: small  
  os.template.kubevirt.io/rhel8.4: "true"  
  workload.template.kubevirt.io/server: "true"  
spec:  
  architecture: amd64
```

- ❶ The `kubevirt.io/domain` label represents the VM name.

- 3.4. List the VM, VMI, and pods in the `network-lb` namespace.

```
[student@workstation network-lb]$ oc get vm  
NAME AGE STATUS READY  
vm1 2m Running True  
  
[student@workstation network-lb]$ oc get vmi  
NAME AGE PHASE IP NODENAME READY  
vm1 1m Running 10.8.2.92 worker02 True ❶
```

```
[student@workstation network-lb]$ oc get pod
NAME           READY   STATUS    RESTARTS   AGE
virt-launcher-vm1-4htsh   1/1     Running   0          1m  ②
```

- ① The VMI resource exists because the VM is started.
- ② The `virt-launcher` pod resource exists because the VM is started.

- 3.5. Get the `virt-launcher` pod name by using the `kubevirt.io/domain=vm1` label selector and save this name in a variable. You use this pod name to create the load balancer service in a later step.

```
[student@workstation network-lb]$ LABEL='kubevirt.io/domain=vm1'
[student@workstation network-lb]$ POD=$(oc get pod -l ${LABEL} -o name)
[student@workstation network-lb]$ echo ${POD}
pod/virt-launcher-vm1-4htsh
```

- 3.6. Use the `oc expose` command with the `-o yaml` and `--dry-run=client` parameters to create the YAML representation of the load balancer service object, which forwards the port 22 traffic to the VM.

You edit this YAML file to add another service port in a later step.

```
[student@workstation network-lb]$ oc expose ${POD} --name=vm1 \
--type=LoadBalancer --selector kubevirt.io/domain=vm1 --port=22 \
-o yaml --dry-run=client | tee service.yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels: ①
    flavor.template.kubevirt.io/small: "true"
    kubevirt.io: virt-launcher
    kubevirt.io/created-by: 7aaf118c-f174-3eba-9ec5-680cd791a020
    kubevirt.io/domain: vm1 ②
    kubevirt.io/nodeName: worker02
    kubevirt.io/size: small
    os.template.kubevirt.io/rhel8.4: "true"
    vm.kubevirt.io/name: vm1
    workload.template.kubevirt.io/server: "true"
  name: vm1
spec:
  ports:
    - port: 22 ③
      protocol: TCP
      targetPort: 22 ④
  selector:
    kubevirt.io/domain: vm1 ⑤
    type: LoadBalancer ⑥
status:
  loadBalancer: {}
```

- ① The labels are copied from the exposed `virt-launcher` pod.

- ❷ The kubevirt.io/domain label represents the VM name.
  - ❸ The connection port on the service IP address.
  - ❹ The target port is copied from the port parameter because it was not specified in the command.
  - ❺ The service selector that matches pod labels.
  - ❻ The service type is set to load balancer.
- 3.7. Create the load balancer service to forward the SSH traffic by using the YAML resource manifest.

```
[student@workstation network-lb]$ oc apply -f service.yaml
service/vm1 created
```

- 3.8. List the service to view the external IP address that MetalLB provides.

```
[student@workstation network-lb]$ oc get services
NAME      TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
vm1       LoadBalancer   172.30.157.246  192.168.50.22   22:30826/TCP  1m
```



### Note

The external IP address is in the range that MetalLB defines, and might differ from the address that is shown here.

- 3.9. Review the VMI IP address and the endpoint which the service connects to.

```
[student@workstation network-lb]$ oc get vmi
NAME      AGE      PHASE      IP           NODENAME      READY
vm1       5m       Running    10.8.2.92    worker02     True

[student@workstation network-lb]$ oc get endpoints
NAME      ENDPOINTS      AGE
vm1      10.8.2.92:22   2m  ❶
```

- ❶ The endpoint contains the IP address and target port of the pods that match the service selector.
- 4. Inspect the TCP connection to port 22 and connect via SSH to the VM from the workstation machine.
- 4.1. Check whether the external IP address of the load balancer service is reachable from the workstation machine, which is outside the OpenShift cluster. Confirm that you can establish a TCP connection to the load balancer IP address on port 22.

```
[student@workstation network-lb]$ nc -vz 192.168.50.22 22
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.50.22:22. ❶
Ncat: 0 bytes sent, 0 bytes received in 0.04 seconds.
```

- ➊ This message indicates that the TCP connection to port 22 was successful.

4.2. Verify that you can connect via SSH to the load balancer service IP address.

```
[student@workstation network-lb]$ ssh developer@192.168.50.22
Warning: Permanently added '192.168.50.22' (ED25519) to the list of known hosts.
...output omitted...
[developer@vm1 ~]$
```



### Note

The connection command does not ask for a password, because it authenticates with an SSH key.

4.3. Exit the SSH session of the vm1 machine.

```
[developer@vm1 ~]$ exit
Connection to 192.168.50.22 closed.
[student@workstation network-lb]$
```

- ▶ 5. Shut down the VM, and verify that the SSH connection cannot be established, even if the load balancer service is still active.

- 5.1. Go to **Virtualization > VirtualMachines** and select the **network-lb** project, if needed. Click the vertical ellipsis icon : at the end of the **vm1** row, and click **Stop**. Wait until the VM status is **Stopped**.

- 5.2. List the virtual machine, virtual machine instance, and pod resources. The **vmi** and **pod** resources do not exist, because the VM is stopped.

```
[student@workstation network-lb]$ oc get vm
NAME    AGE    STATUS      READY
vm1     10m   Stopped    False  ①

[student@workstation network-lb]$ oc get vmi
No resources found in network-lb namespace. ②

[student@workstation network-lb]$ oc get pod
No resources found in network-lb namespace. ③
```

- ➊ The VM state is stopped.
  - ➋ The VMI resource does not exist, because the VM is stopped.
  - ➌ The `virt-launcher` pod resource does not exist, because the VM is stopped.
- 5.3. Verify that no endpoint is associated with the load balancer service, which means that the service selector did not match any pod.

```
[student@workstation network-lb]$ oc get services
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
vm1       LoadBalancer  172.30.157.246  192.168.50.22   22:30826/TCP  5m

[student@workstation network-lb]$ oc get endpoints
NAME      ENDPOINTS      AGE
vm1      <none>      5m  ➌
```

- ➊ No endpoints exist, because the service selector did not match any pods.
- 5.4. Confirm that you cannot establish a TCP connection to the load balancer IP address on port 22.

```
[student@workstation network-lb]$ nc -vz 192.168.50.22 22
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: No route to host. ➌
```

- ➊ This message indicates that a TCP connection cannot be established.



#### Note

The `Connection refused` message also indicates that a TCP connection cannot be established.

- 5.5. Verify that you cannot establish a SSH connection to the VM. Although the `ssh` command tries to establish the connection, it eventually fails and reports the error condition.

```
[student@workstation network-lb]$ ssh developer@192.168.50.22
ssh: connect to host 192.168.50.22 port 22: No route to host ➌
```

- ➊ This message indicates that an SSH connection cannot be established.



#### Note

The `Connection refused` message also indicates that an SSH connection cannot be established.

- 6. Power on the VM and verify that you can access it via SSH from the `workstation` machine.
- 6.1. Go to **Virtualization > Virtual Machines**. Click the vertical ellipsis icon at the end of the `vm1` row, and then click **Start**. Wait until the VM status is **Running**.

Name	Status	Conditions	Node
VM vm1	○ Stopped	Ready=False	-

⋮

Start  
Restart

6.2. Verify that you can connect via SSH to the load balancer service IP address.

```
[student@workstation network-lb]$ ssh developer@192.168.50.22
Warning: Permanently added '192.168.50.22' (ED25519) to the list of known hosts.
...output omitted...
[developer@vm1 ~]$
```

6.3. Exit the SSH session of the `vm1` machine.

```
[developer@vm1 ~]$ exit
Connection to 192.168.50.22 closed.
[student@workstation network-lb]$
```

- ▶ 7. Add a second port to the load balancer service to connect to the database service from the `workstation` machine.

7.1. List the services and endpoints. The load balancer service is configured to forward only SSH connections in port 22 of the external IP address that MetalLB provides.

```
[student@workstation network-lb]$ oc get services
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
vm1       LoadBalancer  172.30.157.246  192.168.50.22  22:30826/TCP  10m  ①

[student@workstation network-lb]$ oc get endpoints
NAME      ENDPOINTS      AGE
vm1       10.8.2.94:22    10m  ②
```

- ① The exposed port 22 is for the SSH service.
- ② The endpoint matches the IP address of the `virt-launcher` pod and the target port 22 forwards the SSH connections.

7.2. Edit the `service.yaml` file and add the `ssh` name to the connection on port 22.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    ...output omitted...
```

```
name: vm1
spec:
  ports:
    - port: 22
      protocol: TCP
      targetPort: 22
      name: ssh ①
  selector:
    kubevirt.io/domain: vm1
  type: LoadBalancer
status:
  loadBalancer: {}
```

- ① The name is optional when only one port is present in the service definition. However, you add another service port in a later step.

- 7.3. Add the parameters for the MySQL service on port 3306. Then, save the changes to the `service.yaml` file and close the editor.

```
...output omitted...
spec:
  ports:
    - port: 22
      protocol: TCP
      targetPort: 22
      name: ssh
    - port: 3306
      protocol: TCP
      targetPort: 3306
      name: mysql
  selector:
    kubevirt.io/domain: vm1
  type: LoadBalancer
...output omitted...
```



### Note

The `~/D0316/solutions/network-lb/service.yaml` file contains the correct configuration, and you can use it for comparison.

You can ignore the differences in the following labels:

- `kubevirt.io/created-by`
- `kubevirt.io/nodeName`

- 7.4. Apply the changes to the load balancer service resource.

```
[student@workstation network-lb]$ oc apply -f service.yaml
service/vm1 configured
```

- 7.5. List the services and endpoints. Notice that the endpoint for port 3306 was added.

```
[student@workstation network-lb]$ oc get services
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      ...
vm1       LoadBalancer  172.30.157.246  192.168.50.22  22:30826/TCP,3306:31186/TCP  ①

[student@workstation network-lb]$ oc get endpoints
NAME      ENDPOINTS      AGE
vm1       10.8.2.94:3306,10.8.2.94:22  15m  ②
```

- ① Two ports are now associated with the service, one for the SSH service and another for the MySQL service.
  - ② The endpoint contains the IP address and target port of the pods that match the service selector.
- 7.6. Use the MySQL client in the `workstation` machine to connect to the database service in the VM by using the load balancer IP address. The MySQL service uses port 3306 by default.

```
[student@workstation network-lb]$ mysql -h 192.168.50.22 \
-u devuser -p'developer' sakila
...output omitted...

mysql>
```

- 7.7. List the tables of the `sakila` database.

```
mysql> SHOW TABLES ;
+-----+
| Tables in sakila           |
+-----+
...output omitted...
+-----+
23 rows in set (0.00 sec)
```

- 7.8. Close the connection to the database.

```
mysql> quit
Bye
[student@workstation network-lb]$
```

- 7.9. Change to the student `HOME` directory.

```
[student@workstation network-lb]$ cd
[student@workstation ~]$
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-lb
```

## ▶ Lab

# Configure Kubernetes Networking for Virtual Machines

Configure services, routes, and network policies to expose external access for a web application and for its database that is hosted on another virtual machine in the same project.

## Outcomes

- Create ClusterIP services to expose the web application and the database service.
- Expose the web application by using a route.
- Create a network policy to control the connections to the database service.
- Expose the SSH service of the web application VM by using a load balancer service.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start network-review
```

## Instructions

1. From the command line on the **workstation** machine, use the `oc` command to log in to the Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.  
Open a web browser and log in to the Red Hat OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`.  
Confirm that the `mariadb-server` and the `front-web` VMs are running in the `network-review` project.
2. Create the **database** service with the **ClusterIP** type, which enables access to the database that is running inside the `mariadb-server` VM. Add the `tier=backend` label to the `mariadb-server` VM.  
This service must listen on TCP port 3306 and forward the traffic to port 3306 of the `tier=backend` pod.  
The `lab` command prepared some examples of resource files in the `~/D0316/solutions/network-review` directory.  
To verify your work, run the `~/D0316/labs/network-review/testdb.sh` script. The script tries to connect to the database through the database service.
3. Add the `allowed=database` label to the `network-review` namespace, and create a network policy for the `mariadb-server` VM. The policy must allow only the ingress connections to TCP port 3306 from the `network-review` namespace.

Create a network policy named `allow-database`. It must target the pods with the `tier=backend` label and allow ingress connections to TCP port 3306 from the namespaces with the `allowed=database` label.

To test the database connection, run the `~/D0316/labs/network-review/testdb.sh` script. The script accepts a namespace as a parameter. The script must succeed when you run `~/D0316/labs/network-review/testdb.sh network-review` and it must fail when you run `~/D0316/labs/network-review/testdb.sh default`.

4. Create a service with the `ClusterIP` type, which enables access to the web application that is running inside the `front-web` VM. To create this service, add the `tier=frontend` label to the `front-web` VM.

Create the service named `web`. The service must listen on TCP port 8080 and forward the traffic to port 80 to the pod with the `tier=frontend` label.

To verify your work, run the `~/D0316/labs/network-review/testweb.sh` script. The script tries to connect to the web application through the service.

5. From the web console, create a route to access the web application that is running inside the `front-web` VM by using the `http://intranet-dev.apps.ocp4.example.com` URL.

Complete the form by using the following information:

Field	Value
Configure via	Form view
Name	<code>web</code>
Hostname	<code>intranet-dev.apps.ocp4.example.com</code>
Path	<code>/</code>
Service	<code>web</code>
Target port	<code>8080 → 80(TCP)</code>
Secure route	<code>Unchecked</code>

To confirm that the web application can connect to the database, access `http://intranet-dev.apps.ocp4.example.com/cgi-bin/dbtest`. The page displays a `PASS` message when the connection is successful.

6. Create a load balancer service named `ssh-web` to access the SSH service of the `front-web` VM from outside the OpenShift cluster. The SSH service runs on TCP port 22. Use the `kubevirt.io/domain=front-web` label for the pod selector.

## Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade network-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-review
```

## ► Solution

# Configure Kubernetes Networking for Virtual Machines

Configure services, routes, and network policies to expose external access for a web application and for its database that is hosted on another virtual machine in the same project.

## Outcomes

- Create ClusterIP services to expose the web application and the database service.
- Expose the web application by using a route.
- Create a network policy to control the connections to the database service.
- Expose the SSH service of the web application VM by using a load balancer service.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start network-review
```

## Instructions

1. From the command line on the `workstation` machine, use the `oc` command to log in to the Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.  
Open a web browser and log in to the Red Hat OpenShift web console at `https://console.openshift-console.apps.ocp4.example.com`.  
Confirm that the `mariadb-server` and the `front-web` VMs are running in the `network-review` project.
  - 1.1. From the command line, log in to your OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Set the `network-review` project as the active project.

```
[student@workstation ~]$ oc project network-review
Now using project "network-review" on server "https://api.ocp4.example.com:6443".
```

- 1.3. Open a web browser and go to the `https://console-openshift-console.apps.ocp4.example.com` URL. Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- 1.4. Go to **Virtualization > VirtualMachines** and then select the `network-review` project. Confirm that the `mariadb-server` and the `front-web` VMs are running.
2. Create the database service with the `ClusterIP` type, which enables access to the database that is running inside the `mariadb-server` VM. Add the `tier=backend` label to the `mariadb-server` VM.

This service must listen on TCP port 3306 and forward the traffic to port 3306 of the `tier=backend` pod.

The `lab` command prepared some examples of resource files in the `~/D0316/solutions/network-review` directory.

To verify your work, run the `~/D0316/labs/network-review/testdb.sh` script. The script tries to connect to the database through the database service.

- 2.1. Go to the `YAML` tab of the `mariadb-server` VM, add the `tier=backend` label to the `.spec.template.metadata.labels` path, and click `Save`.

```
...output omitted...
spec:
  dataVolumeTemplates:
    ...output omitted...
  template:
    metadata:
      creationTimestamp: null
      labels:
        tier: backend
        flavor.template.kubevirt.io/small: "true"
        kubevirt.io/domain: mariadb-server
        kubevirt.io/size: small
    ...output omitted...
```

- 2.2. Restart the VM to re-create the VMI that includes the new label. From the web console, click **Actions > Restart**.
- 2.3. Click **Overview**, and confirm that the VM is running.
- 2.4. Go to **Networking > Services**, ensure that the `network-review` project is selected, and then click **Create Service**. Complete the YAML file with the following content and then click **Create**.

```
apiVersion: v1
kind: Service
metadata:
  name: database
  namespace: network-review
spec:
  type: ClusterIP
  selector:
    tier: backend
  ports:
```

```
- protocol: TCP
  port: 3306
  targetPort: 3306
```

**Note**

The lab command prepared the ~/D0316/solutions/network-review/service-database.yaml file so that you can compare it with your version.

- 2.5. Change to the command-line window and confirm that the database service has an active endpoint.

```
[student@workstation ~]$ oc get service database
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
database   ClusterIP  172.30.21.6    <none>          3306/TCP    ...
[student@workstation ~]$ oc get endpoints database
NAME      ENDPOINTS      AGE
database  10.9.2.61:3306  ...
```

- 2.6. Run the testdb.sh script to verify your work.

```
[student@workstation ~]$ ~/D0316/labs/network-review/testdb.sh
Testing database connection from network-review (be patient)...
Connection successful
```

3. Add the allowed=database label to the network-review namespace, and create a network policy for the mariadb-server VM. The policy must allow only the ingress connections to TCP port 3306 from the network-review namespace.  
Create a network policy named allow-database. It must target the pods with the tier=backend label and allow ingress connections to TCP port 3306 from the namespaces with the allowed=database label.  
To test the database connection, run the ~/D0316/labs/network-review/testdb.sh script. The script accepts a namespace as a parameter. The script must succeed when you run ~/D0316/labs/network-review/testdb.sh network-review and it must fail when you run ~/D0316/labs/network-review/testdb.sh default.

- 3.1. From the command line, add the allowed=database label to the network-review namespace.

```
[student@workstation ~]$ oc label namespace network-review allowed=database
namespace/network-review labeled
```

- 3.2. From the web console, go to **Networking > NetworkPolicies**, ensure that the network-review project is selected, click **Create NetworkPolicy**, and then click **YAML view**. Complete the file with the following content and then click **Create**.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-database
  namespace: network-review
```

```
spec:  
  podSelector:  
    matchLabels:  
      tier: backend  
  ingress:  
    - from:  
      - namespaceSelector:  
          matchLabels:  
            allowed: database  
    ports:  
      - port: 3306  
        protocol: TCP
```

**Note**

The lab command prepared the ~/D0316/solutions/network-review/network-policy.yaml file so that you can compare it with your version.

- 3.3. Switch to the command line, and run the `testdb.sh` script with the `network-review` parameter to test that the database connection is successful.

This script creates a pod in the specified namespace to test the connection to the database service.

```
[student@workstation ~]$ ~/D0316/labs/network-review/testdb.sh network-review  
Testing database connection from network-review (be patient)...  
Connection successful
```

- 3.4. Run the `testdb.sh` script with the `default` parameter to test that the database connection fails.

```
[student@workstation ~]$ ~/D0316/labs/network-review/testdb.sh default  
Testing database connection from default (be patient)...  
Cannot connect ①
```

- ① The script fails because the network policy does not allow the connection.
4. Create a service with the `ClusterIP` type, which enables access to the web application that is running inside the `front-web` VM. To create this service, add the `tier=frontend` label to the `front-web` VM.  
Create the service named `web`. The service must listen on TCP port 8080 and forward the traffic to port 80 to the pod with the `tier=frontend` label.  
To verify your work, run the `~/D0316/labs/network-review/testweb.sh` script. The script tries to connect to the web application through the service.

- 4.1. Go to `Virtualization > VirtualMachines` and then select the `front-web` VM. Go to the `YAML` tab, and add the `tier=frontend` label to the `.spec.template.metadata.labels` path of the `front-web` VM. Click `Save`.

```
...output omitted...  
spec:  
  dataVolumeTemplates:  
  ...output omitted...
```

```
template:  
  metadata:  
    creationTimestamp: null  
    labels:  
      tier: frontend  
      flavor.template.kubevirt.io/small: "true"  
      kubevirt.io/domain: front-web  
      kubevirt.io/size: small  
      ...output omitted...
```

- 4.2. Restart the VM to re-create the VMI that includes the new label. From the web console, click **Actions > Restart**.
- 4.3. Click **Overview**, and confirm that the VM is running.
- 4.4. Go to **Networking > Services**, and then click **Create Service**. Complete the YAML file with the following content and then click **Create**.

```
apiVersion: v1  
kind: Service  
metadata:  
  name: web  
  namespace: network-review  
spec:  
  type: ClusterIP  
  selector:  
    tier: frontend  
  ports:  
    - protocol: TCP  
      port: 8080  
      targetPort: 80
```



### Note

The lab command prepared the ~/D0316/solutions/network-review/service-web.yaml file so that you can compare it with your version.

- 4.5. Change to the command line and confirm that the web service has an active endpoint.

```
[student@workstation ~]$ oc get service web  
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE  
web       ClusterIP  172.30.238.254  <none>          8080/TCP    ...  
  
[student@workstation ~]$ oc get endpoints web  
NAME      ENDPOINTS      AGE  
web       10.9.2.63:80    ...
```



### Note

The IP addresses in the preceding output is different on your system.

- 4.6. Run the `testdb.sh` script to test that the database connection is successful.

```
[student@workstation ~]$ ~/DO316/labs/network-review/testweb.sh  
Testing web application from network-review (be patient)...  
Connection successful
```

5. From the web console, create a route to access the web application that is running inside the front-web VM by using the `http://intranet-dev.apps.ocp4.example.com` URL.

Complete the form by using the following information:

Field	Value
Configure via	Form view
Name	web
Hostname	<code>intranet-dev.apps.ocp4.example.com</code>
Path	/
Service	web
Target port	8080 → 80(TCP)
Secure route	<i>Unchecked</i>

To confirm that the web application can connect to the database, access `http://intranet-dev.apps.ocp4.example.com/cgi-bin/dbtest`. The page displays a PASS message when the connection is successful.

- 5.1. From the web console, go to **Networking > Routes**. Select the `network-review` project and then click **Create Route**. Complete the form, and click **Create**.
- 5.2. From the command line, use the `curl` command, and confirm that you can access the web application from outside the cluster.

```
[student@workstation ~]$ curl http://intranet-dev.apps.ocp4.example.com/  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Hello, World!</title>  
  </head>  
  <body>  
    <p>Hello, World! Welcome to Red Hat Training.</p>  
  </body>  
</html>
```

- 5.3. Use the `curl` command with the `http://intranet-dev.apps.ocp4.example.com/cgi-bin/dbtest` URL to confirm that the web application can access the database.

```
[student@workstation ~]$ curl \
http://intranet-dev.apps.ocp4.example.com/cgi-bin/dbtest
<html><head><title>Database Test</title></head><body>
<p style="color:green;">PASS</p>
</body></html>
```

6. Create a load balancer service named `ssh-web` to access the SSH service of the `front-web` VM from outside the OpenShift cluster. The SSH service runs on TCP port 22. Use the `kubevirt.io/domain=front-web` label for the pod selector.
  - 6.1. List the VMI and the `virt-launcher` pod that match the `kubevirt.io/domain=front-web` label selector.

```
[student@workstation ~]$ oc get vmi,pod -l kubevirt.io/domain=front-web
NAME                                     AGE   PHASE      IP          ...
virtualmachineinstance.kubevirt.io/front-web ...   Running    10.9.2.63  ...
...
NAME           READY   STATUS    RESTARTS   AGE
pod/virt-launcher-front-web-pcr8h     1/1     Running   0          ...
```

- 6.2. Use the label selector to get the pod name. You use this pod name to create the load balancer service in the next step.

```
[student@workstation ~]$ LABEL='kubevirt.io/domain=front-web'
[student@workstation ~]$ POD=$(oc get pod -l ${LABEL} -o name)
[student@workstation ~]$ echo ${POD}
pod/virt-launcher-front-web-pcr8h
```

- 6.3. Use the `oc expose` command to create a load balancer service to access the SSH port.

```
[student@workstation ~]$ oc expose ${POD} --type=LoadBalancer \
--name=ssh-web --selector kubevirt.io/domain=front-web --port=22
service/ssh-web exposed
```

- 6.4. Confirm that the `mariadb-server` load balancer service is created and has an associated endpoint.

```
[student@workstation ~]$ oc get service ssh-web
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
ssh-web   LoadBalancer  172.30.227.154  192.168.50.22  22:30503/TCP  ...
...
[student@workstation ~]$ oc get endpoints ssh-web
NAME      ENDPOINTS      AGE
ssh-web   10.9.2.63:22  ...
```

- 6.5. Check whether the external IP address of the load balancer service is reachable from the `workstation` machine, which is outside the OpenShift cluster. Confirm that you can connect via SSH to the load balancer IP address on port 22.

```
[student@workstation ~]$ ssh developer@192.168.50.22
Warning: Permanently added "192.168.50.22" (ED25519) to the list of known hosts.
...output omitted...
Last login: Mon Apr  8 12:17:21 2024
[developer@front-web ~]$
```



### Note

Instead of a password, the connection command authenticates with an SSH key.

6.6. Exit the SSH session to return to the **workstation** machine.

```
[developer@front-web ~]$ exit
logout
Connection to 192.168.50.22 closed.
[student@workstation ~]$
```



### Note

Confirm that you are in the `/home/student` directory on the **workstation** machine before you use the `lab` command to complete the exercise.

## Evaluation

As the **student** user on the **workstation** machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade network-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-review
```

# Summary

---

- Kubernetes assigns a different IP address each time that it starts a pod.
- Services assign a unique IP address to a set of pods.
- Services use labels to identify pods.
- Applications retrieve the IP address of a service by using the `servicename.namespace.svc.cluster.local` DNS name.
- Network policies filter the traffic to pods.
- Services of the `ClusterIP` type are accessible only from pods that are running inside the cluster.
- Services of the `NodePort` type expose a network port on all the cluster nodes. External clients can connect to that port on any cluster node to reach the service.
- Services of the `LoadBalancer` type rely on the cloud provider load balancer to forward the incoming traffic to the service.
- Red Hat OpenShift routes enable HTTP, HTTPS, and TLS external traffic to reach a service.
- Kubernetes `ingress` resources are similar to Red Hat OpenShift routes.
- Because virtual machines are running inside pods, you can use all the services and route features to make them available on the network.



## Chapter 4

# Connecting Virtual Machines to External Networks

### Goal

Configure node networking to connect virtual machines and nodes to networks outside of the cluster by using Multus CNI plug-ins and the NMState operator.

### Sections

- Advanced Networking with Multus CNI (and Quiz)
- Configuring Multihomed Nodes and Virtual Machines (and Guided Exercise)

### Lab

- Connect Virtual Machines to External Networks

# Advanced Networking with Multus CNI

---

## Objectives

- Describe how the Multus CNI plug-in enables virtual machines to have more than one network interface that might be connected to different software-defined link-layer or IP networks.

## Kubernetes Networking

The Kubernetes network model implements an IP address for every pod in the cluster so that links between pods and mapping of container ports to node ports are no longer necessary. The Kubernetes network model imposes these requirements on any networking implementation:

- Pods in a cluster can communicate without network address translation (NAT).
- Pods and agents (system processes) can communicate when they are on the same node.

Kubernetes assigns IP addresses at the pod level, which means that all containers within a pod can reach the `localhost` address on each other's ports.

Pods are unaware of the existence of node ports. However, you can request node ports that forward to a pod.

See the references section for more information about the Kubernetes network model.

## Multus in Red Hat OpenShift Virtualization

Container runtimes can implement the Kubernetes network model in many ways, including through custom resources (CR) and Container Network Interface (CNI) plug-ins. Similar to a wrapper, the Multus CNI plug-in calls other CNI plug-ins for advanced networking functions, such as attaching multiple network interfaces to pods in an OpenShift cluster.

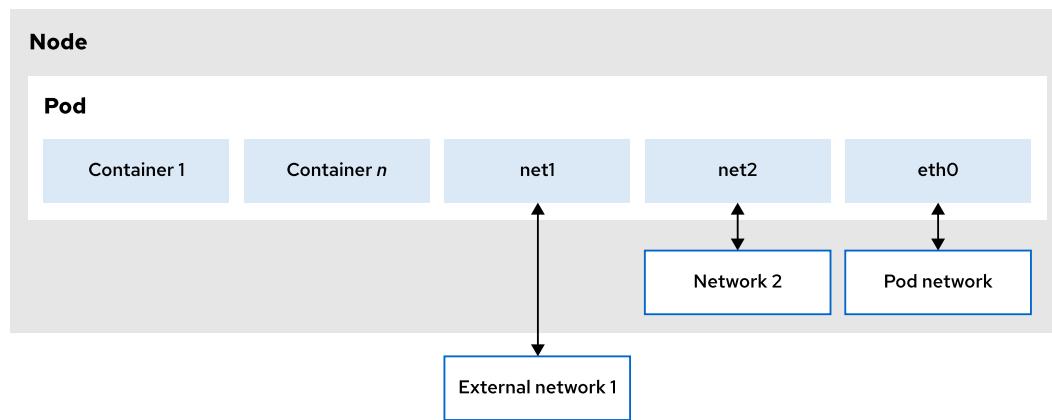
Red Hat OpenShift uses the Multus CNI plug-in to chain other CNI plug-ins. With Multus CNI, you can configure additional networks alongside the default pod network during and after your OpenShift cluster installation. Attaching multiple networks to a VM is called *multihoming*.

Although you can add another network to pods, all pods must contain an `eth0` interface that is attached to the default pod network to maintain connectivity across the cluster. You can view the interfaces that are attached to a pod by using the `ip address` command in that pod:

```
[user@host ~]$ oc exec -it pod_name -- ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
    ...output omitted...
2: eth0@if34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP
    group default
    ...output omitted...
3: net1@if35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default
    ...output omitted...
```

Additional network interfaces that use the Multus CNI plug-in follow the `netN` naming convention.

The following figure shows how additional network interfaces, which are attached by the Multus CNI plug-in, coexist within a pod:



**Figure 4.1: Additional pod interfaces that are attached by Multus CNI**

## Multus CNI Use Cases

Attaching additional networks can be helpful in situations where network isolation is needed. The following use cases demonstrate some reasons for network isolation.

### Performance

You can increase the performance of a network-intensive workload by isolating data and control planes to different node interfaces.

### Security

You can isolate sensitive traffic onto a network plane that is managed specifically for security considerations, such as private data that must not be shared between tenants or customers.

### External network

You can connect pods and VMs to networks outside the cluster.

## Implementing Multus

OpenShift Virtualization uses Custom Resource Definitions (CRDs) and CNI plug-ins, such as Multus CNI, to provide advanced networking functions for your VMs. In OpenShift, the Cluster Network Addons Operator (CNAO) manages additional network configurations for VMs that are based on the Multus CNI plug-in. The Multus CNI plug-in is implemented through the `NetworkAttachmentDefinition` CR; the Network Plumbing Working Group leads the development.

A network attachment definition is a namespaced object that exposes existing layer-2 network devices, such as bridges and switches, to VMs and pods. Red Hat recommends using the Cluster Network Operator (CNO) to centralize managing additional networks for pods in clusters where the OpenShift Virtualization operator is not deployed. For VMs in OpenShift Virtualization, you do not need to edit the CNO to create a network attachment definition.

To attach additional network interfaces to a pod or a VM, you must create a network attachment definition that defines the CNI plug-in configuration to use for the additional interface.

OpenShift provides the following CNI plug-ins for the Multus CNI plug-in to chain:

**bridge**

Configure an additional bridge-based network to enable pods on the same host to communicate with each other and with the host.

A Linux bridge is required to attach VMs to multiple networks. You use a Linux bridge to connect the network interfaces of VMs and pods to a physical network interface on a host or a node. Pods and VMs on a bridge network use the same broadcast domain and subnet as the host's physical network interface.

If the host has access to networks that are external to the OpenShift cluster, then bridge networks enable VMs and pods in the OpenShift cluster to communicate with resources that exist on that external network, such as file shares or databases.

**host-device**

Configure an additional host-device network to enable VMs and pods to have exclusive access to a physical Ethernet network device on the host system.

Direct, exclusive access to a physical network device is beneficial for applications that require high throughput and low latency. Applications that process large volumes of data packets, such as in telecommunications environments, also benefit from this connection type.

**ipvlan**

Configure an additional IPVLAN-based network to enable pods and VMs on a host to have unique IP addresses, but have the same MAC address as the host's physical network. The IP range of the host's physical device is shared with the pods and VMs that are connected to an IPVLAN-based network. An IPVLAN-based network simplifies network administration because you do not need to create or manage unique MAC addresses.

**macvlan**

Configure an additional MACVLAN-based network to enable pods and VMs on a host to communicate with other hosts and their pods and VMs by using a physical network interface. Unlike an IPVLAN-based network, each pod and VM that are attached to a MACVLAN additional network have a unique MAC address. You can use a MACVLAN-based network for applications that require a unique MAC address for operations or licensing. MACVLAN-based networks can also be used to ensure that your pods and VMs are compatible with systems that expect unique MAC addresses, such as network monitoring tools and DHCP.

IPVLAN- and MACVLAN-based networks are commonly used in high-density container environments that require scalable networking, such as telecommunications and Internet of Things (IoT) applications.

**SR-IOV**

Configure an additional SR-IOV network to enable pods to attach to a virtual function interface on SR-IOV-capable hardware on the host system.

These plug-ins require satisfying any limitations or requirements for the additional network. Therefore, installing each plug-in differs for configuration. Each plug-in requires a network attachment definition that defines the configuration. Before you configure additional networks for VMs, a Linux bridge must be configured and attached to your VM-workload nodes by applying a configuration manifest to the cluster.

**Note**

Node networking is explained in more detail later in this course.

You can create network attachment definitions for VMs from the command line with a YAML manifest or from the OpenShift web console.

To configure a network attachment definition from the OpenShift web console, navigate to **Networking > Network Attachment Definitions** and then complete the form, or use the YAML editor to define the additional network. The following figure provides an example of a network attachment definition for a Linux bridge:

**Note**

If VLAN IDs are configured on your additional network, then you can specify the ID numbers in the **VLAN Tag Number** field. Otherwise, you can leave the field empty.

Configuring a Linux bridge network attachment definition	
Name	secondary-network
Description	My secondary network
Network type	CNV Linux bridge
Bridge name	br1
VLAN Tag Number	0
MAC spoof check	Checked

You can also configure the bridge plug-in with a YAML manifest. The following YAML definition shows the configuration of a bridge plug-in:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: bridge-dev ①
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/bridge-dev ②
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "bridge-dev", ③
    "type": "cnv-bridge", ④
    "bridge": "dev-bridge", ⑤
    "macspoochk": true, ⑥
    "vlan": 0 ⑦
  }'
```

- ① The name for the network attachment definition.
- ② **Optional:** An annotation key-value pair to determine the node selection for VM scheduling and execution. In this example, the `bridge-dev` bridge is configured on some cluster nodes. VMs that are attached to this network are scheduled only on a node with the `bridge-dev` bridge.

- ③ The name for the configuration, which Red Hat recommends should match the name of the network attachment definition.
- ④ The name of the CNI plug-in to use. Change this field only if using a different CNI plug-in.
- ⑤ The name of the Linux bridge that is configured on the node.
- ⑥ **Optional:** When set to `true`, the MAC address of the pod or a guest interface cannot be changed, and only one MAC address exits the pod.
- ⑦ **Optional:** The VLAN tag of the network.

Apply the YAML manifest to your cluster with the `oc create -f` or `oc apply -f` commands.



#### Note

The network attachment definition must be in the same namespace as the virtual machine.

```
[user@host ~]$ oc create -f linux-bridge-dev.yml -n multus-test
networkattachmentdefinition.k8s.cni.cncf.io/bridge-dev created
```

You can verify the creation of the network attachment definition by executing the `oc get net-attach-def -n namespace` command.

```
[user@host ~]$ oc get net-attach-def -n multus-test
NAME          AGE
bridge-dev    54s
```



#### References

For more information about the Kubernetes network model, refer to the *Services, Load Balancing, and Networking* documentation at <https://kubernetes.io/docs/concepts/services-networking/#the-kubernetes-network-model>

For more information about Multus and configuring additional network, refer to the *Multiple Networks* chapter in the Red Hat OpenShift Container Platform 4.16 *Networking* documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#multiple-networks](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#multiple-networks)

For more information about Multus, refer to the *Demystifying Multus* article at <https://www.redhat.com/en/blog/demystifying-multus>

## ► Quiz

# Advanced Networking with Multus CNI

Choose the correct answers to the following questions:

- ▶ **1. Which requirement does the Kubernetes network model impose?**
  - a. Communication between pods requires Network Address Translation (NAT).
  - b. Agents on a node can communicate with all pods on the same node.
  - c. Links are required to enable communication between pods in the cluster.
- ▶ **2. Which two options are the names of additional network interfaces that are attached to pods? (Choose two.)**
  - a. eth0
  - b. net1
  - c. eth1
  - d. net2
- ▶ **3. True or False: You can connect pods and virtual machines to networks that are external to the cluster.**
  - a. True
  - b. False
- ▶ **4. Which statement about additional network interfaces and the Multus Container Network Interface (CNI) plug-in is true?**
  - a. The Cluster Network Addons Operator manages additional network configurations for VMs that are based on the Multus CNI plug-in.
  - b. You can create network attachment definitions for VMs through the command line with a YAML manifest or through the OpenShift web console.
  - c. The Multus CNI plug-in is implemented through the `NetworkAttachmentDefinition` custom resource.
  - d. All of the above.
- ▶ **5. Which plug-in does the Multus CNI use to connect a VM to multiple networks?**
  - a. ip-vlan
  - b. host-device
  - c. bridge
  - d. mac-vlan

► **6. What is the role of the Cluster Network Addons Operator (CNAO) in Red Hat OpenShift?**

- a. Deploys and configures the OpenShift Virtualization operator.
- b. Directly manages the physical network interfaces on all nodes.
- c. Manages additional network configurations for VMs based on the Multus CNI plug-in.
- d. Creates and manages all primary network configurations for pods.

► **7. What is the purpose of a network attachment definition in OpenShift Virtualization?**

- a. Defines the CNI plug-in configuration to use for additional network interfaces on a pod or VM.
- b. Manages the storage resources for VMs and pods.
- c. Configures the primary network interface for all cluster nodes.
- d. Automatically deploys new VMs across the cluster.

► **8. Which network type enables VMs and pods to have unique IP addresses but to use the same MAC address as the host's physical network?**

- a. SR-IOV
- b. ipvlan
- c. macvlan
- d. host-device

► **9. You are a network administrator who is tasked with configuring a new VM in an OpenShift cluster to ensure that it can communicate with external networks. You decide to use a bridge network for this purpose. What is the primary reason for choosing a bridge network for VMs and pods in this scenario?**

- a. Bridge networks assign unique IP addresses and share the MAC address of the host.
- b. Bridge networks enable VMs and pods to have exclusive access to a physical network device.
- c. Bridge networks enable VMs and pods to communicate with resources on external networks.
- d. Bridge networks provide unique MAC addresses for each VM and pod.

► **10. You are configuring a network for a high-throughput application in your OpenShift cluster that requires low latency. You must ensure that the VMs and pods have exclusive access to a physical Ethernet network device. Which CNI plug-in should you configure?**

- a. macvlan
- b. bridge
- c. ipvlan
- d. host-device

## ► Solution

# Advanced Networking with Multus CNI

Choose the correct answers to the following questions:

- ▶ **1. Which requirement does the Kubernetes network model impose?**
  - a. Communication between pods requires Network Address Translation (NAT).
  - b. Agents on a node can communicate with all pods on the same node.
  - c. Links are required to enable communication between pods in the cluster.
- ▶ **2. Which two options are the names of additional network interfaces that are attached to pods? (Choose two.)**
  - a. eth0
  - b. net1
  - c. eth1
  - d. net2
- ▶ **3. True or False: You can connect pods and virtual machines to networks that are external to the cluster.**
  - a. True
  - b. False
- ▶ **4. Which statement about additional network interfaces and the Multus Container Network Interface (CNI) plug-in is true?**
  - a. The Cluster Network Addons Operator manages additional network configurations for VMs that are based on the Multus CNI plug-in.
  - b. You can create network attachment definitions for VMs through the command line with a YAML manifest or through the OpenShift web console.
  - c. The Multus CNI plug-in is implemented through the NetworkAttachmentDefinition custom resource.
  - d. All of the above.
- ▶ **5. Which plug-in does the Multus CNI use to connect a VM to multiple networks?**
  - a. ip-vlan
  - b. host-device
  - c. bridge
  - d. mac-vlan

► **6. What is the role of the Cluster Network Addons Operator (CNAO) in Red Hat OpenShift?**

- a. Deploys and configures the OpenShift Virtualization operator.
- b. Directly manages the physical network interfaces on all nodes.
- c. Manages additional network configurations for VMs based on the Multus CNI plug-in.
- d. Creates and manages all primary network configurations for pods.

► **7. What is the purpose of a network attachment definition in OpenShift Virtualization?**

- a. Defines the CNI plug-in configuration to use for additional network interfaces on a pod or VM.
- b. Manages the storage resources for VMs and pods.
- c. Configures the primary network interface for all cluster nodes.
- d. Automatically deploys new VMs across the cluster.

► **8. Which network type enables VMs and pods to have unique IP addresses but to use the same MAC address as the host's physical network?**

- a. SR-IOV
- b. ipvlan
- c. macvlan
- d. host-device

► **9. You are a network administrator who is tasked with configuring a new VM in an OpenShift cluster to ensure that it can communicate with external networks. You decide to use a bridge network for this purpose. What is the primary reason for choosing a bridge network for VMs and pods in this scenario?**

- a. Bridge networks assign unique IP addresses and share the MAC address of the host.
- b. Bridge networks enable VMs and pods to have exclusive access to a physical network device.
- c. Bridge networks enable VMs and pods to communicate with resources on external networks.
- d. Bridge networks provide unique MAC addresses for each VM and pod.

► **10. You are configuring a network for a high-throughput application in your OpenShift cluster that requires low latency. You must ensure that the VMs and pods have exclusive access to a physical Ethernet network device. Which CNI plug-in should you configure?**

- a. macvlan
- b. bridge
- c. ipvlan
- d. host-device

# Configuring Multihomed Nodes and Virtual Machines

---

## Objectives

- Configure multihomed nodes and virtual machines by using the NMState operator and Multus CNI.

## The Kubernetes NMState Operator

The Kubernetes NMState operator provides a centralized and declarative host network configuration tool in a Red Hat OpenShift cluster. The operator reports the node network configuration, validates configuration syntax, and applies network configuration changes without a node reboot. With the NMState operator, you can configure additional node network devices, such as Linux bridges, and use them in network attachment definitions.

As with other operators, you can install the Kubernetes NMState operator by using either the OpenShift web console or the command line.

Consult the references section for installing an operator by using the OperatorHub or the `oc` command.

After the installation is complete, you must create an NMState instance with a name of `nmstate` to deploy the NMState controller on all OpenShift nodes.

From the OpenShift web console, go to **Operators > Installed Operators** and open the **Kubernetes NMState Operator** page. In the NMState card, click **Create instance**.

The screenshot shows the 'Operator details' page for the 'Kubernetes NMState Operator'. The 'Details' tab is selected. The page includes a logo for 'nmstate', the operator's name, version, and provider information. Below the header, there's a section for 'Provided APIs' with a single entry for 'NMState'.

**Figure 4.2: Creating the NMState instance**

The NMState operator provides three API resources for managing the network configuration:

- `NodeNetworkState`
- `NodeNetworkConfigurationPolicy`
- `NodeNetworkConfigurationEnactment`

## NMState Operator Versus Multus-CNI

The NMState operator and Multus-CNI are both used to manage network configurations in a Red Hat OpenShift cluster. Multus-CNI is a Container Network Interface (CNI) plug-in that enables multiple network interfaces to be attached to a single pod in a Kubernetes cluster. Multus-CNI enables you to connect a pod to multiple networks, each with its own network interface.

The NMState operator manages the network configuration of the host nodes in an OpenShift cluster. Multus-CNI manages the network configuration of pods in a Kubernetes cluster. Multus works better together with the NMState operator, which offers a declarative way of configuring those network devices on cluster nodes.

## Node Network State

A node network state (NNS) resource exists on each cluster node and is periodically updated to include the state of the network for that node. To review the current network state for a particular node, use the `oc get nns/node-name -o yaml` command.

The following example shows the output of the `oc get nns/node-name -o yaml` command:

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkState
metadata:
  name: worker01
  ...output omitted...
status:
  currentState:
    interfaces:
      - accept-all-mac-addresses: false
        controller: br-ex
        ...output omitted...
      ipv4:
        address:
          - ip: 192.168.50.13
            preferred-life-time: 437316606sec
            prefix-length: 24
            valid-life-time: 437316606sec
            ...output omitted...
        dhcp: true
        dhcp-send-hostname: true
        enabled: true
        ...output omitted...
      name: br-ex
      profile-name: ovs-if-br-ex
      state: up
      type: ovs-interface
      ...output omitted...
```

You can also review the node network configuration from the OpenShift web console on the [Networking > NodeNetworkState](#) page.

Name	IP address	Ports	MAC address	LLDP	MTU
ens3 ↑	-	-	52:54:00:00:32:0A	<input type="checkbox"/>	1500
genev_sys_6081 ↓	fe80::304a:37ff:fe63:582e/64	-	32:4A:37:63:58:2E	<input type="checkbox"/>	65000

**Figure 4.3: Node network configuration with NodeNetworkState**

For more information about interface types and configuration options, refer to the *NMState Operator* chapter in the Red Hat OpenShift Container Platform 4.16 *Networking* documentation.

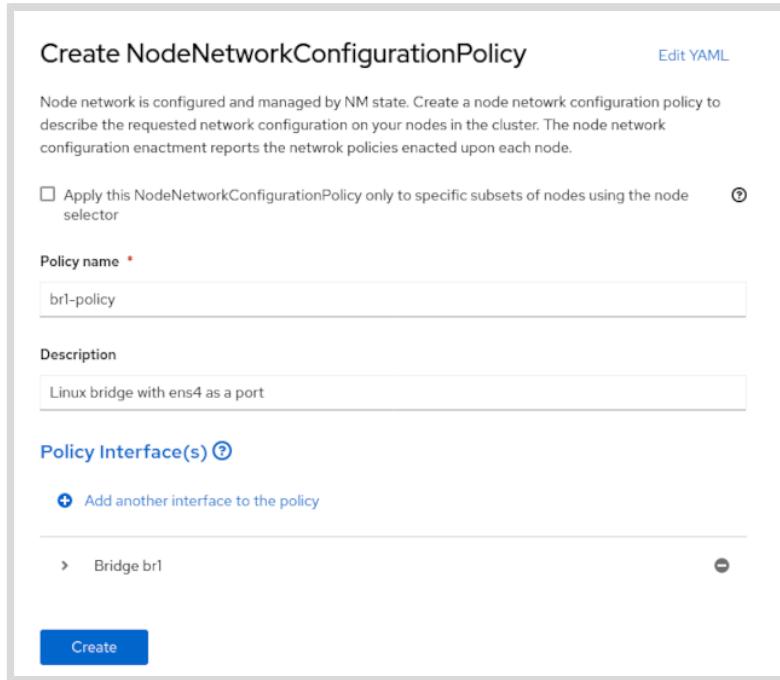
## Node Network Configuration Policy

A node network configuration policy (NNCP) describes the intended network configuration for OpenShift nodes. You can create and manage node network interfaces, such as declaring a Linux bridge, with a node network configuration policy. NMState can manage several interface types,

## Chapter 4 | Connecting Virtual Machines to External Networks

such as Linux bridge, bonding, and Ethernet. You can also configure additional options, such as Spanning Tree Protocol (STP) and IPv4 connectivity with an NNCP.

You can manage an NNCP from the OpenShift web console on the **Networking > NodeNetworkConfigurationPolicy** page.



**Figure 4.4: Creating a node network configuration policy**

By default, an NNCP resource is applied to all nodes in the cluster. However, you can specify which nodes to apply the policy to, such as only compute nodes, by including a node selector with the appropriate label in the NNCP.

The following example shows an NNCP that defines a Linux bridge on a node's ens4 network interface.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-policy 1
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: "" 2
  desiredState:
    interfaces:
      - name: br1 3
        description: Linux bridge with ens4 as a port 4
        type: linux-bridge 5
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
```

```
    enabled: false
  port:
    - name: ens4 ⑥
```

- ① The name for the node network configuration policy.
- ② The `nodeSelector` specifies that this policy applies to all compute nodes in the cluster. If the `nodeSelector` parameter is not included, then the policy applies to all nodes in the cluster.
- ③ The `name` parameter specifies the chosen name for the configured interface.
- ④ An optional description for the policy.
- ⑤ The `type` parameter specifies the type of connection to create.
- ⑥ The physical interfaces on the node to use in the defined Linux bridge.

You can list an NNCP resource from the command line with the `oc get nncp` command. To view the details of an existing NNCP resource, use the `oc get nncp nncp-name -o yaml` command.

To remove an interface from a node, you must modify or create an NNCP that designates the interface as `absent` under the `state` parameter. When setting an interface's state to `absent`, the network interface that is configured with the bridge or bonding interface is placed in the `down` state and connectivity is lost.

If you remove a bridge or bonding interface, then any node NICs in the cluster that were previously attached or subordinate to that bridge or bonding interface are placed in a `down` state and become unreachable. To avoid losing connectivity to the node NIC, configure the node NIC in the same policy to have a status of `up`, and either DHCP or a static IP address. This configuration ensures that the node NIC remains reachable after the bridge or bonding interface is removed.



### Important

Setting an interface to `absent` or deleting an NNCP resource does not restore the previous configuration. A cluster administrator must define a policy with the previous configuration to restore settings.

## Node Network Configuration Enactment

When you apply a node network configuration policy to a cluster, the NMState operator generates a `NodeNetworkConfigurationEnactment` (NNCE) object for each cluster node that the policy affects. The NNCE object reports the execution status of an applied policy and includes the defined settings that the policy enacts on each node. In the event of a failure, the node rolls back to its prior configuration, and the NNCE provides traceback data to troubleshoot the failure.

You can use the `oc get nnce` command to verify the status of a configuration policy and to confirm that it is successfully applied. To view more detailed information, including intended settings and traceback data, use the `oc get nnce nnce-name -o yaml` command.

On the OpenShift web console, the NNCE status is attached to the NNCP and is available on the `Networking > NodeNetworkConfigurationPolicy` page.

NodeNetworkConfigurationPolicy		
Name	Matched nodes	Node network state
NNCP br0-policy	5 nodes	1 Failing
		2 Aborted
		2 Available

Figure 4.5: Node network configuration enactments that are associated with a policy

You can click the node network state to get more information about the policy execution on each node, including error messages for troubleshooting.

Matched nodes summary					
<span style="color: red;">!</span> Failing	<span style="color: red;">✗</span> Aborted	<span style="color: green;">✓</span> Available	<span style="color: orange;">⌚</span> Progressing	<span style="color: grey;">⏳</span> Pending	x
<span style="color: red;">!</span> master01.br0-policy					
error reconciling NodeNetworkConfigurationPolicy on node master01 at desired state apply: "", failed to execute nmstatectl set --no-commit --timeout 480: 'exit status 1' '' 'Using 'set' is deprecated, use 'apply' instead.					

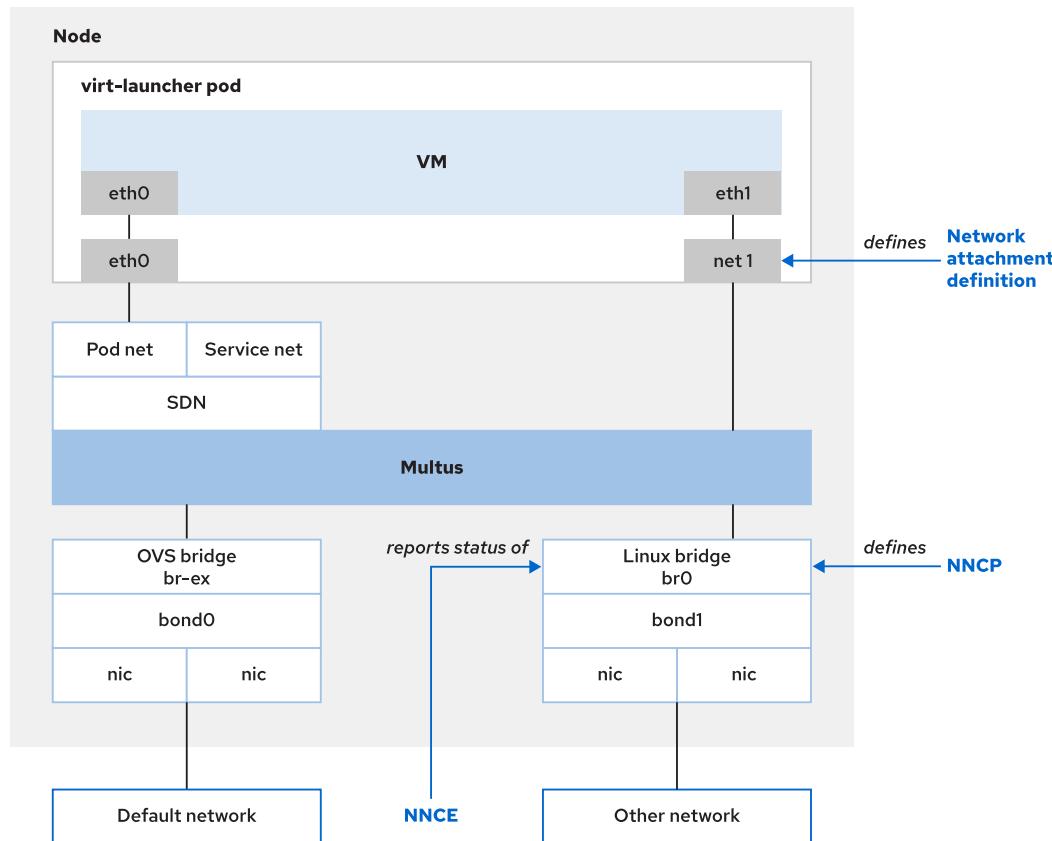
Figure 4.6: Failing node network configuration enactment

## Kubernetes NMState with Multus in OpenShift Virtualization

By default, VMs are connected to the default pod network. The VMs communicate with resources within the cluster, or with any resources that are accessible through the OpenShift node network. If a VM requires access to resources on a different network, then you must connect the VM to that additional network. VMs that are connected to more than one network are considered *multihomed* VMs.

OpenShift Virtualization uses Multus Container Network Interface (CNI) plug-ins and the Kubernetes NMState Operator resources to create a Linux bridge that connects VMs to additional networks. A Linux bridge forwards packets between connected interfaces, similar to the function of a network switch.

Optionally, you can use an Open vSwitch (OVS) bridge instead of a Linux bridge to connect VMs to additional networks. The OVS bridge provides additional features, such as VLAN tagging and network isolation.



**Figure 4.7: Linux bridge node networking with Multus and Kubernetes NMState**

A description follows of the components that connect VMs to additional networks:

#### bridge

Configure an additional bridge-based network to enable pods on the same host to communicate with each other and with the host.

A bridge is required to attach VMs to multiple networks. You use a bridge to connect the network interfaces of VMs and pods to a physical network interface on a host or a node. Pods and VMs on a bridge network use the same broadcast domain and subnet as the host's physical network interface.

If the host has access to networks that are external to the OpenShift cluster, then bridge networks enable VMs and pods in the OpenShift cluster to communicate with resources on that external network, such as file shares or databases.

#### ovs-bridge

Configure an OVS bridge to connect VMs to an external network.

An OVS bridge is a software-based switch that connects VMs to an external network. OVS bridges provide additional features, such as VLAN tagging and network isolation, that are not available with Linux bridges. By using an OVS bridge, VMs can communicate with other devices on the external network.

#### SDN

Configure a software-defined network (SDN) to connect VMs to a network.

## Chapter 4 | Connecting Virtual Machines to External Networks

An SDN is a network to connect VMs to a network, including to a network that is external to the OpenShift cluster.

### Pod network

Configure a pod network to connect pods to a network, including to a network that is external to the OpenShift cluster.

### Service network

Configure a service network to connect services to a network, including to a network that is external to the OpenShift cluster.

### Default network

Configure a default network to connect a VM to the default pod network.

### Linux bridge

Configure a Linux bridge to connect VMs to an external network.

A Linux bridge is a software-based switch that connects VMs to an external network. Linux bridges forward packets between the VM and the physical network interface on the host system. By using a Linux bridge, the VM can communicate with other devices on the external network.

### Bonding

Configure a bonding interface to aggregate multiple network interfaces into a single logical interface.

Bonding interfaces provide fault tolerance and load balancing for network traffic. You can use bonding interfaces to connect VMs to a network by using multiple physical NICs to provide redundancy in the event of network failures.

### NIC

Configure a network interface card (NIC) to connect a VM to a network, including to a network that is external to the OpenShift cluster.

## Connecting Virtual Machines to a Linux Bridge Network

When a Linux bridge is configured on nodes, such as through a node network configuration policy, you must create a network attachment definition (NAD) before you can connect your VMs to the bridge.

You can attach a VM to any NADs in the same namespace. Any NADs that you create in the default namespace are available to all VMs in the cluster.

Use the CNV Linux Bridge CNI plug-in in the network attachment definition to connect a VM to an additional network on the Linux bridge.

Creating a network attachment definition	
Name	br1-bridge-network
Description	Creating a network attachment definition
Network type	Linux bridge
Bridge name	br1
VLAN tag number	0

Creating a network attachment definition	
MAC spoof check	Checked

## Hot Plugging Secondary Network Interfaces

By hot plugging secondary network interfaces, you can add or remove network interfaces in a running VM without stopping the VM. OpenShift Virtualization provides support for hot plugging secondary interfaces that use the VirtIO device driver.

To hot plug a secondary network interface to a running VM, you must attach the network attachment definition (NAD) to the VM. The NAD must be in the same namespace as the VM.

To attach a secondary network interface to a running VM, go to **Virtualization > Virtual Machines**. Click the link for the VM. Click the **Configuration** tab and then click **Network Interfaces** to display the network interfaces that are connected to the VM. Click **Add Network Interface**, select the NAD from the **Network** list, and complete the remainder of the form. Click **Add** to save and attach the interface to the VM.



### Note

Hot unplugging is not supported for Single Root I/O Virtualization (SR-IOV) interfaces.

## Hot Plugging a Secondary Network Interface by Using the CLI

Before hot plugging a secondary network interface to a VM while the VM is running, ensure that a network attachment definition is configured in the same namespace as your VM. Ensure that the `virtcli` tool and the OpenShift CLI (`oc`) are installed on your local machine. Verify that the VM is running by using the `oc get vm` command.

```
[student@workstation ~]$ oc get vm vm-fedora
NAME      AGE
vm-fedora  1h
```

If the VM is not running, then start it by using the `virtctl start` command.

```
[student@workstation ~]$ virtctl start vm-fedora
```

You can edit VM's specification to add a network to a running VM and to a virtual machine instance (VMI) by using the `oc edit` command.

```
[student@workstation ~]$ oc edit vm rhel8-vm

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-fedora
template:
spec:
  domain:
    devices:
```

```
interfaces:
  - name: defaultnetwork
    masquerade: {}
  # new interface
  - name: <secondary_nic> ①
    bridge: {}
networks:
  - name: defaultnetwork
    pod: {}
  # new network
  - name: <secondary_nic> ②
    multus:
      networkName: <nad_name> ③
```

- ① The name of the secondary interface that is connected to the bridge network.
- ② The network name. This value must match the value of the `spec.template.spec.domain.devices.interfaces` object.
- ③ The name of the network attachment definition.

To attach the network interface to the running VM, live migrate the VM by using the following command:

```
[student@workstation ~]$ virtctl migrate vm vm-fedora
```

After you add the NAD to the VM, the secondary network interface is hot plugged to the VM.

## Attaching a Secondary Interface on the Additional Network

To connect a VM to an additional network, you must attach the NAD to a new network interface on the VM.

To create a network interface on the VM, go to **Virtualization > Virtual Machines**. Click the **Configuration** tab and then click **Network Interfaces** to display the network interfaces that are connected to the VM. Click **Add Network Interface**, select the NAD from the **Network** list, and complete the remaining form. Click **Add** to save and attach the interface to the VM.

Attaching a secondary network interface	
Name	nic-0
Model	VirtIO
Network	default/br1-bridge-network
Type	Bridge

You can also edit a stopped VM's manifest by using the `oc edit` command. You must add the bridge interface to the `spec.template.spec.domain.device.interfaces` list. You must also specify the NAD name in the `spec.template.spec.networks` list.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
```

```

name: rhel8-vm
spec:
  template:
    spec:
      domain:
        devices:
          interfaces:
            - masquerade: {}
              name: default
            - bridge: {}
              name: nic-0 ①
              model: virtio
            ...output omitted...
      networks:
        - name: default
          pod: {}
        - name: nic-0 ②
          multus:
            networkName: br1-bridge-network ③
      ...output omitted...

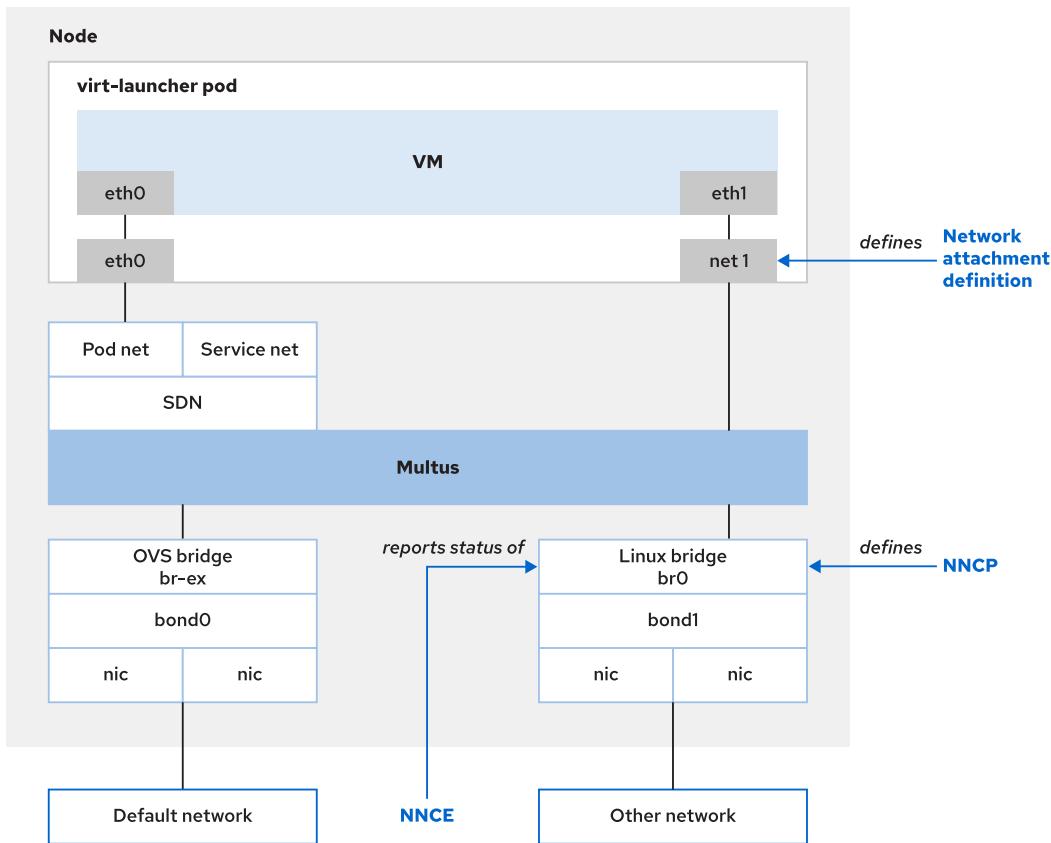
```

- ① The name of the secondary interface that is connected to the bridge network.
- ② The network name. This value must match the value of the `spec.template.spec.domain.devices.interfaces` object.
- ③ The name of the network attachment definition.

After you attach the network interface to the VM, the VM is connected to the Linux bridge network.

## OVS Bridge Overview

An OVS bridge connects the VM to the external network. The OVS bridge acts as a layer-2 switch that forwards packets between the VM and the physical network interface on the host system. By using the OVS bridge, the VM can communicate with other devices on the external network.



Traffic that is sent from the VM to the external network is forwarded to the OVS bridge, which then forwards the traffic to the physical network interface on the host system. The physical network interface sends the traffic to the external network.

The OVS bridge is created on the host system and is connected to the physical network interface. The VM is connected to the OVS bridge through a virtual network interface.

## Implementing Linux Bridges Versus OVS Bridges

Linux bridges and OVS bridges both connect VMs to additional networks. The environment requirements determine the choice between the two.

Linux bridges are more established and easier to configure than OVS bridges, although with fewer features. OVS bridges offer more tunnel types and contemporary features such as network automation, although they are more challenging to troubleshoot.

In OpenShift Virtualization, Red Hat recommends using OVS bridges instead of Linux bridges if the environment requires functions such as the Multi Network Policy, although either option leads to successful deployments.

The following table compares the features of Linux bridges and OVS bridges:

Feature	Linux bridge	OVS bridge
VLAN tagging	No	Yes

Feature	Linux bridge	OVS bridge
Network isolation	No	Yes
Availability	Default	Optional

OpenShift nodes can be connected to multiple networks that use different NICs. Although many configuration options exist, such as bonding and VLANs, the following example uses a dedicated NIC to configure an OVS bridge.

For more information about advanced configuration options, such as creating bonds or using VLANs, refer to the references section.

You can view the node interfaces by using the `oc get nns` command:

```
[student@workstation ~]$ oc get nns/worker0 -o
  jsonpath='{.status.currentState.interfaces[4]}' | jq
{
  ...
  ...
  "mac-address": "52:54:00:92:AA:00",
  "max-mtu": 1500,
  "min-mtu": 68,
  "mtu": 1500,
  "name": "ens3",
  "permanent-mac-address": "52:54:00:92:AA:00",
  "profile-name": "ovs-if-phys0",
  "state": "up",
  "type": "ethernet"
}
```

## OVN-Kubernetes L2 Overlay Network

OVN-Kubernetes enables access to powerful capabilities of Open Virtual Networks (OVN) and Open vSwitch software-defined networks, which Multus now supports directly. OVN supports the concept of multiple overlay networks, which are virtual layer 2 networks. The pod and service network is an OVN overlay network. You can create overlay networks to connect pods to additional networks.

Starting from Red Hat OpenShift 4.14, you can create Multus Network Attachment Definition resources that create additional OVN overlay networks, and connect your pods to those overlay networks as secondary networks.

OVN-Kubernetes layer 2 overlay networks support only east-west traffic, and NICs on these networks cannot communicate outside the cluster.

## Connecting VMs to an OVN-Kubernetes Layer 2 Overlay Network

To connect a VM to an OVN-Kubernetes L2 overlay network, you must attach the NAD to the VM.

To attach a VM to an OVN-Kubernetes L2 overlay network, go to **Virtualization > Virtual Machines**. Click the **Configuration** tab and then click **Network Interfaces** to display the network interfaces that are connected to the VM. Click **Add Network Interface**, select the NAD from the **Network** list, and complete the remainder of the form.

The following example shows a network attachment definition (NAD) that uses the OVN CNI plug-in to connect a VM to an OVN-Kubernetes L2 overlay network.

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: ovn-bridge-network
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "ovn-bridge-network", ①
    "type": "ovn-kubernetes", ②
    "bridge": "ovs-br0", ③
    "isGateway": true, ④
    "ipam": { ⑤
      "type": "host-local", ⑥
      "subnet": "10.0.0.0/24", ⑦
      {
        "ip": "10.131.0.26" ⑧
      }
    }
}'
```

- ①** The name of the network attachment definition.
- ②** The type of connection to create.
- ③** The OVS bridge that the network attachment definition connects to.
- ④** The gateway for the network attachment definition.
- ⑤** The IP address management (IPAM) configuration for the network attachment definition.
- ⑥** The type of IPAM to use.
- ⑦** The subnet mask for the network attachment definition.
- ⑧** The IP address for the network attachment definition.

The `ovn-bridge-network` NAD connects the VM to the `ovs-br0` OVS bridge network.

After you create the NAD, you can attach the network interface to the VM by using the OpenShift web console or the command line.

**Note**

The OVN CNI plug-in connects a VM to an OVN-Kubernetes L2 overlay network.

## OVS Bridge Network Node Configuration Policy with OVN

To configure an OVS bridge on a node, you must create a node network configuration policy that defines the bridge and the physical interface that connects to the bridge.

The following example shows an NNCP that defines an OVS bridge on a node's `ens4` network interface:

```

apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ovs-bridge-policy ①
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: "" ②
  desiredState:
    interfaces:
      - name: ovs-br0 ③
        description: OVS bridge with ens4 as a port ④
        type: ovs-bridge ⑤
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
              enabled: false
        port:
          - name: ens4 ⑥
  ovn:
    bridge-mappings:
      - localnet: br0-bridge-network
        bridge: ovs-br0
        state: present

```

- ① The name for the node network configuration policy.
- ② The `nodeSelector` parameter specifies that this policy applies to all compute nodes in the cluster. If the `nodeSelector` parameter is not included, then the policy applies to all nodes in the cluster.
- ③ The `name` parameter specifies the chosen name for the configured interface.
- ④ An optional `description` for the policy.
- ⑤ The `type` parameter specifies the type of connection to create.
- ⑥ The physical interface on the node to use in the defined OVS bridge.

It is possible to have heterogeneous nodes with different NIC names for the same networks. In that case, Red Hat recommends that you add a label to each node type to identify the applicable configuration. Then, by using the `.spec.nodeSelector` parameter, you can apply the configuration only to nodes that will work with this configuration. For the other node types, you can modify the NNCP and the `nodeSelector` parameter to create the same bridge on those nodes, even if the underlying NIC name is different.

After you create the NNCP, the NMState operator applies the configuration to the node. You can review the node network configuration from the OpenShift web console on the **Networking > NodeNetworkState** page.

## Configuring IP Addresses on a VM

VMs on the pod network have an ephemeral IP address that cannot be statically assigned. If your VM requires a static or dynamic IP address, then you must attach a secondary network interface to your VM that is connected to a bridge network.



### Note

A DHCP server must be available on the bridge network to provide a dynamic IP address to the VM.

To configure an IP address, you can use the `cloud-init` service to specify a static or dynamic IP address for the secondary interface. The network device and address are defined in the `spec.volumes.cloudInitNoCloud.networkData` field of the VM manifest.

The following example specifies a static IP address on an `eth1` network interface:

```
kind: VirtualMachine
spec:
...output omitted...
volumes:
- cloudInitNoCloud:
  networkData: |
    version: 2
    eternets:
      eth1: ①
        addresses:
        - 192.168.51.150/24 ②
```

- ①** The name of the network interface.
- ②** The static IP address.

See the references section for more information about configuring IP addresses with the `cloud-init` service.

After the configuration is complete, you can review the IP addresses of a running VM on the Overview tab.



### Note

The QEMU guest agent is responsible for collecting the IP addresses of the VM. The guest agent must be installed and running on the VM operating system for OpenShift Virtualization to display the network information in the web console.

The screenshot shows the Red Hat Virtual Machines interface. At the top, it displays "VirtualMachines > VirtualMachine details" for a VM named "dev-dbaccess" which is "Running". Below this, there are tabs for Overview, Details, Metrics, YAML, Configuration, Events, Console, Snapshots, and Diagnostics. The Overview tab is selected. On the left, under "Details", there are several configuration items:

Name	dev-dbaccess	VNC console
Status	Running	
Created	10 minutes ago	
Operating system	Red Hat Enterprise Linux 8.5 (Ootpa)	
CPU   Memory	1CPU   2 GiB Memory	
Hostname	dev-dbaccess	
Template	rhel8-server-small	<a href="#">Open web console</a>

On the right side, there are sections for "Alerts (0)", "Schemas (0)", and "Network interfaces (2)". The "Network interfaces (2)" section is highlighted with a red box and contains the following table:

Name	IP address
default	10.8.2.84
database	192.168.51.10

[+1 more](#)

**Figure 4.9: IP addresses that are assigned to a running VM**



## References

For more information about the NMState operator, refer to the *Kubernetes NMState* chapter in the Red Hat OpenShift Container Platform 4.16 *Networking* documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/networking/index#kubernetes-nmstate](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/networking/index#kubernetes-nmstate)

For more information about installing operators by using the OperatorHub, refer to the *Installing from OperatorHub Using the Web Console* section in the *Administrator Tasks* chapter in the Red Hat OpenShift Container Platform 4.16 *Operators* documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/operators/index#olm-installing-from-operatorhub-using-web-console\\_olm-adding-operators-to-a-cluster](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/operators/index#olm-installing-from-operatorhub-using-web-console_olm-adding-operators-to-a-cluster)

For more information about installing operators by using the command line, refer to the *Installing from OperatorHub Using the CLI* section in the *Administrator Tasks* chapter in the Red Hat OpenShift Container Platform 4.16 *Operators* documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/operators/index#olm-installing-operator-from-operatorhub-using-cli\\_olm-adding-operators-to-a-cluster](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/operators/index#olm-installing-operator-from-operatorhub-using-cli_olm-adding-operators-to-a-cluster)

For more information about attaching a VM on an additional network, refer to the *Connecting a Virtual Machine to a Linux Bridge Network* section in the *Networking* chapter in the Red Hat OpenShift Container Platform 4.16 *Virtualization* documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-connecting-vm-to-linux-bridge](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/virtualization/index#virt-connecting-vm-to-linux-bridge)

For more information about configuring IP addresses with the `cloud-init` service, refer to the *Configuring IP Addresses for Virtual Machines* section in the *Networking* chapter in the Red Hat OpenShift Container Platform 4.16 *Virtualization* documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-configuring-viewing-ips-for-vms](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.16/html-single/virtualization/index#virt-configuring-viewing-ips-for-vms)

For more information about OpenShift Virtualization implementation guidelines, refer to the *Networking Architecture* section in the *OpenShift Virtualization Reference Implementation Guide* documentation at

[https://access.redhat.com/sites/default/files/attachments/openshift\\_virtualization\\_reference\\_implementation\\_guide\\_0.pdf](https://access.redhat.com/sites/default/files/attachments/openshift_virtualization_reference_implementation_guide_0.pdf)

## ► Guided Exercise

# Configure Multihomed Nodes and Virtual Machines

Configure a multihomed virtual machine with a static IP address to access an external database on an additional network.

### Outcomes

- Deploy the Kubernetes NMState operator.
- Export the current network configuration for the cluster nodes.
- Create a node network configuration policy to configure a Linux bridge on the compute nodes.
- Observe the policy enactment on the compute nodes.
- Configure a secondary interface on a VM to use the Linux bridge and a custom MAC address.
- Confirm that the VM is connected on an additional network and is configured with a 192.168.51.10/24 static IP address.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start multihomed-nmstate
```

### Instructions

A developer's VM needs access to a database outside your Red Hat OpenShift Container Platform cluster. The database, on `server.srv.example.com`, is configured to accept incoming connections from the `devuser` user, from a specific IP address on the 192.168.51.0/24 network only.

The cluster nodes, `worker01` and `worker02`, each have an additional `ens4` interface, on the 192.168.51.0/24 network. Use the Kubernetes NMState operator to configure the `worker01` and `worker02` nodes as multihomed nodes with an L2 Linux bridge on the `ens4` interface.

After creating the Linux bridge on the compute nodes, create a network attachment definition to enable VMs within the `developer-vms` project to use the database network that the Linux bridge provides. Next, configure the developer's VM to use this additional network as a secondary interface with a static IP address. Confirm that the VM can communicate with the database on `server.srv.example.com` on the database network.

- 1. As the `admin` user, install the Kubernetes NMState operator.

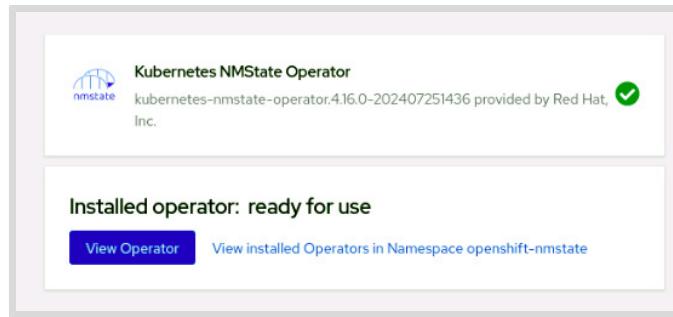
- 1.1. Open a web browser and go to <https://console-openshift-console.apps.ocp4.example.com>
- 1.2. Click **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.
- 1.3. Click **Operators > OperatorHub**. In the **Filter** by keyword field, type **nmstate** to locate the NMState operator, and then click **Kubernetes NMState Operator**.

The screenshot shows the OperatorHub interface. On the left, there's a sidebar with categories like All Items, Cloud Provider, Developer Tools, etc. A search bar at the top right contains the query 'nmstate'. Below the search bar, a card for the 'Kubernetes NMState Operator' is displayed. The card includes the operator's name, source (do316 Operator Catalog), and a brief description: 'Kubernetes NMState is a declarative means of configuring NetworkManager.'

- 1.4. The web console displays information about the Kubernetes NMState operator. Click **Install** to proceed to the **Install Operator** page.

The screenshot shows the 'Install Operator' page for the 'Kubernetes NMState Operator'. It features fields for 'Channel' (set to 'stable') and 'Version' (set to '4.16.0-202407251436'). Under 'Capability level', the 'Basic Install' option is selected. A prominent blue 'Install' button is located at the top left of the form.

- 1.5. Click **Install** to install the operator with the default options in the **openshift-nmstate** namespace.
- 1.6. Wait until the installation is complete and the web console displays the **ready for use** message.



► 2. Create an NMState instance.

- 2.1. After the operator installation completes, click **View Operator**.



**Note**

Alternatively, you can go to Operators > Installed Operators and open the **Kubernetes NMState Operator** page.

- 2.2. In the NMState card, click **Create instance** to view the creation parameters for the NMState instance.

- 2.3. Ensure that the instance name is **nmstate**, and click **Create** at the lower left of the window.

- 2.4. Wait a few minutes until the web console shows the **Web console update is available** message, and then reload the web console.

- 2.5. Verify that both **NodeNetworkConfigurationPolicy** and **NodeNetworkState** items are available in the **Networking** menu.

The screenshot shows the OpenShift web console's left sidebar under the 'Networking' category. The 'NodeNetworkConfigurationPolicy' and 'NodeNetworkState' items are highlighted with a red box at the bottom of the list.

Name	Kind	Status
NMS nmstate	NMState	-

- 3. Observe and then back up the current node network configuration of the **worker01** and **worker02** nodes to YAML files. The backup YAML files are for reference only, and cannot be applied to the cluster to modify or restore node networking to a previous state.
- 3.1. From the left panel of the OpenShift web console, go to **Networking > NodeNetworkState**.
  - 3.2. Observe the network configuration of the **worker01** and **worker02** nodes. Scroll down and expand the worker node details to view the network interfaces. The **ens4** network interface is connected to the same network as the database server.

The screenshot shows the network details for the **worker01** node. The **ens4** interface is highlighted with a red box. The IP address listed is **192.168.51.13/24**.

Name	IP address	Ports	MAC address	LLDP	MTU
ens3	-	-	52:54:00:00:32:0D	<input type="checkbox"/>	1500
ens4	192.168.51.13/24	-	52:54:00:01:33:0D	<input type="checkbox"/>	1500

- 3.3. Click **worker01**, go to the **YAML** tab, and click **Download** to export the node network configuration as a YAML file.

The screenshot shows the 'NodeNetworkState details' page for the node 'worker01'. The page title is 'NNS worker01'. At the top, there are tabs for 'Details' and 'YAML', with 'YAML' being the active tab. Below the tabs is a toolbar with various keyboard shortcuts and a 'View sidebar' button. The main content area displays the YAML configuration of the node, which includes fields like apiVersion, kind, metadata, managedFields, name, ownerReferences, resourceVersion, status, currentState, and dns-resolver. At the bottom of the page are three buttons: 'Save', 'Reload', and 'Cancel', followed by a 'Download' button which is highlighted with a red box.

```

1 apiVersion: nmstate.io/v1beta1
2 kind: NodeNetworkState
3 metadata:
4 creationTimestamp: '2024-03-12T11:00:01Z'
5 generation: 1
6 > managedFields: ...
7 name: worker01
8 ownerReferences:
9 - apiVersion: v1
10 kind: Node
11 name: worker01
12 uid: 4bd42c37-4f83-439c-9908-8770512998f8
13 resourceVersion: '27509'
14 uid: 33de339a-7b73-49bb-b06b-4a1e2aef28c2
15 status:
16 currentState:
17 dns-resolver:

```

3.4. Repeat the operation for the `worker02` node.

- ▶ 4. Add the `external-network=true` label to the `worker01` and `worker02` nodes. You use this label as a node selector in later steps.

4.1. Go to **Compute > Nodes**, click the vertical ellipsis icon `:` next to the `worker01` line, and then click **Edit labels**.

The screenshot shows the 'Nodes' list page. The table header includes columns for Name, Status, Roles, Pods, Memory, CPU, Filesystem, and Created. The data rows show five nodes: master01, master02, master03, worker01, and worker02. The 'worker01' row has a context menu open, with the 'Edit labels' option highlighted by a red box. Other options in the menu include 'Mark as unschedulable', 'Edit annotations', 'Edit Node', and 'Delete Node'.

Name	Status	Roles	Pods	Memory	CPU	Filesystem	Created	
master01	Ready	control-plane, master, worker	39	8.8 GiB / 23.47 GiB	1.491 cores / 4 cores	18.25 GiB / 69.72 GiB	Sep 11, 2024, 7:11 PM	⋮
master02	Ready	control-plane, master, worker	83	10.92 GiB / 23.47 GiB	1.313 cores / 4 cores	21.39 GiB / 69.72 GiB	Sep 11, 2024, 7:11 PM	⋮
master03	Ready	control-plane, master, worker	49	8.25 GiB / 23.47 GiB	0.965 cores / 4 cores	18.79 GiB / 69.72 GiB	Sep 11, 2024, 7:12 PM	⋮
worker01	Ready	worker	75	11.23 GiB / 15.62 GiB	1.082 cores / 6 cores	30.66 GiB / 39.78 GiB	Sep 24, 2024, 11:59 PM	⋮
worker02	Ready	worker	42	9.51 GiB / 15.62 GiB	0.682 cores / 6 cores	25.19 GiB / 39.78 GiB		<span style="border: 1px solid #ccc; padding: 2px;">Mark as unschedulable</span> <span style="border: 1px solid #ccc; padding: 2px;">Edit labels</span> <span style="border: 1px solid #ccc; padding: 2px;">Edit annotations</span> <span style="border: 1px solid #ccc; padding: 2px;">Edit Node</span> <span style="border: 1px solid #ccc; padding: 2px;">Delete Node</span>

4.2. Type `external-network=true` in the form and click **Save**.

4.3. Repeat this step for the `worker02` node.

- 4.4. On the **Compute > Nodes** page, select **Label** in the drop-down menu, type `external` in the **Search by label** field, and select the `external-network=true` label.

**Nodes**

Name	Status	Roles	Memory
master01	Ready	control-plane, master	7.04 GiB / 15.62 GiB

- 4.5. Verify that the node list contains only the **worker01** and **worker02** nodes.

**Nodes**

Name	Status	Roles	Pods	Memory
worker01	Ready	worker	54	8.11 GiB / 15.62 GiB
worker02	Ready	worker	36	6.55 GiB / 15.62 GiB

- ▶ 5. Create a node network configuration policy to configure a new Linux bridge with the ens4 Ethernet device as the port.

- 5.1. Go to Networking > **NodeNetworkConfigurationPolicy** and click **Create NodeNetworkConfigurationPolicy**.



#### Note

Alternatively, you can click **Create** at the upper right and select **From Form**.

- 5.2. Select the **Apply this NodeNetworkConfigurationPolicy** checkbox to open the node selector.

Create NodeNetworkConfigurationPolicy [Edit YAML](#)

Node network is configured and managed by NM state. Create a node netowrk configuration policy to describe the requested network configuration on your nodes in the cluster. The node network configuration enactment reports the netwrok policies enacted upon each node.

Apply this NodeNetworkConfigurationPolicy only to specific subsets of nodes using the node selector [?](#)

**Policy name \***

policy-name

**Description**

- 5.3. Click **Add Label**, and type `external-network` as the key, and `true` as the value. Verify that both `worker01` and `worker02` nodes match that label configuration by clicking **View matching Nodes**.

Key	Value
external-network	true

+ Add Label      Explore Node list [?](#)

2 Nodes matching [View matching Nodes](#)

Save      Cancel

- 5.4. Click **Save** after you verify that these nodes appear in the list.
- 5.5. Complete the **Create NodeNetworkConfigurationPolicy** form by using the following information to create the Linux bridge:

Field	Value
Name	br0-ens4-policy
Description	Linux bridge with ens4 as a port
Interface name	br0
Network state	Up
Type	Bridge
IPv4	Checked
DHCP	Checked
Port	ens4

- 5.6. Click **Create** to create the node network configuration policy.
- 6. Monitor the progress of the configuration policy.
- 6.1. Go to **Networking > NodeNetworkConfigurationPolicy** to review the status of the node network configuration policy.
  - 6.2. Observe the **br0-ens4-policy** status and wait until the **Enactment states** value is **2 Available**.

NodeNetworkConfigurationPolicy		
Filter	Name	Search by name...
	br0-ens4-policy	2 nodes
		✓ 2 Available

- 6.3. Click **2 Available** to review the status summary and confirm that the policy applies successfully on each node.
- Matched nodes summary**

<span style="color: red;">!</span> Failing	<span style="color: red;">×</span> Aborted	<span style="color: green;">✓</span> Available	<span style="color: blue;">⌚</span> Progressing	<span style="color: grey;">⌚</span> Pending
--	--	--	---	---

  - ▼ worker01.br0-ens4-policy
    - successfully reconciled
  - ▼ worker02.br0-ens4-policy
    - successfully reconciled
- 7. Observe the network configuration of the **worker01** and **worker02** nodes, and confirm the configuration of the new Linux bridge.

- 7.1. Go to **Networking > NodeNetworkState**, and notice the **linux-bridge** item on both **worker01** and **worker02** nodes.
- 7.2. Click **linux-bridge** and review the configuration summary of the Linux bridge. The **br0** bridge is connected to the same network as the database server.

The screenshot shows the OpenShift NodeNetworkState interface. It lists four nodes: master02, master03, worker01, and worker02. For each node, it shows network interfaces: ethernet (3), ovs-bridge (1), and ovs-interface (5). A modal window is open over master02 showing details for the linux-bridge (1) on br0. The modal displays the following information:

Name	br0 ↑
IP address	192.168.51.100/24
Ports	1
LLDP	<input type="checkbox"/>
MTU	1500

- 7.3. Click **br0** to display more details about the bridge configuration, and verify that the bridge port is **ens4**.

The screenshot shows the details for the br0 bridge. The "Details" tab is selected, displaying the following configuration:

- LLDP**: Disabled
- Ports**: ens4
- MAC Address**: 52:54:00:01:33:0D
- Features**:
  - highdma
  - rx-gro

The **worker01** and **worker02** nodes are now configured as multihomed nodes with an L2 Linux bridge that provides access to the database network.

- 8. Create a network attachment definition in the **developer -vms** project that uses the Linux bridge to attach VMs to the database network.
  - 8.1. Go to **Networking > NetworkAttachmentDefinitions**.
  - 8.2. Select the **developer -vms** project from the **Project** list, and then click **Create Network Attachment Definition**.

- 8.3. Complete the **Create Network Attachment Definition** form by using the following information to create a network attachment definition for the database network:

Field	Value
Name	database-network
Description	Linux bridge on the worker01 and worker02 nodes with ens4 as a port
Network Type	Linux bridge
Bridge Name	br0

- 8.4. Click **Create** to create the network attachment definition.
- ▶ 9. Configure the developer's VM to use the database network on a secondary interface. Configure the secondary interface to use **00:00:00:00:00:01** as a custom MAC address. The custom MAC address is required for the VM to receive the **192.168.51.10/24** static IP address from the DHCP server in the database network.
- 9.1. Go to **Virtualization > VirtualMachines**.
  - 9.2. Select the **developer-vms** project from the **Projects** list. Click the **dev-dbaccess** VM to open the VM's **Overview** tab.
  - 9.3. Click the **Configuration** tab and select **Network**.
  - 9.4. Click **Add network interface**.
  - 9.5. On the **Add network interface** page, set the name to **database** and select **developer-vms/database-network** from the **Network** field.
  - 9.6. Click **Advanced** to expand the configuration options and enter **00:00:00:00:00:01** in the **MAC address** field. Click **Save** to save and attach the interface to the VM.

Field	Value
Name	database
Model	VirtIO
Network	developer-vms/database-network
Type	Bridge
MAC address	00:00:00:00:00:01

- 9.7. Restart the VM to apply the pending changes. Click the **Restart** icon, and then click **Restart** to confirm the action.

The screenshot shows the Red Hat OpenShift interface for a VM named 'dev-dbaccess'. The 'Configuration' tab is selected. In the 'Network' section, there is a table of network interfaces. The first row, 'database', has a 'Pending' status highlighted with a yellow circle. The table columns are Name, Model, Network, Type, and MAC address.

Name	Model	Network	Type	MAC address
database	virtio	database-network	Bridge	02:57:8e:00:00:01
default	-	Pod networking	Masquerade	02:57:8e:00:00:00

- ▶ 10. Confirm that the VM's secondary network interface, `eth1`, is configured with a `192.168.51.10/24` static IP address.
- 10.1. Click the **Console** tab to open the VM's console.
  - 10.2. Click the console window and log in as the `developer` user with `developer` as the password.
  - 10.3. Use the `ip addr show eth1` command to display the IP address configuration of the `eth1` network interface.

```
[developer@dev-dbaccess ~]$ ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:00:00:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.51.10/24 brd 192.168.51.255 scope global dynamic noprefixroute
        eth1
            valid_lft 437230198sec preferred_lft 437230198sec
            inet6 fe80::b50f:3586:6165:c5a3/64 scope link noprefixroute
                valid_lft forever preferred_lft forever
```

- 10.4. Alternatively, you can observe the VM IPs on the **Overview** tab, in the **Network Interfaces** card.

**VM dev-dbaccess** Running

**Overview** Details Metrics YAML Configuration Events Console Snapshots Diagnostics

Details		...	Alerts (0)
Name	dev-dbaccess	VNC console	Snapshots (0) <a href="#">Take snapshot</a>
Status	<span>Running</span>	No snapshots found	
Created	10 minutes ago	Network interfaces (2)	
Operating system	Red Hat Enterprise Linux 8.5 (Ootpa)	Name	IP address
CPU   Memory	1CPU   2 GiB Memory	default	10.8.2.84
Hostname	dev-dbaccess	database	192.168.51.10 <a href="#">+1 more</a>
Template	<a href="#">rhel8-server-small</a>	Disks (2)	
	<a href="#">Open web console</a>		

- ▶ 11. Connect to the `sakila` database on `server.srv.example.com` with the `mysql` client. The host, `server.srv.example.com`, resolves with the `server` hostname to resources in the `192.168.51.0/24` network.
- 11.1. On the VM's console, use the `mysql` command to connect to the MariaDB instance on the `server.srv.example.com` host. Log in as the `devuser` user with `developer` as the password.

```
[developer@dev-dbaccess ~]$ mysql -u devuser -h server -p
Enter password: developer
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.5.16-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

- 11.2. Connect to the `sakila` database with the `use` command.

```
MariaDB [(none)]> use sakila;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [sakila]>
```

- 11.3. Confirm that the devuser can view the sakila tables with the show tables; command.

```
MariaDB [sakila]> show tables;
+-----+
| Tables_in_sakila      |
+-----+
...output omitted...
+-----+
23 rows in set (0.002 sec)
```

- 11.4. Exit the database with the exit; command.

```
MariaDB [sakila]> exit;
Bye
[developer@dev-dbaccess ~]$
```

## Finish

As the student user on the workstation machine, confirm that you are in the /home/student directory and then use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish multihomed-nmstate
```

## ▶ Lab

# Connect Virtual Machines to External Networks

Configure node networking to connect a virtual machine to an additional network, and then connect the virtual machine to an external web server residing on the additional network.

## Outcomes

- Create a node network configuration policy that configures a Linux bridge where DHCP is enabled on an identified node interface.
- Create a network attachment definition for the Linux bridge.
- Attach a secondary interface with a custom MAC address on a virtual machine.
- Verify communication between the VM and resources on the additional network.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start multihomed-review
```

## Instructions

The `dev-external` VM in the `multihomed-review` project requires access to an Apache HTTP instance on `server.srv.example.com`. That Apache server accepts requests only from a specific IP on the external network within the `192.168.51.0/24` IP range. A DHCP server on the external network provides the required static IP to the `00:00:00:00:00:02` MAC address.

The VM needs an additional network interface that is connected to the external network with the right IP to communicate with the Apache HTTP server.

As the `admin` user, label the nodes `worker01` and `worker02` with the `external-network=true` label. Configure a Linux bridge named `br0` on the `worker01` and `worker02` nodes, and then create a network attachment definition named `br0-network` in the `multihomed-review` project.

For creating those resources from the command line, you can find partial resource YAML files in the `~/DO316/labs/multihomed-review/` path.

Before attaching the second network interface, connect to `server.srv.example.com` on the `dev-external` VM and confirm that a warning is displayed about permissions to the Apache instance. Attach an additional network interface to the VM that uses the `br0-network` network attachment definition, and specify `00:00:00:00:00:02` as the MAC address for the interface.

Finally, test connectivity to `server.srv.example.com` from the VM. Confirm that the response is a `Hello, World!` message.

1. As the OpenShift admin user, label the nodes `worker01` and `worker02` with the `external-network=true` label.
2. Create a Linux bridge network on the `worker01` and `worker02` nodes with the following requirements:

Field	Value
Name	<code>br0-ens4-policy</code>
Interface name	<code>br0</code>
Interface type	Bridge
IPv4	Enabled
DHCP	Enabled
Bridge port	<code>ens4</code>
Node Selector	<code>external-network=true</code>

3. Create a network attachment definition in the `multihomed-review` project with the following requirements:

Field	Value
Name	<code>br0-network</code>
Namespace	<code>multihomed-review</code>
Network Type	Linux bridge
Bridge Name	<code>br0</code>

4. From the console of the `dev-external` VM, use the `curl` command to connect to `server.srv.example.com`, and then confirm that you do not have permission to access the site.

**Note**

The `server.srv.example.com` fully qualified domain name also resolves with the `server` hostname for resources in the `192.168.51.0/24` network.

5. Attach a secondary network interface to the `dev-external` VM with the following information:

Field	Value
Name	<i>Leave the default name</i>
Model	<code>virtio</code>
Network	<code>multihomed-review/br0-network</code>
Type	Bridge
MAC Address	<code>00:00:00:00:00:02</code>

The custom MAC address, `00:00:00:00:00:02`, is required for the VM to receive the `192.168.51.20/24` static IP address from the DHCP server in the Linux bridge network.

6. From the console of the `dev-external` VM, use the `curl` command to connect to `server.srv.example.com`, and confirm that you receive a `Hello, World!` response.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade multihomed-review
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish multihomed-review
```

## ► Solution

# Connect Virtual Machines to External Networks

Configure node networking to connect a virtual machine to an additional network, and then connect the virtual machine to an external web server residing on the additional network.

### Outcomes

- Create a node network configuration policy that configures a Linux bridge where DHCP is enabled on an identified node interface.
- Create a network attachment definition for the Linux bridge.
- Attach a secondary interface with a custom MAC address on a virtual machine.
- Verify communication between the VM and resources on the additional network.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start multihomed-review
```

### Instructions

The `dev-external` VM in the `multihomed-review` project requires access to an Apache HTTP instance on `server.srv.example.com`. That Apache server accepts requests only from a specific IP on the external network within the `192.168.51.0/24` IP range. A DHCP server on the external network provides the required static IP to the `00:00:00:00:00:02` MAC address.

The VM needs an additional network interface that is connected to the external network with the right IP to communicate with the Apache HTTP server.

As the `admin` user, label the nodes `worker01` and `worker02` with the `external-network=true` label. Configure a Linux bridge named `br0` on the `worker01` and `worker02` nodes, and then create a network attachment definition named `br0-network` in the `multihomed-review` project.

For creating those resources from the command line, you can find partial resource YAML files in the `~/D0316/labs/multihomed-review/` path.

Before attaching the second network interface, connect to `server.srv.example.com` on the `dev-external` VM and confirm that a warning is displayed about permissions to the Apache instance. Attach an additional network interface to the VM that uses the `br0-network` network attachment definition, and specify `00:00:00:00:00:02` as the MAC address for the interface.

Finally, test connectivity to `server.srv.example.com` from the VM. Confirm that the response is a `Hello, World!` message.

1. As the OpenShift admin user, label the nodes `worker01` and `worker02` with the `external-network=true` label.
  - 1.1. Open a web browser and go to `https://console.openshift-console.apps.ocp4.example.com`
  - 1.2. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
  - 1.3. Go to `Compute > Nodes`, click the vertical ellipsis icon : next to `worker01`, and then click `Edit labels`.
  - 1.4. Type `external-network=true` in the field and click `Save`.
  - 1.5. Repeat this step with the `worker02` node.
  - 1.6. On the `Compute > Nodes` page, select `Label` from the drop-down menu, type `external` in the `Search by label` field, and select the `external-network=true` label.

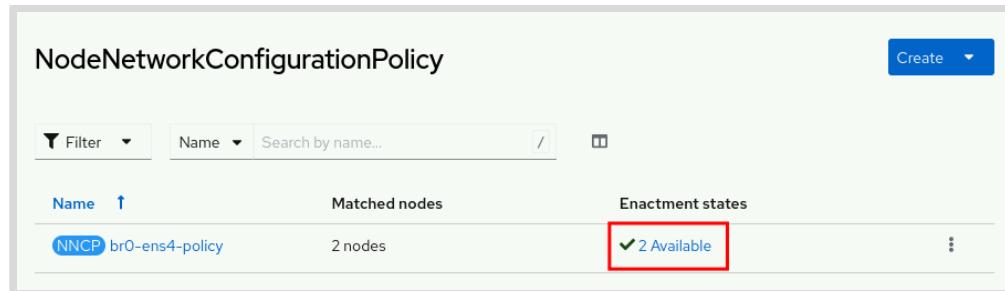
The screenshot shows the 'Nodes' list page in the OpenShift web console. At the top, there are filters for 'Filter' (dropdown), 'Label' (dropdown set to 'external'), and a search bar containing 'external'. Below the filters, there are columns for 'Name', 'Status', and 'Memory'. A red box highlights the 'Label' dropdown and the search bar. Another red box highlights the 'external-network=true' entry in the search results table. The table lists one node: 'master01' which is 'Ready', has 'control-plane, master' roles, and 39GiB of memory. The total memory shown is 7.04 GiB / 15.62 GiB.

- 1.7. Verify that the node list contains only the `worker01` and `worker02` nodes.
2. Create a Linux bridge network on the `worker01` and `worker02` nodes with the following requirements:

Field	Value
Name	<code>br0-ens4-policy</code>
Interface name	<code>br0</code>
Interface type	Bridge
IPv4	Enabled
DHCP	Enabled
Bridge port	<code>ens4</code>
Node Selector	<code>external-network=true</code>

- 2.1. Go to `Networking > NodeNetworkConfigurationPolicy` and click `Create NodeNetworkConfigurationPolicy`.
- 2.2. Select the `Apply this NodeNetworkConfigurationPolicy` checkbox to open the node selector.

- 2.3. Click **Add Label**, and type `external-network` as the key and `true` as the value.
- 2.4. Click **Save** and then complete the **Create NodeNetworkConfigurationPolicy** form with the provided specifications.
- 2.5. Click **Create** to create the node network configuration policy.
- 2.6. Confirm that the policy is successfully configured on the nodes by verifying its status on the **Networking > NodeNetworkConfigurationPolicy** page. Wait until the **Enactment states** value is `2 Available`.



The screenshot shows a table with three columns: Name, Matched nodes, and Enactment states. The 'Name' column contains 'NNCP br0-ens4-policy'. The 'Matched nodes' column contains '2 nodes'. The 'Enactment states' column contains '✓ 2 Available'. A red box highlights the '2 Available' text in the 'Enactment states' column.

3. Create a network attachment definition in the `multihomed-review` project with the following requirements:

Field	Value
Name	<code>br0-network</code>
Namespace	<code>multihomed-review</code>
Network Type	Linux bridge
Bridge Name	<code>br0</code>

- 3.1. Go to **Networking > NetworkAttachmentDefinitions**.
- 3.2. Select the `multihomed-review` project from the **Project** list and then click **Create Network Attachment Definition**.
- 3.3. Complete the **Create network attachment definition** form by using the provided specifications.
- 3.4. Click **Create** to create the network attachment definition.
4. From the console of the `dev-external` VM, use the `curl` command to connect to `server.srv.example.com`, and then confirm that you do not have permission to access the site.



#### Note

The `server.srv.example.com` fully qualified domain name also resolves with the `server` hostname for resources in the `192.168.51.0/24` network.

- 4.1. Go to **Virtualization > VirtualMachines**. Select the `multihomed-review` project from the **Projects** list.

- 4.2. Click the **dev-external** VM and select the **Console** tab to open the VM's console.
- 4.3. Click **Guest login credentials** to display the credentials of the VM, and then use those credentials to log in to the VM.

The screenshot shows the OpenShift web interface for a project named "multihomed-review". In the navigation bar, "VirtualMachines" is selected, and the "VirtualMachine details" page for "dev-external" is shown. The VM status is "Running". The "Console" tab is active. Under the "Console" section, there is a "Guest login credentials" dropdown. Below it, a message states: "The following credentials for this operating system were created via cloud-init. If unsuccessful, cloud-init continues to attempt to log in using these credentials. Contact the image provider for more information." Two fields are displayed: "User name" with value "developer" and "Password" with value "developer". The "User name" and "Password" fields are highlighted with a red rectangular box.

- 4.4. Test the connectivity to the Apache HTTP instance on `server.srv.example.com` with the `curl` command. Confirm that you receive a warning that you do not have permission to access the resource.

```
[developer@dev-external ~]$ curl server.srv.example.com
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

5. Attach a secondary network interface to the **dev-external** VM with the following information:

Field	Value
Name	<i>Leave the default name</i>
Model	<code>virtio</code>
Network	<code>multihomed-review/br0-network</code>
Type	Bridge
MAC Address	<code>00:00:00:00:00:02</code>

The custom MAC address, `00:00:00:00:00:02`, is required for the VM to receive the `192.168.51.20/24` static IP address from the DHCP server in the Linux bridge network.

- 5.1. On the **dev-external** VM page, click the **Configuration** tab and then click **Network** to display the network interfaces that are connected to the VM.
  - 5.2. Click **Add network interface** to create an interface on the VM. Modify the parameters to match the specifications from the table. Click **Save** to attach the interface.
  - 5.3. Restart the VM to apply the pending changes. Click **Actions > Restart**.
  - 5.4. Wait until the VM is in the **running** state.
6. From the console of the **dev-external** VM, use the **curl** command to connect to **server.srv.example.com**, and confirm that you receive a **Hello, World!** response.
    - 6.1. Click the **Console** tab to access the VM's console. Click **Guest login credentials** to display the credentials of the VM and then use those credentials to log in to the VM.
    - 6.2. Connect to the Apache HTTP instance on **server.srv.example.com** with the **curl** command. Confirm that you receive a **Hello, World!** response.

```
[developer@dev-external ~]$ curl server.srv.example.com  
Hello, World!
```

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade multihomed-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish multihomed-review
```

# Summary

---

- To review the current network state for a particular node, use the `oc get nns/node-name -o yaml` command or the **Networking > NodeNetworkState** page.
- A *Node Network Configuration Policy* resource is a YAML manifest that describes the intended network configuration for Red Hat OpenShift nodes, and is applied to all nodes in the cluster by default.
- OpenShift Virtualization supports managing the bridge, VLAN, Ethernet, and bond interface types.
- To remove an interface from a node, you must modify or create an NNCP that designates the interface as **absent** under the `state` parameter.
- OpenShift Virtualization uses the Multus CNI plug-ins and the Kubernetes NMState Operator resources to create a bridge that connects VMs to additional networks.
- Additional network interfaces that use the Multus CNI plug-in follow the `netN` naming convention.
- If the VM requires a static or a dynamic IP address, then you must attach a secondary network interface to your VM that is connected to a bridge network.
- You can verify the network attachment definition in a namespace with the `oc get net-attach-def -n namespace` command or the **Networking > Network Attachment Definitions** page.
- A DHCP server must be available on the bridge network to provide a dynamic IP address to VMs.
- Use the `cloud-init` service to specify a static IP address for a secondary interface.
- OpenShift Virtualization provides support for hot plugging secondary interfaces that use the VirtIO device driver.
- You can create Multus Network Attachment Definition resources that create additional OVN overlay networks.

## Chapter 5

# Configuring Storage for Virtual Machines

### Goal

Manage storage and disks for virtual machines in Red Hat OpenShift by using Kubernetes.

### Sections

- Attaching Persistent Storage to Virtual Machines (and Guided Exercise)
- Managing Storage Devices on Virtual Machines (and Guided Exercise)
- Connecting a Virtual Machine to External Storage (and Guided Exercise)

### Lab

- Configure Storage for Virtual Machines

# Attaching Persistent Storage to Virtual Machines

## Objectives

- Describe Kubernetes storage resources and attach persistent storage to virtual machines.

## Container Storage Concepts

Kubernetes implements various resources to abstract the details of the underlying storage architecture. Developers can request persistent storage for applications without applying specific settings for LUNs, NFS, or other storage back ends.

Kubernetes pods are composed of one or more containers. Each container is stateless by default because the provided storage is ephemeral. You can configure the containers in a pod to request persistent storage to preserve the state when the application is restarted.

## Container Storage

Containers have ephemeral storage by default. When Kubernetes destroys a pod, it deletes all the files and data that are associated with that pod. You need a mechanism to preserve the application data, because Kubernetes destroys and re-creates pods when you scale down your application, deploy a new version, or when a node is drained for maintenance.

### Ephemeral storage

Pods and containers can require temporary or transient local storage for their operation. This ephemeral storage does not extend beyond the lifetime of the individual pod, and this ephemeral storage cannot be shared across pods.

### Persistent storage

Pods and containers can require permanent storage for their operation. OpenShift uses the Kubernetes persistent volume (PV) framework so cluster administrators can provision persistent storage for a cluster. Developers can use a Persistent Volume Claim (PVC) to request PV resources without knowing the underlying storage infrastructure.

## Connection to the Back-end Storage

The administrator configures a connection between the Kubernetes cluster and the back-end storage by using Container Storage Interface (CSI) plug-ins, and then creates storage classes to represent the available storage types in the back-end storage controller.

### CSI plug-ins

The CSI standard enables the use of backing storage to containers. The Kubernetes cluster plug-ins enable the cluster to interface with the specific storage back end and to provide persistent storage to containers.

### Storage Classes

The storage class resources in Kubernetes describe the storage characteristics, such as quality of service, throughput, or technology that the backing data services provide. Kubernetes defines storage classes to provide various cluster storage types to address different application requirements. Applications consume these storage classes as PVs that PVCs define. Storage classes are cluster-wide resources.

Each storage class uses a storage provisioner that determines the volume plug-in for provisioning the PVs. Each storage class also contains the fields for such needs as to configure the reclaim policy (to retain, recycle, or delete the released volumes), or to enable volume expansion. Other storage technologies, such as Ceph File System or Amazon Elastic Block Storage, use different volume provisioners.

## Persistent Volumes and Persistent Volume Claims

Kubernetes uses persistent volumes to preserve data across container executions. Inside the container, you mount a persistent volume in a directory where the application stores its data. For example, MariaDB Server stores its database data in the `/var/lib/mysql/data` directory. By mounting a volume in that directory, you preserve the database data across container executions.

### Persistent Volumes

PVs are Kubernetes resources that abstract from the users how the storage is provisioned. Depending on the underlying backing data service, the administrator can create the persistent volumes in advance, or the storage provisioner can create them dynamically.

The persistent volume specifications contain the access mode, the storage capacity, and the storage class name that provides the backing data service. Persistent volumes are cluster-wide resources.

### Persistent Volume Claims

PVCs are Kubernetes objects to request storage PVs. A PVC can include specific details such as the size and access mode for the storage volume. PVCs are namespaced resources.

## Volume Provisioning

Kubernetes supports two ways of provisioning storage: static and dynamic.

### Static provisioning

Requires the cluster administrators to prepare volumes manually. For example, administrators can create PVs that are mapped to NFS shares or to logical units from a Storage Area Network (SAN) system. Developers can use PVC resources to access these volumes.

### Dynamic provisioning

When a developer creates a PVC to get a volume of a specific storage class, Kubernetes uses the *provisioner* that is associated with the class to allocate the volume dynamically on demand. A provisioner is a plug-in that communicates with the back-end storage to manage volumes.

## Volume Modes

Applications can request a persistent storage file system to mount a directory, or a raw block device for the fastest possible low-level operations. The application developer specifies the intended volume mode in the PVC.

### Filesystem

When set to the **Filesystem** default, the returned volume has a file system that you can directly mount inside your container.

This volume mode is the default if the parameter is not specified.

### Block

When set to **Block**, you access the volume in the container as a block device, such as `/dev/xvda`. You use this mode with applications that require a raw device.

**Note**

Not all back-end storage supports both modes.

## Volume Access Modes

Each storage back end has different characteristics, and the provided volumes can be mounted by one or more containers in read-only or read/write mode. The application developer specifies the intended access mode in the PVC.

### ReadWriteOnce

You can mount the volume read/write on a single node at a time. This mode is often used for block devices and is widely supported across many storage providers. The OpenShift web console and the `oc get pvc` command report this mode as RWO.

### ReadWriteOncePod

You can mount the volume read/write by a single pod on a single node at a time. This mode relies on the SELinux mount feature, is often used for block devices, and supports many storage providers. The OpenShift web console and the `oc get pvc` command report this mode as RWOP.

### ReadOnlyMany

You can mount the volume read-only on multiple containers simultaneously. The OpenShift web console and the `oc get pvc` command report this mode as ROX. Not all back-end storage supports this mode.

### ReadWriteMany

You can simultaneously mount the volume read/write on multiple containers. The OpenShift web console and the `oc get pvc` command report this mode as RWX. Not all back-end storage supports this mode.

**Warning**

VM live migration requires disks of the `ReadWriteMany` (RWX) type.

## Storage Classes

As a developer, you request a volume for your application by creating a Kubernetes PVC resource. The main parameters of a PVC are the size and access mode for the volume. Kubernetes processes your PVC request, prepares a PV, and then binds it to the PVC.

```
[user@host ~]$ oc get pvc
NAME    STATUS    VOLUME   CAPACITY   MODE      STORAGECLASS
vm1     Bound     pvc-...   10Gi      RWX       ocs-external-storagecluster-ceph-rbd ①
vm2     Pending    ...       ...        ...       ocs-external-storagecluster-ceph-rbd-vi... ②
```

- ① The `vm1` PVC is bound and its associated persistent volume is being used in a virtual machine.
- ② The `vm2` PVC has the `Pending` status because it is waiting for Kubernetes to prepare and assign a PV.

As a developer, you do not know the exact details of the storage infrastructure. The cluster administrators manage that infrastructure, which might be composed of several technologies,

## Chapter 5 | Configuring Storage for Virtual Machines

such as iSCSI, Fibre Channel (FC), Network File System (NFS), Amazon Elastic Block Store (EBS), or Red Hat Ceph Storage services.

The cluster administrators expose these storage solutions through *storage classes*. Developers select the storage class in their PVCs. For example, the cluster administrators might create a storage class named `gold` that is using your company's fast Red Hat Ceph Storage infrastructure. The administrators can also create a `silver` storage class based on slower NFS shares.

When you deploy Red Hat OpenShift on a cloud provider infrastructure, the installation process automatically creates a default storage class. For example, the default storage class on Amazon Web Services (AWS) is named `gp2` and uses EBS volumes of the *General Purpose 2 (gp2)* type. On Microsoft Azure, `managed-premium` is the default storage class name.

You can list the available storage classes on your cluster by using the `oc get storageclasses` command:

```
[user@host ~]$ oc get storageclasses
NAME                                     PROVISIONER
nfs-storage (default)                   k8s-sigs.io/nfs-subdir-... ①
ocs-external-storagecluster-ceph-rbd    openshift-storage.rbd.csi...
ocs-external-storagecluster-ceph-rbd-virtualization  openshift-storage.rbd... ②
ocs-external-storagecluster-ceph-rgw     openshift-storage.ceph.rook...
ocs-external-storagecluster-cephfs      openshift-storage.cephfs.csi...
openshift-storage.noobaa.io             openshift-storage.noobaa.io...
```

- ① The default storage class has an indicator to help identify it.
- ② The ODF storage class for virtualization.

You can also use the OpenShift web console. Go to **Storage > StorageClasses** to list the storage classes.

Name	Provisioner
SC nfs-storage - Default ①	k8s-sigs.io/nfs-subdir-external-provisioner
SC ocs-external-storagecluster-cephfs	openshift-storage.cephfs.csi.ceph.com
SC ocs-external-storagecluster-ceph-rbd	openshift-storage.rbd.csi.ceph.com
SC ocs-external-storagecluster-ceph-rbd-virtualization ②	openshift-storage.rbd.csi.ceph.com
SC ocs-external-storagecluster-ceph-rgw	openshift-storage.ceph.rook.io/bucket
SC openshift-storage.noobaa.io	openshift-storage.noobaa.io/obc

- ① The default storage class has an indicator to help identify it.
- ② The ODF storage class for virtualization.

The preceding screen capture shows the available storage classes. In this example, the cluster administrators created the `nfs-storage` class. The other classes are from a Red Hat OpenShift Data Foundation installation. OpenShift Data Foundation is an additional product that is based on Ceph technologies. It can aggregate the available disk space on your cluster nodes as a resilient storage service.

## Storage Class Annotations

The persistent volume claims might contain a field to specify the storage class that is requested from the cluster. The default storage class is used for the new PVCs that do not specify a storage class requirement.

The OpenShift storage classes might contain annotations to identify the default storage class in the cluster, or to indicate that the storage class is specialized for use with virtual machine disks.

`storageclass.kubernetes.io/is-default-class=true`

This annotation indicates that this storage class is used for all the new PVCs if no storage class is specified. This annotation is applied to the default storage class with a value of `true`. You can verify whether the storage class has this annotation by using the following command:

```
[user@host ~]$ oc describe storageclass nfs-storage | grep is-default-class  
Annotations:           storageclass.kubernetes.io/is-default-class=true
```

`storageclass.kubevirt.io/is-default-virt-class=true`

This annotation indicates that the storage class has customized settings for virtual machine disks. OpenShift Data Foundation creates this storage class when it detects that OpenShift Virtualization is installed on the cluster. You can verify whether the storage class has this annotation by using the following command:

```
[user@host ~]$ oc describe storageclass \  
  ocs-external-storagecluster-ceph-rbd-virtualization | grep is-default-virt-class  
Annotations:           description=Provides RWO and RWX Block volumes suitable for  
                       Virtual Machine disks,storageclass.kubevirt.io/is-default-virt-class=true
```



### Note

OpenShift Data Foundation creates the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class only if it is installed in *internal mode*.

You can create the storage class with the `mapOptions: "krbd:rxbounce"` parameter if OpenShift Data Foundation is installed on your cluster in *external mode*.

## Persistent Volume Requests

As a developer, you request a PV through a PVC. You can prepare a PVC by using a YAML resource manifest. PVCs are namespaced resources, whereas PVs are cluster-wide resources.

```
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: dbdata ①  
spec:  
  storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization ②
```

```
accessModes:  
  - ReadWriteOnce ❸  
resources:  
  requests:  
    storage: 10Gi ❹  
volumeMode: Filesystem ❺
```

- ❶ The name of the PVC resource.
- ❷ The `storageClassName` parameter specifies the storage class to use.
- ❸ The `accessModes` parameter indicates how containers can access the volume.
- ❹ The `storage` parameter specifies the requested volume size.
- ❺ The `volumeMode` parameter gives the volume type.

**Note**

Most applications consume volumes in the `Filesystem` mode, because containers store their data as files and directories.

However, for optimal performance, Red Hat recommends using volumes in `Block` mode for VM disks.

You can also use the OpenShift web console to create PVCs. Go to **Storage > PersistentVolumeClaims**, click **Create PersistentVolumeClaim > With Form**, and then complete the form.

The screenshot shows the 'Create PersistentVolumeClaim' page in the OpenShift web interface. The left sidebar has 'PersistentVolumeClaims' selected under 'Storage'. The main form includes fields for StorageClass (set to 'ocp-external-storagecluster-ceph-rbd-virtualization'), PersistentVolumeClaim name ('dbdata'), Access mode ('Single user (RWO)'), Size ('10 GiB'), and Volume mode ('Block'). Buttons for 'Create' and 'Cancel' are at the bottom.

Figure 5.1: Create a Persistent Volume Claim

## Persistent Storage Configuration for Virtual Machines

Red Hat OpenShift Virtualization provides several mechanisms to manage the VM disks. Most of these mechanisms rely on PVCs. A VM disk uses a PV that contains a disk image. For example, the root disk of a VM uses a PV that stores the operating system disk image.

- With PVCs in **Block** mode, OpenShift Virtualization transfers the disk image into the volume. The VM disk uses the volume as its back-end device.
- With PVCs in **Filesystem** mode, OpenShift Virtualization creates a `disk.img` file at the root of the PV file system and then copies the disk image into the file. The VM disk uses the `disk.img` file as its back-end device.

OpenShift Virtualization introduces new resource types to help creating the PVC with optimal parameters for VM disks and copying the disk image into the resulting PV:

### Storage profile

For each storage class, a storage profile resource gives default values that are optimized for VM disks. Storage profiles are cluster-wide resources.

```
[user@host ~]$ oc get storageprofiles
NAME                                AGE
nfs-storage                           1h
ocs-external-storagecluster-ceph-rbd   1h
ocs-external-storagecluster-ceph-rbd-virtualization 1h
ocs-external-storagecluster-ceph-rgw    1h
ocs-external-storagecluster-cephfs     1h
openshift-storage.noobaa.io           1h
```

## Data volume

A data volume resource describes a VM disk. It groups the PVC definition and the details of the disk image to inject into the PV. Data volumes are namespaced resources.

```
[user@host ~]$ oc get datavolumes -n storage-intro
NAME      PHASE      PROGRESS      RESTARTS      AGE
vm1      Succeeded  100.0%        1            5m
vm2      Succeeded  100.0%        1            1m
```

## Storage Profiles

For each storage class, OpenShift Virtualization creates a storage profile. Storage profiles have the same name as their associated storage classes. A storage profile defines the PVC default values that are best suited for VM disks.

The following example shows that the ocs-external-storagecluster-ceph-rbd-virtualization storage profile provides default values for the PVCs that are created through the storage class with the same name:

```
[user@host ~]$ oc describe storageprofile \
  ocs-external-storagecluster-ceph-rbd-virtualization
Name:          ocs-external-storagecluster-ceph-rbd-virtualization ①
Namespace:     ②
Labels:        ...output omitted...
Annotations:  <none>
API Version:  cdi.kubevirt.io/v1beta1
Kind:          StorageProfile
...output omitted...
Spec:
Status:
  Claim Property Sets:
    Access Modes:
      ReadWriteMany ③
    Volume Mode: Block ④
    Access Modes:
      ReadWriteOnce ⑤
    Volume Mode: Block
    Access Modes:
      ReadWriteOnce
    Volume Mode: Filesystem ⑥
    ...output omitted...
  Provisioner:      openshift-storage.rbd.csi.ceph.com
```

```
Storage Class: ocs-external-storagecluster-ceph-rbd-virtualization ⑦
Events: <none>
...output omitted...
```

- ① The name of the storage profile resource.
- ② No namespace is listed, because storage profiles are cluster-wide resources.
- ③ ⑤ This storage profile supports the `ReadWriteMany` and `ReadWriteOnce` access modes.
- ④ ⑥ This storage profile supports the `Block` and `Filesystem` volume modes.
- ⑦ The associated storage class of this storage volume is `ocs-external-storagecluster-ceph-rbd-virtualization`.

The default settings for a storage profile are determined by the order in which the access mode and volume modes are declared. The preceding storage profile declares the `ReadWriteMany` access mode as the default, which enables VM live migrations, and declares the `Block` volume mode as the default, which provides better performance for VM disks than the `Filesystem` mode.

As a developer, when you use a storage profile to prepare a VM disk, the only parameter that you must provide is the disk size. The storage profile ensures that all the other parameters have an optimal value for VM disk usage.

## Virtual Machine Disk Creation by Using a Data Volume

The data volume resources define properties for the associated PVC, and the location of the source image to initialize the volume.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: vm1 ①
  namespace: storage-intro ②
  ...output omitted...
spec:
  pvc:
    accessModes:
      - ReadWriteMany ③
    resources:
      requests:
        storage: 10Gi ④
    storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization ⑤
    volumeMode: Block ⑥
  source:
    http:
      url: http://images.example.com/openShift4/images/mariadb-server.qcow2 ⑦
```

- ① The name of the data volume resource.
- ② The namespace that contains this data volume resource.
- ③ The associated PVC requests a volume with the `ReadWriteMany` access mode.
- ④ The requested volume size for the persistent volume claim.

- ⑤ The requested storage class for the persistent volume claim.
- ⑥ The associated PVC requests a block volume.
- ⑦ Use the disk image from this URL to initialize the persistent volume. OpenShift Virtualization can import the image from this location or by cloning another PVC.



### Note

The following example of a source section instructs the data volume to create an empty disk:

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
...output omitted...
spec:
...output omitted...
source:
blank: {} ①
```

- ① The curly braces represent an *empty map* in YAML syntax.

When the PVC to create is in `Filesystem` mode, the data volume automatically instructs Kubernetes to create the PVC 5.5% larger than requested to account for file system overhead. By doing so, Kubernetes ensures that it can create the disk image file, `disk.img`, with the requested size.



### Warning

If your cluster administrator set storage resource quotas for your project, then these quotas apply to the PVCs that you create through data volumes.



### Note

Deleting a data volume resource triggers the deletion of the associated PVC and PV. However, if the volume is attached to a VM, then the deletion occurs only after you detach the volume.

## Attaching a Data Volume to a Virtual Machine

Use the OpenShift web console to create a data volume and attach it to a VM as an additional disk.

1. Go to **Virtualization > VirtualMachines**.
2. Click the VM name to open its details page.
3. Click **Configuration > Disks** to display the VM disks.
4. Click **Add disk** and complete the form.

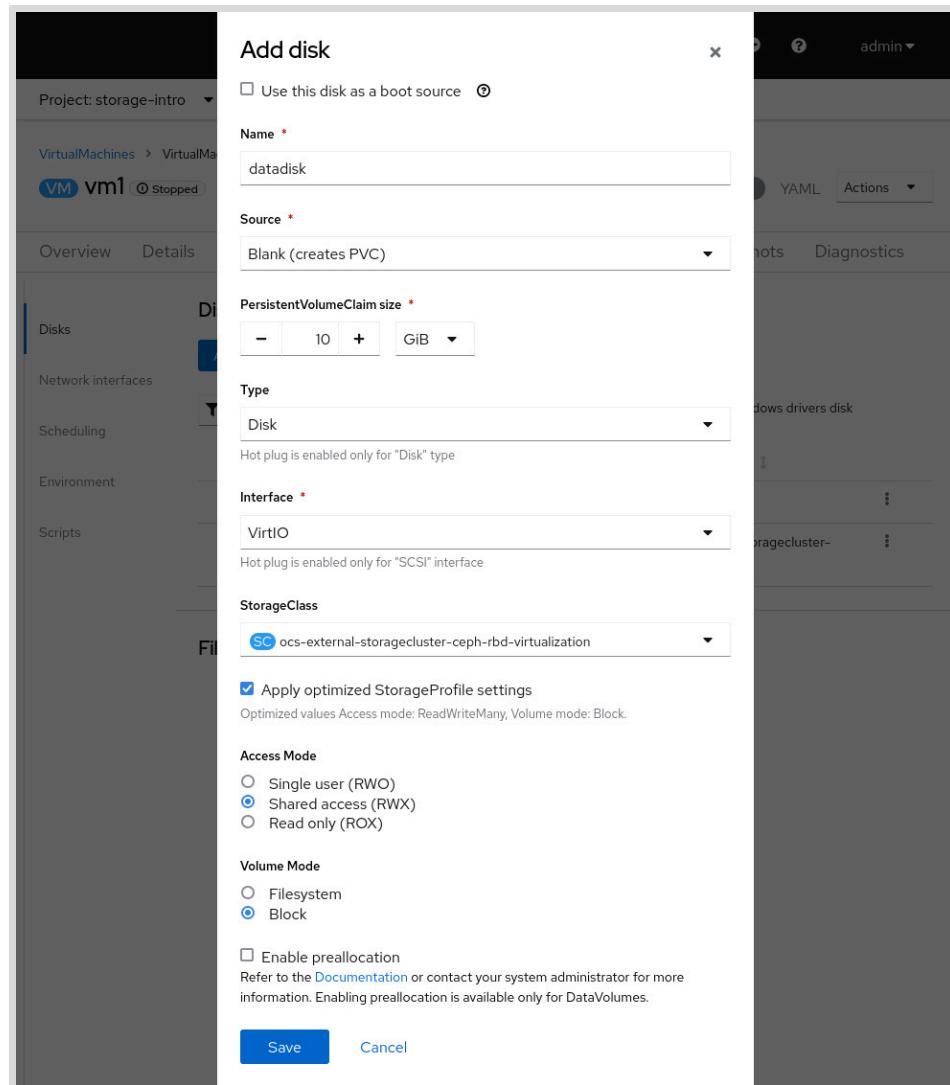


Figure 5.2: Add a disk to a virtual machine

**Note**

You can attach or remove disks in running VMs. The `virtio-scsi` controller on each VM eliminates the prior limitations of `virtio`. The new controller is highly scalable and supports the hot plugging of over four million disks.

## Using Local Node Disk Space for Virtual Machine Disks

Some virtualization features, such as live migration or VM snapshots, rely on your back-end storage. For example, using Red Hat Ceph Storage RBD volumes enables live migration, snapshots, and disk cloning.

For single-node clusters or clusters without a dedicated storage solution, the administrator can configure the cluster to use the local disk space of each node as persistent storage by using any of the following solutions:

- Kubernetes `hostPath` volume
- Local Storage Operator

- LVM Storage Operator

**Warning**

You cannot use advanced features such as live migration when allocating your VM disks on the node disk space.

## Cloning Virtual Machine Disks

Cloning a disk is a common administrative task for duplicating nodes, back up data volumes, and migrating data across nodes. Because PVCs are namespaced objects, only users with appropriate access to the `datavolumes` project can clone disks across multiple namespaces. To clone disks across namespaces, a cluster role and corresponding role binding are necessary to provide permissions for all actions for the `datavolumes` resource. To create these permissions, a user with the `cluster-admin` role is required.

Here is an example manifest for creating the `ClusterRole` that grants a user the necessary permissions:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <EXAMPLE-ROLE-NAME> ①
rules:
- apiGroups: ["cdi.kubevirt.io"]
  resources: ["datavolumes/source"]
  verbs: ["*"]
```

- ① The new cluster role name

## Comparison with VMware Storage Concepts

Adopting OpenShift and migrating VM workloads from a VMware solution is a prevalent path for many businesses. Understanding the terminology and storage concepts of both platforms ensures a smooth transition during migration.

OpenShift manages many storage back ends through storage pools that are available for deployments to create disk allocations. A PVC provides the volume details and requests provisioning the PV for consumption. The PV contains the single disk that the VM uses for storage. In this relationship, a storage administrator is responsible for capacity management in the storage back ends, whereas a cluster or virtualization administrator authors the PVCs to consume the available storage from the back ends.

VMware aggregates the various storage types that are available through the infrastructure as datastores, which contain many VM disks. Deployments in a VMware cluster consume the storage resources from a datastore through the vSAN layer within the hypervisor. In the VMware platform paradigm, a virtualization administrator is responsible for managing the storage capacity for the deployments.

When planning a migration from VMware to OpenShift clusters, you must map the existing VM disk types to the OpenShift storage classes. Understanding the range of options for disk performance and access types in OpenShift storage classes ensures strong VM performance and a smooth transition to the new environment.



## References

For more information about RHOCP storage, refer to the *Red Hat OpenShift Container Platform Storage* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/storage/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/storage/index)

For more information about VM disks, refer to the *Virtual Machine Disks* section in the *Red Hat OpenShift Container Platform Virtualization* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#vm-disks](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#vm-disks)

For more information about Red Hat OpenShift Data Foundation, refer to the *Architecture* guide at  
[https://docs.redhat.com/en/documentation/red\\_hat\\_openshift\\_data\\_foundation/4.16/html-single/red\\_hat\\_openshift\\_data\\_foundation\\_architecture/index](https://docs.redhat.com/en/documentation/red_hat_openshift_data_foundation/4.16/html-single/red_hat_openshift_data_foundation_architecture/index)

For more information about the OpenShift Data Foundation, refer to the *Red Hat OpenShift Data Foundation Data Sheet* article at  
<https://www.redhat.com/en/resources/openshift-data-foundation-datasheet>

For more information about persistent data, refer to the *Persistent Data on Red Hat OpenShift at the Edge* article at  
<https://www.redhat.com/en/blog/persistent-data-red-hat-openshift-edge>

For more information about optimizing ODF persistent volumes, refer to the *Optimizing ODF PersistentVolumes for VMs* article at  
<https://access.redhat.com/articles/6978371>

For more information about Kubernetes persistent volumes, refer to the *Kubernetes Persistent Volumes* documentation at  
<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>

For more information about disks and volumes, refer to the *Kubernetes Disks and Volumes* documentation at  
[https://kubevirt.io/user-guide/virtual\\_machines/disks\\_and\\_volumes/](https://kubevirt.io/user-guide/virtual_machines/disks_and_volumes/)

## ► Guided Exercise

# Attach Persistent Storage to Virtual Machines

Connect virtual machines to different persistent storage services based on performance and cost needs for the running applications.

### Outcomes

- Prepare new VM disks.
- Attach disks to VMs.
- Detach and delete disks from VMs.

### Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the **storage-intro** namespace and starts two virtual machines in that namespace: the **vm1** machine, which hosts a MariaDB database, and the **vm2** machine, which hosts a web server.

```
[student@workstation ~]$ lab start storage-intro
```

### Instructions

- ▶ 1. Use the OpenShift web console to confirm that the two VMs are running.
  - 1.1. Open a web browser and go to the web console URL. Select **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.  
<https://console-openshift-console.apps.ocp4.example.com>
  - 1.2. Go to **Virtualization > VirtualMachines** and select the **storage-intro** project. Confirm that the **vm1** and **vm2** machines are running.
- ▶ 2. Prepare a 5 GiB blank disk for the **vm1** database server. Attach the disk to the VM by using a **virtio** interface. Use the **ocs-external-storagecluster-ceph-rbd-virtualization** storage class, which provides fast block storage.
  - 2.1. Stop the **vm1** machine so that you can use the **virtio** interface to attach the new disk. Remember that the only available interface is **scsi** when the VM is running. Select the **vm1** machine, and then either click **Actions > Stop** or select the square stop icon next to the drop-down menu. Click **Stop** to confirm the operation, if needed. Wait for the machine to stop.
  - 2.2. Either go to the **Configuration > Storage** tab, or use the search field to quickly locate storage settings, and then click **Add disk**. Complete the form by using the following

information. Leave any unmentioned fields with their default settings, and click **Save** to create and attach the disk to the VM.

Field	Value
Name	dbdata
Source	Empty disk (blank)
Size	5 GiB
Type	Disk
Interface	VirtIO
StorageClass	ocs-external-storagecluster-ceph-rbd-virtualization
Apply optimized StorageProfile settings	<i>Checked [implies ReadWriteMany access and Block volume modes]</i>



#### Note

If the **Interface** field is set to `scsi` and `virtio` is not available, then click **Cancel** and start over. The web interface takes a few seconds to detect that the VM stopped.

- 2.3. Either click **Actions > Start** to start the `vm1` machine, or select the triangular start icon next to the drop-down menu.



#### Note

If a yellow admonition appears, then wait a few moments and try again to start the VM.

- 2.4. Log in to the VM console and confirm that a new 5 GiB device is available. Go to the **Console** tab and then log in as the `root` user with `redhat` as the password. Run the `lsblk` command to list the block devices. Notice the 5 GiB `vdc` block device.

```
[root@vm1 ~]# lsblk
NAME   MAJ:MIN   RM    SIZE   RO   TYPE   MOUNTPOINT
vda    252:0     0    10G    0   disk
└─vda1  252:1     0     1M    0   part
└─vda2  252:2     0   100M   0   part   /boot/efi
└─vda3  252:3     0   9.9G   0   part   /
vdb    252:16    0     1M    0   disk
vdc    252:32    0     5G    0   disk  ①
```

- ① The `vdc` block device corresponds to the new disk that you added in a previous step.

- 2.5. Log out of the VM console.

```
[root@vm1 ~]# logout
```

- ▶ 3. Prepare a 10 GiB blank disk for the `vm2` web server. Attach the disk to the VM by using a `virtio` interface. Use the `nfs-storage` storage class.
- 3.1. Go to **Virtualization > VirtualMachines** and select the `vm2` machine.
  - 3.2. Stop the `vm2` machine so that you can use the `virtio` interface to attach the new disk. Either click **Actions > Stop** or select the square stop icon next to the drop-down menu. Click **Stop** to confirm the operation, if needed. Wait for the machine to stop.
  - 3.3. Go to the **Configuration > Storage** tab and then click **Add disk**. Complete the form by using the following information Leave any unmentioned fields with their default settings. At the bottom of the form, notice that the volume mode is set to **Filesystem** by default. Do not change that parameter. Click **Save** to create and attach the disk to the VM.

Field	Value
Name	<code>staticdata</code>
Source	Empty disk (blank)
Size	10 GiB
Type	Disk
Interface	VirtIO
StorageClass	<code>nfs-storage</code>
Apply optimized StorageProfile settings	<i>Checked [implies ReadWriteMany access and Filesystem volume modes]</i>

- 3.4. Wait until OpenShift finishes provisioning the VM disks. Then, either click **Actions > Start**, or select the triangular start icon next to the drop-down menu, to start the `vm2` machine.
- 3.5. Log in to the VM console and confirm that a new 10 GiB device is available. Go to the **Console** tab and then log in as the `root` user with `redhat` as the password. Run the `lsblk` command to list the block devices. Notice the 10 GiB `vdc` block device.



### Note

Even though you create the volume in **Filesystem** mode, OpenShift Virtualization exposes the volume as a block device to the operating system inside the VM. As a consequence, the `lsblk` command that you run inside the VM reports the volume as a new block device.

```
[root@vm2 ~]# lsblk
NAME   MAJ:MIN   RM   SIZE   RO   TYPE   MOUNTPOINT
vda    252:0     0    10G    0   disk
└─vda1 252:1     0    1M    0   part
└─vda2 252:2     0   100M   0   part   /boot/efi
└─vda3 252:3     0   9.9G   0   part   /
vdb    252:16    0    1M    0   disk
vdc    252:32    0    10G    0   disk ①
```

- ① The vdc block device corresponds to the new disk that you added in a previous step.

### 3.6. Log out of the VM console.

```
[root@vm2 ~]# logout
```

## ► 4. Detach the additional disk from the vm2 machine.

- 4.1. Click **Actions > Stop** for the **vm2** machine.
- 4.2. Go to **Configuration > Storage**. At the right side of the row with the **staticdata** disk, click the vertical ellipsis icon : and click **Detach**. Click **Detach** in the confirmation window.
- 4.3. Click **Actions > Start** to start the **vm2** machine.
- 4.4. Log in to the VM console and confirm that the device no longer exists. Go to the **Console** tab and then log in as the **root** user with **redhat** as the password. Run the **lsblk** command to list the block devices. Notice that the command no longer reports the **vdc** device.

```
[root@vm2 ~]# lsblk
NAME   MAJ:MIN   RM   SIZE   RO   TYPE   MOUNTPOINT
vda    252:0     0    10G    0   disk
└─vda1 252:1     0    1M    0   part
└─vda2 252:2     0   100M   0   part   /boot/efi
└─vda3 252:3     0   9.9G   0   part   /
vdb    252:16    0    1M    0   disk
```

### 4.5. Log out of the VM console.

```
[root@vm2 ~]# logout
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-intro
```

# Managing Storage Devices on Virtual Machines

## Objectives

- Manage storage that is attached to virtual machines.

## Connect a Virtual Machine to Multiple Disks

Data volumes enable OpenShift to dynamically create the PVCs that represent the used disks in a VM. The VM disks can be imported from a URL or cloned from an existing PVC. You can attach several data volume resources to a VM. Inside the VM, each data volume shows as a disk that you can partition, format, and then mount by using the operating system tools. The disk type inside the VM depends on the selected interface when you attach the data volume:

### SCSI

The disk shows as a standard SCSI device that most operating systems recognize. Linux systems name these disks in the /dev/sdX format.

### VirtIO

The operating system uses a VirtIO driver that communicates with the virtualization layer for optimal performance. Some operating systems do not provide that driver by default. Using this interface requires an available PCIe slot on the VM to mount the device, which limits the number of disks that you can attach in this manner. Linux systems include the driver and name these disks in the /dev/vdX format.

### VirtIO-SCSI

This interface is available in Red Hat OpenShift 4.16 and later versions for creating hot pluggable disks. The `virtio-scsi` interface relies on a VM controller to provide access for the hot pluggable disks to the `scsi` bus. This approach eliminates the limitations of the `virtio` interface, such as requiring down time for adding and removing disks, or limiting the number of disks that can be attached. Additionally, you can persistently mount a disk on a VM by using this interface to add persistence for the volume.

Use the OpenShift web console to create a data volume and attach it to a VM as an additional disk. Go to **Virtualization > VirtualMachines**, select the VM, and go to **Overview > Storage** in the right side panel. Click **Add disk** and then complete the form.

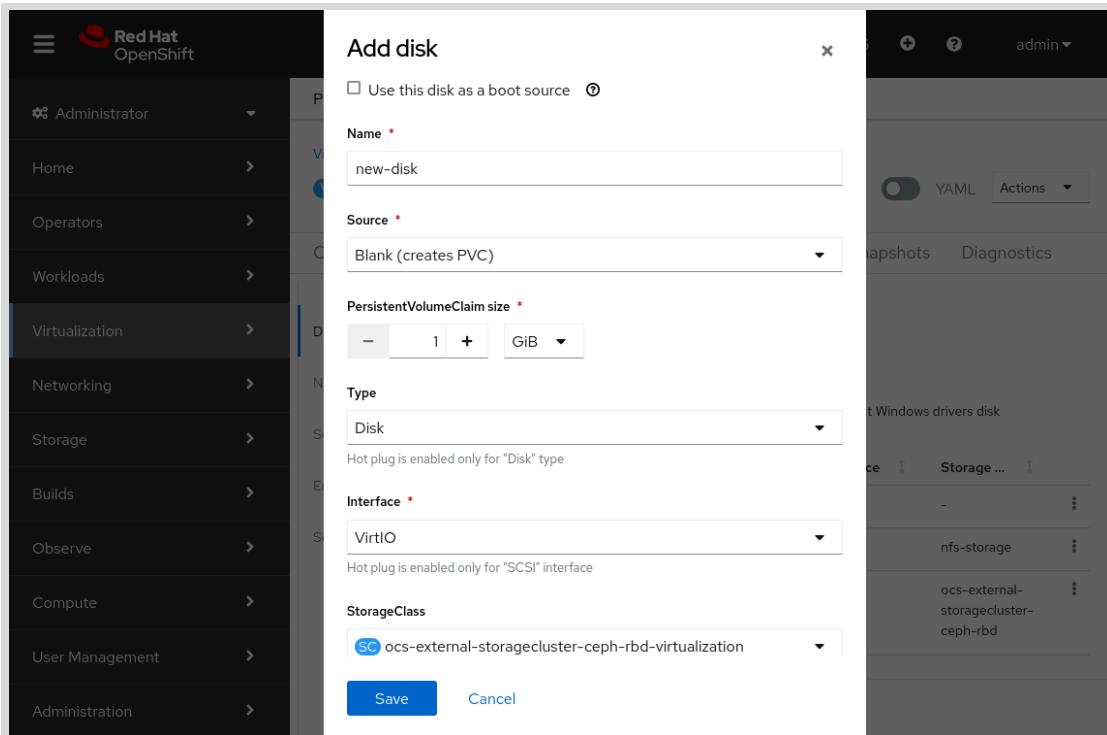


Figure 5.3: Add a disk to a virtual machine

**Note**

When you hot plug a disk to a running VM, SCSI is the only available interface. To use the optimized VirtIO interface, stop the VM before attaching the new disk. This limitation is not present when using the VirtIO-SCSI interface.

## Select a Data Volume Source

The source section of a data volume resource provides the details of the disk image to inject into the PV. You can select the source in the **Add disk** form when using the OpenShift web console. The disk source defines how the new volume is provisioned and made available to the VM.

### Ephemeral disk (container image)

OpenShift Virtualization downloads the container image from a container registry and uses the disk in that container image as the VM disk. This source option is similar to the registry option, except that OpenShift Virtualization does not create a data volume, PVC, or PV resources, and therefore does not consume space on your back-end storage. Although this type of disk is intended for read-only file systems and short-lived VMs, it can be used in other implementations where the durability of data is less important.

Although you can mount the disk read/write to the VM, the data that you write is not persistent. Whenever you restart the VM, OpenShift Virtualization discards your data and restores the disk image to its initial state.

You can use ephemeral storage when you do not need to preserve data, such as temporary files.

## PVC

OpenShift Virtualization copies the PVC that you provide and then uses that copy as the VM disk. Because the source PVC must not be in use for the cloning process to start, you must stop any pods or VMs that are actively using the source PVC.

### Clone existing PVC

OpenShift Virtualization creates and attaches a duplicate of the PVC that you provide as a disk. That PVC might come from a data volume that you detached from another VM and that you want to reuse with your VM.



#### Warning

Do not use a PVC that is currently attached to another VM.

Even though Red Hat OpenShift enables pods and VMs to concurrently access volumes in the `ReadWriteMany` access mode, the file systems on these volumes might not support that mode. The `ext4` or `XFS` file systems, for example, do not support concurrent mounting from multiple systems. Doing so can result in file system corruption or inconsistencies when attempting to read data.

### Empty disk (blank)

The new PVC provides an unformatted raw device. You can use your operating system tools to partition and then format the new block device from inside the VM, after you attach the disk.

### From URL

Red Hat OpenShift Virtualization downloads the virtual disk image from the URL that you provide, and then extracts this image to the new volume. The virtual disk image must be in the raw format or in the `QEMU copy on write version 2 (qcow2)` format. The disk usually includes file systems that you can mount to access their data from inside the VM. Because the disk is read/write, Red Hat OpenShift Virtualization preserves all your changes across VM restarts.

### Registry (creates PVC)

OpenShift Virtualization downloads the container image from a container registry. The container image contains a disk image in the raw or `qcow2` format in the `/disk/` directory. OpenShift Virtualization extracts that disk image and copies it to the volume.

This source option is similar to importing disks from URLs, and is useful when a container registry is already deployed in your organization. Instead of installing a new file server to distribute your disk images, you can store these images in your container registry.

Because container registries can manage only images, you must store your disk images in container images in the `/disk/` directory. These images are not meant to be run as containers, and they usually contain only the `/disk/` directory and the disk image.

### Upload

You can upload an existing `qcow2` disk image to the cluster and use it in a VM. The selected image is uploaded and imported to a PVC that is attached to the VM. You can create the disk image with tools such as `qemu-img` or `virt-make-fs`.

## Detach a Disk from a Virtual Machine

You must stop a VM to detach a disk, unless you are using the `virtio-scsi` interface. After you stop the VM, go to `Configuration > Overview`, click `Overview > Storage` in the right side panel, select the vertical ellipsis icon  $\vdots$  to the right of the disk to detach, and then click `Detach`. When a confirmation message appears, click `Detach` to confirm.

## Delete the Virtual Machine Disk Data

The data is preserved after the disk is detached from a VM. To delete the PVC that contains the disk data, go to **Storage > PersistentVolumeClaims**, and select the project where the VM is located. Locate the PVC with the VM name followed by the disk name, and then click the vertical ellipsis icon. Finally, click **Delete PersistentVolumeClaim**.

A confirmation message indicates that the associated data volume is deleted with the PVC. Click **Delete** to proceed.

You can also delete the data volume resource by using the following command. The underlying persistent volume claim and persistent volume are removed when you delete the data volume.

```
[user@host ~]$ oc delete datavolume/mariadb-server-my-disk
datavolume.cdi.kubevirt.io "mariadb-server-my-disk" deleted
```

## Attach an Existing Disk to a Virtual Machine

After you detach a disk from a VM, you can attach it to a different VM. You can also reattach the disk to the same VM, if needed.

To attach the disk, stop the VM and go to **Overview > Storage**. Click **Add disk** and then complete the form. Choose **PVC** for the source and select the PVC to attach to the VM from the **PersistentVolumeClaim name** drop-down list.



### Warning

Do not use a PVC that is currently attached to another VM. Doing so can result in file system corruption or inconsistencies when attempting to read data.

## Preallocating Disk Space

When OpenShift Virtualization copies a disk image to a PV, it allocates only the space that the image uses.

For example, when you create a 10 GiB blank disk, OpenShift Virtualization does not reserve that space with the back-end storage. Instead, OpenShift Virtualization progressively allocates the space when you start writing data to the disk. This mechanism improves usage of your back-end storage, because it consumes only the space that you use.

However, write speed is often reduced, because the system must allocate the disk blocks before allowing the write operations to proceed.

To preallocate the whole disk space, select the **Enable preallocation** checkbox when adding a disk to a VM. In a data volume resource file, use the **preallocation** parameter:

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: datadisk
spec:
  preallocation: true
  storage:
    resources:
      requests:
```

```
storage: 10Gi
storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
source:
blank: {}
```

With this option enabled, OpenShift Virtualization reserves the entire requested space with the back-end storage. This operation consists of writing zeros to the device's blocks for some back-end storage.



### Note

OpenShift Virtualization always reports the disk size that you request, even if you do not enable preallocation.

## Resize a Virtual Machine Disk

Red Hat OpenShift enables resizing of PVCs. When you change the size of a PVC, Red Hat OpenShift automatically resizes the associated PV. Because most VM disks are based on PVCs, you resize a disk by resizing its PVC.



### Warning

You can only expand PVCs, not shrink them.

## Confirm that the Storage Class Allows Expansion

Each storage class has a field that specifies whether volume expansion is supported. When volume expansion is available, the storage class resource `allowVolumeExpansion` field is set to `true`.

```
[user@host ~]$ oc get storageclass \
ocs-external-storagecluster-ceph-rbd-virtualization -o yaml | \
grep allowVolumeExpansion
allowVolumeExpansion: true
```

The storage class does not support volume expansion if the `allowVolumeExpansion` field is absent or is not set to `true`.

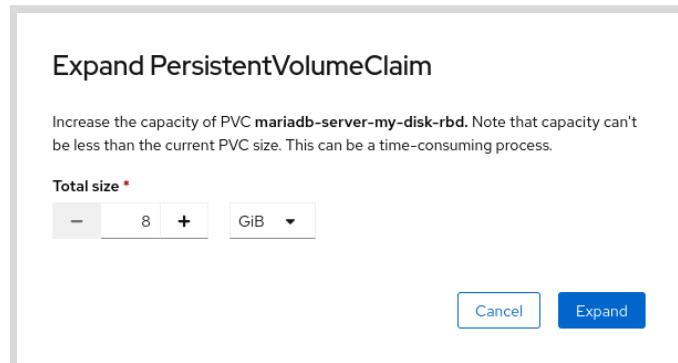
```
[user@host ~]$ oc get storageclass nfs-storage -o yaml | \
grep allowVolumeExpansion ①
```

- ① The output of this command is empty because the `grep` command does not find the `allowVolumeExpansion` string in the YAML representation of the `nfs-storage` resource.

To view the information about the various storage classes in the web console, you can go to **Storage > StorageClasses**, select the name of the storage class, and click **YAML** to inspect the YAML definition. You can verify whether the storage class supports volume expansion through the use of the `allowVolumeExpansion` parameter.

## Expand a Persistent Volume Claim

To expand a PVC by using the OpenShift web console, go to **Storage > PersistentVolumeClaim** and select the PVC to edit. Click **Actions > Expand PVC**, type the new size, and click **Expand** to confirm.



**Figure 5.4: Expand a PVC**

You can expand a PVC by editing its resource file and modifying the `spec.resources.requests.storage` parameter.

```
[user@host ~]$ oc edit pvc/mariadb-server-my-disk-rbd
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb-server-my-disk-rbd
  namespace: storage-multiple
  ...output omitted...
spec:
  accessModes:
    - ReadWriteMany
  ...output omitted...
  resources:
    requests:
      storage: 8Gi ①
  storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
  volumeMode: Block
  volumeName: pvc-c87ee674-4ddc-3efe-a74e-dfe25da5d7b3
  ...output omitted...
```

① `spec.resources.requests.storage`

## Expand the Virtual Machine File System

Resizing a PVC does not require stopping the VM that is using it. However, the VM detects the new disk size only after a restart.

Most operating systems report the new size but do not automatically expand the partitions and the file systems. You must resize the disk to use the additional space with the appropriate tools from within the VM operating system. On Linux systems that enable the `cloud-init` service, these operations might automatically happen during boot.



## References

For more information about resizing volumes, refer to the *Expanding Persistent Volumes* chapter in the *Red Hat OpenShift Container Platform Storage* guide at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/storage/index#expanding-persistent-volumes](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/storage/index#expanding-persistent-volumes)

For more information about the containerized data importer, refer to the *KubeVirt Containerized Data Importer* documentation at [https://kubevirt.io/user-guide/operations/containerized\\_data\\_importer/](https://kubevirt.io/user-guide/operations/containerized_data_importer/)

For more information about data volumes, refer to the *KubeVirt Data Volumes* repository at <https://github.com/kubevirt/containerized-data-importer/blob/main/doc/datavolumes.md>

For more information about data volume preallocation, refer to the *KubeVirt Data Volume Preallocation* repository at <https://github.com/kubevirt/containerized-data-importer/blob/main/doc/preallocation.md>

## ► Guided Exercise

# Attach and Access Storage Devices on a Virtual Machine

Add several new storage devices to a virtual machine, migrate data from an existing storage device to the new storage devices, reconfigure the virtual machine and its applications to use the new storage devices, and then remove the old storage device with the data.

## Outcomes

Improve the storage implementation for a database VM by adding hot pluggable disks to replace the single traditional data volume.

- Create disks with specific requirements.
- Attach disks to an existing VM.
- Detach disks from a VM.
- Verify the implementation.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command also creates the `storage-multiple` namespace and starts the `mariadb-server` VM in that namespace.

```
[student@workstation ~]$ lab start storage-multiple
```

## Instructions

- 1. As the `admin` user, locate and go to the OpenShift web console. Confirm that the `mariadb-server` VM is running.
- 1.1. From a terminal, log in to your Red Hat OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to the web console URL: `https://console-openshift-console.apps.ocp4.example.com`

**Chapter 5 |** Configuring Storage for Virtual Machines

Select **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.

- ▶ 2. Confirm that only the vda, vdb, and vdc disks are attached to the VM.
  - 2.1. Go to **Virtualization > VirtualMachines**. Select the **storage-multiple** project. Confirm that the **mariadb-server** VM is running.
  - 2.2. Click the VM name and go to the **Console** tab. Log in as the **root** user with **redhat** as the password. Run the **lsblk** command to list the block devices.

```
[root@mariadb-server ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda    252:0    0   10G  0 disk
└─vda1 252:1    0    1M  0 part
└─vda2 252:2    0  100M  0 part /boot/efi
└─vda3 252:3    0  9.9G  0 part /
vdb    252:16   0    1M  0 disk
vdc    252:32   0 14.2G  0 disk
```

- 2.3. From the VM console, confirm that the database service is working correctly and that you can access the **sakila** database.

```
[root@mariadb-server ~]# mysql -u devuser -p'developer' sakila
...output omitted...
MariaDB [sakila]> SHOW TABLES;
+-----+
| Tables_in_sakila |
+-----+
...output omitted...
+-----+
23 rows in set (0.001 sec)

MariaDB [sakila]> quit
Bye
```

- 2.4. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

- 2.5. Go to the **Configuration > Storage** tab.

Name	Source	Size	Drive	Interface	Storage
cloudinitdisk	Other	-	Disk	virtio	-
dbroot	PVC dbroot	15.00 GiB	Disk	virtio	nfs-storage

Notice that the storage class for the `dbroot` disk is `nfs-storage`, which is using slow NFS shares as back-end storage. The MariaDB server uses that `dbroot` disk to store the database files.

- 3. Create and attach three 5 GiB disks to the `mariadb-server` VM. Use the fast `ocs-external-storagecluster-ceph-rbd-virtualization` storage class in `Block` mode. Use the SCSI interface to hot plug the disks to the VM without stopping the machine.
- 3.1. Go to the **Configuration > Storage** tab. Click **Add disk** to create and attach the `dbdata` disk. Complete the form by using the following information and click **Add** to create and attach the disk to the VM.
- Repeat this step for the `dbredo` and `dbundo` disks. You configure that disk for the MariaDB server data files in another step.

Field	Value
Use this disk as a boot source	<i>Unset</i>
Name	<ul style="list-style-type: none"><li>• <code>dbdata</code></li><li>• <code>dbredo</code></li><li>• <code>dbundo</code></li></ul>
Source	Empty disk (blank)
Size	5 GiB
Type	Disk
Interface	SCSI
StorageClass	<code>ocs-external-storagecluster-ceph-rbd-virtualization</code>
Apply optimized StorageProfile settings	<i>Checked</i> [implies <code>ReadWriteMany</code> access and <code>Block</code> volume modes]
Enable preallocation	<i>Unset</i>

- 3.2. Verify that the `Persistent Hotplug` label is displayed on the new volumes and wait until OpenShift finishes provisioning the VM disks.

dbdata Persistent H...	PVC mariadb-server-dbdata	5.00 GiB	Disk	SCSI	ocs-external-storagecluster-ceph-rbd-virtualization	...
dbredo Persistent H...	PVC mariadb-server-dbredo	5.00 GiB	Disk	SCSI	ocs-external-storagecluster-ceph-rbd-virtualization	...
dbroot	PVC dbroot	15.00 GiB	Disk	virtio	nfs-storage	...
dbundo Persistent H...	PVC mariadb-server-dbundo	5.00 GiB	Disk	SCSI	ocs-external-storagecluster-ceph-rbd-virtualization	...

- 3.3. Log in to the VM console and confirm that three new 5 GiB devices are available. Go to the **Console** tab and log in as the **root** user with **redhat** as the password.

You might see the system messages for configuring the new SCSI devices. Wait until the **Attached SCSI disk** message is displayed for the three new volumes before proceeding.

- 3.4. Run the **lsblk** command to list the block devices. Notice the three 5 GiB block devices.

```
[root@mariadb-server ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda     8:0      0   5G  0 disk ①
sdb     8:16     0   5G  0 disk ②
sdc     8:32     0   5G  0 disk ③
vda    252:0     0  10G  0 disk
└─vda1 252:1     0   1M  0 part
└─vda2 252:2     0 100M  0 part /boot/efi
└─vda3 252:3     0  9.9G 0 part /
vdb    252:16    0   1M  0 disk
vdc    252:32    0 14.2G 0 disk
```

- ① The **sda** device corresponds to the **dbdata** disk.
- ② The **sdb** device corresponds to the **dbredo** disk.
- ③ The **sdc** device corresponds to the **dbundo** disk.

- 4. The **lab** command prepared the **/root/dbmove.sh** script to format the three disks and to move the database files to these disks. Run the script to migrate the MariaDB server data.

- 4.1. From the **mariadb-server** VM console, run the script:

```
[root@mariadb-server ~]# ./dbmove.sh
...output omitted...
#### Starting the MariaDB Server
Done
```

- 4.2. Confirm that the database service is working correctly and that you can access the sakila database.

```
[root@mariadb-server ~]# mysql -u devuser -p'developer' sakila
...output omitted...
MariaDB [sakila]> SHOW TABLES;
+-----+
| Tables_in_sakila      |
+-----+
...output omitted...
+-----+
23 rows in set (0.001 sec)

MariaDB [sakila]> quit
Bye
```

- 4.3. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

- 5. Detach and then delete the dbroot disk. The MariaDB server stored its data on that disk before you moved the data to the three new disks.

- 5.1. Click **Actions > Stop**, and wait for the machine to stop.
- 5.2. Go to the **Configuration > Storage** tab. Click the vertical ellipsis icon : to the right of the dbroot disk and then click **Detach**. In the confirmation window, click **Detach** to confirm.
- 5.3. Click **Actions > Start**, and wait until the VM is running.
- 5.4. Log in to the VM console and confirm that the 15 GiB /dev/vdc device is no longer shown. Go to the **Console** tab and log in as the **root** user with **redhat** as the password. Run the **lsblk** command to list the block devices. Notice that the deleted 15 GiB block device is no longer listed.

```
[root@mariadb-server ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda     8:0    0   5G  0 disk /mariadb/data
sdb     8:16   0   5G  0 disk /mariadb/redo
sdc     8:32   0   5G  0 disk /mariadb/undo
vda    252:0   0  10G  0 disk 
└─vda1 252:1   0   1M  0 part
└─vda2 252:2   0 100M  0 part /boot/efi
└─vda3 252:3   0  9.9G  0 part /
vdb    252:16  0   1M  0 disk
```

- 5.5. Confirm that the database service is working correctly and that you can access the sakila database. The database is now running from the hot pluggable disk.

```
[root@mariadb-server ~]# mysql -u devuser -p'developer' sakila
...output omitted...
MariaDB [sakila]> SHOW TABLES;
```

```
+-----+  
| Tables_in_sakila      |  
+-----+  
...output omitted...  
+-----+  
23 rows in set (0.001 sec)  
  
MariaDB [sakila]> quit  
Bye
```

5.6. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-multiple
```

# Connecting a Virtual Machine to External Storage

## Objectives

- Connect a virtual machine to external storage by using Multus.

## Prepare External Storage

When you create a PVC to request a volume for your application, Kubernetes prepares a PV and then binds it to the PVC. This approach differs from the storage concept that VMware uses, because administrators define each PVC for VMs in OpenShift instead of relying on a single shared datastore. Cluster administrators must grasp this paradigm shift to storage PVCs within OpenShift to deliver a smooth migration to the platform during VMware migrations.

With dynamic provisioning, Kubernetes creates persistent volumes on demand. A cluster provisioner controls the back-end storage.

However, not all storage types have a provisioner. In some organizations, dynamic provisioning is not available because the back-end storage is managed manually or is used by applications that Kubernetes does not control.

In those environments, Kubernetes uses static provisioning. Storage administrators prepare volumes in the back-end storage, and then Kubernetes administrators declare these volumes in PV resources. When you create the PVC, Kubernetes binds one available PV to the claim.

## Connect a Virtual Machine to External Storage

You can use one of these options to connect a VM to external storage:

- Connect the storage from inside the VM.
- Connect the storage through a Kubernetes PV and then attach the volume as a VM disk.

The first option connects the storage directly from inside the VM by using the VM operating system tools. Each operating system uses a different approach, which a system administrator should know for the relevant environment. You can choose this solution for storage that you access through the network, such as Server Message Block (SMB), Network File System (NFS), or Internet Small Computer Systems Interface (iSCSI).

If you access the back-end storage through a dedicated network, then cluster administrators can use the Kubernetes NMState Operator to configure that network on the cluster nodes. You can then use the Multus plug-in to add a network interface that is connected to the storage network to a VM instance. The Kubernetes NMState Operator and Multus are explained in more detail elsewhere in this course.

Another option is to prepare a static PV for the external storage, create a PVC that binds to the PV, and then attach that PVC as a disk inside the VM. As with the first option, cluster administrators can use the Kubernetes NMState Operator to configure the cluster nodes that you access for network storage through a dedicated network.

## Create a Persistent Volume

A PV represents a piece of storage, and provides all the required details for the cluster nodes to connect to that storage.

The following example shows the resource file for a PV that describes the connection to an iSCSI target:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 50Gi 1
  accessModes:
    - ReadWriteOnce 2
  volumeMode: Block 3
  claimRef: 4
    name: websrv1-staticimsgs
    namespace: vm-project
  iscsi: 5
    targetPortal: 192.168.51.40:3260
    iqn: iqn.1986-03.com.ibm:2145.disk1
    lun: 0
    initiatorName: iqn.1994-05.com.redhat:openshift-nodes
```

- 1** The volume size should match the size of the device that your storage administrators prepared.
- 2** The available access modes depend on the storage type. For iSCSI, only `ReadWriteOnce` and `ReadOnlyMany` are enabled. Because VM live migration requires the `ReadWriteMany` access mode, live migration is not available when using iSCSI PVs.
- 3** For VM disks, Red Hat recommends using the `Block` volume mode. For iSCSI PVs, you can choose between `Block` and `Filesystem` modes. Some other PV types, such as NFS, support only the `Filesystem` mode.

If you set the mode to `Filesystem`, then Kubernetes automatically formats the device. If you use that volume for a VM disk, then Red Hat OpenShift Virtualization creates a disk image file in the file system.

- 4** By default, Kubernetes does not reserve PVs. If a developer creates a PVC before you, then Kubernetes might bind it to your PV. For Kubernetes to reserve a PV for a specific PVC, provide the details of that PVC in the `claimRef` section.
- 5** The `iscsi` section provides the details of the remote iSCSI target. Your storage administrators must provide you with that configuration.

The following example shows the resource file for a PV that describes the connection to an NFS share:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
```

```

spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany ①
  volumeMode: Filesystem ②
  claimRef:
    name: websrv1-logs
    namespace: vm-project
  nfs: ③
    path: /exports-ocp/vm135
    server: 10.20.42.42

```

- ① NFS PVs support the `ReadWriteMany` access mode. That mode enables VM live migration.
- ② For NFS PVs, Kubernetes supports only the `Filesystem` mode. Because the default value for the `volumeMode` parameter is `Filesystem`, you could omit the parameter. If you use the PV for a VM disk, then OpenShift Virtualization creates a disk image file in the NFS share.
- ③ The `nfs` section provides the NFS share connection details. Your storage administrators must provide you with that configuration.

The following example shows the resource file for a PV that describes the connection to a Fibre Channel (FC) volume. Using FC implies that the cluster nodes have host bus adapters (HBAs) that are connected to an FC storage array.

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: fc-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce ①
  volumeMode: Block ②
  claimRef:
    name: dbsrv1-binlog
    namespace: vm-project
  fc: ③
    targetWNNs:
      - "50060e801049cf1"
    lun: 0

```

- ① FC PVs support the `ReadWriteOnce` access mode.
- ② For FC PVs, Kubernetes supports only the `Block` mode.
- ③ The `fc` section provides the connection details. Your storage administrators must provide you with that configuration.

PVs are global objects and are not tied to a project. Only cluster administrators can create PVs.

To create a PV from a resource file, run the `oc apply -f file.yaml` command as a cluster administrator.

## Chapter 5 | Configuring Storage for Virtual Machines

You can also use the OpenShift web console to create PVs. Log in to the web console as an administrator, go to **Storage > PersistentVolumes**, click **Create PersistentVolume**, and then edit the resource file in the YAML editor.

Name	Status	Claim	Capacity	Labels	Created
pvc-09f7d74-5fd2-4946-bcld-60b6bf4fa18	Bound	PVC db-noobaa-db-pg-0	50Gi	No labels	Nov 10, 2024, 10:53 AM
pvc-9799e59a-544a-4cc9-a64b-e46e16e6bb98	Bound	PVC registry-claim	20Gi	No labels	Nov 2, 2024, 11:39 AM

Figure 5.7: Creating a persistent volume in the web console

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: fc-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
  claimRef:
    name: dbsrv1-binlog
    namespace: vm-project
  fc:
    targetWWNs:
      - "50060e801049cf1"
    lun: 0

```

Figure 5.8: Editing the persistent volume YAML definition

Click **Create** and the resulting page shows the details for the new PV.

## Create a Persistent Volume Claim

If you created a PV that contains a `claimRef` section, then you must use the corresponding name and namespace for your PVC.

The following example shows the resource file for a PVC:

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: websrv1-staticimsgs
  namespace: vm-project
spec:

```

```
resources:  
  requests:  
    storage: 50Gi  
accessModes:  
  - ReadWriteOnce  
volumeMode: Block
```

You do not need cluster administrator privileges to create PVCs. To create a PVC from a resource file, run the `oc apply -f file.yaml` command.

You can also use the OpenShift web console to create PVCs. Go to **Storage > PersistentVolumeClaims**, click **Create PersistentVolumeClaim > With Form**, and then complete the form.

The screenshot shows the 'PersistentVolumeClaims' list page. At the top right, there is a 'Create PersistentVolumeClaim' button with a dropdown menu. The 'With Form' option is highlighted with a red box. Below the list, there is a table with columns: Name, Status, PersistentVolumes, and Capacity. One row is shown: 'PVC mariadb-server' (Status: Bound, PersistentVolume: pvc-f918e740-0593-4aff-8e01-175cafd70e34, Capacity: 10 GiB).

Figure 5.9: Creating a persistent volume claim in the web console

Ensure that you use the correct name, access mode, size, and volume mode for the PVC so that Kubernetes binds the PV that you prepared. Keep the default value for the `StorageClass` parameter. Kubernetes does not use that parameter when you explicitly configure the bind.

**Create PersistentVolumeClaim**

**StorageClass**

SC nfs-storage

StorageClass for the new claim

**PersistentVolumeClaim name \***

my-storage-claim

A unique name for the storage claim within the project

**Access mode \***

Single user (RWO)    Shared access (RWX)    Read only (ROX)

Access mode is set by StorageClass and cannot be changed

**Size \***

-	+	GiB ▾
---	---	-------

Desired storage capacity

Use label selectors to request storage

PersistentVolume resources that match all label selectors will be considered for binding.

**Volume mode \***

Filesystem    Block

**Create**   **Cancel**

**Figure 5.10: Form for creating a persistent volume claim**

Click **Create** and the resulting page shows the details for the new PVC.

**Note**

Kubernetes does not delete the static PV when you delete the PVC. To delete a PV requires a user account with cluster administrator privileges. To create a PVC that reuses the PV, a cluster administrator must release the PV first. To release the PV, a cluster administrator can edit the resource and remove the `uid` parameter from the `claimRef` section:

```
...output omitted...
claimRef:
  apiVersion: v1
  kind: PersistentVolumeClaim
  name: websrv1-staticimgs
  namespace: vm-project
  resourceVersion: "325655"
# uid: d9c1a805-f1e8-4370-8d6e-c450dc9c3ef3
...output omitted...
```

## Attach a Persistent Volume to a Virtual Machine

When you attach a PVC as a VM disk, OpenShift Virtualization redirects all the disk read/write operations to the underlying volume on the node. The node then forwards the operations to the back-end storage.

### Inject a Disk Image into the Volume

After you create the PVC, the associated volume is empty. If you attach the volume as a VM disk, then you get an empty disk that you can partition, format, and mount.

However, if you have a disk image in the IMG, ISO, or QCOW2 format, then you can prepopulate the volume.

Use the `virtctl` command to inject a disk image into the PVC:

```
[user@host ~]$ virtctl image-upload pvc websrv1-staticimsgs \
--image-path=./webimgfs.qcow2 --no-create
Using existing PVC vm-project/websrv1-staticimsgs
Waiting for PVC websrv1-staticimsgs upload pod to be ready...
Pod now ready
Uploading data to https://cdi-uploadproxy-openshift-cnv.apps.ocp4.example.com

249.88 MiB / 249.88 MiB [=====] 100.00% 1s

Uploading data completed successfully, waiting for processing to complete, you can
hit Ctrl-C without interrupting the progress
Processing completed successfully
Uploading ./webimgfs.qcow2 completed successfully
```

From the OpenShift web console, you inject the disk image at the same time that you create the PVC. Go to **Storage > PersistentVolumeClaims**, click **Create PersistentVolumeClaim > With Data upload form**, and then complete the form.

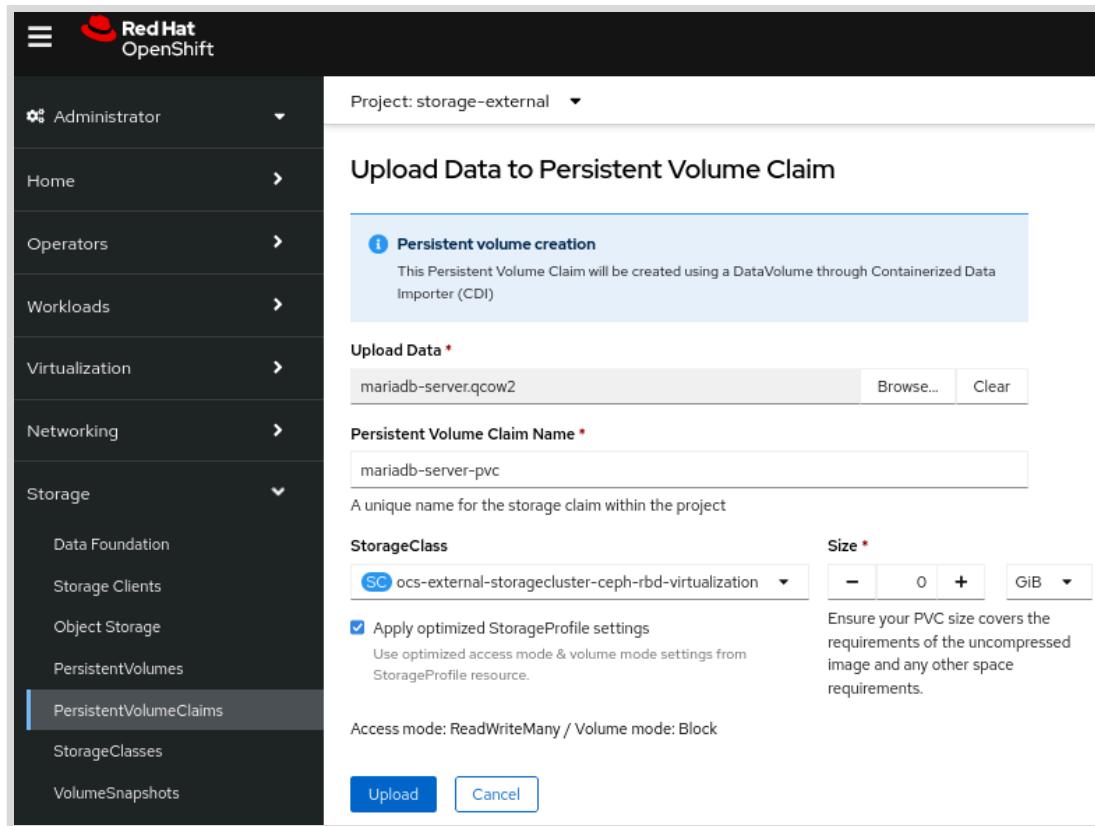


Figure 5.11: Create a PVC and upload a disk image

## Attach the External Volume as a Disk

Use the OpenShift web console to attach the PVC to a VM. Go to **Virtualization > VirtualMachines**, select the VM, and then go to the **Configuration > Storage** tab. Click **Add disk** and complete the form:

- Enter a name for the disk in the **Name** field.
- Select **PVC** in the **Source** field.
- Select your PVC in the **PVC name** field.

Click **Save** to attach the disk.

Remember that you must stop the VM to use the **virtio** interface.



## References

For more information about the access modes in the available per volume plugin, refer to the *Access Modes* section in the *Red Hat OpenShift Container Platform Storage* guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/storage/index#pv-access-modesUnderstanding-persistent-storage](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/storage/index#pv-access-modesUnderstanding-persistent-storage)

For more information about injecting a disk image into a volume, refer to the *Creating VMs by Uploading Images* section in the *Red Hat OpenShift Container Platform Virtualization* guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-creating-vms-uploading-images](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-creating-vms-uploading-images)

For more information about persistent volumes, refer to the *How to Manually Reclaim and Reuse OpenShift Persistent Volumes That Are Released* article at

<https://access.redhat.com/solutions/4651451>

For more information about provisioners, refer to the *Kubernetes List of Provisioners* article at

<https://kubernetes.io/docs/concepts/storage/storage-classes/#provisioner>

For more information about persistent volumes, refer to the *Kubernetes Persistent Volume Reference* article at

<https://kubernetes.io/docs/reference/kubernetes-api/config-and-storage-resources/persistent-volume-v1/>

## ► Guided Exercise

# Connect a Virtual Machine to External Storage

Add a dedicated storage device, which your SAN provides by using the iSCSI protocol, to a virtual machine that runs MariaDB that will use the storage for a new database.

## Outcomes

- Create a Persistent Volume to declare an iSCSI target.
- Create a Persistent Volume Claim and attach it as a VM disk.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command also creates the `storage-external` namespace and starts a virtual machine named `mariadb-server` in that namespace.

```
[student@workstation ~]$ lab start storage-external
```

## Instructions

- 1. As the `admin` user, locate and go to the OpenShift web console. Confirm that the `mariadb-server` VM is running.
- 1.1. From the command line, log in to your Red Hat OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- 1.4. Go to `Virtualization > VirtualMachines`.
- 1.5. Select the `storage-external` project.

- 1.6. Confirm that the `mariadb-server` VM is running.
- ▶ 2. Create the PV resource that declares the iSCSI target to connect.
  - 2.1. From the command line on the workstation machine, review the `~/D0316/labs/storage-external/pv.yaml` resource file that the lab command prepared.

```
[student@workstation ~]$ cat ~/D0316/labs/storage-external/pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
  claimRef: ①
    name: mariadb-binlog
    namespace: storage-external
  iscsi: ②
    targetPortal: 192.168.51.40:3260
    iqn: iqn.2021-10.com.example:server.disk1
    lun: 0
    initiatorName: iqn.2021-10.com.example:client-server.disk1
```

- ① The `claimRef` section specifies that the PV is for a PVC named `mariadb-binlog` in the `storage-external` namespace. A following step instructs you to create that PVC.
- ② The `iscsi` section provides the details of the iSCSI target in the classroom environment.

- 2.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/storage-external/pv.yaml
persistentvolume/iscsi-pv created
```



### Note

If you prefer the OpenShift web console, then go to **Storage > PersistentVolumes**. Click **Create PersistentVolume** and then copy the contents of the `~/D0316/labs/storage-external/pv.yaml` resource file. Press **Shift+Insert** to paste the contents of the file into the YAML editor. Click **Create**.

- 2.3. Run the `oc get pv` command to verify your work.

```
[student@workstation ~]$ oc get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      ...
iscsi-pv   1Gi        RWO          Retain          Available   ...
...output omitted...
```

- 3. Create the `mariadb-binlog` PVC and confirm that Kubernetes binds it to the PV.

- 3.1. From the command line, review the `~/D0316/labs/storage-external/pvc.yaml` resource file that the `lab` command prepared.

```
[student@workstation ~]$ cat ~/D0316/labs/storage-external/pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb-binlog
  namespace: storage-external
spec:
  resources:
    requests:
      storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
  volumeMode: Block
```

Notice that the `name`, `namespace`, `storage`, `accessModes`, and `volumeMode` parameters match the parameters of the PV resource. The `storageClassName` parameter specifies that the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class is used to dynamically bind the PVC to the PV. Although the PVC is provisioned by this storage class, the PVC is bound to a PV that is backed by an iSCSI target.

- 3.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/storage-external/pvc.yaml
persistentvolumeclaim/mariadb-binlog created
```



### Note

If you prefer the OpenShift web console, then go to **Storage > PersistentVolumeClaims** and select the **storage-external** project. Click **Create PersistentVolumeClaim > With Form** and then complete the form by using the following information:

Field	Value
Storage class	<code>ocs-external-storagecluster-ceph-rbd-virtualization</code>
Persistent Volume Claim name	<code>mariadb-binlog</code>
Access mode	Single user ( <code>RW0</code> )
Size	1 GiB
Volume mode	Block

Click **Create**.

- 3.3. Run the `oc get pvc` command to confirm that Kubernetes binds your PVC to the existing PV.

```
[student@workstation ~]$ oc get pvc -n storage-external
NAME           STATUS  VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS
mariadb-binlog  Bound   iscsi-pv  1Gi       RWO          ocs-exter...
```

...output omitted...

The command must report the status as `Bound`.



### Note

If you do not get that status, then delete the PVC and the PV and start again. Ensure that you create the PVC in the `storage-external` project with the correct name and parameters.

Use the `oc` command to delete the resources:

```
[student@workstation ~]$ oc delete pvc mariadb-binlog -n storage-external
persistentvolumeclaim "mariadb-binlog" deleted

[student@workstation ~]$ oc delete pv iscsi-pv
persistentvolume "iscsi-pv" deleted
```

- 4. Attach the `mariadb-binlog` PVC as a disk to the `mariadb-server` VM by using a SCSI interface.

- 4.1. Use the OpenShift web console to attach a hot pluggable disk by using the SCSI interface without stopping the VM.

Go to **Virtualization > VirtualMachines**, and select the `mariadb-server` VM.

- 4.2. Go to the **Configuration > Storage** tab and select **Add disk**. Complete the form by using the following information:

Field	Value
Name	<code>binlogdisk</code>
Source	PVC
PVC name	<code>mariadb-binlog</code>
Type	Disk
Interface	SCSI

- 4.3. Click **Save** to create and attach the disk to the VM.

- 4.4. Log in to the VM console and confirm that a new 1 GiB device is available. Go to the **Console** tab and then log in as the `root` user with `redhat` as the password. Run the `lsblk` command to list the block devices. Notice the 1 GiB `sda` block device.

```
[root@mariadb-server ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   1G  0 disk
vda     252:0    0  10G  0 disk
└─vda1 252:1    0   1M  0 part
└─vda2 252:2    0 100M  0 part /boot/efi
└─vda3 252:3    0  9.9G  0 part /
vdb     252:16   0   1M  0 disk
```

4.5. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-external
```

## ► Lab

# Configure Storage for Virtual Machines

Manage storage and disks for virtual machines in Red Hat OpenShift.

## Outcomes

- Attach and detach disks in VMs.
- Create PVs and PVCs for external storage.
- Attach external storage to VMs.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command creates the `storage-review` namespace and starts two virtual machines in that namespace: the `vm1` and the `vm2` machines.

```
[student@workstation ~]$ lab start storage-review
```

## Instructions

1. From a terminal on the `workstation` machine, use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.  
Open a web browser and log in to the OpenShift web console. Confirm that the two VMs are running under the `storage-review` project.

### Cluster API endpoint

<https://api.ocp4.example.com:6443>

### Web console URL

<https://console-openshift-console.apps.ocp4.example.com>

2. A disk named `webroot` is connected to the `vm1` machine by using the VirtIO interface. Detach the disk and then connect it to the `vm2` machine and convert the disk to a hot-pluggable attachment. Use the SCSI interface when you connect the disk to the second VM. Ensure that you preserve the data when you detach the disk.  
Ensure that the `vm1` machine is running after you complete your work.

To verify the disk, log in to the console of the `vm2` machine as the `developer` user with `developer` as the password, and then run the `lsblk` command. If the disk attachment is correctly configured, then the command output includes the new `sda` disk.

3. Create the PV resource named `nfs-pv`, which declares an external NFS share. Create a `5 Gi ReadWriteMany` device that uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class. Ensure that you reserve the PV for a PVC named `weblog` in the `storage-review` namespace.

You create the `weblog` PVC in a following step.

The `lab` command prepared the `~/D0316/labs/storage-review/nfspv.yaml` resource file to use as a model. To use that file, you must adapt it to the requirements.

4. Create the `weblog` PVC in the `storage-review` namespace with the following parameters:

Field	Value
PVC name	<code>weblog</code>
Access mode	<code>ReadWriteMany (RWX)</code>
Size	<code>5 GiB</code>
Volume mode	<code>Filesystem</code>
Storage class	<code>ocs-external-storagecluster-external-ceph-rbd-virtualization</code>

The `lab` command prepared the `~/D0316/labs/storage-review/nfspvc.yaml` resource file that you can use as a model. To use that file, you must adapt it to the requirements.

Confirm that Kubernetes binds the `weblog` PVC to the PV named `nfs-pv`.

5. Attach the `weblog` PVC as a disk to the `vm2` machine by using a SCSI interface.

Ensure that the `vm2` machine is running after you complete your work.

To verify the disk, log in to the console of the `vm2` machine as the `developer` user with `developer` as the password, and then run the `lsblk` command. If the disk attachment is correctly configured, then the command output includes the new `5 GiB` disk named `sdb`.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade storage-review
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-review
```

## ► Solution

# Configure Storage for Virtual Machines

Manage storage and disks for virtual machines in Red Hat OpenShift.

## Outcomes

- Attach and detach disks in VMs.
- Create PVs and PVCs for external storage.
- Attach external storage to VMs.

## Before You Begin

As the student user on the **workstation** machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command creates the `storage-review` namespace and starts two virtual machines in that namespace: the `vm1` and the `vm2` machines.

```
[student@workstation ~]$ lab start storage-review
```

## Instructions

1. From a terminal on the **workstation** machine, use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.  
Open a web browser and log in to the OpenShift web console. Confirm that the two VMs are running under the `storage-review` project.

### Cluster API endpoint

`https://api.ocp4.example.com:6443`

### Web console URL

`https://console-openshift-console.apps.ocp4.example.com`

- 1.1. From a terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Switch to the `storage-review` project.

```
[student@workstation ~]$ oc project storage-review
Now using project "storage-review" on server "https://api.ocp4.example.com:6443".
```

## Chapter 5 | Configuring Storage for Virtual Machines

- 1.3. Open a web browser and go to the OpenShift web console URL. Select **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.
- 1.4. Go to **Virtualization > VirtualMachines** and select the **storage-review** project. Confirm that the **vm1** and **vm2** machines are running.

The screenshot shows the 'VirtualMachines' list page in the OpenShift web console. It displays two virtual machines: 'vm1' and 'vm2'. Both are listed under the 'storage-review' namespace, are running, and have their 'DataVolumesReady' condition set to true. They are both located on the 'master02' node and were created on November 13, 2024, at 12:45 PM and 12:47 PM respectively, with IP addresses 10.9.0.36 and 10.9.0.37. A 'Create' button is visible in the top right corner.

2. A disk named **webroot** is connected to the **vm1** machine by using the **VirtIO** interface. Detach the disk and then connect it to the **vm2** machine and convert the disk to a hot-pluggable attachment. Use the **SCSI** interface when you connect the disk to the second VM. Ensure that you preserve the data when you detach the disk.  
Ensure that the **vm1** machine is running after you complete your work.  
To verify the disk, log in to the console of the **vm2** machine as the **developer** user with **developer** as the password, and then run the **lsblk** command. If the disk attachment is correctly configured, then the command output includes the new **sda** disk.

  - 2.1. Use the OpenShift web console to stop the **vm1** machine. Select the **vm1** machine and then click **Actions > Stop**. Click **Stop** to confirm the operation, and wait for the machine status to update.
  - 2.2. Go to the **Configuration > Storage** tab.

The screenshot shows the 'Configuration > Storage' tab for the 'vm1' machine in the 'storage-review' project. The 'Storage' section is selected. Under 'Disks', there are two entries: 'vm1' (PVC) and 'webroot' (PVC). The 'vm1' disk is listed as bootable. The 'webroot' disk has a 'Source' of 'PVC webroot', a size of 10.00 GiB, is attached via 'Disk' and 'virtio' interface, and uses the 'ocs-external-storagecluster-ceph-rbd-virtualization' storage class. A red box highlights the 'Storage' tab in the sidebar.

- 2.3. At the end of the row of the **webroot** disk, click the vertical ellipsis icon : and then click **Detach**. Click **Detach** to confirm.
- 2.4. Click **Actions > Start**.
- 2.5. Go to **Virtualization > VirtualMachines** and select the **vm2** machine.

- 2.6. Go to the **Configuration > Storage** tab and click **Add disk**. Complete the form by using the following information and click **Save** to confirm.

Field	Value
Use this disk as a boot source	<i>Not selected</i>
Name	webroot
Source	PVC
PVC project	storage-review
PVC name	webroot
Type	Disk
Interface	SCSI

- 2.7. Log in to the VM console and confirm that a new 1 GiB device is available.

Go to the **Console** tab and log in as the **developer** user with **developer** as the password. Run the `lsblk` command to list the block devices. Notice the 1 GiB `sda` block device.

```
[developer@vm2 ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda     8:0    0  1G  0 disk ①
vda    252:0    0 30G  0 disk
└─vda1 252:1    0   1M  0 part
└─vda2 252:2    0 200M 0 part /boot/efi
└─vda3 252:3    0   1G  0 part /boot
└─vda4 252:4    0 28.8G 0 part /
vdb    252:16   0   1M  0 disk
```

① The `sda` device corresponds to the `webroot` disk that uses the SCSI interface.

- 2.8. Log out of the VM console.

```
[developer@vm2 ~]$ logout
```

3. Create the PV resource named `nfs-pv`, which declares an external NFS share. Create a 5 Gi `ReadWriteMany` device that uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class. Ensure that you reserve the PV for a PVC named `weblog` in the `storage-review` namespace.

You create the `weblog` PVC in a following step.

The `lab` command prepared the `~/D0316/labs/storage-review/nfspv.yaml` resource file to use as a model. To use that file, you must adapt it to the requirements.

- 3.1. From the terminal on the `workstation` machine, edit the `~/D0316/labs/storage-review/nfspv.yaml` resource file.

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  volumeMode: Filesystem
  storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
  claimRef:
    name: weblog
    namespace: storage-review
  nfs:
    path: /exports-ocp4/storage-review
    server: 192.168.50.254

```

3.2. Use the oc command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/storage-review/nfspv.yaml
persistentvolume/nfs-pv created
```

As an alternative, from the OpenShift web console, go to **Storage > PersistentVolumes** and click **Create PersistentVolume**. Copy the contents of the resource file and press **Shift+Insert** to paste the contents into the YAML editor. Click **Create**.

3.3. Run the oc get pv command to verify your work.

```
[student@workstation ~]$ oc get pv nfs-pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
nfs-pv    5Gi        RWX           Retain          Available   storage-review/weblog ...
```

4. Create the weblog PVC in the storage-review namespace with the following parameters:

Field	Value
PVC name	weblog
Access mode	ReadWriteMany (RWX)
Size	5 GiB
Volume mode	Filesystem
Storage class	ocs-external-storagecluster-external-ceph-rbd-virtualization

The lab command prepared the ~/D0316/labs/storage-review/nfspvc.yaml resource file that you can use as a model. To use that file, you must adapt it to the requirements.

Confirm that Kubernetes binds the weblog PVC to the PV named nfs-pv.

- From the terminal, edit the ~/D0316/labs/storage-review/nfspvc.yaml resource file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: weblog
  namespace: storage-review
spec:
  resources:
    requests:
      storage: 5Gi
  storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
  accessModes:
    - ReadWriteMany
  volumeMode: Filesystem
```

- Use the oc command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/storage-review/nfspvc.yaml
persistentvolumeclaim/weblog created
```

As an alternative, from the OpenShift web console, go to **Storage > PersistentVolumeClaims**, select the **storage-review** project, and click **Create PersistentVolumeClaim > With Form**. Complete the form by using the following information, and click **Create**.

Field	Value
StorageClass	ocs-external-storagecluster-ceph-rbd-virtualization
PersistentVolumeClaim name	weblog
Access mode	Shared access (RwX)
Size	5 GiB
Use label selectors to request storage	<i>Not selected</i>
Volume mode	Block

- Run the oc get pvc command to confirm that Kubernetes binds your PVC to the nfs-pv persistent volume.

```
[student@workstation ~]$ oc get pvc weblog -n storage-review
NAME      STATUS    VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS     AGE
weblog    Bound     nfs-pv   5Gi       RWX          ocs-extern...  1m
```

- Attach the weblog PVC as a disk to the vm2 machine by using a SCSI interface.  
Ensure that the vm2 machine is running after you complete your work.

To verify the disk, log in to the console of the `vm2` machine as the `developer` user with `developer` as the password, and then run the `lsblk` command. If the disk attachment is correctly configured, then the command output includes the new 5 GiB disk named `sdb`.

- 5.1. Go to **Virtualization > VirtualMachines** and select the `vm2` machine.
- 5.2. Go to the **Configuration > Storage** tab and click **Add disk**. Complete the form by using the following information and click **Save** to confirm.

Field	Value
Use this disk as a boot source	<i>Not selected</i>
Name	<code> weblog</code>
Source	PVC
PVC project	<code>storage-review</code>
PVC name	<code> weblog</code>
Type	Disk
Interface	SCSI

- 5.3. Log in to the VM console and confirm that a new 5 GiB device is available.

Go to the **Console** tab and log in as the `developer` user with `developer` as the password. Run the `lsblk` command to list the block devices. Notice the 5 GiB `sdb` block device.

```
[developer@vm2 ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   1G  0 disk
sdb      8:16   0   5G  0 disk
vda    252:0    0  30G  0 disk
└─vda1 252:1    0   1M  0 part
  ├─vda2 252:2    0 200M 0 part /boot/efi
  ├─vda3 252:3    0   1G  0 part /boot
  └─vda4 252:4    0 28.8G 0 part /
vdb    252:16   0   1M  0 disk
```

- 5.4. Log out of the VM console.

```
[developer@vm2 ~]$ logout
```

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade storage-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-review
```

# Summary

---

- PVs are Kubernetes resources that abstract the users from how the storage is provisioned.
- Developers request and manage persistent storage for their pods and VMs by creating PVC resources.
- VM disks use PVs as their back-end storage.
- In a Kubernetes environment, developers can request persistent storage for applications without applying specific settings for LUNs, NFS, or other storage back ends.
- For optimal performance, Red Hat recommends using volumes in **Block** mode for VM disks.
  - With PVCs in **Block** mode, OpenShift Virtualization transfers a disk image to the volume.
  - The VM disk uses the volume as its back-end device.
- Red Hat OpenShift Virtualization adds the Data Volume resource type.
- OpenShift manages many storage back ends through storage pools that are available for deployments to create disk allocations.
- You can use Red Hat Ceph Storage RBD volumes to enable live migration, snapshots, and disk cloning.
- A Data Volume resource simplifies creating VM disks by grouping a PVC definition and the disk image to inject into the volume.
- The SCSI interface type supports hot plugging disks, and no longer requires stopping VMs to attach and detach disks.
- Kubernetes supports static or dynamic provisioning of storage.
  - With static provisioning, cluster administrators must create a PVC that binds to the external PV, and then attach that PVC as a disk inside the VM.
  - With dynamic provisioning, Kubernetes creates persistent volumes on demand.



## Chapter 6

# Create and Restore Backups of Virtual Machines

### Goal

Create virtual machine snapshots and back up virtual machine components individually and by using the OpenShift APIs for Data Protection (OADP) operator.

### Sections

- Exporting and Importing Virtual Machine Data and Settings (and Guided Exercise)
- Managing Virtual Machine Snapshots (and Guided Exercise)
- OADP Operator Deployment and Features (and Guided Exercise)
- Back up and Restore with OADP (and Guided Exercise)

### Lab

- Back up and Restore a Virtual Machine with OADP

# Exporting and Importing Virtual Machine Data and Settings

## Objectives

- Export virtual machine data and settings from one project and import them into another project.

## Export and Import a Kubernetes Application

The ability to export and import a Kubernetes application is useful in many scenarios, such as the following ones:

- Partial or full restoration to a previous working state
- Disaster recovery and business continuity
- Migration from one cluster to another
- Duplication on multiple environments

Backup and restore procedures are critical for an organization to recover from data loss or corruption. The most common data loss scenarios are hardware failures, cyberattacks, software bugs, or human errors.

A Kubernetes application backup includes all the needed resources to restore that application to a previous working state. This backup must include the following artifacts:

- Kubernetes resources that define the application and its settings.
- Container images in the internal registry that containers of this application use.
- Data that is stored in persistent volumes or object storage for a stateful application.



### Note

Application data or a configuration that is hosted on external services such as Database-as-a-Service or object storage might be required but is outside the scope of this chapter.

Red Hat OpenShift and Red Hat partners provide data protection solutions for a faster recovery plan. The following list includes examples of data protection solutions:

- OpenShift API for Data Protection
- Veeam Kasten K10
- Storware Backup and Recovery
- IBM Spectrum Protect Plus
- Pure Storage Portworx Backup

## Backing up Application Resources

A Kubernetes application has many resources. The export and import include the following steps:

- List all required resources for the application.
- Export the listed resources.
- Clean the exported resource files.
- Deploy the cleaned resource files.

The first step to export an application is to list all the application resources.



### Note

The `oc get all` command lists only a subset of resources in a project and does not show resources such as secrets and configuration maps.

You do not need to list all resources in a project. A MySQL application might require the following resources:

- Deployment
- Service
- Secret

You can list these resources by using the `oc get` command.

```
[user@host ~]$ oc get deployment,svc,secret
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql   1/1     1           1           112s

NAME          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE
service/mysql  ClusterIP  172.30.210.241  <none>        9001/TCP    115s

NAME          TYPE          DATA   AGE
secret/dockercfg-skqdg  kubernetes.io/dockercfg  1       2m16s
secret/token-cpmxf      kubernetes.io/service-account-token  4       2m16s
...output omitted...
secret/mysql-credentials Opaque          3       60s
```

A MySQL application might use persistent volume claim (PVC) for data. The required resources for application export depend on the application.

## Export Application Resources

You can export an application object to a YAML file by using the `oc -n prod get` command:

```
[user@host ~]$ oc -n prod get deployment/mysql -o yaml > backup_deployment.yaml
```

The `backup_deployment.yaml` file has all deployment details such as specifications, metadata, and status.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "2"
  labels:
    app: mysql
  name: mysql
  namespace: prod
  ...output omitted...
spec:
  progressDeadlineSeconds: 600
```

```

replicas: 1
selector:
  matchLabels:
    app: mysql
    ...output omitted...
status:
  availableReplicas: 1
...output omitted...

```

You can export all other application resources by using the same `oc -n prod get` command.

## Import Application Resources

You can create the application resources in a new project by using the backup YAML files. You can remove the `metadata.namespace` field and use the `oc create -f` command to create each resource in the new project.

Apart from the `metadata.namespace` field, all backup resource files have runtime and status information such as `metadata.annotations`, `metadata.creationTimestamp`, `metadata.resourceVersion`, `metadata.generation`, and `status`. You can remove these fields and keep cleaned backup resource files to create resources in a new project.

The following example demonstrates importing some required resources for an application:

```

[user@host ~]$ oc create -f clean_backup_deployment.yaml -n prod-backup
deployment.apps/mysql created

[user@host ~]$ oc create -f service-mysql.yaml -n prod-backup
The Service "mysql" is invalid: spec.clusterIPs: Invalid value:
[]string{"172.30.210.241"}: failed to allocate IP 172.30.210.241: provided IP is
already allocated

```

The `mysql` deployment was created successfully in the `prod-backup` project by using the backup YAML file.

However, the `metadata` and `status` removal does not work for all resources. Some resources require additional modifications.

The service resource creation failed because the IP address is already allocated to the `mysql` service in the `prod` project. The service backup YAML file requires more modification for a successful creation.

The modification depends on the resource type. Service resource creation requires removing not only the `metadata` and `status` fields but also the `spec.clusterIP` field from the YAML file.

You can use a text editor or other tools to remove these fields. For example, you can use the `jq` tool to process the YAML file and to remove a specific field.

```

[user@host ~]$ cat service-mysql.yaml \
| jq d - metadata.namespace \
| jq d - spec.clusterIP* > clean-service-mysql.yaml

[user@host ~]$ oc create -f clean-service-mysql.yaml -n prod-backup
service/mysql created

```

**Note**

The `yq` tool is a command-line YAML processor that is similar to the `jq` tool for JSON files. The `d` option removes the specified field from the output.

See the references section for more information about the `yq` tool.

Similar to the service resource, the route resource creation requires removing the `metadata.namespace` and `spec.host` fields from the YAML file.

```
[user@host ~]$ cat route-frontend.yaml \
| yq d - metadata.namespace \
| yq d - spec.host > clean-route-frontend.yaml

[user@host ~]$ oc create -f clean-route-frontend.yaml -n prod-backup
route.route.openshift.io/etherpad created
```

You do not need to create all the exported resources. The new project creates some resources, such as service accounts, secrets, and role bindings. An application also creates some resources. The deployment resource creates the replicsets resource, and the deploymentconfig resource creates the replicationcontrollers resource.

## Backing up Application Data

Several methods exist for data backup, depending on the application. Some applications provide dedicated tools or procedures to achieve the most reliable data protection and consistency. Different consistency levels can be achieved depending on the backup method.

### Inconsistent backup

A backup is known as *inconsistent* when the application alters data during the backup process. Traditional data copying when the application is running creates inconsistent backups.

### Crash-consistent backup

A crash-consistent backup is created by suspending disk I/O during the backup, either by using snapshot technology or specialized tools, to ensure data consistency on disk. Application data in memory or pending I/O operations are not captured. The state of the application is kept as if the application was suddenly shut down due to power loss, or crashed.

### Application-consistent backup

Application-consistent backup is the most reliable type of backup because it ensures that all in-memory data and pending I/O operations are written on disk before creating the backup.

Some applications provide a set of tools to flush memory data to disk and to pause file system operations on demand. These tools provide an application-consistent backup without any downtime by using snapshots, which is also known as *hot backup*.

For example, a MySQL database provides the `FLUSH TABLES WITH READ LOCK` statement to flush all operations to disk and to lock all tables before taking a snapshot. You can then unlock tables with the `UNLOCK TABLES` statement after the snapshot is created.

Database applications often come with specialized tools to create and restore backups without stopping the application or using volume snapshots. The following list includes examples of specialized tools:

- `mysqldump` for MySQL and MariaDB

- pg\_dump for PostgreSQL

**Note**

Creating backups with specialized tools is out of the scope of this course.

A general way to create application-consistent backups is to stop the application, copy the data to another location, and then restart the application. This method is also known as *cold backup*, because the application is down during the operation.

Depending on the amount of data, the application can be unavailable for a long time during the backup operation. By using volume snapshot, you can reduce this downtime to only a few minutes and use the cloned volume to back up when the application is back online.

## Volume Snapshot

Volume snapshot capability is available only with Container Storage Interface (CSI) drivers. However, snapshot functions are not implemented for all CSI drivers. The following table includes examples of CSI drivers with the snapshot capability:

Storage provider	CSI driver
AWS Elastic Block Storage	efs.csi.aws.com
Azure Disk	disk.csi.azure.com
CephFS	cephfs.csi.ceph.com
Ceph RBD	rbd.csi.ceph.com
NetApp	csi.trident.netapp.io

Kubernetes provides similar API resources to `PersistentVolume` and `PersistentVolumeClaim` to create and manage volume snapshots.

### VolumeSnapshotClass

Similar to the storage class for a persistent volume claim, a volume snapshot class describes the CSI driver and associated settings to create a volume snapshot.

**Note**

The `VolumeSnapshotClass` driver must match the `StorageClass` provisioner of the source PVC.

The following commands list all available storage and volume snapshot classes:

```
[user@host ~]$ oc get volumesnapshotclasses
NAME                                     DRIVER
ocs-storagecluster-cephfsplugin-snapclass  openshift-storage.cephfs.csi.ceph.com
ocs-storagecluster-rbdplugin-snapclass      openshift-storage.rbd.csi.ceph.com
```

```
[user@host ~]$ oc get storageclasses | egrep "^\w+|csi"
```

NAME	PROVISIONER
ocs-external-storagecluster-cephfs	<b>openshift-storage.cephfs.csi.ceph.com</b>
ocs-external-storagecluster-ceph-rbd	<b>openshift-storage.rbd.csi.ceph.com</b>

## VolumeSnapshot

Similar to the `PersistentVolumeClaim` resource, a `VolumeSnapshot` resource requests creating a snapshot.

The following example is a YAML file for creating a volume snapshot:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: my-snapshot ①
  namespace: application ②
spec:
  volumeSnapshotClassName: ocs-storagecluster-rbdplugin-snapclass ③
  source:
    persistentVolumeClaimName: application-data ④
```

- ① Name of the volume snapshot.
- ② Namespace of the volume snapshot. It must be the same as the source PVC.
- ③ Snapshot class name for the volume snapshot.
- ④ Name of the source PVC that is used for the snapshot.

The following command lists volume snapshots. A snapshot is successfully created when the `READYTOUSE` attribute is set to `true` and a `VolumeSnapshotContent` resource is created:

```
[user@host ~]$ oc get volumesnapshot
NAME      READYTOUSE   SOURCEPVC      ...  SNAPSHOTCONTENT
my-snapshot  true        application-data  ...  snapcontent-798...cf6
```



### Note

When creating an application-consistent backup, the application must be quiesced or scaled down before the snapshot creation. The application can safely be resumed or scaled up after the snapshot is created and ready to use.

## VolumeSnapshotContent

Similar to the `PersistentVolume` resource, a `VolumeSnapshotContent` resource represents a snapshot that a `VolumeSnapshot` resource created.

After the snapshot is created, it can function as a source to create a PVC and for a pod to use it. The following example is a YAML file for creating a persistent volume (PV) from a snapshot:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-snapshot-volume ①
  namespace: application ②
```

```
spec:  
  storageClassName: ocs-external-storagecluster-ceph-rbd ❸  
  accessModes:  
    - ReadWriteOnce  
  dataSource:  
    apiGroup: snapshot.storage.k8s.io  
    kind: VolumeSnapshot  
    name: my-snapshot ❹  
  resources:  
    requests:  
      storage: 1Gi ❺
```

- ❶ Name of the PVC.
- ❷ Namespace of the PVC. It must be the same as the snapshot namespace.
- ❸ Storage class name for the PVC.
- ❹ Name of the snapshot.
- ❺ Size of the new volume. Must be equal to or greater than the snapshot size.

## Export Application Data

You can use any specialized backup tools that your application provides to export the data to your chosen location. The following example is a cron job definition to back up a MariaDB database and to store the backup file on AWS S3 storage:

```
apiVersion: batch/v1  
kind: CronJob  
metadata:  
  name: backup-db  
spec:  
  schedule: "0 0 * * *"  
  jobTemplate:  
    spec:  
      template:  
        spec:  
          initContainers:  
            - envFrom:  
              - secretRef:  
                  name: mariadb  
            image: quay.io/redhattraining/mariadb:10.5  
            command: ["/bin/bash", "-c"]  
            args: ❶  
              - >  
                mariadb-dump -u "${MARIADB_USER}" -p"${MARIADB_PASSWORD}"  
                -h mariadb "${MARIADB_DATABASE}"  
                | bzip2 > /backup/backup-$(date '+%Y%m%d-%H%M').sql.bz2;  
                ls -al /backup  
          name: backup  
          volumeMounts:  
            - mountPath: /backup  
              name: backup  
        containers:
```

```
- image: docker.io/amazon/aws-cli:latest
  command: ["/bin/bash", "-c"]
  args: ②
    - >
      aws s3 cp --no-progress /backup/backup* s3://backup/
  name: s3cli
  volumeMounts:
    - mountPath: /backup
      name: backup
    - mountPath: /root/.aws
      name: aws-creds
  volumes:
    - name: backup
      emptyDir: {}
    - name: aws-creds
      secret:
        secretName: s3config
```

- ① Use the mariadb-dump tool to export the database to a compressed file in an ephemeral volume.
- ② Use the AWS CLI tool to send the backup file to AWS S3-compatible storage.

If the application does not provide backup tools, then you can use volume snapshot to back up the application data. Depending on the storage provider that is available in the OpenShift cluster, you might instead export the snapshot content to an external storage location.

The following example is a job definition that uses an existing snapshot volume to archive the snapshot content, and exports the backup file to a remote S3 bucket:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: backup
  namespace: application
  labels:
    app: backup
spec:
  backoffLimit: 1
  template:
    metadata:
      labels:
        app: backup
    spec:
      containers:
        - name: backup
          image: docker.io/d3fk/s3cmd:latest
          command: ["/bin/sh", "-c"]
          args:
            - | ①
              tar czf -c /snapshot /tmp/mybackup.tar.gz .
              s3cmd cp /tmp/mybackup.tar.gz s3://backup/
      volumeMounts: ②
        - mountPath: /snapshot
          name: snapshot
```

```
...output omitted...
volumes: ③
- name: snapshot
  persistentVolumeClaim:
    claimName: my-snapshot-volume
```

- ① Archive the snapshot content and copy the archive to a remote S3 bucket.
- ② Volume mount definition for the snapshot data in the container.
- ③ Volume definition for the snapshot PVC.



### Important

If the volume storage class does not support volume snapshot, then you can mount the application volume instead to export the data. In this case, you must ensure that no other pods are using the volume during the backup, to avoid data inconsistencies.

Another option is to export the snapshot content locally from a pod by using the `oc cp` command. The following example is a pod definition, where the snapshot data is mounted, which you can use to export the data to your workstation:

```
apiVersion: v1
kind: Pod
metadata:
  name: export
spec:
  containers:
    - image: registry.access.redhat.com/ubi9:latest
      command: ["/bin/bash", "-c"]
      args:
        - sleep infinity ①
        ...output omitted...
      volumeMounts: ②
        - mountPath: /snapshot
          name: snapshot
  volumes: ③
    - name: snapshot
      persistentVolumeClaim:
        claimName: my-snapshot-volume
```

- ① The `sleep infinity` command ensures that the pod stays alive during the manual export.
- ② Volume mount definition for the snapshot data in the container.
- ③ Volume definition for the snapshot PVC.

You can use the following `oc` command to copy the snapshot data from the `export` pod to your local machine:

```
[user@host ~]$ oc cp export:/snapshot /tmp/backup
```

**Note**

The tar binary must be installed in the remote container for the oc cp command to work.

After the snapshot content is exported to a remote location, you can safely remove the snapshot PVC and the volume snapshot, to free up space on the storage back end.

**Important**

If your application uses more than one persistent volume, then you must track the backup names and the volumes that they belong to. You need that information to restore each backup to the correct persistent volume.

## Import Application Data

To import the data to another cluster or namespace, create a pod or a job where the application volume is mounted, and copy the exported data to the pod.

The following example is a YAML file for creating a job that fetches a remote backup from an S3 bucket and extracts the archive to the application volume:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: restore
  namespace: application
  labels:
    app: restore
spec:
  backoffLimit: 1
  template:
    metadata:
      labels:
        app: restore
    spec:
      containers:
        - name: restore
          image: docker.io/d3fk/s3cmd:latest
          command: ["/bin/bash", "-c"]
          args: ①
            - |
              s3cmd cp s3://backup/mybackup.tar.gz /tmp/
              tar xvzf /tmp/mybackup.tar.gz -C /data
      volumeMounts: ②
        - mountPath: /data
          name: application-data
...output omitted...
volumes: ③
  - name: application-data
    persistentVolumeClaim:
      claimName: application-data
```

- ❶ Install the `s3cmd` tool, fetch the backup from the S3 bucket, and extract the archive inside the application data volume.
- ❷ Volume mount definition for the application data in the container.
- ❸ Volume definition for the application PVC.

You can restore the data to an existing volume, or create a one. If you use a new volume, then you must update the application to use that new volume.



### Important

To avoid data corruption, ensure that the application that uses the volume is not running during the restoration procedure.

## Exporting and Importing Virtual Machines

OpenShift Virtualization can export VMs and their associated disks to import them into another OpenShift Virtualization cluster. You can use the `virtctl` command-line tool to export the VM manifest and the disk image, and to import the disk image from compatible hypervisors.

### Exporting a Virtual Machine

OpenShift Virtualization provides a `VirtualMachineExport` custom resource to export a VM and its volumes. The following example shows a `VirtualMachineExport` resource to export the `fedora-vm` VM:

```
apiVersion: export.kubevirt.io/v1alpha1
kind: VirtualMachineExport
metadata:
  name: fedora-export
  namespace: vms
spec:
  source:
    apiGroup: "kubevirt.io"
    kind: VirtualMachine
    name: fedora-vm ❶
    ttlDuration: 1h ❷
```

- ❶ Name of the virtual machine to export
- ❷ Amount of time before the automatic deletion of the export

After the VM export is ready, you can review the download links for the manifests and volumes in the status field with the `oc describe` command.

```
[user@host ~]$ oc get vmexport
NAME      SOURCEKIND      SOURCENAME      PHASE
fedora-export  VirtualMachine  fedora-vm      Ready

[user@host ~]$ oc describe vmexport/fedora-export
Name:      fedora-export
Namespace:  vms
...output omitted...
```

```
Status:  
Links:  
External:  
  Manifests: ①  
    Type: all  
    URL: https://virt-exportproxy-openshift-cnv.apps.ocp4.example.com/.../  
external/manifests/all  
  ...output omitted...  
  Volumes: ②  
    Name: fedora-vm  
    Formats:  
      Format: gzip  
      URL: https://virt-exportproxy-openshift-cnv.apps.ocp4.example.com/.../  
volumes/fedora-vm/disk.img.gz  
  ...output omitted...  
Internal: ③  
  Manifests:  
    Type: all  
    URL: https://virt-export-fedora-export.vms.svc/internal/manifests/all  
  Volumes:  
    Name: fedora-vm  
    Formats:  
      Format: gzip  
      URL: https://virt-export-fedora-export.vms.svc/volumes/fedora-vm/  
disk.img.gz  
  ...output omitted...  
Phase: Ready  
Service Name: virt-export-fedora-export  
Token Secret Ref: export-token-fedora-export ④  
Ttl Expiration Time: 2024-04-19T14:19:16Z ⑤  
Virtual Machine Name: fedora-vm
```

- ① Download links for the VM manifest bundle.
- ② Download links for each VM volume. VM disks are available in two format types: `raw` for an unaltered raw disk image, and `gzip` for a compressed raw disk image.
- ③ Internal links for the manifest and volumes. Those links are valid only inside the cluster.
- ④ Name of the secret with the export token.
- ⑤ Expiration date and time for the export.

**Note**

Each manifest URL is a single line.

The download links are protected with an export token that is generated in the `export-token-export_name` secret. Those links automatically expire after the specified time in the `ttlDuration` value from the `VirtualMachineExport` resource elapsed. If the TTL is not specified, then the default value is two hours. After that time, OpenShift Virtualization deletes the `VirtualMachineExport` resource.

## Chapter 6 | Create and Restore Backups of Virtual Machines

With the `virtctl` command, you can create the `VirtualMachineExport` resource on-demand for a VM and to download manifests and volumes by using the export token from your OpenShift cluster.

To create the `VirtualMachineExport` resource, use the `virtctl vmexport create` command, as follows:

```
[user@host ~]$ virtctl vmexport create \
  --vm=fedora-vm \ ❶
  fedora-export ❷
VirtualMachineExport 'vms/fedora-export' created successfully
```

- ❶ Name of the virtual machine to export
- ❷ Name of the `VirtualMachineExport` resource to create

You can then download the manifest bundle and volumes with the `virtctl vmexport download` command.

```
[user@host ~]$ virtctl vmexport download \
  fedora-export \ ❶
  --keep-vme \ ❷
  --manifest \ ❸
  --include-secret \ ❹
  --output fedora-vm.yml ❺
VirtualMachineExport 'vms/fedora-export' created successfully
```

- ❶ Name of the `VirtualMachineExport` resource.
- ❷ The `virtctl` command automatically deletes the `VirtualMachineExport` resource after the download is complete. Use the `--keep-vme` option to keep the `VirtualMachineExport` resource.
- ❸ Use the `--manifest` option to download the manifest bundle, and the `--volume` option to download the VM disk.
- ❹ Include the export token secret in the manifest bundle.
- ❺ File name for the manifest or disk on your local machine.

The manifest bundle contains the VM manifest, the CA certificate, and the required export token secret for downloading the volume image from the OpenShift cluster. You can import the VM into another OpenShift cluster by using this manifest bundle. The target OpenShift cluster then uses those resources to create the VM and download the VM volume from the source OpenShift cluster.

The following extract shows the content of the manifest bundle from the previous example:

```
---
apiVersion: v1
kind: ConfigMap ❶
metadata:
  name: export-ca-cm-fedora-export
...output omitted...
---
```

```

apiVersion: v1
kind: Secret 2
metadata:
  name: header-secret-fedora-export
...output omitted...
---
apiVersion: kubevirt.io/v1
kind: VirtualMachine 3
metadata:
...output omitted...
  name: fedora-vm
  namespace: vms
spec:
  dataVolumeTemplates:
    - apiVersion: cdi.kubevirt.io/v1beta1
      kind: DataVolume 4
      metadata:
        creationTimestamp: null
        name: fedora-vm
      spec:
        source:
          http:
            certConfigMap: export-ca-cm-fedora-export
            secretExtraHeaders:
              - header-secret-fedora-export
            url: https://virt-exportproxy-openshift-cnv.apps.ocp4.example.com/.../
volumes/fedora-vm/disk.img.gz 5
...output omitted...

```

- 1** Configuration map with the CA certificate
- 2** Secret with the export token
- 3** Virtual machine definition
- 4** Data volume definition that uses the CA certificate and the export token to download the volume image from the source OpenShift Virtualization cluster
- 5** The source URL, which is a single line.

You can also manually download the VM volume by using the `--volume` option with the `virtctl vmexport download` command, as follows:

```
[user@host ~]$ virtctl vmexport download \
  fedora-export \
  --keep-vme \
  --volume fedora-vm \ 1
  --output fedora-vm-disk.img.gz 2
Downloading file: 1.02 MiB [=====>] 235.20 KiB p/s
Download finished successfully
```

- 1** Name of the volume to export.
- 2** File name for the volume image on your local machine. You can use the `.img` extension for the raw volume image, and the `.img.gz` extension for the compressed image.

## Chapter 6 | Create and Restore Backups of Virtual Machines

To ensure data consistency, you must stop the VM before exporting it. If the VM is running, then the `VirtualMachineExport` resource stays in the `Pending` state until the VM is shut down.

To export data from a running VM, you must first create a VM snapshot, and then use the `virtctl vmexport` command to export the VM from the snapshot. In the following example, the `virtctl` command creates a VM export from the `fedora-snapshot` VM snapshot:

```
[user@host ~]$ virtctl vmexport create \
--snapshot=fedora-snapshot \
fedora-export-snapshot
VirtualMachineExport 'vms/fedora-export-snapshot' created successfully

[user@host ~]$ oc get vmexport
NAME                  SOURCEKIND      SOURCENAME    PHASE
fedora-export-snapshot  VirtualMachineSnapshot fedora-snapshot Ready
```



### Note

VM snapshots are explained in more detail later in this course.

After the export is complete, you can remove the VM export by using the `oc delete vmexport` or `virtctl vmexport delete` commands:

```
[user@host ~]$ virtctl vmexport delete fedora-export-snapshot
VirtualMachineExport 'vms/fedora-export-snapshot' deleted successfully
```

## Importing a Virtual Machine

You can import a VM from another OpenShift Virtualization cluster by using the manifest bundle that you export with the `virtctl vmexport` command. The target cluster uses the certificate and export token from the manifest to download the VM disk image from the source cluster.

Use the `oc apply` command to import the manifest bundle that you create with the `virtctl vmexport` command:

```
[user@host ~]$ oc apply -f fedora-vm.yml
configmap/export-ca-cm-fedora-export created
virtualmachine.kubevirt.io/fedora-vm created
secret/header-secret-fedora-export created
```



### Note

OpenShift Virtualization imports the VM into the same namespace as the source cluster. You must create that namespace on the target cluster before importing the VM.

You can also import disk images from another OpenShift Virtualization cluster, or from any compatible hypervisor such as Red Hat Virtualization, OpenStack, or KVM. OpenShift Virtualization can import disk images that use the ISO, raw, or qcow2 format.

To import a disk image into OpenShift Virtualization, use the `virtctl image-upload` command:

```
[user@host ~]$ virtctl image-upload dv \
    fedora-rootdisk \ ❶
    --size 30Gi \ ❷
    --image-path fedora-vm-disk.img.gz \ ❸
    --storage-class ocs-external-storagecluster-ceph-rbd-virtualization ❹
PVC vms/fedora-import not found
DataVolume vms/fedora-import created
Waiting for PVC fedora-import upload pod to be ready...
Pod now ready
Uploading data to https://cdi-uploadproxy-openshift-cnv.apps.ocp4.example.com

776.56 MiB / 776.56 MiB [=====] 100.00% 6s

Uploading data completed successfully, waiting for processing to complete, you can
hit ctrl-c without interrupting the progress
Processing completed successfully
Uploading fedora-vm-disk.img.gz completed successfully
```

- ❶ Name of the data volume to create
- ❷ Size of the data volume, which must be large enough for the image to import
- ❸ Path of the disk image on your local machine to import
- ❹ Storage class for the data volume

OpenShift Virtualization and the `virtctl` command-line tool can import VM definitions only from another OpenShift Virtualization cluster. To import a VM from a different hypervisor, you must use the migration toolkit for virtualization.



## References

For more information about exporting virtual machines, refer to the *Exporting Virtual Machines* section in the *Virtual Machines* chapter in the Red Hat OpenShift Container Platform 4.16 *Virtualization* documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-exporting-vms](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-exporting-vms)

For more information about Kubernetes Volume Snapshots, refer to the *Volume Snapshots* section in the *Kubernetes* documentation at <https://kubernetes.io/docs/concepts/storage/volume-snapshots>

For more information, refer to the *YQ* documentation at <https://mikefarah.gitbook.io/yq/v/v3.x/>

## ► Guided Exercise

# Export and Import Virtual Machine Data and Settings

Export virtual machine data and settings from one project and import them into another project.

### Outcomes

- Export virtual machine data and settings from one project and import them into another project.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command also creates the `backup-export` namespace and starts the `mariadb-server` VM, which hosts a MariaDB database in that namespace.

```
[student@workstation ~]$ lab start backup-export
```

### Instructions

- As the `admin` user, go to the Red Hat OpenShift web console and confirm that the `mariadb-server` VM is running.
  - From the command line, log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- Set the `backup-export` project as the active project.

```
[student@workstation ~]$ oc project backup-export
Now using project "backup-export" on server ...
```

- Verify that the `mariadb-server` VM is running.

```
[student@workstation ~]$ oc get vm -n backup-export
NAME                  AGE     STATUS    READY
mariadb-server        30h     Running   True
```

- Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console  
https://console-openshift-console.apps.ocp4.example.com
```

15. Open a web browser and go to the web console URL.  
<https://console-openshift-console.apps.ocp4.example.com>  
Click **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.
  16. Select the **Administrator** perspective. Go to **Virtualization > VirtualMachines** and then select the **backup-export** project.
- ▶ 2. Log in to the **mariadb-server** VM and verify the database content.
- 2.1. From the web console, go to **Virtualization > VirtualMachines**, select the **mariadb-server** VM, and then select the **Console** tab.
  - 2.2. Click **Serial console**, press **Enter**, and then log in as the **developer** user with **developer** as the password.

```
Successfully connected to mariadb-server console. The escape sequence is ^]  
  
Red Hat Enterprise Linux 9.3 (Plow)  
Kernel 5.14.0-362.24.1.el9_3.x86_64 on an x86_64  
  
Activate the web console with: systemctl enable --now cockpit.socket  
  
mariadb-server login: developer  
Password: developer  
...output omitted...  
[developer@mariadb-server ~]$
```

- 2.3. Log in to the **sakila** database that is running on the **mariadb-server** VM. Use the DNS service name to access the database and log in as the **devuser** user with **developer** as the password.

```
[developer@mariadb-server ~]$ mysql -u devuser -p'developer' sakila  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
...output omitted...  
  
MariaDB [(sakila)]>
```

- 2.4. List the tables of the **sakila** database.

```
MariaDB [sakila]> SHOW TABLES;  
+-----+  
| Tables_in_sakila |  
+-----+  
| actor          |  
...output omitted...
```

- ▶ 3. Modify the **mariadb-server** database.

- 3.1. Add the books table to the sakila database.

```
MariaDB [sakila]> CREATE TABLE books
  ( id_book INT AUTO_INCREMENT PRIMARY KEY, title VARCHAR(50), year DATE );
Query OK, 0 rows affected (0.01 sec)
```

**Note**

You can copy the command to create the table from the ~/D0316/labs/backup-export/commands.sql file.

- 3.2. Insert a row into the books table.

```
MariaDB [sakila]> INSERT INTO books (title, year) VALUES
  ('The Great Gatsby', '1925-04-10');
Query OK, 1 row affected (0.01 sec)
```

**Note**

You can copy the command to create the table from the ~/D0316/labs/backup-export/commands.sql file.

- 3.3. Verify that the books table was created and that the row was inserted.

```
MariaDB [sakila]> SELECT * FROM books;
+-----+-----+-----+
| id_book | title          | year      |
+-----+-----+-----+
|       1 | The Great Gatsby | 1925-04-10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- 4. Shut down the mariadb-server VM and download the data volume.

- 4.1. Go to the terminal and use the `virtctl` command to stop the mariadb-server.

```
[student@workstation ~]$ virtctl stop mariadb-server -n backup-export
VirtualMachine "mariadb-server" scheduled to stop
```

**Note**

To stop the mariadb-server VM, you can alternatively use the OpenShift web console. Go to **Virtualization > VirtualMachines** and select the **mariadb-server** VM. Click **Actions > Stop**. Click **Stop** to confirm the operation, if needed. Wait for the machine to stop.

- 4.2. Verify that the mariadb-server VM is shut down.

```
[student@workstation ~]$ oc get vm -n backup-export
NAME          AGE   STATUS    READY
mariadb-server 30h   Stopped  False
```

- 4.3. Use the `virtctl vmexport` command to download the `mariadb-server` VM data volume.

```
[student@workstation ~]$ virtctl vmexport download mariadb-server \
--vm=mariadb-server --output=mariadb-server-pvc.img
VirtualMachineExport 'backup-export/mariadb-server' created successfully
"waiting for VM Export mariadb-server status to be ready"...
...output omitted...
Download finished successfully
VirtualMachineExport 'backup-export/mariadb-server' deleted successfully
```

- 5. Create the `backup-vm` namespace and import the `mariadb-server` VM data volume.

- 5.1. Use the `oc new-project` command to create the `backup-vm` project.

```
[student@workstation ~]$ oc new-project backup-vm
Now using project "backup-vm" on server "https://api.ocp4.example.com:6443"
```

- 5.2. Use the `virtctl image-upload` command to import the `mariadb-server` VM data volume into the `backup-vm` project.

```
[student@workstation ~]$ virtctl image-upload dv mariadb-server --size=10Gi \
--image-path=mariadb-server-pvc.img
PVC backup-vm/mariadb-server not found
DataVolume backup-vm/mariadb-server created
Waiting for PVC mariadb-server upload pod to be ready...
Pod now ready
...output omitted...
Uploading mariadb-server-pvc.img completed successfully
```

- 6. Create a VM from the `mariadb-server` VM data volume in the `backup-vm` project.

- 6.1. Use the `virtctl create` command to create a VM named `mariadb-server-backup` from the `mariadb-server` VM data volume.

```
[student@workstation ~]$ virtctl create vm --name=mariadb-server-backup \
--volume-pvc=src:mariadb-server | oc apply -f -
virtualmachine.kubevirt.io/mariadb-server-backup created
...output omitted...
```

- 6.2. Verify that the `mariadb-server-backup` VM is running.

```
[student@workstation ~]$ oc get vms
NAME          AGE   PHASE     IP      ...
mariadb-server-backup  1m   Running   10.9.0.65 ...
```

► 7. Log in to the `mariadb-server-backup` VM and verify the database content.

- 7.1. From the command line, connect to the console of the `mariadb-server-backup` VM by using the `virtctl console` command.

```
[student@workstation ~]$ virtctl console mariadb-server-backup
Successfully connected to mariadb-server console. The escape sequence is ^]

Red Hat Enterprise Linux 9.3 (Plow)
Kernel 5.14.0-362.24.1.el9_3.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket
...output omitted...
```

- 7.2. Log in as the `developer` user with `developer` as the password. You can ignore the `systemd-journald` logging messages.

```
mariadb-server login: developer
Password: developer
...output omitted...
[developer@mariadb-server ~]$
```

- 7.3. Connect to the `sakila` database that is running on the `mariadb-server-backup` VM. Use the DNS service name to access the database and log in as the `devuser` user with `developer` as the password.

```
[developer@mariadb-server ~]$ mysql -u devuser -p'developer' sakila
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(sakila)]>
```

- 7.4. Verify that the `books` table that you created is still present in the database.

```
MariaDB [sakila]> SELECT * FROM books;
+-----+-----+-----+
| id_book | title          | year      |
+-----+-----+-----+
|       1 | The Great Gatsby | 1925-04-10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- 7.5. Exit the MySQL prompt.

```
MariaDB [sakila]> quit
```

- 7.6. Log out of the console session, and then press `Ctrl+]` to exit.

```
[developer@mariadb-server ~]$ exit
```

- 7.7. Delete the backup-vm project.

```
[student@workstation ~]$ oc delete project backup-vm  
project.project.openshift.io "backup-vm" deleted
```

- 7.8. Delete the mariadb-server-pvc.img file from the workstation machine.

```
[student@workstation ~]$ rm mariadb-server-pvc.img
```

## Finish

As the student user on the workstation machine, confirm that you are in the /home/student directory and then use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-export
```

# Managing Virtual Machine Snapshots

## Objectives

- Manage virtual machine snapshots and explain appropriate use cases and recommended practices for using them.

## Take Snapshots of Virtual Machines

A *snapshot* is a view of a virtual machine that includes the operating system and applications on all the attached disks at a given point in time. A developer might take a snapshot of a VM before changing it. The snapshot can protect against errors that might have unintended consequences. In the event of a problem, the developer can revert to the state of the VM that is recorded in the snapshot.

With Red Hat OpenShift Virtualization, you can take live snapshots of running VMs, as well as snapshots of stopped VMs.

When the VM is running, OpenShift Virtualization communicates with the Quick Emulator (QEMU) guest agent that is running inside the VM to quiesce the file systems before taking the snapshot. This mechanism ensures that the file systems are in a consistent state. If you do not install the QEMU guest agent, then you can still take live snapshots, but the file systems might not be consistent. The default Red Hat templates come with the QEMU guest agent.

Disk snapshots rely on the back-end storage. For example, Red Hat OpenShift Data Foundation (ODF) provides a storage solution that supports snapshots. Other storage technologies might not provide that feature. The Container Storage Interface (CSI) driver that Kubernetes uses to pilot the back-end storage supports the snapshot feature. The references section provides a link to the list of drivers and their snapshot support.

If one VM disk relies on back-end storage that does not support snapshots, then OpenShift Virtualization displays a warning message when you take a snapshot. When you revert the snapshot, OpenShift Virtualization does not restore the data for that specific disk, which stays unchanged after reverting.

Red Hat supports taking a live snapshot of a VM with hot-plugged disks. However, the snapshot includes only the hot-plugged disks in the VM specification. *Hot-plugged* refers to disks that you attach to a running VM.

A VM snapshot also records the VM configuration, such as the number of CPUs or the amount of memory. If you change the VM configuration after taking a snapshot, then OpenShift Virtualization restores the original configuration when you revert the snapshot.

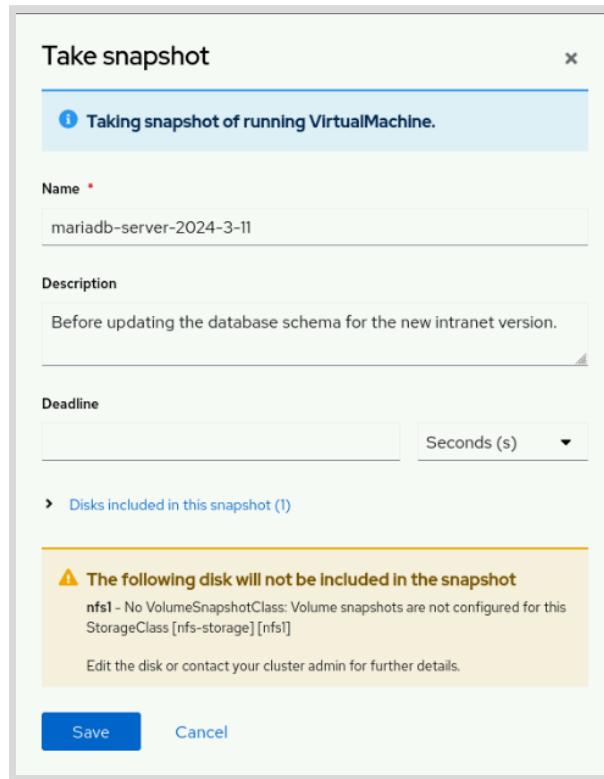
## Create Snapshots

To create a snapshot by using the OpenShift web console, go to **Virtualization > VirtualMachines** and then select the VM to snapshot.

If you did not install the QEMU guest agent in the VM, then Red Hat recommends stopping the VM before taking a snapshot. Click **Actions > Stop** and wait for the VM to stop.

Go to the **Snapshots** tab and click **Take snapshot**. Complete the form and click **Save**. Provide a meaningful name and description to help with snapshot identification.

A warning message is displayed if the VM includes disks that do not support snapshots, as shown in the following screen capture. The `nfs1` disk uses the `nfs-storage` storage class, which does not support snapshots.



**Figure 6.1: Taking a snapshot of a VM with disks that do not support snapshots**

If you stopped the VM before taking the snapshot, then restart it.

To create a snapshot by using the command line, create a `VirtualMachineSnapshot` resource in a file. You trigger the snapshot from the file by running the `oc apply -f resourcefile.yaml` command. The following resource file prepares a snapshot named `mariadb-server-2024-10-23` for the `mariadb-server` VM:

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineSnapshot
metadata:
  name: mariadb-server-2024-10-23
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: mariadb-server
```

## Restore Snapshots

Before you can restore a snapshot, you must stop the VM.

## Chapter 6 | Create and Restore Backups of Virtual Machines

After you stop the machine, go to the **Snapshots** tab, and locate the snapshot to restore. You can click a snapshot name to access the description that you provided at creation time.

To start the restore process from the OpenShift web console, go to the **Snapshots** tab, and then locate the snapshot to restore. Click the vertical ellipsis icon ⋮ at the right of the snapshot, click **Restore VirtualMachine from snapshot**, and click **Restore**. You can restart the VM after the restore completes.

From the command line, use the `oc get virtualmachinesnapshot` command to list the available snapshots in your project:

```
[user@host ~]$ oc get virtualmachinesnapshot
NAME                  SOURCEKIND      SOURCENAME      PHASE      READYTOUSE ...
db-before-schema       VirtualMachine  mariadb-server  Succeeded  true      ...
mariadb-server-2024-10-23  VirtualMachine  mariadb-server  Succeeded  true      ...
```

To restore a snapshot, create a `VirtualMachineRestore` resource in a file, and then use the `oc apply -f resourcefile.yaml` command to start the process. The following resource file restores the `mariadb-server-2024-10-23` snapshot for the `mariadb-server` VM:

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineRestore
metadata:
  name: restore-db
spec:
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: mariadb-server
  virtualMachineSnapshotName: mariadb-server-2024-10-23
```

## Delete Snapshots

Because snapshots consume disk space on back-end storage, delete the snapshots that you are no longer using.

To delete a snapshot, go to the **Snapshots** tab, click the vertical ellipsis icon at the right of the snapshot to delete, click **Delete**, and click **Delete snapshot**. You can delete snapshots without needing to stop the VM.

From the command line, use the `oc delete` command. The following example deletes the `mariadb-server-2024-10-23` snapshot:

```
[user@host ~]$ oc delete virtualmachinesnapshot mariadb-server-2024-10-23
virtualmachinesnapshot.snapshot.kubevirt.io "mariadb-server-2024-10-23" deleted
```

## Update the Virtual Machine Configuration

A VM snapshot includes the VM configuration. If you change the VM disk configuration after taking a snapshot, then review the following considerations before restoring the snapshot:

- If you removed a disk, then OpenShift Virtualization re-creates the disk. If the initial disk does not support snapshots, then OpenShift Virtualization attaches a new blank disk with the same

characteristics as the original disk. With this type of disk, OpenShift Virtualization does not restore the data from the time that you took the snapshot. That data is lost.

- If you add a disk, then OpenShift Virtualization removes the disk when you restore the snapshot.

## Copy Kubernetes Volume Snapshots

To restore a complete VM from a snapshot, you can duplicate individual VM disks from their snapshots.

VM snapshots rely on Kubernetes volume snapshots. When you take a snapshot of a VM, OpenShift Virtualization creates a Kubernetes volume snapshot for each VM disk. For example, if the VM has four disks that use a storage class that supports snapshots, then OpenShift Virtualization creates four Kubernetes volume snapshots, one for each disk.

To list the volume snapshots, use the OpenShift web console, and go to **Storage > VolumeSnapshots**. You can also run the `oc get volumesnapshots` command:

```
[user@host ~]$ oc get volumesnapshots
NAME                               SOURCEPVC
vmsnapshot-...-volume-dbdata       ...
vmsnapshot-...-volume-dbredo       ...
vmsnapshot-...-volume-dbundo       ...
vmsnapshot-...-volume-mariadb-server ... mariadb-server
```

The preceding output lists the Kubernetes volume snapshots and their associated Persistent Volume Claim (PVC). With this information, you can identify the snapshots that correspond to a VM disk.

## Duplicate an Individual Disk and Attach It to a VM

You can restore a Kubernetes volume snapshot as a new PVC. You can then attach the new PVC as a VM disk and access its content.

Duplicating a VM disk snapshot and attaching it to a VM is useful for inspecting the disk's content or for using the same initial data with several VMs.

To restore a Kubernetes volume snapshot and attach it to a VM, use the following steps:

- Go to **Storage > VolumeSnapshots**.
- Identify the Kubernetes volume snapshot to restore from the associated source PVC, which corresponds to the VM disk to copy.
- Click the vertical ellipsis icon at the right of the snapshot, and click **Restore as new PVC**. Complete the form and then click **Restore**.

**Restore as new PVC**

When restore action for snapshot **vmsnapshot-090ac090-b32b-4628-a3c3-34c429a6037f-volume-dbdata** is finished a new crash-consistent PVC copy will be created.

**Name \***  
vmsnapshot-090ac090-b32b-4628-a3c3-34c429a6037f-volume-dbdata-resto...

**StorageClass \***  
ocs-external-storagecluster-ceph-rbd

**Access mode \***  
 Single user (RWO)  Shared access (RWX)  Read only (RO)

**Volume mode \***  
Block

**Size \***  
2 GiB

**VolumeSnapshot details**

<b>Created at</b> Mar 11, 2024, 7:39 AM	<b>Namespace</b> advanced-snapshot
<b>Status</b> Ready	<b>API version</b> snapshot.storage.k8s.io/v1
<b>Size</b> 2 GiB	

**Cancel** **Restore**

Figure 6.2: Restoring a volume snapshot as a PVC

- Go to Virtualization > VirtualMachines, select the VM to attach the PVC to, and click Actions > Stop.
- Go to the Configuration tab and click Add disk. Select Use an existing PVC for the Source field, and set the PersistentVolumeClaim field to the name of the restored PVC. Click Save to attach the disk.
- Click Actions > Start.

**Create VirtualMachine from snapshot**

**Name \***  
rhel9-false6uyey2

**Configuration**

Operating system Guest agent is required
InstanceType VMQ, ulsmall
NICs None
Disks

**Create** **Cancel**

Figure 6.3: Creating a VM from a snapshot

- Go to Virtualization > VirtualMachines and click Create > With YAML. Complete the form or use the console YAML editor and click Create.



## References

For more information about installing the QEMU guest agent, refer to the *Installing the QEMU Guest Agent* section in the Red Hat OpenShift Container Platform 4.17 documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.17/html-single/virtualization/index#virt-installing-qemu-guest-agent](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.17/html-single/virtualization/index#virt-installing-qemu-guest-agent)

For more information about managing virtual machine snapshots, refer to the *Backup and Restore by Using VM Snapshots* section in the Red Hat OpenShift Container Platform 4.17 documentation at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.17/html-single/virtualization/index#backup-and-restore](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.17/html-single/virtualization/index#backup-and-restore)

For more information about volume snapshots, refer to the *CSI Volume Snapshots* section in the *Red Hat OpenShift Container Platform Storage* guide at

[https://docs.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.17/html-single/storage/index#persistent-storage-csi-snapshots](https://docs.redhat.com/documentation/en-us/openshift_container_platform/4.17/html-single/storage/index#persistent-storage-csi-snapshots)

For more information, refer to the *Live Snapshots in OpenShift Virtualization* article at

<https://cloud.redhat.com/blog/live-snapshots-in-openshift-virtualization>

For more information, refer to the *CSI Drivers and Their Snapshot Support Status* article at

<https://kubernetes-csi.github.io/docs/drivers.html>

## ► Guided Exercise

# Manage Virtual Machine Snapshots

Create a virtual machine snapshot, and use it to restore the virtual machine to the state in the snapshot and to create another virtual machine.

## Outcomes

- Take a VM snapshot.
- Revert a snapshot.
- Use a volume snapshot as a root file system for a new VM.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command also creates the `backup-snapshot` namespace and starts the `vm1` machine in that namespace.

```
[student@workstation ~]$ lab start backup-snapshot
```

## Instructions

- 1. As the `admin` user, locate and go to the Red Hat OpenShift web console. Confirm that the VM is running.
- 1.1. Open a terminal window and log in to the OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Identify the URL for the Red Hat OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to the web console URL.  
<https://console-openshift-console.apps.ocp4.example.com>  
Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- 1.4. Go to `Virtualization > VirtualMachines` and select the `backup-snapshot` project. Confirm that the `vm1` machine is running.

- ▶ 2. Confirm that the `vm1` machine runs the Quick Emulator (QEMU) guest agent, review the machine configuration, and then take a live snapshot of the VM.
- 2.1. Select the `vm1` machine. On the **Diagnostics** tab, in the **Status & Conditions** section, notice that the status of the **AgentConnected** type is **True**.  
That status means that the agent is installed and running inside the VM.
  - 2.2. Go to the **Overview** tab. In the **Details** section, locate the **CPU|Memory** subsection.  
Notice that the machine is assigned 1 CPU and 2 GiB of memory.
  - 2.3. Go to the **Configuration** tab and select **Storage**. In the **Disks** section, the VM has two disks. The `cloudinitdisk` volume is a special read-only disk that the VM uses to configure the system initially. The `vm1` disk contains the root file system for the VM.
  - 2.4. Because the QEMU guest agent is running, you can safely take a live snapshot of the VM.  
To take the snapshot, go to the **Snapshots** tab, and click **Take snapshot**. Enter `rhel-vm1-snap1` in the **Name** field. Notice the warning message about the `cloudinitdisk` volume, which Red Hat OpenShift Virtualization cannot include in the snapshot.  
Click **Save**. Wait for the snapshot to report the status as **Operation Complete**.

 **The following disk will not be included in the snapshot**

`cloudinitdisk` - Snapshot is not supported for this volumeSource type  
[`cloudinitdisk`]

Edit the disk or contact your cluster admin for further details.

[Learn more about snapshots](#)

- ▶ 3. Configure the `vm1` machine with 2 CPUs and 4 GiB of memory.

- 3.1. Click **Actions** > **Stop**. Wait for the machine to stop.



**Note**

The OpenShift web console displays a message that the VM is stopping and provides a **Force stop** button for an unresponsive VM.

- 3.2. Go to the **Configuration** tab and select **Details**. In the **VirtualMachine details** section, locate the **CPU|Memory** subsection and click **1CPU | 2 GiB Memory**.
- 3.3. Set the **CPU cores** field to 2, and set the **Memory** field to 4 GiB. Click **Save**.



**Note**

You might not see the saved changes. When you start the VM, the **CPU|Memory** displays the updated values.

- 3.4. Click **Actions > Start**. Wait for the machine to start.
- 3.5. Go to the **Overview** tab. In the **Details** section, locate the **CPU|Memory** subsection. Notice that the machine is assigned 2 CPUs and 4 GiB of memory.
- 4. Use the VM console to modify the `/etc/motd` file, to create the `/root/test.txt` file, and to delete the `/etc/httpd/conf/magic` file.
- These file system updates verify that OpenShift Virtualization restores the initial state of the VM when you revert the snapshot in a later step.
- 4.1. Go to the **Console** tab and log in as the **developer** user with **developer** as the password.
  - 4.2. Verify that the `/etc/motd` file is empty.
- ```
[developer@vm1 ~]$ cat /etc/motd
[developer@vm1 ~]$
```
- 4.3. Use the `sudo` command to become the **root** user.
- ```
[developer@vm1 ~]$ sudo -i
[root@vm1 ~]#
```
- 4.4. Add some content to the `/etc/motd` file.
- ```
[root@vm1 ~]# echo OpenShift Virtualization VM >> /etc/motd
```
- ```
[root@vm1 ~]# cat /etc/motd
OpenShift Virtualization VM
```
- 4.5. Create the `/root/test.txt` file.
- ```
[root@vm1 ~]# echo Hello World > /root/test.txt
```
- 4.6. Delete the `/etc/httpd/conf/magic` file.
- ```
[root@vm1 ~]# rm -f /etc/httpd/conf/magic
```
- 4.7. Log out of the VM console.
- ```
[root@vm1 ~]# logout
[developer@vm1 ~]$ logout
```
- 5. Restore the VM snapshot and then confirm that OpenShift Virtualization reverts the VM to its initial state.
- 5.1. Click **Actions > Stop**. Wait for the machine to stop.
  - 5.2. Go to the **Snapshots** tab and then locate the `rhel-vm1-snap1` snapshot. Click the vertical ellipsis icon : at the right of the `rhel-vm1-snap1` snapshot and click **Restore VirtualMachine snapshot**. Click **Restore** to confirm the operation.

- 5.3. Click **Actions > Start**. Wait for the machine to start.
- 5.4. Go to the **Overview** tab. In the **Details** section, locate the **CPU|Memory** subsection. Notice that the machine is assigned 1 CPU and 2 GiB of memory.
- 5.5. Go to the **Console** tab and log in as the **developer** user with **developer** as the password.
- 5.6. Review the content of the **/etc/motd** file. The file is empty and no longer contains your content. The state of the file is from when you took the snapshot.

```
[developer@vm1 ~]$ cat /etc/motd  
[developer@vm1 ~]$
```

- 5.7. Use the **sudo** command to become the **root** user, and list the content of the **/root** directory. The **test.txt** file no longer exists.

```
[developer@vm1 ~]$ sudo -i  
[root@vm1 ~]#
```

```
[root@vm1 ~]# ls /root  
[root@vm1 ~]#
```

- 5.8. List the contents of the **/etc/httpd/conf** directory. The **magic** file is restored.

```
[root@vm1 ~]# ls /etc/httpd/conf  
httpd.conf  magic
```

- 5.9. Log out of the VM console.

```
[root@vm1 ~]# logout  
[developer@vm1 ~]$ logout
```

▶ **6.** Create a virtual machine from the **rhel-vm1-snap1** snapshot.

- 6.1. Go to **Virtualization > VirtualMachines** and select the **vm1** machine.
- 6.2. Click **Actions > Stop**. Wait for the machine to stop.
- 6.3. Go to the **Snapshots** tab and locate the **rhel-vm1-snap1** snapshot. Click the vertical ellipsis icon at the right of the **rhel-vm1-snap1** snapshot, and click **Create VirtualMachine**.
- 6.4. Name the virtual machine **vm2**, select **Start VirtualMachine once created**, and click **Create**.

▶ **7.** Verify that the **vm2** machine is using a copy of the **vm1** disk as a root disk.

- 7.1. Go to **Virtualization > VirtualMachines** and confirm that the **vm2** machine is running.
- 7.2. Select the **vm2** machine and go to the **Details** section of the **Overview** tab.
- 7.3. From the **Details** section, the QEMU guest agent reports that the **Hostname** value for the **vm2** machine is **vm1**. The machine has the same system configuration as the

vm1 machine, because it uses an exact copy of the vm1 machine root disk from the rhel-vm1-snap1 snapshot.

- 7.4. Find the name of the PersistentVolumeClaim (PVC) that backs the vm2 machine. Go to **Configuration** and select **Storage**.

Notice that the name of the root disk is vm1 and that source for the disk is the restore...vm1 PVC. The PVC name is restore...vm1 because you created the vm2 machine from the restored snapshot of the vm1 machine.

- 8. Take a snapshot of the vm2 machine. Use a volume snapshot to copy the vm2 disk and then use that copy as a root disk for a new VM.



### Note

The following instructions demonstrate how to use a snapshot to make a copy of a volume.

Instead of using that copy to create a VM, another option is to use the clone feature in OpenShift Virtualization to duplicate a VM. VM cloning is explained in more detail elsewhere in the course.

- 8.1. Go to **Virtualization > VirtualMachines** and select the vm2 machine.
- 8.2. Go to the **Snapshots** tab, and click **Take snapshot**. Enter rhel-vm2-snap1 in the **Name** field. Notice the warning message about the cloudfinitdisk volume, which Red Hat OpenShift Virtualization cannot include in the snapshot.  
Click **Save**. Wait for the snapshot to report the status as **Operation Complete**.
- 8.3. Go to **Storage > VolumeSnapshots**. Recall that source of the vm2 machine root disk is the restore...vm1 PVC. Click the vertical ellipsis icon at the right of the snapshot where the source is the restore...vm1 PVC, and click **Restore as new PVC**.

Complete the form by using the following information and click **Restore** when done.

| Field        | Value                                               |
|--------------|-----------------------------------------------------|
| Name         | root-copy                                           |
| StorageClass | ocs-external-storagecluster-ceph-rbd-virtualization |
| Access mode  | Shared access (RWX)                                 |
| Volume mode  | Block                                               |

- 8.4. The lab command prepared the ~/D0316/labs/backup-snapshot/vm3.yaml file, which declares a **VirtualMachine** resource. From the terminal on the **workstation** machine, edit the file to specify the PVC to use as the root disk:

```
...output omitted...
volumes:
  - name: rootdisk
    persistentVolumeClaim:
      claimName: root-copy
```

- 8.5. Use the `oc create` command to create the resource.

```
[student@workstation ~]$ oc create -n backup-snapshot -f \
~/D0316/labs/backup-snapshot/vm3.yaml
virtualmachine.kubevirt.io/vm3 created
```

If using the OpenShift web console, go to **Virtualization > VirtualMachines** and click **Create > With YAML**. Copy the `~/D0316/labs/backup-snapshot/vm3.yaml` file contents and then press **Shift+Insert** to paste the contents of the file into the editor. Set the `claimName` parameter to `root-copy` and click **Create**.

- ▶ **9.** Verify that the `vm3` machine is using the copy of the `vm2` disk as a root disk, which is an exact copy of the `vm1` root disk.
- 9.1. Go to **Virtualization > VirtualMachines** and select the `vm3` machine. Go to the **Console** tab and log in as the **developer** user with **developer** as the password.
  - 9.2. Run the `hostname` command and notice that the machine name is `vm1`. Because you created the `vm3` machine from a snapshot of the `vm2` machine, and the source PVC for the `vm2` root disk is a snapshot of the `vm1` machine, the `vm3` machine has the same system configuration as the `vm1` machine

```
[developer@vm1 ~]$ hostname
vm1
```

- 9.3. Log out of the VM console.

```
[developer@vm1 ~]$ logout
```

- ▶ **10.** Close the web browser.

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-snapshot
```

# OADP Operator Deployment and Features

---

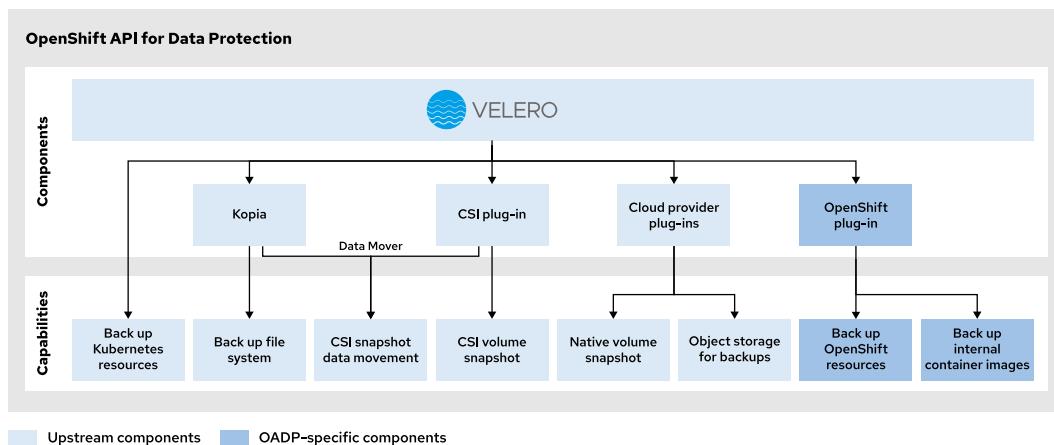
## Objectives

- Deploy the OpenShift API for Data Protection (OADP) operator to create backup and restore APIs in the OpenShift cluster, and describe the features and dependencies of OADP.

## OpenShift API for Data Protection

The OADP product provides a native backup solution for applications that run on OpenShift.

OADP uses several components such as Velero and Kopia to back up all the Kubernetes resources, container images from the internal registry, and persistent volumes that are associated with an application.



## OADP 1.4.1 with OpenShift Virtualization

The OADP 1.4.1 release supports backing up and restoring VM workloads that the OpenShift Virtualization operator manages. You can back up and restore VMs by using the same OADP resources as for backing up and restoring Kubernetes resources.

The following list describes the main components that OADP uses:

### Velero

Velero is the main upstream component of OADP. It provides the backup API to the users and uses plug-ins to add extended capabilities to OADP.

### Data Mover

This feature enables exporting snapshot content to object storage by using Kopia and the CSI plug-in.

### Kopia

Kopia is an open source backup tool that Velero and Data Mover use to back up persistent volumes.

See the references section for more information about those components.

**Note**

Red Hat supports using OpenShift Virtualization 4.14 or later with OADP 1.3.x or later. Earlier OADP versions than 1.3.0 are not supported for backing up and restoring OpenShift Virtualization.

## Kubernetes Resources Backup

OADP uses Velero to back up all the Kubernetes resources to cloud object storage. You can filter objects to back up and restore by namespace, resource type, or label.

## Data Volume Snapshot

OADP backs up volumes by using the cloud-native snapshot API or the Kubernetes Container Storage Interface (CSI) snapshot API.

Red Hat recommends the use of snapshots when possible, because this feature enables fast and consistent backup.

## OADP Data Mover

Some CSI drivers might store the snapshot in the same physical storage location as the original volume and might be vulnerable to data loss in the event of a disaster. The storage solution for Red Hat OpenShift might not guarantee snapshot durability. In such scenarios, you can enable the OADP Data Mover to copy the snapshot content to a remote backup storage location and remove the snapshot from the storage back end.

Data Mover uses Kopia to upload and encrypt snapshot content to a unified backup repository on the object storage.

## Velero Plug-ins

You can extend the OADP capabilities and backup logic by using the Velero plug-in system. Support for various cloud providers, storage solutions, and additional CRDs are available with integrated plug-ins.

OADP includes the following plug-ins that you can enable during the installation:

### aws

Stores and retrieves backups on AWS S3-compatible storage. Manages volume snapshots on AWS EBS volumes.

### gcp

Stores and retrieves backups on Google Cloud Storage. Manages volume snapshots on Google Compute Engine Disks.

### azure

Stores and retrieves backups on Azure Blob Storage. Manages volume snapshots on Azure Managed Disks.

### openshift

Provides additional logic to back up and restore OpenShift resources, including container images from the internal registry.

### kubevirt

Provides additional logic to back up and restore virtual machines and other OpenShift Virtualization resources. This plug-in is mandatory for OpenShift Virtualization.

**csi**

Provides volume snapshot support by using the Kubernetes CSI Snapshot API.

The CSI, kubevirt, and openshift plug-ins are mandatory for Data Mover. You must also specify the available storage plug-in, such as gcp.

## OADP API Resources

The OADP operator provides the following API resources:

**DataProtectionApplication**

The `dataProtectionApplication` resource is the configuration for the OADP operator and its components.

**BackupStorageLocation**

The OADP operator creates and manages a `backupStorageLocation` resource for each storage location that is defined in the `dataProtectionApplication` configuration.

**VolumeSnapshotLocation**

The OADP operator creates and manages a `volumeSnapshotLocation` resource for each snapshot location that is defined in the `dataProtectionApplication` configuration.

**Backup**

The `backup` resource requests the Velero server to create a backup.

**Restore**

The `restore` resource requests the Velero server to restore a backup.

**Schedule**

The `schedule` resource creates a backup on a given schedule by using the Cron format.

**BackupRepository**

The `BackupRepository` resource tracks the lifecycle of the backup repositories.

## Requirements for OADP

Red Hat OpenShift API for Data Protection requires the following resources.

## OpenShift User Permissions

OADP configuration requires a user with the `cluster-admin` role. Backup and restore operations require administrative privileges in the OADP namespace (`openshift-adp` by default).

**Important**

OADP deployment has administrative privileges on the cluster to back up and restore all resources in any namespaces.

## Object Storage

OADP requires an object storage location to store the backups. The following list is an example of object storage providers for storing backups:

- AWS S3-compatible object storage, such as MinIO or OpenShift Data Foundation
- Amazon Web Services
- Microsoft Azure

- Google Cloud Platform
- Multicloud Object Gateway
- IBM Cloud Object Storage S3

## Object Storage with OpenShift Data Foundation

OpenShift Data Foundation provides two object storage services: the MultiCloud Object Gateway (NooBaa MCG) and the Ceph RADOS Object Gateway (Ceph RGW). Both services are fully compatible with OADP by using the aws provider plug-in.



### Note

For more details about OpenShift Data Foundation, refer to the *DO370: Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation* training course.

Similar to a persistent volume claim, you can use an object bucket claim to request an S3-compatible bucket. OpenShift Data Foundation creates a secret and a configuration map, which are both named the same as the object bucket claim, with the credentials and information to access the bucket.

The following example shows a definition of an object bucket claim:

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: vm-bucket-claim
  namespace: my-namespace
spec:
  storageClassName: storage-class-name ①
  generateBucketName: vm-bucket ②
```

- ① Storage class name for the object bucket. To use NooBaa MCG, set the value to `openshift-storage.noobaa.io`. To use the Ceph RGW gateway, set the value to `ocs-external-storagecluster-ceph-rgw`.
- ② Prefix to add to the generated bucket name.

A few minutes after the object bucket claim is created, the status is set to Bound when the bucket is created and ready to use.

```
[student@workstation ~]$ oc get obc -n my-namespace
NAME           STORAGE-CLASS          PHASE   AGE
vm-bucket-claim  openshift-storage.noobaa.io  Bound  5m
```

OpenShift Data Foundation creates a configuration map with the bucket information in the same namespace as the object bucket claim:

```
[student@workstation ~]$ oc -n my-namespace get \
configmap/vm-bucket-claim -o yaml
apiVersion: v1
data:
  BUCKET_HOST: s3.openshift-storage.svc ①
  BUCKET_NAME: vm-bucket-noobaa-e1121900-ae51-41aa-b514-c1d026edad9d ②
```

```
BUCKET_PORT: "443" ③
BUCKET_REGION: "" ④
BUCKET_SUBREGION: ""
kind: ConfigMap
...output omitted...
```

- ①** S3 API URL endpoint.
- ②** The generated name of the bucket.
- ③** The TCP port for the S3 endpoint. Port 443 is for `https` protocol.
- ④** The region name, if the storage class uses it.

If the `BUCKET_HOST` uses an internal service URL that ends with `.svc`, then the URL can be reached only from a workload that is running inside the cluster. The `s3` route in the `openshift-storage` namespace provides the external endpoint URL.

```
[student@workstation ~]$ oc get route/s3 -n openshift-storage \
-o jsonpath='{.spec.host}{"\n"}'
s3-openshift-storage.apps.ocp4.example.com
```

The S3 bucket credentials are stored in a secret in the same namespace as the object bucket claim:

```
[student@workstation ~]$ oc -n my-namespace extract \
secret/vm-bucket-claim --to=-
# AWS_ACCESS_KEY_ID
YEAsbMJnG3o1bGANZprt
# AWS_SECRET_ACCESS_KEY
xjaeCDhskn7lfrdA7WqzoUxpjRYuyjc9uDaWlMw3
```

## CSI Snapshot

To use volume snapshots with CSI drivers, register the CSI driver for creating snapshots. The driver name must be the same as the `provisioner` attribute of the volume storage class to back up.

```
[student@workstation ~]$ oc get storageclasses -o \
custom-columns="NAME:.metadata.name,PROVISIONER:.provisioner"
NAME                               PROVISIONER
localvolume                        kubernetes.io/no-provisioner
nfs-storage                         k8s-sigs.io/nfs-subdir-external-
provisioner
ocs-storagecluster-ceph-rbd        openshift-storage.rbd.csi.ceph.com
ocs-storagecluster-ceph-rbd-virtualization openshift-storage.rbd.csi.ceph.com
ocs-storagecluster-ceph-rgw         openshift-storage.ceph.rook.io/bucket
ocs-storagecluster-ceph-rgfs        openshift-storage.cephfs.csi.ceph.com
openshift-storage.noobaa.io          openshift-storage.noobaa.io/obc

[student@workstation ~]$ oc get volumesnapshotclasses -o \
custom-columns="NAME:.metadata.name,DRIVER:.driver"
```

| NAME                                      | DRIVER                                       |
|-------------------------------------------|----------------------------------------------|
| ocs-storagecluster-cephfsplugin-snapclass | <b>openshift-storage.cephfs.csi.ceph.com</b> |
| ocs-storagecluster-rbdplugin-snapclass    | <b>openshift-storage.rbd.csi.ceph.com</b>    |

**Note**

For storage classes that do not support snapshots, such as `nfs-storage` in the previous example, you can use the file-system backup.

## Installing and Configuring OADP

You can install the OADP operator from the OperatorHub or by using the `oc` command. See the references section for instructions for either of these methods.

**Note**

For more details about Kubernetes operators, refer to the *DO280: Red Hat OpenShift Administration II: Configuring a Production Cluster* training course.

Data Protection Application (DPA) is a custom resource that defines the backup configuration. The DPA resource includes the backup location, the snapshot location, and the backup schedule. The DPA resource is namespace-scoped and can be created in any namespace.

The `dataProtectionApplication` resource file defines the configuration and components that the OADP operator manages. You must create a secret to use the backup location in the `dataProtectionApplication` resource. The following YAML file is an example of a `dataProtectionApplication` resource that stores backups in a OpenShift Data Foundation S3 bucket:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: oadp-backup
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins: ①
        - aws
        - openshift
        - csi
        - kubevirt
      defaultSnapshotMoveData: true ②
      nodeAgent: ③
        enable: true
        uploaderType: kopia ④
      backupLocations:
        - velero:
            config:
              profile: "default"
              region: "us-east-1"
              s3Url: "https://s3.openshift-storage.svc"
              s3ForcePathStyle: "true"
```

```

provider: aws
default: true
credential:
  key: cloud
  name: cloud-credentials
objectStorage:
  bucket: vm-bucket
  prefix: oadp

```

- ❶ The list of storage provider plug-ins to use.
- ❷ Set to true to enable moving CSI snapshots to remote object storage.
- ❸ Enable the node agent if you use Data Mover.
- ❹ For the built-in Data Mover, you must use Kopia.

## Upgrading OADP

Before upgrading OADP from 1.3 to 1.4, perform the following tasks:

- Update the supported OADP version.
- Run the hotfix to update the supported OADP version.
- Update the OADP operator.
- Update the OADP Data Mover.
- Update the OADP Velero plug-in.
- Update the OADP CSI plug-in.

After the upgrade, you can use the new features of OADP 1.4, such as the `defaultSnapshotMoveData` option, which moves the CSI snapshots to remote object storage.

## Backup Storage Location

The `backupLocation` section describes the object storage location for backups. You can define multiple backup locations. The following example uses the `aws` provider plug-in:

```

backupLocations:
  - velero:
      config: ❶
        profile: "default" ❷
        region: "us-east-1"
        s3Url: https://s3.openshift-storage.svc
        s3ForcePathStyle: "true"
        insecureSkipTLSVerify: "true"
      provider: aws ❸
      default: true ❹
      credential: ❺
        key: cloud
        name: cloud-credentials
      objectStorage:
        bucket: vm-bucket ❻
        prefix: oadp ❼
        caCert: LLS0S0...LS0tLS0K ❽

```

- ➊ Storage provider-specific configuration. In this example, the configuration is for the aws provider.
- ➋ The credentials profile name that is defined in the Velero secret.
- ➌ The storage provider plug-in name.
- ➍ Use this backup location as the default if none is specified in the backup and restore resources.
- ➎ Velero secret with the storage provider credentials.
- ➏ Bucket name from the storage provider.
- ➐ Optional subdirectory in the bucket to store backups. If not specified, then the backups are stored at the root level of the bucket.
- ➑ Optional CA bundle in Base64 format to verify TLS connections to the storage provider.

The Velero secret contains the credentials to access the object storage. If the secret name is not specified in the configuration, then Velero uses the `cloud-credentials` default secret name. You can create the Velero secret with a configuration file that is specific to the provider plug-in. The following Velero configuration file is for the aws provider plug-in:

```
[student@workstation ~]$ cat credentials-velero
[default] ①
aws_access_key_id=AWS_ACCESS_KEY_ID
aws_secret_access_key=AWS_SECRET_ACCESS_KEY
```

- ➊ The profile name for the credentials. The name must match the profile name in the configuration for the backup storage location.

You can then create the Velero secret with the following `oc create secret` command:

```
[student@workstation ~]$ oc create secret generic \
  cloud-credentials -n openshift-adp \
  --from-file cloud=credentials-velero
secret/cloud-credentials created
```



### Note

The Velero secret must be created before the `DataProtectionApplication` object is created, or the installation fails.

## CSI Snapshots

For OADP to use a volume snapshot class, you must add the `velero.io/csi-volumesnapshot-class: "true"` label. Only one volume snapshot class per driver needs this label.

You can use the following command to add the required label:

```
[student@workstation ~]$ oc label volumesnapshotclass vm-snapshot-storageclass \
velero.io/csi-volumesnapshot-class="true"
vm-snapshot-storageclass labeled
```

## Verification and Troubleshooting

When the configuration is complete, several OADP components are created depending on the enabled features.

```
[user@host ~]$ oc -n openshift-adp get deploy
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
openshift-adp-controller-manager   1/1     1           1          24h ①
velero                     1/1     1           1          25s ②
```

- ① OADP controller that is deployed on the OADP operator installation.
- ② Velero deployment that manages backup and restore operations.

If the Node Agent that manages volume backups is enabled, then the following daemon set is created:

```
[user@host ~]$ oc -n openshift-adp get daemonset
NAME      DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
node-agent 3       3       3       3       3       <none>    23s ①
```

- ① The node-agent is deployed on all compute nodes, to back up volume file systems.

The OADP operator creates and validates each backup storage location that is defined in the DataProtectionApplication object. OADP tries to connect to the object storage with the provided credentials and validates the configuration. If the configuration is correct, then the backup storage location enters the Available phase.

```
[user@host ~]$ oc -n openshift-adp get backupstoragelocation
NAME      PHASE      LAST VALIDATED   AGE      DEFAULT
oadp-backup-1  Available  7s            2m44s  true
```

If the backup storage location remains in the Unavailable phase, then you can use the `oc describe` command to get more information and error messages from the Status field:

```
[user@host ~]$ oc -n openshift-adp describe \
backupstoragelocation oadp-backup-1
...output omitted...
Status:
  Last Synced Time: 2024-09-22T10:27:27Z
  Last Validation Time: 2024-09-22T12:03:27Z
  Message: BackupStorageLocation "oadp-backup-1" is unavailable:
  rpc error: code = Unknown desc = InvalidAccessKeyId: The AWS access key Id you
  provided does not exist in our records.
  Phase: Unavailable
...output omitted...
```

In this example, the S3 credentials that are configured in the Velero secret are incorrect.

**Note**

The OADP operator checks the validity of the backup storage location configuration every minute. Review the `last_validated` field of the backup storage location to determine when the last check ran.

You can use the AWS CLI or the `s3cmd` tool to validate the S3 configuration and to browse the content of the S3 bucket.

To configure the `s3cmd` tool, create a `.s3cfg` file in your home directory with your S3 credentials. You can use the following configuration template:

```
access_key = AWS_ACCESS_KEY_ID ①
secret_key = AWS_SECRET_ACCESS_KEY
host_base = s3.mydomain.com ②
host_bucket = s3.mydomain.com/%(bucket)s ③
signature_v2 = True ④
```

- ① S3 access key and secret key.
- ② S3 endpoint URL, if the AWS S3 endpoint is not used.
- ③ S3 bucket address model, if your storage provider does not support the DNS-style model.
- ④ If your storage provider does not support the AWS v4 signature, then you can use the AWS v2 signature instead. OpenShift Data Foundation requires this setting.

To list the contents of a bucket and to validate the configuration, use the `s3cmd ls` command to list all available buckets, or use the `s3cmd la` command to list all objects in all buckets:

**Note**

If the S3 bucket is empty, then the `s3cmd la` command returns an empty line.

```
[user@host ~]$ s3cmd ls
2024-10-24 12:33  s3://my-bucket

[user@host ~]$ s3cmd la
2024-10-24 12:31          143  s3://my-bucket/somefile.txt
...output omitted...
```



## References

For more information about installing operators by using the OperatorHub, refer to the *Installing from OperatorHub Using the Web Console* section in the *Administrator Tasks* chapter in the Red Hat OpenShift Container Platform 4.16 Operators documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/operators/index#olm-installing-from-operatorhub-using-web-console\\_olm-adding-operators-to-a-cluster](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/operators/index#olm-installing-from-operatorhub-using-web-console_olm-adding-operators-to-a-cluster)

For more information about installing operators by using the command line, refer to the *Installing from OperatorHub Using the CLI* section in the *Administrator Tasks* chapter in the Red Hat OpenShift Container Platform 4.16 Operators documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/operators/index#olm-installing-operator-from-operatorhub-using-cli\\_olm-adding-operators-to-a-cluster](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/operators/index#olm-installing-operator-from-operatorhub-using-cli_olm-adding-operators-to-a-cluster)

For more information about configuring OADP with other cloud object storage, refer to the *Installing and Configuring OADP* section in the *Application Backup and Restore* chapter in the Red Hat OpenShift Container Platform 4.16 *Backup and Restore* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/backup\\_and\\_restore/index#installing-and-configuring-oadp](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/backup_and_restore/index#installing-and-configuring-oadp)

For more information about the OADP v1.4 Data Mover, refer to the *OADP Data Mover* section in the *Application Backup and Restore* chapter in the Red Hat OpenShift Container Platform 4.16 *Backup and Restore* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/backup\\_and\\_restore/index#about-oadp-data-mover](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/backup_and_restore/index#about-oadp-data-mover)

For more information, refer to the *Velero Back up and Migrate Kubernetes Resources* article at

<https://velero.io/>

For more information, refer to the *Kopia Back up Software* article at

<https://kopia.io/>

For more information, refer to the *S3cmd: Command Line S3 Client* article at

<https://s3tools.org/s3cmd>

## ► Guided Exercise

# Verify the OADP Operator Installation

Review the parameters of the OADP Operator and back up a virtual machine to validate that OADP is functional in the cluster.

## Outcomes

- Review the configuration of the OpenShift APIs for Data Protection operator.
- Back up an OpenShift project by using OADP to validate the configuration.

## Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

The lab command deploys the OADP operator, configures the S3 backup location, and creates the data protection application resource to configure the OADP operator.

```
[student@workstation ~]$ lab start backup-oadp
```

## Instructions

► 1. As the **admin** user, locate and go to the OpenShift web console.

- 1.1. Log in to your OpenShift cluster as the **admin** user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to the web console URL.

<https://console-openshift-console.apps.ocp4.example.com>

- 1.4. Click **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.

► 2. Review the object bucket claim in the backups project.

- 2.1. Click **Storage > Object Storage** and wait until the page loads.

- 2.2. Click Object Bucket Claims to view the object bucket claims. Ensure that the backup-obc object bucket claim is associated with the backups namespace and that its status set to Bound.

| Name           | Namespace  | Status | Secret       | StorageClass                   |
|----------------|------------|--------|--------------|--------------------------------|
| OBO backup-obc | NS backups | Bound  | S backup-obc | SC openshift-storage.noobaa.io |

- 2.3. Click the **backup-obc** object bucket claim name and scroll down to view the associated data. Click **Reveal Values** to display the object bucket access parameters. The values are stored in the configuration map and the secret resources that are associated with the object bucket claim.

### Configuration map

The resources in the configuration map contain parameters to connect to the S3 object bucket.

#### Endpoint

The endpoint of the S3 service. **This endpoint is relative to the cluster.**

#### Bucket Name

The S3 bucket name that is associated with this object bucket claim.

#### Secret

The parameters in the secret contain the access keys to interact with the S3 object bucket.

#### Access Key

The S3 access key that is associated with this object bucket claim.

#### Secret Key

The S3 secret key that is associated with this object bucket claim.

- 2.4. The object bucket claim data is stored in the associated configuration map and secret resources. List the object bucket claim and the related resources by using the command line.

```
[student@workstation ~]$ oc get objectbucketclaim backup-obc -n backups
NAME      STORAGE-CLASS          PHASE   AGE
backup-obc  openshift-storage.noobaa.io  Bound   ...

[student@workstation ~]$ oc get configmap backup-obc -n backups
NAME      DATA    AGE
backup-obc  5      ...
```

```
[student@workstation ~]$ oc get secret backup-objc -n backups
NAME      TYPE      DATA   AGE
backup-objc  Opaque    2      ...
```

- 3. Identify the external endpoint of the S3 service. The bucket host endpoint in the backup-objc configuration map is the name of the s3 service in the openshift-storage namespace.

You can access the S3 service from outside the cluster by using the endpoint of the s3 route in the openshift-storage namespace.

- 3.1. List the s3 service and route in the openshift-storage namespace.

```
[student@workstation ~]$ oc get service s3 -n openshift-storage
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)
s3        LoadBalancer  172.30.154.90  192.168.50.21  80:.../TCP,443:.../TCP, ...
[student@workstation ~]$ oc get route s3 -n openshift-storage
NAME      HOST/PORT          PATH      SERVICES      PORT      ...
s3        s3-openshift-storage.apps.ocp4.example.com      s3        s3-https ...
```

- 3.2. Save the S3 endpoint host to a variable for later use.

```
[student@workstation ~]$ BUCKET_HOST="s3-openshift-storage.apps.ocp4.example.com"
```

- 3.3. Visit the S3 endpoint URL in a web browser to confirm that is active.

- <https://s3-openshift-storage.apps.ocp4.example.com/>

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<Resource></Resource>
<RequestId>1qaz-2wsx-3edc-4rfv</RequestId>
</Error>
```



### Note

You can safely ignore the access denied error message.

- 4. Verify that you can access the S3 bucket.

- 4.1. Extract the backup-objc configuration map data to view the S3 bucket name where the backups are stored.

```
[student@workstation ~]$ oc extract configmap/backup-objc -n backups --to=-
# BUCKET_HOST
s3.openshift-storage.svc
# BUCKET_PORT
443
# BUCKET_REGION
```

```
# BUCKET_SUBREGION  
  
# BUCKET_NAME  
backup-obc-9b990ae2-1c91-4e8b-98cf-d378e0500108
```

- 4.2. Save the S3 bucket name to a variable for later use.

```
[student@workstation ~]$ BUCKET_NAME="backup-obc-9b990ae2-1c91-4e8b-98cf-  
d378e0500108"
```

- 4.3. Extract the backup-obc secret data to view the access keys to interact with the S3 bucket.

```
[student@workstation ~]$ oc extract secret/backup-obc -n backups --to=-  
# AWS_ACCESS_KEY_ID  
QhFEuiusalNBn180RVtB  
# AWS_SECRET_ACCESS_KEY  
IbtwoIBVlyLHFhgSY+Zxb6bXyDIMwhBSv0wasP
```

- 4.4. Verify that the `~/.s3cfg` file has the credentials and the endpoint of the S3 service.

```
[student@workstation ~]$ head -n 5 ~/.s3cfg  
[default]  
access_key = QhFEuiusalNBn180RVtB ①  
secret_key = IbtwoIBVlyLHFhgSY+Zxb6bXyDIMwhBSv0wasP ②  
host_base = s3-openshift-storage.apps.ocp4.example.com ③  
host_bucket = %(bucket).s3-openshift-storage.apps.ocp4.example.com
```

- ① The `access_key` setting matches the `AWS_ACCESS_KEY_ID` field in the `backup-obc` secret in the `backups` namespace.
- ② The `secret_key` setting matches the `AWS_SECRET_ACCESS_KEY` field in the `backup-obc` secret in the `backups` namespace.
- ③ The `host_base` setting matches the host of the `s3` route in the `openshift-storage` namespace.

- 4.5. List the available buckets in the S3 service.

```
[student@workstation ~]$ s3cmd ls s3://  
2024-10-13 17:37 s3://first.bucket  
2024-10-31 01:35 s3://backup-obc-9b990ae2-1c91-4e8b-98cf-d378e0500108 ①
```

- ① The S3 bucket name matches the `BUCKET_NAME` field in the `backup-obc` configuration map in the `backups` namespace.



### Note

You can safely ignore the `first.bucket` S3 bucket.

- 4.6. Extract the `cloud-credentials` secret in the `openshift-adp` namespace.

```
[student@workstation ~]$ oc extract secret/cloud-credentials \
-n openshift-adp --to=-
# cloud
[default]
aws_access_key_id=QhFEuiusalNBnl80RVtB ①
aws_secret_access_key=IbtwoIBVlyLHFhgiSY+Zxb6bXyDIMwhBSv0wasP ②
```

- ① The aws\_access\_key\_id setting matches the AWS\_ACCESS\_KEY\_ID field in the backup-obc secret in the backups namespace.
- ② The aws\_secret\_access\_key setting matches the AWS\_SECRET\_ACCESS\_KEY field in the backup-obc secret in the backups namespace.

► 5. Verify the worker node configuration.

- 5.1. Confirm that the worker nodes have the node-role.kubernetes.io/nodeAgent label by using a label selector.

```
[student@workstation ~]$ oc get nodes -l node-role.kubernetes.io/nodeAgent
NAME      STATUS    ROLES     AGE      VERSION
worker01   Ready     nodeAgent,worker   ...    v1.29.6+aba1e8d ①
worker02   Ready     nodeAgent,worker   ...    v1.29.6+aba1e8d ②
```

- ① ② The worker01 and worker02 nodes have the label and thus match the label selector.

► 6. Review the resources of the OADP operator.

- 6.1. Click Operators > Installed Operators to view the operators list. Scroll down and then click OADP Operator. Click All instances and verify the following conditions:
  - The status of the data protection application resource is set to Condition: Reconciled.
  - The status of the backup storage location resource is set to Phase: Available.
  - The volume snapshot location resource is present.

Name	Kind	Status	Labels
DPA oadp-dpa	DataProtectionApplication	Condition: Reconciled	No labels
BSL oadp-dpa-1	BackupStorageLocation	Phase: Available	app.kubernetes.io/component=bsl app.kubernetes.io/instance=oadp-dpa-1 app.kubernetes.io/manager=oadp-operator app.kubernetes.io/operator-name=oadp-operator-v... openshift.io/oadp=True openshift.io/oadp-registry=True
VSL oadp-dpa-1	VolumeSnapshotLocation	-	No labels

- 6.2. Confirm that the retention policy of the ocs-external-storagecluster-rbdplugin-snapclass volume snapshot class resource is set to Retain and has the velero.io/csi-volumesnapshot-class=true label.

```
[student@workstation ~]$ oc get volumesnapshotclass \
-l velero.io/csi-volumesnapshot-class=true --show-labels
NAME                                     DRIVER
ocs-external-storagecluster-rbdplugin-snapclass  openshift-storage.rbd.csi.ceph...
DELETIONPOLICY   AGE   LABELS
Retain          ...   velero.io/csi-volumesnapshot-class=true  ①
```

① The resource has the label and matches the selector.

► 7. List the resources in the `my-project` namespace.

7.1. Change to the `my-project` project.

```
[student@workstation ~]$ oc project my-project
Now using project "my-project" on server ...
```

7.2. List the resources related to the virtual machine and confirm the following conditions:

- The virtual machine status is set to `Running`.
- The virtual machine instance is present and is marked as ready.
- The `virt-launcher` pod is running and ready.

```
[student@workstation ~]$ oc get vm
NAME      AGE   STATUS   READY
centos-stream9  ...   Running  True

[student@workstation ~]$ oc get vmi
NAME      AGE   PHASE   IP           NODENAME   READY
centos-stream9  ...   Running  10.8.0.35  master02  True

[student@workstation ~]$ oc get pods
NAME      READY   STATUS   RESTARTS   AGE
virt-launcher-centos-stream9-p7gzm  1/1     Running  0          ...
```

7.3. List service and route resources. The service exposes the port 80 TCP of the VM, and the route makes the service accessible to clients outside the cluster.

```
[student@workstation ~]$ oc get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
centos-stream9  ClusterIP  172.30.224.86  <none>        80/TCP      ...

[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS      AGE
centos-stream9  10.8.0.35:80  ...

[student@workstation ~]$ oc get routes
NAME      HOST/PORT      PATH
centos-stream9  centos-stream9-my-project.apps.ocp4.example.com
  SERVICES      PORT      TERMINATION      WILDCARD
  centos-stream9  http      None
```

7.4. Save the route host name to a variable for later use.

```
[student@workstation ~]$ VM_ROUTE="centos-stream9-my-project.apps.ocp4.example.com"
```

- 7.5. Connect to the virtual machine to view the status of the HTTP service.

```
[student@workstation ~]$ virtctl ssh root@vm/centos-stream9
Warning: Permanently added 'vm/centos-stream9.my-project' ...
...output omitted...
[root@centos-stream9 ~]#
```

- 7.6. Confirm that the `httpd` process is listening on port 80 TCP.

```
[root@centos-stream9 ~]# systemctl status --no-pager httpd.service
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service enabled; preset: disabled)
  Active: active (running) since ...
    Docs: man:httpd.service(8)
 Main PID: 8202 (httpd)
   Status: "Total requests: 2; ...
 Tasks: 177 (limit: 10406)
  Memory: 36.3M
     CPU: 587ms
    CGroup: /system.slice/httpd.service
            ├─8202 /usr/sbin/httpd -DFOREGROUND
            ├─8204 /usr/sbin/httpd -DFOREGROUND
            ├─8205 /usr/sbin/httpd -DFOREGROUND
            ├─8206 /usr/sbin/httpd -DFOREGROUND
            └─8207 /usr/sbin/httpd -DFOREGROUND

... centos-stream9.local systemd[1]: Starting The Apache HTTP Server...
... centos-stream9.local systemd[1]: Started The Apache HTTP Server.
... centos-stream9.local httpd[8202]: Server configured, listening on: port 80
```

- 7.7. Exit the virtual machine SSH connection.

```
[root@centos-stream9 ~]# exit
logout
Connection to vm/centos-stream9.my-project closed.
```

- 7.8. Visit the URL that is associated with the route to confirm that the virtual machine responds to HTTP traffic on port 80 TCP.

`http://centos-stream9-my-project.apps.ocp4.example.com/`

```
[student@workstation ~]$ curl -fsSL http://${VM_ROUTE}/
It works!
```

► 8. Back up the my-project namespace.

- 8.1. Inspect the `~/D0316/labs/backup-oadp/backup.yaml` resource manifest and copy the content of this file.

```
[student@workstation ~]$ cat ~/DO316/labs/backup-oadp/backup.yaml
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: my-backup ①
  namespace: openshift-adp ②
  labels:
    velero.io/storage-location: oadp-dpa-1 ③
spec:
  includedNamespaces:
    - my-project ④
  storageLocation: oadp-dpa-1 ⑤
  snapshotVolumes: true ⑥
```

- ① The name of the backup resource.
- ② The namespace of the backup resource.
- ③ The backup storage location that is associated with this backup.
- ④ Specifies the namespaces to include in the backup.
- ⑤ The backup storage location that is associated with this backup.
- ⑥ Create a snapshot of the persistent volumes that are associated with the VM.

- 8.2. Click Operators > Installed Operators to view the operators list. Scroll down and then click OADP Operator. Click Backup and then click Create Backup.

The screenshot shows the 'Backups' tab for the OADP Operator. The 'Backup' tab is active and highlighted with a red box. A 'Create Backup' button is also highlighted with a red box. The interface displays a message: 'No operands found' and 'Operands are declarative components used to define the behavior of the application.'

- 8.3. Click YAML view and press Shift+Insert to paste the contents of the resource manifest. Then, click Create to proceed.

Project: openshift-adp

## Create Backup

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

Configure via:

- Form view
- YAML view

```

3   kind: Backup
4   metadata:
5     name: my-backup
6     namespace: openshift-adp
7     labels:
8       velero.io/storage-location: oadp-dpa-1

```

**Create** **Cancel** **Download**

**Backup**

Schema

Backup is a Velero resource that represents the capture of Kubernetes cluster state at a point in time (API objects and associated volume state).

 **Note**

Optionally, you can create the resource by using the YAML manifest.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/backup-oadp/backup.yaml
...output omitted...
```

#### 8.4. Wait until the backup completes.

Name	Kind	Status	Labels
B my-backup	Backup	Phase: <span style="color: green;">✓</span> Completed	velero.io/storage-location=oadp-dpa-1

**Create Backup**

#### 8.5. List the backup files in the S3 bucket.

```
[student@workstation ~]$ s3cmd ls --recursive s3://${BUCKET_NAME}/velero/backups/
... s3://backup-obc-.../velero/backups/my-backup/my-backup-csi-
volumesnapshotclasses.json.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-csi-
volumesnapshotcontents.json.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-csi-
volumesnapshots.json.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-itemoperations.json.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-logs.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-podvolumebackups.json.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-resource-list.json.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-results.gz
... s3://backup-obc-.../velero/backups/my-backup/my-backup-volumeinfo.json.gz
```

**Chapter 6 |** Create and Restore Backups of Virtual Machines

```
... s3://backup-obj-.../velero/backups/my-backup/my-backup-volumesnapshots.json.gz
... s3://backup-obj-.../velero/backups/my-backup/my-backup.tar.gz
... s3://backup-obj-.../velero/backups/my-backup/velero-backup.json
```

- 8.6. Get the name of the volume snapshot that was created during the backup operation.

```
[student@workstation ~]$ oc get volumesnapshotcontents \
-l velero.io/backup-name=my-backup -o name
volumesnapshotcontent.snapshot.storage.k8s.io/snapshot-9efb7795-c946-46ea-9be7-4d696f171742
```

- 8.7. Click **Storage > VolumeSnapshotContents** and paste the resource name in the search field.

Name	Status	Size	Actions
<b>snapshot-9efb7795-c946-46ea-9be7-4d696f171742</b>	Ready	30 GiB	⋮

- 8.8. Click the name of the volume snapshot content to view its details.

Name	snapshot-9efb7795-c946-46ea-9be7-4d696f171742	Size	30 GiB
Labels	velero.io/backup-name=my-backup	VolumeSnapshot	velero-centos-stream9-gc582
Annotations	3 annotations	VolumeSnapshotClass	ocs-external-storagecluster-rbdplugin-snapclass
Status	Ready	Deletion policy	Retain
Created at	Nov 6, 2024, 1:18 AM	Driver	openshift-storage.rbd.csi.ceph.com

- 9. Clean up the resources that were created in this exercise.

- 9.1. Delete the **my-project** project.

```
[student@workstation ~]$ oc delete project my-project
project.project.openshift.io "my-project" deleted
```

## 9.2. Delete the backup resource.

```
[student@workstation ~]$ oc delete backup my-backup -n openshift-adp
backup.velero.io "my-backup" deleted
```

## 9.3. Delete the volume snapshot content that was created during the backup process.

```
[student@workstation ~]$ oc delete volumesnapshotcontents \
-l velero.io/backup-name=my-backup
volumesnapshotcontent.snapshot.storage.k8s.io "snapcontent-6ef9dbb5-48af-4507-
befc-5780ef921eb3" deleted
```

## 9.4. Delete the backup objects that were stored in the object bucket during the backup process.

```
[student@workstation ~]$ s3cmd rm --recursive \
s3://${BUCKET_NAME}/velero/backups/my-backup/
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-csi-
volumesnapshotclasses.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-csi-
volumesnapshotcontents.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-csi-
volumesnapshots.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-
itemoperations.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-logs.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-
podvolumebackups.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-resource-
list.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-results.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-
volumeinfo.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup-
volumesnapshots.json.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/my-backup.tar.gz'
delete: 's3://backup-obc-.../velero/backups/my-backup/velero-backup.json'
```

**Finish**

As the student user on the workstation machine, confirm that you are in the /home/student directory and then use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-oadp
```

# Back up and Restore with OADP

## Objectives

- Configure one-time and scheduled backups of virtual machines with OADP and restore virtual machines from a backup.

## OADP Custom Resources

Backup and restore are initiated by creating the corresponding Kubernetes resources in the `openshift-adp` namespace. Administrative access to the `openshift-adp` namespace, or to the `cluster-admin` role, is required to create those resources.

OADP provides the following custom resources for backup and restore:

### Backup

The `backup` resource initiates a single backup attempt. This resource defines the namespaces and resources to include in the backup, and can also include a list of commands to run before or after the backup.

The backup resource definition and the backup information, such as backup logs and the list of included backup resources, are stored in the object storage with the backup.

OADP synchronizes backup definitions between the object storage and the OpenShift cluster to enable restoring backups to a different cluster with the same backup storage location.

If a backup resource exists in the OpenShift cluster but is deleted from the object storage, then OADP deletes the Kubernetes resource. Conversely, if a backup exists in the object storage, but not in OpenShift, then OADP creates the matching backup resource in the cluster.



### Note

Only backups with a `Completed` state are synchronized. The object storage synchronization does not automatically create or remove backup resources with a `Failed` or `PartiallyFailed` state.

### Restore

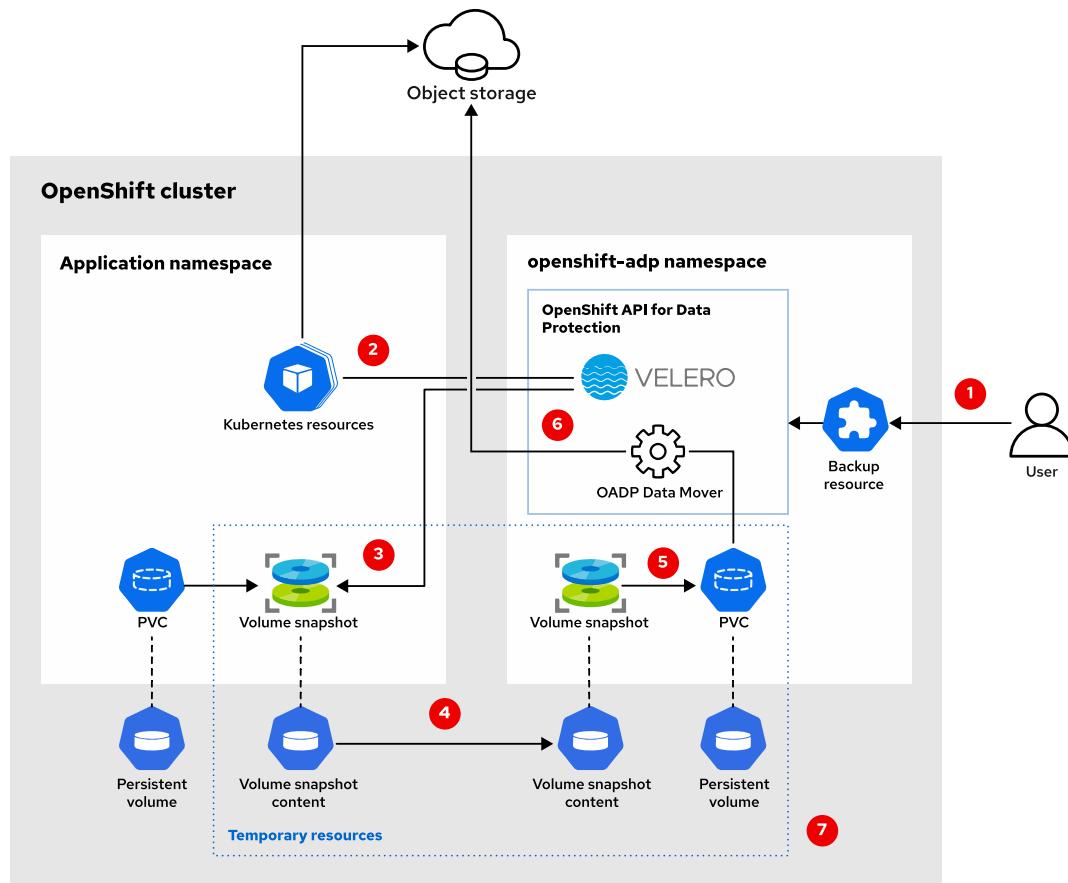
The `restore` resource starts restoring an existing backup resource. The restore result, the list of restored resources, and the restore logs are stored in the object storage.

### Schedule

The `schedule` resource starts a backup on a given schedule that is written in Cron format. A schedule resource is similar to a cron job resource. A schedule defines a backup template to create a backup resource at a recurring interval.

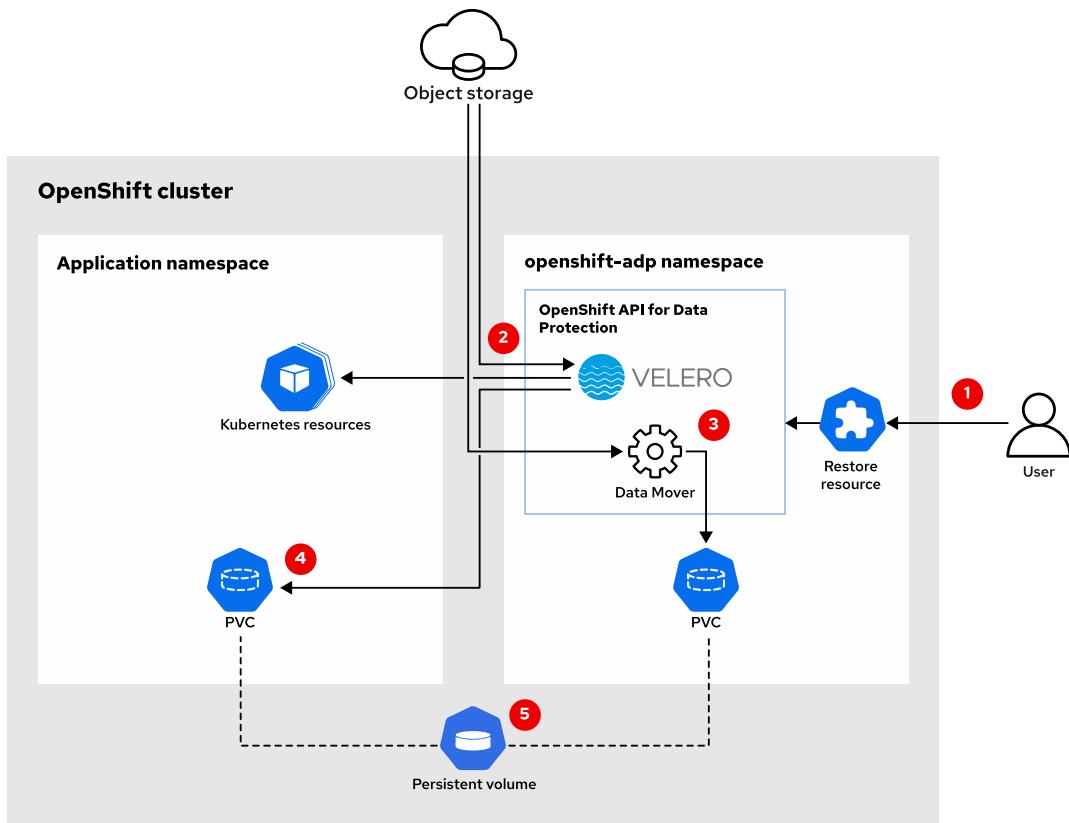
## Backing up and Restoring an OpenShift Application

OADP backs up OpenShift applications by using the following process:



- 1 The administrator creates a backup resource in the `openshift-adp` namespace, which triggers the backup process.
- 2 Velero exports all Kubernetes resources from the application namespace to the backup storage location.
- 3 The Velero CSI plug-in creates a CSI snapshot of the application volume.
- 4 Data Mover clones the `volumeSnapshotContent` and `volumeSnapshot` resources to the `openshift-adp` namespace.
- 5 Data Mover creates a PVC from the volume snapshot.
- 6 Data Mover uses Kopia to transfer the volume data to the backup storage location.
- 7 After the backup completes, Data Mover deletes all volume snapshots and the PVCs that were created during the backup process.

For restoring backups, OADP uses the following process:



1. The administrator creates a restore resource in the **openshift-adp** namespace, which triggers the restore process.
2. Velero imports the Kubernetes resources from the backup storage location to the application namespace.
3. Data Mover creates a temporary PVC in the **openshift-adp** namespace, and transfers the application data from the backup storage location to the new volume.
4. Velero creates the PVC in the application namespace with the same PV as the one that Data Mover is restoring. The PVC stays in the pending state until Data Mover completes the restore.
5. After the transfer is complete, Data Mover releases the PV from the temporary PVC in the **openshift-adp** namespace, and binds that PV to the final PVC in the application namespace.

## The Velero Tool

OADP provides the `velero` command-line tool, which can retrieve backup and restore information from both the object storage and the OpenShift cluster.

The Velero CLI tool is available from the `velero` deployment in the **openshift-adp** namespace. You can define an alias to access the `velero` binary by using the following command:

```
[user@host ~]$ alias velero='\
  oc -n openshift-adp exec deployment/velero -c velero -it -- ./velero'
```

**Note**

The remainder of this section uses the `velero` alias on the command line to refer to the Velero CLI tool.

The `velero` command can use the same syntax as the `kubectl` or `oc` commands, but is limited to OADP Kubernetes custom resources such as the `backup`, `restore`, and `schedule` resources.

The `oc get` command displays limited information about OADP resources. However, the `velero get` command provides the same runtime information from the Kubernetes resources as the `oc describe` command:

```
[user@host ~]$ oc -n openshift-adp get backup,restore
NAME                                AGE
backup.velero.io/website             24h

NAME                                AGE
restore.velero.io/website-stage     4h27m

[user@host ~]$ velero get backup
NAME      STATUS      ERRORS  WARNINGS  CREATED    EXPIRES  STORAGE LOCATION  SELECTOR
website   Completed   0        0          ...        29d      oadp-config-1   app=hugo

[user@host ~]$ velero get restore
NAME      BACKUP      STATUS      STARTED    COMPLETED   ERRORS  WARNINGS  ...
website-stage website   Completed   ...        ...        0        0          ...
```

The Velero tool provides a `describe` command that is similar to the `oc describe` command, and adds a `--details` option that retrieves additional information about the resource from the object storage:

```
[user@host ~]$ velero describe backup website --details
Name:           website
Namespace:      openshift-adp
...output omitted...

Phase: Completed ①

Namespaces: ②
Included:   website
Excluded:  <none>

Resources:
Included:     imagestreams, buildconfigs, deployments, services, routes
Excluded:    <none>
Cluster-scoped: auto

Label selector: app=hugo

...output omitted...

Started: 2024-12-11 14:01:27 +0000 UTC
Completed: 2024-12-11 14:01:42 +0000 UTC
```

```

Expiration: 2024-01-10 14:01:27 +0000 UTC

Total items to be backed up: 7
Items backed up: 7

Resource List: ③
apps/v1/Deployment:
- website/hugo
build.openshift.io/v1/BuildConfig:
- website/hugo
image.openshift.io/v1/ImageStream:
- website/hugo
- website/nginx-122
route.openshift.io/v1/Route:
- website/hugo
v1/Service:
- website/hugo

Velero-Native Snapshots: <none included>

```

- ① Status of the backup.
- ② Definition of the namespaces and resources to back up.
- ③ Resources that are included in the backup. The `--details` option adds this information, which comes from the object storage.

If the backup or restore resource has errors or warnings, then the `--details` option adds them to the command output.

In the following example, the `website` project is backed up without filtering by resource type. OADP backs up all resources with the `app=hugo` label in the `website` namespace.

```

apiVersion: velero.io/v1
kind: Backup
metadata:
  name: website-label
  namespace: openshift-adp
spec:
  includedNamespaces:
  - website
  labelSelector:
    matchLabels:
      app: hugo

```

You can use the `velero create restore` command to restore the previous backup in a new `website-dev` namespace, as follows:

```

[user@host ~]$ velero create restore website-dev \
--from-backup=website-label --namespace-mappings=website:website-dev
Restore request "website-dev" submitted successfully.
Run 'velero restore describe website-dev' or 'velero restore logs website-dev' for
more details.

```

The restore completes with one warning about an existing resource in the target namespace:

```
[user@host ~]$ velero describe restore website-dev --details
Name:      website-dev
Namespace:  openshift-adp
...output omitted...

Phase:          Completed
Total items to be restored: 11
Items restored:   11

Started:    2024-12-12 10:21:21 +0000 UTC
Completed:  2024-12-12 10:21:27 +0000 UTC

Warnings:
  Velero:    <none>
  Cluster:   <none>
  Namespaces:
    website-dev: could not restore, Endpoints "hugo" already exists. Warning: the in-cluster version is different than the backed-up version.
Backup:  website-label

...output omitted...

Resource List:
  apps/v1/Deployment:
    - website-dev/hugo(created)
  build.openshift.io/v1/Build:
    - website-dev/hugo-1(skipped)
  build.openshift.io/v1/BuildConfig:
    - website-dev/hugo(created)
  discovery.k8s.io/v1/EndpointSlice:
    - website-dev/hugo-5hq2x(created)
  image.openshift.io/v1/ImageStream:
    - website-dev/hugo(skipped)
  image.openshift.io/v1/ImageStreamTag:
    - website-dev/hugo:latest(skipped)
  image.openshift.io/v1/ImageTag:
    - website-dev/hugo:latest(skipped)
  route.openshift.io/v1/Route:
    - website-dev/hugo(created)
  v1/Endpoints:
    - website-dev/hugo(failed)
  v1/Namespace:
    - website-dev(created)
  v1/Service:
    - website-dev/hugo(created)
```

In this example, OADP tries unsuccessfully to restore the hugo endpoint. The hugo service in the backup automatically creates this resource.

Avoid including resources that other resources manage, such as builds, endpoints, or replica sets. It is unnecessary and can cause issues during the restore. You must filter to include in the backup only those resources that your application requires for a successful deployment.

## Backing up a Stateful Application with Backup Hooks

OADP creates crash-consistent backups of your application by using volume snapshots. To back up persistent volumes, you must include the `persistentvolumeclaims` and `persistentvolumes` resource types in the backup definition.

OpenShift assigns to each namespace a unique UID and GID that the application pod uses to write data to persistent volumes. If you restore an application to a new namespace, then OpenShift assigns a new set of UIDs and GIDs that prevent the application from accessing its data.



### Note

For more details about user and group ID assignments, refer to the *DO180: Red Hat OpenShift Administration I: Operating a Production Cluster* training course.

So that OADP can restore the UID and GID, include the `namespace` resource type in the backup definition. If you use a label selector in the backup definition, then you must add the corresponding label to the namespace. Alternatively, you can use the `kubernetes.io/metadata.name` label that Kubernetes automatically sets on all namespaces.

To improve the consistency of a backup, you can use backup hooks to specify a list of commands to execute in the application pod before and after the backup is created. You can then use those hooks to quiesce the application and perform an application-consistent backup.

Some applications require additional steps when using volume snapshots to create a usable backup. For more details about the backup procedure, refer to the application documentation. To configure backup hooks, you must specify the target pod (by using labels), the container name, and the commands to run on that container.

You can configure the following hook types according to your needs:

### Pre backup hooks

A pre backup hook is executed before any other backup action on the pod. If the command fails, then the backup stops immediately with the `Failed` status.

As an example, you can use this type of hook to quiesce and prepare the application for backup.

### Post backup hooks

A post backup hook is executed after the backup of the pod and its attached volumes. If the command fails, then the backup stops immediately with the `PartiallyFailed` status.

As an example, you can use this type of hook to resume or unlock the application after the backup is complete.

### Init restore hooks

An init restore hook is executed after the pod and its attached volumes are restored, but before any container on that pod starts. The `init` restore hook defines one or more init containers that follow the same specification as the init container in a pod definition.

OADP does not monitor the status of the init container. Therefore, if the command fails, then the restore continues without any error or warning, but the application pod is in error with the `Init:Error` status.

As an example, you can use this type of hook to restore a database that the application requires and that is external to the OpenShift cluster.

### Post restore hooks

A post restore hook is executed when the application pod is restored and running. If the command fails, then the error is logged and the restore continues.

As an example, you can use this type of hook to run an integrity check on the restored database.

The following example is a backup definition for a MongoDB database. It uses backup hooks to flush all pending writes to the disk and to lock the database to prevent any writes during the backup.

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: mongodb
  namespace: openshift-adp
spec:
  includedNamespaces:
    - mongodb
  orLabelSelectors: ①
    - matchLabels:
        app: mongodb
    - matchLabels:
        kubernetes.io/metadata.name: mongodb
  includedResources: ②
    - deployments
    - services
    - secret
    - pvc
    - pv
    - pods
    - namespace
  hooks:
    resources:
      - name: mongodb-lock
        labelSelector: ③
          matchLabels:
            app: mongodb
    pre: ④
      - exec:
          container: mongodb
          command:
            - /usr/bin/mongosh
            - --eval
            - db.fsyncLock();
    post: ⑤
      - exec:
          container: mongodb
```

```
command:  
- /usr/bin/mongosh  
- --eval  
- db.fsyncUnlock();
```

- ➊ Resources with the `app`: `mongodb` or `kubernetes.io/metadata.name: mongodb` label are included in the backup.
- ➋ The `pvc` and `pv` resource types must be specified in the `includedResources` key to back up the application volume. The `pods` resource type must also be specified for the backup hooks to be executed. The `namespace` resource type must be specified to preserve the UID and GID that are used in the application volume.
- ➌ The hook runs on all pods that match the label selector.
- ➍ The `pre` backup hook executes the `db.fsyncLock()` MongoDB command in the `mongodb` container to lock the database before the volume snapshot.
- ➎ The `post` backup hook executes the `db.fsyncUnlock()` MongoDB command to unlock the database after the backup is completed.



### Important

Because backup and restore hooks are executed only on pods that are included in the backup, you must include the `pod` resource type in the backup.

If the hooks are configured to run on a pod that is not included in the backup, then the hooks are ignored without any error or warning.

The following restore resource definition restores the MongoDB backup from the previous example:

```
apiVersion: velero.io/v1  
kind: Restore  
metadata:  
  name: mongodb  
  namespace: openshift-adp  
spec:  
  backupName: mongodb  
  hooks:  
    resources:  
    - name: mongodb-unlock  
      labelSelector:  
        matchLabels:  
          app: mongodb  
    postHooks:  
    - init: ➊  
      initContainers:  
      - name: remove-lock  
        image: mongodb/mongodb-community-server:7.0-ubi8  
        volumeMounts:  
        - name: mongodb-data  
          mountPath: /data/db
```

```
command:  
- /usr/bin/rm  
- /data/db/mongod.lock
```

- ➊ The `init` restore hook removes the database lock from the backup before the database starts.

OADP Data Mover stores the volume backup in the object storage inside a Kopia unified repository.

A *Kopia unified repository* is a backup repository that Kopia manages for storing volume backups. Kopia encrypts, deduplicates, and compresses the data in the backup repository. OADP uses a dedicated backup repository for each namespace to store all volume backups for that namespace. OADP uses this unified repository to store backups from both volume snapshots and file-system backups.

When the first backup of a namespace is created, OADP initializes a Kopia repository in the object storage, and creates a matching `BackupRepository` resource in the `openshift-adp` namespace.

OADP uses a `BackupRepository` custom resource to store information about the backup repository such as the repository encryption keys and object storage information.

The following excerpt shows the object storage layout after the previous backup and restore examples:

```
s3  
└── oadp  
    ├── backups  
    │   └── mongodb  
    │       ├── mongodb-logs.gz  
    │       └── ...  
    └── kopia ①  
        └── mongodb  
            ├── kopia.repository  
            └── ...  
    └── restores  
        └── mongodb  
            ├── restore-mongodb-logs.gz  
            └── ...
```

- ➊ The `/oadp/kopia` path contains the backup repositories with data from volume backups. Each subdirectory is a unique Kopia repository for a single namespace in the cluster and contains all backups from all volumes in that namespace.

## Scheduling a Recurring Backup

You can back up an application at a recurring interval by using the `schedule` resource. A `schedule` resource is similar to a cron job resource. A `schedule` resource requires a `schedule` in Cron format and a `definition` template of the `backup` resource to create, at the specified time.

The following example is a `schedule` definition of a daily backup for the previous static website example:

```

apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: website-daily
  namespace: openshift-adp
  labels: ①
    app: hugo
spec:
  schedule: "0 7 * * *" ②
  paused: false ③
  template: ④
    includedNamespaces:
    - website
    labelSelector:
      matchLabels:
        app: hugo
    includedResources:
    - imagestreams
    - buildconfigs
    - deployments
    - services
    - routes
  ttl: 720h0m0s

```

- ① The labels that you set on the schedule resource are automatically copied to the backup resources that the schedule creates.
- ② Specifies the schedule for the job in Cron format.
- ③ You can disable the schedule by setting the `paused` parameter to `true`.
- ④ Sets the backup definition template to use the same settings as in a backup resource.

See the references section for more information about the schedule API definition.

To get detailed information about a schedule, use the `velero get schedule` command:

```
[user@host ~]$ velero get schedule
NAME          STATUS  CREATED   SCHEDULE  BACKUP TTL LAST BACKUP  SELECTOR PAUSED
website-daily Enabled ...      0 7 * * * 720h0m0s ...           app=hugo false
```

The status of an activated schedule is `Enabled`. If the schedule is disabled with the `paused` parameter, then its status is `New`. The `LAST BACKUP` column shows the time of the latest backup that the schedule created.

With the Velero tool, you can create a one-time backup by using the same definition as in an existing schedule. To back up an application on demand, you can create a disabled schedule as a backup template. You can then start a new backup with this schedule when you need it.

In this example, you are creating a `pre-upgrade-1.1` backup from the `website-daily` schedule:

```
[user@host ~]$ velero create backup pre-upgrade-1.1 \
--from-schedule website-daily
Creating backup from schedule, all other filters are ignored.
Backup request "pre-upgrade-1.1" submitted successfully.
...output omitted...
```

Backups that are created from a schedule inherit the schedule's labels. In addition, the `velero.io/schedule-name` label is set on the backup resources with the schedule name. You can use those labels to identify the schedules and backups for your application on the `openshift-adp` namespace:

```
[user@host ~]$ velero get backup -l app=hugo
NAME           STATUS    ERRORS   WARNINGS ...
post-upgrade-1.1   Finalizing   0        ...
pre-upgrade-1.1    Completed    0        ...
website-daily-20241215100856  Completed    0        ...
website-daily-20241215091728  Completed    0        ...
```

Because the `openshift-adp` namespace contains the backup and restore resources for all applications that are running on the cluster, it is important to use labels to identify OADP resources that relate to your application.

## Cleaning Backups

Because OADP synchronizes backup definitions with the object storage, OADP automatically recreates any backup that you delete with the `oc` command. To permanently delete a backup, you must delete it from the object storage.

Use the `velero` command to delete backup and restore information from the object storage, and all the associated resources from the cluster:

```
[user@host ~]$ velero delete backup backup-name
Are you sure you want to continue (Y/N)? y
Request to delete backup "backup-name" submitted successfully.
The backup will be fully deleted after all associated data (disk snapshots, backup files, restores) are removed.
```

The backup status changes to `Deleting`, and OADP removes all restore resources that are attached to this backup from both the OpenShift cluster and the object storage. Then, the backup itself is removed from the cluster and the object storage.

You can instruct OADP to delete a backup automatically after a specified elapsed time by using the TTL (Time To Live) setting on the backup and schedule definition. By default, OADP deletes backups after 30 days. The minimum lifetime of a backup is 1 hour.



### Important

OADP does not immediately delete data from the backup repository on the object storage. OADP occasionally runs repository maintenance, such as to delete data that backups no longer need. OADP might start deleting data from the object storage up to 24 hours after a backup is deleted.

## Backing up a Virtual Machine

OADP can back up virtual machines that are running on the OpenShift cluster. To back up a virtual machine, you must include the `kubevirt.io/virtualmachine` resource type in the backup definition.

The following example is a backup definition for a virtual machine named `my-vm`:

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: my-vm-backup
  namespace: openshift-adp
spec:
  includedNamespaces:
    - my-vm
  includedResources:
    - kubevirt.io/virtualmachine
  hooks:
    resources:
      - name: my-vm-pre-backup
        labelSelector:
          matchLabels:
            kubevirt.io/virtualmachine: my-vm
    pre:
      - exec:
          container: my-vm
          command:
            - /usr/bin/virsh
            - suspend
```

In this example, the `pre` backup hook suspends the virtual machine before the backup is created. You can use the `post` backup hook to resume the virtual machine after the backup is complete.

The following example is a restore definition for the `my-vm` virtual machine:

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: my-vm-restore
  namespace: openshift-adp
spec:
  backupName: my-vm-backup
  hooks:
    resources:
      - name: my-vm-post-restore
        labelSelector:
          matchLabels:
            kubevirt.io/virtualmachine: my-vm
    post:
      - exec:
          container: my-vm
```

```
command:  
- /usr/bin/virsh  
- resume my-vm
```

In this example, the `post` restore hook resumes the virtual machine after the restore is complete.

You can use the `velero` command to back up and restore virtual machines in the same way as other Kubernetes resources.

## Backing up Volumes with Kopia

OADP uses the Kopia backup tool to back up volumes that are not compatible with snapshots. Kopia is a backup tool that supports deduplication, encryption, and compression of data. Kopia creates a backup repository in the object storage to store the volume backups.

Kopia backs up volumes by using the following process:

1. The administrator creates a backup resource in the `openshift-adp` namespace that triggers the backup process.
2. Velero exports all Kubernetes resources from the application namespace to the backup storage location.
3. The `node-agent` daemon set, which runs on the same node as the application pod, exports the volume data from the volume mount point on the cluster node to the backup storage location.
4. Kopia creates a backup repository in the object storage to store the volume backups.

Kopia restores volumes by using the following process:

1. The administrator creates a restore resource in the `openshift-adp` namespace, which triggers the restore process.
2. Velero imports the Kubernetes resources from the backup storage location to the application namespace.
3. The `node-agent` daemon set, which runs on the same node as the application pod, imports the application data from the backup storage location to the mount point for the application volume on the cluster node.
4. Kopia restores the volume data from the backup repository to the volume mount point on the cluster node.

You can configure OADP to back up all volumes in a backup definition with Kopia by using the `defaultVolumesToKopiaBackup` option:

```
apiVersion: velero.io/v1  
kind: Backup  
metadata:  
  name: <backup-name>  
  namespace: openshift-adp  
spec:  
  defaultVolumesToKopiaBackup: true
```

You can also annotate the application pod to specify which volumes to back up with Kopia by using the `backup.velero.io/backup-volumes` annotation.

In the following example, the application to back up is a WordPress website that uses a MariaDB database. Use Kopia to back up the volume because the MariaDB volume is not compatible with snapshots.

To enable Kopia for the `mariadb` volume, set the `backup.velero.io/backup-volumes` annotation in the deployment. The following example shows the deployment definition for the MariaDB database:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: website-mariadb
  namespace: org-website
spec:
  template:
    metadata:
      annotations:
        backup.velero.io/backup-volumes: mariadb ①
    spec:
      containers:
        name: db
        image: registry.access.redhat.com/rhscl/mariadb-105-rhel7
        ...output omitted...
        volumeMounts:
          - mountPath: /var/lib/mysql
            name: mariadb
        ...output omitted...
      volumes:
        - name: mariadb
          persistentVolumeClaim:
            claimName: mariadb-data
      ...output omitted...
```

- ① List of volumes to back up with Kopia. Use the same volume name that is defined in the pod definition.

**Note**

You can specify multiple volumes with the `backup.velero.io/backup-volumes` annotation, with a comma to separate each volume. For example, `backup.velero.io/backup-volumes: volume1,volume2,volume3`

**Important**

Only volumes that are compatible with volume snapshots are included in the backup by default. You must enable Kopia to include volumes in your backup that are not compatible with volume snapshots.

OADP stores Kopia backups in the same unified backup repository on the object storage that Data Mover uses for volume snapshots.

## Troubleshooting Backups and Restores

If a backup or restore fails, you can get more information about the failure with the `velero` command:

```
[user@host ~]$ velero get backup mybackup
NAME      STATUS     ERRORS   WARNINGS   ...
mybackup  PartiallyFailed  1        0          ...
```

Use the `velero describe` command for more details about the errors and warnings:

```
[user@host ~]$ velero describe backup mybackup
Name:           mybackup
Namespace:      openshift-adp
Labels:         velero.io/storage-location=oadp-backup-1
Annotations:    velero.io/source-cluster-k8s-gitversion=v1.25.7+eab9cc9
                 velero.io/source-cluster-k8s-major-version=1
                 velero.io/source-cluster-k8s-minor-version=25

Phase:  PartiallyFailed (run `velero backup logs mybackup` for more information)

Errors:
Velero:       <none>
Cluster:      <none>
Namespaces:
database:    resource: /pods name: /mariadb-757c5bdc88-mrwhb error: /command
terminated with exit code 1
...output omitted...
```

You can view the backup logs with the `velero backup logs` command:

```
[user@host ~]$ velero backup logs mybackup
...output omitted...
```

Because the logs are stored in the object storage, you can also download them with the `s3cmd` command, and view them with any text editor.

The logs are stored in the `s3://bucket-name/oadp/backups/backup-name/backup-name-logs.gz` path for backups and in the `s3://bucket-name/oadp/restores/restore-name/restore-name-logs.gz` path for restores.

You can use the following command to download the log file to the current directory with `s3cmd`:

```
[user@host ~]$ s3cmd get \
  s3://backup-bucket/oadp/backups/mybackup/mybackup-logs.gz
download: 's3://backup-bucket/oadp/backups/mybackup/mybackup-logs.gz' ...
  7454 of 7454  100% in    0s    96.14 KB/s  done
```

For a backup that uses hooks, you can search the log with the `hookPhase` keyword to review the status of the hooks. The standard output of the hook command is recorded in the `stdout` line, and the error output in the `stderr` line.

```
[user@host ~]$ velero backup logs mybackup | grep hookPhase
...
... msg="running exec hook" hookCommand="/bin/bash -c mariadb -u root -e \"set
global read_only=1;flush tables\""
... hookPhase=pre

...
... msg="stdout: "
... hookPhase=pre

...
... msg="stderr: ERROR 1045 (28000): Access denied for user
'root'@'localhost' (using password: NO)\n"
... hookPhase=pre

...
... msg="Error executing hook" error="command terminated with exit code 1"
... hookPhase=pre
```

In the previous example, the backup hook was misconfigured and is missing the password to connect to the database.



## References

For more information about backing up applications with OADP, refer to the *OADP Backing Up* section in the *OADP Application Backup and Restore* chapter in the Red Hat OpenShift Container Platform 4.16 *Backup and Restore* documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/backup\\_and\\_restore/index#oadp-backing-up](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/backup_and_restore/index#oadp-backing-up)

For more information about troubleshooting OADP, refer to the *Troubleshooting* section in the *OADP Application Backup and Restore* chapter in the Red Hat OpenShift Container Platform 4.16 *Backup and Restore* documentation at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/backup\\_and\\_restore/index#troubleshooting](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/backup_and_restore/index#troubleshooting)

For more information, refer to the *Velero API Definitions* article at <https://velero.io/docs/v1.14/api-types/>

## ► Guided Exercise

# Back up and Restore an OpenShift Project with OADP

Back up a project, restore the project from backup, and create a backup schedule.

## Outcomes

- Back up and restore an OpenShift project.
- Schedule a backup of an OpenShift project.
- Remove previous backups.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start backup-restore
```

## Instructions

- 1. As the `admin` user, locate and go to the OpenShift web console.

- 1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to the web console URL:

<https://console-openshift-console.apps.ocp4.example.com>

- 1.4. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.

- 2. List the resources in the `my-project` namespace.

- 2.1. Change to the `my-project` project.

```
[student@workstation ~]$ oc project my-project
Now using project "my-project" on server ...
```

2.2. List the resources for the virtual machine and confirm the following statuses:

- The virtual machine status is set to Running.
- The virtual machine instance is present and is marked as ready.
- The virt-launcher pod is running and ready.

```
[student@workstation ~]$ oc get vm
NAME          AGE     STATUS    READY
centos-stream9  ...     Running   True

[student@workstation ~]$ oc get vmi
NAME          AGE     PHASE     IP           NODENAME    READY
centos-stream9  ...     Running   10.10.0.50  master03   True

[student@workstation ~]$ oc get pods
NAME                           READY   STATUS    RESTARTS   AGE
virt-launcher-centos-stream9-frqs5  2/2     Running   0          ...
```

2.3. List the service and route resources. The service exposes the port 80 TCP of the VM, and the route makes the service accessible to clients outside the cluster.

```
[student@workstation ~]$ oc get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
centos-stream9  ClusterIP  172.30.35.206  <none>        80/TCP      ...

[student@workstation ~]$ oc get routes
NAME            HOST/PORT      PATH
centos-stream9  centos-stream9-my-project.apps.ocp4.example.com
  SERVICES      PORT  TERMINATION  WILDCARD
  centos-stream9  http       None
```

2.4. Save the route host name to a variable for later use.

```
[student@workstation ~]$ VM_ROUTE="centos-stream9-my-project.apps.ocp4.example.com"
```

2.5. Connect to the virtual machine to view the status of the HTTP service.

```
[student@workstation ~]$ virtctl ssh root@vm/centos-stream9
Warning: Permanently added 'vm/centos-stream9.my-project' ...
...output omitted...

[root@centos-stream9 ~]#
```

2.6. Confirm that the ansible-playbook command is available.

```
[root@centos-stream9 ~]# which ansible-playbook
/usr/bin/ansible-playbook
```

**Note**

You might need to repeat this command until the intended condition is reached.

2.7. Confirm that the /opt/playbook.yaml Ansible playbook is present.

```
[root@centos-stream9 ~]# ls /opt/playbook.yaml
/opt/playbook.yaml
```

**Note**

You might need to repeat this command until the intended condition is reached.

2.8. Run the /opt/playbook.yaml Ansible playbook to configure the VM.

```
[root@centos-stream9 ~]# ansible-playbook /opt/playbook.yaml
...output omitted...
PLAY [Configure VM] ****
TASK [Gathering Facts] ****
ok: [localhost]
...output omitted...

PLAY RECAP ****
localhost : ok=8  changed=5  unreachable=0  failed=0  skipped=0  rescued=0  ...
```

2.9. Confirm that the httpd process is listening on port 80 TCP.

```
[root@centos-stream9 ~]# systemctl status --no-pager --full httpd.service
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service enabled; preset: disabled)
  Active: active (running) since ...
    Docs: man:httpd.service(8)
   Main PID: 8202 (httpd)
     Status: "Total requests: 2; ...
      Tasks: 177 (limit: 10406)
     Memory: 36.3M
        CPU: 587ms
       CGroup: /system.slice/httpd.service
               ├─8202 /usr/sbin/httpd -DFOREGROUND
               ├─8204 /usr/sbin/httpd -DFOREGROUND
               ├─8205 /usr/sbin/httpd -DFOREGROUND
               ├─8206 /usr/sbin/httpd -DFOREGROUND
               └─8207 /usr/sbin/httpd -DFOREGROUND
```

**Chapter 6 |** Create and Restore Backups of Virtual Machines

```
... centos-stream9.local systemd[1]: Starting The Apache HTTP Server...
... centos-stream9.local systemd[1]: Started The Apache HTTP Server.
... centos-stream9.local httpd[8202]: Server configured, listening on: port 80
```

- 2.10. Exit the virtual machine SSH connection.

```
[root@centos-stream9 ~]# exit
logout
Connection to vm/centos-stream9.my-project closed.
```

- 2.11. Confirm that the banner and the message of the day are shown when connecting to the VM.

```
[student@workstation ~]$ virtctl ssh root@vm/centos-stream9
Red Hat Training ①
#
# This VM is already configured ②
#
...output omitted...
[root@centos-stream9 ~]#
```

- ① The SSH banner is present.
- ② The MOTD message is present.

- 2.12. Disconnect from the SSH session.

```
[root@centos-stream9 ~]# exit
logout
Connection to vm/centos-stream9.my-project closed.
```

- 2.13. Visit the URL that is associated with the route to confirm that the virtual machine responds to HTTP traffic on port 80 TCP.

<http://centos-stream9-my-project.apps.ocp4.example.com/>

```
[student@workstation ~]$ curl -fsSL http://${VM_ROUTE}/
It works!
```

▶ 3. Back up the my-project namespace.

- 3.1. Create a backup of the my-project namespace by using the YAML resource manifest and verify that the backup is completed.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/backup-restore/backup.yaml
backup.velero.io/my-backup created
```

```
[student@workstation ~]$ oc get backup my-backup -n openshift-adp -o yaml
...output omitted...
status:
...output omitted...
phase: Completed
```

```
progress:  
  itemsBackedUp: 70  
  totalItems: 70  
  ...output omitted...
```

**Note**

You might need to repeat this command until the intended condition is reached.

- 3.2. Get the name of the S3 bucket where the backups are stored and save the value to a variable for later use.

```
[student@workstation ~]$ oc extract configmap/backup-obc -n backups \  
  --keys=BUCKET_NAME --to=-  
# BUCKET_NAME  
backup-obc-c32bb5a6-e264-496e-b1ed-9ee79c79898a  
  
[student@workstation ~]$ BUCKET_NAME=backup-obc-c32bb5a6-e264-496e-b1ed-  
9ee79c79898a
```

- 3.3. List the backup files in the S3 bucket.

```
[student@workstation ~]$ s3cmd ls s3://${BUCKET_NAME}/velero/backups/my-backup/  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-csi-  
volumesnapshotclasses.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-csi-  
volumesnapshotcontents.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-csi-  
volumesnapshots.json.gz  
... s3://backup-obc-.../velero/backups/my-backup-itemoperations.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-logs.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-podvolumebackups.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-resource-list.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-results.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-volumeinfo.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup-volumesnapshots.json.gz  
... s3://backup-obc-.../velero/backups/my-backup/my-backup.tar.gz  
... s3://backup-obc-.../velero/backups/my-backup/velero-backup.json
```

- 3.4. Get the name of the volume snapshot that was created during the backup operation.

```
[student@workstation ~]$ oc get volumesnapshotcontents \  
  -l velero.io/backup-name=my-backup -o name  
volumesnapshotcontent.snapshot.storage.k8s.io/snapcontent-1e83282e-8c92-4075-  
af88-4e7f324d61bc
```

- 4. Delete the project and restore it from backup.

- 4.1. Stop the centos-stream9 VM.

```
[student@workstation ~]$ virtctl stop centos-stream9
VM centos-stream9 was scheduled to stop
```

- 4.2. Delete the `my-project` project. This command might take several minutes to complete.

```
[student@workstation ~]$ oc delete --wait project my-project
project.project.openshift.io "my-project" deleted
```

- 4.3. Inspect the `~/D0316/labs/backup-restore/restore.yaml` resource manifest and copy the content of this file.

```
[student@workstation ~]$ cat ~/D0316/labs/backup-restore/restore.yaml
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: my-restore ①
  namespace: openshift-adp ②
spec:
  backupName: my-backup ③
  restorePVs: true ④
```

- ① The name of the restore resource.
  - ② The namespace of the restore resource.
  - ③ The name of the backup that is being restored.
  - ④ Restore all the persistent volumes from a snapshot.
- 4.4. In the web console, click **Operators > Installed Operators** to view the operators list. Select **All projects** from the drop-down list to view the operators that are installed in all projects. Scroll down and click **OADP Operator**.

Operator	Namespace	Status
OADP Operator	openshift-adp	Succeeded Up to date
OpenShift Data Foundation Client	openshift-storage	Succeeded Up to date Plugin available
openshift-storage	openshift-storage	Succeeded Up to date

- 4.5. Scroll horizontally, click **Restore**, and then click **Create Restore**.

**Chapter 6 |** Create and Restore Backups of Virtual Machines

Project: openshift-adp ▾

Installed Operators > Operator details

**OADP Operator**  
1.4.1 provided by Red Hat

Actions ▾

Request PodVolumeBackup PodVolumeRestore **Restore** Schedule ServerStatusRequest VolumeSnapshotLocation

**Restores**

No operands found

Operands are declarative components used to define the behavior of the application.

Create Restore

- 4.6. Click **YAML view** and press **Shift+Insert** to paste the contents of the resource manifest. Then, click **Create** to proceed.

Project: openshift-adp ▾

### Create Restore

Create by completing the form. Default values may be provided by the Operator authors.

Configure via:

- Form view
- YAML view

Alt + F1 Accessibility help | View shortcuts | Show tooltips

```
1 apiVersion: velero.io/v1
2 kind: Restore
3 metadata:
4   name: my-restore
5   namespace: openshift-adp
6 spec:
```

**Create** **Cancel** **Download**

**Restore**

**Schema**

Restore is a Velero resource that represents the application of resources from a Velero backup to a target Kubernetes cluster.

**Note**

Optionally, you can create the resource by using the YAML manifest.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/backup-restore/restore.yaml
...output omitted...
```

- 4.7. Confirm that the restore is completed.

**Restores**

Name ▾ Search by name... /

Name	Kind	Status	Labels
R my-restore	Restore	Phase: <span style="color: green;">✓ Completed</span>	No labels

Create Restore

- 4.8. List the restore resources in the object bucket.

```
[student@workstation ~]$ s3cmd ls s3://${BUCKET_NAME}/velero/restores/my-restore/
... s3://backup-obj-.../velero/restores/my-restore/my-restore-volumeinfo.json.gz
... s3://backup-obj-.../velero/restores/my-restore/restore-my-restore-
itemoperations.json.gz
... s3://backup-obj-.../velero/restores/my-restore/restore-my-restore-logs.gz
... s3://backup-obj-.../velero/restores/my-restore/restore-my-restore-resource-
list.json.gz
... s3://backup-obj-.../velero/restores/my-restore/restore-my-restore-results.gz
```

- 4.9. List the volume snapshot content that is associated with the restore.

```
[student@workstation ~]$ oc get volumesnapshotcontents \
-l velero.io/restore-name=my-restore -o name
volumesnapshotcontent.snapshot.storage.k8s.io/velero-centos-stream9-26htf-48lth
```

▶ 5. Verify that the restore was successful.

- 5.1. Use the `virtctl` command to start the `centos-stream9` VM.

```
[student@workstation ~]$ virtctl start centos-stream9
VM centos-stream9 was scheduled to start
```

- 5.2. Confirm that the VM is running.

```
[student@workstation ~]$ oc get vm
NAME          AGE     STATUS    READY
centos-stream9 ...     Running   True

[student@workstation ~]$ oc get vmi
NAME          AGE     PHASE     IP           NODENAME   READY
centos-stream9 ...     Running   10.10.0.11 master03  True

[student@workstation ~]$ oc get pods
NAME                           READY   STATUS    RESTARTS   AGE
virt-launcher-centos-stream9-nr4fn  2/2     Running   0          ...
```

- 5.3. Confirm that the service and route were restored.

```
[student@workstation ~]$ oc get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
centos-stream9  ClusterIP  172.30.120.169  <none>        80/TCP      ...

[student@workstation ~]$ oc get routes
NAME            HOST/PORT      PATH
centos-stream9  centos-stream9-my-project.apps.ocp4.example.com
               SERVICES      PORT  TERMINATION  WILDCARD
               centos-stream9  http       None
```

- 5.4. Connect to the VM to verify that your changes to the banner and MOTD are present.

```
[student@workstation ~]$ virtctl ssh root@vm/centos-stream9
Red Hat Training ①
#
# This VM is configured ②
#
...output omitted...
[root@centos-stream9 ~]#
```

- ① The SSH banner is present.
- ② The MOTD message is present.



### Note

The `virtctl ssh` command might fail if the operating system on the VM is still starting. You might need to repeat this command until the VM is ready to accept SSH connections.

#### 5.5. Disconnect from the SSH session.

```
[root@centos-stream9 ~]# exit
logout
Connection to vm/centos-stream9.my-project closed.
```

#### 5.6. Confirm that the VM route responds to HTTP traffic.

```
http://centos-stream9-my-project.apps.ocp4.example.com/
```

```
[student@workstation ~]$ curl -fsSL http://${VM_ROUTE}/
It works!
```

### ▶ 6. Create a backup schedule.

#### 6.1. Inspect the `~/D0316/labs/backup-restore/schedule.yaml` resource manifest and copy the content of this file.

```
[student@workstation ~]$ cat ~/D0316/labs/backup-restore/schedule.yaml
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: my-schedule ①
  namespace: openshift-adp ②
spec:
  schedule: "*/5 * * * *" ③
  template: ④
    includedNamespaces:
      - my-project ⑤
  storageLocation: oadp-dpa-1 ⑥
  snapshotVolumes: true ⑦
```

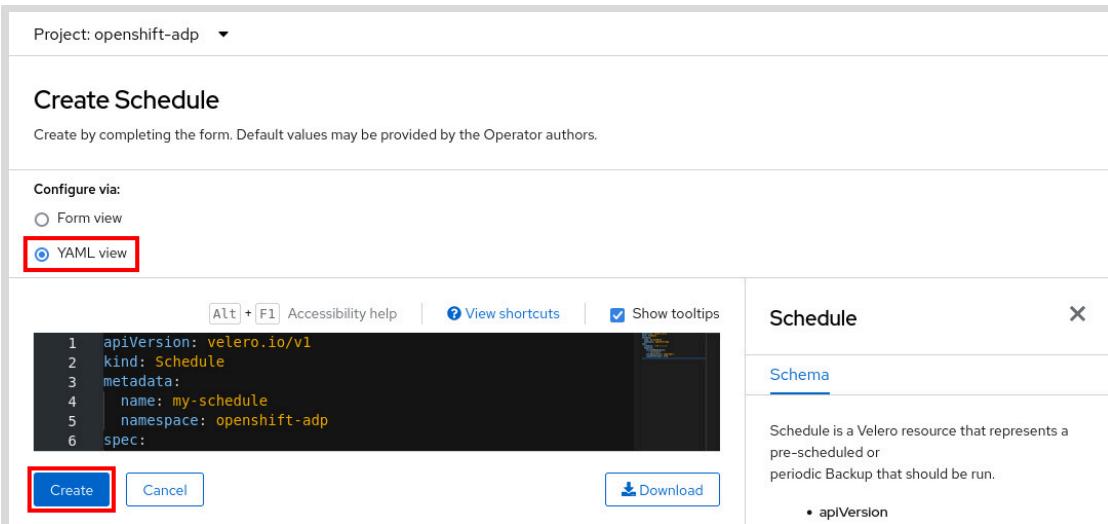
- ① The name of the schedule resource.

**Chapter 6 |** Create and Restore Backups of Virtual Machines

- ② The namespace of the schedule resource.
  - ③ The schedule specification in CRON format (minute, hour, day of the month, month, day of the week). This schedule runs every five minutes.
  - ④ The template of the backup resource.
  - ⑤ Specifies the namespaces to include in the backup.
  - ⑥ The backup storage location that is associated with this backup.
  - ⑦ Create a snapshot of the persistent volumes that are associated with the VM.
- 6.2. Return to the web console and click **Operators > Installed Operators** to view the operators list. Select **All projects** from the drop-down list to view the operators that are installed in all projects. Scroll down and click **OADP Operator**.

- 6.3. Scroll horizontally, click **Schedule**, and then click **Create Schedule**.

- 6.4. Click **YAML** view and press **Shift+Insert** to paste the contents of the resource manifest. Then, click **Create** to proceed.



The screenshot shows the 'Create Schedule' page in the OpenShift web console. At the top, it says 'Project: openshift-adp'. Below that is the title 'Create Schedule' with a note: 'Create by completing the form. Default values may be provided by the Operator authors.' Under 'Configure via:', there are two options: 'Form view' (radio button) and 'YAML view' (radio button, which is selected and highlighted with a red box). The main area contains a code editor showing a YAML manifest:

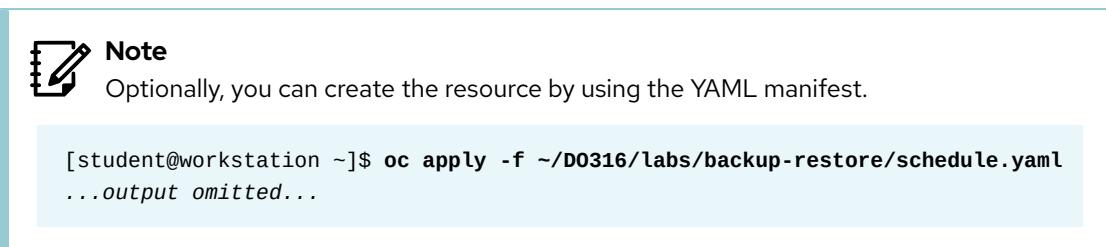
```

1 apiVersion: velero.io/v1
2 kind: Schedule
3 metadata:
4   name: my-schedule
5   namespace: openshift-adp
6 spec:

```

Below the code editor are three buttons: 'Create' (highlighted with a red box), 'Cancel', and 'Download'.

To the right of the code editor is a sidebar titled 'Schedule' with a 'Schema' section. It defines a 'Schedule' as a Velero resource for periodic backups. It includes a link to the schema definition: '• apiVersion'.

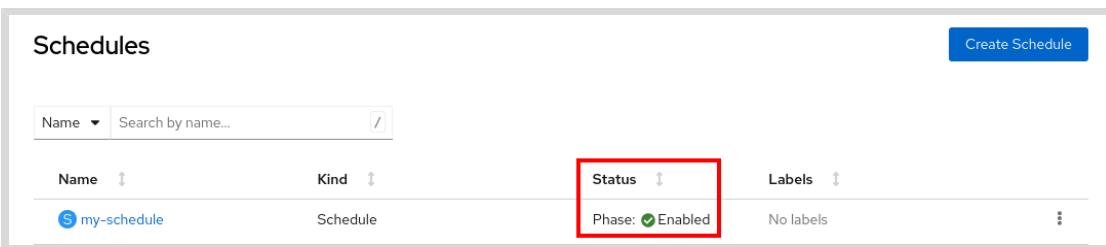


**Note**

Optionally, you can create the resource by using the YAML manifest.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/backup-restore/schedule.yaml
...output omitted...
```

6.5. Confirm that the schedule is enabled.



The screenshot shows the 'Schedules' list page. At the top, there's a search bar labeled 'Search by name...' and a 'Create Schedule' button. Below the search bar is a table header with columns: 'Name' (dropdown), 'Kind' (dropdown), 'Status' (dropdown), and 'Labels'. The table body contains one row for 'my-schedule', which is a 'Schedule' type. The 'Status' column for this row is highlighted with a red box and shows 'Phase: Enabled'.

► 7. View the scheduled backups.

7.1. Scroll horizontally and click **Backup**.

Wait five minutes until the backup is created and completed.

Name	Kind	Status	Labels
my-backup	Backup	Phase: Completed	velero.io/storage-location=oadp-dpa-l
my-schedule-20241116055032	Backup	Phase: Completed	velero.io/schedule-name=my-schedule velero.io/storage-location=oadp-dpa-l

7.2. Get the backup name that was created with the schedule and save it to a variable.

```
[student@workstation ~]$ oc get backups -n openshift-adp \
-l velero.io/schedule-name=my-schedule -o name
backup.velero.io/my-schedule-20241116055032

[student@workstation ~]$ BACKUP_NAME="my-schedule-20241116055032"
```

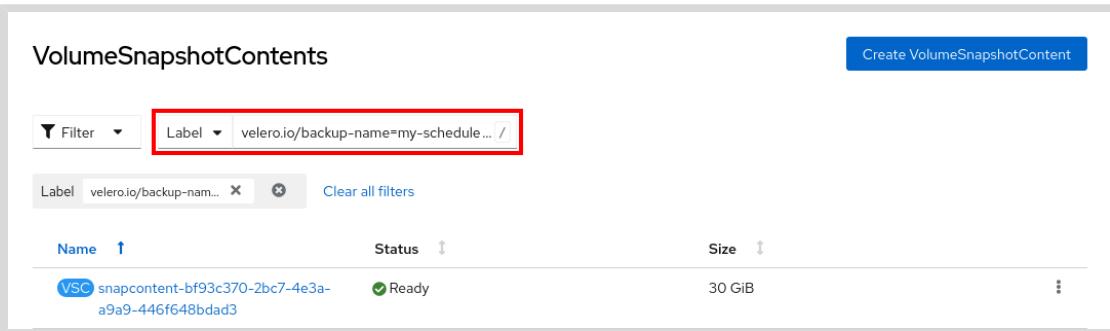
7.3. List the backup files in the S3 bucket.

```
[student@workstation ~]$ s3cmd ls --recursive \
s3://${BUCKET_NAME}/velero/backups/${BACKUP_NAME}/
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-csi-
volumesnapshotclasses.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-csi-
volumesnapshotcontents.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-csi-
volumesnapshots.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-
itemoperations.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-logs.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-
podvolumebackups.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-resource-
list.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-results.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-
volumeinfo.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-
volumesnapshots.json.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../my-schedule-...-tar.gz
... s3://backup-obc-.../velero/backups/my-schedule-.../velero-backup.json
```

7.4. Get the name of the volume snapshot that was created during the backup operation.

```
[student@workstation ~]$ oc get volumesnapshotcontents \
-l "velero.io/backup-name=${BACKUP_NAME}" -o name
volumesnapshotcontent.snapshot.storage.k8s.io/snapcontent-bf93c370-2bc7-4e3a-
a9a9-446f648bdad3
```

- 7.5. In the web console, click **Storage > VolumeSnapshotContents** and paste the resource name into the search field.



The screenshot shows the 'VolumeSnapshotContents' list page. At the top right is a 'Create VolumeSnapshotContent' button. Below it is a search bar with a dropdown menu set to 'Label'. The search term 'velero.io/backup-name=my-schedule ...' is entered and highlighted with a red box. Below the search bar are filter options: 'Label' with a dropdown, 'Clear all filters', and a 'Name' column header. The main table lists one item:

Name	Status	Size	...
VSC snapcontent-bf93c370-2bc7-4e3a-a9a9-446f648bdad3	Ready	30 GiB	...

- 8. Clean up the backup, restore, schedule, and project resources.

- 8.1. Stop the centos-stream9 VM.

```
[student@workstation ~]$ virtctl stop centos-stream9
VM centos-stream9 was scheduled to stop
```

- 8.2. Delete the my-project project. This command might take several minutes to complete.

```
[student@workstation ~]$ oc delete project my-project
project.project.openshift.io "my-project" deleted
```

- 8.3. Delete the backup resource.

```
[student@workstation ~]$ oc delete backup my-backup -n openshift-adp
backup.velero.io "my-backup" deleted
```

- 8.4. Delete the restore resource.

```
[student@workstation ~]$ oc delete restore my-restore -n openshift-adp
restore.velero.io "my-restore" deleted
```

- 8.5. Delete the schedule resource.

```
[student@workstation ~]$ oc delete schedule my-schedule -n openshift-adp
schedule.velero.io "my-schedule" deleted
```

- 8.6. Delete the volume snapshot content that was created during the backup process.

```
[student@workstation ~]$ oc delete volumesnapshotcontents \
-l velero.io/backup-name=my-backup
volumesnapshotcontent.snapshot.storage.k8s.io "snapcontent-1e83282e-8c92-4075-
af88-4e7f324d61bc" deleted
```

- 8.7. Delete the volume snapshot content that was created during the restore process.

```
[student@workstation ~]$ oc delete volumesnapshotcontents \
-l velero.io/restore-name=my-restore
volumesnapshotcontent.snapshot.storage.k8s.io "velero-centos-stream9-26htf-48lth" \
deleted
```

- 8.8. Use the ~/D0316/labs/backup-restore/clean-vsc.sh script to remove the volume snapshot content that was created during the scheduled backups.

```
[student@workstation ~]$ ~/D0316/labs/backup-restore/clean-vsc.sh
oc delete VolumeSnapshotContents \
snapcontent-1e83282e-8c92-4075-af88-4e7f324d61bc \
snapcontent-bf93c370-2bc7-4e3a-a9a9-446f648bdad3
volumesnapshotcontent.snapshot.storage.k8s.io "snapcontent-1e83282e-8c92-4075-
af88-4e7f324d61bc" deleted
volumesnapshotcontent.snapshot.storage.k8s.io "snapcontent-bf93c370-2bc7-4e3a-
a9a9-446f648bdad3" deleted
```

- 8.9. Delete the objects that were stored in the object bucket during this exercise.

```
[student@workstation ~]$ s3cmd rm --recursive s3://${BUCKET_NAME}/velero/
...output omitted...
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-restore
```

## ▶ Lab

# Back up and Restore a Virtual Machine with OADP

Create, restore, and schedule a complete backup of a virtual machine by using the OpenShift API for Data Protection (OADP) operator.

## Outcomes

- Review the configuration of the OADP operator.
- Back up an OpenShift project by using OADP to validate the configuration.
- Back up a project, restore the project from backup, and create a backup schedule.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

The `lab` command deploys the OADP operator, configures the S3 backup location, and creates the data protection application resource to configure the OADP operator.

```
[student@workstation ~]$ lab start backup-review
```

## Instructions

1. As the `admin` user, locate and go to the OpenShift web console.
2. Identify the external endpoint of the S3 service.

The bucket host endpoint in the `backup-obj` configuration map is the name of the `s3` service in the `openshift-storage` namespace.  
You can access the S3 service from outside the cluster by using the endpoint of the `s3` route in the `openshift-storage` namespace.
3. Verify that you can access the S3 bucket.
4. Verify the compute node configuration.
5. Review the resources of the OADP operator. Confirm that the retention policy of the `ocs-external-storagecluster-rbdplugin-snapclass` resource is set to `Retain` and that it has the `velero.io/csi-volumesnapshot-class=true` label.
6. List the resources in the `my-project` namespace.
7. Back up the `my-project` namespace.
8. Delete the project and restore it from backup.
9. Verify that the restore was successful.
10. Create a backup schedule.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade backup-review
```

## Finish

As the student user on the workstation machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-review
```

## ► Solution

# Back up and Restore a Virtual Machine with OADP

Create, restore, and schedule a complete backup of a virtual machine by using the OpenShift API for Data Protection (OADP) operator.

## Outcomes

- Review the configuration of the OADP operator.
- Back up an OpenShift project by using OADP to validate the configuration.
- Back up a project, restore the project from backup, and create a backup schedule.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

The `lab` command deploys the OADP operator, configures the S3 backup location, and creates the data protection application resource to configure the OADP operator.

```
[student@workstation ~]$ lab start backup-review
```

## Instructions

1. As the `admin` user, locate and go to the OpenShift web console.

- 1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to the web console URL.  
<https://console-openshift-console.apps.ocp4.example.com>
  - 1.4. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
2. Identify the external endpoint of the S3 service.

## Chapter 6 | Create and Restore Backups of Virtual Machines

The bucket host endpoint in the backup-`obc` configuration map is the name of the `s3` service in the `openshift-storage` namespace.

You can access the S3 service from outside the cluster by using the endpoint of the `s3` route in the `openshift-storage` namespace.

- 2.1. List the `s3` service and route in the `openshift-storage` namespace.

```
[student@workstation ~]$ oc get service s3 -n openshift-storage
NAME      TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)
s3        LoadBalancer   172.30.154.90  192.168.50.21  80:.../TCP,443:.../TCP, ...
[student@workstation ~]$ oc get route s3 -n openshift-storage
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
s3       s3-openshift-storage.apps.ocp4.example.com          s3        s3-https ...
```

- 2.2. Save the S3 endpoint host to a variable for later use.

```
[student@workstation ~]$ BUCKET_HOST="s3-openshift-storage.apps.ocp4.example.com"
```

- 2.3. Visit the S3 endpoint URL in a web browser to confirm that the endpoint is active.

- <https://s3-openshift-storage.apps.ocp4.example.com/>

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<Resource>/</Resource>
<RequestId>1qaz-2wsx-3edc-4rfv</RequestId>
</Error>
```



### Note

You can safely ignore the Access Denied error message.

3. Verify that you can access the S3 bucket.

- 3.1. Extract the `backup-obc` configuration map data to view the name of the S3 bucket name where the backups are stored.

```
[student@workstation ~]$ oc extract configmap/backup-obc -n backups --to=-
output omitted
# BUCKET_NAME
backup-obc-9b990ae2-1c91-4e8b-98cf-d378e0500108
```

- 3.2. Save the S3 bucket name to a variable for later use.

```
[student@workstation ~]$ BUCKET_NAME="backup-obc-9b990ae2-1c91-4e8b-98cf-
d378e0500108"
```

- 3.3. Extract the `backup-obc` secret data to view the access keys to interact with the S3 bucket.

```
[student@workstation ~]$ oc extract secret/backup-objc -n backups --to=-
# AWS_ACCESS_KEY_ID
QhFEuiusalNBnl80RVtB
# AWS_SECRET_ACCESS_KEY
IbtwoIBVlyLHFhgiSY+Zxb6bXyDIMwhBSv0wasP
```

- 3.4. Verify that the `~/.s3cfg` file has the credentials and the endpoint of the S3 service.

```
[student@workstation ~]$ head -n 5 ~/.s3cfg
[default]
access_key = QhFEuiusalNBnl80RVtB
secret_key = IbtwoIBVlyLHFhgiSY+Zxb6bXyDIMwhBSv0wasP
host_base = s3-openshift-storage.apps.ocp4.example.com
host_bucket = %(bucket).s3-openshift-storage.apps.ocp4.example.com
```

- 3.5. List the available buckets in the S3 service.

```
[student@workstation ~]$ s3cmd ls s3://
2024-10-13 17:37  s3://first.bucket
2024-10-31 01:35  s3://backup-objc-9b990ae2-1c91-4e8b-98cf-d378e0500108
```



### Note

You can safely ignore the `first.bucket` S3 bucket.

4. Verify the compute node configuration.

- 4.1. Confirm that the compute nodes have the `node-role.kubernetes.io/nodeAgent` label by using a label selector.

```
[student@workstation ~]$ oc get nodes -l node-role.kubernetes.io/nodeAgent
NAME      STATUS    ROLES          AGE     VERSION
worker01   Ready     nodeAgent,worker   ...    v1.29.6+aba1e8d
worker02   Ready     nodeAgent,worker   ...    v1.29.6+aba1e8d
```

5. Review the resources of the OADP operator. Confirm that the retention policy of the `ocs-external-storagecluster-rbdplugin-snapclass` resource is set to `Retain` and that it has the `velero.io/csi-volumesnapshot-class=true` label.

- 5.1. Click **Operators > Installed Operators**. Scroll down and click **OADP Operator**. Click **All instances** and verify the status of the data protection resource, the backup storage location resource, and the volume snapshot resource.
- 5.2. Confirm that the retention policy of the `ocs-external-storagecluster-rbdplugin-snapclass` resource is set to `Retain` and that it has the `velero.io/csi-volumesnapshot-class=true` label.

```
[student@workstation ~]$ oc get volumesnapshotclass \
  -l velero.io/csi-volumesnapshot-class=true --show-labels
NAME                                     DRIVER
ocs-external-storagecluster-rbdplugin-snapclass  openshift-storage.rbd.csi.ceph...
  DELETIONPOLICY   AGE   LABELS
  Retain          ...   velero.io/csi-volumesnapshot-class=true
```

6. List the resources in the `my-project` namespace.

6.1. Change to the `my-project` project.

```
[student@workstation ~]$ oc project my-project
Now using project "my-project" on server ...
```

6.2. List the resources for the virtual machine and confirm the following status:

- The virtual machine status is set to **Running**.
- The virtual machine instance is present and is marked as ready.
- The `virt-launcher` pod is running and ready.

```
[student@workstation ~]$ oc get vm
NAME      AGE   STATUS   READY
centos-stream9  ...   Running  True

[student@workstation ~]$ oc get vmi
NAME      AGE   PHASE   IP           NODENAME   READY
centos-stream9  ...   Running  10.8.0.35  master02  True

[student@workstation ~]$ oc get pods
NAME      READY   STATUS   RESTARTS   AGE
virt-launcher-centos-stream9-p7gzm  1/1     Running  0          ...
```

6.3. List the service and route resources.

```
[student@workstation ~]$ oc get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
centos-stream9  ClusterIP  172.30.224.86  <none>        80/TCP      ...

[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS      AGE
centos-stream9  10.8.0.35:80  ...

[student@workstation ~]$ oc get routes
NAME      HOST/PORT      PATH
centos-stream9  centos-stream9-my-project.apps.ocp4.example.com
  SERVICES      PORT      TERMINATION      WILDCARD
  centos-stream9  http      None
```

6.4. Save the route hostname to a variable for later use.

```
[student@workstation ~]$ VM_ROUTE="centos-stream9-my-project.apps.ocp4.example.com"
```

- 6.5. Go to the URL that is associated with the route to confirm that the virtual machine responds to HTTP traffic on port 80 TCP.

```
http://centos-stream9-my-project.apps.ocp4.example.com/
```

```
[student@workstation ~]$ curl http://${VM_ROUTE}/
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from "centos-stream9".</p>
  </body>
</html>
```

7. Back up the `my-project` namespace.

- 7.1. Inspect the `~/D0316/solutions/backup-review/backup.yaml` resource manifest and copy the content of this file.
- 7.2. Click **Operators > Installed Operators**. Scroll down and click **OADP Operator**. Click **Backup** and click **Create Backup**.
- 7.3. Click **YAML view** and press **Shift+Insert** to paste the contents of the resource manifest. Then, click **Create** to proceed.



#### Note

Optionally, you can create the resource by using the YAML manifest.

```
[student@workstation ~]$ oc apply -f \
~/D0316/solutions/backup-review/backup.yaml
...output omitted...
```

- 7.4. Wait until the backup is completed.
- 7.5. Get the name of the volume snapshot that was created during the backup operation.

```
[student@workstation ~]$ oc get volumesnapshotcontents \
-l velero.io/backup-name=my-backup -o name
volumesnapshotcontent.snapshot.storage.k8s.io/snapcontent-9efb7795-
c946-46ea-9be7-4d696f171742
```

- 7.6. Click **Storage > VolumeSnapshotContents** and paste the resource name in the search field.
- 7.7. Click the name of the volume snapshot content to view its details.
8. Delete the project and restore it from backup.

- 8.1. Delete the `my-project` project.

```
[student@workstation ~]$ oc delete --wait project my-project
project.project.openshift.io "my-project" deleted
```

- 8.2. Inspect the `~/D0316/solutions/backup-review/restore.yaml` resource manifest and copy the content of this file.

```
[student@workstation ~]$ cat ~/D0316/solutions/backup-review/restore.yaml
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: my-restore
  namespace: openshift-adp
spec:
  backupName: my-backup
  restorePVs: true
```

- 8.3. In the web console, click **Operators > Installed Operators** to view the operators list. Select All projects from the drop-down list. Scroll down and click **OADP Operator**.
- 8.4. Scroll horizontally, click **Restore**, and click **Create Restore**.
- 8.5. Click **YAML view** and press **Shift+Insert** to paste the contents of the resource manifest. Then, click **Create** to proceed.



#### Note

Optionally, you can create the resource by using the YAML manifest.

```
[student@workstation ~]$ oc apply -f \
~/D0316/solutions/backup-review/restore.yaml
...output omitted...
```

- 8.6. Confirm that the restore is completed.

- 8.7. List the restore resources in the object bucket.

```
[student@workstation ~]$ s3cmd ls s3://${BUCKET_NAME}/velero/restores/my-restore/
... s3://backup-obc-.../velero/restores/my-restore/my-restore-volumeinfo.json.gz
... s3://backup-obc-.../velero/restores/my-restore/restore-my-restore-
itemoperations.json.gz
... s3://backup-obc-.../velero/restores/my-restore/restore-my-restore-logs.gz
... s3://backup-obc-.../velero/restores/my-restore/restore-my-restore-resource-
list.json.gz
... s3://backup-obc-.../velero/restores/my-restore/restore-my-restore-results.gz
```

- 8.8. List the volume snapshot content that is associated with the restore.

```
[student@workstation ~]$ oc get volumesnapshotcontents \
-l velero.io/restore-name=my-restore -o name
volumesnapshotcontent.snapshot.storage.k8s.io/velero-centos-stream9-26htf-48lth
```

**9.** Verify that the restore was successful.

## 9.1. Confirm that the VM is running.

```
[student@workstation ~]$ oc get vm  
NAME          AGE   STATUS    READY  
centos-stream9 ...   Running   True  
  
[student@workstation ~]$ oc get vmi  
NAME          AGE   PHASE     IP           NODENAME   READY  
centos-stream9 ...   Running   10.10.0.11 master03  True  
  
[student@workstation ~]$ oc get pods  
NAME                           READY   STATUS    RESTARTS   AGE  
virt-launcher-centos-stream9-nr4fn  2/2     Running   0          ...
```

## 9.2. Confirm that the service and route were restored.

```
[student@workstation ~]$ oc get services  
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE  
centos-stream9 ClusterIP  172.30.120.169 <none>        80/TCP       ...  
  
[student@workstation ~]$ oc get routes  
NAME          HOST/PORT      PATH  
centos-stream9 centos-stream9-my-project.apps.ocp4.example.com  
  SERVICES      PORT  TERMINATION  WILDCARD  
  centos-stream9  http       None
```

## 9.3. Confirm that the VM route responds to HTTP traffic.

<http://centos-stream9-my-project.apps.ocp4.example.com/>

```
[student@workstation ~]$ curl http://${VM_ROUTE}/  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Hello, World!</title>  
  </head>  
  <body>  
    <p>Hello, World, from "centos-stream9".</p>  
  </body>  
</html>
```

**10.** Create a backup schedule.

## 10.1. Inspect the ~/D0316/solutions/backup-review/schedule.yaml resource manifest and copy the content of this file.

```
[student@workstation ~]$ cat ~/D0316/solutions/backup-review/schedule.yaml  
apiVersion: velero.io/v1  
kind: Schedule  
metadata:  
  name: my-schedule  
  namespace: openshift-adp
```

```
spec:  
  schedule: */5 * * * *  
  template:  
    includedNamespaces:  
      - my-project  
    storageLocation: oadp-dpa-1  
    snapshotVolumes: true
```

- 10.2. Return to the web console and click **Operators > Installed Operators**. Select **All projects** from the drop-down list. Scroll down and click **OADP Operator**.
- 10.3. Scroll horizontally, click **Schedule**, and click **Create Schedule**.
- 10.4. Click **YAML view** and press **Shift+Insert** to paste the contents of the resource manifest. Then, click **Create** to proceed.



### Note

Optionally, you can create the resource by using the YAML manifest.

```
[student@workstation ~]$ oc apply -f \  
~/D0316/solutions/backup-review/schedule.yaml  
...output omitted...
```

- 10.5. Confirm that the schedule is enabled. Scroll horizontally and click **Backup**.

Wait five minutes until the backup is completed.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade backup-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-review
```

# Summary

---

- OADP uses the Velero plug-in to back up and restore the Kubernetes resources.
- Velero uses hooks to run scripts before and after backup and restore operations.
- OADP uses Kopia to move CSI snapshots of application volumes to a remote storage location such as S3.
- Data Mover uses Kopia to upload and encrypt snapshot content to a unified backup repository on the object storage.
- OpenShift Data Foundation provides a unified storage platform for containerized applications.
- ODF uses NooBaa Multicloud Gateway to manage object storage back ends.
- NooBaa MCG provides a single S3-compatible API endpoint for all the object storage back ends.
- Backup and restore operations are started by creating the corresponding Kubernetes resources in the `openshift-adp` namespace.
- OADP synchronizes backup definitions between the object storage and the OpenShift cluster.
- OpenShift Virtualization provides a `VirtualMachineExport` custom resource and the `virtctl vmexport` command to export a VM and its volumes to a storage location.
- With Red Hat OpenShift Virtualization, you can take live snapshots of both running and stopped VMs.
- OpenShift Virtualization uses the `VirtualMachineSnapshot` custom resource to create a snapshot of a VM.



## Chapter 7

# Replicating Virtual Machines by Using Instance Types, Templates, and Clones

### Goal

Create and manage clones, templates, and instance types to provision virtual machines.

### Sections

- Cloning Virtual Machines (and Guided Exercise)
- Create Virtual Machines by Using Predefined Virtual Machine Templates and Instance Types (and Guided Exercise)
- Creating Custom Templates and Instance Types (and Guided Exercise)

### Lab

- Replicating Virtual Machines by Using Instance Types, Templates, and Clones

# Cloning Virtual Machines

## Objectives

- Clone existing virtual machines and individual virtual machine volumes to attach to newly provisioned virtual machines.

## Clone Virtual Machines

A clone is a copy of a VM. Usually, you carefully prepare a VM to use as a model and then clone it to deploy multiple identical machines. Deploying new VMs this way is fast and reliable, because you do not have to install and configure the operating system manually for each machine.

A clone likely has machine-specific data and configuration settings from the original VM. For example, Microsoft Windows clones get the same machine Security Identifier (SID). Red Hat Enterprise Linux clones get the same SSL certificates, SSH host keys, and Red Hat subscriptions.

Attempting to run both the original virtual machine and the cloned virtual machine at the same time can cause problems. This situation is not desirable if you are trying to create another virtual machine with a similar configuration rather than an exact copy.

To prevent the clones from having these same identifying settings, seal the original VM before using it to create clones. Sealing the VM is a process where you clear the machine-specific information. The `virtctl` utility provides tools to assist in the sealing process.

An alternative to using clones is to use templates with boot sources that are sealed. The new virtual machines are then configured with tools such as `cloud-init` or Ansible.

## Prepare Virtual Machines for Cloning

Begin by installing and configuring a VM to use as a model for the clones. To use all the features of OpenShift Virtualization, this VM must include the QEMU guest agent and any tools that are needed for configuring the cloned VMs.



### Note

Linux and Microsoft Windows images from Red Hat include the guest agent. Linux images from Red Hat also include the `cloud-init` configuration utility.

After installing and configuring the VM to use as a model for clones, remove all information that is unique to the VM. Unique information includes hardware information that is specific to the VM, such as MAC addresses, unique system configurations such as the hostname and static IP addresses, and possibly logs and other data.

Depending on the operating system, you might need to perform these steps manually, or tools might be available to seal the image for you.

For Microsoft Windows systems, use the `sysprep.exe` command to initiate a System Out-of-Box-Experience (OOBE). The process for sealing a Windows VM concludes when you shut down the VM.

For Red Hat Enterprise Linux systems, use the `virt-sysprep` command. The command directly edits the disk block device to remove all information that is unique to the VM. You do not run the command from inside the VM but from a Linux system with access to the VM disk block devices. You must stop the VM before using the `virt-sysprep` command.

OpenShift Virtualization provides the `virtctl guestfs` command to start a container that includes the `libguestfs-tools` package. This package includes the `virt-sysprep` utility and other tools for accessing and modifying VM disk images. The container also attaches the VM's persistent volume claim (PVC) as the `/dev/vda` block device. The `virtctl guestfs` command accepts only a single PVC that is attached to the interactive pod.

The virtual machine must be shut down before you use this command, and disk images must not be edited concurrently. If you want to preserve the existing contents of the VM, then you must copy, take a snapshot, or clone the disk first.



### Warning

It can be dangerous to use `virt-sysprep` or similar tools in write mode on live VMs, or concurrently with other disk editing tools, with a risk of causing disk corruption.

The `virt-sysprep` tool modifies the guest or disk image in place. The guest VM must be shut down.

In the following example, the `virtctl guestfs` command is run to access the `dsk1` PVC in the `golden-vms` namespace:

```
[user@host ~]$ virtctl guestfs -n golden-vms dsk1
Use image: registry.redhat.io/container-native-virtualization/libguestfs-
tools@sha256:591e...64ae
The PVC has been mounted at /dev/vda
Waiting for container libguestfs still in pending, reason: ContainerCreating,
message:
...output omitted...
If you don't see a command prompt, try pressing enter.

bash-5.1$
```

The `virt-sysprep` tool can perform more than 40 operations to seal a machine. In the `guestfs` container, use the `virt-sysprep --list-operations` command to see the actions that can be run to seal the disk image.

```
bash-5.1$ virt-sysprep --list-operations
abrt-data * Remove the crash data generated by ABRT
backup-files * Remove editor backup files from the guest
bash-history * Remove the bash history in the guest
blkid-tab * Remove blkid tab in the guest
ca-certificates Remove CA certificates in the guest
...output omitted...
```

To execute operations on a specific disk image, add the disk with the `-a` option as shown in the following example.

```
bash-5.1$ virt-sysprep -a /dev/vda
[ 0.0] Examining the guest ...
[ 4.2] Performing "abrt-data" ...
...output omitted...
[ 6.8] Setting the machine ID in /etc/machine-id
[ 6.8] SELinux relabelling
[ 28.6] Performing "lvm-uuids" ...
```

Use the `virt-sysprep --enable` option to select only some operations. Use a comma-separated list to enable multiple operations. The following example shows how to remove CA certificates and a user account such as the default account that is created when installing a VM.

```
bash-5.1$ virt-sysprep -a /dev/vda --enable ca-certificates,user-account \
--remove-user-accounts cloud-user
[ 0.0] Examining the guest ...
[ 5.3] Performing "ca-certificates" ...
[ 5.3] Performing "user-account" ...
```

For modifications that are not included with the `virt-sysprep` tool, you can use other tools that are included in the `virtctl guestfs` container. The `virt-edit` command enables you to edit any file in the image. The `virt-customize` command enables you to inject SSH keys or to run package manager commands to update an image. Tools are also available to view files and file system information, copy files into or out of the image, and resize an image. Additionally, you can use the `guestfish` command to create an interactive shell and issue multiple `libguestfs` commands.

Exit the container after all modifications are made.

```
bash-5.1$ exit
exit
[user@host ~]$
```



### Note

The first time that the VM boots after sealing, the operating system re-creates many of the items that the sealing tools stripped out.

Thus, never start a VM that you sealed. If you accidentally start the VM, then you must repeat the process of sealing the image.

## Clone Virtual Machines by Using the OpenShift Web Console

Use the OpenShift web console to create a VM clone:

- Go to **Virtualization > VirtualMachines**, select the VM to clone, and then stop the VM. OpenShift Virtualization can clone only stopped VMs.
- Click **Actions > Clone Virtual Machine** and name the VM to create.
- Optionally, select the checkbox to start the VM after it is created.
- Click **Clone**.

The screenshot shows a 'Clone VirtualMachine' dialog box. At the top, there's a 'Name \*' field containing 'rhel9-clone' with a red border around it. Below it is a checkbox labeled 'Start VirtualMachine once created' which is unchecked and also has a red border around it. The main area contains several configuration options: 'Configuration' (Operating system: Not available), 'Flavor' (1CPU | 2 GiB Memory), 'Workload profile' (server), 'NICs' (default - virtio), and 'Disks' (rootdisk, cloudinitdisk). At the bottom, there are two buttons: a blue 'Clone' button on the left with a red border around it, and a 'Cancel' button on the right.

**Figure 7.1: Clone a VM by completing the form**

The cloned VM has the same system settings as the source VM, including any `cloud-init` or `sysprep` instructions to be run at first boot. When the VM boots, the applications configure any system-specific information. For example, when the `sshd` daemon starts, if it does not have host keys, then it generates new ones.

## Clone Virtual Machines by Using the Command Line

You can also create a clone of a VM from the command line by using the `virtualmachineclone` resource type. The `virtctl create` command can generate a manifest to apply to the cluster.

```
[user@host ~]$ virtctl create clone --source-name golden-vm \
--target-name new-vm
apiVersion: clone.kubevirt.io/v1alpha1
kind: VirtualMachineClone
metadata:
  creationTimestamp: null
  name: clone-rglbs
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: golden-vm
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: new-vm
  template: {}
  status: {}
```

## Clone VM Disks by Using Data Volumes

The Containerized Data Importer (CDI) provides a custom resource definition (CRD) for `datavolume` objects that orchestrate the import, clone, and upload operations that are associated with an underlying PVC.

Each `virtualmachine` resource includes a `dataVolumeTemplates` section that is used to create a `datavolume` resource if the volume does not exist. When OpenShift Virtualization clones a VM, it creates a `datavolume` resource to clone the VM disks.

To clone a disk, OpenShift Virtualization can rely on the back-end storage. If the back-end storage provides a cloning feature and the Container Storage Interface (CSI) driver supports the feature, then OpenShift Virtualization uses that method. Otherwise, if the back-end storage and its CSI driver support snapshots, then OpenShift Virtualization uses temporary snapshots for the cloning process. If the back-end storage does not support snapshots, then OpenShift Virtualization starts Kubernetes pods to copy the content of the source volume to the destination volume. That copy method is usually slower than delegating the cloning process to the back-end storage.

The `StorageProfile` resource that OpenShift Virtualization associates with each storage class uses the `cloneStrategy` parameter to specify the cloning method:

### `csi-clone`

The cloning process relies on the back-end storage cloning feature. This method is also called *CSI volume cloning*.

### `snapshot`

The cloning process relies on the back-end storage snapshot feature. This method is also called *smart cloning*.

### `copy`

The cloning process does not rely on back-end storage. OpenShift Virtualization starts Kubernetes pods for the copying. This method is also called *host-assisted cloning*.

For CSI and smart cloning, the source and target PVCs must have the same storage class and volume mode, and the source volume must not be in use. To clone disks across namespaces, a cluster role and corresponding role binding are necessary to provide permissions for all actions for the `datavolume` resource.

When the `StorageProfile` resource does not include the `cloneStrategy` parameter, OpenShift Virtualization uses snapshots if possible. Otherwise, OpenShift Virtualization uses the `copy` method.

## Clone VM Disks

Instead of cloning a whole VM, you can clone individual disks and then attach the copies to another VM. The PVC with a clone of a sealed image can also be a disk source when creating a VM with a template, or be added to the bootable volumes for use with instance types.

The `datavolume` resources orchestrate the cloning process. Therefore, to clone an individual disk, you must create a `datavolume` resource. The following `datavolume` resource clones the `documentroot` PVC from the `golden-vms` namespace into a new PVC:

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: documentroot-clone1
spec:
```

```

storage: ①
  resources:
    requests:
      storage: 1Gi
  storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
source: ②
  pvc:
    name: documentroot
    namespace: golden-vms

```

- ① The `storage` section provides the new disk characteristics. The disk size must be the same as or larger than the source PVC.
- ② The `source` section provides the details of the PVC to clone.

Run the `oc apply -f file.yaml` command to create the resource. If the VM that uses the source PVC is running, then the `datavolume` resource stays in the `CloneInProgress` phase.

```

[user@host ~]$ oc apply -f documentrootDV.yaml
datavolume.cdi.kubevirt.io/documentroot-clone1 created
[user@host ~]$ oc get datavolume
NAME          PHASE      PROGRESS   RESTARTS   AGE
documentroot-clone1  CloneInProgress           10m

```

After you stop the VM, the cloning process completes. You can attach the new PVC, which has the same name as the `datavolume` resource, to a VM and then access its contents.

```

[user@host ~]$ oc get datavolume
NAME          PHASE      PROGRESS   RESTARTS   AGE
documentroot-clone1  Succeeded            24m
[user@host ~]$ oc get pvc
NAME          STATUS    VOLUME          CAPACITY   ACCESS MODES ...
documentroot-clone1  Bound    pvc-bb80...e067  10Gi      RWX     ...

```

When you create or clone a whole VM, OpenShift Virtualization creates the required `datavolume` resources. When you delete the VM, OpenShift Virtualization also deletes the associated `datavolume` resources and the underlying PVC and PV resources.

In contrast, OpenShift Virtualization does not automatically delete the `datavolume` resources that you create independently of a VM. You can reuse your `datavolume` resources with another VM.



## References

For more information about the `virtctl guestfs` command, refer to the *Deploying libguestfs by Using virtctl* section in the *Red Hat OpenShift Container Platform Virtualization* guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-deploying-libguestfs-with-virtctl\\_virt-using-the-cli-tools](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-deploying-libguestfs-with-virtctl_virt-using-the-cli-tools)

For more information about the `libguestfs-tools` package, refer to the upstream site  
<https://libguestfs.org/>

For more information about cloning VM disks by using data volumes, refer to the *Creating VMs by Cloning PVCs* section in the *Red Hat OpenShift Container Platform Virtualization* guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-creating-vms-by-cloning-pvcs](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-creating-vms-by-cloning-pvcs)

For more information about the required permissions, refer to the *Enabling User Permissions to Clone Data Volumes Across Namespaces* section at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-enabling-user-permissions-to-clone-dataVolumes](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-enabling-user-permissions-to-clone-dataVolumes)

For more information about VM templates and clones, refer to the *Steps to Make a Clean VM for Use as a Template or Clone* article at

<https://access.redhat.com/solutions/198693>

For more information about Windows VMs, refer to the *Create a Windows Golden Image for OpenShift Virtualization* article at

<https://developers.redhat.com/articles/2024/09/09/create-windows-golden-image-openshift-virtualization#>

## ► Guided Exercise

# Clone a Virtual Machine

Prepare and clone the persistent volume claim (PVC) of an existing virtual machine into new data volumes, and then use the data volumes to create additional virtual machines.

## Outcomes

- Prepare a VM for cloning.
- Clone a VM.
- Use DataVolume resources to clone VM disks.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the `replicating-clone` namespace and starts two virtual machines in that namespace: `golden-rhel` and `vm1`.

```
[student@workstation ~]$ lab start replicating-clone
```

## Instructions

### ► 1. Confirm that the two VMs are running.

- 1.1. From a command line, log in to your Red Hat OpenShift cluster as the `vt-admin` user.

```
[student@workstation ~]$ oc login -u vt-admin -p vt-redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Use the `oc` command to select the `replicating-clone` project.

```
[student@workstation ~]$ oc project replicating-clone
Now using project "replicating" on server "https://api.ocp4.example.com:6443".
```

- 1.3. Confirm that the `golden-rhel` and `vm1` VMs are running.

```
[student@workstation ~]$ oc get vm
NAME        AGE     STATUS    READY
golden-rhel  11m    Running   True
vm1         10m    Running   True
```

## ▶ 2. Connect to the console and customize the golden-rhel VM.

- 2.1. From the command line, use the `virtctl` command to connect to the VM console.

```
[student@workstation ~]$ virtctl vnc golden-rhel
```

- 2.2. Log in to the console as the `developer` user with `developer` as the password.

- 2.3. Save the output of the `hostnamectl` and `last` commands.

```
[developer@rhel-golden ~]$ hostnamectl > info-host
```

```
[developer@rhel-golden ~]$ last > info-last
```

- 2.4. Confirm that the information is saved.

```
[developer@rhel-golden ~]$ cat info*
Static hostname: golden-rhel
Icon name: computer-vm
Chassis: vm
Machine ID: 608c24c2b786803a650cad1b805a1e10
Boot ID: aef8e770dfc44667a5f070f68115623c
...output omitted...
develope ttyS0                      Fri Sep 13 15:07  still logged in
reboot   system boot  5.14.0-362.24.1. Fri Sep 13 14:34  still running
```

## ▶ 3. Go to the Red Hat OpenShift web console, to stop and then clone the golden-rhel VM.

- 3.1. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Select `htpasswd_provider` and log in as the `vt-admin` user with `vt-redhatocp` as the password.
- 3.2. If necessary, switch to the **Administrator** perspective in the web console. Go to **Virtualization > VirtualMachines** and select the **replicating-clone** project from the **Project** list.
- 3.3. Select the **golden-rhel** VM and click **Actions > Stop**. Wait for the machine to stop.
- 3.4. Click **Actions > Clone**. Set the VM name to `web1`, select **Start VirtualMachine once created**, and then click **Clone**.

**Clone VirtualMachine**

Name *	web1
<input checked="" type="checkbox"/> Start VirtualMachine once created	
Configuration	Operating system rhel9.3
	Flavor 1CPU   2 GiB Memory
	Workload profile Not available
	NICs default - virtio
	Disks golden-rhel
<b>Clone</b>	<b>Cancel</b>

- 3.5. Go to **Virtualization > VirtualMachines** and confirm that the new **web1** VM is running. Confirm that the status is **Running**.

▶ **4.** Verify the contents of the cloned VM disks.

- 4.1. From the command line, connect to the console of the **web1** VM.

```
[student@workstation ~]$ virtctl vnc web1
```

- 4.2. Log in to the console as the **developer** user with **developer** as the password.
- 4.3. System-specific information is included in the clone. Confirm that the hostname and machine ID are the same and that the log data from the source image is available in the cloned VM.

```
[developer@rhel-golden ~]$ ls info*
info-host  info-last
[developer@rhel-golden ~]$ hostnamectl
Static hostname: golden-rhel
  Icon name: computer-vm
  Chassis: vm
Machine ID: 608c24c2b786803a650cad1b805a1e10
  Boot ID: dd6b575de84c4050b65cc48e8cdd4611
...output omitted...

[developer@rhel-golden ~]$ last
develope  ttyS0          Fri Sep 13 15:11  still logged in
reboot    system boot 5.14.0-362.24.1. Fri Sep 13 15:11  still running
develope  ttyS0          Fri Sep 13 15:07 - down  (00:01)
reboot    system boot 5.14.0-362.24.1. Fri Sep 13 14:34 - 15:08  (00:34)
```

- 4.4. Close the web1 VM console window.
- 5. List the data volumes that are associated with the web1 VM. Delete the VM and then confirm that OpenShift Virtualization also deletes the data volumes and the underlying PVCs.

- 5.1. From the command line, list the data volumes in the replicating-clone project.

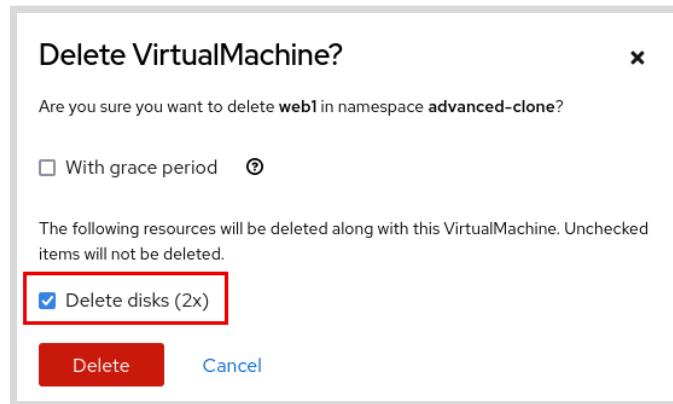
```
[student@workstation ~]$ oc get datavolume
NAME          PHASE      PROGRESS   RESTARTS   AGE
golden-rhel    Succeeded  100.0%    16m
restore-000cefc...4131-golden-rhel  Succeeded  N/A       7m7s
restore-000cefc...4131-webroot    Succeeded  N/A       7m7s
vm1           Succeeded  100.0%    15m
webroot       Succeeded  100.0%    16m
```

The CSI cloning process includes a unique identifier in front of the name of the source data volumes, in the preceding output.

- 5.2. List the PVCs in the project. PVCs have the same name as the associated data volumes.

```
[student@workstation ~]$ oc get pvc -o name
persistentvolumeclaim/golden-rhel
persistentvolumeclaim/restore-000cefc...4131-golden-rhel
persistentvolumeclaim/restore-000cefc...4131-webroot
persistentvolumeclaim/vm1
persistentvolumeclaim/webroot
```

- 5.3. Use the OpenShift web console to delete the web1 VM. Go to **Virtualization > VirtualMachines**, select the web1 VM, and then click **Actions > Delete**. Select **Delete disks (2x)** and then click **Delete**.



- 5.4. From the command line, use the oc command to list the data volume and the PVC resources. Notice that the data volumes and PVC resources for the web1 VM no longer exist.

```
[student@workstation ~]$ oc get datavolume
NAME          PHASE      PROGRESS   RESTARTS   AGE
golden-rhel    Succeeded  100.0%    1          22m
vm1           Succeeded  100.0%    1          18m
```

```
webroot      Succeeded   100.0%           22m
[student@workstation ~]$ oc get pvc
NAME        STATUS    VOLUME          CAPACITY  ACCESS MODES ...
golden-rhel  Bound    pvc-ccb7...3667  10Gi     RWX       ...
vm1         Bound    pvc-9ae1...459a  10Gi     RWX       ...
webroot     Bound    pvc-c1de...2358  1Gi      RWX       ...
```

- 6. From the command line, remove system-specific data from the `golden-rhel` PVC to seal the image.

- 6.1. Verify that the `golden-rhel` VM is stopped.

```
[student@workstation ~]$ oc get vm
NAME        AGE      STATUS    READY
golden-rhel  117m    Stopped   False
vm1         116m    Running   True
```

- 6.2. Start a container with the `libguestfs` tools that is connected to the `golden-rhel` PVC

```
[student@workstation ~]$ virtctl guestfs golden-rhel
Use image: registry.redhat.io/container-native-virtualization/libguestfs-tools...
The PVC has been mounted at /dev/vda
Waiting for container libguestfs still in pending, reason: ContainerCreating,
message:
If you don't see a command prompt, try pressing enter.
bash-5.1$
```

- 6.3. Remove the system-specific information from the `golden-rhel` image and change the hostname.

```
bash-5.1$ virt-sysprep -a /dev/vda --hostname localhost.localdomain
[ 0.0] Examining the guest ...
[ 5.5] Performing "abrt-data" ...
...output omitted...
[ 6.9] Setting the machine ID in /etc/machine-id
[ 6.9] Setting the hostname: localhost.localdomain
[ 6.9] SELinux relabelling
[ 28.5] Performing "lvm-uuids" ...
```

- 6.4. Exit the container.

```
bash-5.1$ exit
[student@workstation ~]$
```

- 7. From the command line, clone the `golden-rhel` VM to a new VM named `web2`.

- 7.1. Use the `virtctl` command to create a sample manifest.

```
[student@workstation ~]$ virtctl create clone \
--source-name golden-rhel --target-name web2 \
--name web2-clone
apiVersion: clone.kubevirt.io/v1alpha1
kind: VirtualMachineClone
metadata:
  creationTimestamp: null
  name: web2-clone
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: golden-rhel
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: web2
  template: {}
status: {}
```

- 7.2. Use the `oc apply` command to create the resource. The `~/D0316/labs/replicating-clone/web2-clone.yaml` file contains the output of the `virtctl create` command.

```
[student@workstation ~]$ oc apply \
-f ~/D0316/labs/replicating-clone/web2-clone.yaml
virtualmachineclone.clone.kubevirt.io/web2-clone created
```

- 7.3. Verify that the clone succeeded.

```
[student@workstation ~]$ oc get virtualmachineclone
NAME      PHASE      SOURCEVIRTUALMACHINE   TARGETVIRTUALMACHINE
web2-clone   Succeeded   golden-rhel           web2
```

- 7.4. Start the VM. The `virtualmachineclone` resource does not have an option to start the VM after creation.

```
[student@workstation ~]$ virtctl start web2
VM web2 was scheduled to start
```

- 8. Verify that the hostname changed and that the log files are cleaned.

- 8.1. Connect to the VM console.

```
[student@workstation ~]$ virtctl vnc web2
```

- 8.2. Log in to the console as the `developer` user with `developer` as the password.

- 8.3. The hostname and machine ID are new, and log entries from the source image are removed.

```
[developer@localhost ~]$ hostnamectl
  Static hostname: localhost.localdomain
    Icon name: computer-vm
      Chassis: vm
    Machine ID: 44e13b579ac5f597dbc8b975b0d82d43
      Boot ID: 4ac83b48e2244a309d9355e6f8471439
...output omitted...

[developer@localhost ~]$ last
develope ttyS0          Fri Sep 13 15:20  still logged in
reboot   system boot 5.14.0-362.24.1. Fri Sep 13 15:18  still running
```

- 8.4. Close the VM console window. Return to the command-line prompt on the workstation machine and delete the web2 VM.

```
[student@workstation ~]$ oc delete vm web2
virtualmachine.kubevirt.io "web2" deleted
```

- ▶ 9. Clone the webroot PVC. The golden-rhel VM uses that PVC as an additional disk to store contents for the web server that is running inside the VM.
- 9.1. From the command line, edit the ~/D0316/labs/replicating-clone/dv.yaml file that the lab command prepares. The file defines a DataVolume resource named documentroot-clone1. In the .spec.source.pvc section, update the name parameter to specify the source PVC.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: documentroot-clone1
  namespace: replicating-clone
spec:
  storage:
    resources:
      requests:
        storage: 1Gi
    storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
  source:
    pvc:
      name: webroot
      namespace: replicating-clone
```

- 9.2. Use the oc apply command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/replicating-clone/dv.yaml
datavolume.cdi.kubevirt.io/documentroot-clone1 created
```

- 9.3. Use the oc command to confirm that OpenShift Virtualization creates the data volume and the PVC resources.

```
[student@workstation ~]$ oc get datavolume
NAME          PHASE      PROGRESS   RESTARTS   AGE
documentroot-clone1  Succeeded           30s
...output omitted...

[student@workstation ~]$ oc get pvc
NAME          STATUS    VOLUME          CAPACITY  ACCESS MODES ...
documentroot-clone1  Bound    pvc-7bc3...f58d  1Gi       RWX      ...
...output omitted...
```

▶ 10. Attach the documentroot-clone1 PVC to the vm1 VM.

- 10.1. From the OpenShift web console, go to **Virtualization > VirtualMachines**. Select the **vm1** VM, and then click **Actions > Stop**. Wait for the machine to stop.
- 10.2. Go to **Configuration > Storage** and then click **Add disk**. Complete the form by using the following information. Click **Save** to attach the disk to the VM.

Field	Value
Name	docroot
Source	PVC
PVC project	replicating-clone
PVC name	documentroot-clone1
Type	Disk
Interface	VirtIO



**Note**

If the **Interface** field is set to **scsi** and **virtio** is not available, then click **Cancel** and start over. The web interface takes a few seconds to detect that the VM is stopped.

10.3. Click **Actions > Start**.

- 10.4. Log in to the VM console and confirm that a new 1 GiB device is available. Go to the **Console** tab and log in as the **root** user with **redhat** as the password. Run the **lsblk** command to list the block devices. Notice the 1 GiB **vdb** block device.

```
[root@localhost ~]# lsblk
NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda   252:0    0 10G  0 disk
└─vda1 252:1    0   1M  0 part
└─vda2 252:2    0 200M 0 part /efi
                  /boot/efi
└─vda3 252:3    0 600M 0 part /boot
└─vda4 252:4    0 9.2G 0 part /
vdb   252:16   0   1G  0 disk
```

10.5. Log out of the VM console.

```
[root@localhost ~]# logout
```

- 11. Delete the `vm1` VM to verify that OpenShift Virtualization does not delete `DataVolume` resources that you create independently of a VM. Confirm that the `documentroot-clone1` data volume and the underlying PVC still exist.

- 11.1. Click **Actions > Delete**. Select **Delete disks (1x)** and then click **Delete**.
- 11.2. From the command line, use the `oc` command to list the data volume and PVC resources. Notice that the `documentroot-clone1` data volume and the PVC still exist.

```
[student@workstation ~]$ oc get datavolume
NAME          PHASE      PROGRESS   RESTARTS   AGE
documentroot-clone1   Succeeded           20m
...output omitted...

[student@workstation ~]$ oc get pvc
NAME          STATUS      VOLUME          CAPACITY   ACCESS MODES ...
documentroot-clone1   Bound      pvc-7bc3...f58d   1Gi        RWX
...output omitted...
```

- 11.3. Delete the `documentroot-clone1` data volume and then confirm that OpenShift Virtualization automatically deletes the underlying PVC.

```
[student@workstation ~]$ oc delete datavolume documentroot-clone1
datavolume.cdi.kubevirt.io "documentroot-clone1" deleted

[student@workstation ~]$ oc get pvc
NAME          STATUS      VOLUME          CAPACITY   ACCESS MODES ...
golden-rhel    Bound      pvc-f874...367d   10Gi       RWX
webroot       Bound      pvc-2588...9f7d   1Gi        RWX
...          
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish replicating-clone
```

# Create Virtual Machines by Using Predefined Virtual Machine Templates and Instance Types

---

## Objectives

- Provision virtual machines by using predefined templates and instance types that are configured by using default or customized template values.

## Virtual Machine Templates

Red Hat OpenShift provides resources to help create and administer workload objects in a cluster. One of these resources is preconfigured templates. Templates contain default parameters that you can customize according to your needs. Red Hat OpenShift provides templates for several resource types, including network interfaces, PVs, and VMs.

Red Hat OpenShift Virtualization provides a catalog of preconfigured templates to create a virtual machine and its resources. Templates are available for different versions of Red Hat Enterprise Linux, CentOS, Fedora, Microsoft Windows 10 and 11, and Microsoft Windows Servers.

Red Hat provides virtual machine templates that are preconfigured with the operating system image, default settings for the operating system, workload type, CPU, and memory resources.

With OpenShift Virtualization templates, you can create a VM from a data volume, including cloning an existing Persistent Volume Claim (PVC) as a new data volume or uploading a local disk image. The cloning process creates a `datavolume` resource in the destination project to create and populate the PVC that is used as the VM's bootable disk.



### Note

Additional permissions are required to clone data volumes across namespaces.

For more information about the required permissions, refer to the *Enabling User Permissions to Clone Data Volumes Across Namespaces* link in the references section.

## Attributes of a Template

Preconfigured templates include changeable fields to meet your VM's needs. The default parameters suggest a basic configuration that works with any VM. However, if you customize the VM configuration, then consider the following template fields:

### Disk Source

This field indicates the source of the image to create your VM, and includes the following options:

#### Template default

Use the default template disk source.

#### PVC (clone PVC)

Clone an existing PVC in the cluster to create a PVC.

#### Registry (creates PVC)

Create a PVC by importing content from a container registry.

**URL (creates PVC)**

Create a PVC by importing content from a URL with an HTTP or an S3 endpoint.

**Upload (Upload a new file to a PVC)**

Create a PVC by uploading a new file to a PVC.

**Blank**

Create a blank PVC.

**CPU | Memory**

This field indicates the size of your VM in terms of CPU and memory. Template names from Red Hat indicate the size of the provisioned VM with the following options:

**Tiny**

Create a VM with 1 CPU and 1 GiB memory. Recommended for testing VM creation.

**Small**

Create a VM with 1 CPU and 2 GiB memory. This option is the default for any preconfigured template.

**Medium**

Create a VM with 1 CPU and 4 GiB memory. Appropriate for code testing or to store basic application resources.

**Large**

Create a VM with 2 CPUs and 8 GiB memory. Recommended for systems that require heavy resource consumption.

**Note**

In templates from Red Hat OpenShift Virtualization, you can no longer change the flavor. The template designations for usage and size determine how much memory and CPU are provisioned, and can be changed when configuring parameters.

**Workload**

This field indicates the workload type for your VM, and includes the following options:

**Desktop**

A configuration for a desktop system that prioritizes VM density over VM performance. Red Hat recommends VMs with this configuration.

**Server**

The default option for any preconfigured template, and compatible with various server workloads. This option balances performance and prioritizes VM density over VM performance.

**High performance**

Optimized for high-performance or high-consumption workloads. This option prioritizes VM performance over VM density.

**Note**

The template name indicates the preconfigured usage by workload type and can be changed when customizing VM parameters.

## Disks

A preconfigured Linux-based template has two partitions by default, `cloud-init` and `root` disk. However, you can configure additional disks. Each field has several available options for customizing your template:

### Source

You can create or import a disk from an existing or blank PVC, from an external source such as a container registry or URL, or by using a container in a registry that is accessible from the cluster.

### Type

You can customize the storage type, such as a disk or CD-ROM, according to the needs of your VM.

### Interface

You can select the communication interface of your disk based on compatibility standards and the intended performance of your VM. The available options are `virtIO`, `SATA`, or `SCSI`.

### StorageClass

Select the storage class for the disk. The storage profile sets the optimized access mode, and the volume mode for the storage class.

### Access Mode

You can customize the disk's access mode and override the default storage profile settings. The available access modes are Single user (`RW0`), Shared access (`RWX`), or Read only (`ROX`).

### Volume Mode

You can choose between `Filesystem` and `Block` storage volume modes for your VM, depending on the selected storage class.



### Note

VMs must have a PVC with the `ReadWriteMany` (`RWX`) shared access mode to enable live migration.

## Network interfaces

Red Hat provides templates with a default network interface that is connected to the pod network. You can configure additional network interfaces for your VM. For each field, you have several options to customize your template:

### Interface model

You can choose either `virtio` or `e1000e`, based on your VM's needs and required performance. The `virtio` model is optimized for performance, and most Linux distributions support it. Additional drivers are needed for Windows VMs. Although most operating systems, including Windows, support the `e1000e` model, its performance is slower than the `virtio` model.

### Network

You can select from a list of available network attachment definitions to connect to additional networks. Additional networks must use the `bridge` binding method.

### Type

You can select from a list of binding methods. You must use the `masquerade` binding method on the default pod network. Additional networks must use the `bridge` binding method.

**MAC address**

You can specify a custom MAC address for the network interface. MAC addresses are automatically assigned unless you specify a custom address.

**Scripts**

In this field, you configure `cloud-init` on Linux systems, add authorized SSH keys, or configure Sysprep for Windows machines. Both `cloud-init` and `sysprep` install tools and packages, configure users and passwords, and manage system applications.

## Create a Virtual Machine with a Template

You can create VMs from preconfigured templates in the OpenShift web console. The templates provide default parameters that suggest a basic configuration for your VM's guest operating system.

To create a VM from a template in the web console, go to **Virtualization > Catalog** and click **Template catalog**. Set the project for the VM to be created, and optionally expand the list of templates to include **All templates**.

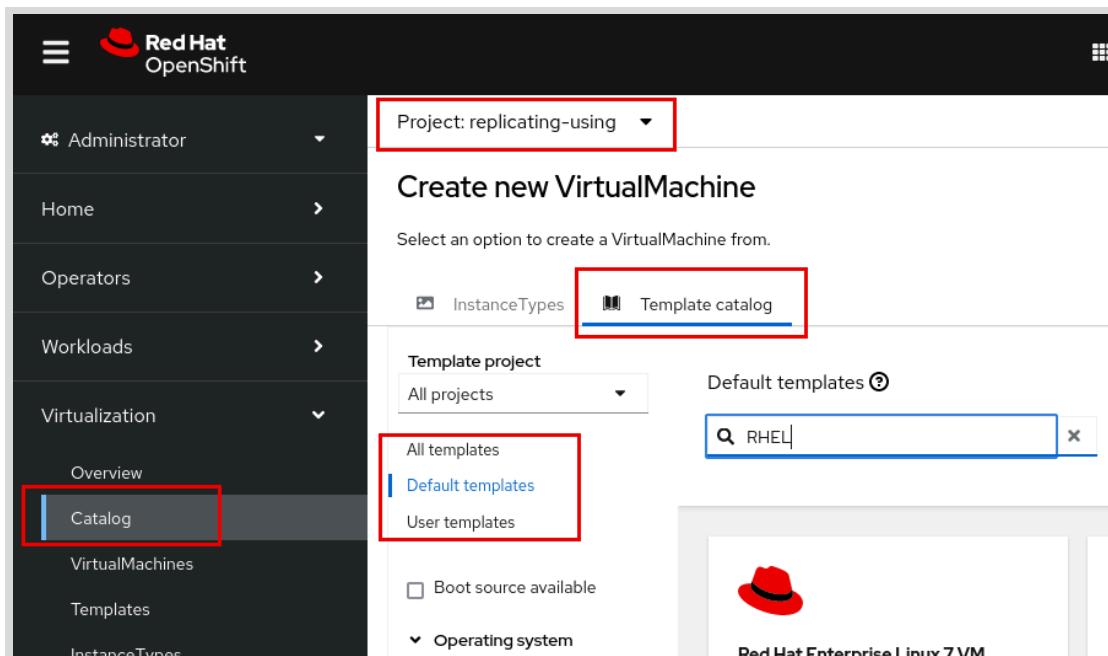
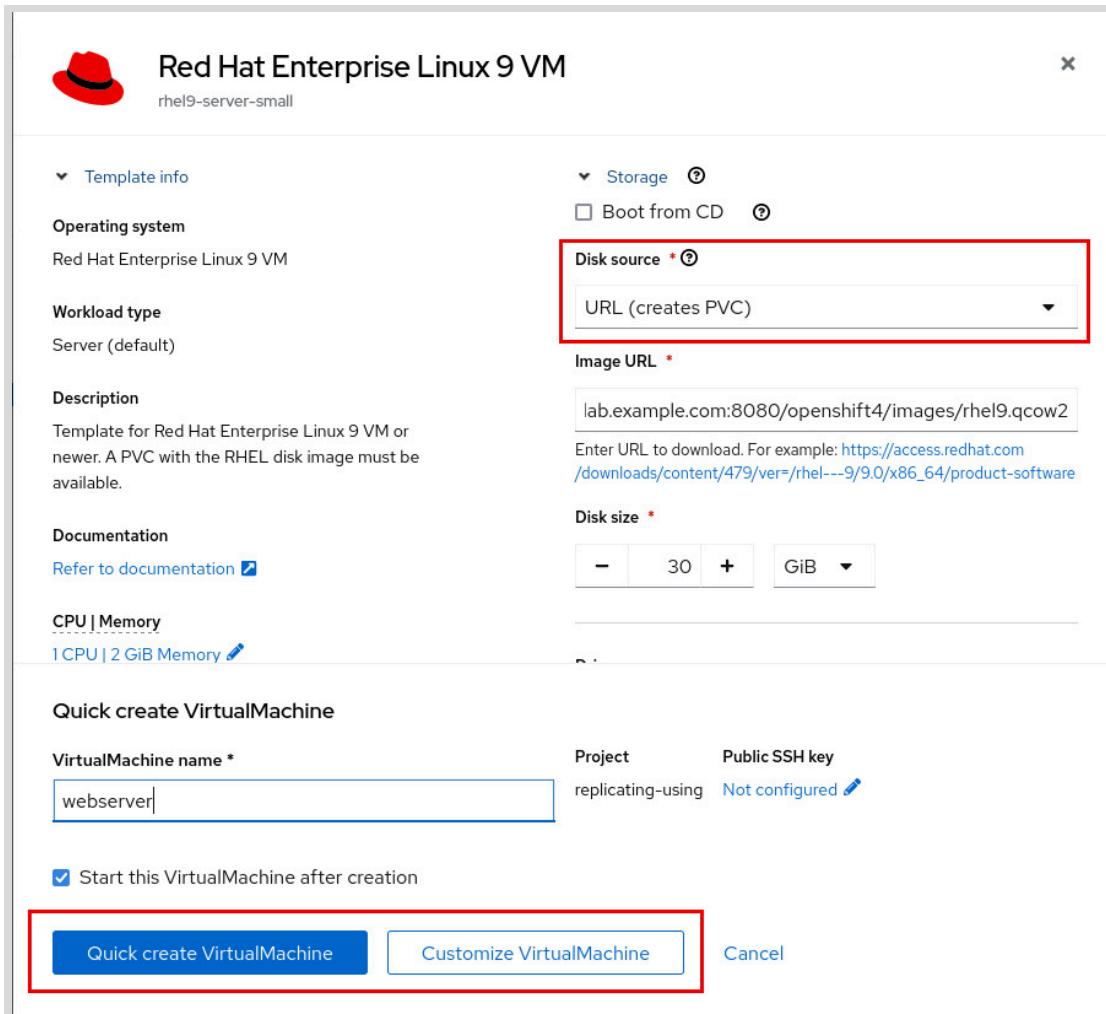
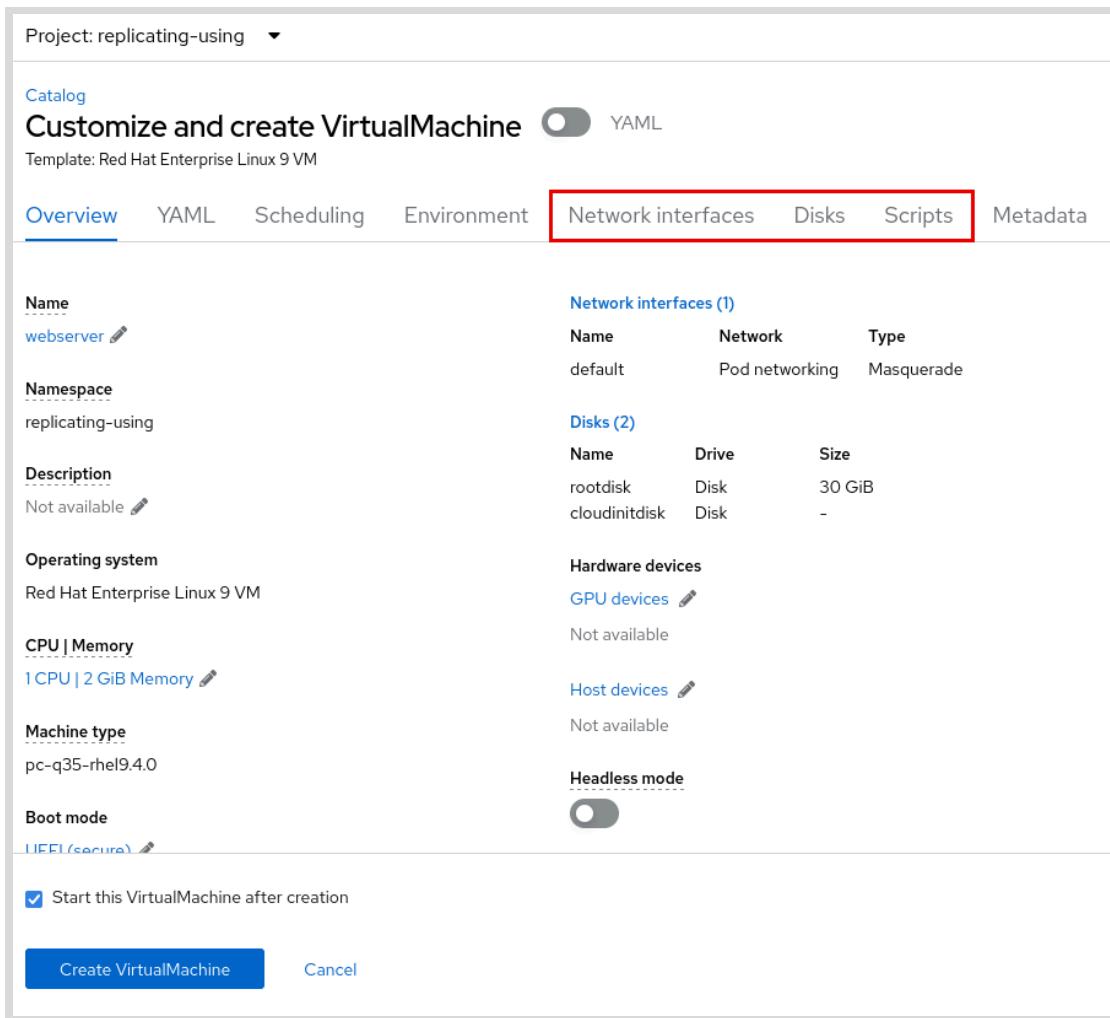


Figure 7.4: Select a template

Select a template and review the initial settings. You can edit the available parameters here and click **Quick create VirtualMachine** or choose **Customize VirtualMachine** to edit these and other attributes. If a template default disk source is not available, then you must update the disk source field before creating the VM.

**Figure 7.5: Red Hat Enterprise Linux 9 VM template**

When you choose **Customize VirtualMachine**, you get the overview page, where you can customize many of the same attributes as on the quick create page. You can also configure network interfaces, disks, scripts, and more from the tabs across the top.

**Figure 7.6: Customize and create a virtual machine**

On the **Scripts** tab, you can configure `cloud-init` scripts and add public SSH keys to the machine. SSH keys that are added with this method are stored in Kubernetes `secret` resources and can be injected into multiple VMs in the namespace.

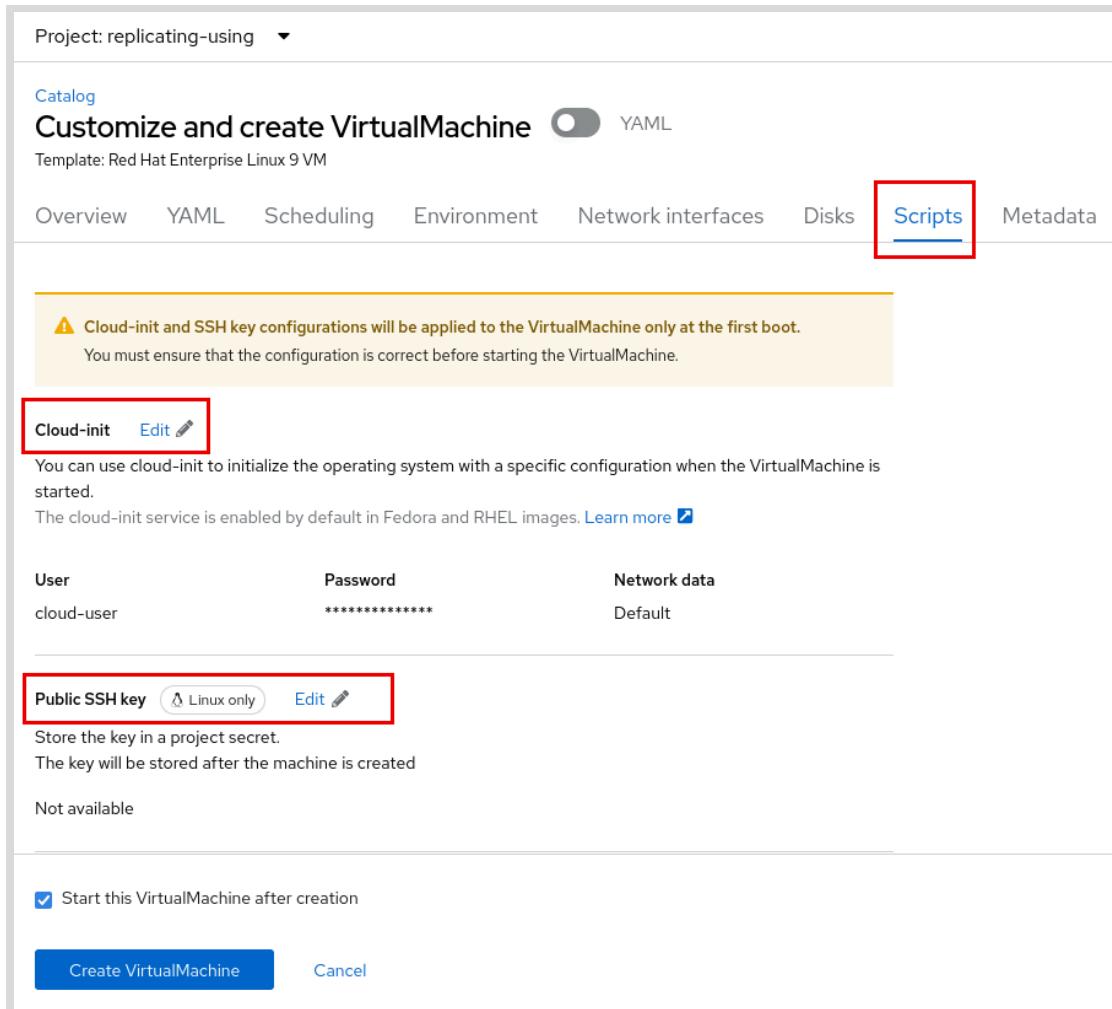


Figure 7.7: Customize initial run scripts

**Note**

A username and a password are created by default. After creating the VM, you can examine the credentials that the `cloud-init` service generates, from `VirtualMachines > VM-Name`. Click the `Console` tab and then click `Guestlogin credentials` above your VM console.

Review the settings of your VM and click `Create VirtualMachine`.

## Virtual Machine Instance Types

Instance types define resources and characteristics to apply to new VMs. Instance types must define CPU and memory resources. All other attributes are optional. When you create a VM from a predefined instance type, you cannot override any parameters that are defined in the instance type.

The following instance types are available on Red Hat OpenShift, each with its own variations of CPU and memory:

**cx1.~**

The CX Series (an abbreviation of "Compute Exclusive") provides exclusive compute resources for compute-intensive applications. The exclusive resources are given to the compute threads of the VM. To ensure that these exclusive compute resources are available, some additional cores (depending on the number of disks and NICs) are requested to offload the I/O threading from the cores that are dedicated to the workload. In addition, in this series, the NUMA topology of the used cores is provided to the VM.

**gn1.~**

The GN Series (an abbreviation of "GPU NVIDIA") provides instance types for VMs with attached NVIDIA GPU resources. This series is intended for VMs that consume GPUs from the NVIDIA GPU Operator [<https://github.com/NVIDIA/gpu-operator>], which is available on OpenShift via OperatorHub.

**m1.~**

The M Series (an abbreviation of "Memory") provides resources for memory-intensive applications.

**n1.~**

The N Series (an abbreviation of "Network") provides resources for network-intensive DPDK applications, such as VNFs. This series of instance types requires nodes that can run DPDK workloads, and the node must have the `node-role.kubevirt.io/worker-dpdk` label.

**u1.~**

The U Series (an abbreviation of "Universal") provides resources for general-purpose applications. VMs of instance types share physical CPU cores on a time-slice basis with other VMs.

To view all the available instance types on Red Hat OpenShift, either go to **Virtualization > InstanceTypes**, or use the following command:

```
[user@host ~]$ oc get vmclusterinstancetypes
NAME          AGE
cx1.2xlarge   7d
...output omitted...
```

You can use the `oc describe` command to list the details of a specific instance type.

```
[user@host ~]$ oc describe vmclusterinstancetype/cx1.large
...output omitted...
Spec:
  Cpu:
    Dedicated CPU Placement:  true
    Guest:  8
    ...output omitted...
  Memory:
    Guest:  16Gi
    ...output omitted...
```

## Virtual Machine Preferences

OpenShift Virtualization includes custom resource definitions for Virtual Machine Cluster Preferences (VMCP), which is a cluster-wide resource, and Virtual Machine Preferences (VMP), which is a namespaced resource.

VMCP and VMP resources contain optional preferences for the VM, such as domain devices, machine, and resource settings. The preferences can also include volume attributes of a VM domain specification. Unlike instance types, which define required values, VMCP and VMP attributes can be overridden when configuring a VM.

OpenShift Virtualization provides VMCPs with various operating systems and workloads.

To view all the available VMCP resources on Red Hat OpenShift, either go to **Virtualization > Preferences**, or use the following command:

```
[user@host ~]$ oc get vmcp
NAME          AGE
alpine        11h
centos.7      11h
...output omitted...
```

You can use the `oc describe` command to list the details of a specific VMCP:

```
[user@host ~]$ oc describe vmcp rhel.9
...output omitted...
Spec:
Devices:
  Preferred Disk Bus:           virtio
  Preferred Disk Dedicated Io Thread: true
  Preferred Interface Model:    virtio
...output omitted...
```

## Bootable Volumes

Bootable volumes are sealed golden images that are used as boot sources when creating VMs and that are defined by a `datasource` resource that is included with OpenShift Virtualization.

The `datasource` metadata includes a recommended default instance type and a virtual machine preference.

Set the `enableCommonBootImageImport` feature gate in the HyperConverged configuration to `true` to download common bootable images. This feature creates and populates the `openshift-virtualization-os-images` project with bootable volumes that are available in Red Hat registries.

To view all the available data source resources, either go to **Virtualization > Bootable volumes**, or use the following command:

```
[user@host ~]$ oc get datasource -A
NAMESPACE          NAME          AGE
openshift-virtualization-os-images centos-stream8 10h
...output omitted...
```

You can use the `oc describe` command to list the details of a specific data source.

```
[user@host ~]$ oc describe datasource centos-stream9 -n
  openshift-virtualization-os-images
Name:      centos-stream9
Namespace:  openshift-virtualization-os-images
...output omitted...
Labels: instancetype.kubevirt.io/default-instancetype=u1.medium
       instancetype.kubevirt.io/default-preference=centos.stream9
...output omitted...
Spec:
  Source:
    Snapshot:
      Name:      centos-stream9-583318eb5646
      Namespace:  openshift-virtualization-os-images
      ...output omitted...
```

The bootable volumes can be updated on a recurring basis. The `dataimportcron` resource defines a cron job for recurring polling or importing disk images as PVCs into a golden image namespace.

You can list the cron job resources for importing golden images with the following command:

```
[user@host ~]$ oc get dataimportcron -n openshift-virtualization-os-images
NAME          FORMAT
centos-7-image-cron   snapshot
centos-stream8-image-cron   snapshot
...output omitted...
```

You can describe the cron job resources with the following command:

```
[user@host ~]$ oc describe dataimportcrons rhel9-image-cron \
  -n openshift-virtualization-os-images
...output omitted...
Spec:
...output omitted...
  Schedule:      29 11/12 * * *
  Template:
  Metadata:
  Spec:
    Source:
      Registry:
        Image Stream:  rhel9-guest
        Pull Method:   node
    Storage:
      Resources:
        Requests:
          Storage:  30Gi
...output omitted...
```

Additional permissions are required to clone data volumes across namespaces. OpenShift Virtualization adds the `os-images.kubevirt.io:view` role to the `authenticated` and `serviceaccounts` groups in the golden image namespace.

```
[user@host ~]$ oc describe rolebinding os-images.kubevirt.io:view \
-n openshift-virtualization-os-images
...output omitted...
Role:
  Kind:  Role
  Name:  os-images.kubevirt.io:view
Subjects:
  Kind   Name           Namespace
  ----  ---           -----
  Group  system:authenticated
  Group  system:serviceaccounts
```

## Create a Virtual Machine with an Instance Type

You can create VMs from preconfigured instance types in the OpenShift web console.

Go to **Virtualization > Catalog**. The default option in the catalog is to create from **Instancetypes**. Select the project to create the VM in, and select the project where bootable volumes can be found.

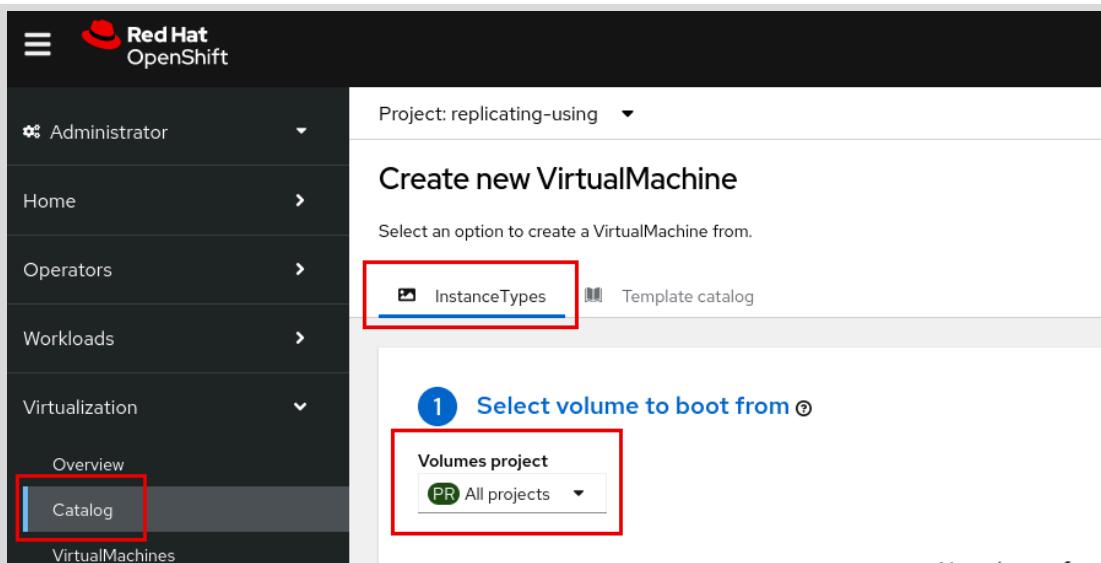


Figure 7.8: Create a VM from an instance type

Select a volume to boot from, and then scroll down to select an instance type. If the datasource metadata for the volume includes a default instance type, then that type is selected. To change the instance type, first select the series and then use the menu for that series to select the size of the instance.

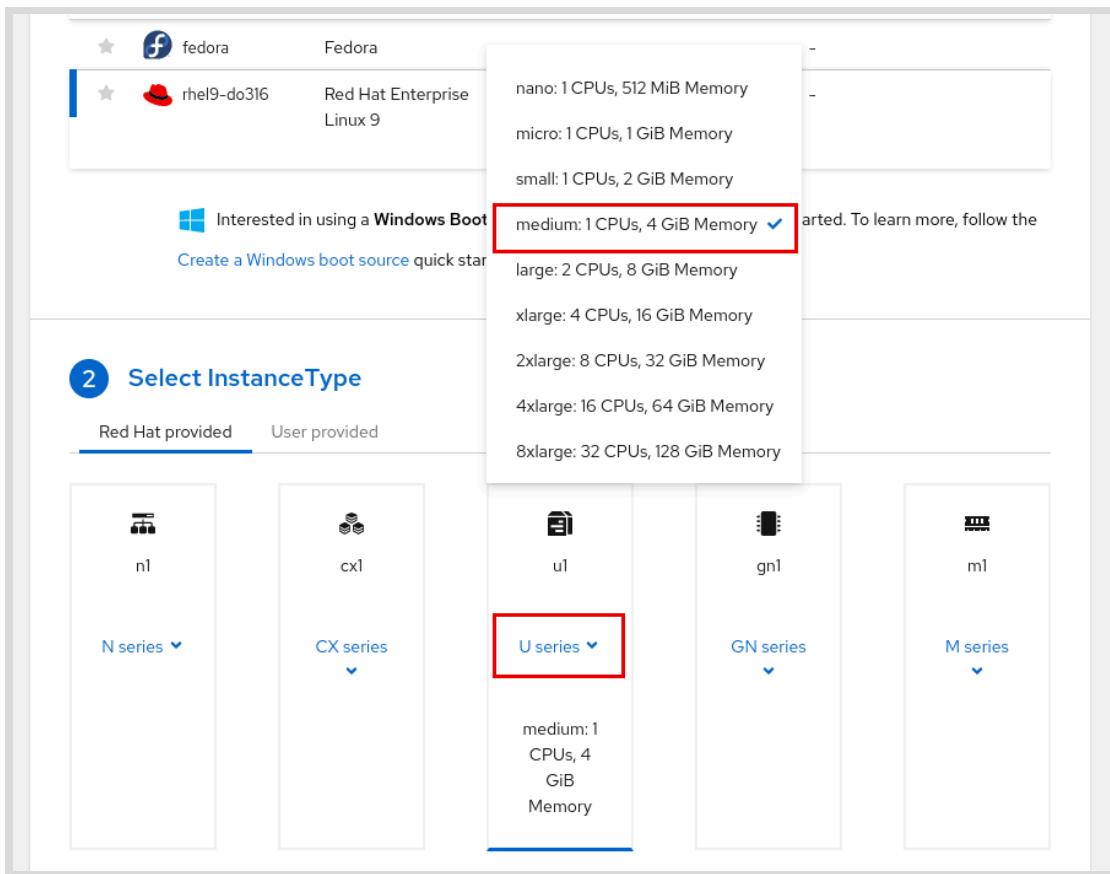


Figure 7.9: Select instance series and size

In the virtual machine details section, you can specify a name for the virtual machine, specify a storage class for the disk, and add a public SSH key.

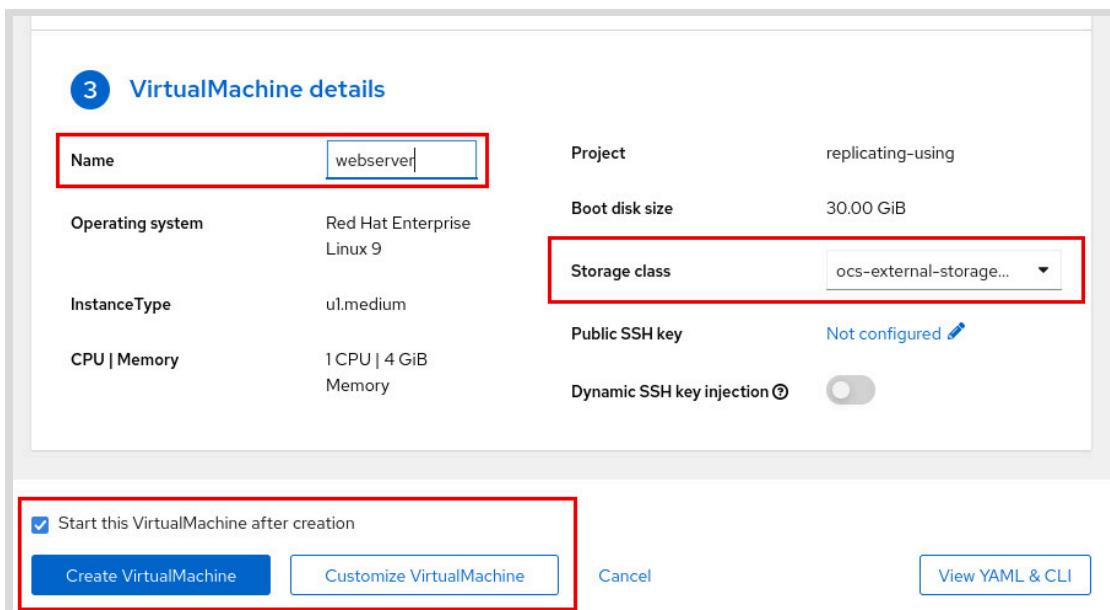


Figure 7.10: Virtual machine details

Settings that are defined in the instance type and the boot source cannot be changed when creating a VM. Other fields, including to add network interfaces, additional disks, and `cloud-init` scripts, can be customized by clicking **Customize VirtualMachine**.

The virtual machine name and hardware settings that are populated from the virtual machine preferences can be changed on the details page. SSH keys are added on the **SSH** tab, and `cloud-init` and sysprep configuration is on the **Initial run** tab.

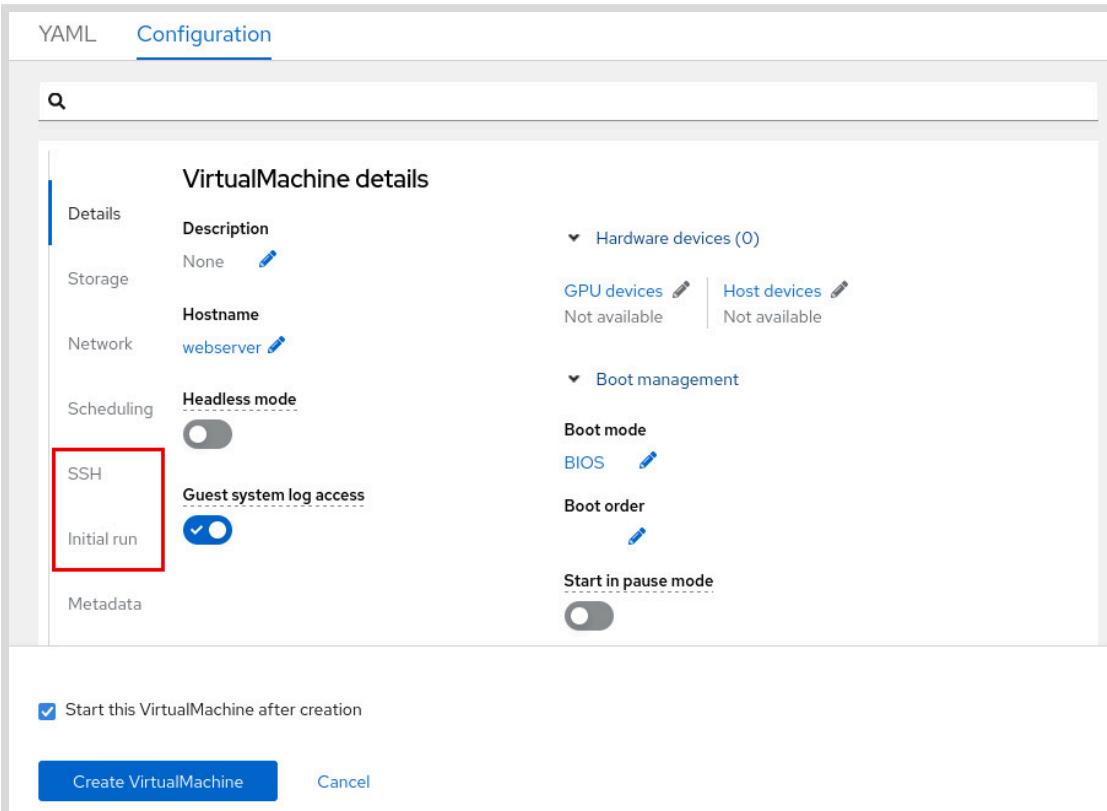


Figure 7.11: Customize and create a Virtual Machine page

Review the settings of your VM and click **Create VirtualMachine**.

## VM Resource File That Uses Instance Types

A VM that is created from the command line can use instance types, preferences, and bootable volumes to specify attributes. A resource file can be generated with the `virtctl` command.

```
[user@host ~]$ virtctl create vm --name rhel9-instancetype \
--instancetype m1.large --infer-preference \
--volume-datasource rhel9 > rhel9-instancetype.yaml
```

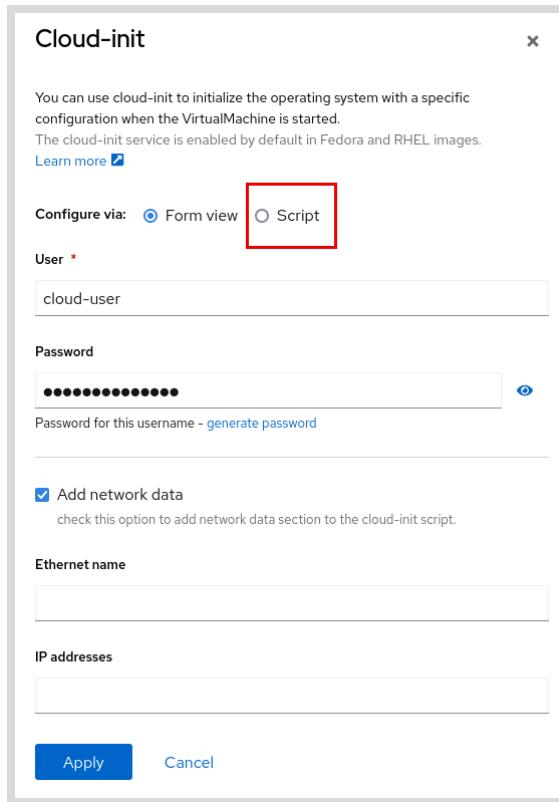
Customize the YAML output, and then use the `oc apply` command to create the VM.

## Initial Run Configuration

The images from Red Hat include the `cloud-init` utility. This tool can customize an image when it first boots. All of the OpenShift Virtualization methods for creating a VM include an option to customize the `cloud-init` scripts.

In the web console, when using either the templates or instance types for creating a VM, when you edit the `cloud-init` script, you can use a form view to change the default user and password, and to configure some network settings.

Switch to the **Script** view to further customize the machine.



**Figure 7.12: Edit cloud-init form view**

Some `cloud-init` modules are shown in the following table:

#### Cloud-init modules

Module name	Usage
users	This module configures users and groups.
rh_subscription	Register a Red Hat system, either by a username and a password, or by an activation key and an organization.
yum_repos	Add yum or dnf repository configuration to /etc/yum.repos.d.
ca_certs	Add CA certificates to the system CA store and update any related files by using the appropriate operating system-specific utility.
rsyslog	Configure remote system logging by using rsyslog.

Module name	Usage
write_files	Write arbitrary content to files and optionally set permissions.
runcmd	Run arbitrary commands at a <code>rc.local</code> -like time frame with output to the console.

**Note**

For a full list of modules and additional examples, refer to the Red Hat *Configuring and Managing cloud-init for RHEL 9* guide and the upstream `cloud-init` documentation links in the References section.

The following example shows adding a user with ssh access and sudo escalation privileges, and also updates the system locale:

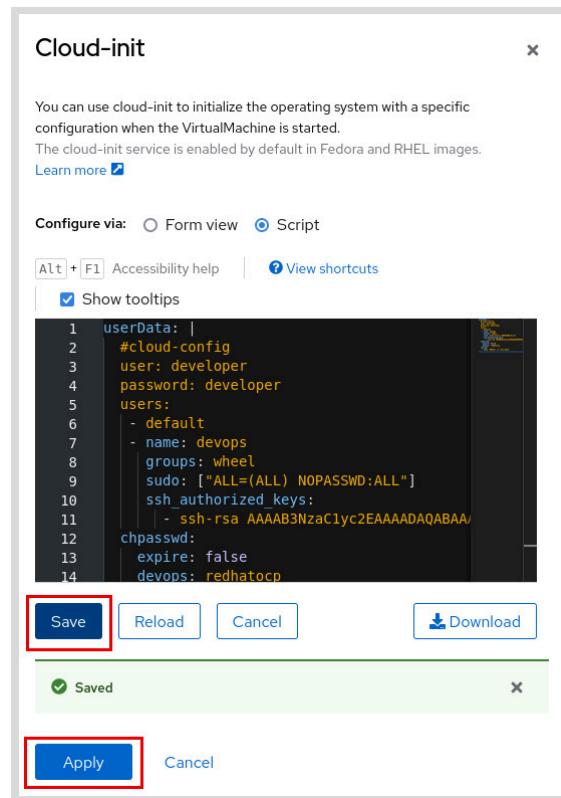
```
#cloud-config
users:
  - default ①
  - name: devops ②
    gecos: Ansible account
    sudo: ["ALL=(ALL) NOPASSWD:ALL"] ③
    groups: wheel,adm,systemd-journal
    ssh_authorized_keys: ④
      - ssh-rsa AA...vz devops@example.com
chpasswd:
  list: |
    root:password
    cloud-user:mypassword
    devops:mypassword2
  expire: False ⑤
  locale: es_MX.UTF-8 ⑥
```

- ① Create the default `cloud-init` user.
- ② Create the `devops` user.
- ③ Add sudo privileges.
- ④ Add SSH public keys.
- ⑤ Force a password change at login: true or false.
- ⑥ Change the system locale.

To manage Red Hat subscriptions, use the `rh_subscription` module:

```
rh_subscription:
  activation-key: example_key
  org: 12345
  auto-attach: True
```

After adding the script, click **Save** and then click **Apply** to return to the virtual machine configuration page.



**Figure 7.13: Edit cloud-init script view**



## References

For more information, refer to the *Red Hat OpenShift Virtualization* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index)

For more information about instance types, refer to the *Red Hat OpenShift Virtualization* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-creating-vms-from-instance-types](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-creating-vms-from-instance-types)

For more information about the `cloud-init` modules, refer to the *Red Hat Support for cloud-init* section at

[https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_and\\_managing\\_cloud-init\\_for\\_rhel\\_9/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html-single/configuring_and_managing_cloud-init_for_rhel_9/index)

Additional `cloud-init` examples are available upstream at

<https://cloudinit.readthedocs.io/en/latest/reference/examples.html>

For more information about the required permissions, refer to the *Enabling User Permissions to Clone Data Volumes Across Namespaces* section at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-enabling-user-permissions-to-clone-datavolumes](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-enabling-user-permissions-to-clone-datavolumes)

For more information on VM instance types and preferences, refer to the *A Simple Path to Virtual Machines* article at

<https://www.redhat.com/en/blog/a-simple-path-to-virtual-machines>

## ► Guided Exercise

# Creating Virtual Machines with Predefined Virtual Machine Templates and Instance Types

Define the attributes of a product template and instance type based on a predefined image and use it to create a medium-sized virtual machine that contains a web server.

## Outcomes

- Create two VMs, one by using a product template and the other by using an instance type.
- Define the boot source as a predefined image and define the size, the workload type, and the custom access credentials.
- Access each VM to confirm that the web servers are up and running.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command also creates the `replicating-using` namespace and exercise resources.

```
[student@workstation ~]$ lab start replicating-using
```

## Instructions

- 1. As the `vt-admin` user, confirm that the `web-server-image` data volume is available in the `vm-images` namespace.
- 1.1. From the command line, log in to your Red Hat OpenShift cluster as the `vt-admin` user with `vt-redhatocp` as the password.

```
[student@workstation ~]$ oc login -u vt-admin -p vt-redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Confirm that the `web-server-image` data volume is accessible in the `vm-images` namespace.

```
[student@workstation ~]$ oc get datavolumes -n vm-images
NAME          PHASE      PROGRESS   RESTARTS   AGE
web-server-image  Succeeded  100.0%        1m
```

- 1.3. Confirm that the corresponding PVC has a 10 GiB capacity and the `Bound` status.

```
[student@workstation ~]$ oc get pvc -n vm-images
NAME           STATUS    VOLUME   CAPACITY   ACCESS MODES   ...
web-server-image  Bound    pvc-...   10Gi       RWX          ...
```

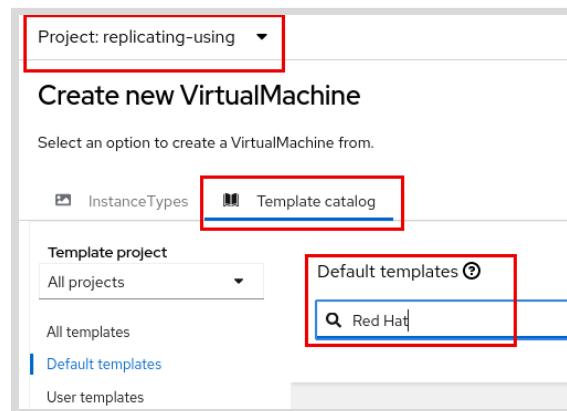
- 2. Use the Red Hat Enterprise Linux 9 VM template to create the `helloworld` VM in the `replicating-using` namespace. Define the boot source as the `web-server-image` PVC in the `vm-images` project.

- 2.1. Open a web browser and go to the web console URL:

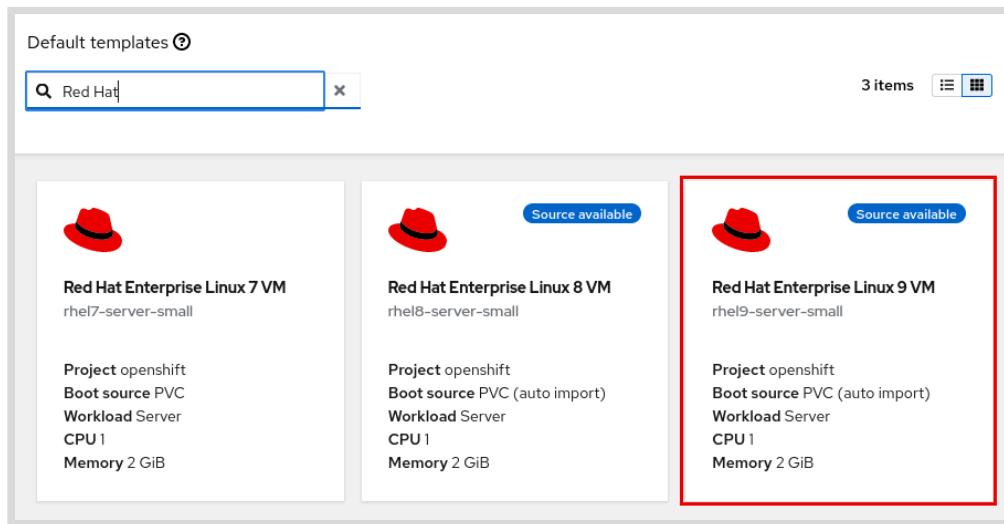
- <https://console-openshift-console.apps.ocp4.example.com>

Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password when prompted.

- 2.2. From the web console left panel, select the **Administrator** view and go to **Virtualization > Catalog**. Select the `replicating-using` project and the **Template catalog** tab. Type `Red Hat` in the search bar to filter the Red Hat VM templates.



Locate and click the Red Hat Enterprise Linux 9 VM template.



- 2.3. When the template overview page is displayed, change the storage and virtual machine name fields according to the following table. Leave the other form fields with their default values and click **Customize VirtualMachine parameters**.

**Red Hat Enterprise Linux 9 VM**

**Template info**

**Operating system**  
Red Hat Enterprise Linux 9 VM

**Workload type**  
Server (default)

**Description**  
Template for Red Hat Enterprise Linux 9 VM or newer. A PVC with the RHEL disk image must be available.

**Documentation**  
[Refer to documentation](#)

**Storage**

Boot from CD

**Disk source \***  
PVC (clone PVC)

**PVC project \***  
PR vm-images

**PVC name \***  
PVC web-server-image

**Disk size \***

**Quick create VirtualMachine**

**VirtualMachine name \***  
helloworld

**Project**  
replicating-using

**Public SSH key**  
Not configured

Start this VirtualMachine after creation

**Buttons**

- Quick create VirtualMachine
- Customize VirtualMachine
- Cancel

Field	Value
Name	helloworld
<b>Storage</b>	
Boot from CD	Unchecked
Disk source	PVC (clone PVC)
PVC project	vm-images
PVC name	web-server-image
Disk size	10 GiB
<b>Drivers</b>	
Mount Windows drivers disk	Unchecked

- 2.4. Go to the **Scripts** tab, locate the **Cloud-init** section, and click **Edit**. Complete the form according to the following table, and click **Apply**.

Field	Value
Configure via	Form view
User	developer
Password	developer

- 2.5. Go to the **Overview** tab, and confirm that the network interface type is **Masquerade**. Confirm that the attached disks include **cloudinitdisk** and **rootdisk**.

Network interfaces (1)		
Name	Network	Type
default	Pod networking	Masquerade

Disks (2)		
Name	Drive	Size
rootdisk	Disk	10 GiB
cloudinitdisk	Disk	-

- 2.6. Scroll down and confirm that the **Workload profile** is set to **Server**. Confirm that **Start this VirtualMachine after creation** is selected. Review the settings of your VM and click **Create VirtualMachine**.

Workload profile: Server

Start this VirtualMachine after creation

**Create VirtualMachine**

- 3. Observe the status of the **helloworld** VM and its resources by using the command line.

- 3.1. Switch to the replicating-using project.

```
[student@workstation ~]$ oc project replicating-using
Now using project "replicating-using" on server "https://
api.ocp4.example.com:6443".
```

- 3.2. Use the `oc get` command to observe the progress of the VM data volume. Repeat the command until the PHASE field has a value of Succeeded.

```
[student@workstation ~]$ oc get datavolumes
NAME      PHASE      PROGRESS   RESTARTS   AGE
helloworld  Succeeded  100.0%        45s
```

- 3.3. Confirm that the `helloworld` VM is running.

```
[student@workstation ~]$ oc get vm
NAME      AGE      STATUS      READY
helloworld  2m      Running    True
```

▶ 4. Access the console of the `helloworld` VM to confirm the system configuration.

- 4.1. From the command line, connect to the console of the `helloworld` VM by using the `virtctl console` command. Press Enter to open the login prompt.

```
[student@workstation ~]$ virtctl console helloworld
Successfully connected to helloworld console. The escape sequence is ^]
...output omitted...
helloworld login:
```

- 4.2. Log in as the `developer` user with `developer` as the password.

```
helloworld login: developer
Password: developer
[developer@helloworld ~]$
```

- 4.3. Use the `systemctl status httpd` command to confirm that the Apache `httpd` service is loaded and running. Press q to exit the status log.

```
[developer@helloworld ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: ...)
  Active: active (running) since Mon 2024-04-25 15:14:33 EDT; 5min ago
    Docs: man:httpd.service(8)
   Main PID: 512 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes ...
...output omitted...
```

- 4.4. Use the `curl` command to confirm that the web server responds to requests. The command returns some HTML content.

```
[developer@helloworld ~]$ curl localhost
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World! Welcome to Red Hat Training.</p>
  </body>
</html>
```

4.5. Disconnect from the console with **Ctrl+]**.

► 5. Shut down and delete the **helloworld** VM.

5.1. Shut down the **helloworld** VM.

```
[student@workstation ~]$ virtctl stop helloworld
VM helloworld was scheduled to stop
```

5.2. Delete the **helloworld** VM.

```
[student@workstation ~]$ oc delete vm helloworld
virtualmachine.kubevirt.io "helloworld" deleted
```

► 6. Create a **web1** VM by using an instance type. Use the **rhel9** boot source and a **cloud-init** script that creates an additional user and modifies the message of the day (MOTD).

6.1. From the web console left panel, go to **Virtualization > Catalog**. Select the replicating-using project and the **Instance Types** tab.



**Note**

If no volumes are shown, then change the volumes project to **openshift-virtualization-os-images**. A user must have permission to view the project to see the available images.

6.2. Locate and select the **rhel9** volume.

**1 Select volume to boot from**

Add volume

Volumes project  
PR openshift-virtualization-os-image ▾ Fil... rhel /

Name rhel X Clear all filters

Volume name	Operating s...	Storage class	Size	Description
rhel8	Red Hat Enterprise Linux 8	ocs-external-storagecluster-rbdplugin-snapclass	30.00 GiB	-
<b>rhel9</b>	Red Hat Enterprise Linux 9	ocs-external-storagecluster-rbdplugin-snapclass	30.00 GiB	-

- 6.3. Scroll down and verify that the instance type is set to U series, medium with 1 CPU and 4 GiB of memory. Set the VM name to web1 and click **Customize VirtualMachine**.

**3 VirtualMachine details**

Name	web1	Project	replicating-using
Operating system	Red Hat Enterprise Linux 9	Boot disk size	30.00 GiB
InstanceType	u1.medium	Storage class	ocs-external-storage...
CPU   Memory	1 CPU   4 GiB Memory	Public SSH key	Not configured
		Dynamic SSH key injection	<input checked="" type="checkbox"/>

Start this VirtualMachine after creation

Create VirtualMachine **Customize VirtualMachine** Cancel View YAML & CLI

- 6.4. Go to the Initial run tab, locate the **Cloud-init** section, and click **Edit**.

**Initial run**

**Cloud-init** [Edit](#)

You can use cloud-init to initialize the operating system with a specific configuration when the VirtualMachine is started.  
The cloud-init service is enabled by default in Fedora and RHEL images. [Learn more](#)

Scheduling	User	Password	Network data
SSH	developer	*****	Default

**Initial run**

**Sysprep** [Edit](#)

Windows only

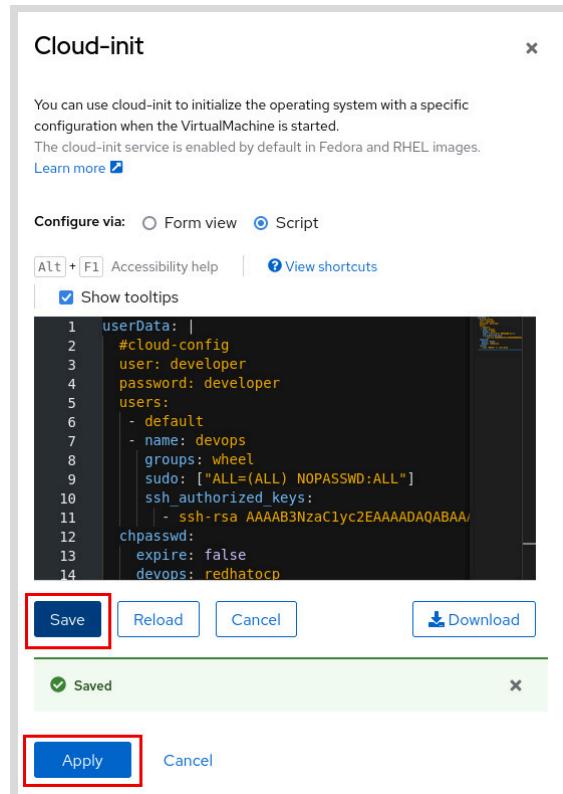
Sysprep is an automation tool for Windows that automates Windows setup, and custom software provisioning. An answer file is an XML-based file that contains setting definitions and values to use during Windows Setup  
[Learn more](#)

Autounattend.xml answer file	Unattend.xml answer file
Not available	Not available

Start this VirtualMachine after creation

[Create VirtualMachine](#) [Cancel](#)

- 6.5. Set the username as developer with developer as the password and then change to the Script view. Add a devops user and update the /etc/motd file. You can copy the configuration script from the ~/D0316/labs/replicating-using/cloud-config file. Press Shift+Insert to paste the contents of the file into the editor.



You might need to scroll down in the `cloud-init` window to see the **Save** button.

6.6. After saving the script, click **Create VirtualMachine**.

► 7. Access the `web1` VM to confirm the custom configuration.

7.1. From the command line, connect to the console of the `web1` VM by using the `virtctl console` command. Press `Enter` to open the login prompt.

```
[student@workstation ~]$ virtctl console web1
Successfully connected to web1 console. The escape sequence is ^]

...output omitted...

web1 login:
```

7.2. Log in as the `developer` user with `developer` as the password. The MOTD is displayed before the prompt.

```
web1 login: developer
Password: developer
Deployed by D0316 class.
[developer@web1 ~]$
```

7.3. Disconnect from the console with `Ctrl+]` and connect with `ssh` as the `devops` user.

```
[student@workstation ~]$ virtctl ssh -i .ssh/lab_rsa --username devops web1
Deployed by DO316 class.
...output omitted...
[devops@web1 ~]$
```

7.4. Exit the ssh session.

```
[devops@web1 ~]$ exit
logout
Connection to vmi/web1.replicating-using closed.
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish replicating-using
```

# Creating Custom Templates and Instance Types

---

## Objectives

- Generate and use custom templates and instance types to provision virtual machines.

## Creating Custom Templates

Virtual machine templates are a Red Hat OpenShift **template** resource. The template includes a list of parameterized resources, and must be processed to generate a resource definition that can be created.

Red Hat OpenShift templates for container applications include a list of objects to create, such as deployments, services, routes, and PVCs. A virtual machine template defines a **virtualmachine** resource.

By default, you cannot access custom templates within a specific project from other projects. The templates from Red Hat OpenShift and the OpenShift Virtualization operator are in a namespace that is available to all users. Use the `oc get templates -n openshift` command to view a list of templates that are available to all projects.

Although the templates from OpenShift Virtualization cannot be modified, they can be cloned. Go to **Virtualization > Templates**, select the template to clone, and select **Actions > Clone**.

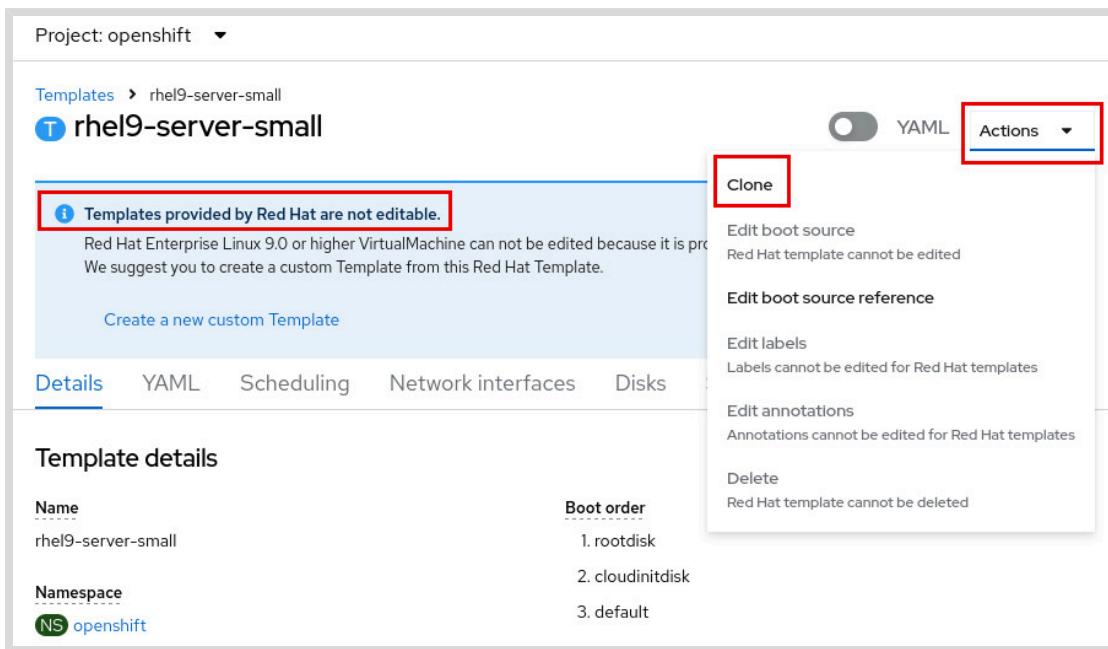


Figure 7.23: Clone a template

The **Clone template** form prompts for the new template's name, project, display name, and provider. After a template is created, you can modify other attributes with the forms in the web console.

To create a template, go to **Virtualization > Templates** and click **Create Template**. Customize the parameters in the YAML editor and click **Create**.

The YAML resource file can also be created on a local system. Create the template by using the `oc apply` command.

## YAML Template Definitions

The YAML for a template resource includes a `metadata` section with information about the template resource, an `objects` section where the VM resource is defined, and a `parameters` section that defines the variables that are used in the template.

The metadata of a template resource includes the name and namespace of the template. The labels and annotations describe the use of the template.

```
apiVersion: template.openshift.io/v1
kind: Template 1
metadata:
  name: my-custom-template 2
  namespace: replicating-create 3
  labels:
    os.template.kubevirt.io/rhel9: "true"
    template.kubevirt.io/type: vm
    workload.template.kubevirt.io/server: "true"
  annotations:
    description: VM template example
    iconClass: icon-rhel
    ...output omitted...
```

- 1** Resource kind
- 2** Template name
- 3** Project namespace

The `objects` section is where most of the parameters are used.

```
objects:
- apiVersion: kubevirt.io/v1
  kind: VirtualMachine 1
  metadata:
    labels:
      app: ${NAME} 2
      kubevirt.io/dynamic-credentials-support: "true"
      vm.kubevirt.io/template: replicating-create
      vm.kubevirt.io/template.namespace: replicating-create
      vm.kubevirt.io/template.revision: "1"
    name: ${NAME} 3
  spec:
    ...output omitted...
```

- 1** Create a VM object.
- 2** The NAME variable is expanded when the template is processed.

- ③ A variable can be used many times.

A complete definition for a `VirtualMachine` resource is included. Fields that are configured when creating a VM from the template show initial values that are defined in the template. Values that must be identical in multiple places can be defined in parameters.

```
spec:
  dataVolumeTemplates:
    - apiVersion: cdi.kubevirt.io/v1beta1
      kind: DataVolume
      metadata:
        name: ${NAME} ①
      spec:
        storage:
          resources:
            requests:
              storage: 30Gi
        storageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
        source: ②
          http:
            url: "https://example.com/openshift4/images/rhel9-mariadb.qcow2"
        running: false
      template:
        metadata:
          ...output omitted...
        spec:
          architecture: amd64
        ...output omitted...
        volumes:
          - dataVolume:
              name: ${NAME} ③
              name: roottdisk
            - cloudInitNoCloud:
                userData: |-  

                  #cloud-config  

                  user: cloud-init  

                  password: ${CLOUD_USER_PASSWORD} ④  

                  chpasswd: { expire: False }  

                name: cloudinitdisk
```

- ① The `dataVolume` name is set from a parameter.
- ② When a data source exists, the template shows *Source Available*.
- ③ The `dataVolume` name must match in the storage volume section.
- ④ Passwords are often generated.

The final section of the template resource defines the parameters. Parameters can be required or optional, can have a default value, or can be generated from an expression.

```
parameters:
  - description: VM name
    from: rhel9-[a-z0-9]{16}
    generate: expression
```

```

name: NAME
- name: CLOUD_USER_PASSWORD ①
  description: Randomized password for the cloud-init user cloud-user
  from: '[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}' ②
  generate: expression
...output omitted...

```

- ① Name of the variable
- ② Generated from a regular expression

## Customizable Template Parameters

When you create custom templates, many parameters provide the appropriate information, definitions, and supporting descriptions. The web console provides forms for customizing the template.

Go to **Virtualization > Templates** and select the custom template to edit. Review each category to view the default selections and the customizable options.

Name	Source	Size	Drive	Interface	Storage cla...
cloudinitdisk	Other	-	Disk	virtio	-
rootdisk	URL	30 GiB	Disk	virtio	ocs-external-storagecluster-ceph-rbd-virtualization

**Figure 7.24: Edit a template**

These categories store the template definitions that define cluster resources:

Field	Description
Details	Overview of the template definition.
YAML	YAML view of the template definition.
Scheduling	Add, edit, or remove node selectors, tolerations, affinity rules, dedicated resources, and eviction strategies.
Network interfaces	Add, edit, or remove network interfaces.

Field	Description
Disks	Add, edit, or remove disks.
Scripts	Configurations for <code>cloud-init</code> , <code>sysprep</code> , and SSH access.
Parameters	Add, edit, or remove additional parameters.

## Creating a VM from a Custom Template

After you create a custom template, it is listed among the available virtual machine templates. From the web console, go to **Virtualization > Catalog** and browse through the list of templates. To refine your search, use the search bar or the provided filters.

## Creating Custom Instance Types

To view or edit the provided instance types, go to **Virtualization > InstanceTypes**. Select an instance type and then use the **Action** menu for editing labels or annotations. To edit the instance type spec section, click the **YAML** tab. Modify the attributes and click **Save**.

An instance type can also be cloned. Go to **Virtualization > InstanceTypes** and click the vertical ellipsis for the instance type to clone.

Name	CPU	Memory	Vendor	Actions
VMCI cx1.2xlarge	7	16 GiB	redhat.com	⋮ Clone
VMCI cx1.4xlarge	16	32 GiB	redhat.com	⋮ Delete
VMCI cx1.8xlarge	32	64 GiB	redhat.com	⋮ Delete
VMCI cx1.large	2	4 GiB	redhat.com	⋮

Figure 7.25: Clone an instance type

Enter the name for the new instance and click **Save**.

You can also create instance types with the `virtctl` command. By default, a `virtualmachineclusterinstances` resource is created. Use the `--namespaced` option to create a `virtualmachineinstances` resource that is available only in the current namespace. The `virtctl create instancetype --help` command shows all the options.

The following command displays the YAML output for an instance type with the `example-gpu` name and a specified GPU:

```
[user@host ~]$ virtctl createinstancetype --name example-gpu \
--cpu 2 --memory 256Mi --gpu name:gpu1,devicename:nvidia
apiVersion:instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterInstancetype
metadata:
  creationTimestamp: null
  name: example-gpu
spec:
  cpu:
    guest: 2
  gpus:
    - deviceName: nvidia
      name: gpu1
  memory:
    guest: 256Mi
```

Save the output to a file and use the `oc apply -f filename` command to create the instance type. You can also pipe the output to the `oc` command as in the following example:

```
[user@host ~]$ virtctl createinstancetype --name example --cpu 2 \
--memory 256Mi | oc apply -f -
virtualmachineclusterinstancetype.instancetype.kubevirt.io/example created
```



### Note

The `-f` option, when it is used with the `oc apply` command, expects a file name and path. The `-` character represents the standard input device as the file.

## Creating Custom Virtual Machine Preferences

To view or edit the provided preferences, go to **Virtualization > Preferences**. You can edit or clone the preference resources by using the same procedure as for instance types.

You can create preferences with the `virtctl` command. By default, a `virtualmachineclusterpreferences` resource is created. Use the `--namespaced` option to create a `virtualmachinepreferences` resource that is available only in the current namespace. The `virtctl create preference --help` command shows all the options.

Unlike instance types, the `virtctl` command does not offer options for all the preference attributes.

The following command displays the YAML output for a preference resource with the `example-preference` name and a specified storage class:

```
[user@host ~]$ virtctl create preference --name example-preference \
--volume-storage-class ocs-external-storagecluster-ceph-rbd-virtualization
apiVersion:instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterPreference
metadata:
  creationTimestamp: null
  name: example-preference
```

```
spec:
  volumes:
    preferredStorageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
```

You can use the `virtctl` command to generate YAML output for a preference resource. This command generates only part of the required resource YAML output. You then use one of the provided resources as a sample for the additional attributes. Save the output to a file, modify the file to add attributes, and then use the `oc apply -f filename` command to create the preference.

## Adding Bootable Volumes

By default, the virtualization operator configures the HyperConverged resource with the `enableCommonBootImageImport` feature gate enabled. This feature configures automatic updates for all system-defined boot sources. The installation creates `imagestream`, `datasource`, and `dataimportcron` resources in the `openshift-virtualization-os-images` namespace.

You can configure the HyperConverged instance to automatically update additional boot sources. A `DataImportCronTemplates` section is added to the `spec` of the `hco` resource. This section contains a list of `DataImportCron` resource specifications. For more information, refer to the *Enabling Automatic Updates for Custom Boot Sources* section of the OpenShift Virtualization documentation.

The system-defined resources import images from the `registry.redhat.io` and `quay.io` registries and cannot be modified. For disconnected clusters, Red Hat recommends disabling automatic updates. Leaving automatic updates enabled uses resources and fills log files with each attempt to import the images.

When the `enableCommonBootImageImport` feature gate is disabled, `datasource` objects are reset so that they no longer point to the original boot source and any system default `DataImportCron` resources are deleted. An administrator can manually provide a boot source by creating a new PVC or volume snapshot for the `datasource` object and then populating it with an operating system image.

For the custom image to be detected as an available boot source, the name of the image's `managedDataSource` must match the name of the template's `DataSource`, which is defined in `spec.dataVolumeTemplates.spec.sourceRef.name` in the VM template YAML file.

To add custom bootable volumes in the web console, go to **Virtualization > Bootable volumes** and select **Add volume > With form**. Parameters for adding a volume include the following ones:

### Source type

You can upload an image from your local system, clone an existing PVC or volume snapshot, or download an image from a container registry. Downloading from a registry creates a `DataImportCron` resource, which defines a `cron` job to poll and import the disk image.

### Source details

When you upload an image, you can drag and drop a local RAW or QCOW image file. Cloning requires the source project and the PVC or snapshot name. The registry URL and credentials are required for polling and importing an image.

### Destination details

The volume name and disk size are required. You can update the storage class and the destination project name. You can also choose to apply optimized storage profile settings or to set those fields independently.

### Volume metadata

The bootable volume must be associated with an existing preference. You can also set a default instance type and specify a description of the volume.

## Creating a VM from a Custom Bootable Volume and an Instance Type

Go to **Virtualization > Catalog** and select the project for the virtual machine. Go to the **InstanceTypes** tab and select the boot volume. Then, scroll down to choose an instance type. The default instance type that is specified when uploading the boot volume is already selected.

In the **VirtualMachine Details** section, accept or update the **Name** and **Storage class** fields. Then, click either **Create VirtualMachine** or **Customize VirtualMachine**.



### References

For more information about templates, refer to the *Creating Virtual Machine from Templates* section in the OpenShift Virtualization guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-creating-vms-from-templates](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-creating-vms-from-templates)

For more information about instance types, refer to the *Creating Virtual Machines from Instance Types* section in the OpenShift Virtualization guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-creating-vms-from-instance-types](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-creating-vms-from-instance-types)

For more information about managing boot sources, refer to the **Storage** chapter in the OpenShift Virtualization guide at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#storage](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#storage)

## ► Guided Exercise

# Create a Custom Template and an Instance Type

Create a custom template and an instance type for an image that is based on Red Hat Enterprise Linux, and then create virtual machines from the custom resources.

### Outcomes

- Create a template with the `mariadb-server` image by defining the attributes.
- Create a VM from the custom template and access it to confirm that the MariaDB service is up and running.
- Create an instance type by defining the attributes.
- Create a custom preferences resource by defining the attributes.
- Add a bootable volume with the `mariadb-server` image.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the `replicating-create` namespace and ensures that all resources are available for this exercise.

```
[student@workstation ~]$ lab start replicating-create
```

### Instructions

- 1. As the `admin` user, clone a template to the `replicating-create` namespace and name it `replicating-create`.
- 1.1. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
  - 1.2. Go to `Virtualization > Templates`. Select `All Projects` from the `Projects` list and search for `rhel9-server-small`. At the right side of that entry, click the vertical ellipsis icon : and select `Clone`.

- 1.3. In the **Clone template** form, update the fields with the following information and then click **Clone**.

Field	Value
Template name	replicating-create
Template project	replicating-create
Template display name	Red Hat Enterprise Linux 9 VM MariaDB
Template provider	Leave blank

- 2. Customize the **replicating-create** template to update the eviction strategy and the boot source.
- 2.1. Go to **Virtualization > Templates**. If necessary, select the **replicating-create** project from the **Projects** list and select the **replicating-create** template.
  - 2.2. Select the **Scheduling** tab and click the pencil icon under **Eviction strategy**. Confirm that the checkbox for **LiveMigrate** is selected, and then click **Save**.

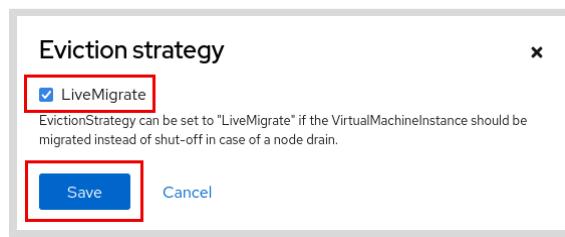


Figure 7.27: Eviction strategy for the VM

- 2.3. Select the **Disk** tab and detach the **rootdisk** disk. Click the vertical ellipsis icon at the right side of the **rootdisk** line, and then click **Detach**. Click **Detach** to confirm.

The screenshot shows the 'Disks' tab of a virtual machine configuration. A table lists two disks: 'cloudinitdisk' and 'rootdisk'. The 'rootdisk' row has a red box around its three-dot menu icon. A dropdown menu is open, with 'Detach' highlighted and also enclosed in a red box.

Name	Source	Size	Drive	Interface	Storage cl...
cloudinitdisk	Other	-	Disk	virtio	-
rootdisk	PVC (auto import)	30 GiB	Disk	virtio	-

Figure 7.28: Detach disk

- 2.4. Create a **rootdisk** disk. Click **Add disk**. Complete the form by using the following information and then click **Save**.

Field	Value
Use this disk as a boot source	<i>Checked</i>
Name	<b>rootdisk</b>
Source	From URL
URL	<a href="http://utility.lab.example.com:8080/openshift4/images/rhel9-mariadb.qcow2">http://utility.lab.example.com:8080/openshift4/images/rhel9-mariadb.qcow2</a>
StorageClass	<b>ocs-external-storagecluster-ceph-rbd-virtualization</b>

Leave the other form fields with their default values.

- 2.5. Review the storage settings to confirm that the **rootdisk** volume storage size is **30 GiB** and that it uses the **virtio** interface and the **ocs-external-storagecluster-ceph-rbd-virtualization** storage class.

Name	Source	Size	Drive	Interface	Storage cl...
cloudinitdisk	Other	-	Disk	virtio	-
rootdisk	URL	30 GiB	Disk	virtio	ocs-external-storagecluster-ceph-rbd-virtualization

Figure 7.29: VM disks

- 2.6. Use the YAML editor to set the name of the `rootdisk` data volume to the name of the VM that the template creates. Go to the `YAML` tab and update the following object values to use the `'${NAME}'` template variable:

- `objects.spec.dataVolumeTemplates[0].metadata.name`
- `objects.spec.template.spec.volumes[1].dataVolume.name`

Click **Save** when done.

```
apiVersion: template.openshift.io/v1
kind: Template
...output omitted...
objects:
- apiVersion: kubevirt.io/v1
  kind: VirtualMachine
  ...output omitted...
spec:
  dataVolumeTemplates:
    - metadata:
        name: '${NAME}' ①
    ...output omitted...
  template:
    ...output omitted...
  spec:
    ...output omitted...
    volumes:
      ...output omitted...
      - name: rootdisk
        dataVolume:
          name: '${NAME}' ②
  evictionStrategy: LiveMigrate
parameters:
  ...output omitted...
```

- ①** Set the `objects.spec.dataVolumeTemplates[0].metadata.name` field to `'${NAME}'`.

- ② Set the `objects.spec.template.spec.volumes[1].datavolume.name` field to `'${NAME}'`.

**Note**

You can use the **Download** button to save a copy of the modified template, and then compare it with the sample in the `~/D0316/labs/replicating-create/template-replicating-create.yaml` file.

```
[student@workstation ~]$ diff \
Downloads/template-replicating-create.yaml \
D0316/labs/replicating-create/template-replicating-create.yaml
```

The UIDs, resource versions, and time values can be different between the solution and the edited YAML.

- 2.7. Configure Cloud-init to create the developer user with developer as the password. Select the **Scripts** tab and click **Edit** next to **Cloud-init**. Complete the form according to the following specification and click **Apply**.

Field	Value
User	developer
Password	developer
Add network data	Unchecked

- ▶ 3. Create a VM from the replicating-create template.

- 3.1. Go to **Virtualization > Catalog** and select **Project > replicating-create** and the **Template catalog** tab.
- 3.2. On the right panel of filter options, under **Template project**, select the **replicating-create** project. Click **User templates** and select the newly created template **Red Hat Enterprise Linux 9 VM MariaDB**.

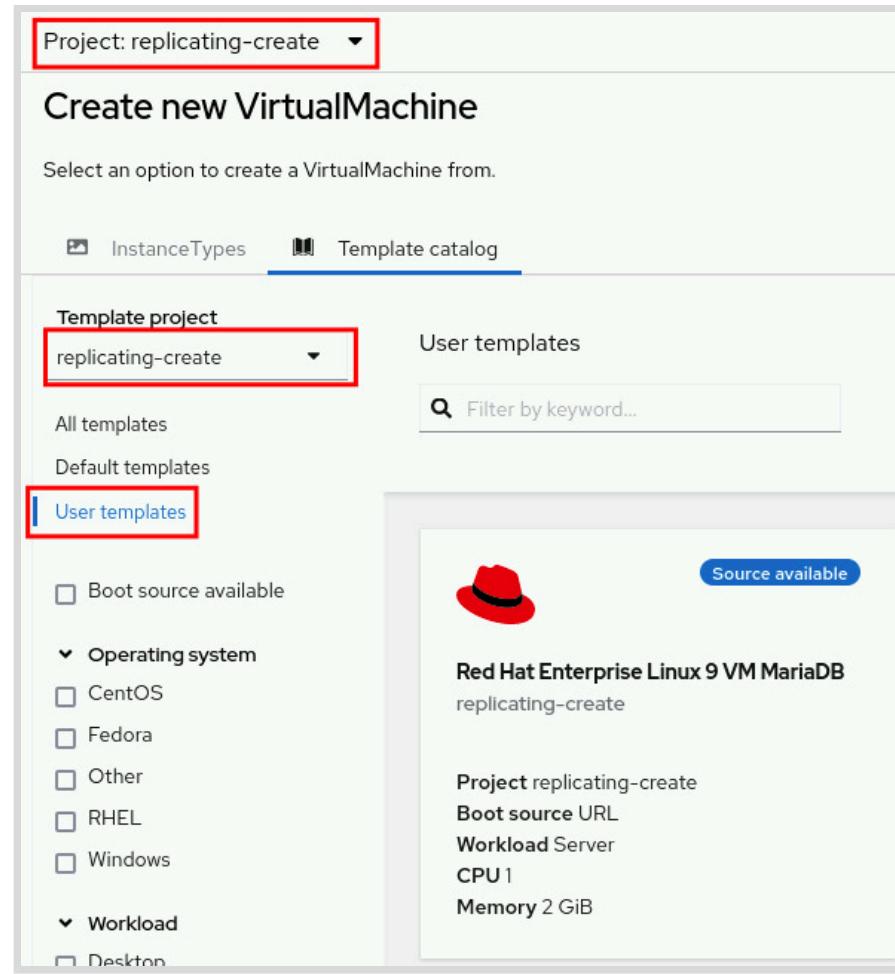


Figure 7.30: Use template

- 3.3. Change the VirtualMachine name to replicating-create-vm and click Quick create VirtualMachine.

The screenshot shows the 'Quick create VirtualMachine' dialog. It has fields for 'VirtualMachine name \*' (containing 'replicating-create-vm'), 'Project' (set to 'replicating-create'), and 'Public SSH key' (labeled 'Not configured'). There's a checkbox for 'Start this VirtualMachine after creation' which is checked. At the bottom, there are two buttons: 'Quick create VirtualMachine' and 'Customize VirtualMachine', with 'Quick create VirtualMachine' being highlighted with a red box.

Figure 7.31: Name the VM

- 3.4. Wait for the replicating-create-vm machine to provision and to display the Running status. The Provisioning phase can take several minutes to complete.

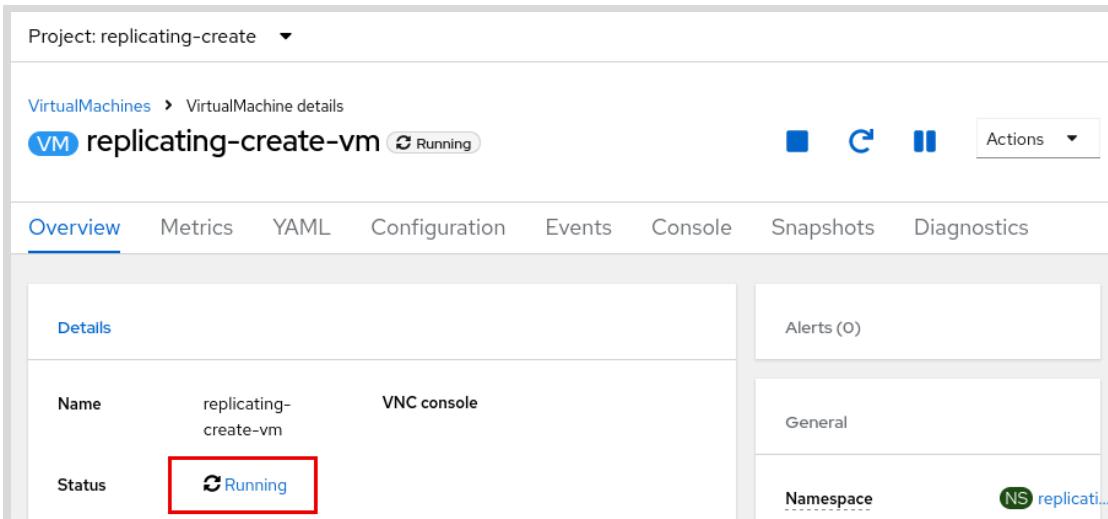


Figure 7.32: Virtual machine status

► 4. Access the console of the replicating-create-vm machine.

- 4.1. Go to the **Console** tab. Wait for the console to display the login prompt.

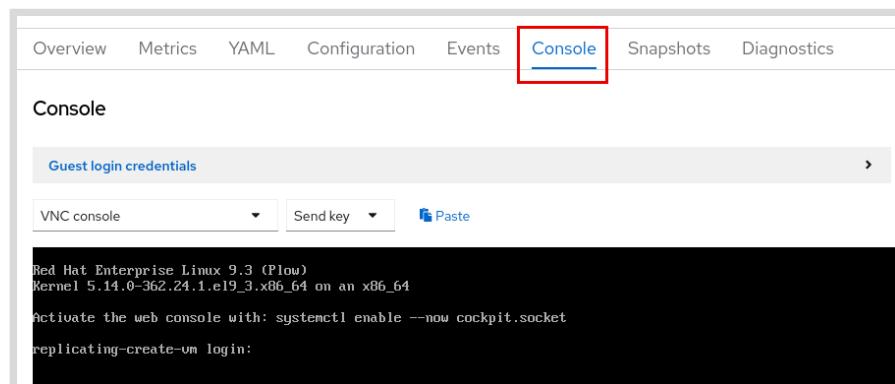


Figure 7.33: Virtual machine console

- 4.2. Use the console to log in as the developer user with developer as the password.

```
replicating-create-vm login: developer
Password: developer
[developer@replicating-create-vm ~]$
```

- 4.3. Confirm that the preconfigured database, sakila, is available. Use the mysql command to log in as devuser with developer as the password.

```
[developer@replicating-create-vm ~]$ mysql -u devuser -p'developer' \
sakila
Welcome to the MariaDB monitor. Command end with ; or \g.
...output omitted...

MariaDB [sakila]>
```

- 4.4. Exit the sakila database and the replicating-create-vm machine.

```
MariaDB [sakila]> exit  
Bye  
[developer@replicating-create-vm ~]$ exit
```

- ▶ 5. Delete the replicating-create-vm machine and the replicating-create template.
  - 5.1. Go to **Virtualization > VirtualMachines**.
  - 5.2. Click the vertical ellipsis icon to the right of the replicating-create-vm VM and select **Delete**. Ensure that **Delete disks** is selected. Click **Delete** in the confirmation window.
  - 5.3. Go to **Virtualization > Templates**.
  - 5.4. Click the vertical ellipsis icon to the right of the replicating-create template row and click **Delete Template**. Click **Delete** in the confirmation window.
- ▶ 6. Create a custom instance type.
  - 6.1. From the command line, log in to your Red Hat OpenShift cluster as the **admin** user with **redhatocp** as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \  
https://api.ocp4.example.com:6443
```

- 6.2. Use the **virtctl** command to generate the YAML output for an instance type resource.

```
[student@workstation ~]$ virtctl create instancetype \  
--name replicating-create --cpu 1 --memory 3Gi  
apiVersion: instancetype.kubevirt.io/v1beta1  
kind: VirtualMachineClusterInstancetype  
metadata:  
  creationTimestamp: null  
  name: replicating-create  
spec:  
  cpu:  
    guest: 1  
  memory:  
    guest: 3Gi
```

- 6.3. Create the instance type.

```
[student@workstation ~]$ virtctl create instancetype \  
--name replicating-create --cpu 1 --memory 3Gi | oc apply -f -  
virtualmachineclusterinstancetype.instancetype.kubevirt.io/replicating-create  
created
```

- ▶ 7. Create a custom preferences
  - 7.1. Use the **virtctl** command to generate the YAML output for a virtual machine cluster preference resource.

```
[student@workstation ~]$ virtctl create preference --name replicating-create \
--volume-storage-class ocs-external-storagecluster-ceph-rbd-virtualization
apiVersion: instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterPreference
metadata:
  creationTimestamp: null
  name: replicating-create
spec:
  volumes:
    preferredStorageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
```

- 7.2. Not all the spec options can be set with the `virtctl` command. Display a sample `vmcp.spec` section with the following command:

```
[student@workstation ~]$ oc get vmcp rhel.9 -o yaml | yq r - spec
...output omitted...
requirements:
  cpu:
    guest: 1
  memory:
    guest: 1536Mi
```

- 7.3. Edit the `~/D0316/labs/replicating-create/custom-preference.yaml` file to use the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class and to require 1CPU and 2Gi of memory.

```
apiVersion: instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterPreference
metadata:
  creationTimestamp: null
  name: replicating-create
spec:
  volumes:
    preferredStorageClassName: ocs-external-storagecluster-ceph-rbd-virtualization
  requirements:
    cpu:
      guest: 1
    memory:
      guest: 2Gi
```

- 7.4. Create the custom preference resource.

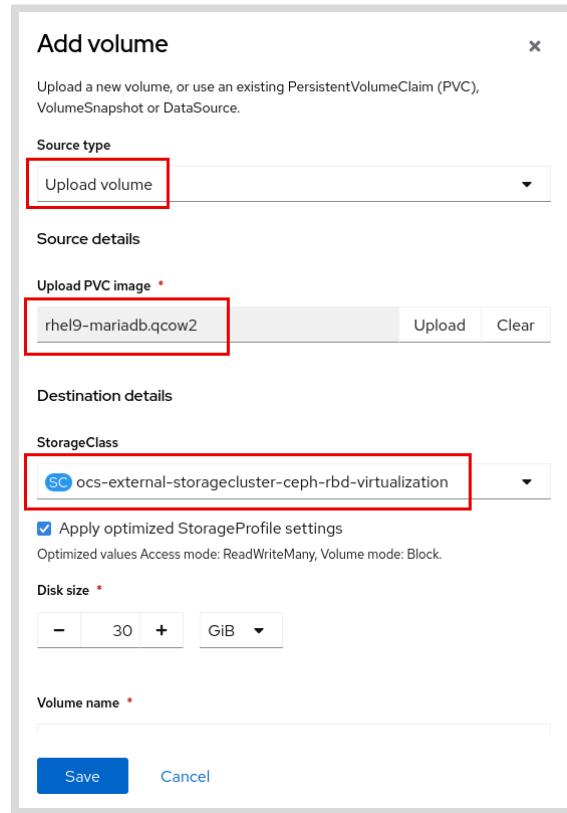
```
[student@workstation ~]$ oc apply -f \
~/D0316/labs/replicating-create/custom-preference.yaml
virtualmachineclusterpreference.instancetype.kubevirt.io/replicating-create
created
```

- 8. Add the `rhel9-mariadb.qcow2` image as a bootable volume for use with instance types.

- 8.1. Download the `rhel9-mariadb.qcow2` image.

```
[student@workstation ~]$ wget \
http://utility.lab.example.com:8080/openshift4/images/rhel9-mariadb.qcow2
```

- 8.2. Go to **Virtualization > Bootable volumes** in the web console. Click **Projects** and then click **Show default projects**. Select **Project > openshift-virtualization-os-images**.
- 8.3. Select **Add volume > With form** and click **Upload** in the source details section. Select the **rhel9-mariadb.qcow2** file that you downloaded in the previous step. Also set the storage class to **ocs-external-storagecluster-ceph-rbd-virtualization**.



- 8.4. Scroll down and complete the remaining destination and metadata fields according to the following table:

Field	Value
<b>Destination details</b>	
StorageClass	ocs-external-storagecluster-ceph-rbd-virtualization
Apply optimized StorageProfile settings	checked
Disk size	30 GiB
Volume name	rhel9-mariadb
Destination project	openshift-virtualization-os-images
<b>Volume metadata</b>	
Preference	replicating-create
Default InstanceType	User-provided, replicating-create

To set the default instance type, select **User provided > template-create**.

- 8.5. Verify the volume settings and then click **Save** to start the clone process.

The screenshot shows the 'Add volume' dialog box. It includes the following fields:

- Volume name:** rhel9-mariadb
- Destination project:** PR openshift-virtualization-os-images
- Preference:** VMCP replicating-create
- Default InstanceType:** VMCI replicating-create

The 'Save' button at the bottom left is highlighted with a red box.

It can take a few minutes for the image to upload and the datasource to be created.

The screenshot shows the 'Bootable volumes' section of the OpenShift Catalog. A red box highlights the search bar where 'mariadb' is typed. Another red box highlights the selected volume 'rhel9-mariadb' in the list, which has a 'replicating-create' preference.

▶ 9. Create a VM by using the custom bootable volume, instance type, and preference.

- 9.1. Go to Virtualization > Catalog and select Project > replicating-create and the InstanceTypes tab.
- 9.2. In the first section, click **rhel9-mariadb** to select the boot volume.
- 9.3. Scroll down and verify that the user-provided **replicating-create** instance type is selected.
- 9.4. In the **VirtualMachine Details** section, change the **Name** field to **instance-vm**. Verify that the **Storage class** is **ocs-external-storagecluster-ceph-rbd-virtualization** and then click **Create VirtualMachine**.

The screenshot shows the 'VirtualMachine details' dialog. The 'Name' field is set to 'instance-vm'. The 'Storage class' dropdown is set to 'ocs-external-storage...'. The 'Create VirtualMachine' button at the bottom left is highlighted with a red box.

Wait for the virtual machine status to show the **Running** state.

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish replicating-create
```

## ► Lab

# Replicating Virtual Machines by Using Instance Types, Templates, and Clones

Create a Red Hat Enterprise Linux template and instance type based on predefined images that contain a MariaDB server and an Apache HTTP server. Create a golden virtual machine image for the web server. You then create two virtual machines from the template and instance type and confirm that the VMs contain a database with preloaded data and a configured web server.

## Outcomes

- Create a custom template that uses the `web-server` image by defining the attributes.
- Create a VM with the `web-dev` name by using the custom template, and then add content before sealing the root disk.
- Create a VM with the `web1` name by cloning the `web-dev` VM.
- Create a bootable volume by using a custom instance type and the sealed image.
- Create a VM with the `www` name by using the custom instance, the custom bootable volume, and a `cloud-init` script.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the `replicating-review` namespace, and creates the `web-server` persistent volume claim in that namespace.

```
[student@workstation ~]$ lab start replicating-review
```

## Instructions

1. From the command line on the `workstation` machine, use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`. Open a web browser and log in to the OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`.
2. Clone the `rhel9-server-small` VM template to the `replicating-review` namespace, name it `web-server`, and use `RHEL9 Web Server` as the display name.  
Define a 15 GiB `rootdisk` volume that uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class with the `ReadWriteMany` access mode and the `Block` volume mode. Define the boot source of the `rootdisk` volume as the `web-server` PVC within the `replicating-review` namespace.

Attach an additional 5 GiB disk named `tmpdata`, which uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class with the `ReadWriteMany` access mode and the `Block` volume mode. Use template variables to set the name of the `rootdisk` data volume to match the name of the VM that the template creates.

Enable the `LiveMigrate` eviction strategy.

Configure the `cloud-init` service to create a `webadmin` user with `redhatocp` as the password on VMs that are created from the template.

3. Create a `web-dev` VM from the custom RHEL9 Web Server template in the `replicating-review` namespace.
4. Connect to the console of the `web-dev` VM. Log in as the `webadmin` user with `redhatocp` as the password, which you defined in a previous step. Confirm that the 5 GiB `tmpdata` disk is available as the `vdc` block device by using the `lsblk` command. Use the `curl localhost` command to confirm that the Apache HTTP web server responds to requests.
5. Update the contents of the `web-dev` machine, and then shut down the machine. Add an "Authorized Use Only" message to the `/etc/motd` file. Update the `httpd` server home page with the contents from the `~/D0316/labs/replicating-review/index.html` file.
6. Seal the `web-dev` rootdisk PVC and then clone the VM. Set the name of the cloned VM name to `web1`.
7. Create an instance type with the attributes from the following table:

Field	Value
name	<code>web-server-prod</code>
CPU	2
RAM	6 GiB

8. Modify the `datasource.yaml` resource file in the `~/D0316/labs/replicating-review/` directory to create a `web-prod` bootable volume with the sealed `web-dev` image as the source. Set the default `instancetype` to `web-server-prod` and the default preference to `rhel.9`.
9. Create the `www` VM by using the `web-prod` bootable image and the `web-server-prod` instance type. Use `cloud-init` to add a `devops` user with `redhatops` as the password. Add the `Lab-rsa` ssh key. The ssh key is already added to the project.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade replicating-review
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish replicating-review
```



## ► Solution

# Replicating Virtual Machines by Using Instance Types, Templates, and Clones

Create a Red Hat Enterprise Linux template and instance type based on predefined images that contain a MariaDB server and an Apache HTTP server. Create a golden virtual machine image for the web server. You then create two virtual machines from the template and instance type and confirm that the VMs contain a database with preloaded data and a configured web server.

## Outcomes

- Create a custom template that uses the `web-server` image by defining the attributes.
- Create a VM with the `web-dev` name by using the custom template, and then add content before sealing the root disk.
- Create a VM with the `web1` name by cloning the `web-dev` VM.
- Create a bootable volume by using a custom instance type and the sealed image.
- Create a VM with the `www` name by using the custom instance, the custom bootable volume, and a `cloud-init` script.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. It also creates the `replicating-review` namespace, and creates the `web-server` persistent volume claim in that namespace.

```
[student@workstation ~]$ lab start replicating-review
```

## Instructions

1. From the command line on the `workstation` machine, use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`. Open a web browser and log in to the OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`.
  - 1.1. From the command line, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Open a web browser and navigate to <https://console-openshift-console.apps.ocp4.example.com>. Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
2. Clone the `rhel9-server-small` VM template to the `replicating-review` namespace, name it `web-server`, and use `RHEL9 Web Server` as the display name.

Define a 15 GiB `rootdisk` volume that uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class with the `ReadWriteMany` access mode and the `Block` volume mode. Define the boot source of the `rootdisk` volume as the `web-server` PVC within the `replicating-review` namespace.

Attach an additional 5 GiB disk named `tmpdata`, which uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class with the `ReadWriteMany` access mode and the `Block` volume mode. Use template variables to set the name of the `rootdisk` data volume to match the name of the VM that the template creates.

Enable the `LiveMigrate` eviction strategy.

Configure the `cloud-init` service to create a `webadmin` user with `redhatocp` as the password on VMs that are created from the template.

- 2.1. Go to **Virtualization > Templates**. Select **All Projects** from the **Projects** list and search for `rhel9-server-small`. Click the vertical ellipsis icon : and select **Clone**.
- 2.2. In the **Clone template** form, change **Template name** to `web-server`. Change **Template project** to `replicating-review`. Change **Template display name** to `RHEL9 Web Server`. Leave the other form fields with their default values. Click **Clone**.
- 2.3. Go to **Virtualization > Templates**. Select the `replicating-review` project from the **Projects** list and select the `web-server` template.
- 2.4. Select the **Scheduling** tab and click the pencil icon under **Eviction strategy**. Confirm that the checkbox for `LiveMigrate` is selected, and then click **Save**.
- 2.5. Select the **Disks** tab and detach the `rootdisk` disk. Click the vertical ellipsis icon next to the `rootdisk` line, and then click **Detach**. Click **Detach** to confirm.
- 2.6. Click **Add disk** to create the `rootdisk` disk. Complete the form by using the following information. Leave the other form fields with their default values, and then click **Save**.

Field	Value
Use this disk as a boot source	<i>Selected</i>
Name	<code>rootdisk</code>
Source	Clone existing PVC
PVC project	<code>replicating-review</code>
PVC name	<code>web-server</code>
Persistent volume claim size	15 GiB
StorageClass	<code>ocs-external-storagecluster-ceph-rbd-virtualization</code>
Apply optimized StorageProfile settings	<i>Selected</i>

- 2.7. Click **Add disk** to create the `tmpdata` disk. Complete the form by using the following information. Leave the other form fields with their default values, and then click **Save**.

Field	Value
Use this disk as a boot source	<i>Deselected</i>
Name	<code>tmpdata</code>
Source	Empty disk (blank)
Persistent volume claim size	5 GiB
StorageClass	<code>ocs-external-storagecluster-ceph-rbd-virtualization</code>
Apply optimized StorageProfile settings	Enabled

- 2.8. Review the storage settings to confirm that the `rootdisk` volume storage size is bootable, that it has 15 GiB, and that it uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class. Review the storage settings to confirm that the `tmpdata` volume storage has 5 GiB and that it uses the `ocs-external-storagecluster-ceph-rbd-virtualization` storage class.
- 2.9. Use the YAML editor to set the name of the `rootdisk` data volume to the name of the VM that the template creates. Go to the **YAML** tab and update the following object values to use the ' `${NAME}` ' template variable. Click **Save** when done.
- `objects.spec.dataVolumeTemplates.metadata[0].name`
  - `objects.spec.template.spec.volumes.datavolume[0].name`
- 2.10. Configure `cloud-init` to create the `webadmin` user with `redhatocp` as the password. Select the **Scripts** tab and click **Edit** next to `Cloud-init`. In the **User** field,

enter **webadmin**, and in the **Password** field, enter **redhatocp**. Confirm that **Add network data** is deselected, and then click **Apply**.

3. Create a **web-dev** VM from the custom RHEL9 Web Server template in the replicating-review namespace.
  - 3.1. Go to **Virtualization > Catalog** and select the **Template catalog** tab. On the right panel of filter options, in **Template project**, select the replicating-review project and **All Templates**, and then select the newly created RHEL9 Web Server template.
  - 3.2. Enter **web-dev** in the **VirtualMachine name** field. Confirm that **Start this virtual machine after creation** is selected. Click **Quick create VirtualMachine**. Wait a few moments for the operation to complete.
4. Connect to the console of the **web-dev** VM. Log in as the **webadmin** user with **redhatocp** as the password, which you defined in a previous step. Confirm that the 5 GiB **tmpdata** disk is available as the **vdc** block device by using the **lsblk** command. Use the **curl localhost** command to confirm that the Apache HTTP web server responds to requests.
  - 4.1. From the command line, change to the replicating-review project.

```
[student@workstation ~]$ oc project replicating-review
```

- 4.2. Use the **virtctl** command to connect to the console of the **web-dev** guest.

```
[student@workstation ~]$ virtctl console web-dev
```

- 4.3. In the console window, log in as the **webadmin** user with **redhatocp** as the password.

```
web-dev login: webadmin
password: redhatocp
[webadmin@web-dev ~]$
```

- 4.4. Confirm that the **tmpdata** disk is available as the **vdc** block device.

```
[webadmin@web-dev ~]$ lsblk
NAME  MAJ:MIN RM  SIZE RO TYPE ...
vda   252:0    0  15G  0 disk
...output omitted...
vdb   252:16   0    1M  0 disk
vdc   252:32   0    5G  0 disk
```

- 4.5. Use the **curl** command to confirm that the web server responds to requests. The command returns some HTML output.

```
[webadmin@web-dev ~]$ curl localhost
...output omitted...
<title>Hello, World!</title>
...output omitted...
```

5. Update the contents of the **web-dev** machine, and then shut down the machine. Add an "Authorized Use Only" message to the **/etc/motd** file. Update the **httpd** server home page with the contents from the **~/D0316/labs/replicating-review/index.html** file.

- 5.1. Log in to the web-dev console as the webadmin user with redhatocp as the password, and use the sudo command to become the root user.

```
[webadmin@web-dev ~]$ sudo -i
```

- 5.2. Update the /etc/motd file.

```
[root@web-dev ~]# echo "Authorized Use Only" > /etc/motd
```

- 5.3. Use a text editor such as vi to update the /var/www/html/index.html file.

```
...output omitted...
<h2>OpenShift Virtualization Demo.</h2>
<p>Hello, World! Welcome to Red Hat Training.</p>
...output omitted...
```

- 5.4. Log out of the VM and then disconnect from the console with Ctrl+]

```
[root@web-dev ~]# exit
[webadmin@web-dev ~]$ logout
```

- 5.5. Use the virtctl command to stop the VM.

```
[student@workstation ~]$ virtctl stop web-dev
```

- 5.6. Wait a few moments and confirm that the VM is stopped.

```
[student@workstation ~]$ oc get vm web-dev
NAME      AGE     STATUS    READY
web-dev   10m    Stopped   False
```

6. Seal the web-dev rootdisk PVC and then clone the VM. Set the name of the cloned VM name to web1.

- 6.1. Start a container with the libguestfs tools that is connected to the web-dev PVC.

```
[student@workstation ~]$ virtctl guestfs web-dev
Use image: registry.redhat.io/container-native-virtualization/libguestfs-tools...
The PVC has been mounted at /dev/vda
Waiting for container libguestfs still in pending, reason: ContainerCreating,
message:
If you don't see a command prompt, try pressing enter.
bash-5.1$
```

- 6.2. Remove the system-specific information from the web-dev image and change the hostname.

```
bash-5.1$ virt-sysprep -a /dev/vda --hostname localhost.localdomain
[ 0.0] Examining the guest ...
[ 5.5] Performing "abrt-data" ...
...output omitted...
[ 6.9] Setting the machine ID in /etc/machine-id
[ 6.9] Setting the hostname: localhost.localdomain
[ 6.9] SELinux relabelling
[ 28.5] Performing "lvm-uuids" ...
```

- 6.3. Exit the container.

```
bash-5.1$ exit
[student@workstation ~]$
```

- 6.4. Clone the web-dev VM to a new VM named web1. Go to **Virtualization > VirtualMachines** and select the web-dev VM. Click **Actions > Clone**. Set the VM name to web1, select **Start VirtualMachine once created**, and then click **Clone**.
- 6.5. Go to **Virtualization > VirtualMachines** and confirm that the new web1 VM is running.
- 6.6. Select the **Console** tab and log in to the console as the webadmin user with redhatocp as the password.
- 6.7. Use the curl command to confirm that the web server responds to requests. The command returns the updated HTML content.

```
[webadmin@web1 ~]$ curl localhost
...output omitted...
<h2>OpenShift Virtualization Demo.</h2>
<p>Hello, World! Welcome to Red Hat Training.</p>
...output omitted...
```

7. Create an instance type with the attributes from the following table:

Field	Value
name	web-server-prod
CPU	2
RAM	6 GiB

- 7.1. From the command line, use the virtctl command to generate the YAML output for an instance type resource.

```
[student@workstation ~]$ virtctl create instancetype --name \
web-server-prod --cpu 2 --memory 6Gi | tee ~/instancetype.yaml
apiVersion:instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterInstancetype
metadata:
  creationTimestamp: null
  name: replicating-create
```

```
spec:
  cpu:
    guest: 2
  memory:
    guest: 6Gi
```

7.2. Create the instance type.

```
[student@workstation ~]$ oc apply -f ~/instancetype.yaml
virtualmachineclusterinstancetypeinstancetype.kubevirt.io/web-server-prod
```

8. Modify the `datasource.yaml` resource file in the `~/D0316/labs/replicating-review/` directory to create a `web-prod` bootable volume with the sealed `web-dev` image as the source. Set the default `instancetype` to `web-server-prod` and the default preference to `rhel.9`.
  - 8.1. Update the `datasource.yaml` resource file in the `~/D0316/labs/replicating-review/` directory.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataSource
metadata:
  labels:
   instancetype.kubevirt.io/default-instanceType: web-server-prod
   instancetype.kubevirt.io/default-preference: rhel.9
  name: web-prod
  namespace: replicating-review
spec:
  source:
    pvc:
      name: web-dev
      namespace: replicating-review
```

8.2. Create the `web-prod` data source.

```
[student@workstation ~]$ oc apply \
-f ~/D0316/labs/replicating-review/datasource.yaml
datasource.cdi.kubevirt.io/web-prod created
```

9. Create the `www` VM by using the `web-prod` bootable image and the `web-server-prod` instance type. Use `cloud-init` to add a `devops` user with `redhatops` as the password. Add the `lab-rsa` ssh key. The ssh key is already added to the project.
  - 9.1. Go to **Virtualization > Catalog** and select the `replicating-review` project. Select the `web-prod` bootable volume and the user defined `web-server-prod` instance type. Set the name of the VM to `www` and click **Customize VirtualMachine**.
  - 9.2. Go to **Initial run** and click **Edit** next to `Cloud-init`. Set user to `devops` and use `redhatops` as the password. Click **Apply** to save the changes.
  - 9.3. Go to **SSH** and click **Edit** next to `Public SSH key`. Click **Use existing** and select the `lab-rsa` public key in the `replicating-review` project. Click **Save**.
  - 9.4. Click **Create VirtualMachine** and wait until the VM status is `Running`.

9.5. Use the `virtctl ssh` command to connect to the machine as the `devops` user.

```
[student@workstation ~]$ virtctl ssh -i .ssh/lab_rsa devops@www -n replicating-review
Warning: Permanently added 'vmi/www.replicating-review' (ED25519) to the list of known hosts.
Authorized Use Only

[devops@www ~]$
```

9.6. Test the web service content by using the `curl` command.

```
[devops@www ~]$ curl localhost
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <h2>OpenShift Virtualization Demo.</h2>
    <p>Hello, World! Welcome to Red Hat Training.</p>
  </body>
</html>
```

9.7. Log out of the ssh session to return to `workstation`.

```
[devops@www ~]$ logout
Connection to vmi/www.replicating-review closed.
```

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade replicating-review
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish replicating-review
```

# Summary

---

- Clones are useful to deploy identical VMs from a model.
- You use the `sysprep.exe` command on Microsoft Windows systems to seal the VM before cloning. For Linux systems, you use the `virtctl guestfs` command and then the `virt-sysprep` command.
- You use an `autounattend.xml` file to automate configuring the first boot of a Microsoft Windows VM. You use `cloud-init` scripts to automate configuring the first boot of a Linux guest.
- Red Hat OpenShift Virtualization provides virtual machine templates, instance types, and preferences that are preconfigured and customizable.
- When creating a custom template, you can configure default parameters for the template, such as the boot source, workload type, CPU and memory, storage, network interfaces, and post-installation tasks. You can also provide a description of the template and include a message about how to use the template.
- To view the list of templates from the command line, use the `oc get templates -n openshift` command.
- When creating a custom instance type, you configure the CPU and memory parameters. You can also provide a description of the instance type, set default parameters for hardware devices, and set scheduling parameters.
- To view the list of cluster-scoped instance types from the command line, use the `oc get vmcf` command. Use the `oc get vmcp` command to list cluster-scoped preferences.
- Containerized Data Importer provides the `DataVolume` Custom Resource Definition to orchestrate the import, clone, and upload operations of PVCs for your VMs.
- With OpenShift Virtualization, you can create a VM from a data volume, including cloning an existing PVC as a new data volume or uploading a local disk image.
- Bootable volumes that are used with instance types are defined by using the `DataSource` custom resource definition.



## Chapter 8

# Advanced Virtual Machine Management

### Goal

Import, control the placement of, monitor the health of, and live migrate virtual machines.

### Sections

- Migrating Virtual Machines from Foreign Hypervisors (and Guided Exercise)
- Controlling the Scheduling of Virtual Machines (and Guided Exercise)
- Configuring Health Probes for Virtual Machines (and Guided Exercise)
- Live Migrating a Virtual Machine (and Guided Exercise)

### Lab

- Advanced Virtual Machine Management

# Migrating Virtual Machines from Foreign Hypervisors

## Objectives

- Migrate virtual machines from VMware vSphere, Red Hat Virtualization, OpenStack, or another OpenShift Virtualization cluster.

## The Migration Toolkit for Virtualization

With the migration toolkit for virtualization (MTV), you can migrate multiple VMs from a compatible hypervisor, such as VMware vSphere or Red Hat Virtualization, to OpenShift Virtualization.

You can install the MTV with the MTV operator by using either the OpenShift web console or the command line. Consult the references section for installing an operator by using the OperatorHub or the `oc` command. The following screenshots show the process of installing the MTV operator from the web console:



Figure 8.1: Installing the MTV operator

The MTV operator is installed in the `openshift-mtv` namespace.

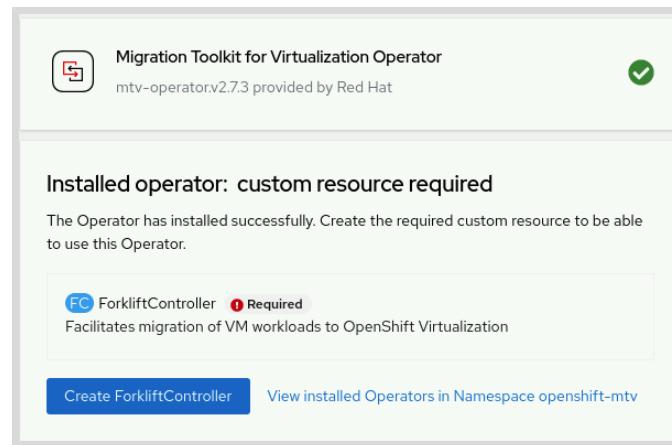
The screenshot shows the 'OperatorHub > Operator Installation' section titled 'Install Operator'. It instructs the user to 'Install your Operator by subscribing to one of the update channels to keep the Operator up to date.' Below this, there are three configuration sections:

- Update channel \***: A dropdown menu set to 'release-v2.7'.
- Version \***: A dropdown menu set to '2.7.3'.
- Installation mode \***: A radio button group where 'A specific namespace on the cluster' (selected) is chosen over 'All namespaces on the cluster (default)'.

Under 'Installed Namespace \*', the 'Operator recommended Namespace: openshift-mtv' option (radio button selected) is chosen over 'Select a Namespace'.

**Figure 8.2: MTV operator parameters**

After the installation is complete, you must create a VM migration controller by creating a `ForkliftController` instance.

**Figure 8.3: The MTV operator requires a ForkliftController instance**

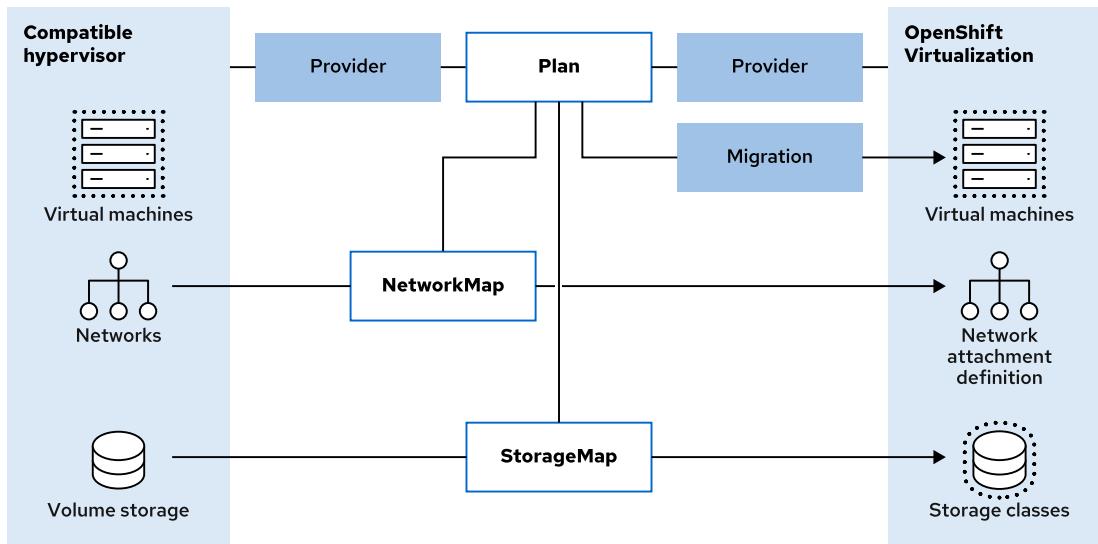
**Note**

Alternatively, to create the migration controller, you can go to Operators > Installed Operators, open the Migration Toolkit for Virtualization Operator page, and then click Create ForkliftController.

The screenshot shows the 'Installed Operators' section of the OpenShift web interface. The 'Migration Toolkit for Virtualization Operator' is selected. The 'Details' tab is active. A prominent yellow warning box contains the text 'ForkliftController required' and a button labeled 'Create ForkliftController'.

**Figure 8.4: Creating the ForkliftController instance**

The MTV operator provides several API resources that you create in the `openshift-mtv` namespace for managing VM migration:



**Figure 8.5: MTV custom resources**

**Provider**

Contains the API endpoint and the credentials to interact with the source and destination virtualization systems.

**NetworkMap**

Declares the source network on the remote virtualization system and the destination network for the imported VMs in OpenShift Virtualization.

**StorageMap**

Declares the source datastores on the remote virtualization system and the destination storage class for the imported VMs in OpenShift Virtualization.

## Plan

Contains the parameters to migrate the VMs from the source Provider to OpenShift Virtualization.

## Migration

Executes the VM migration as defined in the plan CRD. The migration CRD also tracks the status of each migration phase:

- **Initialize:** Initialize migration.
- **DiskAllocation:** Allocate disks.
- **ImageConversion:** Convert image to kubevirt format.
- **DiskTransferV2v:** Copy disks.
- **VirtualMachineCreation:** Create the VM.

## VM Providers

The Provider CR stores attributes that enable MTV to connect to and interact with the source and target VM providers. The MTV can interact with the following hypervisors:

- VMware vSphere
- Red Hat Virtualization (RHV)
- OpenStack
- Remote OpenShift Virtualization clusters

The MTV can also import Open Virtual Appliances (OVAs) that VMware vSphere created.



### Note

The MTV can migrate VMs to an OpenShift Virtualization cluster only.

The MTV automatically creates the host provider for the local OpenShift Virtualization cluster in the `openshift-mtv` namespace.

To create a VM provider from the OpenShift web console, go to **Migration > Providers for virtualization** and click **Create Provider**.

**Create Provider**  
Create by using the form or manually entering YAML or JSON definitions. Provider CR stores attributes that enable MTV to connect to and interact with the source and target providers.

Select provider type \*

<b>vSphere</b> vSphere is VMware's cloud computing virtualization platform.	<b>Red Hat Virtualization</b> Red Hat Virtualization (RHV) is a virtualization platform from Red Hat.	<b>OpenStack</b> OpenStack is a cloud computing platform that controls large pools of resources.
<b>Open Virtual Appliance (OVA)</b> OVA file is a virtual appliance used by virtualization applications.	<b>OpenShift Virtualization</b> OpenShift Virtualization runs and manages virtual machines in OpenShift.	

**Create provider**   **Cancel**  
Create new provider

**Figure 8.6: Creating a VM provider**

## Chapter 8 | Advanced Virtual Machine Management

The following table shows the creation parameters of a Red Hat Virtualization (RHV) Provider from the web console:

Field	Value	Description
Select provider type	Red Hat Virtualization	Source provider to import virtual machines from
Provider resource name	rhv	Unique Kubernetes resource name identifier
URL	<code>https://rhvm.example.com/ovirt-engine/api</code>	URL of the Red Hat Virtualization Manager (RHVM) API endpoint
Username	<code>ocpvirt@internal</code>	RH Virtualization engine REST API username
Password	*****	RH Virtualization engine REST API password

The screenshot shows the 'Create Provider' form. The 'Select provider type' dropdown is set to 'Red Hat Virtualization'. The 'Provider resource name' input field contains 'rhv'. The 'URL' input field contains 'https://rhvm.example.com/ovirt-engine/api'. The 'Username' input field contains 'ocpvirt@internal'. The 'Password' input field contains '\*\*\*\*\*'. Each input field has a green checkmark icon to its right, indicating successful validation.

Figure 8.7: Creating an RHV VM provider from the web console

Consult the references section for the configuration parameters that each VM provider type requires. After you create a **Provider**, the MTV connects to it, collects VM information, and stores VM configurations in the MTV inventory service.

You can review these information types, such as the list of VMs that can be imported, on the **Migration > Providers for virtualization > Provider Details** page.

Provider inventory			
Virtual machines	40	Network interfaces	2
Data centers	1	Storage domains	2
Clusters	1	Hosts	3

**Figure 8.8: VM Provider inventory**

The MTV provides a validation service that verifies the suitability of each VM for migration. The validation service generates a list of *concerns* for each VM based on the VM configuration from the MTV inventory. Those concerns are VM settings that might cause risk during migration, such as requirements on the target OpenShift Virtualization cluster, or features that the VM uses and that OpenShift Virtualization does not support.

You can review the validation service results on the **Provider Details > Virtual Machines** page.

The screenshot shows the 'Providers' section with 'rhv' selected. The 'Virtual Machines' tab is active. A modal window titled 'Warning Concerns' lists three items: 'Placement policy affinity' (with 2 concerns), 'NIC with network filter detected' (with 3 concerns), and 'Cluster has KSM enabled'. The total number of concerns is 3.

Name	Concerns
webrhv-7t2nz	2
webrhv-7zp7b	3

**Figure 8.9: Summary of concerns on the Provider details page**

You can also review the validation service results while creating the migration plan.

The screenshot shows a 'Select VMs' interface with a table header: Migration ..., VM name, Datacenter, Cluster, Host. A single row is selected, showing 'Warning' under Migration, 'webrhv-7t2nz' under VM name, 'Default' under Datacenter, 'Default' under Cluster, and 'N/A' under Host. Below the table, a message states: 'Conditions have been identified that make this VM a **moderate risk** to migrate.' A list of five concerns follows:

- Placement policy affinity:** The VM has a placement policy affinity setting that requires live migration to be enabled in OpenShift Virtualization for compatibility. The target storage classes must also support RWX access mode.
- VM has memory ballooning enabled:** The VM has memory ballooning enabled. This is not currently supported by OpenShift Virtualization.
- NIC with network filter detected:** The VM is using a vNIC Profile configured with a network filter. These are not currently supported by OpenShift Virtualization.
- VM Display Type:** The VM is using the SPICE protocol for video display. This is not supported by OpenShift Virtualization.
- Cluster has KSM enabled:** The host running the source VM has kernel samepage merging enabled for more efficient memory utilization. This feature is not currently supported by OpenShift Virtualization.

At the bottom, a link says 'See the [product documentation](#) for more information.'

**Figure 8.10: Detailed list of concerns when creating the migration plan**

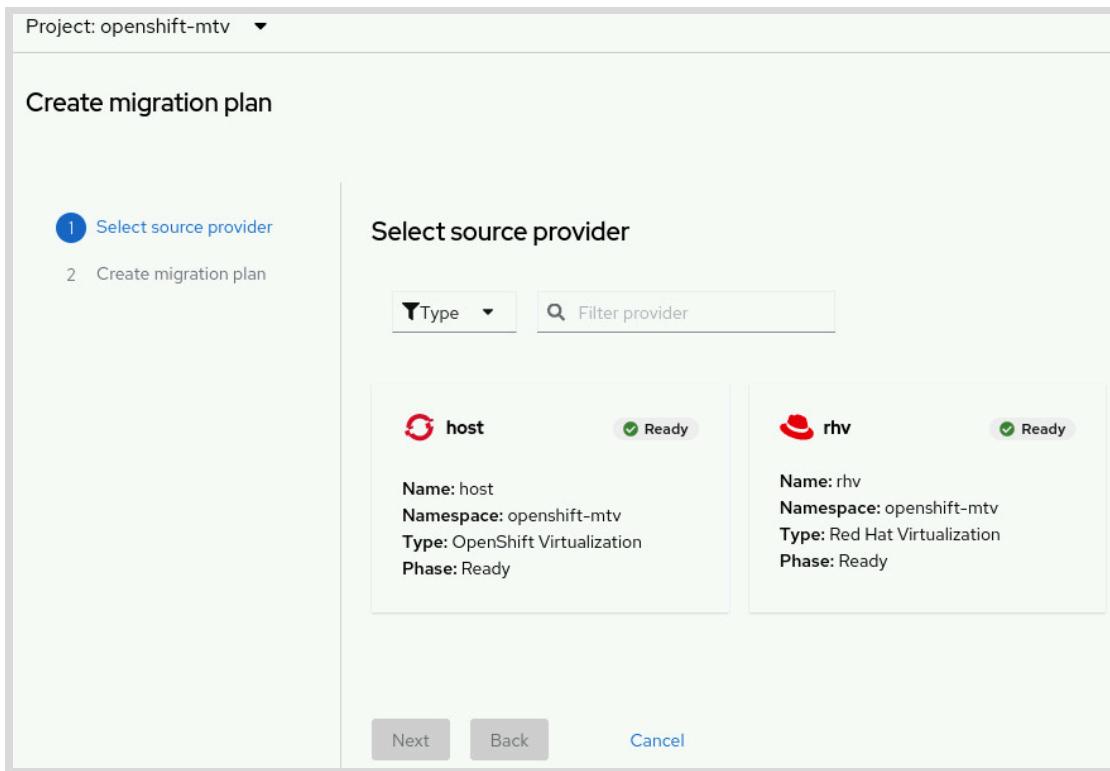
## Migration Plan

The **Plan CR** defines a migration plan with a list of VMs from the same VM provider and associated infrastructure mappings. The **Migration CR** runs a migration plan.

Creating the migration plan requires the following information:

- The plan name
- The source VM provider
- The target OpenShift Virtualization cluster
- The target namespace for the VMs
- The list of VMs to import
- The infrastructure mappings

The MTV provides a migration plan wizard to help you create and start a migration plan by following a few steps. Go to **Migration > Plans for virtualization** and click **Create plan** to start the wizard. Select the source provider that you previously created and click **Next**.



**Figure 8.11: Selecting the source provider of the migration plan**

Select the VMs to import and click **Next**. You then provide the plan name, the target OpenShift Virtualization cluster, and the target namespace for the VMs.

This screenshot shows the 'Set target settings' step of the migration plan creation. It includes fields for 'Plan name \*' (set to 'rhv-migration'), 'Source provider' (set to 'rhv'), 'Selected VMs' (showing '2 VMs selected'), 'Target provider \*' (set to 'host'), and 'Target namespace \*' (set to 'vms-import').

**Figure 8.12: Setting the target settings of the migration plan**

## Chapter 8 | Advanced Virtual Machine Management

After you set the plan name and target information, scroll down and configure the infrastructure mappings for the migration plan.

The MTV inventory service collects the networks and volume storage that the VMs for each VM provider use. Before starting a VM migration, you must create infrastructure mappings to instruct the MTV which network and storage class to use for the VMs on the target cluster.

The MTV provides a **NetworkMap** CR to map a source network to an OpenShift network, and provides a **StorageMap** CR to map source storage to an OpenShift storage class.

You must create both network and storage mappings for a VM provider before starting a migration from that provider. When you create a migration plan by using the web console, the MTV operator provides suggested network and storage mappings.

The screenshot shows the 'Storage and network mappings' section of the migration plan configuration. It is divided into two main sections: 'Network map' and 'Storage map'.  
**Network map:** Contains two rows of mappings:

- Row 1: Source network 'admin-network' mapped to target network 'Pod Networking'.
- Row 2: Source network 'private-network' mapped to target network 'default/vm-private-net'.

Each row has a delete icon on the right. Below this is a '+ Add mapping' button.  
**Storage map:** Contains one row of mappings:

- Source storage 'Dummy storage for source provider vsphere-ova' mapped to target storage 'ocs-external-storagecluster-ceph-rbd-virtualization'.

Below this is a '+ Add mapping' button.

Figure 8.13: Suggested resource mappings

Adjust the suggested mappings or create additional mappings as necessary and click **Create migration plan**.



### Note

OpenShift Virtualization uses the pod network as the default network for VMs. To map more than one source network, you must create a network attachment definition for each additional network.

To create or edit network mappings outside the migration plan wizard, go to **Migration > NetworkMaps for virtualization**.

The screenshot shows the 'NetworkMaps' list view. At the top, there is a search bar labeled 'Filter by name' and a 'Create NetworkMap' button. Below the search bar is a header row with columns: Name, Status, Source pro..., Target pro..., and Owner. A table below lists one item:

Name	Status	Source pro...	Target pro...	Owner
NM rhv-p89q4	Ready	P rhv	P host	P rhv-migration

Figure 8.14: Network mappings

To create or edit storage mappings, go to **Migration > StorageMaps for virtualization**.

Name	Status	Source pro...	Target pro...	Owner
SM rhv-twq2g	Ready	RHV	host	RHV-migration

Figure 8.15: Storage mappings

The MTV provides two migration types: cold migration and warm migration.

### Cold migration

*Cold migration* is the default migration type for all provider types. The source VM is shut down, and then the data is transferred to the target OpenShift cluster. After the transfer is complete, the VM is powered on, in OpenShift Virtualization.



#### Note

The MTV restores the VM state in OpenShift Virtualization when the migration is complete. If the VM is already shut down on the source hypervisor when the migration starts, then the MTV does not power on the VM in OpenShift at the end of the migration.

### Warm migration

*Warm migration* handles the migration in two stages:

The *precopy stage* uses changed block tracking (CBT) snapshots on the source VM provider to copy the VM data while the VM is running. You must enable CBT for each source VM and each VM disk to use warm migration.

The *cutover stage* shuts down the VM and copies the remaining data to the target cluster. The VM is then started on OpenShift Virtualization. You can manually start the cutover stage from the web console, or you can schedule a cutover time in the migration plan manifest.



#### Note

Consult the references section for more information and requirements for warm migration.

After you create the migration plan, you can run the migration by clicking **Start** on the **Migration > Plans for virtualization** page.

The screenshot shows the 'Plans' section of the MTV interface. At the top, there's a search bar labeled 'Filter by name' and a 'Create Plan' button. Below the search bar, there are filters for 'Status', 'Name', and 'Show archived'. The main table lists a single migration plan:

Name	Source ...	Virtual ...	Status	Migratio...
rhv-migration	rhv	2 VMs	Ready	<span>▶ Start</span>

Figure 8.16: Starting the migration process

During the migration, the MTV displays the status of each VM that you import.

The screenshot shows the 'Migration details by VM' interface. It lists three VMs being migrated:

Name	Start t...	End ti...	Data copied	Status
webrhv-7t...	19 Apr 2024, ...		0.00 / 10.00 GB	Transfer disks. █ - █ - █
webrhv-7z...	19 Apr 2024, ...		0.00 / 10.00 GB	Transfer disks. █ - █ - █
webrhv-9...	19 Apr 2024, ...		0.00 / 10.00 GB	Initialize migration. █ - █ - █

Figure 8.17: Status of the migration process

Depending on the VM provider, the MTV creates either an importer pod or a conversion pod to migrate each VM.

### Importer pod

The importer pod transfers the VM image from the source Provider to a data volume in the target OpenShift cluster.

### Conversion pod

The conversion pod runs the `virt-v2v` tool to install and configure the required device drivers on the VM root disk for the VM to run on OpenShift Virtualization, and then transfers the disk to a PVC.

You can consult the log of the migration pod in the target namespace with the `oc logs` command:

```
[user@host ~]$ oc get pod -n target_namespace
NAME          READY   STATUS    RESTARTS   AGE
importer-rhv-...  1/1     Running   0          2m

[user@host ~]$ oc logs -n target_namespace -f importer-rhv-...
...output omitted...
```

To clean up the temporary resources that the MTV creates during the migration, such as the migration pods and temporary PVCs, you must archive the migration plan. Click the vertical ellipsis icon : on the migration plan line and then click **Archive**.

When you archive a migration plan, although you can still consult the detailed status of the migration, the migration logs are deleted. Click the **Show archived** switch to display archived migration plans.



### Important

You must archive a migration plan before removing it. Removing a migration plan does not clean up temporary resources.

Temporary migration resources might prevent some VM operations from running such as VM snapshots.

When you archive a migration plan, the infrastructure mappings that the migration plan wizard created are still available. If you delete the migration plan, then the infrastructure mappings are also deleted. Infrastructure mappings that are created outside the migration plan wizard must be manually deleted.

## Importing VMs from VMware

You can configure the vSphere Provider from the MTV operator to import virtual machines from a VMware vSphere cluster or an ESXi host to OpenShift Virtualization. You can migrate virtual machines from vCenter, or from an ESX/ESXi server that vCenter does not manage, or from an ESX/ESXi server that vCenter manages but does not go through vCenter.

The vSphere Provider has the following prerequisites:

- A container image with the VMware Virtual Disk Development Kit (VDDK) SDK
- The URL of the vSphere SDK endpoint of the vCenter cluster
- Credentials to access the VMware vCenter REST API

## Creating a VDDK Container Image

The MTV operator uses the VDDK SDK to accelerate transferring virtual disks from VMware vSphere. Creating a VDDK image is highly recommended because it can increase the speed of migration.

To create a VDDK image, download the VMware Virtual Disk Development Kit, build a VDDK image, and push the VDDK image to your private image registry.

The VDDK package contains symbolic links. Therefore, you must create a VDDK image on a file system that preserves symbolic links (symlinks).

Before creating a VDDK image, make sure that the following prerequisites are met:

- You configured the Red Hat OpenShift image registry in the Red Hat OpenShift cluster.
- You installed the `podman` package on your machine.
- You are working on a file system that preserves symbolic links (symlinks).
- If you are using an external registry, then OpenShift Virtualization must be able to access it.

**Warning**

Storing the VDDK image in a **public** container registry might violate the VMware license terms.

Use the following procedure to create the VDDK container image and to push it to the container registry.

1. Create and go to a temporary directory:

```
[user@host ~]$ mkdir -vp /tmp/vddk  
[user@host ~]$ cd /tmp/vddk
```

2. In a browser, go to the VMware VDDK version 8 download page at <https://developer.vmware.com/web/sdk/8.0/vddk>
3. Select version 8.0.3, click **Download**, and save the VDDK archive file in the temporary directory that you created.
4. Extract the TAR archive of the VDDK SDK:

```
[user@host vddk]$ tar -xzf VMware-vix-disklib-version.x86_64.tar.gz
```

5. Create a `Dockerfile` text file to build the VDDK container image:

```
FROM registry.access.redhat.com/ubi8/ubi-minimal  
USER 1001  
COPY vmware-vix-disklib-distrib /vmware-vix-disklib-distrib  
RUN mkdir -vp /opt  
ENTRYPOINT ["cp", "-vr", "/vmware-vix-disklib-distrib", "/opt"]
```

6. Build the VDDK container image and push it to the private registry.

```
[user@host vddk]$ TAG=<registry_route_or_server_path>/vddk:latest  
[user@host vddk]$ podman build . -t ${TAG}  
...output omitted...  
[user@host vddk]$ podman push ${TAG}  
...output omitted...
```

7. Ensure that the container image is accessible to your OpenShift Virtualization environment.

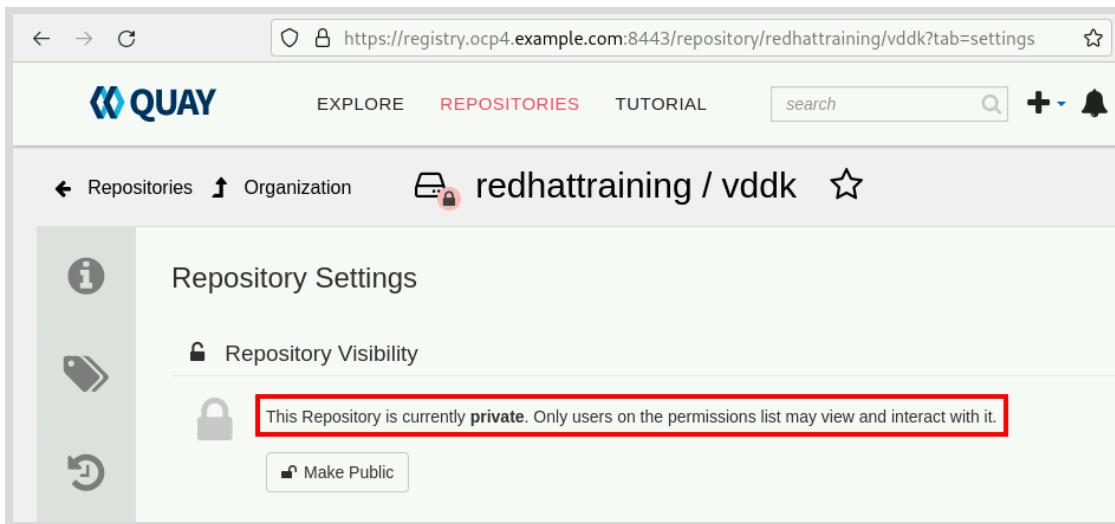


Figure 8.18: Private VDDK container image

**Warning**

The VDDK image must not be stored in a **public** repository. Configure the OpenShift cluster with the appropriate image pull secrets to access the **private** VDDK image from the registry.

## Increasing the Memory of the Network File Copy Service

To migrate more than 10 VMs from an ESXi host in the same migration plan, you must increase the Network File Copy (NFC) service memory of the host. NFC service memory is limited to 10 parallel connections. If you do not increase the NFC service memory, then the migration fails.

Use the following procedure to increase the NFC service memory of a host:

1. Log in to the ESXi host as root.
2. Change the value of `maxMemory` to `1000000000`` in the `/etc/vmware/hostd/config.xml` file.

```
...output omitted...
<nfcsvc>
  <path>libnfcsvc.so</path>
  <enabled>true</enabled>
  <maxMemory>1000000000</maxMemory>
  <maxStreamMemory>10485760</maxStreamMemory>
</nfcsvc>
...output omitted...
```

3. Restart the `hostd` service.

```
/etc/init.d/hostd restart
```

You do not need to reboot the host.

## Creating the vSphere Provider

Go to Migration > Providers for virtualization, and then click Create Provider.

Name	Status	Endpoint	Type	VMs	Netw...	Hosts
host (Host cluster)	Ready		OpenShift	0	1	0

Figure 8.19: MTV provider list

Click **vSphere** to create a VMware vSphere Provider.

Figure 8.20: Creating a vSphere provider

Create the VMware vSphere source provider with the following parameters. Click **Create provider** when done.

Field	Value	Description
Select provider type	vSphere	Source provider to import virtual machines from
Provider resource name	vmware	Unique Kubernetes resource name identifier

Field	Value	Description
Endpoint type	vCenter	Specify whether the endpoint is vCenter or ESXi. Selecting vcenter enables you to see all virtual machines in the cluster, whereas selecting ESXi shows only the virtual machines that are running on that ESXi server.
URL	<code>https://vcenter.example.com/sdk</code>	URL of the vCenter SDK endpoint. Ensure that the URL includes the /sdk path.
VDDK init image	<code>registry.ocp4.example.com:8443/redhattraining/vddk:latest</code>	VDDK container image of the VMware provider. If no VDDK image is provided, then the migration might be slow.
Username	<code>user@vmware</code>	VMware vCenter REST API username
Password	.....	VMware vCenter REST API password
Skip certificate validation	<i>Not selected</i>	If selected, then the provider's REST API TLS certificate is not validated.

You can view the provider settings after creation.

Project: openshift-mtv ▾

Providers > Provider Details

**P** **vmware** Ready

**Actions** ▾

**Details**    YAML    Credentials    Virtual Machines    Hosts

**Provider details**

**Type**: VMware

**Name**: vmware

**Namespace**: NS openshift-mtv

**URL**: <https://vcenter.demo.redhat.com/sdk>

**Secret**: S vmware-sfhsd

**Created at**: Jun 12, 2024, 12:17 AM

**VDDK init image**: registry.ocp4.example.com:8443/redhattraining/vddk:latest

**Owner**: No owner

Figure 8.21: vSphere provider settings

## Creating the Plan

Go to Migration > Plans for Virtualization and click **Create plan**. Select the vSphere source provider and the VMs to migrate. Click **Next** and create the plan with the following parameters:

Field	Value
Plan resource namespace	openshift-mtv
Plan name	migrate-vmware
Target provider	host
Target namespace	default
Network map	PodNetworking
Storage map	ocs-storagecluster-ceph-rbd-virtualization



### Important

The web console warns about the following condition:

#### Changed Block Tracking (CBT) not enabled

Changed Block Tracking has not been enabled on the VM. This feature is a prerequisite for VM warm migration.

You can enable CBT by setting the `ctkEnabled=TRUE` parameter in the VM settings, or by editing the `VMX` file of the VM.

For more information, refer to the MTV documentation in the references section.

**Note**

To perform a cold migration, ensure that the source VMs are powered off. MTV attempts to power off the VMs during migration.

Review the migration plan and click **Create migration plan** when done.

The plan that you created appears in the list. Click **Start** to initiate the VM migration and **Start** in the dialog to confirm.

The screenshot shows a table titled 'Plans' within a 'Plans' section. The table has columns: Name, Source pr..., Target pr..., VMs, Status, and Description. A 'Create plan' button is located at the top right. The table displays one row:

Name	Source pr...	Target pr...	VMs	Status	Description
migrate-vmware (cold)	vmware	host	2	Ready	Migrate VMs from VMware vCenter ...

At the bottom right of the table, there is a 'Start' button and a '...' button.

**Figure 8.22: Migration plan list**



## References

For more information about installing operators by using the OperatorHub, refer to the *Installing from OperatorHub Using the Web Console* section in the *Administrator Tasks* chapter in the Red Hat OpenShift Container Platform 4.16 Operators documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/operators/index#olm-installing-from-operatorhub-using-web-console\\_olm-adding-operators-to-a-cluster](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/operators/index#olm-installing-from-operatorhub-using-web-console_olm-adding-operators-to-a-cluster)

For more information about installing operators by using the command line, refer to the *Installing from OperatorHub Using the CLI* section in the *Administrator Tasks* chapter in the Red Hat OpenShift Container Platform 4.16 Operators documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/operators/index#olm-installing-operator-from-operatorhub-using-cli\\_olm-adding-operators-to-a-cluster](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/operators/index#olm-installing-operator-from-operatorhub-using-cli_olm-adding-operators-to-a-cluster)

For more information about configuring VM providers, refer to the *Adding Providers* section in the *Migration Toolkit for Virtualization* documentation at

[https://docs.redhat.com/en/documentation/migration\\_toolkit\\_for\\_virtualization/2.7/html-single/installing\\_and\\_using\\_the\\_migration\\_toolkit\\_for\\_virtualization/index#adding-source-providers](https://docs.redhat.com/en/documentation/migration_toolkit_for_virtualization/2.7/html-single/installing_and_using_the_migration_toolkit_for_virtualization/index#adding-source-providers)

For more information about warm migration, refer to the *Warm Migration* section in the *Migration Toolkit for Virtualization* documentation at

[https://docs.redhat.com/en/documentation/migration\\_toolkit\\_for\\_virtualization/2.7/html-single/installing\\_and\\_using\\_the\\_migration\\_toolkit\\_for\\_virtualization/index#warm-migration\\_mtv](https://docs.redhat.com/en/documentation/migration_toolkit_for_virtualization/2.7/html-single/installing_and_using_the_migration_toolkit_for_virtualization/index#warm-migration_mtv)

For more information about converting VMs to KVM, refer to the article at  
<https://access.redhat.com/articles/1351473>

For more information about Changed Block Tracking (CBT), refer to the article at  
<https://kb.vmware.com/s/article/1020128>

For more information about migrations from VMware, refer to the video at  
<https://www.youtube.com/watch?v=Rp8YcDgSrZo>

## ► Guided Exercise

# Migrate Virtual Machines from Foreign Hypervisors

Import a VMware vSphere virtual machine to OpenShift Virtualization by using the migration toolkit for virtualization.

## Outcomes

- Deploy the migration toolkit for virtualization.
- Create infrastructure mapping to map source networks and datastores to OpenShift Virtualization networks and storage classes.
- Create and start a migration plan.
- Monitor the progress of the migration.
- Confirm that the VM is connected to the additional network and is configured with a static IP address in the 192.168.51.0/24 network.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start advanced-import
```

## Instructions

A VM was exported from a VMware vSphere cluster for importing into OpenShift. The `rhel9-web` VM is available in the Open Virtual Appliance (OVA) format on the `utility` machine. The VM has access to two networks: a management network (`admin-network`) and a private network (`private-network`).

The `lab` command creates a `vm-private-net` network attachment definition for the private network in the `default` namespace.

The private network is in the 192.168.51.0/24 range and is accessible from the `utility` machine. A web service is running on the `rhel9-web` VM to test connectivity from the `utility` machine by using the `curl` command.

Deploy and use the migration toolkit for virtualization (MTV) to import the VMware VM into your OpenShift cluster. Confirm that the VM is accessible from the private network.

- 1. As the `admin` user, install the MTV operator.
  - 1.1. Open a web browser and go to the web console URL: <https://console-openshift-console.apps.ocp4.example.com>
  - 1.2. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.

- 1.3. Click Operators > OperatorHub. In the Filter by keyword field, type `mtv` to locate the MTV operator, and click Migration Toolkit for Virtualization Operator.

The screenshot shows the Red Hat OpenShift web console. On the left, there's a sidebar with navigation links like Home, Operators, Workloads, Networking, Storage, Builds, Observe, and Compute. Under Operators, 'OperatorHub' is selected. The main area is titled 'OperatorHub' and describes it as a place to discover operators from the Kubernetes community and Red Hat partners. A search bar at the top right contains the text 'mtv'. Below the search bar, there are several filter categories: Source (do316 Operator Catalog), Provider (Red Hat, Community), and Install state (Installed, Not Installed). A red box highlights a specific operator card for 'Migration Toolkit for Virtualization Operator' provided by Red Hat. The card includes a small icon, the operator name, its provider, and a brief description: 'Facilitates migration of VM workloads to OpenShift Virtualization'.

- 1.4. The web console displays information about the MTV operator. Set the Channel to `release-2.7` and confirm that the `2.7.3` version is displayed. Click **Install** to proceed to the **Install Operator** page.

This screenshot shows the 'Migration Toolkit for Virtualization Operator' configuration page. At the top, there's a logo and the text '2.7.3 provided by Red Hat'. Below that is a large blue 'Install' button, which is also highlighted with a red box. To the right of the button, there's a description: 'The Migration Toolkit for Virtualization Operator manages the deployment and life cycle of Migration Toolkit for Virtualization on OpenShift Container Platform.' Underneath, there are two dropdown menus: 'Channel' set to 'release-v2.7' and 'Version' set to '2.7.3'. Both of these dropdowns are also highlighted with red boxes. At the bottom, there's a section titled 'Installation' with the note: 'OpenShift Virtualization must be installed on an OpenShift migration target cluster before you can use MTV to transfer any VMs to that cluster'.

- 1.5. Review the default configuration and click **Install** to install the operator in the `openshift-mtv` namespace.

OperatorHub > Operator Installation

## Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date.

**Update channel \***

release-v2.7

**Version \***

2.7.3

**Installation mode \***

All namespaces on the cluster (default)  
This mode is not supported by this Operator

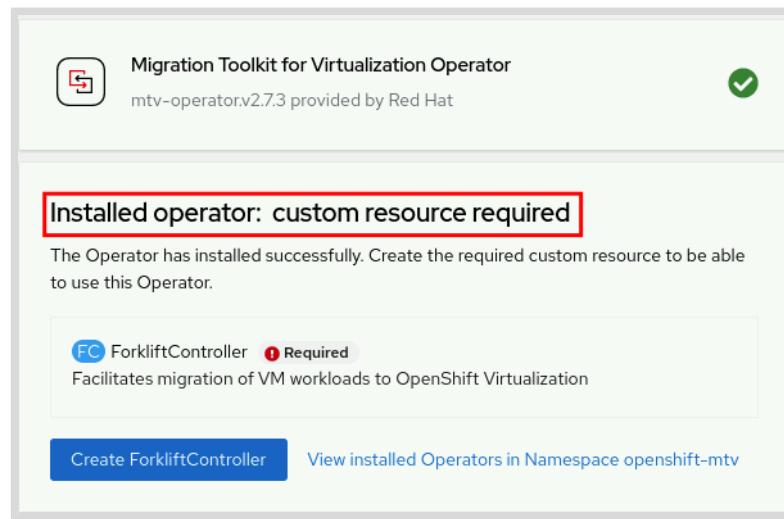
A specific namespace on the cluster  
Operator will be available in a single Namespace only.

**Installed Namespace \***

Operator recommended Namespace: PR openshift-mtv

Select a Namespace

Wait until the installation is complete and the web console displays the **custom resource required** message.



▶ 2. Create the VM migration controller.

- 2.1. After the operator installation completes, click **Create ForkliftController**.

**Note**

Alternatively, you can go to Operators > Installed Operators, open the Migration Toolkit for Virtualization Operator page, and click Create ForkliftController to view the creation parameters for the migration controller.

The screenshot shows the 'Operator details' page for the 'Migration Toolkit for Virtualization Operator'. The 'ForkliftController required' section contains a red box around the 'Create ForkliftController' button.

- 2.2. Leave the default values and click **Create**.
- 2.3. Wait a few minutes until the web console shows the `Web console update is available` message and the controller status is **Successful**. Click **Refresh web console**.

The screenshot shows the 'ForkliftControllers' list page. A red box highlights the 'Refresh web console' button in a modal window. Another red box highlights the 'Successful' status of the 'forklift-controller' entry in the list table.

Name	Kind	Status	Labels
forklift-controller	ForkliftController	Running, Successful	No labels

- 2.4. Confirm that the **Migration > Overview** page is available.

The screenshot shows the Red Hat OpenShift Migration Toolkit for Virtualization interface. The left sidebar has a navigation tree with 'Migration' selected and 'Overview' highlighted. The main panel displays the 'Welcome' screen for the Migration Toolkit, which includes an illustration of a person interacting with a cloud icon containing a virtual machine.

- ▶ 3. Create the vsphere-ova OVA provider.
  - 3.1. Go to **Migration > Providers for virtualization**. Click **Show default projects** and select the **openshift-mtv** project from the **Projects** list. Click **Create Provider**.
  - 3.2. Select the **Open Virtual Appliance (OVA)** provider type. Configure the provider with the following values, and click **Create provider**:

Field	Value
Select provider type	Open Virtual Appliance (OVA)
Provider resource name	vsphere-ova
URL	utility.lab.example.com:/exports-ocp4/ova

Project: openshift-mtv ▾

### Provider details

Select provider type \*

Open Virtual Appliance (OVA)

Click to unselect.

Provider resource name \*

vsphere-ova ✓

Unique Kubernetes resource name identifier.

URL \*

utility.lab.example.com:/exports-ocp4/ova ✓

The NFS shared directory containing the Open Virtual Appliance (OVA) files, for example: 10.10.0.10:/ova .

Create provider Cancel

Create new provider

**Note**

After creating the provider, the web console might briefly display error messages, such as **The provider is not ready - ConnectionTestFailed** and **Inventory server is not reachable**.

Those errors are temporary and should disappear after the MTV operator validates the provider.

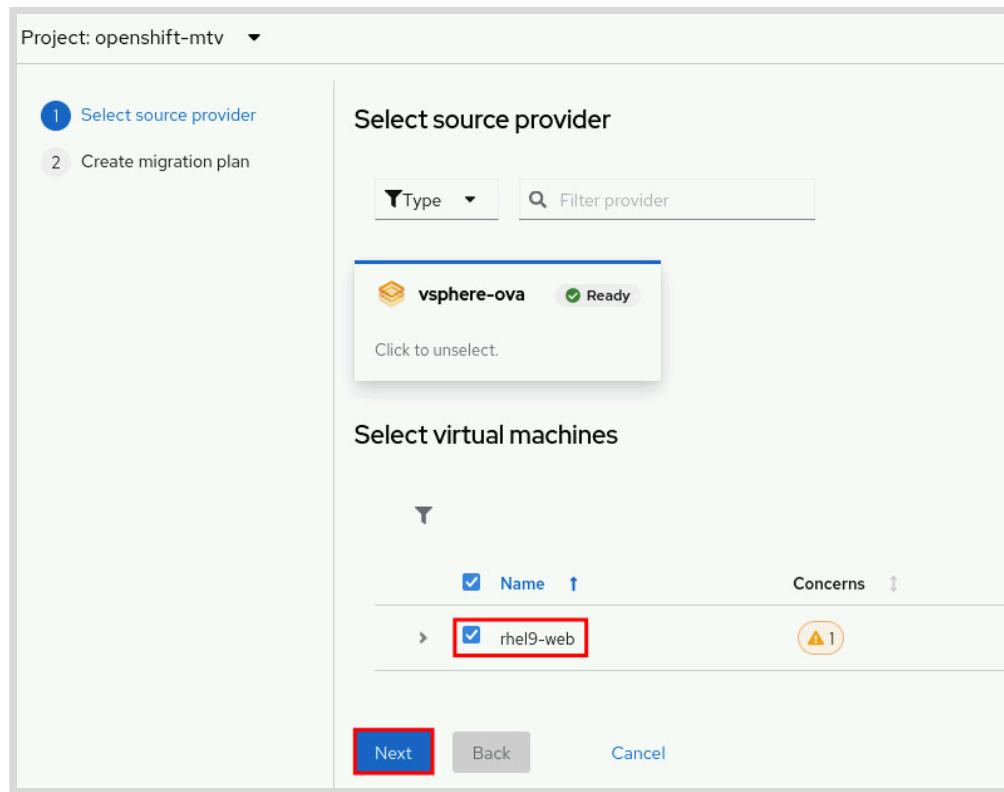
- 3.3. Go to **Migration > Providers for virtualization** and wait until the **vsphere-ova** provider status is Ready. Confirm that the VM count is 1.

Project: openshift-mtv ▾

### Providers

Name	Status	Type	VMs	Net...
PR host	Host cluster	Ready	0	2
PR vsphere-ova		Ready	1	2

- 4. Create a migration plan to migrate the **rhel9-web** VM from the **vsphere-ova** provider to OpenShift Virtualization.
- 4.1. Go to **Migration > Plans for virtualization**, ensure that the **openshift-mtv** project is selected, and click **Create plan**.
  - 4.2. Select the **vsphere-ova** source provider. Select the **rhel9-web** VM from the **Select virtual machines** list. Click **Next**.



- 4.3. The **Create migration plan** pane opens. Configure the migration plan with the following values:

Field	Value
Plan name	<code>rhel9-web-import</code>
Target provider	<code>host</code>
Target namespace	<code>vms-import</code>

Project: openshift-mtv ▾

### Create migration plan

1 Select source provider  
2 Create migration plan

**Plan name \***  
rhel9-web-import ✓

**Source provider**  
Source provider  
P vsphere-ova

**Selected VMs**  
1 VMs selected

**Target provider**  
Target provider \*  
host

**Target namespace \***  
vms-import

- 4.4. Go to the **Storage and network mappings** section. The form displays the suggested networking and storage mapping from MTV.

Confirm that **admin-network** is mapped to **Pod Networking**. Edit the mapping for **private-network** to use **default/vm-private-net**.

### Storage and network mappings

Network map: NM

admin-network	Pod Networking
private-network	default/vm-private-net

+ Add mapping

- 4.5. Edit the suggested storage mapping for **Dummy storage** to use the **ocs-external-storagecluster-ceph-rbd-virtualization** storage class.

### Storage map: SM

Dummy storage for source provider vsphere-ova
ocs-external-storagecluster-ceph-rbd-virtualization

- 4.6. Click **Create migration plan**. The **Details** page of the **rhel9-web-import** migration plan appears.

**Note**

After you create a migration plan that uses suggested network and storage mappings, MTV creates NetworkMap and StorageMap resources that include the suggested mappings and your changes.

To review and manage the NetworkMap resource that MTV creates, go to **Migration > NetworkMaps for virtualization**. Select the NetworkMap resource name that begins with the name of the source provider that is used for migration. From the **Details** tab, you can edit the network mapping settings and review their status.

To review and manage the StorageMap resource that MTV creates, go to **Migration > StorageMaps for virtualization**. Select the StorageMap resource name that begins with the name of the source provider that is used for migration. From the **Details** tab, you can edit the storage mapping settings and review their status.

▶ **5.** Start and monitor the VM migration plan.

- 5.1. From the **Details** page of the `rhel9-web-import` migration plan, click **Start migration** and click **Start** to begin the migration process. You can also go to the **Migration > Plans for virtualization** page, click **Start** next to the `rhel9-web-import` line, and click **Start** to begin the migration.
- 5.2. Go to the **Virtual Machines** tab of the `rhel9-web-import` migration plan. Expand the `rhel9-web` VM to review details about the migration, including the name of the conversion pod and the status of the migration tasks.

Scroll down and go to the **Pipelines** section to monitor the migration.

Wait until the `Convert image to kubevirt` step starts.

The screenshot shows the Red Hat OpenShift Pipeline interface. At the top, there's a navigation bar with the Red Hat OpenShift logo, a project dropdown set to "openshift-mtv", and user information for "admin". Below the header, the pipeline table has columns: Name, Description, Tasks, Started at, and Error. There are five rows in the table:

Name	Description	Tasks	Started at	Error
Initialize	Initialize migration.	1/1	Jan 8, 2025, 1:45 PM	
DiskAllocation	Allocate disks.	2/2	Jan 8, 2025, 1:45 PM	
<b>ImageConversion</b>	Convert image to kubevirt.	1/1	Jan 8, 2025, 1:45 PM	
DiskTransferV2v	Copy disks.	0/2	-	
VirtualMachineCreation	Create VM.	0/1	-	

The "ImageConversion" row is highlighted with a red box around its entire row. The "DiskAllocation" row is also partially visible below it.

- 5.3. From the command line, log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 5.4. Change to the `vms-import` project and verify that the conversion pod is running.

```
[student@workstation ~]$ oc project vms-import
Now using project "vms-import" on server "https://api.ocp4.example.com:6443".

[student@workstation ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
rhel9-web-import-...   1/1     Running   0          13s
```

- 5.5. Observe the conversion pod logs. The pod name might differ in your environment.

```
[student@workstation ~]$ oc logs -f rhel9-web-import-...
...output omitted...
```

Press `Ctrl+C` to exit the `oc logs` command.

- 5.6. Change to the OpenShift web console and wait until the migration is complete, which might take up to ten minutes.

Name	Description	Tasks	Started at	Error
Initialize	Initialize migration.		Jan 8, 2025, 1:45 PM	
DiskAllocation	Allocate disks.	2 / 2	Jan 8, 2025, 1:45 PM	
ImageConversion	Convert image to kubevirt.		Jan 8, 2025, 1:45 PM	
DiskTransferV2v	Copy disks.	2 / 2	Jan 8, 2025, 1:48 PM	
VirtualMachineCreation	Create VM.		Jan 8, 2025, 1:50 PM	

- 6. Review the `rhel9-web` VM configuration.

- 6.1. Go to **Virtualization > VirtualMachines**, select the **vms-import** project from the **Projects** list, and select the **rhel9-web** VM.
- 6.2. Go to the **Configuration > Storage** tab and confirm that both disks use the **ocs-external-storagecluster-ceph-rbd-virtualization** storage class.

Name	Source	Size	Drive	Interface	Storage class
vol-0	PVC rhel9-web-import-7e94le ca329771d2a0 b604153a5018 70eb23-qwwdt	4.00 GiB	Disk	virtio	ocs-external-storagecluster-ceph-rbd-virtualization
vol-1	PVC rhel9-web-import-7e94le ca329771d2a0 b604153a5018 70eb23-xfwk4	1.00 GiB	Disk	virtio	ocs-external-storagecluster-ceph-rbd-virtualization

- 6.3. Go to the **Configuration > Network** tab and confirm that the VM has two networks: one that is using the pod networking, and the other that is using the **default/vm-private-net** network.

Name	Model	Network	Type
net-0	virtio	Pod networking	Masquerade
net-1	virtio	default/vm-private-net	Bridge

- ▶ 7. Start the **rhel9-web** VM and confirm that the web service is accessible on the **192.168.51.0/24** network.
- 7.1. Click **Actions > Start**.
  - 7.2. Go to the **Overview** tab and wait until the VM is running and the VM IPs are displayed in the **Network Interfaces** card. Note the IP of the **net-1** interface, which you use in the next step to verify that the web service is accessible from this network.

Network interfaces (2)	
Name	IP address
net-0	10.8.2.67
net-1	192.168.51.102
+1 more	

**Note**

The IP might be different in your environment.

- 7.3. From the command line, connect to the **utility** machine with SSH as the student user.

```
[student@workstation ~]$ ssh utility
...output omitted...
```

- 7.4. Use the `curl` command to confirm that the web server responds to requests. Use the VM IP from the previous step.

```
[student@utility ~]$ curl 192.168.51.102
<html>
<body>
<h1>Hello from VMware virtual machine!</h1>
<small>Provisioned on VMware ESXi 8.0.1</small>
</body>
</html>
```

- 7.5. Disconnect from the utility machine.

```
[student@utility ~]$ exit
logout
Connection to utility closed.
[student@workstation ~]$
```

- 8. Archive the migration plan to clean up the temporary resources.

- 8.1. On the web console, go to **Migration > Plans for virtualization**. Click **Show default projects** and select the `openshift-mtv` project from the **Projects** list. Click the vertical ellipsis icon : next to the `rhel9-web-import` line and click **Archive**. Click **Archive** to confirm the operation.

## Finish

As the student user on the workstation machine, confirm that you are in the /home/student directory and then use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advanced-import
```

# Controlling the Scheduling of Virtual Machines

---

## Objectives

- Use run strategies, labels, selectors, affinity rules, and taints to control the placement of virtual machines on cluster nodes.

## Virtualization Node Placement

Red Hat OpenShift schedules VMs and pods to nodes according to node placement rules. The OpenShift built-in scheduler identifies the most suitable node for a VM or pod when you create it. The default scheduler meets the needs of most OpenShift Virtualization users.

However, in some situations, administrators might want more control over which node is used for VM or pod placement. You can use the Red Hat OpenShift advanced scheduling features to configure VMs and pods to run on a particular node or alongside a specific VM or pod.

Red Hat OpenShift includes several advanced scheduling features that apply to both VMs and pods:

- *Pod affinity rules*, to keep sets of pods and VMs close to each other, on the same nodes. For example, you can run a REST service and its database on the same node to minimize network latency.
- *Pod anti-affinity rules*, to keep sets of pods and VMs far away from each other, on different nodes. For example, you can run replica pods of the same deployment in different nodes, so that if a node fails, then you do not lose all the pods for the workload.
- *Node affinity*, to keep sets of pods and VMs running on the same group of nodes. For example, you can configure a VM or a pod to run on nodes with a specific CPU.
- *Node selectors*, to schedule VMs and pods to a specific set of nodes. For example, you can schedule a pod to a node that provides special hardware that the pod needs.
- *Taints and tolerations*, to avoid scheduling VMs and pods to a specific set of nodes. For example, you can block a pod from running on a node that is reserved for OpenShift cluster services or control plane services.

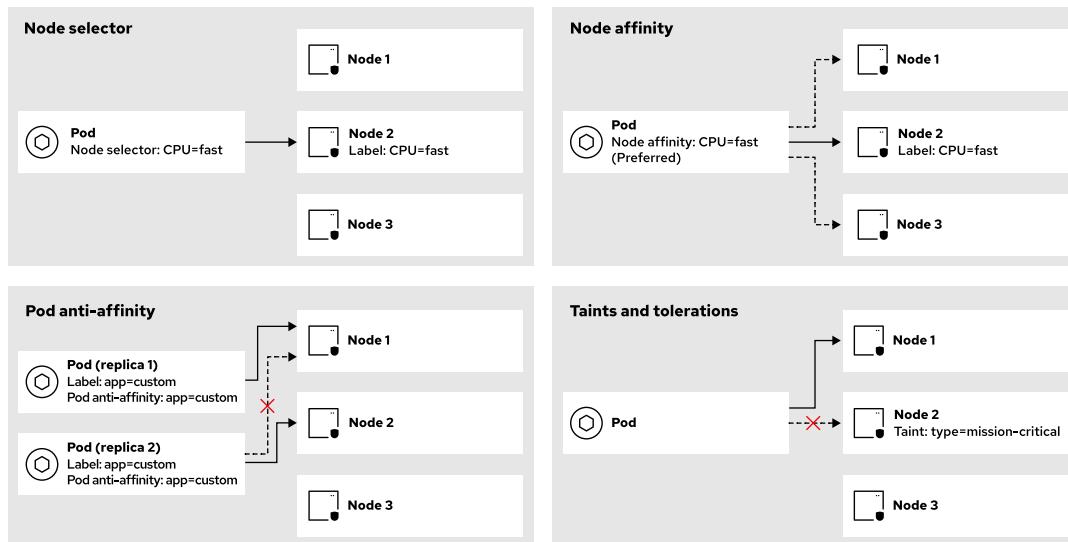


Figure 8.41: OpenShift advanced scheduling features

You can configure node placement rules to ensure that VMs run on the most appropriate nodes. As an administrator, you might want particular VMs to run on the same nodes, on separate nodes, or on a specific node to ensure that the VMs have access to the features that they need from the underlying hardware.

Node placement rules are declared in the `spec` field of the `VirtualMachine` YAML manifest by using the rule types as follows:

#### `nodeSelector`

VMs are scheduled on nodes that are labeled with the key-value pairs as specified in this field. The node's labels must exactly match all of the listed key-value pairs. Because a node selector specifies target node labels on your VM, OpenShift schedules the VM only on nodes that match the target labels. Thus, with a node selector, you must define the target labels in the VM specification file and label the nodes with those labels.

For example, if you want a VM to run on a node that includes the `example-key-1@example-value-1` and `example-key-2@example-value-2` labels, then use the following metadata in your VM manifest:

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-node-selector
spec:
  template:
    spec:
      nodeSelector:
        example-key-1: example-value-1
        example-key-2: example-value-2
...output omitted...
```

If the node does not meet all requirements, then the VM is not scheduled.

**affinity**

*Affinity* refers to a VM property that controls the VM preference for a node. *Anti-affinity* is a VM property that restricts or prevents the placement of a VM on a node.

You use node affinity rules to schedule a VM to run on a group of nodes. For example, you can configure a VM to run on nodes in a specific availability zone. You can also specify anti-affinity rules to prevent a VM from running on a particular group of nodes.

Node affinity, which is conceptually similar to node selectors, provides a more flexible way to specify constraints on node selection. Whereas node selectors specify the target nodes by using a set of label key-value pairs, node affinity enables a conditional approach with logical operators in the matching process.

Because VMs are based on the `pod` object, you can also use pod affinity and pod anti-affinity rules with VMs. With pod affinity and pod anti-affinity, you can control the nodes for your VM based on the labels of other VMs or pods.

By using pod affinity rules, you can locate a VM on the same group of nodes as other VMs and pods, based on its label selector. As an example, you can configure the scheduler to place VMs of two services onto the same group of nodes, because the services communicate with each other regularly.

By using pod anti-affinity rules, you can prevent the scheduler from locating a VM on the same group of nodes as other VMs and pods based on its label selector. For example, you can configure the scheduler to prevent a VM of a particular service from being on the same nodes as VMs and pods of another service, because the two services interfere and reduce the performance of the services.

For more flexibility during scheduling, you can specify whether the affinity rule is required for the VM to run, or is preferred but not required. Because required rules work similarly to node selectors, the OpenShift scheduler schedules the VM only if the conditions are met. By using preferred rules, the OpenShift scheduler tries to schedule the VM on a node that meets the rules. If the scheduler does not find a node, then the scheduler schedules the VM on a node that does not meet the rules.

**Note**

Affinity rules apply only during scheduling; running VMs are not rescheduled even if the constraints are no longer met.

In the following example, the affinity rule specifies that the VM must run on a node with the `example.io/example-key=example-value-1` label or with the `example.io/example-key=example-value-2` label. The `preferredDuringSchedulingIgnoredDuringExecution` specification tells the scheduler to prioritize nodes with the `example-node-label-key=example-node-label-value` label, although it can ignore the constraint if none of the nodes have the `example-node-label-key=example-node-label-value` label.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-node-affinity
spec:
  template:
    spec:
```

```

affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: example.io/example-key
              operator: In
              values:
                - example-value-1
                - example-value-2
    preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
            - key: example-node-label-key
              operator: In
              values:
                - example-node-label-value
...output omitted...

```

In the following example, the pod affinity rule specifies that the VM must run on a node that has a running pod with the `example.io/example-key-1@example-value-1` label. The pod anti-affinity rule also specifies that the VM must not run on a node that has a running pod with the `example.io/example-key-2@example-value-2` label. The `preferredDuringSchedulingIgnoredDuringExecution` specification tells the scheduler that if all nodes have a pod with this label, then the scheduler can ignore this constraint and schedule the VM on the node.

```

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-node-affinity
spec:
  template:
    spec:
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: example.io/example-key-1
                    operator: In
                    values:
                      - example-value-1
          topologyKey: kubernetes.io/hostname ⓘ
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: example.io/example-key-2
                      operator: In
                      values:

```

```

- example-value-2
topologyKey: kubernetes.io/hostname ②
...output omitted...

```

- ① ② The `topologyKey` object cannot be empty in pod affinity and anti-affinity rules. The `topologyKey` object specifies the node label to denote the topology domain of a node, such as the node name, rack, or cloud provider zone or region. For pod anti-affinity rules that use the `requiredDuringSchedulingIgnoreDuringExecution` specification, Kubernetes limits the value of the `topologyKey` object to the `kubernetes.io/hostname` value.

## tolerations

You can configure taints on nodes so they schedule a VM only if it has a matching toleration. The node with a taint repels all the VMs except those VMs that have a toleration for that taint. A toleration specifies that a VM can be scheduled on a node if the VM's toleration matches the taint on the node. However, if a VM is configured to tolerate a taint, then it is not required to schedule the VM on a node that is configured with that taint.

You can apply taints to a node in the node specification YAML file by setting the `spec.taints` parameter. You apply tolerations to a VM in the VM specification YAML file by setting the `spec.tolerations` parameter.

Taints and tolerations consist of a key, a value, and an effect.

- The key is any string, up to 253 characters.
- The value is any string, up to 63 characters.
- The effect is one of the following options:

### NoSchedule

If the pod does not match the taint, then the node prevents scheduling it. Existing pods on the node continue running on the node.

### PreferNoSchedule

If the pod does not match the taint, then the scheduler tries to avoid scheduling it on the node. Existing pods on the node continue running on the node.

### NoExecute

If the pod does not match the taint, then the node prevents scheduling it. The scheduler removes any existing pods on the node that do not have a matching toleration.

Tolerations also include the `operator` parameter. The following values for the `operator` parameter are possible:

### Equal

The toleration matches the taint if the key, value, and effect parameters match. This behavior is the default.

### Exists

The toleration matches the taint if the key and effect parameters match. You must leave a blank parameter for `value`.

In this example, the toleration of the VM requires a node with the `key=virtualization:NoSchedule` taint.

```

apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-tolerations
spec:
  tolerations:
    - key: "key"
      operator: "Equal"
      value: "virtualization"
      effect: "NoSchedule"
...output omitted...

```

## Create Node Placement Rules for VMs by Using the OpenShift Web Console

You can use the OpenShift web console to create node placement rules for VMs. From the web console, go to **Virtualization > VirtualMachines**. From the **Project** list, select the project of the VM and select the VM. Go to the **Configuration > Scheduling**. From the **Scheduling and resource requirements** page, you can add a node selector, tolerations, and affinity rules to a VM.

The screenshot shows the 'VirtualMachines' section of the OpenShift web console. A specific VM named 'server1' is selected and is currently 'Running'. The 'Configuration' tab is active, and the 'Scheduling and resource requirements' section is displayed. This section includes fields for 'Node selector' (set to 'No selector'), 'Tolerations' (0 rules), and 'Affinity rules' (0 rules). Each of these fields has a blue pencil icon next to it, indicating they are editable.

**Figure 8.42: Scheduling and resource requirements**

Select the pencil icon under the type of rule to create. A form is provided for each rule type. Complete the form and restart the VM to apply the rule.

The following example shows the form and the values for a node selector rule:

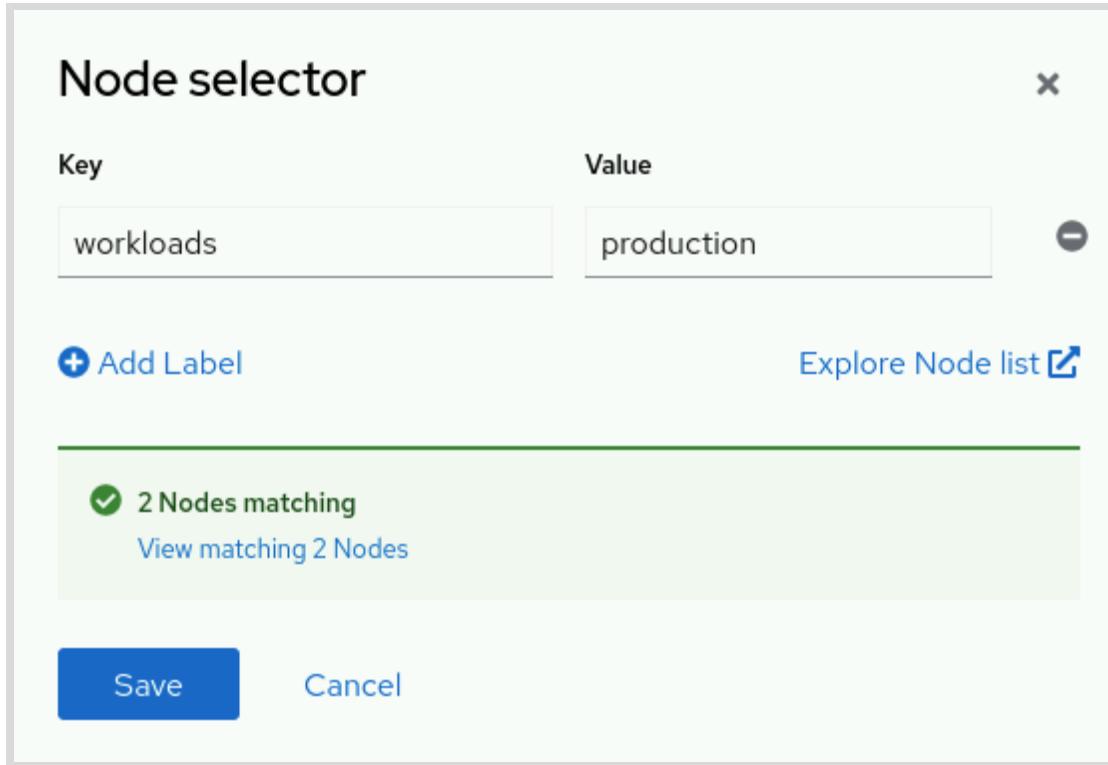


Figure 8.43: VM node selector

You can configure multiple rules, including different types, for a VM.

The screenshot shows the "Scheduling and resource requirements" tab of a VM configuration. On the left is a sidebar with "Details", "Storage", "Network", "Scheduling", and "SSH" sections. The "Scheduling" section is expanded, showing "Node selector" with the value "workloads=production", "Tolerations" with "1 Tolerations rules", and "Affinity rules" with "1 Affinity rules". Each rule section has a blue edit icon.

Figure 8.44: Multiple node placement rules

Node placement rules that conflict or are too restrictive can prevent a VM from running. An error is displayed if OpenShift cannot schedule a VM to a node that satisfies the node placement rules.

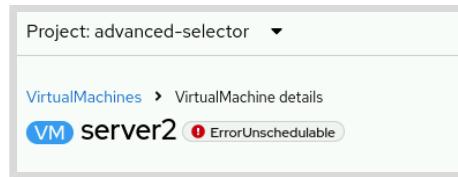


Figure 8.45: Scheduler error

## Node Placement Rules for Virtualization Components

OpenShift Virtualization also uses node placement rules for its components. Edit the `spec.infra.nodePlacement` and `spec.workloads.nodePlacement` objects of the HyperConverged CR to specify the nodes for infrastructure and workload resources. In the following example, node selectors specify the nodes for OpenShift Virtualization components:

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement:
      nodeSelector:
        example.io/example-infra-key: example-infra-value ①
  workloads:
    nodePlacement:
      nodeSelector:
        example.io/example-workloads-key: example-workloads-value ②
```

- ① Infrastructure resources must be scheduled on nodes with the `example.io/example-infra-key: example-infra-value` label.
- ② Workload resources must be scheduled on nodes with the `example.io/example-workloads-key: example-workloads-value` label.

## Configure the Run Strategy

VMI have settings that control the running state of the VM. A VM's run strategy determines the behavior of a VMI according to a series of conditions, which are defined in the `.spec.running` or the `.spec.runStrategy` parameter.

### `.spec.running`

The `.spec.running` Boolean value specifies whether the VM is running.

#### `true`

When the Boolean is set to `true`, Red Hat OpenShift Virtualization ensures that the VM is always running. OpenShift Virtualization restarts the VM if it fails. OpenShift Virtualization also restarts the VM when you stop it by gracefully shutting down the operating system or when you delete the VMI resource.

#### `false`

When you set the `.spec.running` Boolean to `false`, OpenShift Virtualization ensures that the VM is not running.

OpenShift Virtualization automatically updates the Boolean when you control the VM by using the `virtctl` command or by using the web console. If you run the `virtctl stop` command or click **Actions > Stop** in the OpenShift web console, then OpenShift Virtualization switches the `.spec.running` Boolean to `false`. If you run the `virtctl start` command or click **Actions > Start** in the OpenShift web console, then OpenShift Virtualization switches the `.spec.running` Boolean to `true`.

#### `.spec.runStrategy`

The `.spec.runStrategy` parameter provides more control over the VM status than the `.spec.running` Boolean. For example, when the `.spec.running` Boolean is `true`, you cannot stop a VM by gracefully shutting down its operating system, because OpenShift Virtualization detects that the VM is not running and then restarts it.

The `.spec.runStrategy` parameter accepts any of the following values:

##### `RerunOnFailure`

Restarts the VM when it fails. If you gracefully shut down the operating system from inside the VM, then OpenShift Virtualization does not restart the VM.

##### `Always`

Ensures that the VM is always running. Using this value has the same effect as setting the `.spec.running` Boolean to `true`.

##### `Halted`

Ensures that the VM is not running. Using this value has the same effect as setting the `.spec.running` Boolean to `false`.

##### `Manual`

Performs no automatic action. If the VM fails, then OpenShift Virtualization does not restart it. You must manage the VM by using the `virtctl start` or `virtctl stop` command, or by using the OpenShift web console.

The `.spec.running` and `.spec.runStrategy` parameters are mutually exclusive. If you specify both parameters, then OpenShift Virtualization returns an error.

## Update the Run Strategy Parameters

To set the `.spec.running` or `.spec.runStrategy` parameters from the command line, edit the VM resource by using the `oc edit` command:

```
[user@host ~]$ oc edit vm mariadb-server
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be reopened with the relevant failures.
#
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
  dataVolumeTemplates:
  ...output omitted...
  runStrategy: RerunOnFailure
  template:
    metadata:
    ...output omitted...
```

To set the `.spec.running` or `.spec.runStrategy` parameters from the OpenShift web console, go to **Virtualization > VirtualMachines**.

Select the VM to edit, and go to the **YAML** tab.

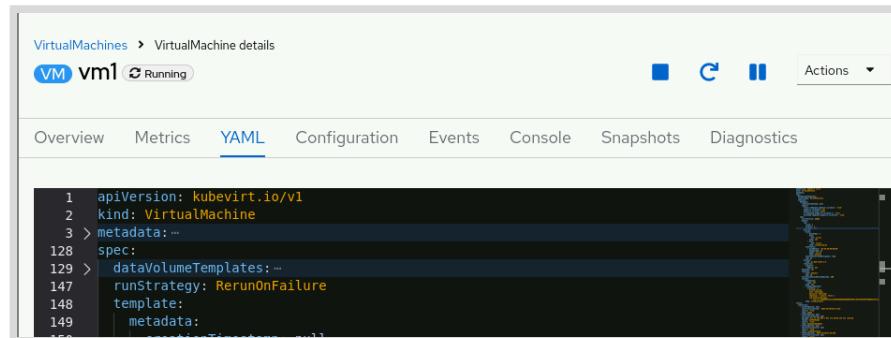


Figure 8.46: Run strategy settings

Then, use the YAML editor to set the `.spec.running` or the `.spec.runStrategy` parameter, and click **Save**.



## References

For more information about run strategies, refer to the *Run Strategies* section in the Red Hat OpenShift Container Platform 4.16 *Virtualization* guide at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#run-strategies](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#run-strategies)

For more information about node placement, refer to the *Specifying Nodes for OpenShift Virtualization Components* section in the Red Hat OpenShift Container Platform 4.16 *Virtualization* guide at [https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-node-placement-virt-components](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-node-placement-virt-components)

## ► Guided Exercise

# Control the Scheduling of a Virtual Machine

Configure the run strategy and node placement for a virtual machine.

### Outcomes

- Configure the run strategy for a VM.
- Configure node placement rules for a VM.
- Taint a node and then configure VMs to tolerate the taint.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start advanced-selector
```

### Instructions

- 1. As the `admin` user, configure the run strategy for the `server1` VM in the `advanced-selector` project.
- 1.1. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
  - 1.2. Go to `Virtualization > Virtual Machines`. Select the `advanced-selector` project from the `Project` list. Select the `server1` machine.
  - 1.3. Click the `Console` tab to open the VM's console.
  - 1.4. Click the console window and log in as the `developer` user with `developer` as the password.
  - 1.5. Shut down the operating system of the `server1` VM.

```
[developer@server1 ~]$ shutdown now
```

Because you shut down the operating system from inside the VM and the VM uses the `RerunOnFailure` run strategy, OpenShift Virtualization does not restart the VM.

- 1.6. Go to the `YAML` tab and update the value of the `spec.runStrategy` object to the `Always` value. Click `Save`.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...output omitted...
```

```
spec:  
...output omitted...  
runStrategy: Always  
template:  
  metadata:  
...output omitted...
```

Because you set the run strategy to the **Always** value, OpenShift Virtualization starts the VM.

► 2. Label the `worker01` and `worker02` nodes with the `workloads=production` label.

- 2.1. Go to **Compute > Nodes**, click the vertical ellipsis icon  $\vdots$  next to the `worker01` line, and then click **Edit labels**.

Name	Status	Roles	Pods	Memory	CPU	Filesystem	Created	⋮
master01	Ready	control-plane, master, worker	39	8.8 GiB / 23.47 GiB	1.491 cores / 4 cores	18.25 GiB / 69.72 GiB	Sep 11, 2024, 7:11 PM	⋮
master02	Ready	control-plane, master, worker	83	10.92 GiB / 23.47 GiB	1.313 cores / 4 cores	21.39 GiB / 69.72 GiB	Sep 11, 2024, 7:11 PM	⋮
master03	Ready	control-plane, master, worker	49	8.25 GiB / 23.47 GiB	0.965 cores / 4 cores	18.79 GiB / 69.72 GiB	Sep 11, 2024, 7:12 PM	⋮
worker01	Ready	worker	75	11.23 GiB / 15.62 GiB	1.082 cores / 6 cores	30.66 GiB / 39.78 GiB	Sep 24, 2024, 7:12 PM	⋮
worker02	Ready	worker	42	9.51 GiB / 15.62 GiB	0.682 cores / 6 cores	25.19 GiB / 39.78 GiB	⋮	<div style="border: 1px solid #ccc; padding: 2px;">Mark as unschedulable</div> <div style="border: 2px solid red; padding: 2px; margin-top: 2px;">Edit labels</div> <div style="margin-top: 2px;">Edit annotations</div> <div style="margin-top: 2px;">Edit Node</div> <div style="margin-top: 2px;">Delete Node</div>

- 2.2. Type `workloads=production` in the form and click **Save**.
- 2.3. Add the `workloads=production` label to the `worker02` node. Click the vertical ellipsis icon next to the `worker02` line, and then click **Edit labels**.
- 2.4. Type `workloads=production` in the form and click **Save**.
3. Configure a node placement rule for the `server1` VM by using a node selector for the `workloads=production` label.
  - 3.1. Go to **Virtualization > Virtual Machines** and select the `server1` VM.
  - 3.2. Go to **Configuration > Scheduling** and click the pencil icon under **Node Selector**.

The screenshot shows the 'Configuration' tab for a virtual machine named 'server1'. The 'Node selector' section is highlighted with a red box around the 'No selector' button. Other sections visible include 'Storage', 'Network', 'Scheduling', and 'SSH'.

- 3.3. Click Add Label to specify qualifying nodes and enter the workloads key and the production value. Click Save.

The screenshot shows the 'Node selector' configuration dialog. It displays a single entry: 'workloads' with value 'production'. Below the table, a message indicates '2 Nodes matching' with a link to 'View matching 2 Nodes'. At the bottom are 'Save' and 'Cancel' buttons.



### Note

An admission webhook warning is displayed when you add node placement rules through the web console. It is safe to ignore this warning throughout this exercise.

The screenshot shows an admission webhook warning message. It states: 'Admission Webhook Warning: VirtualMachine server1 violates policy 299 - "unknown field \"spec.template.template\""' and includes a 'Learn more' link.

- 4. Restart the **server1** VM and confirm that the VM is scheduled on either the **worker01** node or the **worker02** node.
- 4.1. Go to **Virtualization > VirtualMachines**. Click the vertical ellipsis icon next to the **server1** line, and then click **Restart**.
  - 4.2. Wait until the VM status is **Running** and then confirm that **Node** value is either the **worker01** node or the **worker02** node.

The identified node in your lab might differ from this example.

- ▶ 5. Create the server2 VM by cloning the server1 VM. Configure a pod affinity rule for the server2 VM to ensure that the server2 VM runs on the same node as the server1 VM.
  - 5.1. Click the vertical ellipsis icon next to the server1 line, and click **Clone**.
  - 5.2. Set the VM name to server2 and click **Clone**.
  - 5.3. Go to **Virtualization > VirtualMachines** and select the server2 VM.
  - 5.4. Go to **Configuration > Scheduling** and click the pencil icon under **Affinity rules**.

Scheduling and resource requirements

**Node selector**  
workloads=production

**Tolerations**  
0 Tolerations rules

**Affinity rules**  
0 Affinity rules

- 5.5. Click **Add affinity rule**. Set the Type to **Workload (pod) Affinity** and set the Condition to **Required during scheduling**. Leave the default value for the Topology key.

### Add affinity rule

Define an affinity rule. This rule will be added to the list of affinity rules applied to this workload.

Type \*

Workload (pod) Affinity ▾

Condition \*

Required during scheduling ▾

Topology key \*

kubernetes.io/hostname

Topology key must not be empty

- 5.6. Click **Add expression**. Set the **Key** to app and the **Operator** to In. Enter web for the **Values** and click **Create "web"** to confirm the value.

#### Workload labels

Select workloads that must have all the following expressions.

Key

Operator

Values

app

In

web

**+ Add expression**

**Create "web"**

Missing fields in Workload labels

**Save affinity rule** **Cancel**

- 5.7. Click **Save affinity rule**.

### Add affinity rule

Workload (pod) Affinity ▾

Condition \*

Required during scheduling ▾

Topology key \*

kubernetes.io/hostname

Topology key must not be empty

#### Workload labels

Select workloads that must have all the following expressions.

Key

Operator

Values

app

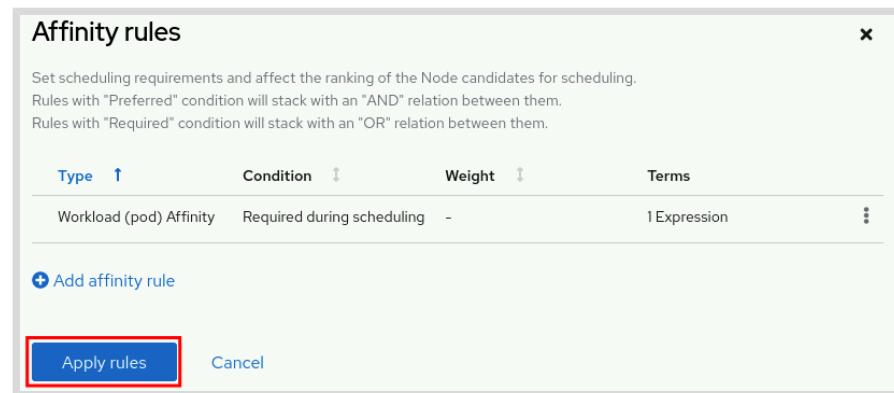
In

web X Enter value

**+ Add expression**

**Save affinity rule** **Cancel**

5.8. Click **Apply rules**.



- 6. Start the **server2** VM to confirm that the VM is scheduled to the same node as the **server1** VM.
- 6.1. Go to **Virtualization > VirtualMachines**.
  - 6.2. Click the vertical ellipsis icon next to the **server2** line, and click **Start**.
  - 6.3. Wait until the **server2** VM status is **Running**, and then confirm that the **Node** value for the **server2** VM is the same value as for the **server1** VM.

Name	Status	Conditions	Node
VM server1	Running	DataVolumesReady=True	worker02 (highlighted with a red box)
VM server2	Running	DataVolumesReady=True	worker02 (highlighted with a red box)

- 7. Taint the **worker02** node to prevent new workloads from running on the node. Apply the `key=virtualization:NoSchedule` taint to the node. By using the `NoSchedule` effect with the taint, existing workloads that are running on the **worker02** node are not impacted by the taint.

- 7.1. Go to **Compute > Nodes**. Select the **worker02** node and go to the **Details** tab.
- 7.2. Scroll down to the **Taints** section and click the pencil icon under **Taints**.

hyperv.node.kubevirt.io/tlbflush=true   cpu-feature.node.kubevirt.io/vmx=true-ctls=true   Show Less

**Taints** (highlighted with a red box)  
0 Taints

**Annotations**  
26 Annotations

- 7.3. Click **Add more**. Enter **key** for **Key**, **virtualization** for **Value**, and select **NoSchedule** for **Effect**.

Key	Value
key	virtualization

Effect  
NoSchedule

+ Add more

Cancel Save

- 7.4. Click **Save**.

- 8. Restart the **server1** VM. Confirm that the VM is scheduled on the **worker01** node.

- 8.1. Go to **Virtualization > VirtualMachines**.
- 8.2. Click the vertical ellipsis icon next to the **server1** line, and click **Restart**.
- 8.3. Wait until the VM status is **Running**, and then confirm that the **server1** VM is running on the **worker01** node.

VirtualMachines			
Name	Status	Conditions	Node
VM server1	Running	DataVolumesReady=True	N worker01
VM server2	Running	DataVolumesReady=True	N worker02

The node selector for the **server1** VM requires OpenShift to schedule the VM on nodes with the **workloads=production** label, which you applied to the **worker01** and **worker02** nodes.

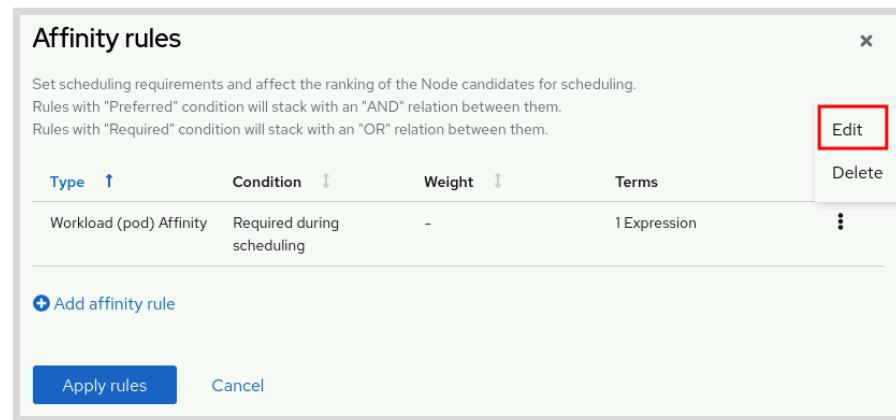
The taint on the **worker02** node prevents new workloads from running on the node unless the workloads have a matching toleration.

Because the **server1** VM is limited to the **worker01** and **worker02** nodes, but does not have a matching toleration for the **worker02** taint, OpenShift schedules the VM on the **worker01** node.

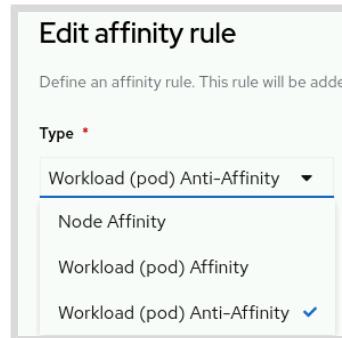
- 9. Prevent the **server2** VM from running on the same node as the **server1** VM. Change the pod affinity rule for the **server2** VM to a pod anti-affinity rule.

- 9.1. Go to **Virtualization > VirtualMachines** and select the **server2** VM.
- 9.2. Go to **Configuration > Scheduling** and click the pencil icon under **Affinity rules**.

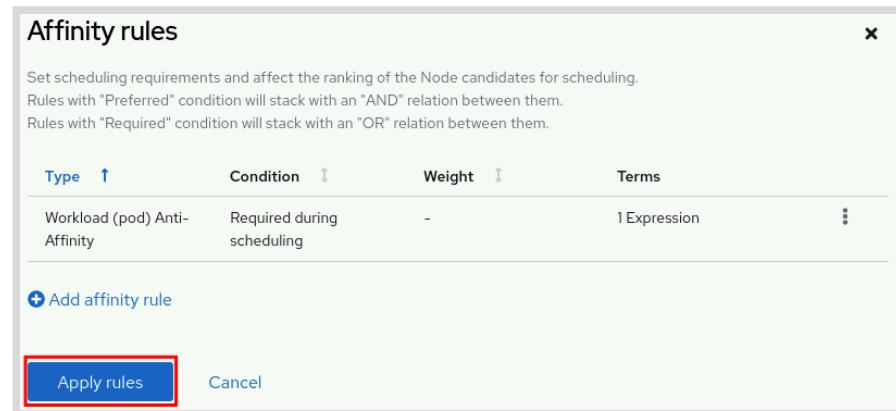
- 9.3. Click the vertical ellipsis icon in the Workload (pod) affinity line and select Edit.



- 9.4. Change the Type to Workload (pod) Anti-Affinity.

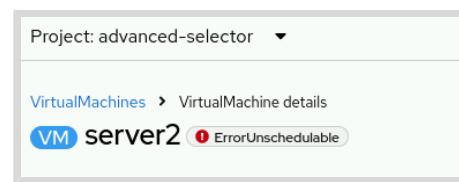


- 9.5. Click Save affinity rule and click Apply rules.



- 9.6. Click Actions > Restart to apply the changes.

The server2 VM displays an ErrorUnscheduable status.



This error occurs because OpenShift cannot schedule the VM on a node that satisfies the node placement rules.

The **server2** VM is configured with the `workloads=production` node selector, which requires OpenShift to schedule the VM on the **worker01** and **worker02** nodes.

The pod anti-affinity rule for the **server2** VM prevents the VM from running on the same node as the **server1** VM. Because the **server1** VM is running on the **worker01** node, the **server2** VM cannot run on the **worker01** node and must be scheduled to run on the **worker02** node.

However, the **worker02** node has a taint with the `NoSchedule` effect and the **server2** VM does not have a toleration for the taint. Thus, OpenShift cannot schedule the **server2** VM and an error displays.

- 10. Add a toleration to the **server2** VM for the `key=virtualization:NoSchedule` taint so that OpenShift can schedule the VM on the **worker02** node.

10.1. Go to **Virtualization > Virtual Machines** and select the **server2** VM.

10.2. Go to the **Configuration** tab and select **Scheduling**.

10.3. Click the pencil icon next under **Tolerations**.

VM server2 ErrorUnschedulable

Overview Metrics YAML Configuration Events

Scheduling and resource requirements

Node selector workloads=production

Tolerations 0 Tolerations rules

Affinity rules 1 Affinity rules

10.4. Click **Add toleration to specify qualifying Nodes**

Tolerations

⚠ Restart the VirtualMachine to apply changes.

Tolerations are applied to VirtualMachines, and allow (but do not require) the VirtualMachines to schedule onto Nodes with matching taints.

Add tolerations to allow a VirtualMachine to schedule onto Nodes with matching taints.

[Taints and Toleration documentation](#)

Add toleration to specify qualifying Nodes

Save Cancel

10.5. Set **Taint key** to `key`, set **Value** to `virtualization`, and set **Effect** to `NoSchedule`. Click **Save**

Taint key	Value	Effect
key	virtualization	NoSchedule

**1 Node matching**

- 10.6. Click Actions > Restart to apply the changes.
- 10.7. Go to Virtualization > VirtualMachines and confirm that the server2 VM is running on the worker02 node.

Name	Status	Conditions	Node
VM server1	Running	DataVolumesReady=True	N worker01
VM server2	Running	DataVolumesReady=True	N worker02

## Finish

As the student user on the workstation machine, confirm that you are in the /home/student directory and then use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advanced-selector
```

# Configuring Health Probes for Virtual Machines

## Objectives

- Configure health probes and a Watchdog device to monitor the health and responsiveness of a virtual machine and its services.

## Configure Health Probes

Applications might become unreliable for different reasons:

- Temporary connection loss
- Configuration errors
- Application errors

Developers can configure probes for monitoring their applications. A *probe* is a periodic check that monitors the health of an application. You can use probes for applications that run in pods or VMs.

For monitoring applications that run inside VMs, Kubernetes provides two types of probes:

### Readiness probes

Readiness probes determine whether the application is ready to serve requests. If the readiness probe fails, then Kubernetes prevents client traffic from reaching the application by removing the VM's IP address from the service resource.

Readiness probes help detect temporary issues that might affect your applications. For example, the application might be temporarily unavailable when it starts because it must establish initial network connections, load files in a cache, or perform initial tasks that take time to complete. The application might also periodically have to run long batch jobs, and become temporarily unavailable to clients.

Kubernetes continues to run the probe even after the application fails. When the application is available again, Kubernetes adds back the VM's IP address to the service so that the application can receive client traffic.

### Liveness probes

Liveness probes determine whether the application is in a healthy state. If the liveness probe detects an unhealthy state, then OpenShift Virtualization deletes the VMI resource and redeploys a new instance.

Liveness probes help detect unresponsive applications that require a complete restart of the VM to recover.

Kubernetes provides the following methods for monitoring your application with probes:

### HTTP GET

Some web applications provide a dedicated HTTP API endpoint to expose their status. Use the HTTP GET method for these applications.

The HTTP GET method performs GET requests to control the health of an application. The check is successful if the HTTP response code is in the 200-399 range.

**Note**

To use the HTTP GET method with a probe, you must configure your application to send HTTP response codes for requests that are received at the specified HTTP API endpoint.

**TCP socket**

Some applications, such as database servers, file servers, or application servers, provide their service through TCP ports instead of using HTTP. When using a TCP socket check, Kubernetes attempts to establish a TCP connection to the application's TCP port. If the connection is successful, then Kubernetes considers that the application is working.

**Guest agent ping**

The QEMU guest agent is a daemon that runs on VMs and provides information about the virtual machine, users, file systems, and secondary networks. The guest agent ping probe uses the `guest-ping` command to determine whether the QEMU guest agent is running on the virtual machine. The guest agent ping probe is a Technology Preview feature only and is outside the scope of this course.

## Add Probes to Virtual Machines

You can edit a VM resource to configure health probes. Use the `oc edit` command to add `readinessProbe` and `livenessProbe` sections to the VM resource from the command line. The following example configures a readiness probe for the `www1` VM:

```
[user@host ~]$ oc edit vm www1
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be reopened with the relevant failures.
#
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
    domain:
      ...output omitted...
    readinessProbe:
      httpGet: ①
        path: /health
        port: 8080
      initialDelaySeconds: 120 ②
      periodSeconds: 20 ③
      timeoutSeconds: 10 ④
      failureThreshold: 3 ⑤
      successThreshold: 3 ⑥
    ...output omitted...
```

- ➊ The `httpGet` section provides the parameters for the HTTP request. The `path` parameter provides the URL path to the probe endpoint.
- ➋ Specifies how long Kubernetes waits after the VM starts before probing the application. By default, Kubernetes immediately starts probing the VM.
- ➌ Specifies the frequency of the check. By default, Kubernetes checks every 10 seconds.
- ➍ Determines how long to wait for the probe to finish. If the probe exceeds this time, then Kubernetes assumes that it fails. By default, Kubernetes waits one second for the check to complete.
- ➎ Specifies the number of consecutive failures for Kubernetes to consider that the probe fails. By default, Kubernetes tries three times.
- ➏ Specifies the minimum consecutive successes after a probe fails for Kubernetes to consider that the probe succeeds. By default, Kubernetes sets the probe as successful after one successful check.

The following example configures a liveness probe for the `mariadb-server` VM:

```
[user@host ~]$ oc edit vm mariadb-server
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be reopened with the relevant failures.
#
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
    spec:
      domain:
        ...output omitted...
      livenessProbe:
        tcpSocket: ①
          port: 3306
        initialDelaySeconds: 120
        periodSeconds: 20
...output omitted...
```

- ➊ The `tcpSocket` section provides the parameters of the TCP socket. The `port` parameter provides the TCP socket port to test.

To add health probes from the Red Hat OpenShift web console, go to **Virtualization > VirtualMachines**. Select the VM to edit, and go to the **YAML** tab. Then, use the YAML editor to declare the probes, and click **Save**.

## Configure a Watchdog Device

The watchdog feature detects and restarts unresponsive operating systems. The feature relies on two components:

- A hardware component, which sets by default a 30-second timer. You can adjust this timer if needed. When the timer expires, the component triggers a system restart. On bare-metal machines, a chipset provides the feature. For VMs, OpenShift Virtualization emulates the chipset.
- A software component, which the operating system runs, to reset the hardware timer regularly to prevent it from expiring. If the operating system hangs, then the software component also hangs and cannot refresh the timer. Then, the timer expires and the system restarts.

Watchdogs monitor only operating systems, and do not detect application failures. For example, a watchdog does not trigger if the application hangs but the operating system is still responding.

Do not use watchdogs for applications that you can monitor by using liveness probes. In an unresponsive OS, all the running applications become unresponsive. Thus, a liveness probe indirectly detects an unresponsive OS when all the applications become unresponsive. Use watchdog monitoring for applications only when you have no other way to test your application.

## Activate Watchdog Monitoring

To activate watchdog monitoring in your VM, add the emulated watchdog device to your VM resource. You can do it by using the `oc edit` command to add a `device` section to the VM resource, as follows:

```
[user@host ~]$ oc edit vm mariadb-server
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be reopened with the relevant failures.
#
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
    domain:
      cpu:
        ...output omitted...
    devices:
      disks:
        - bootOrder: 1
          disk:
            bus: virtio
            name: mariadb-server
          interfaces:
            - macAddress: "02:48:09:00:00:00"
              masquerade: {}
              name: default
            networkInterfaceMultiqueue: true
          rng: {}
        watchdog:
          i6300esb: ①
```

```

action: poweroff ②
name: mywatchdog
...output omitted...

```

- ① OpenShift Virtualization can emulate only the Intel 6300ESB chipset.
- ② Action to perform when the watchdog triggers. The `poweroff` action stops the VM, and then OpenShift Virtualization restarts the VM according to the `.spec.running` or the `.spec.runStrategy` parameter. Other options for this parameter include the `reset` or `shutdown` actions. The references section provides more details about the watchdog actions.

To activate watchdog monitoring from the Red Hat OpenShift web console, go to **Virtualization** > **VirtualMachines**. Select the VM to edit, and go to the **YAML** tab. Then, use the YAML editor to create the watchdog device, and click **Save**.

Watchdog monitoring works only for operating systems that support the Intel 6300ESB chipset. Red Hat Enterprise Linux automatically loads the kernel module that supports the device.



## References

For more information about OpenShift Virtualization probes, refer to the *Monitoring Virtual Machine Health* section in the Red Hat OpenShift Container Platform 4.16 *Virtualization* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-monitoring-vm-health](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-monitoring-vm-health)

For more information about Kubernetes probes, refer to the *Monitoring Application Health by Using Health Checks* chapter in the Red Hat OpenShift Container Platform 4.16 *Building Applications* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/building\\_applications/index#application-health](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/building_applications/index#application-health)

For more information about configuring a watchdog device, refer to the *Defining a Watchdog* section in the Red Hat OpenShift Container Platform 4.16 *Virtualization* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#watchdog\\_virt-monitoring-vm-health](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#watchdog_virt-monitoring-vm-health)

`watchdog(8)` manual page

## ► Guided Exercise

# Configure Health Probes for Virtual Machines

Create health probes for virtual machines behind a service to enable Kubernetes to detect an application's availability and to modify service networking accordingly.

### Outcomes

- Declare readiness and liveness probes to VMs.
- Add watchdog devices to VMs.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start advanced-probe
```

### Instructions

The `lab` command creates the `advanced-probe` namespace and starts four virtual machines in that namespace:

- The `www1` and `www2` VMs, which host a web application
- The `watch1` VM, which hosts a test VM
- The `mariadb-server` VM, which hosts a MariaDB database

The `lab` command creates a service that dispatches the client requests between the two web servers, and also creates a route resource so the clients can reach the web application by using the `http://www-advanced-probe.apps.ocp4.example.com` URL.

- 1. As the `vt-admin` user, confirm that the four VMs are running in the `advanced-probe` project.

- 1.1. From a command-line window, log in to your Red Hat OpenShift cluster as the `vt-redhatocp` user with `vt-redhatocp` as the password.

```
[student@workstation ~]$ oc login -u vt-admin -p vt-redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Change to the `advanced-probe` project.

```
[student@workstation ~]$ oc project advanced-probe
Now using project "advanced-probe" on server "https://api.ocp4.example.com:6443".
```

13. Confirm that the `mariadb-server`, `watch1`, `www1`, and `www2` VMs are running.

```
[student@workstation ~]$ oc get vm
NAME        AGE     STATUS    READY
mariadb-server  13m   Running   True
watch1       15m   Running   True
www1         18m   Running   True
www2         16m   Running   True
```

- ▶ 2. Review the service and the route that the `lab` command deployed. Confirm that Kubernetes dispatches the requests to the `http://www-advanced-probe.apps.ocp4.example.com` URL between the `www1` and `www2` VMs.

- 2.1. Use the `oc` command to list the VMI resources in the `advanced-probe` project. Note the IP addresses of the `www1` and `www2` VM instances. The IP addresses might differ in your environment.

```
[student@workstation ~]$ oc get vmi
NAME        AGE     PHASE    IP           NODENAME    READY
mariadb-server  12m   Running  10.9.0.26  master02   True
watch1       15m   Running  10.8.2.50  worker02   True
www1         18m   Running  10.8.2.47  worker02   True
www2         16m   Running  10.11.0.40  worker01   True
```

- 2.2. Confirm that the `front` service exists.

```
[student@workstation ~]$ oc get service
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
front    ClusterIP  172.30.49.84  <none>          80/TCP      12m
```

- 2.3. List the service endpoints. Notice that the service uses the IP addresses of the two VMs for its endpoints.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS      AGE
front    10.11.0.40:80,10.8.2.47:80  14m
```

- 2.4. Confirm that the `www-front` route exists. Notice that the DNS name for the route is `www-advanced-probe.apps.ocp4.example.com`. This DNS name points to the `front` service.

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT      PATH      SERVICES      PORT  ...
www-front  www-advanced-probe.apps.ocp4.example.com  front      80     ...
```

- 2.5. Run the `curl` command several times to confirm that Kubernetes dispatches the requests between the two VMs. For distinguishing the VMs, the web servers return a web page that includes the hostname.

```
[student@workstation ~]$ curl http://www-advanced-probe.apps.ocp4.example.com
Welcome to www2
```

```
[student@workstation ~]$ curl http://www-advanced-probe.apps.ocp4.example.com
Welcome to www2

[student@workstation ~]$ curl http://www-advanced-probe.apps.ocp4.example.com
Welcome to www1

[student@workstation ~]$ curl http://www-advanced-probe.apps.ocp4.example.com
Welcome to www1

[student@workstation ~]$ curl http://www-advanced-probe.apps.ocp4.example.com
Welcome to www2
```

- 3. The web servers that run inside the `www1` and `www2` VMs expose the `/health` endpoint to monitor the application status. Add a readiness probe to the `www1` and `www2` VMs that uses HTTP GET requests to test the `/health` endpoint.
- 3.1. Add a readiness probe to the `www1` VM. Use the `oc edit` command to declare the probe as follows, and press `:wq!` to save the changes. You can copy and paste the `readinessProbe` section from the `~/D0316/labs/advanced-probe/readiness.yaml` file.

```
[student@workstation ~]$ oc edit vm www1
...output omitted...
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
metadata:
...output omitted...
spec:
  architecture: amd64
  domain:
    ...output omitted...
  readinessProbe:
    httpGet:
      path: /health
      port: 80
    initialDelaySeconds: 10
    periodSeconds: 5
    timeoutSeconds: 2
    failureThreshold: 2
    successThreshold: 1
  evictionStrategy: LiveMigrate
  hostname: www1
...output omitted...
```

- 3.2. Restart the `www1` VMI so Red Hat OpenShift Virtualization recognizes the new probe.

```
[student@workstation ~]$ virtctl restart www1
VM www1 was scheduled to restart.
```

**Chapter 8 |** Advanced Virtual Machine Management

- 3.3. Wait a few minutes until the `www1` VMI is again in the Running state.

```
[student@workstation ~]$ oc get vmi www1
NAME    AGE    PHASE     IP          NODENAME   READY
www1   28s    Running   10.8.2.70  worker02  True
```

- 3.4. Add the readiness probe to the `www2` VM. Use the `oc edit` command to declare the probe as follows, and press `:wq!` to save the changes. You can copy and paste the `readinessProbe` section from the `~/D0316/labs/advanced-probe/readiness.yaml` file.

```
[student@workstation ~]$ oc edit vm www2
...output omitted...
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
    architecture: amd64
    domain:
      ...output omitted...
    readinessProbe:
      httpGet:
        path: /health
        port: 80
      initialDelaySeconds: 10
      periodSeconds: 5
      timeoutSeconds: 2
      failureThreshold: 2
      successThreshold: 1
    evictionStrategy: LiveMigrate
    hostname: www2
...output omitted...
```

- 3.5. Restart the `www2` VMI so Red Hat OpenShift Virtualization recognizes the new probe.

```
[student@workstation ~]$ virtctl restart www2
VM www2 was scheduled to restart.
```

- 3.6. Wait a few minutes until the `www2` VMI is again in the Running state.

```
[student@workstation ~]$ oc get vmi www2
NAME    AGE    PHASE     IP          NODENAME   READY
www2   26s    Running   10.10.0.14  master03  True
```

- ▶ 4. Test the probe by stopping the `httpd` service that is running inside the `www1` VM.

- 4.1. Open a new command-line window, and run the `~/D0316/labs/advanced-probe/loop.sh` script. This script runs the `curl` command in an infinite loop. Leave the command running.

```
[student@workstation ~]$ ~/D0316/labs/advanced-probe/loop.sh
Welcome to www2
Welcome to www2
Welcome to www1
Welcome to www1
Welcome to www2
Welcome to www2
Welcome to www1
...output omitted...
```

Notice that Kubernetes dispatches the requests between the two VMs.

- 4.2. Use the `virtctl` command to connect to the console of the `www2` VM.

```
[student@workstation ~]$ virtctl vnc www2
```

- 4.3. Log in as the `root` user with `redhat` as the password.

- 4.4. Stop the `httpd` service.

```
[root@www2 ~]# systemctl stop httpd
```

Leave the console open.

- 4.5. Switch back to the command-line window on the `workstation` machine that is running the `loop.sh` script. After a few seconds, the output of the `loop.sh` script shows that Kubernetes sends the requests only to the `www1` VM.

Press `Ctrl+C` to stop the script.

```
...output omitted...
Welcome to www2
Welcome to www2
Welcome to www1
Welcome to www2
Welcome to www1
^C
```

- 4.6. Use the `oc` command to confirm that the `front` service has only one remaining endpoint.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS      AGE
front    10.8.2.51:80    28m
```

- 4.7. Switch back to the `www2` VM console and restart the `httpd` service. Log out of and then close the console when done.

```
[root@www2 ~]# systemctl start httpd
[root@www2 ~]# logout
```

- 4.8. Return to the command line on the `workstation` machine. Run the `~/D0316/labs/advanced-probe/loop.sh` script to confirm that Kubernetes dispatches the requests between the two VMs again.

Press `Ctrl+C` to stop the script.

```
[student@workstation ~]$ ~/D0316/labs/advanced-probe/loop.sh
Welcome to www2
Welcome to www2
Welcome to www1
Welcome to www1
Welcome to www2
Welcome to www2
Welcome to www2
^C
```

- 4.9. Use the `oc` command to confirm that the `front` service has two endpoints.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS          AGE
front     10.11.0.45:80,10.8.2.51:80   30m
```

- ▶ 5. The MariaDB database that is running inside the `mariadb-server` VM listens on TCP port 3306. Add a liveness probe that tests the service by sending requests to the TCP socket.
- 5.1. Add a liveness probe to the `mariadb-server` VM. Use the `oc edit` command to declare the probe as follows, and press `:wq!` to save the changes. You can copy and paste the `livenessProbe` section from the `~/D0316/labs/advanced-probe/liveness.yaml` file.

```
[student@workstation ~]$ oc edit vm mariadb-server
...output omitted...
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
    spec:
      architecture: amd64
      domain:
        ...output omitted...
      livenessProbe:
        tcpSocket:
          port: 3306
```

```
initialDelaySeconds: 10
periodSeconds: 5
evictionStrategy: LiveMigrate
hostname: mariadb-server
...output omitted...
```

- 5.2. Restart the `mariadb-server` VMI resource so Red Hat OpenShift Virtualization recognizes the new probe.

```
[student@workstation ~]$ virtctl restart mariadb-server
VM mariadb-server was scheduled to restart
```

- 5.3. Wait a few minutes until the VMI is again in the Running state.

```
[student@workstation ~]$ oc get vmi mariadb-server
NAME          AGE     PHASE   IP           NODENAME   READY
mariadb-server 35s    Running  10.9.0.13  master02  True
```

- ▶ 6. Test the probe by stopping the `mysql` service that is running inside the `mariadb-server` VM.

- 6.1. Use the `virtctl` command to connect to the console of the `mariadb-server` VM.

```
[student@workstation ~]$ virtctl vnc mariadb-server
```

- 6.2. Log in as the `root` user with `redhat` as the password.

- 6.3. Stop the `mysql` service.

```
[root@localhost ~]# systemctl stop mysql
```

- 6.4. Because the liveness probe fails, the VM restarts after a few seconds.

- ▶ 7. Add a watchdog device to the `watch1` VM.

- 7.1. Add a watchdog device to the `watch1` VM. Use the `oc edit` command to declare the probe as follows, and press `:wq!` to save the changes. You can copy and paste the `watchdog` section from the `~/D0316/labs/advanced-probe/watchdog.yaml` file.

```
[student@workstation ~]$ oc edit vm watch1
...output omitted...
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
```

```

architecture: amd64
domain:
cpu:
  ...output omitted...
devices:
disks:
- bootOrder: 1
disk:
  bus: virtio
  name: watch1
interfaces:
- macAddress: 02:7e:be:00:00:02
  masquerade: {}
  name: default
networkInterfaceMultiqueue: true
rng: {}
watchdog:
  i6300esb:
    action: poweroff
    name: testwatchdog
machine:
  type: pc-q35-rhel9.4.0
...output omitted...

```

- 7.2. Restart the `watch1` VM resource so Red Hat OpenShift Virtualization recognizes the new probe.

```
[student@workstation ~]$ virtctl restart watch1
VM watch1 was scheduled to restart
```

- 7.3. Wait a few minutes until the VM is again in the `Running` state.

```
[student@workstation ~]$ oc get vmi watch1
NAME      AGE     PHASE     IP          NODENAME   READY
watch1    9s      Running   10.9.0.41  master02  True
```

## ▶ 8. Test the watchdog device.

- 8.1. Use the `virtctl` command to connect to the console of the `watch1` VM.

```
[student@workstation ~]$ virtctl vnc watch1
```

- 8.2. Log in as the `root` user with `redhat` as the password.

- 8.3. The Linux kernel regularly resets the watchdog hardware timer by writing to the `/dev/watchdog` device file. However, if the kernel detects that another process accesses the file in write mode, then it stops refreshing the timer. The kernel no longer considers it necessary to manage the timer because the new process took over this function.

The `watchdog` service is an example of a program that writes to the `/dev/watchdog` device file. Because the RPM package that provides the `watchdog` service is not available on the `watch1` VM, use the `echo` command to simulate a

process that opens the `/dev/watchdog` device file in write mode. The kernel stops refreshing the timer.

```
[root@localhost ~]# echo > /dev/watchdog
[ 156.709341] watchdog: watchdog0: watchdog did not stop!
```

The preceding echo command returns the kernel error message about the watchdog not being reset. After 30 seconds, the Intel 6300ESB emulated chipset restarts the VM.



### Note

Confirm that you are in the `/home/student` directory on the **workstation** machine before you use the `lab` command to complete the exercise.

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advanced-probe
```

# Live Migrating a Virtual Machine

## Objectives

- Manage live migration of a virtual machine from one node in the cluster to another.

## Live Migration

During maintenance or troubleshooting, you might need to migrate a VM to a different cluster node. *Live migration* is the process of moving a running VM to a different cluster node without disrupting access or the virtual workload, and is similar to the VMware vMotion feature for VMs. The destination node for a live migration is determined by the node placement rules for a VM, such as node selectors, tolerations, and affinity rules.

To perform a live migration, the cluster must have sufficient RAM and network bandwidth, and the VM must meet the following conditions:

- The underlying PVC must use ReadWriteMany (RWX) access mode. If you do not configure the access mode and volume mode parameters in the `DataVolume` object for a VM, then the defined values in the storage profile for the storage class are applied to the PVC.
- The pod network must not be configured with the `bridge` binding type.
- Ports 49152 and 49153 must be available in the VM's `virt-launcher` pod; live migration fails if these ports are specified in a `masquerade` network interface.
- If a VM defines a CPU model, then the cluster must have an available node that supports the CPU.

During installation, the Red Hat OpenShift Virtualization operator creates a storage profile resource for each storage class.

```
[user@host ~]$ oc get storageprofiles.cdi.kubevirt.io
NAME                      AGE
nfs-storage                4m1s
ocs-external-storagecluster-ceph-rbd    4m1s
ocs-external-storagecluster-ceph-rbd-virtualization 4m1s
ocs-external-storagecluster-ceph-rgw     4m1s
ocs-external-storagecluster-cephfs      4m
openshift-storage.noobaa.io            4m
```

You can modify the storage profile resource to configure the default access and volume modes for a particular storage class.

You can set the RWX access mode for a storage class by using the `oc edit` command:

```
[user@host ~]$ oc edit storageprofile nfs-storage -n openshift-cnv
...output omitted...
Spec:
  Claim Property Sets:
    Access Modes:
      ReadWriteMany
    Volume Mode: Filesystem
...output omitted...
```

**Note**

Modifications to the storage profiles affect new PVCs and not existing PVCs. You cannot modify the access mode of an existing PVC.

## Limitations of Live Migrations

You can adjust the default settings for live migration limits and timeouts by editing the HyperConverged CR. Use the `oc edit hco -n openshift-cnv kubevirt-hyperconverged` command to modify the default parameters. To restore the default value for any `spec.liveMigrationConfig` parameter, delete the key-value pair and save the file.

### `parallelMigrationsPerCluster`

The maximum number of migrations that can run in parallel in the cluster. **The default value is 5.**

**Note**

Your cluster must have enough spare memory to support node drains that result in live migrations. You can use the following formula to determine the approximate required spare memory:

Product of (maximum number of nodes that can drain in parallel) and (highest total VM memory request allocations across nodes)

### `parallelOutboundMigrationsPerNode`

The maximum number of parallel outbound migrations per node. **The default value is 2.**

### `bandwidthPerMigration`

Limits the bandwidth, in MiB/s, that is used for each migration. **The default value is 0, which is unlimited.**

### `completionTimeoutPerGiB`

If the migration exceeds the defined time, in seconds per GiB of memory, then the migration is canceled. The size of the migrating disks is included in the calculation if you use the `BlockMigration` migration method. **The default value is 800.**

### `progressTimeout`

The migration is canceled if the memory copy fails to make progress in this time (in seconds). **The default value is 150.**

You can change some live migration settings through the OpenShift web console. From the web console, go to **Virtualization > Overview**. Select the **Settings** tab and then select **Cluster**. Go to

**General Settings > Live migration** to define the maximum number of migrations per cluster or per node. The following example changes the maximum number of migrations per cluster from the default value of 5 to 10.

The screenshot shows the 'Virtualization' interface with the 'Settings' tab selected. Under 'General settings', the 'Live migration' section is expanded. It displays the 'Installed version' as 4.16.2, 'Update status' as 'Up to date', and the 'Channel' as 'stable'. Below this, the 'Max. migrations per cluster' is set to 10, and the 'Max. migrations per node' is set to 2. A note indicates that a dedicated network for live migrations can also be defined via the 'Live migration network' dropdown, which is currently set to 'Primary live migration network'.

**Figure 8.68: Live migration settings**

You can also define a dedicated network for live migrations by using the **General Settings > Live migration** page.

## Configuring a Dedicated Network for Live Migrations

Red Hat recommends that you configure a dedicated network for live migrations. A dedicated network limits the effects of network saturation on workloads during migration.

To create a dedicated network for live migration, you must first use the Kubernetes NMState operator to configure a bridge network on your cluster nodes. The node NICs that are used for the bridge network must be connected to the same VLAN. You must then create a network attachment definition for the dedicated network in the `openshift-cnv` project.



### Note

The Kubernetes NMState operator and network attachment definitions are described elsewhere in this course.

After you create the network attachment definition for the bridge, edit the live migration configuration in the HyperConverged CR. You can use the `oc edit hco -n openshift-cnv kubevirt-hyperconverged` command to specify the live migration network.

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
...output omitted...
spec:
```

```
...output omitted...
liveMigrationConfig:
  completionTimeoutPerGiB: 800
  network: live-migration-network ①
  parallelMigrationsPerCluster: 5
  parallelOutboundMigrationsPerNode: 2
  progressTimeout: 150
...output omitted...
```

- ① Specify the name of the network attachment definition object to use for live migrations.

You can also use the OpenShift web console to select the dedicated network for live migrations. From the web console, go to **Virtualization** > **Overview**. Select the **Settings** tab and then select **Cluster**. Go to **General Settings** > **Live migration**. Select the network attachment defintion for your dedicated migration network from the **Live migration network** list.

## Live Migration Strategies

A live migration transfers, or converges, a source VM's state, which is mostly its memory, to a target VM on another node. During migration, the VM state is copied from the source VM to the target VM in chunks until the VM state is fully converged on the target VM. If sufficient resources are available, such as network bandwidth and CPU, then migration is successful. Red Hat OpenShift Virtualization provides a choice of three migration strategies to ensure a successful live migration.

### pre-copy

The default strategy for live migrations. A target VM is created but the guest keeps running on the source VM. The VM state is then copied to the target VM until the entire VM state is transferred. When the VM state is transferred to the target VM, the guest starts running from the target VM and the source VM is deleted.

### post-copy

A target VM is created and the guest starts running on the target VM. The source VM begins to send its state in chunks to the target VM. If the guest that is running on the target VM requests memory that is currently available only on the source VM, then that chunk of memory is copied to the target VM. The source VM is deleted after its state is copied to the target VM. This strategy consumes less bandwith, but is slower than the **pre-copy** method. This strategy also prevents a single source of truth for the VM state because both the target and source VMs contain parts of the VM's state. If either VM crashes, then the state cannot be recovered.

### auto-converge

This strategy enhances the **pre-copy** strategy to ensure a successful migration. Similar to the **pre-copy** strategy, the guest keeps running on the source VM while the VM state is copied to the target VM. However, if the migration cannot complete before the source VM mutates or modifies the VM state, then the guest CPU is throttled on the source VM. Guest CPU throttling is gradually increased until the migration is successful. If the migration can complete fast enough, then the guest CPU is not throttled or is negligibly throttled.

The **pre-copy** strategy is appropriate for most situations. However, the **pre-copy** strategy brings a risk that by the time a VM state chunk is copied from the source VM to the target VM, the guest on the source VM already mutated or modified the chunk. The **post-copy** and **auto-converge** strategies can mitigate that risk to ensure a sucessful convergence. Use the **post-copy** and **auto-converge** strategies for migrations that might not converge easily, such as high-consumption database servers.

To change the migration strategy at the cluster level, you must edit the HyperConverged CR in the `openshift-cnv` project. In the `.spec.liveMigrationConfig` parameter, locate the `allowAutoConverge` and `allowPostCopy` keys and set their value to `true` for your preferred strategy. The following example configures live migration to use the auto-converge strategy:

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
  ...output omitted...
spec:
  ...output omitted...
  liveMigrationConfig:
    allowAutoConverge: true
    allowPostCopy: false
  ...output omitted...
```

You can also define a live migration strategy for individual or groups of VMs by creating a *live migration policy*.

## Live Migration Policy

Sometimes the live migration configuration set at the cluster level is not sufficient for a particular workload. In those situations, OpenShift Virtualization provides the `MigrationPolicy` CR, which defines a live migration policy.

A live migration policy declares the migration configuration to apply to a VM or to a group of VMs. You can use a live migration policy to define the migration strategy, set the bandwidth per migration, and set completion timeout (in seconds) for a VM or a group of VMs. Live migration policies use VM and project labels as selectors for the policy. To apply a migration policy to a specific group of VMs, all labels on the group of VMs must match the labels of the policy.

In the following example, the live migration policy sets the auto-converge migration strategy for a group of VMs:

```
apiVersion: migrations.kubevirt.io/v1alpha1
kind: MigrationPolicy
metadata:
  name: prod-database-policy ①
spec:
  allowAutoConverge: true
  selectors:
    namespaceSelector: ②
      database-workloads: "True"
    virtualMachineInstanceSelector: ③
      kubevirt.io/environment: "production"
```

- ① The name of the migration policy
- ② The labels for the projects that are impacted by the migration policy
- ③ The labels for the VMs that are impacted by the migration policy

You can use the `oc create migrationpolicy -f migration-policy.yaml` command to create the live migration policy.

You can also create live migration policies through the OpenShift web console. To create a live migration policy through the web console, go to **Virtualization > Migration Policies**. Click **Create MigrationPolicy** and select **With form**. Complete the form and click **Create**.

## Starting, Monitoring, and Canceling a Live Migration

As a cluster administrator, you can also manually migrate a VMI.

To start a live migration by using the Red Hat OpenShift web console, go to **Virtualization > VirtualMachines**. Select your project from the **Project** list. Locate the row with the VM to migrate, and click the vertical ellipsis icon : at the right of the row. Click **Migrate**. Wait a few moments for the migration to complete.

You can also click the VM's name to go to the VM's **Overview** tab. Click **Actions > Migrate**. Wait a few moments for the migration to complete.

To migrate from the command line, you can create a `VirtualMachineInstanceMigration` manifest for the VMI as shown in the following example:

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstanceMigration
metadata:
  name: migration-job
spec:
  vmiName: VMI name
```

Use the `oc create -f manifest-name.yaml -n project-name` command to start the live migration of the VMI that is specified in the manifest.

You can also use the `virtctl migrate vm-name -n project-name` command to start a live migration. The `virtctl migrate` command generates a `VirtualMachineInstanceMigration` object for the VMI.

```
[user@host ~]$ virtctl migrate vm1 -n web-servers
VM vm1 was scheduled to migrate
```

You can view the status of a live migration on the web console by going to the **Overview** tab and then the **Details** section of the VM.

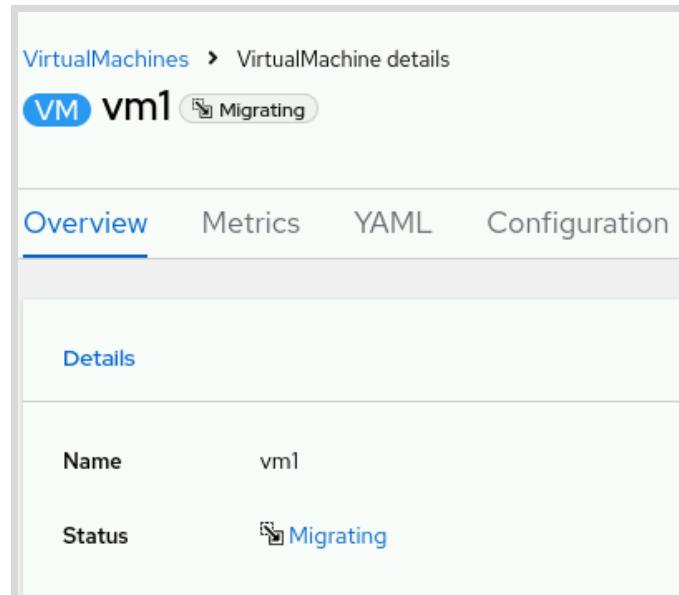


Figure 8.69: VM migration status

You can also view the migration metrics and status from **Virtualization > Overview** on the **Migrations** tab.

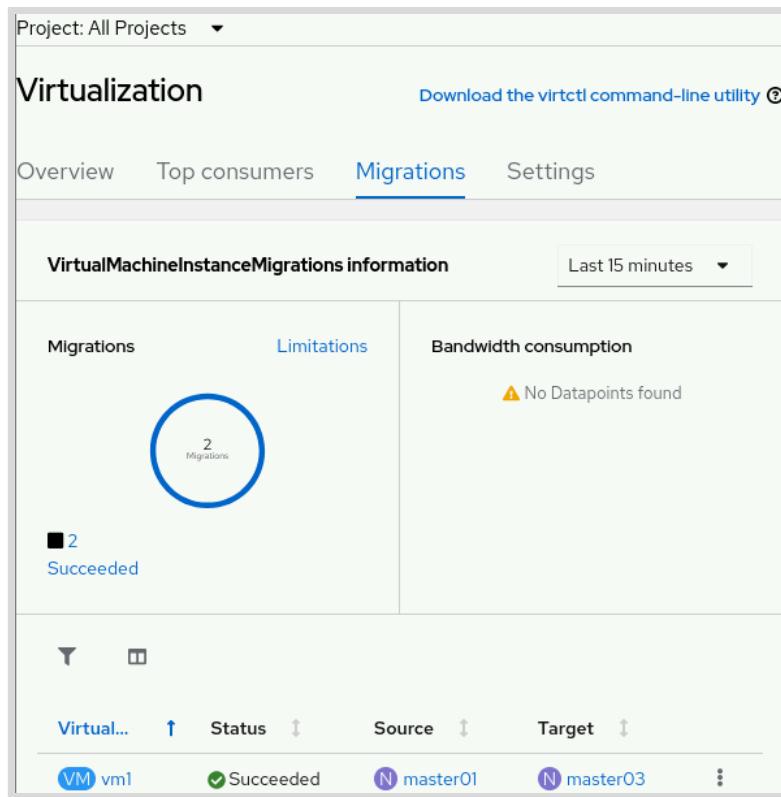


Figure 8.70: Live migration metrics

You can use the command line to monitor the status of a VM's live migration with the `oc describe vmi vm1-name` command or with the `oc get vmim` command:

```
[user@host ~]$ oc get vmim -n web-servers
NAME          PHASE      VMI
kubevirt-migrate-vm-2wt72   Succeeded   vm1
```

To cancel a live migration in the web console, go to **Virtualization > VirtualMachines**. Select your project from the **Project** list. Click the vertical ellipsis icon at the right of the VM to migrate, and click **Cancel Migration**.

To cancel a live migration from the command line, you can delete the `VirtualMachineInstanceMigration` object that is associated with the migration by using the `oc delete vmim migration-job` command. You can alternatively use the `virtctl migrate-cancel vm-name -n project-name` command to cancel a live migration.



## References

For more information, refer to the *Live Migration* chapter in the Red Hat OpenShift Container Platform 4.16 documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#live-migration](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#live-migration)

For more information about configuring a dedicated network for live migrations, refer to the *Networking* chapter in the Red Hat OpenShift Container Platform 4.16 documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-configuring-secondary-network-vm-live-migration\\_virt-dedicated-network-live-migration](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-configuring-secondary-network-vm-live-migration_virt-dedicated-network-live-migration)

## ► Guided Exercise

# Live Migrate a Virtual Machine

Move a running virtual machine from one node in the cluster to another by using live migration.

### Outcomes

- Observe and set the access mode for a storage profile.
- Initiate a live migration through the web console and command line.
- Monitor the progress of a live migration.
- Cancel a live migration through the web console and command line.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start advanced-migrate
```

### Instructions

- 1. As the `admin` user, confirm that the `vm1` machine is running on the `advanced-migrate` project.
- 1.1. Open a command-line window and log in to the OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
  - 1.4. Go to `Virtualization > VirtualMachines`. Select the `advanced-migrate` project from the `Project` list. Confirm that the `vm1` machine is running.
- 2. Confirm that the `vm1` machine is not migratable. Determine the storage class of the VM's empty `tempdata` disk.

- 2.1. Click the vertical ellipsis icon : at the right of the `vm1` machine and then confirm that **Migrate** is disabled.

The **Status** and **Conditions** columns show the **Not migratable** and **LiveMigratable=False** values for the `vm1` machine.

The screenshot shows the Red Hat OpenShift Virtual Machines interface. At the top, there's a navigation bar with the Red Hat OpenShift logo, user information (admin), and a 'Create' button. Below it, the project is set to 'advanced-migrate'. The main area is titled 'VirtualMachines' and shows a single entry for 'vm1'. The table has columns: Name, Status, Conditions, and Node. The 'Name' column shows 'VM vm1', 'Status' shows 'Running', 'Conditions' shows 'Not migratable' and 'LiveMigratable=False' (both highlighted with a red box), and 'Node' shows 'master02'. To the right of the table is a context menu with options: Stop, Restart, Pause, Clone, and Migrate (also highlighted with a red box). Below the table, a note states: 'The live migration is not available because the empty tempdata PersistentVolumeClaim (PVC) is not configured with the ReadWriteMany access mode.'

- The live migration is not available because the empty `tempdata` PersistentVolumeClaim (PVC) is not configured with the `ReadWriteMany` access mode.
- 2.2. Identify the storage class for the `tempdata` disk. Select the `vm1` machine. Go to the **Configuration** tab. In the **Storage** section, notice that the `tempdata` disk uses the `nfs-storage` storage class.
  - ▶ 3. Retrieve the current default storage class settings for Red Hat OpenShift Virtualization. Configure the `nfs-storage` storage class to use the `ReadWriteMany` access mode.



### Note

In your cluster, both the `ocs.openshift.io` and `cdi.kubevirt.io` API groups provide a custom resource definition (CRD) for the `storageprofile` object. Depending on your configuration, you might need to specify the API group for the storage profile resource in the following commands. If the following commands do not work, then you must modify the commands to specify the `cdi.kubevirt.io` API group for the `storageprofile` object. You can do this modification by replacing the `storageprofile` object in the commands with either the `storageprofile.cdi` or `storageprofile.cdi.kubevirt.io` objects.

- 3.1. From the command line, execute the `oc describe` command to retrieve the `nfs-storage` storage profile. Notice that the `nfs-storage` storage class defaults to the `ReadWriteOnce` access mode.

```
[student@workstation ~]$ oc describe storageprofile nfs-storage
...output omitted...
Spec:
  Claim Property Sets:
    Access Modes:
      ReadWriteOnce
    Volume Mode: Filesystem
...output omitted...
```

- 3.2. Change the nfs-storage access mode to ReadWriteMany by using the `oc edit` command.

```
[student@workstation ~]$ oc edit storageprofile nfs-storage
...output omitted...
spec:
  claimPropertySets:
    - accessModes:
      - ReadWriteMany
    volumeMode: FileSystem
...output omitted...
```



### Note

You can also use the `oc patch` command to update the storage profile settings.

You can copy and paste the command from the `~/D0316/labs/advanced-migrate/commands.txt` file.

```
[student@workstation ~]$ oc patch -n openshift-cnv \
storageprofile nfs-storage --type json -p='[
{"op": "add", "path": "/spec/claimPropertySets",
"value": [{"accessModes": ["ReadWriteMany"], "volumeMode": "Filesystem"}]}]'
```

- 3.3. Confirm that the nfs-storage storage profile is updated with the correct access mode.

```
[student@workstation ~]$ oc describe storageprofile nfs-storage
...output omitted...
Spec:
  Claim Property Sets:
    Access Modes:
      ReadWriteMany
    Volume Mode: Filesystem
...output omitted...
```

- 4. Delete the empty tempdata disk from the `vm1` machine. Attach a new 2 GiB blank disk named `newtempdata` that uses the `nfs-storage` storage class.

- 4.1. You must stop the `vm1` machine before you can modify the VM's disks. In the OpenShift web console, go to **Virtualization > VirtualMachines**. Select the `vm1` machine and then click **Actions > Stop**. Wait for the VM to stop.

- 4.2. Go to the **Configuration** tab. In the **Storage** section, click the vertical ellipsis icon at the right of the `tempdata` disk and then click **Detach**. Click **Detach** to confirm the operation.
  - 4.3. Click **Add disk**. Select **Empty disk (blank)** as the source and enter `newtempdata` for the name.  
Set the **Size** field to the `2 GiB` value and leave the `virtio` value for the **Interface** field. Confirm that the **StorageClass** field is set to the `nfs-storage` value, and select **Apply optimized StorageProfile settings**. Click **Save** to add the disk.
  - 4.4. Click **Actions > Start** to power on the VM. Wait a few moments for the operation to complete.
- ▶ 5. Create a live migration policy for the `vm1` machine that uses the `post-copy` migration strategy.
- 5.1. In the OpenShift web console, go to **Virtualization > MigrationPolicies**.
  - 5.2. Click **Create MigrationPolicy** and select **With form**.
  - 5.3. Use `post-copy-policy` for the name, click **Add configurations**, and select **Post-copy**. Click **Yes** to the right of **Post-copy**.
  - 5.4. Set the project label, or selector, for the policy. Click **Enter key=value** under **Project Labels** and enter `kubernetes.io/metadata.name=advanced-migrate`.
  - 5.5. Set the virtual machine instance label, or selector, for the policy. Click **Enter key=value** under **VirtualMachineInstance Labels** and enter `app=webapp`.
  - 5.6. Leave the remaining settings with their default values and click **Create**.
- ▶ 6. Live migrate the `vm1` machine in the OpenShift web console. Afterwards, initiate and then cancel another live migration for the VM.
- 6.1. In the web console, go to **Virtualization > VirtualMachines**. The **Node** column shows the compute node for the `vm1` machine. Note the compute node that is hosting the VM.
  - 6.2. Click the vertical ellipsis icon at the right of the `vm1` machine and then click **Migrate**. Wait a few moments for the migration to complete.
  - 6.3. Confirm that the **Node** column shows a different compute node for the `vm1` machine than was identified in an earlier step.
  - 6.4. Initiate another live migration for the VM. Click the vertical ellipsis icon at the right of the `vm1` machine and then click **Migrate**.
  - 6.5. Cancel the live migration. Click the vertical ellipsis icon at the right of the `vm1` machine and then click **Cancel migration**.
- ▶ 7. Add a secondary network interface to the `vm1` machine. Live migrate the VM to apply the configuration without restarting the VM.
- 7.1. In the web console, go to **Virtualization > VirtualMachines** and select the `vm1` machine.

- 7.2. Go to the **Configuration** tab and select **Network**.
  - 7.3. Click **Add Network Interface** to create an interface on the VM.
  - 7.4. On the **Add Network Interface** page, select **advanced-migrate/br0** from the **Network** field. Leave the remaining settings with their default values. Click **Save** to save and attach the interface to the VM.
  - 7.5. Live migrate the VM to apply the pending changes. Click **Actions > Migrate** to live migrate the VM to a different node in the cluster. Wait a few moments for the migration to complete.
  - 7.6. Confirm that the secondary network interface is active on the **vm1** machine. Select the **Overview** tab and then locate the **Network** card. Confirm that the secondary network interface is configured with an IP address in the **192.168.51.0/24** network.
- 8. Initiate another live migration for the **vm1** machine by using a **VirtualMachineInstanceMigration** manifest.
- 8.1. In a text editor, modify the **migrate-vmi.yaml** file in the **~/D0316/labs/advanced-migrate/** directory.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstanceMigration
metadata:
  name: vm1-migration-job
spec:
  vmiName: vm1
```

- 8.2. In the terminal, change to the **advanced-migrate** project.

```
[student@workstation ~]$ oc project advanced-migrate
Now using project "advanced-migrate" on server "https://
api.ocp4.example.com:6443".
```

- 8.3. Use the **oc create** command to initiate the live migration with the **migrate-vmi.yaml** manifest.

```
[student@workstation ~]$ oc create -f \
~/D0316/labs/advanced-migrate/migrate-vmi.yaml
virtualmachineinstancemigration.kubevirt.io/vm1-migration-job created
```

- 8.4. Monitor the live migration with the **oc describe vmi** command. Confirm that **vm1** machine is migrated to another node, and note the node's name. You might have to wait a few minutes and rerun the command several times.

```
[student@workstation ~]$ oc describe vmi vm1
...output omitted...
Events:
  Type    Reason            Age     From           Message
  ----   -----           ----   ----           -----
  ...output omitted...
```

```
Normal SuccessfulUpdate 119s virtualmachine-controller Expanded
PodDisruptionBudget kubevirt-disruption-budget-8dg2q
Normal Migrating 84s (x2 over 7m23s) virt-handler
VirtualMachineInstance is migrating.
Normal PreparingTarget 84s (x4 over 7m23s) virt-handler
VirtualMachineInstance Migration Target Prepared.
Normal PreparingTarget 84s virt-handler Migration Target
is listening at 10.8.2.35, on ports: 39015,38699,40715
Normal Migrated 81s virt-handler The
VirtualMachineInstance migrated to node worker02.
Normal Deleted 81s virt-handler Sighaled Deletion
Normal SuccessfulUpdate 77s disruptionbudget-controller shrank
PodDisruptionBudget kubevirt-disruption-budget-8dg2q
```

- 9. Use the command line to initiate and then cancel a live migration for the vm1 machine.

- 9.1. In your terminal, delete the previous `vm1-migration-job` with the `oc delete vmim` command.

```
[student@workstation ~]$ oc delete vmim vm1-migration-job
virtualmachineinstancemigration.kubevirt.io "vm1-migration-job" deleted
```

- 9.2. Use the `oc create` command to initiate a live migration with the `migrate-vmi.yml` file.

```
[student@workstation ~]$ oc create -f \
~/DO316/labs/advanced-migrate/migrate-vmi.yml
virtualmachineinstancemigration.kubevirt.io/vm1-migration-job created
```

- 9.3. Cancel the live migration by deleting the `VirtualMachineInstanceMigration` object.

```
[student@workstation ~]$ oc delete vmim vm1-migration-job
virtualmachineinstancemigration.kubevirt.io "vm1-migration-job" deleted
```

- 9.4. Confirm that the VM is not migrated to another node.

```
[student@workstation ~]$ oc describe vmi vm1
...output omitted...
Events:
  Type    Reason        Age     From           Message
  ----  -----        ---   ----           -----
  ...output omitted...
  Normal  Migrating    10s    virt-handler  VirtualMachineInstance is
  migrating.
  Normal  PreparingTarget 10s    virt-handler  VirtualMachineInstance Migration
  Target Prepared.
  Normal  PreparingTarget 10s    virt-handler  Migration Target is listening at
  10.8.2.7, on ports: 43169,35525,40499
  Normal  Migrating    10s    virt-handler  VirtualMachineInstance is aborting
  migration.
  ...output omitted...
```

**Note**

If the migration is too fast to cancel via the given steps, then you can combine the `oc create`, `sleep`, and `oc delete` commands to cancel the migration.

```
[student@workstation ~]$ oc create -f \
~/DO316/labs/advanced-migrate/migrate-vmi.yaml ; \
sleep 5 ; oc delete vmim vm1-migration-job
```

- 10. Initiate and then cancel a live migration for the `vm1` machine by using the `virtctl` command.

10.1. Retrieve the node that is running the `vm1` machine.

```
[student@workstation ~]$ oc get vmi
NAME    AGE     PHASE      IP          NODENAME   READY
vm1     17m     Running    10.10.0.14  worker02  True
```

10.2. Use the `virtctl migrate` command to start a live migration for the `vm1` machine.

```
[student@workstation ~]$ virtctl migrate vm1
VM vm1 was scheduled to migrate
```

10.3. Cancel the live migration with the `virtctl migrate-cancel` command.

```
[student@workstation ~]$ virtctl migrate-cancel vm1
VM vm1 was scheduled to migrate-cancel
```

10.4. Confirm that the `vm1` machine did not migrate to another node.

```
[student@workstation ~]$ oc get vmi
NAME    AGE     PHASE      IP          NODENAME   READY
vm1     18m     Running    10.10.0.14  worker02  True
```

- 11. Close the web browser.

**Note**

Confirm that you are in the `/home/student` directory on the `workstation` machine before you use the `lab` command to complete the exercise.

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advanced-migrate
```

## ► Lab

# Advanced Virtual Machine Management

Control the placement of, monitor the health of, and live migrate a virtual machine.

## Outcomes

- Configure node placement rules for VMs.
- Configure health probes and watchdog devices on VMs.
- Live migrate a VM.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start advanced-review
```

## Instructions

- Open a command-line window and use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`  
Open a web browser and log in to the OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`  
Confirm that the `www1` and `mariadb-server` VMs run under the `advanced-review` project.
- The web server that runs inside the `www1` VM exposes the `/health` endpoint to monitor the application status. Add a readiness probe to the `www1` VM that uses HTTP GET requests to test the `/health` endpoint.  
The `lab` command prepares the `~/D0316/Labs/advanced-review/readiness.yaml` file for you. Refer to this file to create the readiness probe for the `www1` VM.  
After you configure the readiness probe, restart the VM to regenerate the `www1` VMI.  
If the readiness probe fails, then OpenShift removes the `www1` VMI as a web service endpoint. To test the readiness probe, log in to the VM console as the `developer` user with `developer` as the password. Use the `sudo systemctl stop httpd` command to become the `root` user and stop the `httpd` service. Confirm that OpenShift removes the `www1` VMI from the web service endpoints.  
Finally, start the `httpd` service and confirm that the web service endpoints include the `www1` VMI again.
- Add an `i6300esb` watchdog device named `testwatchdog` on the `www1` VM. Set the `action` field to the `poweroff` value, to power off the VM if the VM's operating system is unresponsive. The `lab` command prepares the `~/D0316/Labs/advanced-review/watchdog.yaml` file for your reference. Restart the VM to apply the changes.

Then, log in to the `www1` VM as the `developer` user with `developer` as the password. Use the `sudo -i` command to become the `root` user. Confirm that the watchdog device works by using the `echo > /dev/watchdog` command.

4. The MariaDB database that runs inside the `mariadb-server` VM listens on TCP port 3306. Add a liveness probe that tests the service by sending requests to the TCP socket.

The `lab` command prepares the `~/DO316/labs/advanced-review/liveness.yaml` file for you. Refer to this file to create the liveness probe on the `mariadb-server` VM.

After you configure the liveness probe, restart the VM to regenerate the `mariadb-server` VMI.

If the liveness probe fails, then OpenShift automatically restarts the `mariadb-server` VM. To test the liveness probe, log in to the VM console as the `developer` user with `developer` as the password. Use the `sudo systemctl stop mysql` command to stop the `mysql` service. Confirm that OpenShift restarts the VM.

5. Attempt a live migration of the `www1` VM. The live migration fails because of a misconfigured disk. Identify the disk and then delete it. (This misconfigured disk contains only temporary data that you can safely discard.)
6. Configure node placement rules for the `mariadb-server` and the `www1` VMs. The `lab` command labels the `worker01` and `worker02` nodes with the `workloads=production` label. Create a node selector for the `mariadb-server` VM by using a node selector for the `workloads=production` label. Create a pod affinity rule for the `www1` VM to ensure that the `www1` VM runs on the same node as the `mariadb-server` VM. Configure the Expression for the pod affinity rule to use the `app=mariadb` label.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade advanced-review
```

## Finish

As the `student` user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advanced-review
```

## ► Solution

# Advanced Virtual Machine Management

Control the placement of, monitor the health of, and live migrate a virtual machine.

### Outcomes

- Configure node placement rules for VMs.
- Configure health probes and watchdog devices on VMs.
- Live migrate a VM.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start advanced-review
```

### Instructions

- Open a command-line window and use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`  
Open a web browser and log in to the OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`  
Confirm that the `www1` and `mariadb-server` VMs run under the `advanced-review` project.
  - Open a command-line window and log in to the OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- Identify the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- Go to `Virtualization > VirtualMachines`. Select the `advanced-review` project from the `Project` list. Confirm that the `www1` and `mariadb-server` VMs are running.

2. The web server that runs inside the `www1` VM exposes the `/health` endpoint to monitor the application status. Add a readiness probe to the `www1` VM that uses HTTP GET requests to test the `/health` endpoint.

The `lab` command prepares the `~/D0316/labs/advanced-review/readiness.yaml` file for you. Refer to this file to create the readiness probe for the `www1` VM.

After you configure the readiness probe, restart the VM to regenerate the `www1` VMI.

If the readiness probe fails, then OpenShift removes the `www1` VMI as a web service endpoint. To test the readiness probe, log in to the VM console as the `developer` user with `developer` as the password. Use the `sudo systemctl stop httpd` command to become the `root` user and stop the `httpd` service. Confirm that OpenShift removes the `www1` VMI from the web service endpoints.

Finally, start the `httpd` service and confirm that the web service endpoints include the `www1` VMI again.

- 2.1. Select the `www1` VM and then click the **YAML** tab. Use the YAML editor to create a readiness probe for the VM.

You can refer to the `readinessProbe` section in the `~/D0316/labs/advanced-review/readiness.yaml` file.

Click **Save**.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
    architecture: amd64
    domain:
      ...output omitted...
    readinessProbe:
      httpGet:
        path: /health
        port: 80
      initialDelaySeconds: 10
      periodSeconds: 5
      timeoutSeconds: 2
      failureThreshold: 2
      successThreshold: 1
    evictionStrategy: LiveMigrate
    hostname: www1
...output omitted...
```

- 2.2. Click **Actions > Restart** to restart the `www1` VMI resource so Red Hat OpenShift Virtualization recognizes the probe. Wait a few minutes until the VMI is again in the **Running** state.
- 2.3. Click the **Console** tab and then log in to the `www1` VM console as the `developer` user with `developer` as the password.

```
...output omitted...
www1 login: developer
Password: developer
```

- 2.4. Use the `sudo` command to become the `root` user and stop the `httpd` service on the VM.

```
[developer@www1 ~]$ sudo systemctl stop httpd
```

- 2.5. From the terminal on the `workstation` machine, use the `oc` command to confirm that the `web` service endpoints do not include the IP address of the `www1` VMI.

```
[student@workstation ~]$ oc get endpoints -n advanced-review
NAME      ENDPOINTS      AGE
web           7m
```

- 2.6. From the `www1` console, use the `sudo` command to become the `root` user and start the `httpd` service.

```
[developer@www1 ~]$ sudo systemctl start httpd
```

- 2.7. Return to the terminal on the `workstation` machine and confirm that the `web` service endpoints include the IP address of the `www1` VMI.

```
[student@workstation ~]$ oc get endpoints -n advanced-review
NAME      ENDPOINTS      AGE
web     10.11.0.44:80     8m
```

The IP address might differ in your environment.

3. Add an `i6300esb` watchdog device named `testwatchdog` on the `www1` VM. Set the `action` field to the `poweroff` value, to power off the VM if the VM's operating system is unresponsive. The `lab` command prepares the `~/D0316/labs/advanced-review/watchdog.yaml` file for your reference. Restart the VM to apply the changes.

Then, log in to the `www1` VM as the `developer` user with `developer` as the password. Use the `sudo -i` command to become the `root` user. Confirm that the watchdog device works by using the `echo > /dev/watchdog` command.

- 3.1. From the OpenShift web console, go to `Virtualization > VirtualMachines`. Select the `www1` VM, and then go to the `YAML` tab.
- 3.2. Use the YAML editor to declare the watchdog device, and then click **Save**. You can copy and paste the `watchdog` section from the `~/D0316/labs/advanced-review/watchdog.yaml` file, which the `lab` command prepares.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
```

```

...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
    architecture: amd64
    domain:
      cpu:
        ...output omitted...
      devices:
        disks:
          ...output omitted...
        rng: {}
      watchdog:
        i6300esb:
          action: poweroff
          name: testwatchdog
    machine:
      type: pc-q35-rhel9.4.0
...output omitted...

```

- 3.3. Click **Actions > Restart** to restart the **www1** VMI resource so Red Hat OpenShift Virtualization recognizes the probe. Wait a few minutes until the VMI is again in the **Running** state.
- 3.4. Confirm that the watchdog device works on the **www1** VM. Click the **Console** tab and log in as the **developer** user with **developer** as the password.

```

...output omitted...
www1 login: developer
Password: developer

```

- 3.5. Use the **sudo** command to become the **root** user. Simulate an unresponsive operating system by using the **echo > /dev/watchdog** command. Confirm that OpenShift restarts the VM after 30 seconds.

```
[developer@www1 ~]$ sudo -i
[root@www1 ~]#
```

```
[root@www1 ~]# echo > /dev/watchdog
[ 45.680382] watchdog: watchdog0: watchdog did not stop!
```

4. The MariaDB database that runs inside the **mariadb-server** VM listens on TCP port 3306. Add a liveness probe that tests the service by sending requests to the TCP socket. The **lab** command prepares the **~/D0316/labs/advanced-review/liveness.yaml** file for you. Refer to this file to create the liveness probe on the **mariadb-server** VM. After you configure the liveness probe, restart the VM to regenerate the **mariadb-server** VMI. If the liveness probe fails, then OpenShift automatically restarts the **mariadb-server** VM. To test the liveness probe, log in to the VM console as the **developer** user with **developer** as the password. Use the **sudo systemctl stop mysql** command to stop the **mysql** service. Confirm that OpenShift restarts the VM.

- 4.1. From the OpenShift web console, go to **Virtualization > VirtualMachines**. Select the **mariadb-server** VM, and then go to the **YAML** tab.
- 4.2. Use the YAML editor to declare the probe, and then click **Save**. You can copy and paste the **livenessProbe** section from the `~/D0316/labs/advanced-review/liveness.yaml` file.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
  metadata:
    ...output omitted...
  spec:
    architecture: amd64
    domain:
      ...output omitted...
    livenessProbe:
      tcpSocket:
        port: 3306
        initialDelaySeconds: 20
        periodSeconds: 5
    evictionStrategy: LiveMigrate
    hostname: mariadb-server
...output omitted...
```

- 4.3. Click **Actions > Restart** to restart the **mariadb-server** VMI resource so Red Hat OpenShift Virtualization recognizes the probe. Wait a few minutes until the VMI is again in the **Running** state.
- 4.4. Go to the **Console** tab. Log in as the **developer** user with **developer** as the password, and then stop the **mysql** service.

```
mariadb-server login: developer
Password: developer
```

- 4.5. Become the **root** user and stop the **mysql** service.

```
[developer@mariadb-server ~]$ sudo systemctl stop mysql
```

Because the liveness probe fails, the VM restarts after a few seconds.

5. Attempt a live migration of the **www1** VM. The live migration fails because of a misconfigured disk. Identify the disk and then delete it. (This misconfigured disk contains only temporary data that you can safely discard.)
  - 5.1. Go to **Virtualization > VirtualMachines**. Click the vertical ellipsis icon **:** at the right of the **www1** VM and then confirm that **Migrate** is disabled.  
The **Status** and **Conditions** columns show the **Not migratable** and **LiveMigratable=False** values for the **www1** VM.

- 5.2. Select the `www1` VM and go to the **Diagnostics** tab. The error message for the **LiveMigratable** type indicates that the `tempdata` PVC is not using the `ReadWriteMany` access mode.
  - 5.3. Click **Actions > Stop**. Wait for the machine to stop.
  - 5.4. Go to the **Configuration** tab. In the **Storage** section, click the vertical ellipsis icon at the right of the `tempdata` disk and then click **Detach**. Click **Detach** to confirm the operation.
  - 5.5. Click **Actions > Start** and wait for the VM to restart.
  - 5.6. Live migrate the VM. Click **Actions > Migrate**. Wait for the migration to complete.
6. Configure node placement rules for the `mariadb-server` and the `www1` VMs. The `lab` command labels the `worker01` and `worker02` nodes with the `workloads=production` label. Create a node selector for the `mariadb-server` VM by using a node selector for the `workloads=production` label. Create a pod affinity rule for the `www1` VM to ensure that the `www1` VM runs on the same node as the `mariadb-server` VM. Configure the Expression for the pod affinity rule to use the `app=mariadb` label.
    - 6.1. Go to **Virtualization > VirtualMachines**.
    - 6.2. Select the `mariadb-server` VM and go to **Configuration > Scheduling**.
    - 6.3. Click the pencil icon under **Node Selector**.
    - 6.4. Click **Add Label to specify qualifying nodes** and enter the `workloads` key and the `production` value. Click **Save**.

**Note**

An admission web hook warning is displayed when you add node placement rules through the web console. It is safe to ignore this warning throughout this exercise.

- 6.5. Restart the `mariadb-server` VM. Click **Actions > Restart**.
- 6.6. Go to **Virtualization > VirtualMachines**. Wait until the `mariadb-server` VM status is **Running** and then confirm that the **Node** value is either the `worker01` node or the `worker02` node.
- 6.7. Select the `www1` VM and go to **Configuration > Scheduling**. Click the pencil icon under **Affinity rules**.
- 6.8. Click **Add affinity rule**. Set the **Type** to **Workload (pod) Affinity** and set the **Condition** to **Required during scheduling**. Leave the default value for the **Topology** key.
- 6.9. Click **Add expression**. Set the **Key** to `app` and the **Operator** to `In`. Enter `mariadb` for the **Values** and click **Create "mariadb"** to confirm the value. Click **Save affinity rule** and click **Apply rules**.
- 6.10. Restart the `www1` VM. Click **Actions > Restart**.
- 6.11. Go to **Virtualization > VirtualMachines**. Wait until the `www1` VM status is **Running**, and then confirm that the **Node** value for the `www1` VM is the same value as for the `mariadb-server` VM.

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade advanced-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advanced-review
```

# Summary

---

- With the migration toolkit for virtualization (MTV), you can migrate multiple VMs from a compatible hypervisor, such as VMware vSphere or Red Hat Virtualization, to OpenShift Virtualization.
- Red Hat OpenShift schedules VMs and pods to nodes according to node placement rules.
- You can use the Red Hat OpenShift advanced scheduling features to configure VMs and pods to run on a specific node or alongside a specific VM or pod.
- Use readiness and liveness probes to monitor the availability of a VM's application.
- The VMI spec . running and spec . runstrategy settings control the running state of the VM.
- The watchdog feature detects and restarts unresponsive operating systems, but it does not detect application failures.
- Live migration moves a running VMI to a different cluster node without disrupting access or the virtual workload.
- For live migration, the VM's PVCs must use the `ReadWriteMany` access mode.

## Chapter 9

# Configuring Kubernetes High Availability for Virtual Machines

### Goal

Implement high availability virtual machines that are resilient to failures, planned maintenance, and cluster upgrades by configuring Kubernetes resources.

### Sections

- Virtual Machine Load Balancing with Kubernetes Networking Resources (and Guided Exercise)
- Preparing for Node Maintenance (and Guided Exercise)
- Configuring Virtual Machines to Survive Node Failure (and Guided Exercise)
- OpenShift and OpenShift Virtualization Updates (and Quiz)

### Lab

- Configure Kubernetes High Availability for Virtual Machines

# Virtual Machine Load Balancing with Kubernetes Networking Resources

## Objectives

- Implement network load balancing for virtual machines that preserve session stickiness for high availability purposes, by using Kubernetes networking resources.

## High Availability with Hypervisors

*High Availability* (HA) is the ability to maintain continuous operations and to recover from unexpected failures in the shortest possible time.

HA in traditional hypervisors, such as vSphere HA from VMware, often focuses on the ability to restart or migrate VM workloads to other hosts in the cluster. This approach requires frequent manual intervention to provide application monitoring, clustering, and load balancing for workloads in a hypervisor.

## Fencing Hosts for VM Integrity

A virtual machine must never be running on two hosts at the same time, or its disk image is likely to become corrupt, which leads to data loss. It is important to isolate the node, which is known as *fencing*, before initiating recovery of the workload, which is known as *remediation*, and recovery of the node.

A traditional hypervisor uses an out-of-band management agent to fence a nonresponsive host. The agent forces a power-off, which ensures that the host and its virtual machines are down. Only then does the agent start the virtual machine on a new host.

## Load Balancing with Hypervisors

Load balancing helps avoid traffic bottlenecks by balancing the distribution of requests across the network. The traffic is distributed across available workloads that are ready for incoming requests.

By default, a stand-alone hypervisor does not include the software technology to load balance VMs. However, depending on the hypervisor provider, you can implement a load balancer to work with the hypervisor. By implementing hypervisor agents, you can also add node management actions, such as fencing and restarting. You must configure the load balancer, readiness and liveness probes, and watchdog functions for health monitoring, in addition to configuring the hypervisor.

## Manage Virtual Machines with Kubernetes

KubeVirt is the Kubernetes technology that helps you create and manage VMs that run in the same layer as your container workloads. KubeVirt delivers container-native virtualization by using a Kernel-based Virtual Machine (KVM) within a Kubernetes container. KubeVirt also provides features that are associated with traditional hypervisors, such as live migration and active resource balancing.

## Kubernetes Features for Highly Available VMs

Kubernetes uses declarative resource files to define an intended state, and controllers to check and remediate any variations from the intended state. This design enables maintaining continuous operations and recovering from unexpected failures in the shortest possible time.

A description follows of some Kubernetes and KubeVirt features that enable high availability:

### VM run strategies

When a VMI or `virt-launcher` pod is deleted for any reason, they are restarted according the `spec.running` or `spec.runStrategy` parameters that are defined in the VMI resource.

### Readiness and liveness probes

Developers can configure probes for monitoring their applications. A probe periodically monitors application health. You can use probes for applications that run in pods or VMs.

Readiness probes determine whether the application is ready to serve requests. If the readiness probe fails, then Kubernetes prevents client traffic from reaching the application by removing the VM's IP address from the service resource.

Liveness probes determine whether the application is in a healthy state. If the liveness probe detects an unhealthy state, then OpenShift Virtualization deletes the VMI resource and redeploys a new instance.

### Watchdog devices

You can configure a watchdog device inside a VM to verify the state of the OS and act according to the run strategy. Watchdogs monitor the OS only, and do not detect application failures.

### Live migration

A live migration ensures that the VM is not interrupted if the node is placed into maintenance or drained.

### Machine health checks

Machine health checks automatically remediate an unhealthy machine, which is the host for a node, if the machine exists in a particular machine pool. You can use machine health checks to monitor the health of a host by creating a resource that defines the condition to verify, the label for the set of hosts to monitor, and the remediation process to use.

### Fencing nodes

Fencing is a remediation method that reboots and deletes Machine custom resource definitions to solve problems with automatically provisioned nodes.

When the `MachineHealthCheck` controller detects that a node is in the `NotReady` state, it removes the associated `Machine` resource, and the node is deleted from the pool host.

### Load balancing

You can configure a load balancer service to enable external access to an OpenShift cluster, by allocating a unique IP from a configured pool. The load balancer has a single edge router IP, which can be a virtual IP (VIP), although it is still a single machine for initial load balancing.

However, load balancer services require the use of network features that are not available in all environments. For example, cloud providers typically provide their own load balancer services. If you run a Kubernetes cluster on a cloud provider, then controllers in Kubernetes use the cloud provider's APIs to configure the required cloud provider resources for a load balancing service.

On environments where managed load balancer services are not available, such as bare metal clusters or clusters that run on hypervisors, you must configure a load balancer component

according to the specifics of your network. For those environments, you can use the MetalLB operator, which provides a load balancing service.

### Sticky sessions

Sticky sessions enable stateful application traffic by ensuring that all traffic hits the same endpoint. However, if the endpoint pod terminates, then the statefulness can end.

## Configuring Route Statefulness

OpenShift can use cookies to configure session persistence. The ingress controller selects an endpoint to handle any user requests, and creates a cookie for the session. The cookie is passed back in the response to the request and the user sends the cookie back with the next request in the session. The cookie tells the ingress controller which endpoint is handling the session, ensuring that client requests use the cookie so that they are routed to the same pod.

You can set a cookie name to overwrite the default, auto-generated one for the route. Applying the `router.openshift.io/cookie_name="cookie_name"` annotation to the route creates a custom-named cookie.

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    router.openshift.io/cookie_name: "hello"
...output omitted...
```



### References

For more information about route configuration, refer to the *Route Configuration* section at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#route-configuration](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#route-configuration)

For more information about live migration, refer to the *Live Migration* chapter at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#live-migration](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#live-migration)

For more information about the MetalLB operator, refer to the *Load Balancing with MetalLB* chapter at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/networking/index#load-balancing-with-metallb](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/networking/index#load-balancing-with-metallb)

For more information about VM run strategies, refer to the *Run Strategies* section at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#run-strategies](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#run-strategies)

For more information about fencing and remediation, refer to the *Remediation, fencing, and maintenance* guide at

[https://docs.redhat.com/en/documentation/workload\\_availability\\_for\\_red\\_hat\\_openshift/24.3/html-single/remediation\\_fencing\\_and\\_maintenance/index](https://docs.redhat.com/en/documentation/workload_availability_for_red_hat_openshift/24.3/html-single/remediation_fencing_and_maintenance/index)

## ► Guided Exercise

# Load Balance a Virtual Machine with Kubernetes Networking Resources

Create a service and a route with a custom cookie for two web servers to host the `helloworld` application to confirm session stickiness and service load balancing.

## Outcomes

- Create a `helloworld` service for the web servers, and verify the service endpoints.
- Create a route with a custom cookie for the `helloworld` service.
- Observe the session stickiness that the custom cookie provides.
- Stop one of the web servers to observe the service resilience and the automatic load balancing from Kubernetes.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start ha-loadbalance
```

## Instructions

- 1. As the `admin` user, confirm that the `server1` and `server2` virtual machines (VMs) are running on the `ha-loadbalance` project.
- 1.1. From the command line, log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Change to the `ha-loadbalance` project.

```
[student@workstation ~]$ oc project ha-loadbalance
Now using project "ha-loadbalance" on server "https://api.ocp4.example.com:6443".
```

- 1.3. Verify that the `server1` and `server2` VMs are running.

```
[student@workstation ~]$ oc get vm
NAME      AGE      STATUS     READY
server1   4m25s   Running   True
server2   2m23s   Running   True
```

- ▶ 2. Create a `helloworld` service and verify that the service endpoints resolve to the virtual machine instances (VMIs) of the `server1` and `server2` VMs.
- 2.1. Create the `helloworld` service CR YAML file. You can find an incomplete example in the `~/D0316/labs/ha-loadbalance/svc.yml` file.  
The resource file must define the `helloworld` service in the `ha-loadbalance` namespace. The resource file must also define the service selector as the `app: helloworld` label. The `server1` and `server2` VMIs contain the `app: helloworld` label.

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
  namespace: ha-loadbalance
spec:
  type: ClusterIP
  selector:
    app: helloworld
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- 2.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/ha-loadbalance/svc.yml
service/helloworld created
```

- 2.3. Verify the IP addresses for the `server1` and `server2` VMIs. The IP addresses might differ on your system.

```
[student@workstation ~]$ oc get vmi
NAME      AGE      PHASE     IP           NODENAME   READY
server1   4m51s   Running   10.8.2.43   worker02   True
server2   3m7s    Running   10.11.0.36  worker01   True
```

- 2.4. Verify that the `helloworld` service endpoints resolve to the IP addresses of the `server1` and `server2` VMIs.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS          AGE
helloworld  10.11.0.36:80,10.8.2.43:80  52s
```

- ▶ 3. Create a `helloworld` route for the `helloworld` service.

- 3.1. Create the `helloworld` route CR YAML file. You can find an incomplete example in the `~/DO316/labs/ha-loadbalance/route.yml` file.

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: helloworld 1
  namespace: ha-loadbalance
  annotations:
    router.openshift.io/cookie_name: "hello" 2
spec:
  host: hello-ha-loadbalance.apps.ocp4.example.com 3
  port:
    targetPort: 80
  to:
    kind: Service
    name: helloworld 4
```

- 1** The name of the route.
- 2** The name of the custom cookie for the route. Applying the `router.openshift.io/cookie_name="cookie_name"` annotation to the route creates a custom-named cookie.
- 3** The `.spec.host` object defines a custom URL for the route. If the `.spec.host` object is not included, then the default naming convention, `route_name-namespace.apps.ocp4.example.com`, is used for the host.
- 4** The name of the service for the route.

- 3.2. Use the `oc` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/ha-loadbalance/route.yml
route.route.openshift.io/helloworld created.
```

- 3.3. Confirm that the route is configured with the `hello-ha-loadbalance.apps.ocp4.example.com` host.

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT
helloworld  hello-ha-loadbalance.apps.ocp4.example.com
            PATH  SERVICES   ...
                  helloworld ...
```

- **4.** Use the `curl` command to confirm that the `helloworld` route is accessible. Confirm that the route provides the `hello` custom-named cookie, and observe the session stickiness from the cookie.
- 4.1. Use the `curl` command to access the `helloworld` route. The output states the name of the web server that is servicing the request. You might need to run the command many times to verify that the load is balanced between the two web servers.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from server1.</p>
  </body>
</html>

[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from server2.</p>
  </body>
</html>
```

- 4.2. Use the `curl` command to save the `hello` cookie to the `/tmp/cookie-jar` file.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com \
  -c /tmp/cookie-jar
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from server2.</p>
  </body>
</html>
```

- 4.3. Confirm that the `hello` cookie exists in the `/tmp/cookie-jar` file.

```
[student@workstation ~]$ cat /tmp/cookie-jar
# Netscape HTTP Cookie File
# https://curl.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

#HttpOnly_hello-ha-loadbalance.apps.ocp4.example.com FALSE / FALSE 0 hello
a673...6722
```

- 4.4. The `hello` cookie provides session stickiness for connections to the `helloworld` route.

Use the `curl` command and the `hello` cookie to connect to the `helloworld` route again. Run the command many times to confirm that you are connected to the same web server that handled the request in the previous step.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com \
-b /tmp/cookie-jar
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <p>Hello, World, from server2.</p>
</body>
</html>

[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com \
-b /tmp/cookie-jar
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <p>Hello, World, from server2.</p>
</body>
</html>
```

- 4.5. Use the `curl` command to connect to the `helloworld` route without the `hello` cookie. Run the command many times and observe that session stickiness occurs only with the `hello` cookie.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <p>Hello, World, from server1.</p>
</body>
</html>

[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <p>Hello, World, from server2.</p>
</body>
</html>
```

- ▶ 5. Use the `virtctl` client to stop the `server1` VMI. Confirm that Kubernetes automatically updates the `helloworld` service endpoints not to include the `server1` VMI.

- 5.1. Retrieve the `helloworld` service endpoints.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS          AGE
helloworld  10.11.0.36:80,10.8.2.43:80  4m21s
```

- 5.2. Retrieve the IP addresses for the `server1` and `server2` VMIs.

```
[student@workstation ~]$ oc get vmi
NAME     AGE     PHASE    IP           NODENAME   READY
server1  9m31s  Running  10.8.2.43  worker02  True
server2  7m47s  Running  10.11.0.36  worker01  True
```

- 5.3. Use the `virtctl` client to shut down the `server1` VM.

```
[student@workstation ~]$ virtctl stop server1
VM server1 was scheduled to stop
```

- 5.4. Confirm that OpenShift updates the `helloworld` service endpoint to the IP address of the `server2` VMI, and that it does not contain the IP address of the `server1` VMI.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS          AGE
helloworld  10.11.0.36:80  5m20s
```

- 5.5. Use the `curl` command to verify that you can connect to the `helloworld` route. Confirm that the `server2` web server responds to the request.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from server2.</p>
  </body>
</html>
```

- ▶ 6. Use the `virtctl` client to start the `server1` VM. Confirm that the `helloworld` service endpoint contains the IP address of the `server1` VMI. Use the `curl` command to connect to the `helloworld` route, and confirm that the service is load balanced between the two web servers.

- 6.1. Start the `server1` VM with the `virtctl` client.

```
[student@workstation ~]$ virtctl start server1
VM server1 was scheduled to start
```

- 6.2. Verify that the `server1` VMI is running.

```
[student@workstation ~]$ oc get vmi
NAME      AGE     PHASE    IP          NODENAME   READY
server1   63s    Running  10.8.2.44  worker02  True
server2   9m55s   Running  10.11.0.36  worker01  True
```

- 6.3. Confirm that OpenShift updates the `helloworld` service endpoint to include the IP address of the `server1` VMI.

```
[student@workstation ~]$ oc get endpoints
NAME        ENDPOINTS           AGE
helloworld  10.11.0.36:80,10.8.2.44:80  7m15s
```

- 6.4. Use the `curl` command to send three subsequent requests to the `helloworld` route. Observe that both the `server1` and `server2` VMIs respond to the request. OpenShift load balances the requests between the two web servers. You might need to run the command many times to verify the load balance.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com/?[1-3]
[1/3]: hello-ha-loadbalance.apps.ocp4.example.com/?1 --> <stdout>
--_curl_--hello-ha-loadbalance.apps.ocp4.example.com/?1
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from server2.</p>
  </body>
</html>

[2/3]: hello-ha-loadbalance.apps.ocp4.example.com/?2 --> <stdout>
--_curl_--hello-ha-loadbalance.apps.ocp4.example.com/?2
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
    <p>Hello, World, from server1.</p>
  </body>
</html>

[3/3]: hello-ha-loadbalance.apps.ocp4.example.com/?3 --> <stdout>
--_curl_--hello-ha-loadbalance.apps.ocp4.example.com/?3
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
```

```
<p>Hello, World, from server2.</p>
</body>
</html>
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-loadbalance
```

# Preparing for Node Maintenance

---

## Objectives

- Put a node into maintenance mode and describe how node maintenance affects virtual machines and virtual machine placement.

## Node Maintenance

To prepare a node for maintenance, cluster administrators must perform two operations:

- Cordon* the node to prevent the cluster from deploying new workloads on the node.
- Drain* the node to move the current workload, including pods and VMs, to the remaining nodes.

You can use the `oc describe node nodename` command to see which workloads are running on a node. To drain a node, the cluster must have enough resources to relocate the workloads. You can display the node utilization with the `oc adm top nodes` command.

```
[user@host ~]$ oc adm top nodes
NAME      CPU(cores)   CPU%    MEMORY(bytes)   MEMORY%
master01  1420m        40%    14721Mi        64%
master02  1406m        40%    12871Mi        56%
master03  931m         26%    9128Mi         39%
worker01  134m         2%     4455Mi         29%
worker02  109m         1%     4309Mi         28%
```

## Moving a Node into Maintenance from the Command Line

To prevent the cluster from scheduling new workloads on the node, use the `oc adm cordon` command:

```
[user@host ~]$ oc adm cordon node2
node/node2 cordoned
```

The node is cordoned and marked as unschedulable with the `SchedulingDisabled` status. You can verify the node status with the `oc get node` command:

```
[user@host ~]$ oc get node node2
NAME      STATUS            ROLES   AGE     VERSION
node2    Ready, SchedulingDisabled   worker  4d20h  v1.27.10+28ed2d7
```

To move the current workload to the remaining nodes, use the `oc adm drain` command:

```
[user@host ~]$ oc adm drain node2 \
--delete-emptydir-data \ ①
--ignore-daemonsets \ ②
--force ③
...output omitted...
node/node2 drained
```

- ① The `--delete-emptydir-data` option prevents the drain operation from failing when some pods or VMs use the local node storage as an ephemeral volume. In that case, OpenShift restarts the pod or VM on the other nodes with a new empty volume.
- ② With the `--ignore-daemonsets` option, OpenShift skips moving the daemon set pods.
- ③ The `--force` option prevents the drain operation from failing when some pods that OpenShift does not manage are running on the node.

**Note**

The `oc adm drain` command automatically cordons the node.

After the maintenance is done, you can remove the node from maintenance mode by using the `oc adm uncordon` command:

```
[user@host ~]$ oc adm uncordon node2
node/node2 uncordoned
```

## The Node Maintenance Operator

The Node Maintenance operator enables cluster administrators to declaratively place nodes into maintenance by creating a `NodeMaintenance` custom resource. For tracking long maintenance operations, such as hardware replacement, you can attach a message to the `NodeMaintenance` resource as a reason for the maintenance.

As with other operators, you can install the Node Maintenance operator by using either the OpenShift web console or the command line.

Consult the references section for installing an operator by using the OperatorHub or the `oc` command.

## Moving a Node into Maintenance with the Node Maintenance Operator

As a cluster administrator, you set a node in maintenance mode by creating a `NodeMaintenance` resource.

From the OpenShift web console, go to **Operators > Installed Operators** and open the **Node Maintenance Operator** page. In the **Node Maintenance** card, click **Create instance**.

Complete the form with the node name and an optional maintenance message, and then click **Create**.

When a node is set to maintenance mode, the Node Maintenance operator cordons and drains the node.

On the **Node Maintenance Operator** page, click the **Node Maintenance** tab to get the status of the node maintenance. The node maintenance is in the Succeeded state when the draining process is complete.

The screenshot shows the 'Node Maintenance' tab selected in the navigation bar. Below it, a table lists a single node maintenance entry:

Name	Kind	Status	Labels
NM worker02-maintenance	NodeMaintenance	Phase: ✓ Succeeded	No labels

You can instead use the command line to create a `NodeMaintenance` resource. The following example shows the resource file that sets the `node2` node into maintenance mode:

```
apiVersion: nodemaintenance.medik8s.io/v1beta1
kind: NodeMaintenance
metadata:
  name: maintenance-node2
spec:
  nodeName: node2
  reason: "Node maintenance"
```

Use the `oc apply -f resourcefile.yaml` command to create the resource from the preceding file.

You can follow the draining process by using the `oc describe NodeMaintenance` command:

```
[user@host ~]$ oc describe NodeMaintenance maintenance-node2
...output omitted...
Status:
  Drain Progress: 100
  Phase:          Succeeded
  Totalpods:      47
...output omitted...
```

To verify that the node does not accept new workloads, confirm that the node has the `SchedulingDisabled` status:

```
[user@host ~]$ oc get nodes
NAME     STATUS            ROLES   AGE    VERSION
...output omitted...
node1    Ready             worker   13d   v1.27.10+28ed2d7
node2    Ready, SchedulingDisabled   worker   13d   v1.27.10+28ed2d7
```

To remove the node from maintenance mode and prepare it to accept new workloads, delete the `NodeMaintenance` resource.

From the OpenShift web console, on the **Node Maintenance** page, click the vertical ellipsis icon next to the node maintenance line, and then click **Delete Node Maintenance**.

From the command line, use the `oc delete NodeMaintenance` command:

```
[user@host ~]$ oc delete NodeMaintenance maintenance-node2  
nodemaintenances.nodemaintenance.medik8s.io "maintenance-node2" deleted
```



**Note**

OpenShift does not move back the pods and VMs that were running on the node before maintenance to the original node.

## Live Migrate Virtual Machines During Node Drain

When you create a VM, the eviction strategy is set to `LiveMigrate` by default.

During node drain, OpenShift Virtualization migrates these VMs live. For VMs with the eviction strategy of `None`, OpenShift Virtualization shuts down the VMs, and then restarts them on another node.



**Important**

Even though the eviction strategy is set, some VMs might not support live migration. For example, OpenShift Virtualization can live migrate only VMs with storage that supports the `ReadWriteMany` (RWX) access mode.

These VMs prevent the drain process from completing. A cluster administrator must manually shut down the VM or disable the live migration.

To verify whether a VM supports live migration, go to **Virtualization > VirtualMachines** and then select the VM. When a VM does not support live migration, OpenShift Virtualization adds the **Not migratable** badge to the VM.

The screenshot shows the OpenShift VM details page for a VM named 'rhel-9-blush-panther-28'. The 'Overview' tab is selected. In the top right, there is a 'Not migratable' status indicator, which is highlighted with a red box. On the right side, under the 'Diagnostics' tab, there is a warning message: 'VMCannotBeEvicted: The VM's eviction strategy is set to Live Migration but the VM is not migratable', also highlighted with a red box.

Figure 9.2: VM live migration warning

You can go to the **Diagnostics** tab and expand the status condition to get more information about this status.

The screenshot shows the 'Diagnostics' tab for the same VM. It displays a single error message: 'cannot migrate VMI: PVC rhel-9-blush-panther-28-disk-nfs is not shared, live migration requires that all PVCs must be shared (using ReadWriteMany access mode)', which is highlighted with a red box.

Figure 9.3: Verifying the VM live migration status

For VMs that are not migratable, set the eviction strategy to **None** to prevent the drain process from blocking.

Go to the **Configuration** tab, and click **Scheduling** to view the scheduling parameters. Click the **LiveMigrate** flag from the **Eviction Strategy** parameter, clear the **LiveMigrate** checkbox, and click **Save**.

You must restart the VM to apply the changes.

## Pod disruption budget

The *pod disruption budget* is a policy object where you can specify the percentage or number of pods that must remain running when a voluntary disruption occurs in the cluster. You can define the pod disruption budget resource with either the `minAvailable` or the `maxUnavailable` attribute.

Kubernetes clusters have two types of disruptions: *involuntary* and *voluntary*. Involuntary disruptions occur when a node fails and is disconnected from the cluster. Voluntary disruptions occur when nodes are cordoned or taken offline for maintenance. When a voluntary disruption

occurs, all the pods on the affected node are evicted at the same time, and the scheduler creates replacement pods in another node. If all the pods of a workload are running in the same node and that node is drained, then the minimum availability constraints for the application are not fulfilled.

A pod disruption budget resource is created and deleted with each VMI to prevent the `virt-launcher` pod from being deleted before the live migration process is completed.

You can list or describe the pod disruption budget resources to inspect their properties.

```
[user@host ~]$ oc get pdb
NAME          MIN AVAILABLE  MAX UNAVAILABLE ALLOWED DISRUPTIONS AGE
kubevirt-disrup...  1           N/A            0             34m  ①
```

- ① The minimum percentage or number of pods to be available

You can also describe the resource to view the label selector for the affected pods.

```
[user@host ~]$ oc describe pdb kubevirt-disruption-budget-wrztv
Name:          kubevirt-disruption-budget-wrztv
Namespace:     ha-maintenance
Max unavailable:  1  ①
Selector:      kubevirt.io/created-by=70d7a11a-c322-4227-82e7-9e2ab4fc2cbd  ②
Status:
Status:
  Allowed disruptions:  0
  Current:              1
  Desired:              1
  Total:                1
Events:        <none>
```

- ① The maximum percentage or number of pods that can be unavailable
- ② The label selector for the affected pods

## Node Drain with Pod Disruption Budget

With a pod disruption budget resource, the node drain is blocked until the replacement pods for the application are scheduled in another node and become ready. The pod eviction is blocked until the pod disruption budget availability constraints are met. The pod eviction operation retries after five seconds.

The following shows messages where the `virt-launcher` pod cannot be evicted because of a disruption budget. The node drain process continues to retry until the pod is evicted.

```
[user@host ~]$ oc adm drain worker01 --ignore-daemonsets --delete-emptydir-data
...output omitted...
error when evicting pods/"virt-launcher-vm1-nj7fv" -n "ha-maintenance" (will retry
after 5s): Cannot evict pod as it would violate the pod's disruption budget.
...output omitted...
pod/virt-launcher-vm1-nj7fv evicted
...output omitted...
```

**Important**

The drain process is blocked until all the pods are evicted from the node.

## Enabling Descheduler Evictions on a Virtual Machine

OpenShift does not move back to the original node the pods and VMs that were running on the node before maintenance. When you create VMs or manually live migrate machines, the scheduler gradually rebalances the workloads.

You can use the descheduler to evict pods, to be rescheduled to more appropriate nodes. If the pod is a virtual machine, then the pod eviction causes the virtual machine to be live migrated to another node.

**Important**

Descheduler eviction for virtual machines is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using Technology Preview features in production.

To enable the descheduler, you must install the Kube Descheduler Operator from OperatorHub.

Before starting the VM, add the `descheduler.alpha.kubernetes.io/evict` annotation to the `VirtualMachine` resource. Use the Technology Preview `DevPreviewLongLifecycle` profile to enable the descheduler on a virtual machine. Only this descheduler profile is currently available for OpenShift Virtualization.



## References

For more information about node maintenance, refer to the *Understanding How to Evacuate Pods on Nodes* section in the *Red Hat OpenShift Container Platform Nodes* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/nodes/index#nodes-nodes-working-evacuating\\_nodes-nodes-working](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/nodes/index#nodes-nodes-working-evacuating_nodes-nodes-working)

For more information about virtual machines and node maintenance, refer to the *Node Maintenance* chapter at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-node-maintenance](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-node-maintenance)

For more information about the Node Maintenance Operator, refer to the *Workload Availability for Red Hat OpenShift* guide at  
[https://docs.redhat.com/en/documentation/workload\\_availability\\_for\\_red\\_hat\\_openshift/24.3/html-single/remediation\\_fencing\\_and\\_maintenance/index#node-maintenance-operator](https://docs.redhat.com/en/documentation/workload_availability_for_red_hat_openshift/24.3/html-single/remediation_fencing_and_maintenance/index#node-maintenance-operator)

For more information about pod disruption budgets, refer to the documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/nodes/index#nodes-pods-pod-disruption-about\\_nodes-pods-configuring](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/nodes/index#nodes-pods-pod-disruption-about_nodes-pods-configuring)

For more information about the node descheduler, refer to the documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/nodes/index#nodes-descheduler-about](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/nodes/index#nodes-descheduler-about)

For more information about descheduler evictions on virtual machines, refer to the documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#virt-enabling-descheduler-evictions](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#virt-enabling-descheduler-evictions)

## ► Guided Exercise

# Perform Node Maintenance

Put compute nodes into maintenance mode, drain the workloads from those nodes, and observe node status.

## Outcomes

- Mark a node as unschedulable.
- Drain the workloads from nodes.
- Monitor node maintenance status.
- Resume a node from maintenance mode.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable and creates the required resources for this exercise.

```
[student@workstation ~]$ lab start ha-maintenance
```

## Instructions

- 1. As the `admin` user, confirm that the `vm1` machine is running in the `ha-maintenance` project.
- 1.1. Log in to the OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Change to the `ha-maintenance` project.

```
[student@workstation ~]$ oc project ha-maintenance
Now using project "ha-maintenance" on server "https://api.ocp4.example.com:6443".
```

- 1.3. Use the `oc get vmi` command to verify that the `vm1` machine is running. Note the name of the node that runs the VM. You use that node name in a later step.

```
[student@workstation ~]$ oc get vmi
NAME      AGE     PHASE      IP          NODENAME    READY
vm1      50m     Running   10.10.0.29  worker02   True
```

The node might be different in your environment.

- 2. To prepare for maintenance mode, verify that OpenShift Virtualization can live migrate the `vm1` machine. Also display the VM eviction strategy and any pod disruption budgets.

- 2.1. Use the `-o wide` option with the `oc get vmi` command to get additional information about the VMs. Verify that the `vm1` machine is live migrateable.

```
[student@workstation ~]$ oc get vmi -o wide
NAME  AGE  PHASE  IP      NODENAME  READY  LIVE-MIGRATABLE  PAUSED
vm1  53m  Running  10.10.0.29  worker02  True   True
```

- 2.2. Display the pod disruption budget that prevents the pod from being deleted until a new `virt-launcher` pod is running on another node and the migration is complete.

```
[student@workstation ~]$ oc get pdb
NAME          MIN AVAILABLE  MAX UNAVAILABLE  ALLOWED DISRUPTIONS ...
kubevirt-disruption-budget-q86r4  1           N/A            0           ...
```

You can also view the pod disruption budget from **Workloads > PodDisruptionBudgets** in the web console.

- 3. Cordon the node that runs the VM.

- 3.1. Use the `oc adm cordon` command to mark the node as unschedulable. In the following command, replace the `worker02` node with the node name from the previous step.

```
[student@workstation ~]$ oc adm cordon worker02
node/worker02 cordoned
```

- 3.2. Confirm that the node has the `SchedulingDisabled` status.

```
[student@workstation ~]$ oc get node worker02
NAME      STATUS           ROLES          AGE      VERSION
worker02  Ready, SchedulingDisabled  nodeAgent,worker  5d20h  v1.27.10+28ed2d7
```



### Note

If the OADP operator is not installed on your cluster, then the roles on your nodes might differ from the example output.

- 4. Evacuate the workload from the node.

- 4.1. Run the `oc adm drain` command to evacuate all workloads from the node. In the following command, replace the `worker02` node with the node name from the previous step. The command might take a few minutes to complete.

```
[student@workstation ~]$ oc adm drain worker02 \
--delete-emptydir-data --ignore-daemonsets --force
node/worker02 already cordoned
...output omitted...
evicting pod ha-maintenance/virt-launcher-vm1-n75xj
error when evicting pods/"virt-launcher-vm1-n75xj" -n "ha-maintenance" (will retry
after 5s): Cannot evict pod as it would violate the pod's disruption budget.
evicting pod ha-maintenance/virt-launcher-vm1-n75xj
pod/virt-launcher-vm1-n75xj evicted
node/worker02 drained
```

Ignore the error messages about the failed eviction of the `virt-launcher` pod. OpenShift Virtualization can delete the pod only after a new `virt-launcher` pod is running on another node and the migration is complete.

- 4.2. Confirm that the VM is now running on another node.

```
[student@workstation ~]$ oc get vmi
NAME        AGE     PHASE      IP          NODENAME   READY
vm1         63m    Running    10.10.0.29  master03   True
```

In this example, the `vm1` machine is now on the `master03` node.

► 5. Resume the node from maintenance mode.

- 5.1. Use the `oc adm uncordon` command to remove the node from maintenance mode. Replace the `worker02` node in the following command with the name of the node that you set into maintenance mode in the previous step.

```
[student@workstation ~]$ oc adm uncordon worker02
node/worker02 uncordoned
```

- 5.2. Confirm that the node is no longer in maintenance mode.

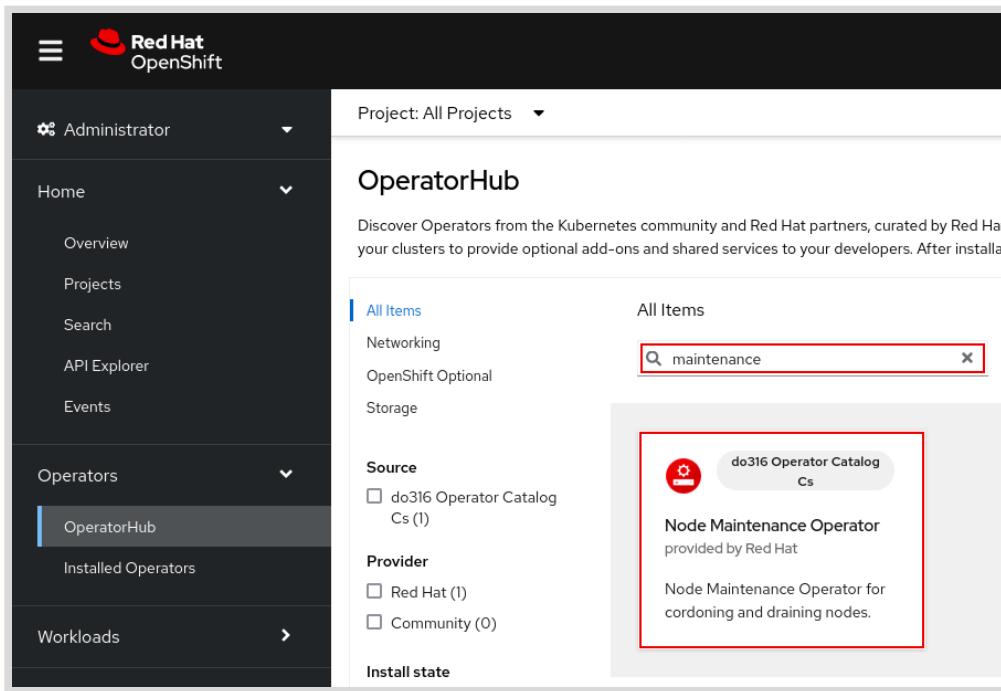
```
[student@workstation ~]$ oc get nodes
NAME      STATUS    ROLES                  AGE     VERSION
master01   Ready     control-plane,master,worker  15d    v1.27.10+28ed2d7
master02   Ready     control-plane,master,worker  15d    v1.27.10+28ed2d7
master03   Ready     control-plane,master,worker  15d    v1.27.10+28ed2d7
worker01   Ready     nodeAgent,worker           5d20h   v1.27.10+28ed2d7
worker02   Ready     nodeAgent,worker           5d20h   v1.27.10+28ed2d7
```

► 6. As an alternative to the `oc adm cordon` and `oc adm drain` commands from the preceding steps, use the Node Maintenance operator to set a node into maintenance mode and to drain its workload.

As the `admin` user, use the OpenShift web console to install the Node Maintenance operator.

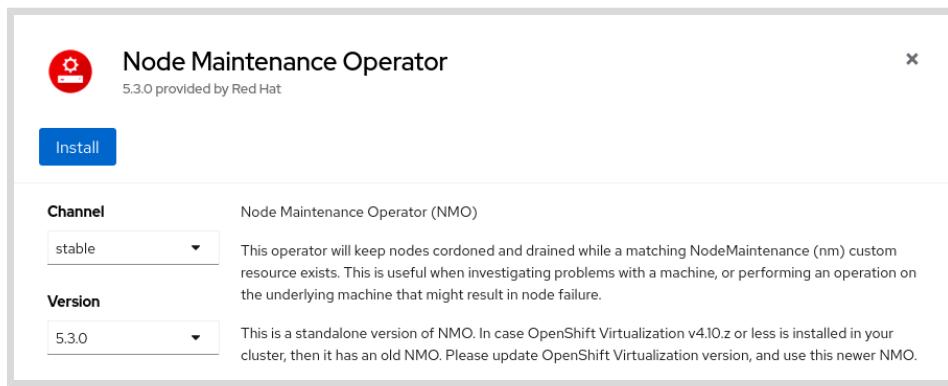
- 6.1. Open a web browser and go to <https://console.openshift-console.apps.ocp4.example.com>
- 6.2. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.

- 6.3. Click Operators > OperatorHub and select All Projects from the Projects drop-down menu. In the Filter by keyword field, type maintenance to locate the Node Maintenance operator, and then click Node Maintenance Operator.



The screenshot shows the Red Hat OpenShift web console. On the left, there's a sidebar with 'Administrator' at the top, followed by 'Home', 'Projects', 'Search', 'API Explorer', and 'Events'. Below that is the 'Operators' section with 'OperatorHub' selected, indicated by a blue bar. Under 'OperatorHub', there are links for 'Installed Operators' and 'Workloads'. The main content area is titled 'OperatorHub' and has a sub-header 'Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat to help you manage your clusters to provide optional add-ons and shared services to your developers. After installation, operators can be used to manage your cluster.' There are tabs for 'All Items' (selected) and 'Networking', 'OpenShift Optional', and 'Storage'. A search bar at the top right contains the text 'maintenance' with a red border around it. Below the search bar, a card for the 'Node Maintenance Operator' is shown, also highlighted with a red box. The card includes the 'do316 Operator Catalog' logo, the name 'Node Maintenance Operator provided by Red Hat', and a description 'Node Maintenance Operator for cordonning and draining nodes.'

- 6.4. The web console displays information about the Node Maintenance operator. Click Install to proceed to the Install Operator page.



The screenshot shows the 'Node Maintenance Operator' install page. At the top, there's a logo and the text 'Node Maintenance Operator 5.3.0 provided by Red Hat'. Below that is a large blue 'Install' button. The page is divided into sections: 'Channel' (set to 'stable') and 'Version' (set to '5.3.0'). The 'Channel' section contains a description: 'This operator will keep nodes cordoned and drained while a matching NodeMaintenance (nm) custom resource exists. This is useful when investigating problems with a machine, or performing an operation on the underlying machine that might result in node failure.' The 'Version' section contains a description: 'This is a standalone version of NMO. In case OpenShift Virtualization v4.10.0 or less is installed in your cluster, then it has an old NMO. Please update OpenShift Virtualization version, and use this newer NMO.'

- 6.5. Click **Install** to install the operator with the default options in the `openshift-workload-availability` namespace.
- 6.6. Wait until the installation is complete and the web console displays the `ready for use` message.

The screenshot shows the 'Installed operator' section of the OpenShift web console. It displays the 'Node Maintenance Operator' card, which includes a red circular icon with a gear and wrench, the name 'node-maintenance-operator.v5.3.0 provided by Red Hat', and a green checkmark indicating it is ready for use. Below the card, a message says 'Installed operator: ready for use'. There are two buttons: 'View Operator' and 'View installed Operators in Namespace openshift-workload-availability'.

- ▶ 7. Use the OpenShift web console to retrieve the name of the node that runs the `vm1` machine.
  - 7.1. Go to **Virtualization > VirtualMachines**. Select the `ha-maintenance` project from the **Projects** list, and note the name of the node that runs the VM. You use that node name in the next step.

Name	Status	Conditions	Node	Created	IP address
VM vm1	Running	DataVolumesReady=True	N master03	4 minutes ago	10.9.0.44

The node might be different in your environment.

- ▶ 8. Create a `NodeMaintenance` resource with the Node Maintenance operator to set the node from the previous step into maintenance mode and to drain its workload.
  - 8.1. Go to **Operators > Installed Operators** and open the **Node Maintenance Operator** page.
  - 8.2. In the Node Maintenance card, click **Create instance** to create a `NodeMaintenance` resource.

The screenshot shows the 'Installed Operators' page for the 'Node Maintenance Operator'. The left sidebar shows navigation options like Home, Operators, Workloads, Virtualization, Networking, and Storage. The 'Operators' section is expanded, and 'Installed Operators' is selected. The main content area shows the 'Node Maintenance Operator' card with version 5.3.0 provided by Red Hat. Below the card, there are tabs for Details, YAML, Subscription, Events, and Node Maintenance. The 'Details' tab is selected. Under 'Provided APIs', there is a section for 'Node Maintenance' with a description: 'NodeMaintenance is the Schema for the nodemaintainances API'. A red box highlights the 'Create instance' button.

- 8.3. Complete the **Create Node Maintenance** form by using the following information:

Field	Value
Name	live-migration-test
Node Name	<i>Use the node name from the previous step</i>
Reason	Testing live migration

- 8.4. Click **Create** to create the **NodeMaintenance** resource.
- 8.5. Observe the **live-migration-test** status and wait until the node maintenance state is **Succeeded**.

Name	Kind	Status	Labels
NM live-migration-test	NodeMaintenance	Phase: Succeeded	No labels

- 9. Verify that the node does not accept new workloads and that the VM is now running on another node.

- 9.1. Go to **Compute > Nodes** and confirm that the node has the **Scheduling disabled** status.

Name	Status	Roles	Pods	Mem...	CPU	Filesy...	Creat...
master-01	Ready	control-plane, master, worker	81	12.76 GiB / 23.47 GiB	2.269 cores / 4 cores	20.63 GiB / 69.72 GiB	Sep 30, 2024, 5:27 PM
master-02	Ready	control-plane, master, worker	59	8.31 GiB / 23.47 GiB	1.502 cores / 4 cores	24.79 GiB / 69.72 GiB	Sep 30, 2024, 5:27 PM
master-03	Ready	control-plane, master, worker	26	5.3 GiB / 23.47 GiB	0.582 cores / 4 cores	27.45 GiB / 69.72 GiB	Sep 30, 2024, 5:27 PM
worker-01	Ready	worker	65	7.87 GiB / 15.62 GiB	0.980 cores / 6 cores	25.74 GiB / 39.78 GiB	Oct 13, 2024, 12:48 PM
worker-02	Ready	worker	28	3.31 GiB / 15.62 GiB	0.349 cores / 6 cores	24.65 GiB / 39.78 GiB	Oct 13, 2024, 12:48 PM

- 9.2. Go to **Virtualization > VirtualMachines**, and confirm that the VM is running on a different node from the previous step.

► **10.** End the node maintenance.

- 10.1. Go to **Operators > Installed Operators** and open the **Node Maintenance Operator** page. Click the **Node Maintenance** tab.
- 10.2. On the **Node Maintenance** page, click the vertical ellipsis icon next to the **live-migration-test** node maintenance CR, and then click **Delete Node Maintenance**. Click **Delete** to confirm the operation and resume the node from maintenance mode.
- 10.3. Go to **Compute > Nodes** and confirm that the node no longer has the **Scheduling disabled** status.

## Finish

As the student user on the **workstation** machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-maintenance
```

# Configuring Virtual Machines to Survive Node Failure

## Objectives

- Configuring virtual machines to fail over to another node when their nodes fail, by using Kubernetes resources.

## VM Scheduling

OpenShift schedules VMs to healthy nodes according to the node placement rules and to the presence or absence of scheduler profiles. The OpenShift built-in scheduler identifies the most suitable node for the VMs when you create them. The default scheduler meets the needs of most OpenShift users.

Node placement strategies such as node selectors, affinity rules, and taints provide more control over which node is used for VM placement. These strategies are covered elsewhere in this course.

## Scheduler Profiles

The OpenShift scheduler profile controls how OpenShift schedules pods on nodes. The following scheduler profiles are available:

### LowNodeUtilization

Attempts to spread pods evenly across nodes for low resource usage per node.

### HighNodeUtilization

Attempts to place as many pods on as few nodes as possible. This approach minimizes the node count and creates high resource usage per node.

### NoScoring

A low latency profile that strives for the quickest scheduling cycle by disabling all scoring plugins.

## Eviction Strategies

Nodes fail for different reasons, such as insufficient disk space or package issues. If a node enters a failing state, whether the node is a control plane or a compute node, then eviction strategies determine the appropriate actions to take with orphaned resources, such as pods, persistent volumes (PVs), and virtual machines (VMs).

When a worker node enters the `NotReady` state after failing health checks, OpenShift reschedules the remaining pods on the node to healthy nodes with the `Ready` status.

Eviction strategies determine whether OpenShift moves the VMs on the failed node to another node or terminates them. You can configure the following eviction strategies for VMs:

### LiveMigrate

OpenShift migrates live, to ensure that the VM is not interrupted if you place the node into maintenance or if you drain it. This eviction strategy is the default. Non-migrateable VMs with this eviction strategy might prevent nodes from draining or might block a node upgrade because OpenShift does not evict the VM from the node, and you must shut down manually the VM.

#### LiveMigrateIfPossible

If users do not request a live migration on VMs, then OpenShift terminates non-migrateable VMs during evictions.

#### None

OpenShift does not migrate the VM, and it restarts or terminates the VM depending on the run strategy.

VMs with a live migration strategy must have a persistent volume claim (PVC) with a shared `ReadWriteMany` (RWX) access mode.

You can also configure an eviction strategy at the cluster level. Set the cluster eviction strategy by using the `.spec.evictionStrategy` parameter in the `HyperConverged` custom resource.



#### Important

Configuring a cluster eviction strategy is a Technology Preview feature only.

Technology Preview features are not supported with Red Hat production service level agreements and might not be functionally complete. Red Hat does not recommend using Technology Preview features in production.

## Node Failures

The `virt-handler` daemon set communicates with the `libvirtd` instance to define VMs.

VMs are scheduled only on nodes from which the control plane received a `virt-handler` heartbeat. It might take up to five minutes for the `virt-handler` daemon set and Kubernetes to detect the failure. After that time, the control plane nodes mark the node as unschedulable, and then migrate the node's workloads according to the resource's node placement rules and the scheduling profile.

If the `virt-handler` daemon set loses the connection to the cluster's API server, then the node cannot communicate its status. The node enters a failed state, and the remaining VMs cannot migrate to the healthy nodes.

## VM Rescheduling with Monitoring Tools

By default, Red Hat OpenShift includes a set of monitoring tools that periodically read the status of the cluster services and the created resources, to enable monitoring your VM instances. You can monitor the state of your VMs by using health checks and event logs.

Monitoring tools are updated during OpenShift cluster updates.

## Watchdog Monitor

You can configure a watchdog device to verify the state of the VM's guest OS, and act according to the run strategy. The VM's watchdog service monitors only for guest OS failures, and it does not detect application failures.

You can configure the following actions for a watchdog device to proceed when the guest OS is unresponsive:

#### poweroff

The VM powers off immediately. If the `spec.running` Boolean is set to `true`, or if the `spec.runStrategy` parameter is not set to `manual`, then the VM reboots.

**reset**

The VM reboots in place, and the guest OS cannot react.

**shutdown**

The VM gracefully powers off by stopping all services.

**Note**

The run strategies and installing the watchdog device are explained in more detail elsewhere in this course.

## Health Checks for VM Applications and Nodes

You can monitor the status of a VM's application and take the necessary actions with health checks. A health check periodically verifies the availability of a VM's application by using readiness and liveness probes. You can use either a readiness probe or a liveness probe, or both at the same time.

**Note**

The liveness, readiness, and watchdog device health checks are explained in more detail elsewhere in this course.

## Machine Health Checks

Machine health checks automatically remediate an unhealthy machine, which is the host for a node, if the machine exists in a particular machine pool. You can use machine health checks to monitor the health of a host by creating a resource that defines the condition to verify, the label for the set of hosts to monitor, and the remediation process to use. A bare metal machine controller that observes a `MachineHealthCheck` resource checks for the defined condition. If the condition is detected, then the controller starts the remediation process.

**Note**

Machine health checks are available only for clusters that are installed as bare-metal installer-provisioned infrastructure (IPI).

Depending on the condition, OpenShift removes or reboots the host.

## Standard Remediation for Failed Nodes

If a host fails the health check, then the bare metal machine controller drains the remaining VMs on the node and reschedules the workloads to healthy nodes that the machine set owns. This rescheduling action follows the node placement rules, the scheduling profile, and the eviction strategies that are configured for the VMs. After OpenShift drains all the VMs and the node registers itself again to the cluster, the bare metal machine controller restores the annotations and labels from the unhealthy node to the new node.

To limit the disruptive impact of the host deletion, the controller drains and deletes one node at a time. If multiple unhealthy hosts exceed the `maxUnhealthy` specified threshold for the targeted pool of hosts, then remediation stops and requires manual intervention.

Consider some limitations when using machine health checks:

- A machine health check can remediate only hosts that a machine set owns.
- If you remove the node for a host from the cluster, then the machine health check identifies that the host is unhealthy and remediates it immediately.
- A machine health check remediates a host immediately if the Machine resource enters the Failed status.
- If the node for a machine does not join the cluster after the configured nodeStartupTimeout parameter, then the machine health check remediates the machine.
- For cloud environments, a machine health check relies on cloud provider integration for the machine to forcibly reboot, reprovision, and rejoin the cluster.

## Power-based Remediation for Failed Bare Metal Nodes

Instead of reprovisioning the nodes, power-based remediation uses a power controller to power off an inoperable node. This type of remediation is also called *power fencing*. OpenShift uses the MachineHealthCheck controller to detect faulty bare metal nodes. Power-based remediation is fast and reboots faulty nodes instead of removing them from the cluster.

Power-based remediation provides the following capabilities:

- Enables the recovery of control plane nodes.
- Reduces the risk of data loss in hyperconverged environments.
- Reduces the associated downtime with recovering physical hosts.

If the power operations do not complete, then the bare metal machine controller triggers reprovisioning the unhealthy node, except for a control plane node or an externally provisioned node.



### Note

If the node has a *Baseboard Management Controller (BMC)*, then BMC credentials and network access to the BMC interface on the node are required.

## Self Node Remediation Operator

Available in non-IPMI bare metal clusters, the Self Node Remediation Operator runs on the cluster nodes and reboots unhealthy nodes. This remediation strategy minimizes downtime for stateful applications and ReadWriteOnce (RWO) volumes, and restores compute capacity if transient failures occur.

The Self Node Remediation Operator creates a `SelfNodeRemediationConfig` CR in the operator's namespace. You can edit the `SelfNodeRemediationConfig` CR to configure parameters such as the time that the Operator waits before recovering affected workloads that are running on an unhealthy node, or the file path of the watchdog device in the nodes, or the frequency to verify connectivity with each API server. If the Self Node Remediation Operator detects a change in the `SelfNodeRemediationConfig` CR, then it re-creates the Self Node Remediation daemon set.

The operator uses the `MachineHealthCheck` controller to detect the health of a node in the cluster. If an unhealthy node is detected according to the `SelfNodeRemediationConfig` CR, then the `MachineHealthCheck` controller creates the `SelfNodeRemediation`

CR to trigger the Self Node Remediation Operator. Then, the operator uses the `SelfNodeRemediationTemplate` custom resource definition (CRD) to define the remediation strategy for the unhealthy nodes.

You can install the Self Node Remediation Operator from the command line by creating a subscription in the `openshift-operators` namespace, or via the web console through the `OperatorHub` menu.

**Note**

For more information about the Self Node Remediation Operator, refer to the *Using Self Node Remediation* chapter in the *Workload Availability for Red Hat OpenShift* guide at [https://docs.redhat.com/en/documentation/workload\\_availability\\_for\\_red\\_hat\\_openshift/24.3/html-single/remediation\\_fencing\\_and\\_maintenance/index#self-node-remediation-operator-remediate-nodes](https://docs.redhat.com/en/documentation/workload_availability_for_red_hat_openshift/24.3/html-single/remediation_fencing_and_maintenance/index#self-node-remediation-operator-remediate-nodes)

## Manually Recover VMs from a Failed Node

If a node fails, and you do not deploy machine health checks and the Self Node Remediation Operator on the cluster, then OpenShift does not automatically relocate VMs that are configured with the `RunStrategy: Always` parameter to healthy nodes. Instead, you must manually remove the node by deleting the node object.

However, if you installed the cluster with IPI and correctly configured machine health checks, then OpenShift automatically recycles the failed nodes, and schedules the VMs with the `RunStrategy: Always` or the `RunStrategy: RerunOnFailure` parameter on healthy nodes.

## Delete a Node

Although deleting a node deletes the node object in Kubernetes, Kubernetes does not delete the pods on the node. If a replica set does not back the pods on the node, then the pods become inaccessible. OpenShift reschedules only pods that are backed by replica sets to other available nodes.

You can delete a node from your cluster by completing the following steps:

1. Mark the node as not schedulable:

```
[user@host ~]$ oc adm cordon worker01  
node/worker01 cordoned
```

2. Drain all pods and VMs on the node:

```
[user@host ~]$ oc adm drain worker01 \  
--ignore-daemonsets=true --delete-emptydir-data --force  
node "worker01" drained
```

**Note**

This step might fail if the node is offline or unresponsive. To avoid data corruption, power off the physical hardware before you proceed.

- Delete the node from the cluster:

```
[user@host ~]$ oc delete node worker01
node "worker01" deleted
```



### Note

The node can still rejoin the cluster after a reboot or if the kubelet service is restarted. To permanently delete the node and all its data, you must decommission the node. For more information about decommissioning a node, refer to the *How to Destroy All the Data from Server for Decommission* solution at <https://access.redhat.com/solutions/84663>

- Verify the status of the VMs that are rescheduled on healthy nodes:

```
[user@host ~]$ oc get vmis --all-namespaces
NAMESPACE      NAME          AGE     PHASE    IP           NODENAME   READY
vm-qa          fedora-test   4h6m   Running   10.9.0.47  worker02   True
```

## Disaster Recovery Solutions

To ensure that your environment can recover after a site outage, a disaster recovery solution includes multiple clusters with replication mechanisms. OpenShift Virtualization supports both a *Metro-DR* solution that uses synchronous replication and a *Regional-DR* solution that uses asynchronous replication.

Red Hat Advanced Cluster Management (RHACM) for Kubernetes offers end-to-end visibility and control for managing your Kubernetes clusters and application lifecycle across multiple data centers and public clouds. RHACM uses OpenShift GitOps to make configuring clusters for application deployment more consistent and reproducible. Data is replicated with a *mover* application such as `rsync` for file system volumes or `VolSync` for either file system volumes or block volumes.



### Note

For more information about OpenShift Virtualization DR solutions, review the Knowledgebase article at <https://access.redhat.com/articles/7041594>



## References

For more information about OpenShift node failures, refer to the *Remediation, Fencing, and Maintenance* guide at

[https://docs.redhat.com/en/documentation/workload\\_availability\\_for\\_red\\_hat\\_openshift/24.3/html-single/remediation\\_fencing\\_and\\_maintenance/index](https://docs.redhat.com/en/documentation/workload_availability_for_red_hat_openshift/24.3/html-single/remediation_fencing_and_maintenance/index)

For more information about eviction strategies, refer to the *Nodes* chapter in the Red Hat OpenShift Container Platform 4.16 *Virtualization* documentation at

[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#nodes](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#nodes)

For more information about Openshift Virtualization disaster recovery, refer to the Red Hat Knowledgebase article at

<https://access.redhat.com/articles/7041594>

For more information about disaster and site recovery, refer to the Developer Blog at

<https://developers.redhat.com/articles/2024/05/24/virtualization-vsphere-disaster-recovery#>

## ► Guided Exercise

# Configure Virtual Machines to Survive a Node Failure

Configure virtual machines to automatically fail over to another cluster node if the node that it runs on becomes unresponsive.

## Outcomes

- Monitor the connectivity to a web application from two VMs on a failed node.
- Identify and drain the failed node that hosts the web application VMs.
- Manually recover a VM from the failed node.
- Adjust the eviction strategy of a VM.
- Delete the node from the cluster.
- Restart the node to rejoin the cluster.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start ha-node
```

## Instructions

The `lab` command creates the `ha-node` namespace and starts two virtual machines, `web1` and `web2`, which host a web application.

This command also creates a service that load balances client requests between the two VMs, and a route resource for clients to access the web application.

- 1. As the `admin` user, confirm that the two VMs are running in the `ha-node` project.
  - 1.1. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Select `httpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
  - 1.2. Go to `Virtualization > VirtualMachines` and then select the `ha-node` project. Confirm that the `web1` and `web2` VMs are running.
- 2. Verify the eviction and run strategies of the `web1` and `web2` VMs.
  - 2.1. Select the `web1` VM, and then go to the `Configuration > Scheduling` menu. Confirm that the eviction strategy is set to `LiveMigrate`.

The screenshot shows the 'VirtualMachines' details page for a VM named 'web1'. The 'Configuration' tab is selected. Under the 'Scheduling and resource requirements' section, the 'Scheduling' tab is highlighted with a red box. Within the 'Scheduling' tab, the 'Eviction strategy' field is set to 'LiveMigrate' and is also highlighted with a red box.

- 2.2. Go to the **YAML** tab to open the VM's manifest in the YAML editor. Within the YAML manifest, confirm that the `.spec.runStrategy` object is set to the `RerunOnFailure` run strategy.

```
...output omitted...
spec:
...output omitted...
runStrategy: RerunOnFailure
template:
  metadata:
...output omitted...
```

- 2.3. Go to **Virtualization > VirtualMachines** and select the **web2** VM. Go to the **Configuration > Scheduling** menu. Confirm that the eviction strategy is set to **None**.
- 2.4. Go to the **YAML** tab to open the VM's manifest in the YAML editor. Within the YAML manifest, confirm that the `.spec.runStrategy` object is set to the `RerunOnFailure` run strategy.

```
...output omitted...
spec:
...output omitted...
runStrategy: RerunOnFailure
template:
  metadata:
...output omitted...
```

- 3. Confirm that the `www-web` service endpoints resolve to the IP addresses of the `web1` and `web2` VMs. Identify and monitor the node that runs the `web1` and `web2` VMs.
- 3.1. From a command-line window, log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 3.2. Change to the ha-node project.

```
[student@workstation ~]$ oc project ha-node
Now using project "ha-node" on server "https://api.ocp4.example.com:6443".
```

- 3.3. Use the `oc` command to list the VMI resources in the ha-node project. Note the IP addresses of the `www1` and `www2` VM instances, and the node that hosts the VMIs that are listed in the previous step. The IP addresses might differ in your environment.

```
[student@workstation ~]$ oc get vmi
NAME    AGE     PHASE      IP          NODENAME   READY
web1    18m    Running    10.11.0.24  worker01  True
web2    17m    Running    10.11.0.29  worker01  True
```

- 3.4. Confirm that the `www-web` service has active endpoints that resolve to the IP addresses of the `web1` and `web2` VMIs.

```
[student@workstation ~]$ oc get endpoints
NAME        ENDPOINTS           AGE
www-web    10.11.0.24:80,10.11.0.29:80  18m
```

- 3.5. Open a command-line window and execute the `loop.sh` file in the `~/D0316/labs/ha-node/` directory. The `loop.sh` file executes the `curl` command against the `web-ha-node.apps.ocp4.example.com` route. Leave the command running.

```
[student@workstation ~]$ ~/D0316/labs/ha-node/loop.sh
Welcome to web1
Welcome to web2
Welcome to web1
...output omitted...
```

- 3.6. Open a command-line window and use the `watch` command to monitor the VMI's availability during this exercise. Leave the command running.

```
[student@workstation ~]$ watch oc get vmi
Every 2.0s: oc get vmi                         workstation.lab.example.com:...
NAME    AGE     PHASE      IP          NODENAME   READY
web1    24m    Running    10.11.0.24  worker01  True
web2    22m    Running    10.11.0.29  worker01  True
```

- 4. Developers report that resources on the node that runs the VMIs are experiencing performance and connectivity issues. These issues do not affect resources on other cluster nodes.

As the cluster administrator, you suspect that the node is failing due to an incorrect configuration.

Prevent new workloads from running on the node that runs the VMIs, and then drain the node of its current workloads. Manually recover and adjust the eviction strategy of a VM on the failed node.

Then, power off the node and delete it from the cluster.

- 4.1. On the workstation machine, open a command-line window and mark the worker01 node as not schedulable, with the `oc adm cordon` command.

```
[student@workstation ~]$ oc adm cordon worker01  
node/worker01 cordoned
```

- 4.2. Confirm that the node has the Ready, SchedulingDisabled status.

```
[student@workstation ~]$ oc get node worker01  
NAME      STATUS            ROLES          AGE       ...  
worker01   Ready, SchedulingDisabled   nodeAgent,worker   4d16h   ...
```

**Note**

If the OADP operator is not installed on your cluster, then the roles on your nodes might differ from the example output.

- 4.3. Drain the node of its workloads.

```
[student@workstation ~]$ oc adm drain worker01 \  
--ignore-daemonsets=true --delete-emptydir-data --force  
node/worker01 already cordoned  
...output omitted...  
node/worker01 drained
```

- 4.4. Monitor the command-line window that executes the `loop.sh` command. Observe the high availability of the web application.

```
...output omitted...  
Welcome to web1  
...output omitted...
```

- 4.5. Monitor the command-line window that executes the `watch` command. Notice that OpenShift sets to the `Succeeded` value the phase for the `web2` VMI, and that the VMI has a `False` ready status, because OpenShift shuts down the `web2` VMI. Kubernetes cannot automatically relocate the `web2` VM to a healthy node in the cluster, because the `web2` VMI uses the `None` eviction strategy and you did not configure machine health checks.

```
Every 2.0s: oc get vmi                         workstation.lab.example.com:...
NAME    AGE     PHASE      IP          NODENAME   READY
web1   36m    Running    10.8.2.55  worker02   True
web2   34m    Succeeded  10.11.0.29 worker01   False
```

- ▶ 5. Configure the web2 VM with the **LiveMigrate** eviction strategy, and then manually recover the VM from the failed node.
- 5.1. From the web console, go to **Virtualization > VirtualMachines**. Select the web2 VMI and click the **Configuration** menu.
  - 5.2. Go to the **Scheduling** section, and click **None** in the **Eviction strategy** subsection. Click the **LiveMigrate** flag and click **Save**.
  - 5.3. Click **Actions > Start** to power on and reschedule the VM on another node in the cluster.
  - 5.4. Monitor the command-line window where the `loop.sh` command is running, and observe that the web1 VMI serves all the requests until the web2 VMI reaches the **Running** status.

```
...output omitted...
Welcome to web1
Welcome to web1
Welcome to web1
Welcome to web1
Welcome to web2
Welcome to web2
Welcome to web1
...output omitted...
```

- 5.5. Go to the command-line window where the `watch` command is running. The **worker02** node hosts the web1 VMI, and the **master02** node hosts the web2 VMI. Nodes might differ in your environment.

```
Every 2.0s: oc get vmi                         workstation.lab.example.com:...
NAME    AGE     PHASE      IP          NODENAME   READY
web1   43m    Running    10.8.2.55  worker02  True
web2   3m31s   Running   10.9.0.41  master02  True
```

- ▶ 6. Delete the node from the cluster.
- 6.1. To prevent potential data corruption, power off the drained node. From the **Lab Environment** page for this course, locate the drained host machine, click **ACTION**, and then click **Power Off**. Wait for the machine to display the **Stopped** status before proceeding.

master01	Active	ACTION -	OPEN CONSOLE
master02	Active	ACTION -	OPEN CONSOLE
master03	Active	ACTION -	OPEN CONSOLE
worker01	Active	ACTION -	OPEN CONSOLE
worker02	Active	ACTION -	OPEN CONSOLE

- 6.2. Return to the command-line window on the **workstation** machine where you initiated the node drain. Delete the drained node from the cluster.

```
[student@workstation ~]$ oc delete node worker01
node "worker01" deleted
```

- 6.3. List the nodes to confirm that the deleted node is no longer available.

[student@workstation ~]\$ oc get nodes					
NAME	STATUS	ROLES	AGE	VERSION	
master01	Ready	control-plane, master, worker	14d	v1.27.10+28ed2d7	
master02	Ready	control-plane, master, worker	14d	v1.27.10+28ed2d7	
master03	Ready	control-plane, master, worker	14d	v1.27.10+28ed2d7	
worker02	Ready	nodeAgent, worker	4d16h	v1.27.10+28ed2d7	

#### ► 7. Instruct the deleted node to rejoin the cluster.

- 7.1. From the **Lab Environment** page for this course, locate the deleted host machine, click **ACTION**, and then click **Start**. Wait for the machine to display the **Active** status before proceeding.
- 7.2. From the command-line window on the **workstation** machine, list the nodes to confirm that the deleted node rejoined the cluster. It might take a few minutes for the node to display the **Ready** status.

[student@workstation ~]\$ oc get nodes					
NAME	STATUS	ROLES	AGE	VERSION	
master01	Ready	control-plane, master, worker	14d	v1.27.10+28ed2d7	
master02	Ready	control-plane, master, worker	14d	v1.27.10+28ed2d7	
master03	Ready	control-plane, master, worker	14d	v1.27.10+28ed2d7	
worker01	Ready	nodeAgent, worker	3m7s	v1.27.10+28ed2d7	
worker02	Ready	nodeAgent, worker	4d16h	v1.27.10+28ed2d7	

- 7.3. In the command-line window that is executing the `loop.sh` command, press **Ctrl+C** to stop the command. Close the command-line window.
- 7.4. In the command-line window that is executing the `watch` command, press **Ctrl+C** to stop the command. Close the command-line window.

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-node
```

# OpenShift and OpenShift Virtualization Updates

## Objectives

- Describe how the OpenShift Virtualization update process affects virtual machines.

## Updating Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform uses a software distribution system that provides a suitable upgrade path to update the cluster components and the underlying operating system.

This distribution system enables clusters to upgrade directly from the internet. This system provides all the necessary resources to update a cluster to a particular version. This system also enables a cluster to use new features as they become available, including the latest bug fixes and security patches.



### Note

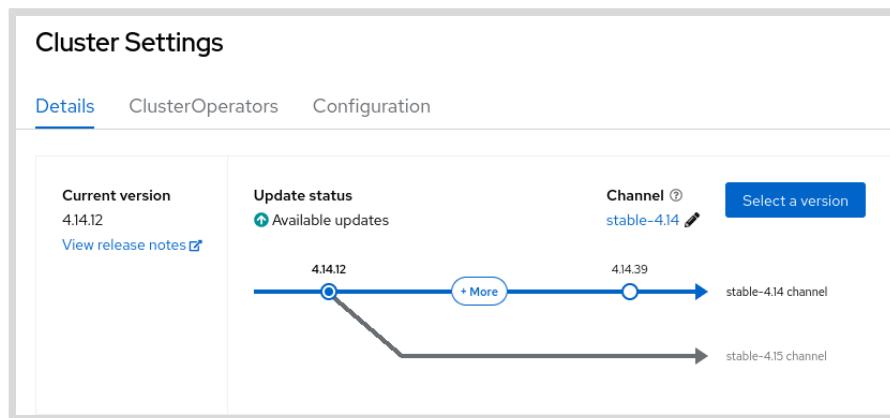
An on-premise version of the distribution system is available when no internet access is available.

The Cluster Version Operator (CVO), which runs in your cluster, regularly contacts the OpenShift Update Service (OSUS) on the internet. The CVO provides the current component versions, and the OSUS returns an update path.

The OSUS computes the update path from your selected channel for your cluster. A channel corresponds to a minor Red Hat OpenShift version, such as 4.14.

Go to **Administration > Cluster Settings** to manage your cluster updates.

The following screen capture shows the path to update a 4.14.12 cluster to 4.14.39, with an alternative path to the next minor version, 4.15.



**Figure 9.13: Updating from 4.14.12**

After the cluster administrator selects the `stable-4.15` channel, the CVO automatically contacts the OSUS to get the update path. The path shows that to migrate from 4.14.12, which is the current cluster version, to 4.15, you must first update your cluster to version 4.14.38.

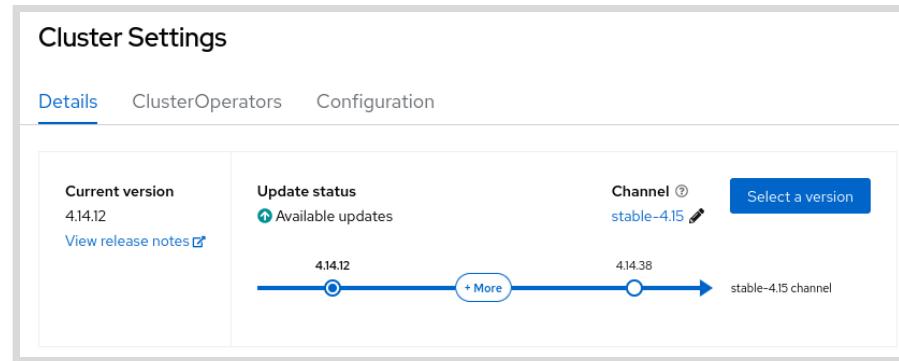


Figure 9.14: Updating from 4.14.12 to 4.15

From the command line, you can monitor the progress of updating the cluster operators with the following command:

```
[student@host ~]$ oc get co
NAME          VERSION  AVAILABLE  PROGRESSING  DEGRADED
SINCE      MESSAGE
...output omitted...
marketplace    4.16.14  True        False        False
32d
monitoring     4.16.2   True        True         False
32d          Rolling out the stack.
network        4.16.2   True        False        False
32d
node-tuning     4.16.14  True        True         False       6s
Working towards "4.16.14"
...output omitted...
```

The Machine Config Operator (MCO) updates the operating system on the cluster nodes during the update process. During the update, the MCO cordons off and drains the nodes to keep the pods and VMs running.

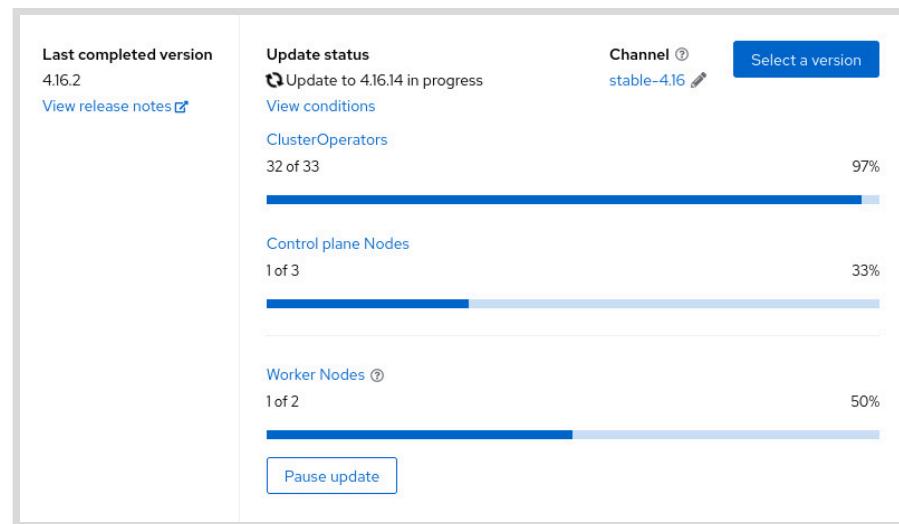


Figure 9.15: Update in progress

## Machine Configuration Pools

OpenShift uses Red Hat Enterprise Linux CoreOS (RHCOS) as the underlying operating system in the hosts. The entire operating system is updated as a single image, instead of by package. The machine configuration operator manages the RHCOS operating system upgrades and configuration changes.

The MCO defines two custom resources (CRs), which are part of the `machineconfiguration.openshift.io/v1` API group.

### MachineConfig

A Machine Config (MC) CR declares instance customizations by using the Ignition configuration format. An MC object can manage `systemd` services such as NTP, kernel arguments, SSH keys, and files in the `/var` or `/etc` directories.

### MachineConfigPool

A Machine Config Pool (MCP) CR uses labels to match one or more MCs to one or more nodes by using the `machineConfigSelector` and `nodeSelector` parameters, respectively. This resource creates a pool of nodes with the same configuration. The MCO uses the MCP to track status when the MCO applies MCs to the nodes.

In addition to tracking the status of nodes as they are updated, the `mcp` resource has a `maxUnavailable` parameter that defines the maximum number of nodes that can be updated at once.

From the command line, you can use the following command to view the update progress:

```
[student@host ~]$ oc get mcp
NAME      CONFIG                                     UPDATED   UPDATING
DEGRADED  MACHINECOUNT    READY MACHINECOUNT  UPDATED MACHINECOUNT
DEGRADED MACHINECOUNT  AGE
master    rendered-master-d64ffda46f9d87ebc6e2580ae9b3317d  True     False
False      3           3           3
            31d
worker    rendered-worker-a240564e337412e8a445e93a0cce8646  False    True
False      2           0           0
            31d
```

## Updating Red Hat OpenShift Virtualization

For Red Hat to support your installation, the version of the OpenShift Virtualization operator must match the version of Red Hat OpenShift. By default, Red Hat OpenShift automatically updates the operator.

Go to Operators > Installed Operators and click **OpenShift Virtualization** to access the operator details. From the **Subscription** tab, review the **Update approval** section to confirm that Red Hat OpenShift automatically applies the updates.

The screenshot shows the 'Subscription' tab selected in the navigation bar. Below it, the 'Subscription details' section is displayed. It includes three main items: 'Update channel' set to 'stable' with a pencil icon, 'Update approval' set to 'Automatic' with a pencil icon, and 'Upgrade status' showing 'Up to date' with 1 installed and 0 installing. The 'Update approval' field is highlighted with a red box.

Figure 9.16: Automatic operator update

Red Hat recommends using the automatic update process.

## Virtual Machine Pod Updates

The updates might include `virt-launcher` pod components, such as the `libvirt` or `qemu` services. To update these pods, OpenShift Virtualization migrates the VMs live. During the live migration, OpenShift Virtualization starts the new updated `virt-launcher` pods.

If a VM does not support live migration, then OpenShift Virtualization does not update the pod until you restart the VM.

## Troubleshooting a Cluster Update

Go to Administration > Cluster Settings to view the cluster update history or use the following command:

```
[student@host ~]$ oc describe clusterversions/version
...output omitted...
Spec:
  Channel:      stable-4.16
  Cluster ID:  171b7a32-16f1-4139-b6d0-e18c7d8ff057
Status:
  Available Updates:
    Channels:
      candidate-4.16
      candidate-4.17
      eus-4.16
      fast-4.16
      fast-4.17
      stable-4.16
    Image:      quay.io/openshift-release-dev/ocp-
release@sha256:a5bfe05de4c9a5cf4a5609105c539e6f8ba92731f068f13bbac94ceb855ce1d7
    URL:        https://access.redhat.com/errata/RHSA-2024:7944
    Version:    4.16.17
...output omitted...
```

The `oc adm must-gather` command collects the likely needed information from your cluster for debugging issues such as resource definitions and service logs. Options are used to limit the collection to specific components or time ranges, and to specify the destination directory.

The following command uses the `cnv-must-gather` container to gather information about the OpenShift Virtualization features:

```
[student@host ~]$ oc adm must-gather \
--image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel9:v4.16.2
...output omitted...
[must-gather-6wg6b] OUT 2024-10-30T18:16:27.049938181Z namespaces/openshift-ovn-
kubernetes/snapshot.kubevirt.io/virtualmachinesnapshots.yaml
[must-gather-6wg6b] OUT 2024-10-30T18:16:27.049950699Z namespaces/openshift-
storage/
...output omitted...
Reprinting Cluster State:
When opening a support case, bugzilla, or issue please include the following
summary data along with any other requested information:
ClusterID: 171b7a32-16f1-4139-b6d0-e18c7d8ff057
ClientVersion: 4.16.2
ClusterVersion: Stable at "4.16.2"
ClusterOperators:
    All healthy and stable
```

The output is saved locally in a directory with a name such as `must-gather.local.3458318474251885956`, unless you specify a destination with the `--dest-dir` option on the `must-gather` command.



## References

For more information about updating Red Hat OpenShift, refer to the *Red Hat OpenShift Container Platform Updating Clusters* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/updating\\_clusters/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/updating_clusters/index)

For more information about updating the OpenShift Virtualization operator, refer to the *Updating OpenShift Virtualization* chapter in the *Red Hat OpenShift Container Platform Virtualization* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/virtualization/index#upgrading-virt](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/virtualization/index#upgrading-virt)

For more information about machine configuration, refer to the documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/machine\\_configuration/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/machine_configuration/index)

For information about adding capacity to a cluster, refer to the *Scalability and Performance* guide at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/scalability\\_and\\_performance/index](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/scalability_and_performance/index)

For more information about the `must-gather` tool, refer to the *Support* documentation at  
[https://docs.redhat.com/en/documentation/openshift\\_container\\_platform/4.16/html-single/support/gathering-cluster-data#gathering-cluster-data](https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html-single/support/gathering-cluster-data#gathering-cluster-data)

## ► Quiz

# OpenShift and OpenShift Virtualization Updates

Choose the correct answers to the following questions:

- ▶ **1. Which component retrieves the updated cluster images from the internet?**
  - a. Cluster Monitoring (Prometheus)
  - b. Operator Lifecycle Manager (OLM)
  - c. Cluster Version Operator (CVO)
  - d. Telemetry client (Telemeter)
  
- ▶ **2. In which two ways can you configure operator updates? (Choose two.)**
  - a. With automatic updates, the OLM updates an operator as soon as the configured channel has a later version of the operator.
  - b. With automatic updates, the OLM switches the update channel automatically to the channel with the latest version of the operator, and updates to this version.
  - c. With manual updates, the OLM does not monitor channels, and you apply updates manually.
  - d. With manual updates, the OLM updates an operator when the configured channel has a later version of the operator, and an administrator approves the update.
  
- ▶ **3. When updates to the virt-launcher pod components are available, in which two ways are these updates applied to any running pods and VMs? (Choose two.)**
  - a. If possible, a VM is live migrated to another node and to an updated pod.
  - b. If live migration is not possible, then the VM is evicted so that it is started on another node and on an updated pod.
  - c. All VMs are restarted during the update process.
  - d. A VM that is not migrated continues to use the older pod until it is restarted.
  - e. All VMs continue to use the older pod until they are restarted.
  
- ▶ **4. Which command collects information from your cluster, such as resource definitions and service logs, for debugging issues?**
  - a. oc adm sosreport
  - b. oc adm must-gather
  - c. oc debug
  - d. oc logs

## ► Solution

# OpenShift and OpenShift Virtualization Updates

Choose the correct answers to the following questions:

- ▶ **1. Which component retrieves the updated cluster images from the internet?**
  - a. Cluster Monitoring (Prometheus)
  - b. Operator Lifecycle Manager (OLM)
  - c. Cluster Version Operator (CVO)
  - d. Telemetry client (Telemeter)
- ▶ **2. In which two ways can you configure operator updates? (Choose two.)**
  - a. With automatic updates, the OLM updates an operator as soon as the configured channel has a later version of the operator.
  - b. With automatic updates, the OLM switches the update channel automatically to the channel with the latest version of the operator, and updates to this version.
  - c. With manual updates, the OLM does not monitor channels, and you apply updates manually.
  - d. With manual updates, the OLM updates an operator when the configured channel has a later version of the operator, and an administrator approves the update.
- ▶ **3. When updates to the virt-launcher pod components are available, in which two ways are these updates applied to any running pods and VMs? (Choose two.)**
  - a. If possible, a VM is live migrated to another node and to an updated pod.
  - b. If live migration is not possible, then the VM is evicted so that it is started on another node and on an updated pod.
  - c. All VMs are restarted during the update process.
  - d. A VM that is not migrated continues to use the older pod until it is restarted.
  - e. All VMs continue to use the older pod until they are restarted.
- ▶ **4. Which command collects information from your cluster, such as resource definitions and service logs, for debugging issues?**
  - a. oc adm sosreport
  - b. oc adm must-gather
  - c. oc debug
  - d. oc logs

## ▶ Lab

# Configure Kubernetes High Availability for Virtual Machines

Implement high availability virtual machines that are resilient to failures, planned maintenance, and cluster upgrades by configuring Kubernetes resources.

## Outcomes

- Configure load balancing for a web application on two VMs.
- Configure VMs to automatically recover from a failed node.
- Put a cluster node into maintenance mode.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start ha-review
```

## Instructions

1. From a command-line window on the `workstation` machine, use the `oc` command to log in to your Red Hat OpenShift cluster as the `admin` user with `redhatocp` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`.  
Confirm that the `www1` and `www2` VMs run under the `ha-review` project.
2. Configure Kubernetes networking resources to provide high availability to the web application that is hosted on the `www1` and `www2` VMs.  
Create a web service that uses the `app: web` selector in the `ha-review` project. Confirm that the web service endpoints resolve to the IP addresses of the `www1` and `www2` VMs.  
Create a `www` route that exposes the web service for external access. Set the route host to `web-ha-review.apps.ocp4.example.com` and configure a `web` custom cookie for the route.  
The `lab` command prepares the `service.yaml` and `route.yaml` files in the `~/D0316/labs/ha-review` directory for you. Edit and then apply the files to create the web service and the `www` route.
3. Confirm that clients can access the `www` route at the `web-ha-review.apps.ocp4.example.com` host, by using the `curl` command. Confirm that OpenShift load balances the web application between the `www1` and `www2` VMs.  
Extract the `web` custom cookie from the `www` route, and then use the cookie to test session stickiness. Use the `curl` command with the `-c /tmp/cookie-jar` option to save the `web` cookie to the `workstation` machine. Confirm that the `web` cookie provides session stickiness by using the `curl` command with the `-b /tmp/cookie-jar` option.
4. Monitor the availability of the web application during a node failure. Open a command-line window and then execute the `~/D0316/labs/ha-review/loop.sh` script.

Put the cluster node that hosts the `www2` VM into maintenance mode by using a `NodeMaintenance` custom resource named `node-maintenance`, and move its workload to the remaining nodes.

The `lab` command prepared the `~/D0316/labs/ha-review/nm.yaml` resource file, which you can use as a model.

5. Configure the `www2` VM to automatically fail over to another cluster node during node failure, by setting a `LiveMigrate` eviction strategy for the VM. Then, recover the VM from the failed node and confirm that the VM responds to requests.

## Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade ha-review
```

## Finish

As the student user on the `workstation` machine, confirm that you are in the `/home/student` directory and then use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-review
```

## ► Solution

# Configure Kubernetes High Availability for Virtual Machines

Implement high availability virtual machines that are resilient to failures, planned maintenance, and cluster upgrades by configuring Kubernetes resources.

## Outcomes

- Configure load balancing for a web application on two VMs.
- Configure VMs to automatically recover from a failed node.
- Put a cluster node into maintenance mode.

## Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start ha-review
```

## Instructions

- From a command-line window on the **workstation** machine, use the **oc** command to log in to your Red Hat OpenShift cluster as the **admin** user with **redhatocp** as the password. The OpenShift cluster API endpoint is <https://api.ocp4.example.com:6443>.

Confirm that the **www1** and **www2** VMs run under the **ha-review** project.

- From a command-line window, log in to your OpenShift cluster as the **admin** user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- Change to the **ha-review** project.

```
[student@workstation ~]$ oc project ha-review
Now using project "ha-review" on server "https://api.ocp4.example.com:6443".
```

- Confirm that the **www1** and **www2** VMs run under the **ha-review** project.

```
[student@workstation ~]$ oc get vm
NAME      AGE     STATUS    READY
www1     22m     Running   True
www2     21m     Running   True
```

2. Configure Kubernetes networking resources to provide high availability to the web application that is hosted on the `www1` and `www2` VMs.

Create a web service that uses the app: `web` selector in the `ha-review` project. Confirm that the web service endpoints resolve to the IP addresses of the `www1` and `www2` VMIs.

Create a `www` route that exposes the web service for external access. Set the route host to `web-ha-review.apps.ocp4.example.com` and configure a `web` custom cookie for the route.

The lab command prepares the `service.yaml` and `route.yaml` files in the `~/D0316/labs/ha-review` directory for you. Edit and then apply the files to create the web service and the `www` route.

- 2.1. Edit the `~/D0316/labs/ha-review/service.yaml` file to configure the web service in the `ha-review` project with the app: `web` selector.

```
apiVersion: v1
kind: Service
metadata:
  name: web
  namespace: ha-review
spec:
  type: ClusterIP
  selector:
    app: web
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- 2.2. Create the service resource.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/ha-review/service.yaml
service/web created
```

- 2.3. List the VMI resources in the `ha-review` project. Note the IP addresses for the `www1` and `www2` VMIs. You use these IP addresses in a later step.

```
[student@workstation ~]$ oc get vmi
NAME          AGE     PHASE   IP           NODENAME   READY
www1          17m    Running  10.8.2.63  worker01  True
www2          6m     Running  10.9.2.71  worker02  True
```

The identified addresses in your lab might differ from the example output.

- 2.4. Confirm that the `web` service endpoints resolve to the IP addresses of the `www1` and `www2` VMIs.

```
[student@workstation ~]$ oc get endpoints
NAME      ENDPOINTS
web      10.8.2.63:80,10.9.2.71:80
```

- 2.5. Expose the web service for external access by creating a route. Edit the `~/D0316/labs/ha-review/route.yaml` file to configure the `www` route in the `ha-review`

project. Specify the `web-ha-review.apps.ocp4.example.com` host, and then configure a web custom cookie for the route.

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: www
  namespace: ha-review
  annotations:
    router.openshift.io/cookie_name: "web"
spec:
  host: web-ha-review.apps.ocp4.example.com
  port:
    targetPort: 80
  to:
    kind: Service
    name: web
```

2.6. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/ha-review/route.yaml
route.route.openshift.io/www created
```

2.7. Use the `oc` command to verify the `www` route settings.

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
www       web-ha-review.apps.ocp4.example.com        web        80  ...
```

3. Confirm that clients can access the `www` route at the `web-ha-review.apps.ocp4.example.com` host, by using the `curl` command. Confirm that OpenShift load balances the web application between the `www1` and `www2` VMIs.  
Extract the `web` custom cookie from the `www` route, and then use the cookie to test session stickiness. Use the `curl` command with the `-c /tmp/cookie-jar` option to save the `web` cookie to the `workstation` machine. Confirm that the `web` cookie provides session stickiness by using the `curl` command with the `-b /tmp/cookie-jar` option.

3.1. Use the `curl` command to verify that the `www` route is accessible.

```
[student@workstation ~]$ curl web-ha-review.apps.ocp4.example.com
Welcome to www1
```

The response shows the server that answered the HTTP query.

- 3.2. Use the `~/D0316/labs/ha-review/loop.sh` script to send requests to the `www` route. Confirm that the web application is load balanced between the `www1` and `www2` VMs. Press `Ctrl+C` to stop the command after you see responses from each of the servers.

```
[student@workstation ~]$ ~/D0316/labs/ha-review/loop.sh
"Welcome to www1"
"Welcome to www2"
"Welcome to www1"
"Welcome to www2"
"Welcome to www1"
^C
```

- 3.3. Use the `curl` command to save the web cookie to the workstation machine. Note the server that responds to the request.

```
[student@workstation ~]$ curl web-ha-review.apps.ocp4.example.com \
  -c /tmp/cookie-jar
Welcome to www1
```

- 3.4. Confirm that the web cookie provides session stickiness. Use the `curl` command with the web cookie to access the `www` route to verify the same response as the previous step.

```
[student@workstation ~]$ curl web-ha-review.apps.ocp4.example.com \
  -b /tmp/cookie-jar
Welcome to www1
```

4. Monitor the availability of the web application during a node failure. Open a command-line window and then execute the `~/D0316/labs/ha-review/loop.sh` script.

Put the cluster node that hosts the `www2` VMI into maintenance mode by using a `NodeMaintenance` custom resource named `node-maintenance`, and move its workload to the remaining nodes.

The `lab` command prepared the `~/D0316/labs/ha-review/nm.yaml` resource file, which you can use as a model.

- 4.1. Open a command-line window on the `workstation` machine. Execute the `~/D0316/labs/ha-review/loop.sh` file.

```
[student@workstation ~]$ ~/D0316/labs/ha-review/loop.sh
Welcome to www2
Welcome to www2
Welcome to www1
Welcome to www1
...output omitted...
```

- 4.2. Change to the command-line window where you confirmed the new VMI instance. Identify the node that hosts the `www2` VMI. You use that node in a later step.

```
[student@workstation ~]$ oc get vmi
NAME          AGE     PHASE      IP           NODENAME   READY
www1          19m    Running    10.8.2.63   worker01   True
www2          8m     Running    10.9.2.71   worker02   True
```

**Important**

The identified node in your lab might differ from the example output. Be sure to modify your commands accordingly.

- 4.3. From the terminal on the **workstation** machine, edit the `~/D0316/labs/ha-review/nm.yaml` file:

```
apiVersion: nodemaintenance.medik8s.io/v1beta1
kind: NodeMaintenance
metadata:
  name: node-maintenance
spec:
  nodeName: worker02
  reason: "Node maintenance"
```

- 4.4. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/D0316/labs/ha-review/nm.yaml
nodemaintenance.nodemaintenance.medik8s.io/node-maintenance created
```

- 4.5. Confirm that the node has the **SchedulingDisabled** status.

```
[student@workstation ~]$ oc get nodes
NAME      STATUS            ROLES
master01   Ready            control-plane, master, worker ...
master02   Ready            control-plane, master, worker ...
master03   Ready            control-plane, master, worker ...
worker01   Ready            nodeAgent, worker ...
worker02  Ready, SchedulingDisabled  nodeAgent, worker ...
```

**Note**

If the OADP operator is not installed on your cluster, then the roles on your nodes might differ from the example output.

- 4.6. Wait until the **node-maintenance** CR reports a **Succeeded** status.

```
[student@workstation ~]$ oc describe nm node-maintenance
Name:          node-maintenance
...output omitted...
API Version:  nodemaintenance.medik8s.io/v1beta1
Kind:          NodeMaintenance
...output omitted...
Status:
  Drain Progress: 100
  Eviction Pods: 23
  Last Error:     error when evicting pods/"virt-launcher-www1-wccmq" -n "ha-review": global timeout reached: 30s
  Last Update:    2024-10-31T14:35:48Z
```

Phase:	<b>Succeeded</b>
Totalpods:	43
Events:	<none>

- 4.7. From a command-line window, confirm that the `www2` VM is no longer running.

```
[student@workstation ~]$ oc get vmi
NAME      AGE     PHASE   IP          NODENAME  READY
www1     22      Running  10.8.2.63  worker01  True
```

The `www2` VM has an eviction strategy of `None` and was stopped when the node was drained.

5. Configure the `www2` VM to automatically fail over to another cluster node during node failure, by setting a `LiveMigrate` eviction strategy for the VM. Then, recover the VM from the failed node and confirm that the VM responds to requests.
  - 5.1. Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`
  - 5.2. Click `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
  - 5.3. From the OpenShift web console, go to **Virtualization > VirtualMachines**. Select the `www2` VM and click the **Configuration** menu.
  - 5.4. Go to the **Scheduling** section, and click **None** in the **Eviction strategy** subsection. Click the **LiveMigrate** flag and click **Save**.
  - 5.5. Click **Actions > Start** to power on and reschedule the VM on another node in the cluster. Wait a few moments before you proceed to the next step.
  - 5.6. From a command-line window, confirm that the `www2` VM is scheduled on another node.

```
[student@workstation ~]$ oc get vmi
NAME      AGE     PHASE   IP          NODENAME  READY
www1     23m    Running  10.8.2.63  worker01  True
www2     45s    Running  10.9.0.46  master02  True
```

- 5.7. Return to the command-line window that executes the `loop.sh` file. Confirm that the `www2` VM responds to requests. Press `Ctrl+C` to stop the command, and then close the command-line window.

```
...output omitted...
Welcome to www1
Welcome to www1
Welcome to www1
Welcome to www2
Welcome to www1
Welcome to www2
Welcome to www2
Welcome to www1
Welcome to www1
```

```
Welcome to www2
^C
[student@workstation ~]$ exit
```

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade ha-review
```

## Finish

As the **student** user on the **workstation** machine, confirm that you are in the **/home/student** directory and then use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-review
```

# Summary

---

- Putting a cluster node into maintenance mode requires cordoning off the node and draining the node's workload.
- During the draining process, VMs that support live migration are moved uninterrupted. Other VMs are stopped and then restarted on a different node.
- You create a `NodeMaintenance` resource to put a node into maintenance mode. You delete the resource to remove the node from maintenance mode.
- You can also use the `oc adm cordon`, `oc adm drain`, and `oc adm uncordon` commands for managing maintenance mode.
- Eviction strategies determine the appropriate actions to take with orphaned resources, such as pods, persistent volumes, and VMs.
- Machine health checks automatically remediate an unhealthy machine, if the machine exists in a particular machine pool.
- As the standard remediation for unhealthy nodes, OpenShift drains the remaining VMs on the failed node, and reschedules them to healthy nodes so that the failed node can register itself again within the cluster and restore its annotations and labels.
- Power-based remediation uses a power controller to power off an inoperable node.
- Use the Self Node Remediation Operator to automatically reboot unhealthy nodes.
- Deleting a node deletes the node object in Kubernetes, but it does not delete the pods on the node.
- Upgrades to the `virt-launcher` pod require starting the new container on another node by either live migrating or restarting the VM.

## Chapter 10

# Comprehensive Review

### Goal

Review tasks from *Managing Virtual Machines with Red Hat OpenShift Virtualization*.

### Sections

- Comprehensive Review

### Lab

- Deploy OpenShift Virtualization and Configure External Network Access for a Virtual Machine
- Use a Template to Deploy a Virtual Machine and Prepare a Node for Maintenance
- Clone a Virtual Machine and Configure Load Balancing

# Comprehensive Review

---

## Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills that you learned in *Managing Virtual Machines with Red Hat OpenShift Virtualization*.

### Reviewing Managing Virtual Machines with Red Hat OpenShift Virtualization

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter. Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

#### **Chapter 1, Red Hat OpenShift Virtualization**

Distinguish Red Hat OpenShift Virtualization from container technologies and from traditional virtual machine technologies. Describe the features and use cases of OpenShift Virtualization. Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift environment.

#### **Chapter 2, Running and Accessing Virtual Machines**

Create, manage, inspect, and monitor virtual machines in Red Hat OpenShift Virtualization.

#### **Chapter 3, Configuring Kubernetes Networking for Virtual Machines**

Configure standard Kubernetes network objects and external access for virtual machines and virtual machine-backed applications.

#### **Chapter 4, Connecting Virtual Machines to External Networks**

Configure node networking to connect virtual machines and nodes to networks outside of the cluster by using Multus CNI plug-ins and the NMState operator.

#### **Chapter 5, Configuring Storage for Virtual Machines**

Manage storage and disks for virtual machines in Red Hat OpenShift by using Kubernetes.

#### **Chapter 6, Create and Restore Backups of Virtual Machines**

Create virtual machine snapshots and back up virtual machine components individually and by using the OpenShift APIs for Data Protection (OADP) operator.

#### **Chapter 7, Replicating Virtual Machines by Using Instance Types, Templates, and Clones**

Create and manage clones, templates, and instance types to provision virtual machines.

## **Chapter 8, Advanced Virtual Machine Management**

Import, control the placement of, monitor the health of, and live migrate virtual machines.

## **Chapter 9, Configuring Kubernetes High Availability for Virtual Machines**

Implement high availability virtual machines that are resilient to failures, planned maintenance, and cluster upgrades by configuring Kubernetes resources.

## ► Lab

# Deploy OpenShift Virtualization and Configure External Network Access for a Virtual Machine

Configure node networking, connect a virtual machine to an additional network, and configure SSH access for a virtual machine.

## Outcomes

- Configure cluster nodes with an additional network.
- Prepare a project for an additional network.
- Deploy a virtual machine with multiple network interfaces.
- Configure SSH access to a virtual machine.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable, creates the `review-cr1` namespace, and creates the `lab-rsa` secret in the `review-cr1` namespace. This command also installs and instantiates the Kubernetes NMState operator.

```
[student@workstation ~]$ lab start review-cr1
```

## Specifications

The Red Hat OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443` and the OpenShift web console is available at `https://console-openshift-console.apps.ocp4.example.com`. Log in to your OpenShift cluster as the `admin` user with `redhatocp` as the password. The `oc` command is available on the `workstation` machine.

- Add the `orgnet=true` label to the `worker01` and `worker02` cluster nodes.
- The `worker01` and `worker02` cluster nodes have an `ens4` network interface, which is connected to an additional network. Create a Linux bridge named `br0` on each node, and then add the `ens4` interface to that bridge.
  - Configure the bridge interface to request an IPv4 address from DHCP.
  - Use the `orgnet: true` label to select the nodes to configure.
- Create a network attachment definition resource in the `review-cr1` namespace with the following characteristics:

Parameter	Value
Project	review-cr1
Resource name	ext-net
Network type	Linux bridge
Bridge	br0

- Create a virtual machine named `web1` in the `review-cr1` namespace with the Red Hat Enterprise Linux 9 VM template and the **Template default** disk source.
  - Use `Cloud-init` to add the `developer` user with `developer` as the password.
  - Add a public SSH key to the `web1` VM by using the `lab-rsa` secret in the `review-cr1` namespace that the `Lab` command prepares for you.
  - Add a second network interface named `nic-0`, which is connected to the bridge that you declared in the `ext-net` network attachment definition resource.
- Ensure that the `web1` VM is running before grading your work.

## Evaluation

As the `student` user on the `workstation` machine, use the `Lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade review-cr1
```

## Finish

On the `workstation` machine, change to the `student` user home directory and use the `Lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish review-cr1
```

## ► Solution

# Deploy OpenShift Virtualization and Configure External Network Access for a Virtual Machine

Configure node networking, connect a virtual machine to an additional network, and configure SSH access for a virtual machine.

### Outcomes

- Configure cluster nodes with an additional network.
- Prepare a project for an additional network.
- Deploy a virtual machine with multiple network interfaces.
- Configure SSH access to a virtual machine.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable, creates the `review-cr1` namespace, and creates the `lab-rsa` secret in the `review-cr1` namespace. This command also installs and instantiates the Kubernetes NMState operator.

```
[student@workstation ~]$ lab start review-cr1
```

- Open a web browser and log in to the OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`. Log in to your OpenShift cluster as the `admin` user with `redhatocp` as the password.

- Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`. Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.



#### Note

If the web console is open from a previous exercise, then refresh your browser to complete the Kubernetes NMState operator installation.

- Label the `worker01` and `worker02` nodes with the `orgnet=true` label.
  - From the OpenShift web console, go to `Compute > Nodes`, click the vertical ellipsis icon : next to `worker01`, and click `Edit labels`.
  - Type `orgnet=true` in the field and click `Save`.

- 2.3. Click the vertical ellipsis icon next to `worker02`, and click **Edit labels**.
  - 2.4. Type `orgnet=true` in the field and click **Save**.
  - 2.5. Go to the **Compute > Nodes** page and list the nodes with the `orgnet=true` label. Select **Label** from the drop-down menu, type `orgnet` in the **Search by label** field, and select the `orgnet=true` label.
  - 2.6. Verify that the node list contains only the `worker01` and `worker02` nodes.
3. Configure a Linux bridge named `br0` for each cluster node with the `orgnet=true` label. The Linux bridge has the following requirements:

Parameter	Value
Name	<code>br0-ens4-policy</code>
Interface name	<code>br0</code>
Network state	<code>Up</code>
Interface type	<code>Bridge</code>
IPv4	<code>Enabled</code>
DHCP	<code>Enabled</code>
Bridge port	<code>ens4</code>
Node selector	<code>orgnet=true</code>

- 3.1. Go to **Networking > NodeNetworkConfigurationPolicy** and click **Create NodeNetworkConfigurationPolicy**.
- 3.2. Select the **Apply this NodeNetworkConfigurationPolicy** checkbox to open the node selector.

### Create NodeNetworkConfigurationPolicy

[Edit YAML](#)

Node network is configured and managed by NM state. Create a node netwrok configuration policy to describe the requested network configuration on your nodes in the cluster. The node network configuration enactment reports the netwrok policies enacted upon each node.

Apply this NodeNetworkConfigurationPolicy only to specific subsets of nodes using the node selector ⓘ

- 3.3. Click **Add Label**, and type `orgnet` as the key and `true` as the value. Click **Save**.
- 3.4. Complete the **Create NodeNetworkConfigurationPolicy** form with the provided specifications and click **Create**.
- 3.5. Confirm that the policy is successfully configured on the nodes by verifying its status on the **Networking > NodeNetworkConfigurationPolicy** page. Wait until the node network state is `2 Available`.

Name	Matched nodes	Node network state
NNCP br0-ens4-policy	2 nodes	✓ 2 Available

4. Create a network attachment definition resource named `ext-net` for the `br0` Linux bridge in the `review-cr1` namespace.
  - 4.1. From the OpenShift web console, go to **Networking > NetworkAttachmentDefinitions**, select the `review-cr1` project, and click **Create network attachment definition**.
  - 4.2. Complete the form by using the following information and click **Create**.

Parameter	Value
Name	<code>ext-net</code>
Namespace	<code>review-cr1</code>
Network type	<code>Linux bridge</code>
Bridge name	<code>br0</code>

5. Use the Red Hat Enterprise Linux 9 VM template to create the `web1` VM in the `review-cr1` namespace.  
Configure `Cloud-init` to add the `developer` user with `developer` as the password.  
Add a public SSH key to the `web1` VM by using the `lab-rsa` secret in the `review-cr1` namespace that the `lab` command prepares for you.  
Add a second network interface named `nic-0`, which is connected to the bridge that you declared in the `ext-net` network attachment definition resource.
  - 5.1. Go to **Virtualization > Catalog**.
  - 5.2. Select the `review-cr1` project and the **Template catalog** tab.
  - 5.3. Locate and click the **Red Hat Enterprise Linux 9 VM** template.
  - 5.4. Specify `web1` as the VM name.
  - 5.5. Click **Customize VirtualMachine**.
  - 5.6. Go to **Network interfaces** and click **Add network interface**.
  - 5.7. Complete the form by using the following information, and click **Save**.

Parameter	Value
Name	nic-0
Model	virtio
Network	review-cr1/ext-net

- 5.8. Go to the **Scripts** tab. Click **Edit** to the right of **Cloud-init**.
  - 5.9. Add the **developer** user with developer as the password and click **Apply**.
  - 5.10. Scroll down and click **Edit** next to **Public SSH key**. Click **Use existing** and select the **lab-rsa** public key in the **review-cr1** project. Click **Save**.
  - 5.11. Click **Create VirtualMachine**.
  - 5.12. Go to **Virtualization > VirtualMachines** and then wait for the **web1** VM to have the **Running** status.
  - 5.13. Click **web1** to open the **Overview** tab for the **web1** VM. Locate the **Network interfaces (2)** panel and confirm that the **nic-0** interface on the VM has an IP address in the **192.168.51.0/24** address range.
- 6.** From the command line, log in to your OpenShift cluster as the **admin** user with **redhatocp** as the password. Confirm that the **developer** user can connect to the **web1** VM by using the **.ssh/lab\_rsa** key and the **virtctl ssh** command.

- 6.1. Log in to your OpenShift cluster as the **admin** user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 6.2. Use the **virtctl ssh** command to confirm that the **developer** user can connect to the **web1** VM by using the **.ssh/lab\_rsa** private key.

```
[student@workstation ~]$ virtctl -n review-cr1 ssh -i .ssh/lab_rsa developer@web1
Warning: Permanently added 'vmi/web1.review-cr1' (ED25519) to the list of known
hosts.
...output omitted...
```

- 6.3. Close the SSH connection.

```
[developer@web1 ~]$ exit
logout
Connection to vmi/web1.review-cr1 closed.
```

## Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade review-cr1
```

## Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish review-cr1
```

## ▶ Lab

# Use a Template to Deploy a Virtual Machine and Prepare a Node for Maintenance

Create a virtual machine template, use that template to deploy a virtual machine, seal the root disk of a virtual machine, create a bootable volume by using a custom instance type and a sealed image, and put a cluster node into maintenance mode.

## Outcomes

- Create a virtual machine template.
- Deploy a virtual machine from a template.
- Seal the root disk of a virtual machine.
- Create a bootable volume by using a custom instance type and a sealed image.
- Manage user access rights.
- Put a cluster node into maintenance mode.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable, sets the default storage class to `ocs-external-storagecluster-ceph-rbd-virtualization`, and creates the `review-cr2` namespace. This command also adds the `developer` user to the `vm-admins` group and installs the Node Maintenance Operator.

```
[student@workstation ~]$ lab start review-cr2
```

## Specifications

The Red Hat OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443` and the OpenShift web console is available at `https://console-openshift-console.apps.ocp4.example.com`. Use the `admin` user with `redhatocp` as the password to log in to your OpenShift cluster. The `oc` command is available on the `workstation` machine.

During this exercise, you also use the `developer` user account with `developer` as the password. The `developer` user is a member of the `vm-admins` group.

- Create a virtual machine template in the `review-cr2` project with the following properties:

Parameter	Value
Project	review-cr2

Parameter	Value
Template name and display name	dev-web-rhel9
Template provider	Red Hat Training
Operating system	Red Hat Enterprise Linux 9.0 or later
Boot disk URL	<a href="http://utility.lab.example.com:8080/">http://utility.lab.example.com:8080/</a> openshift4/images/rhel9-helloworld.qcow2
Size	Tiny
Workload type	Server
Root disk size	10 GiB
Root disk interface	virtio
Root disk storage class	ocs-external-storagecluster-ceph-rbd-virtualization
Cloud-init user	developer
Cloud-init password	developer

- Modify the template to use the template variable, \${NAME}, to configure the name of the rootdisk data volume to the name of the VM that the template creates.

The lab command prepares the ~/D0316/labs/review-cr2/template\_parameters.txt file, which lists these parameters. You can copy and paste the parameters from this file into the OpenShift web console.

- Grant admin access to the vm-admins group for the review-cr2 project.
- As the developer user, create a VM named web-dev in the review-cr2 namespace. Use the dev-web-rhel9 template.
- Update the contents of the web-dev machine. Add an "Authorized Use Only" message to the /etc/motd file and then shut down the machine.
- As the admin user, remove the system-specific information and change the hostname to localhost.localdomain to seal the web-dev rootdisk PVC.
- Create the web-server-prod instance type with 2 CPUs and 6 GiB RAM.
- Create a web-prod bootable volume in the review-cr2 namespace with the sealed web-dev image as the source.
  - Configure the bootable volume to use the ocs-external-storagecluster-ceph-rbd-virtualization storage class and a 30 Gi disk size.
  - Set the default instance type to web-server-prod and the default preference to rhel.9.
- Create the web1 VM by using the web-prod bootable image and the web-server-prod instance type. Use Cloud-init to add the devops user with devops as the password.

- Put the `worker02` cluster node into maintenance mode by using the command line, and move its workload to the remaining nodes.
- Ensure that the `web1` VM is running before grading your work.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade review-cr2
```

## Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish review-cr2
```

## ► Solution

# Use a Template to Deploy a Virtual Machine and Prepare a Node for Maintenance

Create a virtual machine template, use that template to deploy a virtual machine, seal the root disk of a virtual machine, create a bootable volume by using a custom instance type and a sealed image, and put a cluster node into maintenance mode.

## Outcomes

- Create a virtual machine template.
- Deploy a virtual machine from a template.
- Seal the root disk of a virtual machine.
- Create a bootable volume by using a custom instance type and a sealed image.
- Manage user access rights.
- Put a cluster node into maintenance mode.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable, sets the default storage class to `ocs-external-storagecluster-ceph-rbd-virtualization`, and creates the `review-cr2` namespace. This command also adds the `developer` user to the `vm-admins` group and installs the Node Maintenance Operator.

```
[student@workstation ~]$ lab start review-cr2
```

1. From a command line on the `workstation` machine, use the `oc` command to log in to your OpenShift cluster as the `admin` user with `redhatocp` as the password.  
Open a web browser and log in to the OpenShift web console at `https://console-openshift-console.apps.ocp4.example.com`
  - 1.1. From a command line, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp \
https://api.ocp4.example.com:6443
Login Successful
...output omitted...
```

- 1.2. Open a web browser and go to <https://console-openshift-console.apps.ocp4.example.com>. Select **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.
2. Create the **dev-web-rhel9** VM template in the **review-cr2** project.
  - 2.1. Go to **Virtualization > Templates** and ensure that **All Projects** is selected in the **Project** list. Enter **rhel9-server** in the **Search by Name** field.
  - 2.2. Click the vertical ellipsis icon : next to the **rhel9-server-tiny** line, and click **Clone**.
  - 2.3. Complete the form by using the following information and click **Clone**.

Field	Value
Template name	<b>dev-web-rhel9</b>
Template project	<b>review-cr2</b>
Template display name	<b>dev-web-rhel9</b>
Template provider	<b>Red Hat Training</b>

- 2.4. From the **dev-web-rhel9 Details** page, go to the **Disks** tab.
- 2.5. Click the vertical ellipsis icon next to the **rootdisk** line, and click **Detach**. Click **Detach** to confirm.
- 2.6. Click **Add disk**. Complete the form by using the following information and click **Save**.

Field	Value
Use this disk as a boot source	<i>Selected</i>
Name	<b>rootdisk</b>
Source	<b>From URL</b>
URL	<a href="http://utility.lab.example.com:8080/openshift4/images/rhel9-helloworld.qcow2">http://utility.lab.example.com:8080/openshift4/images/rhel9-helloworld.qcow2</a>
PersistentVolumeClaim Size	10 GiB
Type	<b>Disk</b>
Interface	<b>virtio</b>
Storage Class	<b>ocs-external-storagecluster-ceph-rbd-virtualization</b>
Apply optimized StorageProfile settings	<i>Selected</i>

- 2.7. Use the YAML editor to set the name of the **rootdisk** data volume to the name of the VM that the template creates. Go to the **YAML** tab and locate the `spec.dataVolumeTemplates[0].metadata.name` and

`spec.template.spec.volumes[1].dataVolume.name` objects. Update the values of the objects to use the  `${NAME}`  template variable and click **Save**.

```
apiVersion: template.openshift.io/v1
kind: Template
...output omitted...
spec:
  dataVolumeTemplates:
    - metadata:
        name: '${NAME}'
...output omitted...
  template:
    volumes:
      - name: rootdisk
        dataVolume:
          name: '${NAME}'
...output omitted...
```

- 2.8. Configure Cloud-init to create the developer user with developer as the password. Select the **Scripts** tab and click **Edit** to the right of Cloud-init. Complete form according to the following specification and click **Apply**.

Field	Value
User	developer
Password	developer
Add network data	<i>Not selected</i>

3. Grant admin access to the `vm-admins` group for the `review-cr2` project.

- 3.1. Use the `oc create rolebinding` command to grant admin access to the `vm-admins` group.

```
[student@workstation ~]$ oc create rolebinding admin --clusterrole admin \
--group vm-admins -n review-cr2
rolebinding.rbac.authorization.k8s.io/admin created
```

- 3.2. Confirm that the `vm-admins` group has admin rights.

```
[student@workstation ~]$ oc get rolebindings admin -n review-cr2 -o wide
NAME      ROLE           AGE     USERS      GROUPS      SERVICEACCOUNTS
admin     ClusterRole/admin  39s      vm-admins
```

4. As the developer user, use the `dev-web-rhel9` template to create the `web-dev` VM in the `review-cr2` namespace.

- 4.1. From the OpenShift web console, click **admin > Log out**.
- 4.2. Select `htpasswd_provider` and log in as the developer user with developer as the password.
- 4.3. Select the **Administrator** perspective.

- 4.4. Go to Virtualization > Catalog and set the project to review-cr2.

**Note**

If you cannot see the catalog, then log out and access the web console as the **admin** user, and then log out again and access the web console as the **developer** user.

- 4.5. Select the **Template catalog** tab. Click **User templates** and select the **dev-web-rhel9** template.
- 4.6. Set the **Virtual Machine Name** field to **web-dev** and click **Quick create VirtualMachine**. Wait for the **web-dev** VM to have the **Running** status.
- 4.7. Log out of the web console.
5. From the command line, log in to your OpenShift cluster as the **developer** user with **developer** as the password. Log in to the **web-dev** VM as the **developer** user by using the **virtctl** command. Use the **sudo -i** command to become the **root** user. Add an "Authorized Use Only" message to the **/etc/motd** file and then shut down the machine.
  - 5.1. From a command line, log in to your OpenShift cluster as the **developer** user.

```
[student@workstation ~]$ oc login -u developer -p developer
```

- 5.2. Select the **review-cr2** project.

```
[student@workstation ~]$ oc project review-cr2
Now using project "review-cr2" on server "https://api.ocp4.example.com:6443"
```

- 5.3. Log in to the **web-dev** console by using the **virtctl vnc** command. Log in as the **developer** user with **developer** as the password.

```
[student@workstation ~]$ virtctl vnc web-dev -n review-cr2
```

```
web-dev login: developer
Password: developer
```

- 5.4. Use the **sudo -i** command to become the **root** user.

```
[developer@web-dev ~]$ sudo -i
```

- 5.5. Update the **/etc/motd** file.

```
[root@web-dev ~]# echo "Authorized Use Only" > /etc/motd
```

- 5.6. Log out of the VM.

```
[root@web-dev ~]# exit
[developer@web-dev ~]$ logout
```

- 5.7. Close the VM console window.
6. Seal the web-dev rootdisk PVC.
- 6.1. Use the `virtctl` command to stop the web-dev VM.
- ```
[student@workstation ~]$ virtctl stop web-dev -n review-cr2
```
- 6.2. Start a container with the libguestfs tools that is connected to the web-dev PVC.
- ```
[student@workstation ~]$ virtctl guestfs web-dev -n review-cr2
Use image: registry.redhat.io/container-native-virtualization/libguestfs-tools...
The PVC has been mounted at /dev/vda
Waiting for container libguestfs still in pending, reason: ContainerCreating,
message:
...output omitted...
If you don't see a command prompt, try pressing enter.
bash-5.1$
```
- 6.3. Remove the system-specific information from the web-dev image and change the hostname to `localhost.localdomain`.

```
bash-5.1$ virt-sysprep -a /dev/vda --hostname localhost.localdomain
[ 0.0] Examining the guest ...
[ 5.5] Performing "abrt-data" ...
...output omitted...
[ 6.9] Setting the machine ID in /etc/machine-id
[ 6.9] Setting the hostname: localhost.localdomain
[ 6.9] SELinux relabelling
[ 28.5] Performing "lvm-uuids" ...
```

- 6.4. Exit the container.
- ```
bash-5.1$ exit
[student@workstation ~]$
```
7. Create the web-server-prod instance type with 2 CPUs and 6 GiB RAM. Create a web-prod bootable volume with the sealed web-dev PVC as the source. Set the default instance type to web-server-prod and the default preference to `rhel.9`.
- 7.1. Use the `virtctl` command to generate the YAML output for an instance type resource.

```
[student@workstation ~]$ virtctl create instancetype \
--name web-server-prod --cpu 2 --memory 6Gi
apiVersion: instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterInstancetype
metadata:
  creationTimestamp: null
  name: web-server-prod
spec:
  cpu:
```

```
guest: 2
memory:
guest: 6Gi
```

- 7.2. Create the instance type.

```
[student@workstation ~]$ virtctl create instancetype \
--name web-server-prod --cpu 2 --memory 6Gi | oc apply -f -
virtualmachineclusterinstancetype.instancetype.kubevirt.io/web-server-prod created
```

- 7.3. Return to the OpenShift web console. Select `htpasswd_provider` and log in as the `admin` user with `redhatocp` as the password.
- 7.4. Go to **Virtualization > Bootable volumes** and select the `review-cr2` project.
- 7.5. Click **Add bootable volume** and select **With form**.
- 7.6. Complete the form with the following values and click **Save** to start the clone process.

| Field                 | Value   |
|-----------------------|---|
| Source type           | Use existing volume                                 |
| PVC project           | review-cr2  |
| PVC name              | web-dev   |
| Storage class         | ocs-external-storagecluster-ceph-rbd-virtualization |
| Disk size             | 30 Gi   |
| Volume name           | web-prod  |
| Destination project   | review-cr2  |
| Preference            | rhel.9  |
| Default instance type | User provided > web-server-prod                     |

It can take a few minutes to provision the image and to create the datasource.

8. Create the `web1` VM by using the `web-prod` bootable image and the `web-server-prod` instance type. Use `Cloud-init` to add the `devops` user with `devops` as the password.
- 8.1. Go to **Virtualization > Catalog** and select the `review-cr2` project.
- 8.2. Select the `web-prod` bootable volume and the `web-server-prod` instance type. Set the name of the VM to `web1` and click **Customize VirtualMachine**.
- 8.3. Go to **Initial run** and click **Edit** next to `Cloud-init`. Add the `devops` user and use `devops` as the password. Click **Apply** to save the changes.
- 8.4. Click **Create VirtualMachine** and wait until the VM status is **Running**.
9. As the `admin` user, put the `worker02` cluster node into maintenance mode and drain its workload.

- 9.1. From the command line, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhatocp
```

- 9.2. Use the `oc adm cordon` command to mark the node as unschedulable.

**Note**

You can also use the Node Maintenance operator to put the `worker02` node into maintenance mode. From the OpenShift web console, go to **Operators > Installed Operators** and open the **Node Maintenance Operator** page. In the **Node Maintenance** card, click **Create Instance** and complete the form.

```
[student@workstation ~]$ oc adm cordon worker02  
node/worker02 cordoned
```

- 9.3. Confirm that the node has the `SchedulingDisabled` status.

```
[student@workstation ~]$ oc get nodes  
NAME      STATUS            ROLES          AGE   VERSION  
master01   Ready            control-plane, master, worker   19d  
           v1.27.10+28ed2d7  
master02   Ready            control-plane, master, worker   19d  
           v1.27.10+28ed2d7  
master03   Ready            control-plane, master, worker   19d  
           v1.27.10+28ed2d7  
worker01   Ready            worker          19d  
           v1.27.10+28ed2d7  
worker02   Ready, SchedulingDisabled   worker          19d  
           v1.27.10+28ed2d7
```

- 9.4. Run the `oc adm drain` command to evacuate all workloads from the node. This command might take a few minutes to complete.

```
[student@workstation ~]$ oc adm drain worker02 --force\  
--delete-emptydir-data --ignore-daemonsets  
node/worker02 already cordoned  
...output omitted...  
node/worker02 drained
```

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade review-cr2
```

## Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish review-cr2
```

## ▶ Lab

# Clone a Virtual Machine and Configure Load Balancing

Clone a virtual machine and then configure the clones to provide a highly available web application.

### Outcomes

- Clone a virtual machine.
- Take a snapshot of a virtual machine.
- Configure readiness probes.
- Attach PVCs as disks to virtual machines.
- Create a service of the ClusterIP type.
- Link a service to virtual machines.
- Create a route.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command also creates the `review-cr3` namespace and starts the `golden-web` virtual machine in that namespace.

```
[student@workstation ~]$ lab start review-cr3
```

### Specifications

The Red Hat OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443` and the OpenShift web console is available at `https://console-openshift-console.apps.ocp4.example.com`. Use the `admin` user and `redhatocp` as the password to log in to your OpenShift cluster. The `oc` command is available on the `workstation` machine. All the resources that you create must belong to the `review-cr3` namespace.

- Clone the `golden-web` VM. Use `web1` for the name of the clone.
- Clone the `golden-web` VM a second time. Use `web2` for the name of the clone.
- Create a snapshot of the `web1` VM. Use `web1-snap1` for the name of the snapshot.
- The web application that is running inside the VMs exposes the `/cgi-bin/health` endpoint on port 80. Add a readiness probe to the `web1` and `web2` VMs. The probe must monitor the endpoint every five seconds and must fail after two unsuccessful attempts. The `lab` command prepares the `~/D0316/labs/review-cr3/readiness.yaml` file, which you can use as an example.

- The `lab` command prepares the `web1-documentroot` and `web2-documentroot` PVCs. They contain web content for the `web1` and `web2` VMs. Attach the `web1-documentroot` PVC as the `web1-disk` to the `web1` VM. Attach the `web2-documentroot` PVC as the `web2-disk` to the `web2` VM. Use the `virtio` interface when attaching the disks.
- Access the console of the `web1` VM. Log in as the `root` user with `redhat` as the password. Run the `/root/mount.sh` script, which the `lab` command prepares. The script mounts the additional disk in the `/var/www/html` directory. Perform the same operation for the `web2` VM.
- Add the `tier: front` label to the `web1` and `web2` VM resources. Red Hat OpenShift Virtualization must automatically assign this label to the virtual machine instance (VMI) resources that it creates when you start the VMs.
- Create a service named `front` with the `ClusterIP` type. The service must dispatch the web requests between the VMs with the `tier: front` label. These VMs host a web service that listens on port TCP 80. The `lab` command prepares the `~/D0316/labs/review-cr3/service.yaml` file, which you can use as an example.
- Create a route named `front` that sends the requests from external clients to the `front` service. Ensure that the URL for the route is `https://front-review-cr3.apps.ocp4.example.com`. Add TLS edge termination to encrypt traffic between the client and the router.

You can test the route and the service by using the `curl https://front-review-cr3.apps.ocp4.example.com` command from the command line on the `workstation` machine. You can confirm that the service correctly dispatches the requests between the two VMs by running the `curl` command several times. The `web1` and `web2` VMs serve different web content.

- Ensure that the `web1` and `web2` VMs are running before grading your work.

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade review-cr3
```

## Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish review-cr3
```

## ► Solution

# Clone a Virtual Machine and Configure Load Balancing

Clone a virtual machine and then configure the clones to provide a highly available web application.

### Outcomes

- Clone a virtual machine.
- Take a snapshot of a virtual machine.
- Configure readiness probes.
- Attach PVCs as disks to virtual machines.
- Create a service of the ClusterIP type.
- Link a service to virtual machines.
- Create a route.

### Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

This command ensures that the cluster API is reachable. The command also creates the **review-cr3** namespace and starts the **golden-web** virtual machine in that namespace.

```
[student@workstation ~]$ lab start review-cr3
```

1. Open a web browser on the **workstation** machine and go to the OpenShift web console at <https://console-openshift-console.apps.ocp4.example.com>. Log in to your OpenShift cluster as the **admin** user with **redhatocp** as the password.
  - 1.1. Open a web browser and go to <https://console-openshift-console.apps.ocp4.example.com>. Select **htpasswd\_provider** and log in as the **admin** user with **redhatocp** as the password.
2. Clone the **golden-web** VM to a new VM named **web1**.
  - 2.1. The cloning process requires you to stop the source machine. Go to **Virtualization > VirtualMachines** and change to the **review-cr3** project. Select the **golden-web** VM and click **Actions > Stop**. Wait for the machine to stop.
  - 2.2. Click **Actions > Clone**. Set the VM name to **web1**, select **Start VirtualMachine once created**, and click **Clone**.
  - 2.3. Go to **Virtualization > VirtualMachines** and confirm that the new **web1** VM is running.

- 2.4. Repeat this step to create a clone named web2.
3. Take a snapshot of the web1 VM. The name of the snapshot must be web1-snap1.
    - 3.1. Select the web1 VM, go to the **Snapshots** tab, and click **Take snapshot**.
    - 3.2. Enter web1-snap1 in the **Name** field and click **Save**.
    - 3.3. Wait for the snapshot to report the **Operation complete** status.
  4. The web servers that are running inside the web1 and web2 VMs expose the /cgi-bin/health endpoint to monitor the application status. Add a readiness probe to the web1 and web2 VMs that uses HTTP GET requests to test the /cgi-bin/health endpoint.
    - 4.1. Go to **Virtualization > VirtualMachines**, select the web1 VM, and go to the **YAML** tab. Use the YAML editor to declare the probe and click **Save**. You can copy and paste the **readinessProbe** section from the ~/D0316/labs/review-cr3/readiness.yaml file, which the **lab** command prepares.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
...output omitted...
template:
metadata:
...output omitted...
spec:
domain:
...output omitted...
readinessProbe:
  httpGet:
    path: /cgi-bin/health
    port: 80
    initialDelaySeconds: 60
    periodSeconds: 5
    timeoutSeconds: 2
    failureThreshold: 2
    successThreshold: 1
evictionStrategy: LiveMigrate
hostname: web1
...output omitted...
```

- 4.2. Restart the web1 VM to apply the changes. Click **Actions > Restart** and wait for the VM to reach the **Running** status before proceeding.
- 4.3. Repeat this step to add the readiness probe to the web2 VM.
5. Attach the web1-documentroot PVC as a disk named web1-disk to the web1 VM by using the **virtio** interface. Repeat this step to attach the web2-documentroot PVC as a disk named web2-disk to the web2 VM.
  - 5.1. Use the OpenShift web console to stop the web1 VM. Go to **Virtualization > VirtualMachines**, select the web1 VM, and click **Actions > Stop**. Wait for the machine to stop.

- 5.2. Go to the Configuration tab and select Storage. Click Add Disk.
- 5.3. Complete the form by using the following information and click Save:

| Field                         | Value             |
|-------------------------------|-------------------|
| Name                          | web1-disk         |
| Source                        | PVC               |
| PersistentVolumeClaim project | review-cr3        |
| PersistentVolumeClaim name    | web1-documentroot |
| Type                          | Disk              |
| Interface                     | virtio            |

**Note**

If the **Interface** field is set to `scsi`, and `virtio` is not available, click **Cancel** and start over. The web interface takes a few seconds to detect that the VM is stopped.

- 5.4. Click Actions > Start.
- 5.5. Repeat this step to attach the `web2-documentroot` PVC as a disk named `web2-disk` to the `web2` VM.
6. Log in to the `web1` VM console as the `root` user with `redhat` as the password. Then, run the `/root/mount.sh` script. Repeat this step for the `web2` VM.
  - 6.1. Go to Virtualization > VirtualMachines, select the `web1` VM, and go to the Console tab.
  - 6.2. Log in to the VM console as the `root` user with `redhat` as the password.
  - 6.3. Run the `/root/mount.sh` command.

```
[root@golden-web ~]# /root/mount.sh
...output omitted...
Mount successful
```

- 6.4. Log out of the VM console.
- 6.5. Repeat this step to mount the disk on the `web2` VM.
7. Add the `tier: front` label to the `virt-launcher` pods for the `web1` and `web2` VMs.
  - 7.1. Go to Virtualization > VirtualMachines, select the `web1` VM, and go to the YAML tab.
  - 7.2. In the YAML editor, add the `tier: front` label in the `.spec.template.metadata.labels` path.

```
...output omitted...
spec:
  dataVolumeTemplates:
    ...output omitted...
  template:
    metadata:
      creationTimestamp: null
      labels:
        tier: front
        flavor.template.kubevirt.io/small: "true"
        kubevirt.io/domain: golden-web
        kubevirt.io/size: small
    ...output omitted...
```

- 7.3. Restart the web1 VM to apply the changes. Click **Actions > Restart** and wait for the VM to reach the **Running** status before proceeding.
- 7.4. Repeat this step to add the same label to the web2 VM.
8. Create a service named **front** with the **ClusterIP** type in the **review-cr3** project. Use the **tier: front** label for the selector. The service must listen on TCP port 80 and forward the traffic to the VM on port 80.
  - 8.1. From the OpenShift web console, go to **Networking > Services**.
  - 8.2. Click **Create Service**. Use the YAML editor to create the **front** service with the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: front
  namespace: review-cr3
spec:
  type: ClusterIP
  selector:
    tier: front
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

The `lab` command prepares the `~/D0316/labs/review-cr3/service.yaml` file as an example.

- 8.3. Click **Create**. The console displays the **Details** tab for the **front** service.
- 8.4. Confirm that the **front** service has two active endpoints. Click the **Pods** tab and confirm that list includes the **virt-launcher** pods for the **web1** and **web2** VMs.
9. Create a route to access the web application that is running inside the VMs by using the `https://front-review-cr3.apps.ocp4.example.com` URL. Add TLS edge termination to encrypt traffic between the client and the router.

- 9.1. From the web console, go to **Networking > Routes**. Click **Create Route** and complete the form by using the following information:

| Field           | Value         |
|-----------------|---------------|
| Name            | front         |
| Service         | front         |
| Target port     | 80 → 80 (TCP) |
| Secure Route    | Selected      |
| TLS Termination | Edge          |

Click **Create**.

- 9.2. From the command line, use the `curl` command to confirm that you can access the web application from outside the cluster.

```
[student@workstation ~]$ curl https://front-review-cr3.apps.ocp4.example.com
<!doctype html>
<html lang="en">
<head>
  <title>Test page web2</title>
</head>

<body>
  <h1>Test Page web2</h1>
  Welcome to web2.
</body>
</html>
```

- 9.3. Rerun the `curl` command several times to confirm that the service dispatches the requests between the two VMs.

```
[student@workstation ~]$ curl https://front-review-cr3.apps.ocp4.example.com/?
[1-3]
--curl--front-review-cr3.apps.ocp4.example.com/?1
<!doctype html>
<html lang="en">
<head>
  <title>Test page web2</title>
</head>

<body>
  <h1>Test Page web2</h1>
  Welcome to web2.
</body>
</html>

[2/3]: front-review-cr3.apps.ocp4.example.com/?2 --> <stdout>
--curl--front-review-cr3.apps.ocp4.example.com/?2
<!doctype html>
```

```
<html lang="en">
<head>
    <title>Test page web1</title>
</head>

<body>
    <h1>Test Page web1</h1>
    Welcome to web1.
</body>
</html>

[3/3]: front-review-cr3.apps.ocp4.example.com/?3 --> <stdout>
--curl--front-review-cr3.apps.ocp4.example.com/?3
<!doctype html>
<html lang="en">
<head>
    <title>Test page web2</title>
</head>

<body>
    <h1>Test Page web2</h1>
    Welcome to web2.
</body>
</html>
```

## Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade review-cr3
```

## Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish review-cr3
```

