For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Red Hat
Training and Certification

Student Workbook (ROLE)

OCP 4.10 DO316

Managing Virtual Machines with Red Hat OpenShift Virtualization

Edition 4

DO316-OCP4.10-en-4-20230623

Copyright ©2023 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Red Hat
Learning Community

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 comunity 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

Red Hat

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.

Copyright ©2023 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Managing Virtual Machines with Red Hat OpenShift Virtualization

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

OCP 4.10 DO316
Managing Virtual Machines with Red Hat OpenShift Virtualization
Edition 4 20230623
Publication date 20230623

Authors:	Chyna Sanders, Maria Fernanda Ordonez Casado, Austin Garrigus, Natalie Lind, Christopher Caillouet, Hervé Quatremain
Course Architect:	Fernando Lozano
DevOps Engineer:	Benjamin Chardi Marco
Editor:	Nicole Muller

© 2023 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2023 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] 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Document Conventions **ix**
Admonitions ... ix
Inclusive Language ... x

Introduction **xi**
Managing Virtual Machines with Red Hat OpenShift Virtualization ... xi
Orientation to the Classroom Environment ... xii
Performing Lab Exercises ... xx

1. Introducing Red Hat OpenShift Virtualization **1**
Describing OpenShift Virtualization ... 2
Quiz: Describing OpenShift Virtualization ... 6
Describing Underlying Kubernetes Architecture ... 8
Quiz: Describing Underlying Kubernetes Architecture ... 11
Deploying the OpenShift Virtualization Operator ... 13
Guided Exercise: Deploying the OpenShift Virtualization Operator ... 17
Summary ... 22

2. Running and Accessing Virtual Machines **23**
Describing Virtual Machine Resources ... 24
Quiz: Describing Virtual Machine Resources ... 29
Creating and Accessing Virtual Machines ... 32
Guided Exercise: Creating and Accessing Virtual Machines ... 37
Inspect and Monitor Virtual Machines ... 46
Guided Exercise: Inspect and Monitor Virtual Machines ... 51
Lab: Create, Configure and Monitor a Virtual Machine ... 58
Summary ... 77

3. Configuring Kubernetes Networking for Virtual Machines **79**
Describing Kubernetes Networking Objects ... 80
Guided Exercise: Describing Kubernetes Networking Objects ... 88
Configuring External Access to Virtual Machines ... 96
Guided Exercise: Configuring External Access to Virtual Machines ... 106
Lab: Configuring Kubernetes Networking for Virtual Machines ... 111
Summary ... 120

4. Connecting Virtual Machines to External Networks **121**
Introducing Multus ... 122
Quiz: Introducing Multus ... 127
Configuring Multihomed Nodes and Virtual Machines ... 131
Guided Exercise: Configuring Multihomed Nodes and Virtual Machines ... 140
Lab: Connecting Virtual Machines to External Networks ... 150
Summary ... 161

5. Configuring Kubernetes Storage for Virtual Machines **163**
Attaching Persistent Storage to Virtual Machines ... 164
Guided Exercise: Attaching Persistent Storage to Virtual Machines ... 171
Attaching and Accessing Disks on Virtual Machines ... 175
Guided Exercise: Attaching and Accessing Disks on Virtual Machines ... 181
Connecting Virtual Machines to External Storage ... 185
Guided Exercise: Connecting Virtual Machines to External Storage ... 192
Lab: Configuring Storage for Virtual Machines ... 197
Summary ... 205

6. Virtual Machine Template Management **207**
Virtual Machine Template Usage ... 208
Guided Exercise: Virtual Machine Template Usage ... 216
Create Custom Templates for Virtual Machines ... 220

Guided Exercise: Create Custom Templates for Virtual Machines 226
Lab: Virtual Machine Template Management 232
Summary 239

7. Advanced Virtual Machine Management **241**
Managing Virtual Machine Snapshots 242
Guided Exercise: Managing Virtual Machine Snapshots 248
Cloning Virtual Machines 253
Guided Exercise: Cloning Virtual Machines 259
Performing Virtual Machine Live Migrations 264
Guided Exercise: Performing Virtual Machine Live Migrations 269
Node Maintenance and OpenShift Virtualization Updates 275
Guided Exercise: Node Maintenance and OpenShift Virtualization Updates 281
Lab: Advanced Virtual Machine Management 287
Summary 293

8. Configuring Kubernetes High Availability for Virtual Machines **295**
Virtual Machine Load Balancing with Kubernetes Networking Resources 296
Guided Exercise: Virtual Machine Load Balancing with Kubernetes Networking Resources 299
Configure Health Probes for Virtual Machines 306
Guided Exercise: Configure Health Probes for Virtual Machines 313
Surviving Node Failure with Virtual Machines 321
Guided Exercise: Surviving Node Failure with Virtual Machines 327
Lab: Configuring Kubernetes High Availability for Virtual Machines 333
Summary 347

9. Comprehensive Review **349**
Comprehensive Review 350
Lab: Deploying OpenShift Virtualization and Configuring External Network 352
Lab: Using Templates to Deploy Virtual Machines 358
Lab: Cloning Virtual Machines and Configuring Load Balancing 364

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 required to create and manage virtual machines (VM) on OpenShift 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 using the same Kubernetes features that would also control usage and access to these resources for containers.
- Manage high availability (HA) of VMs 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 interested in moving virtualized workloads from traditional Hypervisors to OpenShift Virtualization.
- Kubernetes Administrators (Cluster Administrators, Clusters Engineers) interested in supporting containerized and virtualized workloads in the same OpenShift cluster.

Prerequisites

- DO180 and DO280 are suggested but optional prerequisites. DO316 introduces the essential Kubernetes concepts required to manage VMs in OpenShift.
- Linux skills are not required to manage OpenShift clusters and OpenShift Virtualization, but managing individual Linux VMs requires the Linux system administration skills covered in RH124 and RH134.

Orientation to the Classroom Environment

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

The `workstation` machine has a standard user account, `student` with the password `student`. 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`.

It is from the `workstation` machine that you type the `oc` commands to manage the OpenShift cluster, which comes preinstalled as part of your classroom environment.

It is also from the `workstation` machine that you run shell scripts and Ansible Playbooks required to complete the exercises for this course.

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

Note

The first time 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 using either the `oc` command or the web console without first running a `lab` command, then you might find that your cluster is not yet available. If that happens, then wait a few minutes and try again.

Log in on 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 redhat \
  https://api.ocp4.example.com:6443
```

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

Accessing the OpenShift Web Console

If you prefer to use the OpenShift web console, 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 log in credentials for either the `admin` or `developer` user.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The Classroom Environment

Every student gets a complete remote classroom environment. As part of that environment, every student gets a dedicated OpenShift cluster to perform 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 on the Student, Classroom, Server, and Cluster networks run Red Hat Enterprise Linux 8 (RHEL 8), except those machines that are nodes of the OpenShift cluster. These nodes run RHEL CoreOS.

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

Usually, the `lab` commands from exercises access these machines when there is a requirement to setup 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 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The systems called `master`*XX* and `worker`*XX* 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
`workstation.lab.example.com`	172.25.250.9	Graphical workstation used for system administration.
`classroom.example.com`	172.25.254.254	Router linking the Classroom Network to the Internet.
`bastion.lab.example.com`	172.25.250.254	Router linking the Student Network to the Classroom Network.
`utility.lab.example.com`	172.25.250.253	Router linking the Student Network to the Cluster Network and also storage server.
`server.srv.example.com`	192.168.51.40	Server and router linking the Cluster Network to a private DHCP network.
`ceph.ocp4.example.com`	192.168.50.30	Server with a Red Hat Storage Ceph pre-installed cluster.
`master01.ocp4.example.com`	192.168.50.10	Control plane node
`master02.ocp4.example.com`	192.168.50.11	Control plane node
`master03.ocp4.example.com`	192.168.50.12	Control plane node
`worker01.ocp4.example.com`	172.25.250.13	Compute node
`worker02.ocp4.example.com`	172.25.250.14	Compute node

Dependencies on Internet Services

Red Hat OpenShift Container Platform 4 requires access to two container registries to download container images for operators, S2I builders, and other cluster services. These registries are:

- `registry.redhat.io`
- `quay.io`

If either registry is unavailable when starting the classroom environment, then the OpenShift cluster might not start or could enter a degraded state.

If these container registries experience an outage while the classroom environment is up and running, then it might not be possible to complete exercises until the outage is resolved.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The Dedicated OpenShift Cluster

The Red Hat OpenShift Container Platform 4 cluster inside the classroom environment is preinstalled using the Pre-existing Infrastructure installation method. All nodes are treated as bare metal servers, even though they are actually virtual machines in an OpenStack cluster.

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

Your OpenShift cluster is in the state left by running the OpenShift installer with default configurations, excepting a few day-2 customizations:

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

The section *Troubleshooting Access to your OpenShift Cluster* provides information about 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 anymore 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 was used to run 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` folder of the `utility` machine.

Logging in to the `utility` server is rarely required to perform exercises. If it looks like your OpenShift cluster is taking too long to start, or is in a degraded state, then you can log in on 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
```

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

This allows you to 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:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

In the preceding example, replace `master01` with the name of the desired 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 that was stopped after installation and will be stopped and restarted several times before you finish this course. This scenario requires special handling that would not be required by a production cluster.

The control plane and compute nodes in an OpenShift cluster frequently 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 they 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 is required to 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 you might find that your cluster is not 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 required CSRs requests, and the system service did not wait long enough. It is also possible that some OpenShift cluster nodes did not wait long enough for their CSRs to be approved, issuing new CSRs that superseded previous ones.

If these issues arise, then you will notice that your cluster is taking too long to come up, and your `oc login` or `lab` commands fail. To resolve the problem, you can log in on 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 in case your cluster nodes issue new CSRs that supersede the ones it approved.

After you approve, or the system service in the `utility` machine approves all CSRs, then OpenShift must restart some cluster operators. It takes a few 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 that waits until your OpenShift cluster is ready to accept authentication and API requests from remote clients.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Although unlikely, if neither the service on the `utility` machine nor running the `sign.sh` and `wait.sh` scripts make 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 include:

- `oc get node` to verify if all of your cluster nodes are ready.
- `oc get csr` to verify if your cluster still has any pending, unapproved CSRs.
- `oc get co` to verify if any of your cluster operators are unavailable, in a degraded state, or progressing through configuration and rolling out pods.

If these fail, you can try destroying and recreating 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]. 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.

Figure 0.2: An example course Lab Environment management page

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Machine States

Virtual machine state	Description
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 that it was in before shutdown. The disk state is preserved.

Classroom Actions

Button or action	Description
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. **All saved work on those systems' disks is lost.**
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

Button or action	Description
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 `workstation` virtual machine only, and from there, use `ssh` 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. **All saved work on that system's disks is lost.**

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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 may 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-create       ha-node              review-cr2
accessing-monitor      ha-probe             review-cr3
accessing-review       ha-review            storage-external
advanced-clone         multihomed-nmstate   storage-intro
advanced-maintenance   multihomed-review    storage-multiple
advanced-migrate       network-external     template-intro
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
advanced-review        network-services     template-create
advanced-snapshot      network-review       template-review
ha-loadbalance         review-cr1           virtualization-deploy
```

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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 1

Introducing Red Hat OpenShift Virtualization

Goal

Distinguish Red Hat OpenShift Virtualization from container technologies and traditional virtual machine technologies. Describe the features and use cases of OpenShift Virtualization.

Objectives

- Describe the features and use cases of OpenShift Virtualization.
- Describe Kubernetes high-level architecture and operators.
- Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift environment.

Sections

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Describing OpenShift Virtualization

Objectives

- Describe the features and use cases of OpenShift Virtualization.

Introduction to OpenShift Virtualization

Red Hat OpenShift Container Platform (RHOCP) is a set of modular components and services built on top of Kubernetes container infrastructure. RHOCP is a software product derived from an open source project called Kubernetes.

Both RHOCP and Kubernetes are container orchestration software, but RHOCP is packaged as a downstream enterprise open source platform.

Features of Red Hat OpenShift Container Platform

RHOCP adds some features to the cluster:

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.

Routes
Exposes services to the outside world easily.

Metrics and logging
Includes a built-in and self-analyzing metrics service and aggregated logging.

Unified UI
Provides unified tools and an integrated UI to manage the different capabilities.

Common infrastructure
Supports deployment and management of virtual machines (VMs) alongside containerized deployments by sharing common infrastructure and robust management tools, including the console and API.

Red Hat OpenShift Virtualization

Red Hat OpenShift Virtualization is a Kubernetes operator on RHOCP, based on the Kubernetes virtualization add-on, *KubeVirt*, which allows the management of VM workloads alongside container workloads.

KubeVirt delivers container-native virtualization by leveraging Linux *Kernel-based Virtual Machine (KVM)*, the Linux Kernel hypervisor, within a Kubernetes container. KubeVirt includes services similar to traditional virtualization platforms, providing 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 open source upstream project.

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.

The following figure shows the architecture of the OpenShift Virtualization operator:

Figure 1.1: Architecture of the OpenShift Virtualization operator

Benefits of OpenShift Virtualization

The following list describes some of the benefits of managing VMs through OpenShift Virtualization.

- Provides new *custom resource definitions* (CRDs) and controllers to extend Kubernetes functionality by enabling virtualization-related tasks.
- Supports creating and managing Linux and Windows VMs by using *custom resources* (CRs), the web console, the command line, and YAML definition templates so that VMs can be easily and quickly customized.
- Allows creating VMs by cloning a *persistent volume claim* (PVC) pointing to an existing VM's *persistent volume* (PV).
- Supports importing and managing VMs created outside the OpenShift cluster.
- Permits moving a *Virtual Machine Instance* (VMI) and its resources between cluster nodes with the help of live migration strategies, eliminating the need to stop VMs before moving them.

Features of OpenShift Virtualization

You can use OpenShift Virtualization to administer both VMs and all the resources that are required to manage them:

Storage management
: Manage storage by assigning a PV that contains your application files to a VM.

Network management
: Attach network interfaces to VMs to expose your application connections and services.

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.

Workloads management
: Manage virtualized workloads in the same unified platform.

Side by side and consistent management
: Perform side by side management of containers and VMs, allowing a flexible, agile working environment that centralizes infrastructure management.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Pipelines for VMs (CI/CD)
Use one pipeline to test applications in both containers and VMs.

Management with Kubernetes resources
Manage virtual resources, either with the resources that RHOCP offers or with Kubernetes resources by using the command line.

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

Virtualization and cloud computing are often confused because both concepts relate to the management of virtualized resources. The following table defines virtualization, *Infrastructure-as-a-Service* (IaaS), and hypervisor technology:

Virtualization	IaaS	Hypervisor
This technology helps you to create virtualized resources from physical hardware, such as VMs.	Infrastructure-as-a-Service (IaaS) is a form of cloud computing that provides an IT infrastructure to users through a cloud.	Hypervisor is virtualization software that helps you to create and manage VMs.

Virtualization Components

The following virtual components are provided by KubeVirt:

`virt-api`
A cluster-level component that provides an HTTP RESTful entry point to manage VM and VM-related workflows within the cluster. It is responsible for updating virtualization CRDS and is responsible for the defaulting and validation of VMI CRDs.

`virt-controller`
A cluster-level component responsible for cluster-wide virtualization functionality and managing the lifecycle of pods associated with VMIs. The operator creates the pod where the VM objects execute.

`virt-handler`
A host-level DaemonSet resource running on each node that monitors for changes to a VM object and performs 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 VM CRD object is passed to the `virt-launcher` component by the `virt-handler` 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 that is used by the `virt-launcher` component to manage the lifecycle of a VMI process.

OpenShift Virtualization Operators and Components

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

KubeVirt
KubeVirt is an open source technology that makes it possible to run VMs in RHOCP with the OpenShift Virtualization operator. KubeVirt runs and manages traditional VMs as if they were containers.

In contrast to Kubevirt, Kata Containers is an *Open Container Initiative* (OCI) runtime used to build lightweight VMs and put containers inside, allowing each container to run on its own kernel. Kata Containers runs application containers within VMs to increase their isolation from a security perspective.

Hostpath Provisioner Operator (HPP)
The HPP operator is a local storage provisioner for virtual machines. Local storage or volumes mounted on nodes can be provisioned to 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 built into the Linux kernel that allows a host machine to run multiple, isolated virtual environments known as guest VMs.

OpenShift Virtualization supports management and orchestration of VMs with tooling and methodology similar to that used for container management. These shared tools and methods enable administrators and developers to centralize infrastructure management.

References
For more information, refer to the *OpenShift Virtualization* documentation at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Quiz

Describing OpenShift Virtualization

Choose the correct answers to the following questions:

1. Red Hat OpenShift Virtualization provides which of the following features for your OpenShift cluster?

a. Role-based access control features 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

2. Which of the following three tasks does the OpenShift Virtualization provide for an OpenShift cluster through CRDs? (Choose three.)

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

3. On which open source upstream project is OpenShift Virtualization based?

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

4. With OpenShift Virtualization, you can deploy either Linux or Windows VMs.

a. True
b. False

5. With OpenShift Virtualization, you must stop a running VM before moving it between cluster nodes.

a. True
b. False

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Describing OpenShift Virtualization

Choose the correct answers to the following questions:

1. **Red Hat OpenShift Virtualization provides which of the following features for your OpenShift cluster?**
   a. Role-based access control features 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

2. **Which of the following three tasks does the OpenShift Virtualization provide for an OpenShift cluster through CRDs? (Choose three.)**
   a. Provides encryption and security audits for the OpenShift cluster
   b. Creates and manages Linux and Windows virtual machines
   c. Manages storage allocation for the OpenShift cluster
   d. Supports live migration of deployed virtual machines between cluster nodes
   e. Manages a VMware cluster using open source technology
   f. Supports connectivity to virtual machines by using common protocols, such as RDP and SSH.

3. **On which open source upstream project is OpenShift Virtualization based?**
   a. Hyper V
   b. KubeVirt
   c. oVirt
   d. KVM

4. **With OpenShift Virtualization, you can deploy either Linux or Windows VMs.**
   a. True
   b. False

5. **With OpenShift Virtualization, you must stop a running VM before moving it between cluster nodes.**
   a. True
   b. False

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Describing Underlying Kubernetes Architecture

Objectives

- Describe Kubernetes high-level architecture and operators.

Kubernetes Architecture

Kubernetes is an orchestration service that simplifies the deployment, management, and scaling of containerized applications. Kubernetes manages complex pools of resources, such as CPU, RAM, storage, and networking, and provides high uptime and fault tolerance for containerized application deployments.

Kubernetes uses a declarative management approach to manage the cluster resources and deploy applications. Most system administrators are accustomed to imperative architectures, where you perform actions that directly change the system state, such as starting and stopping containers on a server. In a declarative architecture, you change the state of the system, and the system updates itself to comply with the new state.

This course does not focus on running containers, but managing virtual machines in this environment requires basic Kubernetes concepts.

Kubernetes Features

Kubernetes clusters provide a modern container platform that addresses the concerns and challenges of running applications at scale. No matter the deployment size, Kubernetes implementations deliver a robust infrastructure and ease of management.

Kubernetes delivers the following features for a cluster:

- Service discovery
- Load balancing
- Horizontal scaling
- Self-healing
- Automated rollout
- Configuration management
- Secrets management

Kubernetes Terminology

The following list defines some common Kubernetes terms:

Operator
: A cluster component that simplifies the management of another application or function

Resource
: Any of the configurable and consumable components managed by the OpenShift cluster

Control Plane
: The cluster layer, responsible for container lifecycle management through its provided API

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Data Plane
: The cluster layer, responsible for providing the resources required to run containers, such as storage, networking, CPU, and memory

Pod
: A group of running containers within the cluster that provides a single application, service, or function

Container
: A small executable image that defines the libraries and dependencies for an application

Kubernetes Operators

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

Kubernetes OperatorHub

Operators for Kubernetes are shared through the open source community OperatorHub. Red Hat, in collaboration with AWS, Google Cloud, and Microsoft, launched the OperatorHub, which is accessible at https://operatorhub.io. The 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 created by using the Operator Framework are shared with the open source community through the OperatorHub.

Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform (RHOCP) is a set of modular components and services built on top of Kubernetes container infrastructure. RHOCP 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 RHOCP provides enterprise scaling and reliability for modern deployments.

Red Hat Marketplace

Red Hat provides a catalog of certified enterprise operators for installation on OpenShift clusters through the Red Hat Marketplace. You can peruse, test, and adopt operators from this curated collection to provide the features that your cluster requires.

As you evolve your OpenShift cluster to provide more functionality and integration with other business systems, such as storage, the operators available 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 diverse functionalities to the cluster that aid the business in the transition from virtual machines to containers.

As container adoption grows, businesses exploring container technologies often require the ability to continue management and development of existing or new virtual machine deployments. OpenShift clusters with the OpenShift Virtualization operator installed handle both virtualized and containerized applications to aid your container adoption journey.

Virtualization Meets Containerization

This course explores the features of OpenShift Virtualization that provide the management and orchestration layer for virtual machine workloads within an OpenShift cluster. With OpenShift Virtualization, you can deliver the resiliency and ease of managing both VM-based and containerized applications from a single interface and infrastructure. Clusters with the OpenShift Virtualization operator can run your virtual machines' workloads alongside your containerized workloads, thus eliminating the need for a separate and potentially expensive virtualization management solution.

This capability creates a valuable opportunity for businesses that are undergoing container adoption while still maintaining a traditional deployment of virtual machines.

References

For more information about Red Hat OpenShift Virtualization, refer to the *Virtualization* section of the *OpenShift Container Platform 4.10 Documentation* at https://docs.openshift.com/container-platform/4.10/virt/about-virt.html
.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Quiz

Describing Underlying Kubernetes Architecture

Match the following items to their counterparts in the table.

A cluster component that simplifies the management of another application or function.

A group of running containers within the cluster that provides a single application, service, or function.

A small executable image that defines the libraries and dependencies for an application.

Any of the configurable and consumable components managed by the Red Hat OpenShift cluster.

The cluster layer responsible for container lifecycle management through its provided API.

The cluster layer that provides the resources required to run containers, such as storage, networking, CPU, and memory.

Terminology	Definition
Operator	
Resource	
Control Plane	
Data Plane	
Pod	
Container	

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Describing Underlying Kubernetes Architecture

Match the following items to their counterparts in the table.

Terminology	Definition
Operator	A cluster component that simplifies the management of another application or function.
Resource	Any of the configurable and consumable components managed by the Red Hat OpenShift cluster.
Control Plane	The cluster layer responsible for container lifecycle management through its provided API.
Data Plane	The cluster layer that provides the resources required to run containers, such as storage, networking, CPU, and memory.
Pod	A group of running containers within the cluster that provides a single application, service, or function.
Container	A small executable image that defines the libraries and dependencies for an application.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Deploying the OpenShift Virtualization Operator

Objectives

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

Installing the Red Hat OpenShift Virtualization Operator

Red Hat OpenShift Container Platform (RHOCP) must meet specific minimum requirements to support different features of *Red Hat OpenShift Virtualization*.

- You must install the cluster on-premise bare metal with Red Hat Enterprise Linux CoreOS workers.
- All CPUs must be supported by Red Hat Enterprise Linux 8, 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.

Red Hat recommends not exceeding tested maximums to maintain performance. Review the tested maximums for Red Hat OpenShift Virtualization 4.10 in the article *Supported Limits for OpenShift Virtualization 4.x* at https://access.redhat.com/articles/6571671.

Installation

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

Click **Install** to open the **Install Operator** page. Review the installation options and click **Install** to deploy the operator to the `openshift-cnv` namespace.

OpenShift Virtualization requires that the KubeVirt *HyperConverged Operator (HCO)* is installed in order to create virtual machines. The HCO deploys and manages OpenShift Virtualization and its components, such as the `virt-controller` cluster-level component and the `virt-handler` host-level Daemonset. The HCO also creates *Custom Resources (CR)* to deploy and manage supplemental operators for OpenShift Virtualization. These CRs include the *Containerized Data Importer* operator, the *Scheduling, Scale, and Performance* operator, *Cluster Network Addons* operator, and the *Node Maintenance* operator.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

OpenShift Virtualization operators are discussed in greater detail throughout the course.

Installation From the Command Line

To use the command-line installation method, create a YAML file defining the components needed for the subscription and apply it by using the `oc apply -f file name.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
---
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.v4.10.0
  channel: "stable"
```

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

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
```

The Hostpath Provisioner and Local Storage

By using the *HostPathProvisioner (HPP)* custom resource, you can create virtual machines that use local node storage. In an environment without available network storage, you can deploy the hostpath provisioner to provision VMs on a node's local storage. Additionally, you can use local storage to troubleshoot performance issues for a virtual machine if you suspect that the root cause is the network storage.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

With the hostpath provisioner, a virtual machine is backed by a local directory on the host node. The hostpath provisioner must be scheduled on nodes that are also scheduled for virtualization pods.

> **Note**
> For information about configuring the hostpath provisioner, refer to *Configuring local storage for virtual machines* at https://docs.openshift.com/container-platform/4.10/virt/virtual_machines/virtual_disks/virt-configuring-local-storage-for-vms.html

Deployment Strategy for OpenShift Virtualization Supplemental Operators and Workloads

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

Node Selector

A `nodeSelector` object defines node placement by using labels. You can specify multiple labels, but each specified label must be present for the node to be selected. Specify a node selector by editing the YAML of an object during or after deployment to include the key-value pair of one of the following selectors.

- `spec.config.nodePlacement.nodeSelector`: Defines the node selector for operators, such as the HCO, that are deployed by the *Operator Lifecycle Manager (OLM)* during operator installation. To use the selector, you must edit the `Subscription` object for the `hco-operatorhub` during the OpenShift Virtualization operator installation.

- `spec.infra.nodePlacement.nodeSelector`: Defines the nodes for supplemental components and operators, such as the Containerized Data Importer operator, that are deployed by the HCO, and can be used during or after the OpenShift Virtualization operator installation.

- `spec.workloads.nodePlacement.nodeSelector`: Defines the nodes that the HCO can use to deploy workloads, and can be used during or after the OpenShift Virtualization operator installation.

Affinity

Affinity rules specify node placement with more flexibility than the node selector. When determining scheduling, you can use affinity to preferentially select or avoid specific nodes.

To establish affinity rules, adjust `Affinity` under `Node Placement` during deployment, or edit the YAML of an object to include an entry under `spec.affinity.nodeAffinity`.

> **Note**
> For more information about configuring node affinity, refer to *Controlling pod placement on nodes using node affinity rules* at https://docs.openshift.com/container-platform/4.10/nodes/scheduling/nodes-scheduler-node-affinity.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Toleration

Tolerations define node placement with node taints. Taints and tolerations work similarly to labels and node selectors, but include a specified effect. The effect determines whether the scheduler assigns pods with a toleration to a node with a matching taint and what to do with existing pods on a node that do not match. Specify toleration by adjusting the `Toleration` under `Node Placement` during deployment or edit the YAML of an object to include an entry under `spec.config.tolerations`.

Note

For more information about configuring tolerations and taints, refer to *Controlling pod placement using node taints* at https://docs.openshift.com/container-platform/4.10/nodes/scheduling/nodes-scheduler-taints-tolerations.html

Virtual Machine Creation

You can create virtual machines by using the Quick Start guided tour, the virtual machine deployment wizard, or by applying a YAML file in the wizard or from the command line.

To begin the Quick Start guided tour, select **Help** > **Quick Starts**. The **Help** menu is represented by a ? icon. You can then select either **Creating a Red Hat Enterprise Linux virtual machine** or **Creating a Microsoft Windows 10 virtual machine**. 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 by navigating to **Workloads** > **Virtualization**.

References

For more information about installing via the web console, refer to *Installing OpenShift Virtualization using the web console* at https://docs.openshift.com/container-platform/4.10/virt/install/installing-virt-web.html

For more information about installing via the command line, refer to *Installing OpenShift Virtualization using the CLI* at https://docs.openshift.com/container-platform/4.10/virt/install/installing-virt-cli.html

For more information about the hostpath provisioner, refer to the *Configuring local storage for virtual machines* guide at https://docs.openshift.com/container-platform/4.10/virt/virtual_machines/virtual_disks/virt-configuring-local-storage-for-vms.html

For more information about specifying nodes, refer to *Specifying nodes for OpenShift Virtualization components* at https://docs.openshift.com/container-platform/4.10/virt/install/virt-specifying-nodes-for-virtualization-components.html

For more information about creating virtual machines, refer to the *Creating virtual machines* guide at https://docs.openshift.com/container-platform/4.10/virt/virtual_machines/virt-create-vms.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Deploying the OpenShift Virtualization Operator

Deploy the OpenShift Virtualization operator in an existing Red Hat OpenShift environment and create a VM to prove the operator's functionality.

Outcomes

- Deploy the OpenShift Virtualization operator in an existing OpenShift instance by using the web console.
- Review the installed operators and confirm that the OpenShift Virtualization operator is installed.
- Create a Fedora virtual machine from a template.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

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

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

Instructions

1. Open a web browser and log in to https://console-openshift-console.apps.ocp4.example.com from the `workstation` machine.

   1.1. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

2. From the Red Hat OpenShift web console left panel, navigate to **Operators** > **OperatorHub**. If you do not see the left panel, click the three vertical bars at the top left of the web console.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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:

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

Click **Install** to initiate the operator installation.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 view the options for the HyperConverged instance.

Note

You can also create a HyperCoverged 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** to view the options for the HyperConverged instance.

5.2. Accept the default options and click **Create** in the lower left of the window.

5.3. Wait for the operation to finish. The `kubevirt-hyperconverged` HyperConverged instance displays a **Status** of **Conditions: ReconcileComplete, Available, Upgradable** when the operation completes.

6. From the OpenShift web console left panel, navigate to **Operators** > **Installed Operators** to confirm that the OpenShift Virtualization installation succeeded.

7. Create a simple VM to confirm the functionality of OpenShift Virtualization.

7.1. From the OpenShift web console left panel, navigate to **Virtualization** > **VirtualMachines**.

7.2. From the **Project** menu, click **Create Project**. Name the project, `test-vms`, and click **Create**.

7.3. Click **Create virtual machine**. Find and select the **Fedora 34+ VM** template. Click **Next** to use this template.

> **Note**
> If a message about the support for this template appears, click **Continue** to proceed.

7.4. Select **URL (Creates PVC)** from **Boot source** > **Boot source type**. In the **Import URL** field, specify the `fedora34.qcow2` image located at http://utility.lab.example.com:8080/openshift4/images/fedora34.qcow2.

7.5. Leave the rest of the boot source settings at the default values and click **Next**.

7.6. Provide a name for your virtual machine or leave the autogenerated name.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

7.7. Leave the remaining settings at the default values and click **Create virtual machine**. Click **See virtual machine details** to navigate to the new VM's **Overview** tab. Wait for the import to complete and for the machine to power on automatically.

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- Kubernetes uses declarative management of cluster resources to simplify the deployment, management, and scaling of containerized applications.
- *Red Hat OpenShift Container Platform (RHOCP)* is a set of modular components built on top of a Kubernetes container infrastructure that provides a production platform through remote management.
- RHOCP offers the *Red Hat OpenShift Virtualization operator* through the Red Hat Marketplace.
- Virtual machines created outside the OpenShift cluster can be imported and then managed with the OpenShift Virtualization operator.
- OpenShift Virtualization provides new custom resource definitions and controllers to extend Kubernetes functionality.
- RHOCP must meet specific minimum requirements to support different features of OpenShift Virtualization:
  - The cluster must be installed on premise, on bare metal with Red Hat Enterprise Linux CoreOS workers.
  - All CPUs must be supported by Red Hat Enterprise Linux 8, support Intel 64 or AMD64 CPU extensions, have Intel VT or AMD-V hardware virtualization enabled, and have the no-execute (NX) flag enabled.
- The OpenShift Virtualization operator requires that the HyperConverged Operator is installed in order to create virtual machines.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 2

Running and Accessing Virtual Machines

Goal

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

Objectives

- Describe and compare virtual machine instance components.
- Provision and access a new virtual machine.
- Monitor VM resources and inspect a VM's configuration.

Sections

- Describing Virtual Machine Resources (and Matching Quiz)
- Creating and Accessing Virtual Machines (and Guided Exercise)
- Inspect and Monitor Virtual Machines (and Guided Exercise)

Lab

- Create, Configure and Monitor a Virtual Machine

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Describing Virtual Machine Resources

Objectives

- Describe and compare virtual machine instance components.

Workload Controllers

Red Hat OpenShift Container Platform (RHOCP) offers a set of resources to help you run applications inside your cluster. Containers are the most basic units with which you can interact in an OpenShift cluster.

The most common resources are *pods*, which are composed of one or more *containers*. These resources contain your running application instances, and you can interact with them by using pod management tools.

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

The following topics explain the most common workload resources.

Deployments and Deployment Configurations

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

Similar to deployments, a `DeploymentConfig` object is a resource that contains a template that defines a set of objects needed to create and run your application workload.

However, unlike deployments, deployment configurations interact with `ReplicationController` object to manage your application's pods.

Both of these workload resources enable declarative application management, but they use different resources for replication.

Replica Sets

A `ReplicaSet` resource, used independently or by a deployment, ensures that the specified quantity of pod replicas are available in a healthy state. Replica sets use set-based selectors, filtered by key values, to identify pods in the cluster. Replica sets create, update, or delete pods automatically to maintain the desired state specified by the deployment.

Replication Controllers

A `ReplicationController` object, similar to the `ReplicaSet` object, is a resource that ensures a certain number of pod replicas are always available for use by a deployment configuration or independently.

Replication controllers instantiate additional pods if a pod enters a failed state or exits. Replication controllers also delete excessive pods to match the number declared in the deployment

configuration. Unlike a replica set, a replication controller uses equality-based selectors to identify pods. Pods are identified by their labels and cannot be filtered further.

> **Note**
> The `deployment` and `ReplicaSet` resources are considered successors to `DeploymentConfig` and `ReplicationController` objects.

Daemon Sets

A `DaemonSet` resource ensures that a pod runs on every selected node. Unlike `ReplicaSet` and `ReplicationController` resources, a `DaemonSet` resource is not bound by a specific number of replicas.

When a node matching the selector is added to the cluster, a pod is automatically created on the node. If a `DaemonSet` resource is deleted, the associated pods will automatically be cleaned up across the cluster.

Jobs

You can use a `job` object to execute or automate a one-off task, such as a report generation or a file clean-up. A job creates and executes one or more pods to run the task. A job's **parallelism parameter** specifies the number of pods to run concurrently during the task execution. A job runs until the number of pod replicas specified in the `completions` parameter successfully terminate.

Stateful Sets

A Stateful Set is a set of pods executing stateful applications with consistent, unique identities (network and storage). This `StatefulSet` workload resource is useful when you want to point to the same storage identity to maintain the application's persistent data.

Difference Between Workload Controllers, VMs, and VMIs

In general, a *virtual machine* (VM) is a virtual environment that simulates the resources of a physical machine, including CPU, memory, network interfaces, and storage. VMs are typically used to maintain application statefulness and persistent data, similar to the `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 *virtual machine instance* (VMI), and it is executed and managed by a container located inside a pod.

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 the VMI's pod. If a VMI is deleted, another instance is generated based on the VM object configuration.

When a VM is created and started, a `virt-controller` pod signals a `virt-handler` pod on a cluster node to create a new `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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.1: Execution level of VMs and containers

Components of an OpenShift Virtual Machine

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

Virtual Machine Consoles

Red Hat OpenShift Virtualization provides three consoles that you can use 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 upon 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)* console provides a graphical console through the RDP for Windows-based VMIs. The *Quick EMUlator* (QEMU) guest agent must be installed and running on the VMI to use the RDP.

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 *Accessing virtual machine consoles* section at https://docs.openshift.com/container-platform/4.10/virt/virtual_machines/virt-accessing-vm-consoles.html#virt-vm-rdp-console-web_virt-accessing-vm-consoles.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 the VMI an IP address 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 allow other pods in the cluster to communicate with the VMI. By default, as with other pods on the pod network, the VMI is only accessible within the cluster.

Multus

You can connect a VM to multiple interfaces and to external networks with the *Container Networking Interface (CNI)* plug-in, Multus. To connect to an external network, you must create a `linux-bridge` network attachment definition that exposes the layer-2 device to a specific namespace.

Single Root I/O Virtualization

A VMI can also use a *Single Root I/O Virtualization* (SR-IOV) device that allows a VMI to connect to a virtual function network for high performance. A SR-IOV device can be attached as a secondary network on the VMI. The SR-IOV standard allows multiple VMIs to share a single network device.

Figure 2.2: Network interface options

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Note
VM networking is explained in greater detail elsewhere in this course.

Persistent Volumes and Persistent Volume Claims

A *Persistent Volume* (PV) is a storage resource provisioned by the *PersistentVolume* API object and is a representation of existing storage within the cluster. You can attach a PV to a VMI by defining a *Persistent Volume Claim* (PVC), which represents a request for a specific storage resource.

Note
Persistent volumes and other storage management concepts are discussed elsewhere in this course.

VMI Execution Pods

A `virt-launcher` pod is created by the `virt-handler` daemon set when a VM object is started. The `virt-launcher` pod manages and executes the VMI with a `libvirtd` container instance.

When the VMI is provisioned, the `virt-launcher` pod routes IPv4 traffic to the *Dynamic Host Configuration Protocol* (DHCP) address of the VMI. This routing makes it possible to also connect to a VMI with a port-forwarding connection.

Standalone VMIs

When you create and start a VMI independently, such as by using automation, scripts, or the command line, it is considered as a *standalone* VMI. Standalone VMIs can be managed by using the web console or the command line with the `oc` command. To list all the VMIs in the cluster, including standalone VMIs, use the `oc get vmis` command.

```
[user@host ~]$ oc get vmis
```

Note
The `oc` commands for VM and VMI management are explained in further detail throughout this course.

References
For more information, refer to the *Red Hat OpenShift Virtualization* documentation at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

Quiz

Describing Virtual Machine Resources

Match the following items to their counterparts in the table.

A deployed set of defined resources, including CPU, RAM, and storage, that provide a software-based computer system

A resource containing a template that defines the set of objects required to create and run your application workload

A resource that uses pod templates to describe the desired state of your application and its components

A resource used independently or by a DeploymentConfig object to ensure a pod replica runs on every selected node

A resource used independently or by a DeploymentConfig object to ensure that a certain number of pod replicas are available and does not require a selector

A resource used independently or by a deployment to ensure that a specified quantity of pod replicas are available and requires a selector

A software-based computer that simulates the entire environment of a hardware-based physical computer, including an operating system and applications

Provides communication with a virtual machine for configuration and management of the instance through VNC, serial, or RDP connections

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Component	Function
Virtual Machine (VM)	
Virtual Machine Instance (VMI)	
Deployment	
DeploymentConfig	
Virtual Machine Console	
ReplicaSet	
ReplicationController	
DaemonSet	

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Describing Virtual Machine Resources

Match the following items to their counterparts in the table.

Component	Function
Virtual Machine (VM)	A software-based computer that simulates the entire environment of a hardware-based physical computer, including an operating system and applications
Virtual Machine Instance (VMI)	A deployed set of defined resources, including CPU, RAM, and storage, that provide a software-based computer system
Deployment	A resource that uses pod templates to describe the desired state of your application and its components
DeploymentConfig	A resource containing a template that defines the set of objects required to create and run your application workload
Virtual Machine Console	Provides communication with a virtual machine for configuration and management of the instance through VNC, serial, or RDP connections
ReplicaSet	A resource used independently or by a deployment to ensure that a specified quantity of pod replicas are available and requires a selector
ReplicationController	A resource used independently or by a DeploymentConfig object to ensure that a certain number of pod replicas are available and does not require a selector
DaemonSet	A resource used independently or by a DeploymentConfig object to ensure a pod replica runs on every selected node

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Creating and Accessing Virtual Machines

Objectives

- Provision and access a new virtual machine.

Virtual Machine Creation

You can create a Virtual Machine (VM) from a template by using the **Virtual Machine Creation Wizard** in the Red Hat OpenShift web console or by using YAML definitions through the web console or the command-line client.

Create a VM by Using the Wizard

The fastest and easiest way to create VMs from templates is by using the wizard in the OpenShift web console. Default templates include Red Hat Enterprise Linux, CentOS, Fedora, and Microsoft Windows Server. When creating a VM with the wizard, you must first select an operating system template. When using a default template, you must define a boot source. You can define additional storage settings, including the storage class, access mode, and volume mode, by using the **Advanced Storage Settings** in the web console. After you configure the storage settings, you then select configuration options for the VM.

Project
: The namespace where the VM resides.

Virtual Machine Name
: The name of the VM to use in your Red Hat OpenShift cluster.

Flavor
: Predefined CPU and RAM allocations based upon the operating system defined in template.

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. Additional configuration is needed to enable SSH access from outside of the cluster.

Start this VM after creation
: Determines whether the VM should be powered on after provisioning or if it should be left powered off.

Click **Customize Virtual Machine** to edit the VM template further before continuing.

Virtual Machine Templates

The default templates are provided by Red Hat. These templates include settings to create generic systems with networking, users, and storage preconfigured. Default templates create a basic installation with generally useful settings that suit a variety of needs. You can modify these templates to create new templates with more specific settings. A modified template can include a specific boot source, additional storage, additional network interfaces, or other configuration differences.

You can create entirely new templates by using the *Create Virtual Machine Template* wizard or with a YAML definition. New templates start with nothing preconfigured. You can easily create more specialized templates this way.

You can only create templates for the existing default operating systems, but you can use other operating systems, such as Ubuntu or openSUSE, by installing the *KubeVirt common-templates* package in your OpenShift cluster.

Note

For more information about installing KubeVirt `common-templates`, see https://cloud.redhat.com/blog/working-with-operating-system-templates-in-openshift-virtualization

Base Images

A base image is a source used to instantiate a VM. This base can be a cloud image or an installation disk image. Cloud images can be in either RAW or QCOW2 format and can be compressed with `gz` or `xz` compression. Operating system providers that create images in these formats generally release them alongside the general ISO-based installation disk image. You can also use base operating system images from container registries, such as Quay.io. Default templates include a prompt with an example of where to obtain an image for each source. You can also use an existing *Persistent Volume Claim (PVC)* as a base image. This creates a clone of the PVC to use with the new VM. Cloning existing PVCs is a quick way to create copies of an existing VM.

YAML Definition

In addition to using a wizard for both VM and VM template creation, you can also use YAML definition. You can define YAML from either the web console or the command line with a text editor. If using the web console, an editable default YAML configuration is provided. The YAML file includes all of the settings you find in the wizard 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 command line to edit a 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 changes are applied when the editor's buffer is written. If you wish to copy an existing YAML configuration, you can write the configuration to a file for editing with the `oc get` command. Then, you can create an object from the file with the `oc create -f` command.

Components of Virtual Machine Creation

Red Hat OpenShift utilizes several components to provide virtualization. The components are structured in a hierarchy where higher components orchestrate lower components.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.3: Components of virtual machine creation

virt-controller
The `virt-controller` operator monitors for new VM objects. When a new VM object is detected, the `virt-controller` operator creates a pod for the VM and then assigns the pod to a node.

virt-handler
The `virt-handler` daemon set runs on each node to execute any necessary actions to meet a VM's object defined state and to monitor for changes to a VM object. When a change is detected, virt-handler creates a domain within the `libvirtd` instance of a VM's pod. If a VM object is deleted, virt-handler turns off the domain within the VM's pod.

virt-launcher
The `virt-launcher` container runs within each VM's pod and instantiates the VM using a local `libvirtd` instance.

libvirtd
The `libvirtd` instance provides a low-level virtualization architecture and interface with the kernel to manage the lifecycle of the VM process.

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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

- The **Overview** tab contains a log of cluster-level VM events, general details about the VM, resource utilization, and status information.
- The **Details** tab contains detailed information about the components of the VM.
- The **YAML** tab provides an interface to edit the VM's YAML configuration.
- The **Environment** tab enables you to map configuration maps, secrets, or service accounts to a disk. The disk is then mounted within VM.
- The **Events** tab contains a detailed and customizable event log.
- The **Console** tab provides a serial and *Virtual Network Computing(VNC)* console to access a VM directly.
- The **Network Interfaces** tab contains the network interfaces connected to a VM. You can also use this tab to manage the connected network interfaces.
- The **Disks** tab lists the disks and file systems used by a VM. From this tab, you can edit, add, or delete a VM's disks and observe details of a VM's file systems.
- The **Snapshots** tab lists and provides management of a VM's snapshots.

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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The serial console provides a direct serial connection and command-line access to a VM. When using the serial console, you might need to press `Enter` a few times for a prompt to appear.

Command Line Virtual Machine Interface

You can also access the serial or VNC consoles of a VM through the `virtctl` command-line tool. The `virtctl` tool is a client application for managing and connecting to VMs 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 can then reference the Kubernetes configuration created by the `oc` client. Select the VM's project with the `oc project` *`vm-project-name`* command.

To access the serial console of a VM, use the `virtctl console` *`vm-name`* command. The escape character for the serial console is `Ctrl-]`.

To access the VNC or graphical interface of a VM, use the `virtctl vnc` *`vm-name`* command.

Other VNC Clients

The `virtctl` tool calls the `remote-viewer` application, which 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 provided in your VNC application of choice to connect to your VM. Some other popular VNC applications are TigerVNC and TightVNC.

References

For more information about working with YAML templates, see *Using templates* at https://docs.openshift.com/container-platform/4.10/openshift_images/using-templates.html

For more information about installing the `virtctl` client, see *Enabling the virtctl client* at https://docs.openshift.com/container-platform/4.10/virt/install/virt-enabling-virtctl.html

For more information about using `virtctl`, see *Using the CLI tools* at https://docs.openshift.com/container-platform/4.10/virt/virt-using-the-cli-tools.html#virt-using-the-cli-tools

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Creating and Accessing Virtual Machines

Create a Red Hat Enterprise Linux 8.5 VM from an existing MariaDB image and use it to manage a database server.

Outcomes

- Create a VM.
- Access the VNC console with the `virtctl` command.
- Access the database with the `oc port-forward` command.
- Manage the VM from the command line.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

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

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

Instructions

1. As the `admin` user, locate and navigate to the Red Hat OpenShift web console.

   1.1. Use the terminal to log in to your Red Hat OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
 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 navigate to https://console-openshift-console.apps.ocp4.example.com.

   1.4. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

2. Create a VM containing a preconfigured MariaDB database server.

2.1. From the OpenShift web console, locate the left panel. If you do not see the left panel, click the main menu icon at the top left of the web console. Navigate to **Virtualization** > **VirtualMachines**. Select the `creating-accessing-vms` project from the **Projects** list.

Figure 2.5: Select the project

2.2. Click **Create virtual machine**, select the **Red Hat Enterprise Linux 8.0 + VM** template and click **Next**.

Figure 2.6: Select the Red Hat Enterprise Linux 8.0 template

2.3. Select the **URL (creates PVC)** option under the **Boot source type** menu.

Specify http://utility.lab.example.com:8080/openshift4/images/mariadb-server.qcow2 in the **Import URL** field. Leave the rest of the values as default and click **Customize virtual machine**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.7: Selecting the boot source

2.4. Specify `databasevm` as the VM name.

Figure 2.8: Naming the VM

2.5. Select `Medium` as the **Flavor** and click **Next**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.9: Flavor and Customize option

2.6. Confirm that the network interface type is `masquerade` and click **Next**.

Figure 2.10: Network interface definition

2.7. Confirm that the attached disks include `cloudinitdisk` and `rootdisk`, and then click **Next**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.11: Disk definition

2.8. Define the `operator1` user and the `redhat123` password in the **Cloud-init** form and click **Review and confirm**.

Figure 2.12: Cloud-init configuration

2.9. Review the settings and click **Create Virtual Machine** to create the VM.

3. Review the details of your new VM.

3.1. Click **See virtual machine details** and wait until the **Status** changes to `Running` before continuing this exercise.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.13: Review the VM

4. Access the VNC console with the `virtctl` command and retrieve the status of the database server.

4.1. In your terminal, select the `creating-accessing-vms` project. List the available VM instances.

```
[student@workstation ~]$ oc project creating-accessing-vms
Now using project "creating-accessing-vms" on server "https://
api.ocp4.example.com:6443".
[student@workstation ~]$ oc get vmi
NAME                  AGE    PHASE      IP           NODENAME    READY
databasevm        3m     Running    10.8.2.12    worker02    True
```

4.2. Open a new terminal window and use the `virtctl vnc` command to access the VM's VNC console. Log in to the console as the `operator1` user with the `redhat123` password previously defined in the `cloud-init` settings.

```
[student@workstation ~]$ virtctl vnc databasevm
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.14: virtctl VNC login

> **Note**
> You can find the credentials specified in `cloud-init` in the OpenShift web console. Navigate to **Virtualization** > **VirtualMachines** and select your VM. Click the **Console** tab and then click **Guest login credentials**.

4.3. Retrieve the status of the database server. Press `q` to exit the status screen.

```
[operator1@databasevm ~]$ systemctl status mariadb
 mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled)...
   Active: active (running)...
...output omitted...
```

5. Access the MariaDB service with the `oc port-forward` command.

5.1. Open a new terminal window and use the `oc get pods` command to determine the name of `virt-launcher` pod of the VMI. Your `virt-launcher` pod has a different name than in the following example.

```
[student@workstation ~]$ oc get pods -n creating-accessing-vms
NAME                              READY   STATUS    RESTARTS   AGE
virt-launcher-databasevm-4qqj6    1/1     Running   0          5m
```

5.2. Use the `oc port-forward` command to connect to the `virt-launcher` pod of the VMI. Specify the `33066:3306` ports.

```
[student@workstation ~]$ oc port-forward virt-launcher-databasevm-4qqj6 33066:3306
Forwarding from 127.0.0.1:33066 -> 3306
Forwarding from [::1]:33066 -> 3306
```

A successful connection does not return your terminal to the command line prompt. This is expected behavior.

5.3. Open another terminal window and use the `mysql` command to connect to the MariaDB database on `127.0.0.1`. Log on as the `devuser` user with the `developer` password and specify the port with the `-P 33066` option.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ mysql -u devuser -h 127.0.0.1 -P 33066 -p
Enter password: developer
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...
```

5.4. The VM already contains a loaded database named `sakila` that has several tables. Retrieve the data from the `actor` and `city` tables.

```
MariaDB [(none)]> USE sakila;
...output omitted...
Database changed
MariaDB [sakila]> SELECT * FROM actor;
+----------+------------+--------------+---------------------+
| actor_id | first_name  | last_name    | last_update         |
+----------+------------+--------------+---------------------+
|        1 | PENELOPE    | GUINESS      | 2006-02-15 04:34:33 |
|        2 | NICK        | WAHLBERG     | 2006-02-15 04:34:33 |
|        3 | ED          | CHASE        | 2006-02-15 04:34:33 |
|        4 | JENNIFER    | DAVIS        | 2006-02-15 04:34:33 |
...output omitted...

MariaDB [sakila]> SELECT * FROM city;
+---------+---------------------------+------------+---------------------+
| city_id | city                      | country_id | last_update         |
+---------+---------------------------+------------+---------------------+
|       1 | A Corua (La Corua)        |         87 | 2006-02-15 04:45:25 |
|       2 | Abha                      |         82 | 2006-02-15 04:45:25 |
|       3 | Abu Dhabi                 |        101 | 2006-02-15 04:45:25 |
|       4 | Acua                      |         60 | 2006-02-15 04:45:25 |
...output omitted...
```

5.5. Exit the database. Return to the terminal that is executing the `oc port-forward` command. Press `Ctrl+C` to end the connection.

```
MariaDB [sakila]> EXIT;
bye
[student@workstation ~]$ oc port-forward virt-launcher-databasevm-4qqj6 33066:3306
Forwarding from 127.0.0.1:33066 -> 3306
Forwarding from [::1]:33066 -> 3306
Handling connection for 33066
^C[student@workstation ~]$
```

6. Manage the VM instance from the command line.

6.1. Use the `virtctl` command to stop the VMI and then delete the VM with the `oc delete` command.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get vmi
NAME                      AGE   PHASE     IP          NODENAME    READY
databasevm                3m    Running   10.8.2.12   worker02    True
[student@workstation ~]$ virtctl stop databasevm
VM databasevm was scheduled to stop
[student@workstation ~]$ oc delete vm databasevm
virtualmachine.kubevirt.io "databasevm" deleted
```

6.2. Confirm that the resource no longer exists.

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Inspect and Monitor Virtual Machines

Objectives

- Monitor VM resources and inspect a VM's configuration.

Virtual Machine Monitoring

Red Hat OpenShift Container Platform (RHOCP) 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 Virtual Machine (VM) in the OpenShift web console.

To view a VM's metrics, such as resource utilization and cluster-level events, use the OpenShift web console to navigate 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 in the **Overview** tab provides identifying information about the VM, including its name, namespace, date of creation, node name, and IP address.

The **Inventory** card lists resources attached to the VM, such as NICs and disks.

The **Status** card includes VM state information and the `QEMU` guest agent status.

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 (1 hour, 6 hours, or 24 hours).

The **Events** card displays messages about the VM's activity and events over the past hour, including resource creation, start and stop events, and configuration errors. Cluster administrators can also review VM event logs by selecting the **Events** tab.

In 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.

Command Line Monitoring

The `oc` client is a command-line utility for managing RHOCP 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 use the `oc` command to perform various tasks, including review and diagnosis of 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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

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, such as recent events and specifications, for a VM.

```
[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 `virsh` commands within the VM's `virt-launcher` pod and its `libvirtd` container.

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

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

RHOCP 4.10 provides new functionalities to the `virtctl` command that incorporate several `virsh` commands, so that you no longer need to access the `virt-launcher` pod's terminal. For example, as a cluster administrator, you can view a list of file systems available to the VMI with the `virtctl fslist vmi-name` command. 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. You can also use the `virsh help` command for a descriptive list of `virsh` commands.

Querying Virtual Resource Metrics

As a cluster administrator, you can monitor and query metrics on all projects in the cluster. RHOCP uses *Role Based Access Control* (RBAC) to grant permissions to cluster resources, including project monitoring.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

By default, developer users can only monitor projects where they have the `view` role permissions. Developer users can view monitoring information in the OpenShift web console by navigating to 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 adm policy add-role-to-user view user -n project
```

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

```
[user@host ~]$ oc adm policy add-role-to-group view group -n project
```

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
```

Only cluster administrators have access to all namespaces concurrently in the **Observe** > **Metrics** interface. From the Developer perspective, you can navigate 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.15: Developer view of monitoring dashboard

As a cluster administrator, you can navigate 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.

Figure 2.16: Administrator view of monitoring dashboard

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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.

There are several metrics that can be helpful to both 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 traffic received (in bytes) on the VM's network.

`kubevirt_vmi_network_transmit_bytes_total`
: Returns the total amount of traffic transmitted (in bytes) on the VM's network.

`kubevirt_vmi_storage_read_traffic_bytes_total`
: Returns the total amount of storage read (in bytes) by the VM.

`kubevirt_vmi_storage_write_traffic_bytes_total`
: Returns the total amount of storage writes (in bytes) of the VM's storage related traffic.

`kubevirt_vmi_storage_iops_read_total`
: Returns the amount of read I/O operations the VM is performing per second.

`kubevirt_vmi_storage_iops_write_total`
: Returns the amount of write I/O operations the VM is performing per second.

`kubevirt_vmi_memory_swap_in_traffic_bytes_total`
: Returns the total amount (in bytes) of memory the virtual guest is swapping in.

`kubevirt_vmi_memory_swap_out_traffic_bytes_total`
: Returns the total amount (in bytes) of memory the virtual guest is swapping out.

References

For more information on 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://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

For more information about using RBAC permissions, refer to *Using RBAC to define and apply permissions* in the Red Hat OpenShift Container Platform 4.10 documentation at
https://docs.openshift.com/container-platform/4.10/authentication/using-rbac.html
This topic is covered in more detail in the training course *Red Hat OpenShift Administration II: Operating a Production Kubernetes Cluster*.

For more information about using the Prometheus Query Language, refer to the official PromQL documentation at
https://prometheus.io/docs/prometheus/latest/querying/basics/

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Inspect and Monitor Virtual Machines

Retrieve the performance metrics of virtual machines located in different projects.

Outcomes

- Inspect the current RBAC policies of different projects.
- Use RBAC policies to permit a non-cluster admin to view the performance metrics of virtual machines in a project.
- Retrieve performance metrics from virtual machines with 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 system for this exercise.

```
[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 located in different projects.

   1.1. Use the terminal to log in to your Red Hat OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
 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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
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. Add the `view` role binding on all resources to the `developer` user in the projects `developers-vms`, `production-vms`, and `testing-vms`.

```
[student@workstation ~]$ oc adm policy add-role-to-user -n developers-vms \
  view developer
clusterrole.rbac.authorization.k8s.io/view added: "developer"
[student@workstation ~]$ oc adm policy add-role-to-user -n production-vms \
  view developer
clusterrole.rbac.authorization.k8s.io/view added: "developer"
[student@workstation ~]$ oc adm policy add-role-to-user -n testing-vms \
  view developer
clusterrole.rbac.authorization.k8s.io/view added: "developer"
```

1.4. Confirm the `developer` user has the assigned permissions on the specified projects.

```
[student@workstation ~]$ oc get rolebindings view -n developers-vms -o wide
NAME   ROLE               AGE   USERS       GROUPS   SERVICEACCOUNTS
view   ClusterRole/view   2m    developer
[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 `developer` user, log in to the OpenShift web console.

2.1. Open a web browser and navigate to https://console-openshift-console.apps.ocp4.example.com.

2.2. Click **htpasswd_provider** and log in as the `developer` user with the `developer` password.

3. Observe the metrics dashboard for the `production-vms` project as the `developer` user.

3.1. From the OpenShift web console left panel, navigate to **Observe** .

3.2. From the **Project** drop-down menu, select the `production-vms` project.

3.3. Read through the performance metrics. Set the time range to `Last 5 minutes` for a more granular view.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.17: Developer performance metrics dashboard

4. View the results of a custom PromQL query for the `testing-vms` project as the `developer` user.

4.1. Navigate to the **Metrics** tab.

4.2. From the **Project** list, select the `testing-vms` project.

4.3. From the **Select Query** list, select `Custom query`.

4.4. Retrieve the `vcpu wait` metric from all virtual machines located in the `testing-vms` project. Execute the following query:

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_vcpu_wait_seconds[5m])))
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.18: vCPU wait metrics testing

5. Generate resource loads on all VMs.

5.1. From the OpenShift web console left panel, select the **Administrator** perspective. Navigate to the **Virtualization** > **VirtualMachines** and change to the `developers-vms` project.

5.2. Click `developers-mariadb` and click the **Console** tab. Click **Open Console in New Window** to open the VNC console of the `developers-mariadb` virtual machine. Log in with the guest credentials located above the console prompt.

Figure 2.19: VNC console of the 'developers-mariadb' virtual machine

5.3. Execute the `stress-load.sh` script located in the home directory of the `developer` user to generate stress load on the virtual machine. The script takes two minutes to complete. You can proceed to the next step while the script executes.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[developer@developers-mariadb ~]$ ./stress-load.sh
stress: info [1847] dispatching hogs: 2 cpu, 2 io, 4 vm, 6 hdd
stress: info [1847] successful run completed in 120s
```

5.4. Return to the OpenShift web console and change to the `production-vms` project. Click `production-mariadb` and click the **Console** tab. Open the VNC console of the `production-mariadb` virtual machine and log in with the guest credentials located above the console prompt.

5.5. Execute the `stress-load.sh` script located in the home directory of the `developer` user to generate a stress load on the virtual machine. The script takes two minutes to complete. You can proceed to the next step while the script executes.

```
[developer@production-mariadb ~]$ ./stress-load.sh
stress: info [1772] dispatching hogs: 8 cpu, 6 io, 1 vm, 2 hdd
stress: info [1772] successful run completed in 120s
```

5.6. Navigate to **Virtualization** > **VirtualMachines** and change to the `testing-vms` project. Click **testing-mariadb** and click the **Console** tab to open virtual machine console. Log in with the guest credentials located above the console prompt.

5.7. Execute the `stress-load.sh` script located in the home directory of the `developer` user to generate a stress load on the virtual machine. The script takes two minutes to complete. You can proceed to the next step while the script executes.

```
[developer@testing-mariadb ~]$ ./stress-load.sh
stress: info [1768] dispatching hogs: 6 cpu, 8 io, 1 vm, 4 hdd
stress: info [1768] successful run completed in 120s
```

6. As the `admin` user, log in to the OpenShift web console.

6.1. Click **User** at the top right of the window and click **Log out**.

6.2. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

7. Use the **Observe** > **Metrics** interface to retrieve updated performance metrics for all VMs located in the `production-vms`, `developers-vms` and `testing-vms` projects.

7.1. Navigate to **Observe** > **Metrics**.

7.2. Identify VMs waiting for Input/Output (I/O) with the `vcpu_wait_seconds` metric. In the **Expression** field, type the following query and press `Enter` to run the query.

```
topk(3, sum by (name,namespace)
 (rate(kubevirt_vmi_vcpu_wait_seconds{namespace=~".*-vms"}[5m])))
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.20: Input expression query

A returned value above 0 indicates that the vCPU wants to run, but the host scheduler is unable to run it. Values above 0 indicate a potential issue with I/O.

7.3. Click **Add query** to add extra **Expression** fields and add the following queries:

A file containing all of the queries for this exercise is accessible at `~/DO316/labs/accessing-monitor/queries.txt`. Use this file to copy and paste the PromQL queries as needed.

- Identify VMs under memory pressure with the guest memory swapping metrics.

```
topk(3, sum by (name, namespace)
 (rate(kubevirt_vmi_memory_swap_in_traffic_bytes_total{namespace=~".*-vms"}[5m])))
```

```
topk(3, sum by (name, namespace)
 (rate(kubevirt_vmi_memory_swap_out_traffic_bytes_total{namespace=~".*-vms"}
[5m])))
```

- Determine if the VMs have encountered any errors receiving network traffic.

```
topk(3, sum by (name, namespace)
 (rate(kubevirt_vmi_network_receive_errors_total{namespace=~".*-vms"}[5m])))
```

- Calculate the total amount (in bytes) of network packets that the VMs received.

```
topk(3, sum by (name, namespace)
 (rate(kubevirt_vmi_network_receive_packets_total{namespace=~".*-vms"}[5m])))
```

- Calculate the I/O performance of the storage devices on the VMs.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
topk(3, sum by (name, namespace)
 (rate(kubevirt_vmi_storage_iops_read_total{namespace=~".*-vms"}[5m])) + sum by
 (name, namespace) (rate(kubevirt_vmi_storage_iops_write_total{namespace=~".*-
vms"}[5m])))
```

7.4. Disable all but the I/O performance query to view it isolated in the graph with the **Disable query** toggle to the right of the expression.

Figure 2.21: I/O performance graph isolated

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Create, Configure and Monitor a Virtual Machine

Create *Virtual Machines (VMs)* located in different projects and retrieve metrics to monitor the resource consumption of the VMs.

Outcomes

- Grant permissions to users and groups to create, manage, and monitor VMs.
- Create VMs through the OpenShift web console and the command line.
- Inspect the configuration of a VM.
- Access the console of a VM through the OpenShift web console and 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 system for this exercise.

```
[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` project.

As the cluster administrator, grant the `admin` role to the `database-admins` group within the `development-db` and `staging-db` projects. You must also grant the `view` role to the `database-admins` group within the `vm-images` project to permit the `database-admins` group access to the `mariadb` *Persistent Volume Claim (PVC)*.

As a member of the `database-admins` group, create the `dev-mariadb` and `staging-mariadb` VMs in their respective projects by using the `mariadb` PVC in the `vm-images` project. Permit the `developer` user to view and access the `dev-mariadb` VM and its performance metrics. Finally, use the collected metrics to determine if the VMs are experiencing resource contention.

1. Log in to the *Red Hat OpenShift Container Platform (RHOCP)* as the `admin` user with the `redhat` password. List the role bindings in the `development-db` and `staging-db` projects.
2. Add the `admin` role binding for all resources in the `development-db` and `staging-db` projects, and the `view` role binding for all resources in the `vm-images` project to the `database-admins` group.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3. Log in to the OpenShift web console as the `dbadmin1` user with the `redhat` password. Create a VM with the **Red Hat Enterprise Linux 8.0+ VM** template in the `development-db` project. Select the `mariadb` PVC in the `vm-images` project as the boot source. Name the VM `dev-mariadb` and select the `small` flavor to assign 1 CPU and 2 GiB memory to the VM. Customize the VM to create the `developer` user with the `developer` password by using the cloud-init form.

4. Use the terminal to log in to the OpenShift cluster as the `dbadmin1` user with the `redhat` password. Add the `view` role binding for the `developer` user in the `development-db` project.

5. Log in to the OpenShift web console as the `developer` user with the `developer` password. Confirm that the `dev-mariadb` VM is accessible to the `developer` user. Open the console of the `dev-mariadb` VM and log in as the `developer` user with the `developer` password.

   In the `dev-mariadb` VM console, use the `watch vmstat -S M` command to retrieve the performance metrics, in MiB, reported by the VM's operating system. 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 amount of memory used in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

6. As the `developer` user, use the OpenShift web console to retrieve the performance metrics of the `dev-mariadb` VM in the `development-db` project. Review the memory and CPU usage metrics for the last 15 minutes by using the performance monitoring dashboards.

7. As the `dbadmin1` user, use the `oc` client to deploy the `staging-mariadb` VM in the `staging-db` project. Use the resource file available at `~/DO316/labs/accessing-review/staging-mariadb.yml` to create the VM.

   Before deploying the VM, make the necessary changes to the file. Specify the `mariadb` PVC in the `vm-images` project as the source PVC. Modify the `cloudInitNoCloud` object to create a local account for the `dbadmin2` user with the `redhat` password.

   After creating the VM, wait a few moments and confirm that the VM is running.

8. 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 the `redhat` 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, in MiB, reported by the VM's operating system. Recall that the output of the `watch vmstat -S M` command includes the `swpd` column to report the amount of memory used in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

9. The `developer` and `dbadmin2` users have 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. Identify if the VMs are waiting for Input/Output (I/O) with the `vcpu_wait_seconds` metric. Verify if the VMs have sufficient available memory with the `memory_available_bytes` metric.

   A file accessible at `~/DO316/labs/accessing-review/queries.txt` contains the necessary queries. Use this file to copy and paste the PromQL queries as needed.

10. 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 modify the

Flavor setting of the `dev-mariadb` VM. Increase the available CPU by 1. Increase the available memory by 2 GiB. Restart the VM to apply the changes.

11. As the `admin` user, modify the **Flavor** setting 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.
12. As the `dbadmin1` user, confirm that the performance metrics have improved for the `dev-mariadb` and `staging-mariadb` VMs with the `watch vmstat -S M` command. Use the `virtctl` client to connect to the VNC console of the VMs. In the console window of each VM, log on as the local user accounts to run the `watch vmstat -S M` command.
13. Performance testing is finished for the `dev-mariadb` and `staging-mariadb` VMs. As the `dbadmin1` user, stop the `dev-mariadb` and `staging-mariadb` VMs with the `virtctl` client.
14. The `dev-mariadb` VM is no longer needed by the `developer` user. Use the `oc delete` command to delete the VM. Log off from the OpenShift cluster with the `oc logout` command. You can close any remaining terminal 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

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Create, Configure and Monitor a Virtual Machine

Create *Virtual Machines (VMs)* located in different projects and retrieve metrics to monitor the resource consumption of the VMs.

Outcomes

- Grant permissions to users and groups to create, manage, and monitor VMs.
- Create VMs through the OpenShift web console and the command line.
- Inspect the configuration of a VM.
- Access the console of a VM through the OpenShift web console and 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 system for this exercise.

```
[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` project.

As the cluster administrator, grant the `admin` role to the `database-admins` group within the `development-db` and `staging-db` projects. You must also grant the `view` role to the `database-admins` group within the `vm-images` project to permit the `database-admins` group access to the `mariadb` *Persistent Volume Claim (PVC)*.

As a member of the `database-admins` group, create the `dev-mariadb` and `staging-mariadb` VMs in their respective projects by using the `mariadb` PVC in the `vm-images` project. Permit the `developer` user to view and access the `dev-mariadb` VM and its performance metrics. Finally, use the collected metrics to determine if the VMs are experiencing resource contention.

1. Log in to the *Red Hat OpenShift Container Platform (RHOCP)* as the `admin` user with the `redhat` password. List the role bindings in the `development-db` and `staging-db` projects.

   1.1. Use the terminal to log in to your Red Hat OpenShift cluster as the `admin` user.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

1.2. Retrieve the role bindings for the `development-db` project.

```
[student@workstation ~]$ oc get rolebinding -n development-db
NAME                    ROLE                                AGE
system:deployers        ClusterRole/system:deployer         2m
system:image-builders   ClusterRole/system:image-builder    2m
system:image-pullers    ClusterRole/system:image-puller     2m
```

1.3. Retrieve the role bindings for the `staging-db` project.

```
[student@workstation ~]$ oc get rolebinding -n staging-db
NAME                    ROLE                                AGE
system:deployers        ClusterRole/system:deployer         2m
system:image-builders   ClusterRole/system:image-builder    2m
system:image-pullers    ClusterRole/system:image-puller     2m
```

2. Add the `admin` role binding for all resources in the `development-db` and `staging-db` projects, and the `view` role binding for all resources in the `vm-images` project to the `database-admins` group.

2.1. Add the `admin` role binding on all resources in the `development-db` project. Confirm that the role binding is successfully applied to the project.

```
[student@workstation ~]$ oc adm policy add-role-to-group admin database-admins \
  -n development-db
clusterrole.rbac.authorization.k8s.io/admin added: "database-admins"
[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
```

2.2. Add the `admin` role binding on all resources in the `staging-db` project. Confirm that the role binding is successfully applied to the project.

```
[student@workstation ~]$ oc adm policy add-role-to-group admin database-admins \
  -n staging-db
clusterrole.rbac.authorization.k8s.io/admin added: "database-admins"
[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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2.3. Add the `view` role binding on all resources in the `vm-images` project. Confirm that the role binding is successfully applied to the project.

```
[student@workstation ~]$ oc adm policy add-role-to-group view database-admins \
  -n vm-images
clusterrole.rbac.authorization.k8s.io/view added: "database-admins"
[student@workstation ~]$ oc get rolebinding -n vm-images
NAME                    ROLE                                AGE
allow-clone-t-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
```

3. Log in to the OpenShift web console as the `dbadmin1` user with the `redhat` password. Create a VM with the **Red Hat Enterprise Linux 8.0+ VM** template in the `development-db` project. Select the `mariadb` PVC in the `vm-images` project as the boot source. Name the VM `dev-mariadb` and select the `small` flavor to assign 1 CPU and 2 GiB memory to the VM. Customize the VM to create the `developer` user with the `developer` password by using the cloud-init form.

3.1. Open a web browser and navigate to https://console-openshift-console.apps.ocp4.example.com. Click **htpasswd_provider** and log in as the `dbadmin1` user with the `redhat` password.

3.2. Change to the **Administrator** perspective and navigate to **Virtualization** > **Templates**. Select the `development-db` project from the **Projects** list.

3.3. Locate the **Red Hat Enterprise Linux 8.0+ VM** template and click **Create VM**.

3.4. Select **PVC (creates PVC)** from the **Boot source type** list. Select the `vm-images` project from the **Persistent Volume Claim project** list. Select the `mariadb` PVC from the **Persistent Volume Claim name** list. Click **Customize virtual machine** to specify additional settings.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.22: Selecting the boot source

3.5. Specify `dev-mariadb` as the VM name. Leave the remaining options at the default values and click **Next**.

3.6. Confirm that the network interface type is `masquerade` and click **Next**.

3.7. Confirm that the `cloudinitdisk` and `rootdisk` disks are attached and click **Next**.

3.8. Define the `developer` user with the `developer` password in the **Cloud-init** form. Click **Review and Confirm**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.23: Defining the user account and password

3.9. Review the settings and click **Create Virtual Machine** to create the VM. Wait a few moments and confirm that the VM's status is `Running`. Log out of the OpenShift web console.

4. Use the terminal to log in to the OpenShift cluster as the `dbadmin1` user with the `redhat` password. Add the `view` role binding for the `developer` user in the `development-db` project.

4.1. Log in as the `dbadmin1` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u dbadmin1 -p redhat
Login successful.

You have access to the following projects and can switch between them with the 'oc
 project projectname' command:

  * development-db
    openshift-virtualization-os-images
    staging-db
    vm-images

Using project "development-db".
```

4.2. Add the `view` role binding to the `developer` user.

```
[student@workstation ~]$ oc adm policy add-role-to-user view developer
clusterrole.rbac.authorization.k8s.io/view added: "developer"
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5. Log in to the OpenShift web console as the `developer` user with the `developer` password. Confirm that the `dev-mariadb` VM is accessible to the `developer` user. Open the console of the `dev-mariadb` VM and log in as the `developer` user with the `developer` password.

In the `dev-mariadb` VM console, use the `watch vmstat -S M` command to retrieve the performance metrics, in MiB, reported by the VM's operating system. 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 amount of memory used in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

5.1. Open a web browser and navigate to https://console-openshift-console.apps.ocp4.example.com. Click **htpasswd_provider** and log in as the `developer` user with the `developer` password.

5.2. Switch to the **Administrator** perspective and navigate to **Virtualization** > **VirtualMachines**. Select the `development-db` project from the **Projects** list.

5.3. Click `dev-mariadb` and click the **Console** tab.

5.4. Click **Open Console in New Window** to open the VNC console of the `dev-mariadb` VM in a new browser window.

5.5. Confirm that the local `developer` account is successfully created. Log in as the `developer` user with the `developer` password in the VM's VNC console window.

5.6. Within the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics as reported by the operating system.

```
[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 that there are potential issues with CPU wait and available free memory.

5.7. Press `Ctrl+C` to exit the `watch vmstat` command. Log out of the `dev-mariadb` console and close the VNC console window.

```
[developer@dev-mariadb ~]$ logout
```

6. As the `developer` user, use the OpenShift web console to retrieve the performance metrics of the `dev-mariadb` VM in the `development-db` project. Review the memory and CPU usage metrics for the last 15 minutes by using the performance monitoring dashboards.

6.1. Switch to the **Developer** perspective in the OpenShift web console.

6.2. Navigate to **Observe** and confirm that the `developement-db` project is selected from the **Projects** list.

6.3. Confirm that the **Kubernetes / Compute Resources / Namespaces(Pods)** dashboard is selected from the **Dashboard** drop-down menu. Click the **Last 15 minutes** option from the **Time Range** menu.

Figure 2.24: Kubernetes/ Compute Resources/ Namespaces (Pods) dashboard

6.4. Scroll down to review the `CPU Usage` and `CPU Quota` charts.

Figure 2.25: CPU Usage and Quota charts

6.5. Scroll down to review the `Memory Usage` and `Memory Quota` charts.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.26: Memory Usage and Quota charts

6.6. Log out of the OpenShift web console.

7. As the `dbadmin1` user, use the `oc` client to deploy the `staging-mariadb` VM in the `staging-db` project. Use the resource file available at `~/DO316/labs/accessing-review/staging-mariadb.yml` to create the VM.

Before deploying the VM, make the necessary changes to the file. Specify the `mariadb` PVC in the `vm-images` project as the source PVC. Modify the `cloudInitNoCloud` object to create a local account for the `dbadmin2` user with the `redhat` password.

After creating the VM, wait a few moments and confirm that the VM is running.

7.1. Open the `~/DO316/labs/accessing-review/staging-mariadb.yml` file in a text editor.

```
[student@workstation ~]$ vi ~/DO316/labs/accessing-review/staging-mariadb.yml
```

7.2. Locate the `.spec.dataVolumeTemplates.spec.source` object. Modify the file to specify `mariadb` as the source PVC name and `vm-images` as the source PVC namespace.

```
...ouput omitted...
  name: staging-mariadb
  namespace: staging-db
spec:
  dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1beta1
    kind: DataVolume
    metadata:
      name: staging-mariadb
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
    spec:
      pvc:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 10Gi
        storageClassName: ocs-external-storagecluster-ceph-rbd
        volumeMode: Block
      source:
        pvc:
          name: mariadb
          namespace: vm-images
...output omitted...
```

7.3. Locate the `spec.template.spec.volumes.cloudInitNoCloud` object. Modify the `#cloud-config` object to create the `dbadmin2` user.

```
...ouput omitted...
      volumes:
      - dataVolume:
          name: staging-mariadb
        name: staging-mariadb
      - cloudInitNoCloud:
          userData: |-
            #cloud-config
            user: dbadmin2
            password: redhat
            chpasswd: { expire: False }
...output omitted...
```

Confirm that all *<CHANGE_ME>* objects are updated before saving and closing the file.

7.4. Use the `oc create -f` command to create the `staging-mariadb` VM. Wait a few moments and confirm that the VM's status is `Running`.

```
[student@workstation ~]$ oc create -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
```

8. 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 the `redhat` 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, in MiB, reported by the VM's operating system. Recall that the output of the `watch vmstat -S M` command includes the `swpd` column to report the amount of memory used in swap spaces. The output also includes the `free` column to report available free memory and the `wa` column to report CPU wait.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

8.1. Open another terminal window. Log in as the `dbadmin2` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u dbadmin2 -p redhat
Login successful.

You have access to the following projects and can switch between them with the 'oc
 project _projectname' command:

  * development-db
    openshift-virtualization-os-images
    staging-db

Using project "development-db".
```

8.2. Switch to the `staging-db` project.

```
[student@workstation ~]$ oc project staging-db
Now using project "staging-db" on server "https://api.opc4.example.com:6443".
```

8.3. Connect to the VNC console of the `staging-mariadb` VM with the `virtctl` client.

```
[student@workstation ~]$ virtctl vnc staging-mariadb
```

Note

The VM takes several minutes to be completely available. Wait for it before moving to the next step.

8.4. Log in as the `dbadmin2` user with the `redhat` password in the `staging-mariadb` VNC console window.

Figure 2.27: Accessing the VNC console

8.5. Within the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics as reported by the VM's operating system.

```
[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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The `wa` column reports a value above 0. The `free` column reports a low value of free memory. The results show that there are potential issues with CPU wait and available free memory.

9. The `developer` and `dbadmin2` users have 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. Identify if the VMs are waiting for Input/Output (I/O) with the `vcpu_wait_seconds` metric. Verify if the VMs have sufficient available memory with the `memory_available_bytes` metric.

A file accessible at `~/DO316/labs/accessing-review/queries.txt` contains the necessary queries. Use this file to copy and paste the PromQL queries as needed.

9.1. In a new browser window, navigate to https://console-openshift-console.apps.ocp4.example.com. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

9.2. If necessary, switch to the **Administrator** perspective in the OpenShift web console.

9.3. Navigate to **Observe** > **Dashboards**.

9.4. Select **KubeVirt / Infrastructure Resources / Top Consumers** dashboard from the **Dashboard** menu. Select **Last 5 minutes** from the **Time Range** menu.

Figure 2.28: KubeVirt top consumers

The charts indicate that the `dev-mariadb` and `staging-mariadb` are experiencing issues with memory and CPU.

9.5. Navigate to **Observe** > **Metrics**. Use PromQL syntax to confirm that the VMs are waiting for Input/Output (I/O) with the `vcpu_wait_seconds` metric. In the

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Expression field, type the following query and press `Enter` to retrieve the performance metric:

```
topk(3, sum by (name,namespace)
 (rate(kubevirt_vmi_vcpu_wait_seconds{namespace=~".*-db"}[5m])))
```

A returned value above 0 indicates that the vCPU wants to run, but the host scheduler is unable to run it. Values above 0 indicate a potential issue with I/O and a need to increase available CPU resources to the VMs.

Figure 2.29: vCPU wait metrics

9.6. Identify if the VMs have available free memory with the `memory_available_bytes` metric. Click **Add query** and execute the following 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 do not have sufficient available memory and need more memory allocated.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 2.30: Available memory metrics

10. 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 modify the **Flavor** setting of the `dev-mariadb` VM. Increase the available CPU by 1. Increase the available memory by 2 GiB. Restart the VM to apply the changes.

10.1. In the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `development-db` project from the **Project** list.

10.2. Click the `dev-mariadb` VM to open the **Virtual Machine Overview** window.

10.3. Click the **Details** tab and scroll down to the **Scheduling and resource requirements** section.

10.4. Under **Flavor**, click the **Small: 1 CPU|2 GiB Memory** link to open the **Edit Flavor** window. Selection `Custom` from the **Flavor** list.

Figure 2.31: Flavor settings for dev-mariadb

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

10.5. Enter 2 in the **CPU** field. Enter 4 `GiB` in the **Memory** field. Click **Save and Restart** to apply the changes.

Figure 2.32: Edit Flavor

10.6. Click **Restart** to confirm the VM restart. Wait for the VM to enter the `Running` status.

11. As the `admin` user, modify the **Flavor** setting 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.

11.1. In the OpenShift web console, select the `staging-db` project from the **Project** list.

11.2. Click the `staging-mariadb` VM to open the **Virtual Machine Overview** window.

11.3. Click the **Details** tab and scroll down to the **Scheduling and resource requirements** section.

11.4. Under **Flavor**, click the **Small: 1 CPU|2 GiB Memory** link to open the **Edit Flavor** window. Selection `Custom` from the **Flavor** list.

11.5. Enter 2 in the **CPU** field. Enter 4 `GiB` in the **Memory** field. Click **Save and Restart** to apply the changes.

11.6. Click **Restart** to confirm the VM restart. Wait for the VM to enter the `Running` status.

12. As the `dbadmin1` user, confirm that the performance metrics have improved for the `dev-mariadb` and `staging-mariadb` VMs with the `watch vmstat -S M` command. Use the `virtctl` client to connect to the VNC console of the VMs. In the console window of each VM, log on as the local user accounts to run the `watch vmstat -S M` command.

12.1. Open a new terminal window and log in to the cluster as the `dbadmin1` user.

```
[student@workstation ~]$ oc login -u dbadmin1 -p redhat
Login successful.
```

12.2. Use the `virtctl` client to open the VNC console of the `dev-mariadb` VM in the `developement-db` project.

```
[student@workstation ~]$ virtctl vnc dev-mariadb -n development-db
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

12.3. In the new VNC console window, log in as the `developer` user with the `developer` password.

12.4. Within the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics as reported by the operating system.

```
[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. Close the console window.

12.5. In the terminal window, use the `virtctl` client to open the VNC console of the `staging-mariadb` VM in the `staging-db` project.

```
[student@workstation ~]$ virtctl vnc staging-mariadb -n staging-db
```

12.6. In the new VNC console window, log in as the `dbadmin2` user with the `redhat` password.

12.7. Within the VNC console window, use the `watch vmstat -S M` command to retrieve the VM's performance metrics as reported by the operating system.

```
[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. Close the console window.

13. Performance testing is finished for the `dev-mariadb` and `staging-mariadb` VMs. As the `dbadmin1` user, stop the `dev-mariadb` and `staging-mariadb` VMs with the `virtctl` client.

13.1. In the terminal window, use the `virtctl` client to stop the `dev-mariadb` VM in the `development-db` project. 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
```

13.2. Use the `virtctl` client to stop the `staging-mariadb` VM in the `staging-db` project. 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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

14. The `dev-mariadb` VM is no longer needed by the `developer` user. Use the `oc delete` command to delete the VM. Log off from the OpenShift cluster with the `oc logout` command. You can close any remaining terminal windows.

14.1. Delete the VM.

```
[student@workstation ~]$ oc delete vm/dev-mariadb -n development-db
virtualmachine.kubevirt.io "dev-mariadb" deleted
```

14.2. Log off from the OpenShift cluster.

```
[student@workstation ~]$ oc logout
Logged "dbadmin1" out on "https://api.ocp4.example.com:6443"
```

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

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

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

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- In Red Hat OpenShift Virtualization, a VM object defines the template to create a virtual machine instance (VMI), which is the running instance of the VM inside your cluster.
- You can attach a persistent volume to a VMI by defining a persistent volume claim (PVC), representing 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** > **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 bring up 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, but you can install other operating systems, such as Ubuntu or openSUSE, by installing the KubeVirt `common-templates` package in your OpenShift cluster.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 3

Configuring Kubernetes Networking for Virtual Machines

Goal

Configure standard Kubernetes network objects and external access for VMs and virtual machine-backed applications.

Objectives

- Describe VM communication on the Kubernetes SDN, create a ClusterIP service for a VM, and create a network policy for virtual machines.
- Configure external access for virtual machines.

Sections

- Describing Kubernetes Networking Objects (and Guided Exercise)
- Configuring External Access to Virtual Machines (and Guided Exercise)

Lab

- Configuring Kubernetes Networking for Virtual Machines

Describing Kubernetes Networking Objects

Objectives

- Describe VM communication on the Kubernetes SDN, create a ClusterIP service for a VM, and create a network policy for virtual machines.

Review the Kubernetes SDN

Kubernetes automatically assigns an IP address to every pod. By default, the pods can communicate with each other even if they run on different cluster nodes or belong to different Kubernetes namespaces.

Kubernetes implements this infrastructure by managing a *software-defined network (SDN)*. The SDN allows Kubernetes to control the network traffic and the network resources programmatically.

Introducing the Cluster Network Operator

Red Hat OpenShift Container Platform (RHOCP) 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. That specification allows vendors to provide different plug-ins for their implementation.

RHOCP includes a few plug-in providers such as *OpenShift SDN*, *OVN-Kubernetes*, and *Kuryr*. By default, the installation process selects the OpenShift SDN provider, which implements the SDN by using *Open vSwitch (OVS)* virtual switches.

Expose Pods and Virtual Machines by Using Services

The following schema shows that the Kubernetes pod SDN connects all the pods to a shared network.

Figure 3.1: Kubernetes pod SDN

Pods get an IP address on the shared network when they start. Pods are constantly created and destroyed 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 assigns a different IP address each time it creates a pod, pods are not easily addressable.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Instead of having a pod discover the IP address of another pod, you can use Kubernetes *services* to provide a single, unique IP address for other pods to use, independent of where the pods are running.

Services rely on selectors, or *labels*, that indicate which pods receive the traffic through the service. Kubernetes adds each pod matching these selectors to the service resource as an *endpoint*. As pods are created and destroyed, the service automatically updates the endpoints. The service load balances client requests across member pods.

Kubernetes uses one subnet for pods and one subnet for services. When you address the service IP, Kubernetes forwards the traffic transparently to the pods.

The following diagram shows three pods running the API of an application. These pods are not all running on the same node. The `service1` service balances the load between the pods. The `service2` service forwards the requests to a VM.

Figure 3.2: Pod and service subnets

You can configure the address range of each network during installation. Run the `oc get network/cluster -o yaml` command to list the ranges that your cluster is using.

Kubernetes provides several types of services. With a service of the `ClusterIP` type, the default type, Kubernetes assigns an internal IP address to the service. That IP address is only accessible from inside the cluster. Other service types expose the IP address outside the cluster, but Red Hat OpenShift provides a different mechanism for that purpose, which is described elsewhere in this course.

Address a Service by its DNS Record

Your application can find the IP address of a Kubernetes service by querying the internal DNS service.

The *DNS Operator* deploys and runs a DNS server. The operator monitors the services and automatically creates and updates DNS records. For services, the DNS Operator manages the `svc.cluster.local` domain name and creates records in the form `servicename.namespace.svc.cluster.local`. For example, you can retrieve the IP address of the `backend` service in the `prod2` namespace by querying the `backend.prod2.svc.cluster.local` record.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

In addition to managing the DNS server, the operator automatically creates the `/etc/resolv.conf` configuration file inside the pods so that name resolution is immediately available without further configuration.

Create Services for Virtual Machines

VMs in Kubernetes run inside `virt-launcher` pods. The pods get an IP address on the pod SDN and you can create a Kubernetes service to access them through a fixed IP address.

> **Note**
> The IP address Kubernetes assigns to the pod is not the IP address you get from inside the VM.
>
> The `virt-launcher` process running inside the pod runs a DHCP server that 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 expose the VM by creating a Kubernetes service. With that configuration, pods and other VMs inside the cluster can access the application that the VM hosts.

Prepare a Label for a Service

Because services rely on labels to select the targeted pods, you need to add a label to the `VirtualMachine` resource.

Use the `oc edit vm` *`vmname`* command to edit the resource from the command line. From the Red Hat OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the VM, and then navigate to the **YAML** tab.

> **Note**
> The web console displays some other labels under the **Details** tab. These labels are not the labels you have to edit.

The following `VirtualMachine` resource definition shows the label `tier: backend` that the Kubernetes service will use 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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
      labels:
        tier: backend
...output omitted...
```

Note

Add the label to the `labels` section under 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.

Service resources rely on labels set at the `virt-launcher` pod level. However, when you edit the `VirtualMachine` resource, Kubernetes does not automatically propagate your changes to the `VirtualMachineInstance` and `virt-launcher` pod resources. You need to delete these resources for Kubernetes to recreate them automatically from the `VirtualMachine` resource definition.

One method to recreate the resources is by restarting the VM. When you restart the VM, Kubernetes recreates 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, navigate to **Virtualization** > **VirtualMachines**, select the VM, and then click **Actions** > **Restart**. From the command line, use the `oc delete vmi vmname` command.

In your environment, if you cannot restart the VM for the moment, then manually set the 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-q4n2k   1/1     Running   0          41m
[user@host ~]$ oc label pod virt-launcher-backendvm-q4n2k tier=backend
pod/virt-launcher-backendvm-q4n2k labeled
```

From the OpenShift web console, navigate to **Workloads** > **Pods**, select the `virt-launcher` pod, and then navigate to the **Details** tab. Click **Edit** under the **Labels** section and then add the label.

Important

Even though you set the label to the `virt-launcher` pod, you still need to 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 recreates them from the `VirtualMachine` resource. If you do not associate the label to the `VirtualMachine` resource also, then the label you set to the `virt-launcher` pod is lost.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Configure a Service for a Virtual Machine

After you set a label for the `VirtualMachine` resource, define the service resource definition as follows:

```
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 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 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.

To create the service from the preceding resource file, use the `oc apply -f file.yaml` command. From the OpenShift web console, navigate to **Networking** > **Services**, click **Create Service**, and then use the YAML editor to declare the service.

After you create the Kubernetes service, the web console displays the IP address. From the command line, use the `oc get svc` command:

```
[user@host ~]$ oc get svc
NAME      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
backend   ClusterIP   172.30.123.204   <none>        80/TCP     2m57s
```

For each pod matching the selector, Kubernetes automatically creates an endpoint resource. 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
```

When you create the preceding 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 running on the VM from another pod or another VM

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 `svc.cluster.local` and `prod2.svc.cluster.local` as search domains, you can also use the short names `backend.prod2` or `backend` 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.
sh-4.4# getent hosts backend.prod2.svc.cluster.local
172.30.123.204  backend.prod2.svc.cluster.local
sh-4.4# getent hosts backend.prod2
172.30.123.204  backend.prod2.svc.cluster.local
sh-4.4# getent hosts backend
172.30.123.204  backend.prod2.svc.cluster.local
```

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 running in a namespace can connect to a pod or a VM running in another namespace.

In some environments, you might want to establish stricter rules. For example, you might want to allow connections between pods running in the same namespace only. Or you might want to control the traffic between two namespaces tightly.

Kubernetes *network policies* allow you to configure isolation policies for individual pods. You apply network policies to pods and namespaces by using labels.

From the OpenShift web console, navigate to **Administration** > **Namespaces**, select the namespace, and then navigate to the **Details** tab. Click **Edit** under the **Labels** section and then add the label. From the command line, use the `oc label namespace` command to add a label to a namespace. The following command adds the `name=client-ns` label to the `prod-front` namespace:

```
[user@host ~]$ oc label namespace prod-front name=client-ns
```

The following network policy applies to all pods with the label `tier=backend` in the current namespace. The policy allows ingress traffic to port 8080 from pods and VMs whose label is `tier=front` in the namespace with the `name=client-ns` label.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: network-1-policy
spec:
  podSelector:  (1)
    matchLabels:
      tier: backend

  ingress:  (2)
  - from:  (3)
    - namespaceSelector:  (4)
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        matchLabels:
          name: client-ns
      podSelector:
        matchLabels:
          tier: front
    - namespaceSelect: ❺
        matchLabels:
          name: server-ns
    ports: ❻
    - port: 8080
      protocol: TCP
```

❶ The top-level `podSelector` field is required and defines to which pods and VMs in the current namespace the network policy applies. If the `podSelector` field is empty, then the policy applies to all the pods and VMs in the current namespace.

❷ The `ingress` field lists the ingress traffic rules.

❸ The `from` field lists the allowed sources. The incoming traffic is allowed if any of the rules match (logical OR).

❹ The first rule allows traffic from namespaces with the `name=client-ns` label and from pods and VMs with the `tier=front` label.

❺ The second rule allows traffic coming for all the pods and VMs in the namespaces with the `name=server-ns` label.

❻ The `ports` field lists the destination ports that allow traffic to reach the selected pods.

The following less restrictive network policy allows traffic from all the pods and ports in namespaces with the `name=client-ns` label to all pods and ports in the current namespace.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: network-2-policy
spec:
  podSelector: {}

  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          name: client-ns
```

You can combine multiple network policies to satisfy network requirements.

To create a network policy from a resource file, use the `oc apply -f file.yaml` command. From the OpenShift web console, navigate to **Networking** > **NetworkPolicies**, click **Create Network Policy**, and then complete the form. You can also click **YAML view** to use the YAML editor to create the resource.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

References

For more information, refer to the *Cluster Network Operator in OpenShift Container Platform*, *DNS Operator in OpenShift Container Platform*, and *Network Policy* sections in the *Red Hat OpenShift Container Platform Networking* guide at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/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://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#virtual-machine-networking

For more information about service resources, refer to the *Service* section in the Kubernetes documentation at https://kubernetes.io/docs/concepts/services-networking/service/

Guided Exercise

Describing Kubernetes Networking Objects

Configure a service to establish a connection between two VMs, one with a MariaDB server and the other with a database client. You will then define a network policy to limit access to the database server to pods within the `network-services` project.

Outcomes

- Create a service of the `ClusterIP` type.
- Link the service to a virtual machine.
- 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `network-services` namespace and starts two virtual machines in that namespace: the `mariadb-server` VM hosts a MariaDB database, and the `mariadb-client` VM provides the client `mysql` command-line tool.

```
[student@workstation ~]$ lab start network-services
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console. Confirm that the two VMs are 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 redhat \
  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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1.3. 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 the `redhat` password when prompted.

1.4. Navigate to **Virtualization** > **VirtualMachines** and then select the `network-services` project. Confirm that the `mariadb-client` and `mariadb-server` VMs are running.

2. Add a label to the `virt-launcher` pod associated with the `mariadb-server` VM. You use this label later when you create the service.

2.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `mariadb-server` VM, and then navigate to the **YAML** tab to access the YAML editor.

If you prefer the command line, then access the terminal on the `workstation` machine and use the `oc edit` command to edit the configuration of the `mariadb-server` VM.

```
[student@workstation ~]$ oc edit vm mariadb-server -n network-services
```

2.2. Add the `app: mariadb-server` label under the `.spec.template.metadata.labels` path.

Important

There is a `labels` section before the `spec` section that already includes the `app: mariadb-server` label. That is not the section you have to edit.

You must assign the label to the `virt-launcher` pod that Kubernetes creates from the `VirtualMachine` resource. Setting a label on that pod requires declaring it under the `.spec.template.metadata.labels` path.

```
...output omitted...
spec:
  dataVolumeTemplates:
  ...output omitted...
  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...
```

If you use the OpenShift web console, then click **Save** when done. If you use the command line, then save the file and exit the editor.

2.3. Restart the VM so that Kubernetes recreates an instance that includes the new label. From the OpenShift web console, click **Actions** > **Restart**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

If you prefer the command line, then delete the `VirtualMachineInstance` (VMI) resource named `mariadb-server`. The command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi mariadb-server -n network-services
virtualmachineinstance.kubevirt.io "mariadb-server" deleted
```

2.4. From the terminal on the `workstation` machine, confirm that Kubernetes recreates the `VMI` resource and the `virt-launcher` pod. You might have to run the command several times because it takes time for the pod to start.

```
[student@workstation ~]$ oc get vmi,pods -n network-services
NAME                                                AGE  PHASE    IP          ...
virtualmachineinstance.kubevirt.io/mariadb-client   24m  Running  10.8.2.33   ...
virtualmachineinstance.kubevirt.io/mariadb-server   94s  Running  10.11.0.41  ...

NAME                                      READY   STATUS    RESTARTS   AGE
pod/virt-launcher-mariadb-client-dcjtc    1/1     Running   0          24m
pod/virt-launcher-mariadb-server-274ts    1/1     Running   0          94s
```

3. Create a service named `mariadb` of 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 port 3306.

3.1. From the OpenShift web console, navigate to **Networking** > **Services**, select the `network-services` project, and then click **Create Service**. Complete the YAML file with the following content and then click **Create**.

If you prefer the command line, then create the `mariadb.yaml` resource definition file with the following content and run the `oc apply -f mariadb.yaml` command.

```
apiVersion: v1
kind: Service
metadata:
  name: mariadb
  namespace: network-services
spec:
  type: ClusterIP
  selector:
    app: mariadb-server
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
```

By using that configuration, the DNS operator creates the `mariadb.network-services.svc.cluster.local` record.

The `lab` command has prepared the `/home/student/DO316/labs/network-services/mariadb.yaml` file so that you can compare it with your version.

3.2. From the terminal on the `workstation` machine, confirm that the `mariadb` service has an active endpoint.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get endpoints -n network-services
NAME      ENDPOINTS         AGE
mariadb   10.11.0.41:3306   110s
```

4. Log in to the `mariadb-client` VM, and then verify database connectivity.

4.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `mariadb-client` VM, and then access the **Console** tab.

4.2. Log in as the `developer` user with `developer` as the password.

4.3. Connect to the database 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 --user=devuser --password=developer \
  --host=mariadb.network-services.svc.cluster.local
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.3.28-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)]>
```

4.4. Select the `sakila` database and run an SQL query to verify connectivity.

```
MariaDB [(none)]> USE sakila;
...output omitted...
Database changed.
MariaDB [sakila]> SHOW TABLES;
+------------------+
| Tables_in_sakila |
+------------------+
| actors           |
| actor_info       |
| address          |
| category         |
...output omitted...
```

4.5. Log out of the database.

```
MariaDB [sakila]> quit
Bye
[developer@mariadb-client ~]$
```

4.6. Confirm that the DHCP server running inside the `virt-launcher` pod configures the VM system so that you can use the short service name instead of the *fully qualified domain name (FQDN)*.

Review the `search` line in the `/etc/resolv.conf` file.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[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
```

With that configuration, you can refer to the `mariadb` service as `mariadb`, `mariadb.network-services`, or `mariadb.network-services.svc`.

4.7. Access the database by using the `mariadb.network-services` name. Log out of the database after you have confirmed that you can use the short name.

```
[developer@mariadb-client ~]$ mysql --user=devuser --password=developer \
  --host=mariadb.network-services
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...
MariaDB [sakila]> quit
Bye
[developer@mariadb-client ~]$
```

5. Verify that you can access the database running on the `mariadb-server` VM from any namespace. Instead of creating a client VM to perform that test, start an interactive pod by using a container image that provides the `mysql` client command.

5.1. From the terminal on the `workstation` machine, run a temporary interactive pod in the `default` namespace. The pod uses the `rhel8/mariadb-105` container image that provides the `mysql` command.

```
[student@workstation ~]$ oc run testdb -it --rm --image=rhel8/mariadb-105 \
  -n default --command -- mysql --connect-timeout=5 --user=devuser \
  --password=developer --host=mariadb.network-services.svc.cluster.local
If you don't see a command prompt, try pressing enter.

Enter

MariaDB [(none)]>
```

You can copy and then paste the command from the `/home/student/DO316/labs/network-services/commands.txt` file that the `lab` command has prepared for you.

5.2. Select the `sakila` database and then run an SQL query to verify connectivity. When done, log out of the database.

```
MariaDB [(none)]> USE sakila;
...output omitted...
Database changed
MariaDB [sakila]> SHOW TABLES;
+---------------------------+
| Tables_in_sakila          |
+---------------------------+
| actor                     |
| actor_info                |
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
| address                    |
| category                   |
...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 ~]$
```

The test confirms that you can access the service from any namespace.

6. Protect the database server so that only clients running in the `network-services` namespace can access it.

6.1. Add the `allowing: db-clients` label to the `network-services` namespace. You use that label later when you create the network policy.

From the OpenShift web console, navigate to **Administration** > **Namespaces**, select the `network-services` namespace, and then access the **Details** tab. Click **Edit** under the **Labels** section, add the `allowing=db-clients` label, and then click **Save**. Do not remove the labels that might already exist.

If you prefer the command line, then use the `oc label` command:

```
[student@workstation ~]$ oc label namespace network-services allowing=db-clients
namespace/network-services labeled
```

6.2. Prepare the network policy resource. From the OpenShift web console, navigate to **Networking** > **NetworkPolicies**, select the `network-services` project, click **Create Network Policy**, and then click **YAML view**. Complete the file with the following content and then click **Create**.

If you prefer the command line, then create the `policy.yaml` resource definition file with the following content and run the `oc apply -f policy.yaml` command.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-db-clients
  namespace: network-services
spec:
  podSelector:
    matchLabels:
      app: mariadb-server

  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          allowing: db-clients
    ports:
    - port: 3306
      protocol: TCP
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The `lab` command has prepared the `/home/student/DO316/labs/network-services/policy.yaml` file so that you can compare it with your version.

The `podSelector` section indicates that the policy applies to pods with the `app: mariadb-server` label. You set that label to the `virt-launcher` pod associated with the `mariadb-server` VM in a preceding step.

The `ingress` section allows inbound connections to port 3306 from all pods in namespaces with the `allowing: db-clients` label.

7. Confirm that the `mariadb-client` VM can still access the database. That VM is running in the `network-services` namespace. Confirm that pods running on a different namespace cannot access the database anymore.

7.1. Use the OpenShift web console to access the console of the `mariadb-client` VM. Connect to the database server and then execute some SQL queries.

```
[developer@mariadb-client ~]$ mysql --user=devuser --password=developer \
  --host=mariadb.network-services
...output omitted...
MariaDB [(none)]> USE sakila;
...output omitted...
Database changed.
MariaDB [sakila]> SHOW TABLES;
+------------------+
| Tables_in_sakila |
+------------------+
| actors           |
| actor_info       |
| address          |
| category         |
...output omitted...
```

7.2. From the terminal on the `workstation` machine, run the test pod again in the `default` namespace. This time, the command does not display the MariaDB prompt and times out after five seconds. This error confirms that you cannot access the database from namespaces that do not have the `allowing: db-clients` label.

```
[student@workstation ~]$ oc run testdb -it --rm --image=rhel8/mariadb-105 \
  -n default --command -- mysql --connect-timeout=5 --user=devuser \
  --password=developer --host=mariadb.network-services.svc.cluster.local
If you don't see a command prompt, try pressing enter.

Enter

ERROR 2002 (HY000): Can't connect to MySQL server on 'mariadb.network-
services.svc.cluster.local' (115)
```

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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ lab finish network-services
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Configuring External Access to Virtual Machines

Objectives

- Configure external access for virtual machines.

Describe Kubernetes Services Objects with External Access

Red Hat OpenShift Container Platform (RHOCP) and Kubernetes provide several mechanisms to make your applications available from outside the cluster.

Kubernetes provides different service resource types to configure external access:

- The `NodePort` resource type exposes a network port on all your cluster nodes. That port redirects the incoming traffic to the service's pods or virtual machines (VMs).

- The `LoadBalancer` resource type deploys and configures a load balancer at the cloud provider level. If a cloud provider does not host your cluster, then the load balancer allocates an external IP address to your service.

Red Hat OpenShift provides an alternative to using services: `route` resources provide ingress traffic to web and TLS applications. Kubernetes provides `ingress` resources that are similar to Red Hat OpenShift `route` resources.

Because virtual machines run in `virt-launcher` pods, all these methods also work with VMs.

When you migrate a VM into RHOCP 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 running outside your OpenShift cluster. You can expose legacy applications that you plan later to convert to containers. After converting the application, you reconfigure the external access so that clients can continue accessing the application with the same DNS name.

Configure Node Port Service Resources

Remember that the service resources use the `ClusterIP` resource type by default. With that type, the service is only accessible from inside the cluster.

By setting the `type` parameter to `NodePort`, Kubernetes opens the same network port on all the cluster nodes and then redirects the incoming traffic to your pods or VMs. By default, Kubernetes 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 3.3: Incoming traffic through a node port service

The following `service` definition declares a service resource of type `NodePort`:

```
apiVersion: v1
kind: Service
metadata:
  name: ssh
  namespace: prod2
spec:
  type: NodePort  (1)
  selector:
    vmtype: linux
  ports:
    - protocol: TCP
      port: 22  (2)
      targetPort: 22
      nodePort: 30022  (3)
```

1. The `type` parameter declares the service type: `ClusterIP`, `NodePort`, or `LoadBalancer`.
2. Inside the cluster, the service makes the application available on port 22. The service behaves similarly to a `ClusterIP` service.
3. The `nodePort` parameter provides the port number that Kubernetes must open on all the cluster nodes. The port 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 Kubernetes selects an available port.

You can use the `NodePort` service type for TCP and UDP traffic. If possible, Red Hat recommends avoiding using the `NodePort` service type for the following reasons:

- You must expose the IP addresses of your cluster nodes from outside the cluster.
- You cannot select a port outside the default range (30000-32767).
- In cloud environments, the cluster nodes' IP address might not be permanent. For example, the cloud infrastructure might add and remove nodes to adapt the cluster to the load.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Configure Load Balancer Service Resources for Cloud Environments

On clusters deployed on cloud infrastructure, you can create services of the `LoadBalancer` type to provide external access to your applications.

When you create that type of service, Kubernetes automatically configures an external load balancer from the cloud provider. For example, when you create a service on a cluster deployed on *Amazon Web Services (AWS)*, Kubernetes creates a load balancer with *Amazon Elastic Load Balancing (ELB)*. With a cluster deployed on *IBM Cloud*, Kubernetes creates the load balancers with *IBM Cloud load balancers*. On *Microsoft Azure*, Kubernetes creates the load balancers with *Azure Load Balancer*.

Figure 3.4: Incoming traffic through the cloud provider load balancer

The following `service` definition declares a service resource of type `LoadBalancer`:

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  namespace: prod2
spec:
  type: LoadBalancer  (1)
  loadBalancerIP: 40.121.123.12  (2)
  selector:
    tier: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

1. The `type` parameter declares the service type: `ClusterIP`, `NodePort`, or `LoadBalancer`.
2. Some cloud providers accept an IP address to assign to the load balancer. With some providers, you must reserve that address before creating the service. If you do not set the `loadBalancerIP` parameter, then cloud providers assign an ephemeral address. Use the `oc get svc` command to retrieve that IP address and the associated DNS name.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

You can use the `LoadBalancer` service types for TCP and UDP traffic. Because each `LoadBalancer` service creates a load balancer with the cloud provider, it can become expensive to use that kind of service.

Configure Load Balancer Service Resources for Non-cloud Environments

For clusters 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 prepared by the cluster administrators.

Your cluster administrators must configure these external IP addresses so that the network routes them to at least one 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 virtual machines associated with the service.

> **Note**
> For security reasons, Red Hat OpenShift disables by default the use of the `LoadBalancer` type for non-cloud environments.

The following `service` definition declares a service resource of type `LoadBalancer`:

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  namespace: prod2
spec:
  type: LoadBalancer (1)
  externalIPs:
    - 192.168.0.42 (2)
  selector:
    tier: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

1. The `type` parameter declares the service type: `ClusterIP`, `NodePort`, or `LoadBalancer`.
2. The `externalIPs` parameter lists the external IP addresses to assign to the service. Those addresses must be available in the pool of addresses prepared by the cluster administrators. If you do not give an IP address, then Kubernetes selects one for you.

You can use `LoadBalancer` service types for TCP and UDP traffic.

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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Routes are not a type of service. Instead, they are Red Hat OpenShift resources. Because routes rely on services to locate applications' pods and virtual machines, you still need to create services of type `ClusterIP` for your applications.

Red Hat OpenShift implements routes by deploying a cluster-wide router service, which by default 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 3.5: 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 form *`routename-namespace.default_domain`*. Your RHOCP administrators configure the *`default_domain`* base name during cluster installation. For example, if your default domain is `apps.mycompany.com` and you create a route named `intranet` in the `prod` project, then the DNS name is `intranet-prod.apps.mycompany.com`.

An important consideration for your RHOCP administrators is that they 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 running the router.

> **Note**
> The DNS server that hosts the wildcard domain knows nothing about route hostnames. It merely resolves any name to the configured IP addresses. Only the Red Hat OpenShift router knows about route hostnames, treating 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 only work with certain types of traffic: HTTP, HTTPS with Server Name Indication (SNI), and TLS with SNI. SNI is a TLS feature that allows the client to send the name of the host 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 types of traffic, such 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 type `ClusterIP` that points to your pods or virtual machines, you can create a route for your application. Like any other Red Hat OpenShift resource, you can create a resource file in the YAML or the JSON format to declare your route. However, Red Hat OpenShift provides the `oc expose service` command to simplify the creation process:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[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 if you want to give it a different name.

Red Hat OpenShift automatically assigns a DNS name to the route. You can use the `--hostname` option to force that DNS name.

Use the `oc get route` command to retrieve the DNS name that Red Hat OpenShift assigns 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
```

Create Path-based Routes

The `route` resource definition accepts the `path` variable that you can use 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 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   ...
```

Secure Routes

Routes can be secured (TLS or HTTPS) or unsecured (HTTP). Secure routes provide the ability to use several types of TLS termination to serve certificates to the client.

A secured route specifies the TLS termination of the route. The following list describes the available types of termination:

Edge

TLS termination occurs at the router before Red Hat OpenShift routes the traffic to the pods or 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 on 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.

Configure routes for Virtual Machines

Configuring a route for a virtual machine is identical to configuring a route for a pod.

The preparation step requires that you create a service for the `virt-launcher` pod:

1. Add a label to the `VirtualMachine` resource under 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 have created the service, you can add the route. You can use the `oc` command or the web console.

With the command line, use the `oc expose service` command to create the route:

```
[user@host ~]$ oc expose service/web
route.route.openshift.io/web exposed
```

Use the web console to navigate to **Networking** > **Routes**, click **Create Route**, and then fill the form with the route details. You must name the route and select the associated service and its ports.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 3.6: Create a route by using the web console

Allow Traffic from the Router Pods

When you use network policies in your project, you must add a rule to allow the traffic coming from the router pods. Otherwise, the `route` resources do not work because the router pods cannot reach the destinations 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
apiVersion: v1
kind: Namespace
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 allow the traffic from the router pods:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
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)
```

1. The network policy allows the traffic coming from the `openshift-ingress` namespace because that namespace has the `network.openshift.io/policy-group: ingress` label.
2. 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 that have some of the same features as routes. Some route features, such as TLS re-encryption, are not available with `ingress` resources.

The following example declares 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
```

1. The `host` parameter provides the URL to assign to the application.
2. The `path` parameter provides the path component of the URL.
3. The `pathType` parameter specifies how to match the path component of the URL. If set to `Exact`, then the path must be exactly equal to the `path` parameter. If set to `Prefix`, then any path that starts with the `path` component matches.
4. The `service` section provides the service to associate with the `ingress` resource.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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
NAME CLASS   HOSTS                                ADDRESS ...
web  <none>  web-production.apps.mycompany.com  router-default.apps.mycompany.com
[user@host ~]$ oc get routes
NAME      HOST/PORT                          PATH   SERVICES   PORT ...
web-djh2b web-production.apps.mycompany.com   /      web        8080-tcp ...
```

References

Kubernetes Ingress versus OpenShift Route
https://cloud.redhat.com/blog/kubernetes-ingress-vs-openshift-route

For more information, refer to the *Configuring ingress cluster traffic* chapter in the *Red Hat OpenShift Container Platform Networking* guide at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/networking/index#configuring-ingress-cluster-traffic

For more information on secured routes, refer to the *Secured Routes* section in the *Red Hat OpenShift Container Platform Networking* guide at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/networking/index#configuring-default-certificate

Kubernetes: "Publishing Services"
https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Configuring External Access to Virtual Machines

Configure a service and a route to establish external access for a web server 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.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

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 then navigate to the Red Hat OpenShift web console. Confirm that the 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 redhat \
  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 navigate to `https://console-openshift-console.apps.ocp4.example.com`. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

   1.4. Navigate to **Virtualization** > **VirtualMachines** and then select the `web-servers` project. Confirm that the `hello-web` VM is running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. You need a `service` resource for the VM to create a `route` resource later. Before you can create the service, you must add the `app: helloworld` label to the `virt-launcher` pod.

2.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `hello-web` VM, and then navigate to the **YAML** tab to access the YAML editor.

If you prefer the command line, then access the terminal on the `workstation` machine and use the `oc edit` command to edit the `hello-web` VM configuration.

```
[student@workstation ~]$ oc edit vm hello-web -n web-servers
```

2.2. Add the `app: helloworld` label under the `.spec.template.metadata.labels` path. Remember that this is the `labels` section that Kubernetes uses 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...
```

If you use the OpenShift web console, then click **Save** when done. If you use the command line, then save the file and exit the editor.

2.3. Restart the VM so that Kubernetes recreates an instance that includes the new label. From the OpenShift web console, click **Actions** > **Restart**.

If you prefer the command line, then delete the `VirtualMachineInstance` (VMI) resource named `hello-web`. The command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi hello-web -n web-servers
virtualmachineinstance.kubevirt.io "hello-web" deleted
```

2.4. From the terminal on the `workstation` machine, confirm that Kubernetes recreates the `VMI` resource and the `virt-launcher` pod. You might have to run the command several times because it takes time for the pod to start.

```
[student@workstation ~]$ oc get vmi,pods -n web-servers
NAME                                          AGE PHASE   IP        NODENAME READY
virtualmachineinstance.kubevirt.io/hello-web 89s Running 10.8.2.33 worker02 True

NAME                                 READY   STATUS    RESTARTS   AGE
pod/virt-launcher-hello-web-ldr5b    1/1     Running   0          89s
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3. Create a service named `helloworld` of the `ClusterIP` type. Use the `app: helloworld` label for the selector. The `helloworld` service must accept traffic on TCP port 80 and forward it to the VM on port 80.

3.1. Create the `helloworld-svc.yaml` resource definition file with the following content.

> **Note**
> If you prefer, you can also use the web console to create the service.
>
> To create the service, navigate to **Networking** > **Services**, select the `web-servers` project, and then click **Create Service**. Complete the YAML file with the following content and then click **Create**.

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
  namespace: web-servers
spec:
  type: ClusterIP
  selector:
    app: helloworld
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

The `lab` command has prepared the `/home/student/DO316/labs/network-external/helloworld-svc.yaml` file so that you can compare it with your version.

3.2. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f helloworld-svc.yaml
service/helloworld created
```

3.3. Confirm that the `helloworld` service has an active endpoint.

```
[student@workstation ~]$ oc get endpoints -n web-servers
NAME         ENDPOINTS      AGE
helloworld   10.8.2.33:80   46s
```

4. Use the OpenShift web console to create a route for the `helloworld` service.

> **Note**
> You can also run the `oc expose service/helloworld -n web-servers` command from the command line to create the route.

4.1. From the web console, navigate to **Networking** > **Routes**.

4.2. Select the `web-servers` project and then click **Create Route**.

4.3. Complete the **Create Route** form using the following information.

Field	Value
Name	`helloworld`
Service	`helloworld`
Target port	`80 → 80(TCP)`

4.4. Click **Create** to create the route. The **Route details** page displays.

4.5. Click the link under **Location** to confirm that you can access the web application running on the VM.

The web application displays an `Hello world` page in a new web browser tab.

4.6. From a terminal, use the `curl` command to confirm that you can reach the web application.

```
[student@workstation ~]$ oc get route -n web-servers
NAME       HOST/PORT                                    PATH  SERVICES   PORT ...
helloworld helloworld-web-servers.apps.ocp4.example.com       helloworld 80 ...
[student@workstation ~]$ curl helloworld-web-servers.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>
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 network-external
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Configuring Kubernetes Networking for Virtual Machines

Configure services, routes, and network policies to expose external access for a web application and its database that is hosted on another virtual machine in the same project.

Outcomes

- Create services of the `ClusterIP` type
- Link services to virtual machines
- Create network policies to protect virtual machines
- Create routes

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `network-review` namespace and starts two virtual machines in that namespace: the `mariadb-server` VM hosts a MariaDB database, and the `front-web` VM hosts a web application that uses the database.

```
[student@workstation ~]$ lab start network-review
```

Instructions

1. From a terminal on the `workstation` machine, use the `oc` command to log in to your Red Hat OpenShift cluster as `admin` using `redhat` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`.

   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 two VMs are running under the `network-review` project. If you need to log in to the console of the VMs during the exercise, then log in as `root` using `redhat` as the password.

2. Create a service of the `ClusterIP` type that allows access to the database running inside the `mariadb-server` VM.

   To create this service, add the `tier: backend` label to the `virt-launcher` pod associated with the `mariadb-server` VM.

   Create the service using `database` as the name. This service must listen on TCP port 3306 and forward the traffic to port 3306 to the `tier: backend` pod.

The `lab` command has prepared some examples of resource files in the `/home/student/DO316/labs/network-review/` directory. If you choose to use those files, then you need to adapt them to the exercise requirements.

To verify your work, run the `/home/student/DO316/labs/network-review/testdb.sh` script. The script tries to connect to the database through the service.

3. Create a network policy to protect the `mariadb-server` VM. The policy must only allow the ingress connections to TCP port 3306 from the `network-review` namespace.

   To create this policy, add the `allowed: database` label to the `network-review` namespace.

   Create the network policy using `allow-database` as the name. 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 verify your work, run the `/home/student/DO316/labs/network-review/testdb.sh` script. The script accepts a namespace as a parameter. The script must succeed when you run `~/DO316/labs/network-review/testdb.sh network-review` and it must fail when you run `~/DO316/labs/network-review/testdb.sh default`.

4. Create a service of the `ClusterIP` type that allows access to the web application running inside the `front-web` VM.

   To create this service, add the `tier: frontend` label to the `virt-launcher` pod associated with the `front-web` VM.

   Create the service using `web` as the name. 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 `/home/student/DO316/labs/network-review/testweb.sh` script. The script tries to connect to the web application through the service.

5. Create a route so that you can access the web application running inside the `front-web` VM by using the `http://intranet-dev.apps.ocp4.example.com` URL.

   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.

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

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 network-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Configuring Kubernetes Networking for Virtual Machines

Configure services, routes, and network policies to expose external access for a web application and its database that is hosted on another virtual machine in the same project.

Outcomes

- Create services of the `ClusterIP` type
- Link services to virtual machines
- Create network policies to protect virtual machines
- Create routes

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `network-review` namespace and starts two virtual machines in that namespace: the `mariadb-server` VM hosts a MariaDB database, and the `front-web` VM hosts a web application that uses the database.

```
[student@workstation ~]$ lab start network-review
```

Instructions

1. From a terminal on the `workstation` machine, use the `oc` command to log in to your Red Hat OpenShift cluster as `admin` using `redhat` as the password. The OpenShift cluster API endpoint is `https://api.ocp4.example.com:6443`.

   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 two VMs are running under the `network-review` project. If you need to log in to the console of the VMs during the exercise, then log in as `root` using `redhat` as the password.

   1.1. From a terminal, log in to your OpenShift cluster as `admin`.

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


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 the `redhat` password when prompted.

1.3. Navigate to **Virtualization** > **VirtualMachines** and then select the `network-review` project. Confirm that the `mariadb-server` and the `front-web` VMs are running.

2. Create a service of the `ClusterIP` type that allows access to the database running inside the `mariadb-server` VM.

To create this service, add the `tier: backend` label to the `virt-launcher` pod associated with the `mariadb-server` VM.

Create the service using `database` as the name. This service must listen on TCP port 3306 and forward the traffic to port 3306 to the `tier: backend` pod.

The `lab` command has prepared some examples of resource files in the `/home/student/DO316/labs/network-review/` directory. If you choose to use those files, then you need to adapt them to the exercise requirements.

To verify your work, run the `/home/student/DO316/labs/network-review/testdb.sh` script. The script tries to connect to the database through the service.

2.1. From the terminal on the `workstation` machine, use the `oc edit` command to edit the configuration of the `mariadb-server` VM.

```
[student@workstation ~]$ oc edit vm mariadb-server -n network-review
```

2.2. Add the `tier: backend` label under the `.spec.template.metadata.labels` path.

```
...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...
```

2.3. Delete the `VirtualMachineInstance` (VMI) resource named `mariadb-server` so that Kubernetes recreates an instance that includes the new label. The command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi mariadb-server -n network-review
virtualmachineinstance.kubevirt.io "mariadb-server" deleted
```

2.4. Confirm that Kubernetes recreates the VMI resource and the `virt-launcher` pod.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get vmi,pods -n network-review
NAME                                                AGE  PHASE    IP          ...
virtualmachineinstance.kubevirt.io/front-web        40m  Running  10.8.2.31   ...
virtualmachineinstance.kubevirt.io/mariadb-server   71s  Running  10.11.0.41  ...

NAME                                       READY   STATUS    RESTARTS   AGE
pod/virt-launcher-front-web-rljdn          1/1     Running   0          40m
pod/virt-launcher-mariadb-server-jbfh5     1/1     Running   0          71s
```

2.5. Create the `mariadb.yaml` resource definition file with the following content.

> **Note**
> If you prefer, you can also use the web console to create the service.
>
> To do so, navigate to **Networking** > **Services**, select the `network-review` project, 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
```

2.6. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f mariadb.yaml
service/database created
```

2.7. Confirm that the `database` service has an active endpoint.

```
[student@workstation ~]$ oc get endpoints -n network-review
NAME       ENDPOINTS          AGE
database   10.11.0.41:3306    9s
```

2.8. Run the `testdb.sh` script to verify your work.

```
[student@workstation ~]$ ~/DO316/labs/network-review/testdb.sh
Testing database connection from network-review (be patient)...
  Connection successful
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3. Create a network policy to protect the `mariadb-server` VM. The policy must only allow the ingress connections to TCP port 3306 from the `network-review` namespace.

To create this policy, add the `allowed: database` label to the `network-review` namespace.

Create the network policy using `allow-database` as the name. 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 verify your work, run the `/home/student/DO316/labs/network-review/testdb.sh` script. The script accepts a namespace as a parameter. The script must succeed when you run `~/DO316/labs/network-review/testdb.sh network-review` and it must fail when you run `~/DO316/labs/network-review/testdb.sh default`.

3.1. 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. Create the `policy.yaml` resource definition file with the following content:

```
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
```

3.3. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f policy.yaml
networkpolicy.networking.k8s.io/allow-database created
```

3.4. Run the `testdb.sh` script to verify your work. The script succeeds when you use the `network-review` parameter. It fails when you use the `default` namespace.

```
[student@workstation ~]$ ~/DO316/labs/network-review/testdb.sh network-review
Testing database connection from network-review (be patient)...
  Connection successful
[student@workstation ~]$ ~/DO316/labs/network-review/testdb.sh default
Testing database connection from default (be patient)...
  Cannot connect
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

4. Create a service of the `ClusterIP` type that allows access to the web application running inside the `front-web` VM.

To create this service, add the `tier: frontend` label to the `virt-launcher` pod associated with the `front-web` VM.

Create the service using `web` as the name. 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 `/home/student/DO316/labs/network-review/testweb.sh` script. The script tries to connect to the web application through the service.

4.1. Use the `oc edit` command to edit the configuration of the `front-web` VM.

```
[student@workstation ~]$ oc edit vm front-web -n network-review
```

4.2. Add the `tier: frontend` label under the `.spec.template.metadata.labels` path.

```
...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...
```

4.3. Delete the `VirtualMachineInstance` (VMI) resource named `front-web` so that Kubernetes recreates an instance that includes the new label. The command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi front-web -n network-review
virtualmachineinstance.kubevirt.io "front-web" deleted
```

4.4. Confirm that Kubernetes recreates the VMI resource and the `virt-launcher` pod.

```
[student@workstation ~]$ oc get vmi,pods -n network-review
NAME                                              AGE   PHASE    IP          ...
virtualmachineinstance.kubevirt.io/front-web      21s   Running  10.8.2.33   ...
virtualmachineinstance.kubevirt.io/mariadb-server 132m  Running  10.11.0.41  ...

NAME                                      READY   STATUS    RESTARTS   AGE
pod/virt-launcher-front-web-rw4vp         1/1     Running   0          21s
pod/virt-launcher-mariadb-server-jbfh5    1/1     Running   0          132m
```

4.5. Create the `web.yaml` resource definition file with the following content.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Note

If you prefer, you can also use the web console to create the service.

To do so, navigate to **Networking** > **Services**, select the `network-review` project, 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
```

4.6. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f web.yaml
service/web created
```

4.7. Confirm that the `web` service has an active endpoint.

```
[student@workstation ~]$ oc get endpoints -n network-review
NAME       ENDPOINTS         AGE
database   10.11.0.41:3306   44m
web        10.8.2.33:80      22s
```

4.8. Run the `testweb.sh` script to verify your work.

```
[student@workstation ~]$ ~/DO316/labs/network-review/testweb.sh
Testing web application from network-review (be patient)...
  Connection successful
```

5. Create a route so that you can access the web application running inside the `front-web` VM by using the `http://intranet-dev.apps.ocp4.example.com` URL.

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. Expose the web service. Use the `--hostname` option to force the hostname. Otherwise, Red Hat OpenShift sets it to `web-network.review.apps.ocp4.example.com`.

```
[student@workstation ~]$ oc expose service web \
  --hostname intranet-dev.apps.ocp4.example.com -n network-review
route.route.openshift.io/web exposed
```

5.2. Use the `curl` command from the `workstation` machine to 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>
```

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

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 network-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- Pods get an IP address on the pod SDN. Kubernetes assigns a different IP address each time it starts a pod.
- Services assign a unique IP address to a set of pods. A service identifies its pods by using labels.
- 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 only accessible 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 one 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 allow 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 routes features to make them available on the network.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 4

Connecting Virtual Machines to External Networks

Goal

Configure node networking to connect virtual machines and nodes to networks that are outside the cluster.

Objectives

- Describe the Multus CNI plug-in and use cases.
- Configure multihomed nodes and virtual machines using the NMstate operator and Multus.

Sections

- Introducing Multus (and Quiz)
- Configuring Multihomed Nodes and Virtual Machines (and Guided Exercise)

Lab

- Connecting Virtual Machines to External Networks

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Introducing Multus

Objectives

- Describe the Multus CNI plug-in and use cases.

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 host ports are no longer necessary. The Kubernetes network model imposes some requirements on any networking implementation:

- Pods in a cluster can communicate without *network address translation (NAT)*.
- Pods and agents (running processes) can communicate when they are located 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 blind to the existence of host ports. However, you can request host ports on a node that forward to a pod.

> **Note**
> For more information about Kubernetes Networking, refer to the *Kubernetes Networking* documentation at https://kubernetes.io/docs/concepts/services-networking/#the-kubernetes-network-model

Multus in Red Hat OpenShift Virtualization

The Kubernetes networking model can be implemented in a number of ways, including through *custom resources (CR)* and *Container Network Interface (CNI)* plug-ins. The *Multus CNI* plug-in acts similar to a wrapper by calling other CNI plug-ins for advanced networking functionalities, such as attaching multiple network interfaces to pods in an OpenShift cluster. *Red Hat OpenShift Container Platform (RHOCP)* 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 virtual machine (VM) creates a *multihomed* VM.

Although you can add an additional 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 attached to a pod by using the `oc exec -it` *`pod_name`* `– ip a` command. Pods with additional network interfaces that use the Multus CNI plug-in follow the *net n* naming convention.

The following figure shows how additional network interfaces, attached by the Multus CNI plug-in, coexist within a pod:

Figure 4.1: Pod additional interfaces 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 that are outside of the cluster.

Implementing Multus

OpenShift Virtualization uses *Custom Resource Definitions (CRDs)* and CNI plug-ins, such as Multus CNI, to provide advanced networking functionality for your VMs. In RHOCP, additional network configurations for VMs based on the Multus CNI plug-in are managed by the *Cluster Network Addons Operator (CNAO)*. The Multus CNI plug-in is implemented through the `NetworkAttachmentDefinition` CR, with development led by the Network Plumbing Working Group.

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 the management of 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 VM, you must create a network attachment definition that defines the CNI plug-in that will create the additional interface on the resource.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

RHOCP provides the following CNI plug-ins for the Multus CNI plug-in to chain:

bridge
: Configure an additional bridge-based network to allow pods on the same host to communicate with each other and the host. A Linux bridge is required to attach VMs to multiple networks.

host-device
: Configure an additional host-device network to allow pods access to a physical Ethernet network device on the host system.

ipvlan
: Configure an additional ipvlan-based network to allow pods on a host to have different IP addresses with the same MAC address as the host's physical device.

macvlan
: Configure an additional macvlan-based network to allow pods on a host to communicate with other hosts and their pods by using a physical network interface. Unlike an ipvlan-based network, each pod that is attached to a macvlan additional network has a unique MAC address.

SR-IOV
: Configure an additional SR-IOV network to allow pods to attach to a virtual function (VF) interface on SR-IOV capable hardware on the host system.

These plug-ins require satisfying any limitations or requirements for the additional network. Therefore, the installation of each plug-in varies in terms of configuration. Each plug-in requires a network attachment definition that defines the configuration. Before configuring 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 greater detail elsewhere in this course.

You can create network attachment definitions for VMs through the command-line with a YAML manifest or through the OpenShift web console.

For example, to configure a network attachment definition through 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 Linux bridge network attachment definition:

Figure 4.2: Configuring a Linux bridge network attachment definition

> **Note**
> If you have VLAN IDs configured on your additional network, you can specify the ID numbers in the **VLAN Tag Number** field. Otherwise, you can leave the field empty.

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)
    "macspoofchk": true,  (6)
    "vlan": 0  (7)
  }'
```

1. The name for the network attachment definition.
2. Optional: Annotation key-value pair to determine node selection for VM scheduling and execution. In this example, `bridge-dev` is the name of the bridge configured on some cluster nodes. VMs will not run on a node that is not configured with the `bridge-dev` bridge.
3. The name for the configuration, which should match the name of the network attachment definition.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

4. The name of the CNI plug-in to use. Do not change this field unless you want to use a different CNI plug-in.
5. The name of the Linux bridge 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 you 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 network attachment definition creation 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 OpenShift Virtualization, refer to the *Red Hat OpenShift Virtualization* documentation at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Quiz

Introducing Multus

Choose the correct answers to the following questions:

1. Which of the following requirements are imposed by the Kubernetes network model?

a. Communication between pods requires a NAT.
b. Agents on a node can communicate with all pods on the same node.
c. Links are required to allow communication between pods in the cluster.

2. Which two of the following options are names of additional network interfaces attached to pods? (Choose two.)

a. `eth0`
b. `net1`
c. `eth1`
d. `net2`

3. You can connect pods and virtual machines to networks that are external to the cluster.

a. True
b. False

4. Which of the following statements about additional network interfaces and the Multus CNI plug-in is true?

a. Additional network configurations for VMs based on the Multus CNI plug-in are managed by the Cluster Network Addons Operator.
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 above.

5. Which of the following plug-ins does the Multus CNI plug-in chain to connect a VM to multiple networks?

a. ip-vlan
b. host-device
c. bridge
d. mac-vlan

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

6. Which of the following statements represents a security use case?

a. You can increase the performance of a network-intensive workload by isolating data and control planes to different node interfaces.

b. You can use a separate network to isolate private data that must not be shared between tenants or customers.

c. You can connect pods and virtual machines to networks that are external to the cluster.

Solution

Introducing Multus

Choose the correct answers to the following questions:

1. Which of the following requirements are imposed by the Kubernetes network model?

a. Communication between pods requires a NAT.
b. Agents on a node can communicate with all pods on the same node.
c. Links are required to allow communication between pods in the cluster.

2. Which two of the following options are names of additional network interfaces attached to pods? (Choose two.)

a. `eth0`
b. `net1`
c. `eth1`
d. `net2`

3. You can connect pods and virtual machines to networks that are external to the cluster.

a. True
b. False

4. Which of the following statements about additional network interfaces and the Multus CNI plug-in is true?

a. Additional network configurations for VMs based on the Multus CNI plug-in are managed by the Cluster Network Addons Operator.
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 above.

5. Which of the following plug-ins does the Multus CNI plug-in chain to connect a VM to multiple networks?

a. ip-vlan
b. host-device
c. bridge
d. mac-vlan

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

6. Which of the following statements represents a security use case?

a. You can increase the performance of a network-intensive workload by isolating data and control planes to different node interfaces.

b. You can use a separate network to isolate private data that must not be shared between tenants or customers.

c. You can connect pods and virtual machines to networks that are external to the cluster.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Configuring Multihomed Nodes and Virtual Machines

Objectives

- Configure multihomed nodes and virtual machines using the NMstate operator and Multus.

The Kubernetes NMState Operator

Red Hat OpenShift Virtualization uses the *Kubernetes NMState Operator* to report on and configure node networking in a declarative way. The Kubernetes NMstate Operator provides the components for declarative node networking in a Red Hat OpenShift cluster.

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:
  labels:
    app.kubernetes.io/component: network
    app.kubernetes.io/managed-by: cnao-operator
    app.kubernetes.io/part-of: hyperconverged-cluster
    app.kubernetes.io/version: 4.10.0
    nmstate.io/force-nns-refresh: "1649170623184016261"
  name: worker01 ...output omitted...
status:
  currentState:
    interfaces:
    - ipv4:
        address:
        - ip: 192.168.50.13
          prefix-length: 24
        auto-dns: true
        auto-gateway: true
        auto-route-table-id: 0
        auto-routes: true
        dhcp: true
        enabled: true
      ipv6:
        address:
        - ip: fe80::5054:ff:fe00:320d
          prefix-length: 64
        auto-dns: true
        auto-gateway: true
        auto-route-table-id: 0
```

```
          auto-routes: true
          autoconf: true
          dhcp: true
          enabled: true
        lldp:
          enabled: false
        mac-address: 52:54:00:00:32:0D
        mtu: 8192
        name: ens3
        state: up
        type: ethernet
...output omitted...
```

Node Network Configuration Policy

A *node network configuration policy (NNCP)* resource is a YAML manifest that describes the desired network configuration for Red Hat OpenShift nodes. You can create and manage node network interfaces, such as declaring a *Linux bridge*, with a node network configuration policy. Red Hat OpenShift Virtualization supports the management of Linux bridge, VLAN, Ethernet, and bond interface types. You can also configure additional options, such as *Spanning Tree Protocol (STP)*, and IPv4 connectivity with an NNCP.

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 configured with the bridge or bonding interface is placed in a `down` state and connectivity is lost. To avoid losing connectivity, declare your desired settings for the node interface in the same NNCP that removes the bridge or bonding interface.

> **Note**
> Setting an interface to `absent` or deleting an NNCP does not restore the previous configuration. A cluster administrator must define a policy with the previous configuration to restore settings.

To apply an NNCP to your cluster, use the `oc create -f nncp-name.yaml` command. By default, an NNCP resource is applied to all nodes in the cluster. However, you can specify to which nodes to apply the policy, such as only worker nodes, by including the `spec: nodeSelector` parameter and the appropriate key value in the NNCP YAML manifest.

You can list existing policies in your OpenShift cluster with the `oc get nncp` command. To view the details of an existing NNCP, use the `oc get nncp nncp-name -o yaml` command.

The following example shows an NNCP that defines a Linux bridge on a node's `ens4` network interface.

```
apiVersion: v1
items:
- apiVersion: nmstate.io/v1
  kind: NodeNetworkConfigurationPolicy
  metadata:
    name: br1-policy (1)
  spec:
    nodeSelector: (2)
      node-role.kubernetes.io/worker: "" (3)
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
  desiredState:
    interfaces:
    - name: br1 ❹
      description: Linux bridge with ens4 as a port ❺
      type: linux-bridge ❻
      state: up
      ipv4:
        dhcp: true
        enabled: true
      bridge:
        options:
          stp:
            enabled: false
        port:
        - name: ens4 ❼
```

❶ The name for the node network configuration policy.

❷ Optional: If the `nodeSelector` parameter is not included, the policy applies to all nodes in the cluster.

❸ The `nodeSelector` specifies that this policy applies to all worker nodes in the cluster.

❹ The `name` parameter specifies the desired name for the configured interface.

❺ Optional: A description for the policy.

❻ The `type` parameter specifies the type of connection to create.

❼ The physical interface on the node to use in the defined Linux bridge.

Node Network Configuration Enactment

When you apply a node network configuration policy to a cluster, *Red Hat OpenShift Container Platform (RHOCP)* generates a *NodeNetworkingConfigurationEnactment (NNCE)* object for each cluster node impacted by the policy. The NNCE object reports the execution status of an applied policy and includes the defined settings enacted on each node by the policy. 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 an NNCE policy and to confirm that it is successfully applied. To view more detailed information, including desired settings and traceback data, use the `oc get nnce` *`nnce-name`* `-o yaml` command.

Multus CNI and the Kubernetes NMState Operator in OpenShift Virtualization

By default, virtual machines (VMs) are only connected to the pod network and can only communicate with and access other cluster resources. If a VM requires access to resources outside of the cluster, you must connect the VM to an additional network with access to those resources. VMs 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 functionality of a network switch.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Connect Virtual Machines to Additional Networks with a Linux Bridge

When a Linux bridge is configured on nodes, such as through a node network configuration policy, you must create a `NetworkAttachmentDefinition` resource before you can connect your VMs to the bridge. A network attachment definition is a Multus CNI *custom resource definition (CRD)* used to connect pods and VMs to one or more networks. Network attachment definitions are namespaced objects that define the Multus CNI plug-in to use in order to connect to an additional network.

A network attachment definition must specify the `cnv-bridge` Multus CNI plug-in to connect a VM to an additional network on the Linux bridge. You must create the resource in the VM's namespace. To make the resource available to all namespaces, create the network attachment definition within the `default` namespace.

Figure 4.3: Linux bridge node networking with Multus and Kubernetes NMState

Create Network Attachment Definitions from the Command Line

You can create a network attachment definition by applying a YAML manifest through the `oc` command-line client. The network attachment definition must include the name for the configuration, the type of Multus CNI plug-in used to connect to the network, and the name of the Linux bridge interface on the configured nodes.

The following example shows a network attachment definition YAML manifest for a Linux bridge.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: br1-bridge-network (1)
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "br1-bridge-network", (2)
```

```
    "type": "cnv-bridge", ❸
    "bridge": "br1", ❹
    "macspoofchk": true, ❺
    "vlan": 1 ❻
  }'
```

❶ The name of the `NetworkAttachmentDefinition` object.

❷ The name for the configuration.

❸ The actual name of the CNI plug-in to use.

❹ The name of Linux bridge configured on the nodes.

❺ Optional: If set to `true`, the MAC address for an interface cannot be changed and only a single MAC address exits the pod. Enabling this setting can provide security against a MAC address spoofing attack. This option is only available if a node supports *nftables* and has the `nft` binary deployed.

❻ Optional: The VLAN tag for the network.

> **Note**
> Red Hat recommends matching the configuration name to the name of the network attachment definition.

After defining the network attachment definition manifest, you can use the `oc create -f network-attachment-definition.yaml -n namespace` command to create the resource.

Create Network Attachment Definitions in the OpenShift Web Console

You can also use the OpenShift web console to create a network attachment definition. Within the web console, navigate to **Networking** > **NetworkAttachmentDefinitions**. Select your VM's namespace or the `default` namespace within the **Projects** list. Click **Create Network Attachment Definition** to open the **Create Network Attachment Definition** form. You can also click **Edit YAML** to create or paste in a the network attachment definition YAML manifest. Fill out the form with the appropriate configurations. Click **Create** to create the network attachment definition.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 4.4: Creating a network attachment definition

Attach a Secondary Interface on the Additional Network

To connect a VM to an additional network, you must attach a secondary network interface on the VM that specifies the network attachment resource.

To attach an additional interface to the VM, navigate to **Virtualization** > **Virtual Machines**. Click the **Network Interfaces** tab and then click **Add Network Interface**. Select the network attachment definition from the **Network** list and fill out the remaining form. Click **Add** to save and attach the interface to the VM. If your VM is in a `running` state, you must restart the VM to complete the process.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 4.5: Attaching a secondary network interface

You can also edit a stopped VM's manifest by using the `oc edit` command. The bridge interface must be added to the `spec.template.spec.domain.device.interfaces` list. You must also specify the network attachment definition 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)
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
          multus:
            networkName: br1-bridge-network ❸
...output omitted...
```

❶ The name of the secondary interface connected to the bridge network.

❷ The name of network. This value must match the value of the `spec.template.spec.domain.devices.interfaces` object.

❸ The name of the network attachment definition.

Configure 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, you must attach a secondary network interface to your VM that is connected to a bridge network. A DHCP server must be available on the bridge network.

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 within 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:
...
  volumes:
  - cloudInitNoCloud:
      networkData: |
        version: 2
        ethernets:
          eth1: ❶
            addresses:
            - 192.168.51.150/24 ❷
```

❶ The name of the network interface.

❷ The static IP address.

The following example declares that DHCP creates a dynamic address on the `eth1` network interface.

```
kind: VirtualMachine
spec:
...
  volumes:
  - cloudInitNoCloud:
      networkData: |
        version: 2
        ethernets:
          eth1: ❶
            dhcp4: true ❷
```

❶ The name of the network interface.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. When set to `true`, the VM uses the network's DHCP server to provision an IP address.

References

For more information about the NMState operator, refer to the *About nmstate* section in the *Red Hat OpenShift Container Platform Virtualization* guide at https://docs.openshift.com/container-platform/4.10/virt/node_network/virt-observing-node-network-state.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Configuring Multihomed Nodes and Virtual Machines

Configure a multihomed VM with a static IP address to access an external database residing on the additional network.

Outcomes

- Export current network configuration for the cluster nodes.
- Create and apply a `NodeNetworkConfigurationPolicy` manifest to create a Linux bridge on the worker nodes.
- Observe the policy enactment on the worker nodes.
- Configure a secondary interface on a VM to use the Linux bridge and a custom MAC address.
- Confirm that the VM is connected to the 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 system for this exercise.

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

```
[student@workstation ~]$ lab start multihomed-nmstate
```

Instructions

A developer's VM needs access to a database that exists outside of your Red Hat OpenShift Container Platform cluster. The database resides on `server.srv.example.com` and is configured to accept incoming connections from the user, `devuser`, from a specific IP address on the `192.168.51.0/24` network only.

The cluster nodes, `worker01` and `worker02`, each have an additional interface, `ens4`, on the `192.168.51.0/24` network. Use the Kubernetes NMState Operator CRDs to configure `worker01` and `worker02` as multihomed nodes with an L2 Linux bridge on the `ens4` interface.

After creating the Linux bridge on the worker nodes, create a Multus CRD to enable VMs within the `developer-vms` project to use the Linux bridge. Next, configure the developer's VM to use the Linux bridge 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 Linux bridge network.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1. Log in to the OpenShift cluster as the `admin` user and apply the label, `external-network=true` to the `worker01` and `worker02` nodes. You use this label as a node selector in later steps.

   1.1. Use the terminal to log in to your OpenShift cluster as the `admin` user.

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

   1.2. Use the `oc label` command to label the `worker01` and `worker02` nodes with `external-network=true`.

```
[student@workstation ~]$ oc label node worker01 external-network=true
node/worker01 labeled
[student@workstation ~]$ oc label node worker02 external-network=true
node/worker02 labeled
```

2. Use the NMState Operator CRD, *node network state (NNS)* to observe and then back up the current node networking configuration of `worker01` and `worker02` 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.

   2.1. Use the `oc get nns` *`node`* `-o yaml` command to view a YAML output of a node's current network configuration. The `ens4` network interface is connected to the same network as the database server but is not configured with an IP address.

```
student@workstation ~]$ oc get nns worker01 -o yaml
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkState
...output omitted...
status:
...output omitted...
    interfaces:
...output omitted...
    - ipv4:
        address: []
        enabled: false
      ipv6:
        address: []
        enabled: false
      lldp:
        enabled: false
      mac-address: 52:54:00:00:33:0D
      mtu: 1500
      name: ens4
      state: down
      type: ethernet
...output omitted…
[student@workstation ~]$ oc get nns worker02 -o yaml
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkState
status:
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
...output omitted...
    interfaces:
...output omitted...
    - ipv4:
        address: []
        enabled: false
      ipv6:
        address: []
        enabled: false
      lldp:
        enabled: false
      mac-address: 52:54:00:00:33:0E
      mtu: 1500
      name: ens4
      state: down
      type: ethernet
...output omitted...
```

2.2. Create a backup of the current configuration of `worker01` and `worker02` by redirecting the output of the `oc get nns` command to a YAML file.

```
[student@workstation ~]$ oc get nns worker01 -o yaml > worker01-nns-backup.yaml
[student@workstation ~]$ oc get nns worker02 -o yaml > worker02-nns-backup.yaml
```

3. Create and apply a *node network configuration policy (NNCP)* manifest to configure the `ens4` Ethernet interface as a Linux bridge, with `ens4` as the port. Modify and then apply the `br1-ens4-policy.yaml` file located in the `~/DO316/labs/multihomed-nmstate` directory to create the Linux bridge.

3.1. Open the `br1-ens4-policy.yaml` file located in the `~/DO316/labs/multihomed-nmstate` directory with your preferred text editor.

```
[student@workstation ~]$ vi ~/DO316/labs/multihomed-nmstate/br1-ens4-policy.yaml
```

3.2. Modify the `<CHANGE_ME>` parameters to match the following content:

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
 name: br1-ens4-policy
spec:
 nodeSelector:
   external-network: "true"
 desiredState:
   interfaces:
     - name: br1
       description: Linux bridge with ens4 as a port
       type: linux-bridge
       state: up
       ipv4:
         dhcp: true
         enabled: true
       bridge:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        options:
          stp:
            enabled: false
        port:
          - name: ens4
```

3.3. Use the manifest to create the node network configuration policy.

```
[student@workstation ~]$ oc create -f \
  ~/DO316/labs/multihomed-nmstate/br1-ens4-policy.yaml
nodenetworkconfigurationpolicy.nmstate.io/br1-ens4-policy created
```

4. Monitor the progress of the configuration policy.

4.1. Use the `oc get nnce` command to observe the current *node network configuration enactment (NNCE)* status for the policy.

```
[student@workstation ~]$ oc get nnce
NAME                        STATUS
worker01.br1-ens4-policy    Pending
worker02.br1-ens4-policy    Progressing
```

4.2. Optional: View the status of a particular node's enactment, including any errors or failures, with the command, `oc get nnce node.br1-ens4-policy -o yaml`.

```
[student@workstation ~]$ oc get nnce worker01.br1-ens4-policy -o yaml
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationEnactment
metadata:
...output omitted...
status:
 conditions:
 - lastHeartbeatTime: "2021-11-23T20:34:18Z"
   lastTransitionTime: "2021-11-23T20:34:18Z"
   reason: SuccessfullyConfigured
...output omitted...
```

4.3. Confirm that the policy is available to the cluster with the `oc get nncp` and `oc get nnce` commands.

```
[student@workstation ~]$ oc get nncp
NAME               STATUS
br1-ens4-policy    Available
[student@workstation ~]$ oc get nnce
NAME                        STATUS
worker01.br1-ens4-policy    Available
worker02.br1-ens4-policy    Available
```

5. Observe and confirm the Linux bridge configuration of `worker01` and `worker02` with the `oc get nns node -o yaml` command.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get nns worker01 -o yaml
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkState
...output omitted...
status:
 currentState:
   interfaces:
...output omitted...
   - bridge:
       options:
         group-addr: 01:80:C2:00:00:00
...output omitted...
     description: Linux bridge with ens4 as a port
     ipv4:
       address:
       - ip: 192.168.51.13
         prefix-length: 24
       auto-dns: true
       auto-gateway: true
       auto-route-table-id: 0
       auto-routes: true
       dhcp: true
       enabled: true
     ipv6:
       address: []
       autoconf: false
       dhcp: false
       enabled: false
     lldp:
       enabled: false
     mac-address: 52:54:00:00:33:0D
     mtu: 1500
     name: br1
     state: up
     type: linux-bridge
...output omitted...
[student@workstation ~]$ oc get nns worker02 -o yaml
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkState
...output omitted...
status:
 currentState:
   interfaces:
...output omitted...
   - bridge:
       options:
         group-addr: 01:80:C2:00:00:00
...output omitted...
     description: Linux bridge with ens4 as a port
     ipv4:
       address:
       - ip: 192.168.51.14
         prefix-length: 24
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        auto-dns: true
        auto-gateway: true
        auto-route-table-id: 0
        auto-routes: true
        dhcp: true
        enabled: true
      ipv6:
        address: []
        autoconf: false
        dhcp: false
        enabled: false
      lldp:
        enabled: false
      mac-address: 52:54:00:00:33:0E
      mtu: 1500
      name: br1
      state: up
      type: linux-bridge
...output omitted...
```

The nodes `worker01` and `worker02` are now configured as multihomed nodes with an L2 Linux bridge.

6. Remember that Multus provides a CRD called a *network attachment definition* that enables VMs, VMIs, and pods to attach to one or more networks. Use the web console to create a network attachment definition for the Linux bridge in the `developer-vms` project.

6.1. Identify the URL for the OpenShift web console with the `oc whoami` command.

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

6.2. Open a web browser and navigate to https://console-openshift-console.apps.ocp4.example.com.

6.3. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

6.4. From the OpenShift web console left panel, navigate to **Networking** > **NetworkAttachmentDefinitions**.

6.5. Select the project, `developer-vms`, from the **Project** list and then click **Create network attachment definition**.

6.6. Complete the **Create Network Attachment Definition** form using the following information to create a network attachment definition for the bridge network.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Create Network Attachment Definition

Field	Value
Name	`br1-network`
Description	Linux bridge on nodes `"Worker01"` and `"Worker02"` with ens4 as a port
Network Type	CNV Linux bridge
Bridge Name	`br1`

6.7. Click **Create** to create the network attachment definition.

7. Configure the developer's VM to use the bridge 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 that resides in the Linux bridge network.

7.1. From the OpenShift web console left panel, navigate to **Virtualization** > **VirtualMachines**.

7.2. Select the `developer-vms` project from the **Projects** list. Click the **dev-dbaccess** VM to open the VM's **Overview** tab.

7.3. Click the **Network Interfaces** tab to display the network interfaces connected to the VM.

7.4. Click **Add Network Interface** to create a new interface on the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

7.5. On the **Add Network Interface** page, select **br1-network** from the **Network** field.

7.6. Confirm that the **Type** is set to `Bridge` and enter `00:00:00:00:00:01` in the **MAC Address** field. Click **Add** to save and attach the interface to the VM.

7.7. Restart the VM to apply the pending changes. Click **Actions** > **Restart**, and then click **Restart** to confirm the action.

8. Confirm that the VM's secondary network interface, `eth1`, is configured with a `192.168.51.10/24` static IP address.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

8.1. Click the **Console** tab, and then click **Open Console in New Window** to open the VM's console.

8.2. Click the console window and log in as the `developer` user with the `developer` password.

8.3. Use the `ip addr show` command to display the IP address configuration of the `eth1` network interface.

```
[developer@dev-dbaccess ~]$ ip addr show eth1 | grep inet
    inet 192.168.51.10/24 brd 192.168.51.255 scope global noprefixroute eth1
    inet6 fe80::169b:9d21:af93:afc7/64 scope link
```

9. 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.

9.1. Use the `mysql` command to connect to the MariaDB instance on the `server.srv.example.com` host. Log in as the `devuser` user with the `developer` 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 9
Server version: 10.3.27-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)]>
```

9.2. 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]>
```

9.3. Confirm that the `devuser` can view the `sakila` tables with the `SHOW TABLES;` command.

```
MariaDB [sakila]> SHOW TABLES;
+----------------------------+
| Tables_in_sakila           |
+----------------------------+
| actor                      |
| actor_info                 |
| address                    |
| category                   |
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
| city                          |
...output omitted...
| staff_list                    |
| store                         |
+-----------------------------+
23 rows in set (0.002 sec)
```

9.4. Exit the database with the `EXIT;` command.

```
MariaDB [sakila]> EXIT;
Bye
[developer@dev-dbaccess ~]$
```

The `dev-dbaccess` VM is now a multihomed VM with a static IP address . You have confirmed that the VM can connect to resources on the Linux bridge network, `192.168.51.0/24`.

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 multihomed-nmstate
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Connecting Virtual Machines to External Networks

Configure node networking to connect a VM to an additional network, and then connect the VM to an external web server residing on the additional network.

Outcomes

- Create a node network configuration policy that configures a Linux bridge with DHCP 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `multihomed-review` namespace and starts the `dev-external` virtual machine in that namespace.

```
[student@workstation ~]$ lab start multihomed-review
```

Instructions

The `dev-external` VM requires access to an Apache HTTP instance on `server.srv.example.com` that only accepts requests to resources residing on an external network. The VM needs an additional network interface connected to the external network to communicate with the Apache HTTP server. The `dev-external` VM exists within the `multihomed-review` project.

As the `admin` user, label the nodes `worker01` and `worker02` with the `external-network=true` label. Configure a Linux bridge named `br1` on the nodes `worker01` and `worker02`, and then create a network attachment definition named `br1-network` in the `multihomed-review` project.

Before attaching the second network interface, connect to `server.srv.example.com` on the `dev-external` VM and confirm that a warning is displayed regarding permissions to the Apache instance. After attaching the second interface, attach an additional network interface to the VM that uses the `br1-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 by using the `bridge-nncp.yml` file located in the `~/DO316/labs/multihomed-review` directory. Modify the file with the following information:

Parameter	Value
Name	`br1-ens4-policy`
Interface name	`br1`
Interface type	`linux-bridge`
Bridge port	`ens4`

3. Create a network attachment definition in the `multihomed-review` project by using the `br1-net-attach-def.yml` file located in the `~/DO316/labs/multihomed-review` directory. Modify the file according to the following requirements:

Parameter	Value
Name	`br1-network`
Namespace	`multihomed-review`
Bridge	`br1`

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.

5. Attach a secondary network interface to the `dev-external` VM with the following information:

Parameter	Value
Name	Leave the default name
Model	`virtio`
Network	`br1-network`
Type	`Bridge`
MAC Address	`00:00:00:00:00:02`

   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 residing in the Linux bridge network.

6. From 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ lab grade multihomed-review
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish multihomed-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Connecting Virtual Machines to External Networks

Configure node networking to connect a VM to an additional network, and then connect the VM to an external web server residing on the additional network.

Outcomes

- Create a node network configuration policy that configures a Linux bridge with DHCP 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `multihomed-review` namespace and starts the `dev-external` virtual machine in that namespace.

```
[student@workstation ~]$ lab start multihomed-review
```

Instructions

The `dev-external` VM requires access to an Apache HTTP instance on `server.srv.example.com` that only accepts requests to resources residing on an external network. The VM needs an additional network interface connected to the external network to communicate with the Apache HTTP server. The `dev-external` VM exists within the `multihomed-review` project.

As the `admin` user, label the nodes `worker01` and `worker02` with the `external-network=true` label. Configure a Linux bridge named `br1` on the nodes `worker01` and `worker02`, and then create a network attachment definition named `br1-network` in the `multihomed-review` project.

Before attaching the second network interface, connect to `server.srv.example.com` on the `dev-external` VM and confirm that a warning is displayed regarding permissions to the Apache instance. After attaching the second interface, attach an additional network interface to the VM that uses the `br1-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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1.1. Use the terminal to log in to the cluster as the `admin` user.

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

1.2. Add the label `external-network=true` to the `worker01` and `worker02` nodes.

```
[student@workstation ~]$ oc label node worker01 external-network=true
node/worker01 labeled
[student@workstation ~]$ oc label node worker02 external-network=true
node/worker02 labeled
```

1.3. Confirm that the `external-network=true` label is successfully applied to the nodes.

```
[student@workstation ~]$ oc describe node worker01 | grep external-network
                    external-network=true
[student@workstation ~]$ oc describe node worker02 | grep external-network
                    external-network=true
```

2. Create a Linux bridge network on the `worker01` and `worker02` nodes by using the `bridge-nncp.yml` file located in the `~/DO316/labs/multihomed-review` directory. Modify the file with the following information:

Parameter	Value
Name	`br1-ens4-policy`
Interface name	`br1`
Interface type	`linux-bridge`
Bridge port	`ens4`

2.1. Open the `bridge-nncp.yml` file located in the `~/DO316/labs/multihomed-review` directory with your preferred text editor.

```
[student@workstation ~]$ vim ~/DO316/labs/multihomed-review/bridge-nncp.yml
```

2.2. Modify the `<CHANGE_ME>` parameters according to the requested specifications. Save the changes.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
   name: br1-ens4-policy
spec:
  nodeSelector:
    external-network: "true"
  desiredState:
    interfaces:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
      - name: br1
        description: Linux bridge with ens4 as a port
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: ens4
```

2.3. Create the Linux bridge network with the `oc` command.

```
[student@workstation ~]$ oc apply -f \
  ~/DO316/labs/multihomed-review/bridge-nncp.yml
nodenetworkconfigurationpolicy.nmstate.io/br1-ens4-policy created
```

2.4. Confirm that the policy is successfully configured on the nodes by checking the node network configuration enactments with the `oc get nnce` command. You might need to run this command multiple times until each node has the `Available` status.

```
[student@workstation ~]$ oc get nnce
NAME                        STATUS
worker01.br1-ens4-policy    Available
worker02.br1-ens4-policy    Available
```

3. Create a network attachment definition in the `multihomed-review` project by using the `br1-net-attach-def.yml` file located in the `~/DO316/labs/multihomed-review` directory. Modify the file according to the following requirements:

Parameter	Value
Name	`br1-network`
Namespace	`multihomed-review`
Bridge	`br1`

3.1. Open the `br1-net-attach-def.yml` file located in the `~/DO316/labs/multihomed-review` directory with your preferred text editor.

```
[student@workstation ~]$ vim ~/DO316/labs/multihomed-review/br1-net-attach-def.yml
```

3.2. Modify the `<CHANGE_ME>` parameters according to specified requirements. Save the changes.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
  name: br1-network
  namespace: multihomed-review
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/br1
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "br1-network",
    "plugins": [
      {
        "type": "cnv-bridge",
        "bridge": "br1"
      },
      {
        "type": "cnv-tuning"
      }
    ]
  }'
```

3.3. Create the network attachment definition in the `multihomed-review` project.

```
[student@workstation ~]$ oc create -f \
  ~/DO316/labs/multihomed-review/br1-net-attach-def.yml
networkattachmentdefinition.k8s.cni.cncf.io/br1-network created
```

3.4. Confirm that the network attachment definition is successfully created in the project by using the `oc get network-attachment-definition` *`net-attachment-definition-name`* command.

```
[student@workstation ~]$ oc get network-attachment-definition br1-network \
  -n multihomed-review
NAME          AGE
br1-network   99s
```

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.

4.1. Open a web browser and navigate to https://console-openshift-console.apps.ocp4.example.com. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

4.2. If necessary, change to the **Administrator** perspective. Navigate to **Virtualization** > **VirtualMachines**. Select the `multihomed-review` project from the **Projects** list.

4.3. Click the `dev-external` VM and select the **Console** tab.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 4.9: Access the VM console

4.4. Click **Guest login credentials** to display the credentials of the VM and then use the credentials to log in to the VM.

Figure 4.10: Display login credentials

4.5. Test the connectivity to the Apache HTTP instance on `server.srv.example.com` with the `curl server.srv.example.com` command. Confirm that you receive a warning stating that you do not have permission to access the resource.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 4.11: Test connectivity

5. Attach a secondary network interface to the `dev-external` VM with the following information:

Parameter	Value
Name	Leave the default name
Model	`virtio`
Network	`br1-network`
Type	`Bridge`
MAC Address	`00:00:00:00:00:02`

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 residing in the Linux bridge network.

5.1. From the web console, navigate to **Virtualization** > **VirtualMachines**, click the vertical ellipsis to the right of the `dev-external` VM, click **Stop**, and then click **Stop** in the confirmation window that displays.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 4.12: Stop the VM

5.2. Click the **dev-external** VM to open the **Overview** tab for the VM. Click the **Network Interfaces** tab and click **Add Network Interface**. Modify the parameters to match the specifications from the table. Click **Add** to attach the interface.

5.3. Click **Actions** and then click **Start** to start the VM and apply the changes.

6. From 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 on to the VM.

6.2. Connect to the Apache HTTP instance on `server.srv.example.com` with the `curl server.srv.example.com` command. Confirm that you receive a `Hello, World!` response.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 4.13: Test connectivity

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

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish multihomed-review
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- To review the current network state for a particular node, use the `oc get nns/node-name -o yaml` command.
- An *Node Network Configuration Policy* resource is a YAML manifest that describes the desired network configuration for Red Hat OpenShift nodes and is applied to all nodes in the cluster by default.
- OpenShift Virtualization supports the management of Linux 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 Linux bridge that connects VMs to additional networks.
- The Cluster Network Addons Operator manages additional network configurations for VMs based on the Multus CNI plug-in.
- Pods with additional network interfaces that use the Multus CNI plug-in follow the *net n* naming convention.
- If the VM requires a static or dynamic IP address, you must attach a secondary network interface that is connected to a bridge network to your VM, and a DHCP server must be available on the bridge network.
- Use the `cloud-init` service to specify a static or dynamic IP address for a secondary interface.
- You can verify the network attachment definition in a namespace with the `oc get net-attach-def -n` *`namespace`* command.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 5

Configuring Kubernetes Storage for Virtual Machines

Goal

Manage storage and disks for VMs in Red Hat OpenShift.

Objectives

- Configure appropriate services for persistent VM storage.
- Attach disks to a VM.
- Connect a VM to external storage using Multus.

Sections

- Attaching Persistent Storage to Virtual Machines (and Guided Exercise)
- Attaching and Accessing Disks on Virtual Machines (and Guided Exercise)
- Connecting Virtual Machines to External Storage (and Guided Exercise)

Lab

- Configuring Storage for Virtual Machines

Attaching Persistent Storage to Virtual Machines

Objectives

- Configure appropriate services for persistent VM storage.

Introduce Container Storage Concepts

Containers have ephemeral storage by default. When Kubernetes destroys a pod, it deletes all the files and data associated with that pod from the container. You need a mechanism to preserve your applications' data because Kubernetes automatically destroys and recreates pods frequently, such as when you deploy a new version of your application or when a node goes down.

Kubernetes uses *persistent volumes (PV)* to preserve data across container executions. Inside the container, you mount a persistent volume into a directory that the application can use to store its data. For example, MariaDB Server stores its database data under the `/var/lib/mysql/data` directory. By mounting a volume in that directory, you preserve the database data across container executions.

Select a Storage Class

As a developer, you request a volume for your application by creating a Kubernetes *persistent volume claim (PVC)* resource. The main parameters of a PVC are the size and the access mode. Kubernetes processes your PVC, prepares a PV, and then binds it to the PVC.

The following command lists the PVCs in the project. The `logdata` PVC is assigned a 1 GiB PV. The `refdata` PVC has the `Pending` status because it is waiting for Kubernetes to prepare and assign a PV.

```
[user@host ~]$ oc get pvc
NAME     STATUS  VOLUME            CAPACITY  ACCESS MODES   STORAGECLASS   AGE
logdata  Bound   pvc-5980...167ac  1Gi       RWO            nfs-storage    19s
refdata  Pending                                            nfs-storage     2s
```

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, such as *Network File System (NFS)*, *Fibre Channel*, *Amazon Elastic Block Store (EBS)*, or *Red Hat Ceph Storage* services.

The cluster administrators expose these different 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 Container Platform (RHOCP)* 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

You can list the available storage classes on your cluster by using the `oc get StorageClass` command. You can also use the OpenShift web console. Navigate to **Storage** > **StorageClasses** to list the storage classes.

Figure 5.1: Retrieve the available storage classes

The preceding screen capture shows five storage classes. In this example, the cluster administrators created the `nfs-storage` class. The four other classes are from a *Red Hat OpenShift Data Foundation* installation. OpenShift Data Foundation is an additional product based on *Ceph* technologies. It can aggregate the disk space available on your cluster nodes as a resilient storage service.

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 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 creates PVs on demand. When a developer creates a PVC to get a volume of a specific storage class, Kubernetes uses the *provisioner* associated with the class to allocate the volume dynamically. A provisioner is a plug-in that communicates with the back-end storage to manage volumes.

Request a Persistent Volume

As a developer, you request a PV through a PVC. You can prepare a PVC by using a resource file in the YAML format. Run the `oc apply -f resourcefile.yaml` command to create the resource.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: dbdata
spec:
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
  storageClassName: ocs-external-storagecluster-ceph-rbd  (1)
  accessModes:
    - ReadWriteOnce  (2)
  resources:
    requests:
      storage: 10Gi  (3)
  volumeMode: Filesystem  (4)
```

1. The `storageClassName` parameter specifies the storage class to use.
2. The `accessModes` parameter indicates how containers can access the volume:
   - When set to `ReadWriteOnce`, you can mount the volume read/write on one container at a time. The OpenShift web console and the `oc get pvc` command report this mode as `RWO`.
   - When set to `ReadWriteMany`, you can mount the volume read/write on several containers simultaneously. The OpenShift web console and the `oc get pvc` command report this mode as `RWX`. Not all back-end storage supports this mode. VM live migration requires disks of the `RWX` type.
   - When set to `ReadOnlyMany`, you can simultaneously mount the volume read-only on several containers. The OpenShift web console and the `oc get pvc` command report this mode as `ROX`.
3. The `storage` parameter specifies the requested volume size.
4. The `volumeMode` parameter gives the volume type:
   - When set to `Filesystem`, the default, the returned volume has a file system that you can directly mount inside your container.
   - 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.

   Not all back-end storage supports both modes.

> **Note**
> Most applications running in containers store their data as files and directories. They consume volumes in the `Filesystem` mode.
>
> 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. Navigate to **Storage** > **PersistentVolumeClaims**, click **Create PersistentVolumeClaim** > **With Form**, and then complete the form.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 5.2: Create a PVC

Configure Persistent Storage 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 facilitate the process of creating the PVC with optimal parameters for VM disks and copying the disk image into the resulting PV:

`StorageProfile`
: For each storage class, a `StorageProfile` resource gives default values optimized for VM disks.

`DataVolume`
: A `DataVolume` resource describes a VM disk. It groups the PVC definition and the details of the disk image to inject into the PV.

List the Available 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 PVC default values best suited for VM disks.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The following example shows that the `ocs-external-storagecluster-ceph-rbd` storage profile provides default values for the PVCs created through the `ocs-external-storagecluster-ceph-rbd` storage class:

```
[user@host ~]$ oc describe storageprofile ocs-external-storagecluster-ceph-rbd
...output omitted...
Status:
  Claim Property Sets:
    Access Modes:
      ReadWriteMany
    Volume Mode:  Block
  Provisioner:    openshift-storage.rbd.csi.ceph.com
  Storage Class:  ocs-external-storagecluster-ceph-rbd
...output omitted...
```

The preceding storage class declares the `ReadWriteMany` access mode as default, which allows VM live migrations, and declares the `Block` volume mode as default, which provides better performances 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.

Create a Virtual Machine Disk by Using a Data Volume

`DataVolume` resources have two parts:

- The storage profile specification
- The source image details

The following example declares a data volume:

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: osdisk
spec:
  storage:  (1)
    resources:
      requests:
        storage: 10Gi
    storageClassName: ocs-external-storagecluster-ceph-rbd  (2)
  source:  (3)
    http:
      url: http://images..example.com/rhel-8.5-x86_64-kvm.qcow2
```

1. The `storage` section provides the details of the PVC to create. You only need to specify the disk size because OpenShift Virtualization retrieves the other parameters from the storage class.

2. The `storageClassName` parameter specifies the storage class of the PVC. OpenShift Virtualization uses the storage profile that has the same name to retrieve the default parameters for the PVC, such as the access and the volume modes.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3. The `source` section provides the disk image to inject into the PV. OpenShift Virtualization can import the image from a URL or by cloning another PVC. The following example of a `source` section instructs the data volume to create an empty disk:

```
source:
  blank: {}
```

When the PVC to create is in `Filesystem` mode, the data volume automatically instructs Kubernetes to create the PVC with a size 5.5% bigger 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.

If your cluster administrator set storage resource quotas for your project, then these quotas apply to the PVCs that you create through data volumes.

Deleting a `DataVolume` resource triggers the deletion of the associated PVC and PV. However, if the volume is attached to a VM, then the deletion only occurs after you have detached the volume.

Attach 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. Navigate to **Virtualization** > **VirtualMachines**, select the VM, and then navigate to the **Disks** tab. Click **Add disk** and complete the form.

Figure 5.3: Add a disk to a virtual machine

You can attach disks to running VMs. However, you might find that some options are unavailable. For example, when the VM is running, the only **Interface** option available is `scsi`, which might not provide the best performances compared to the `virtio` interface.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Use Advanced Virtualization Features

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 allows live migration, snapshots, and disk cloning.

Use Local Node Disk Space for Virtual Machine Disks

If your cluster installation does not provide a storage back end, then your administrators can configure the cluster to use the local disk space on the nodes as the back-end space for VM disks.

Your cluster administrators configure that solution through the Red Hat OpenShift Virtualization *hostpath* provisioner.

Note
You cannot use advanced features such as live migration when allocating your VMs' disks on the nodes' disk space.

References
For more information about RHOCP storage, refer to the *Red Hat OpenShift Container Platform Storage* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/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://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#virtual-machine-disks

For more information about Red Hat OpenShift Data Foundation, refer to the *Architecture* guide at
https://access.redhat.com/documentation/en-us/red_hat_openshift_data_foundation/4.10/html-single/red_hat_openshift_data_foundation_architecture/index

Red Hat OpenShift Data Foundation Data Sheet
https://www.redhat.com/en/resources/openshift-data-foundation-datasheet

Kubernetes Persistent Volumes
https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Kubernetes Disks and Volumes
https://kubevirt.io/user-guide/virtual_machines/disks_and_volumes/

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Attaching Persistent Storage to Virtual Machines

In your environment, NFS storage runs on slower, less expensive disks, and Red Hat OpenShift Data Foundation block storage runs on faster, more expensive backing devices. You need two new VMs for an upcoming project. The `vm1` database machine requires faster storage, whereas `vm2` is an internal-use file server that can use less expensive storage. In this exercise, you will connect these two VMs to the appropriate storage service that meets the requirements.

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 system for this exercise.

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` VM hosts a MariaDB database and the `vm2` VM 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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

   1.2. Navigate to **Virtualization** > **VirtualMachines**. Confirm that the `vm1` and `vm2` VMs are running.

2. Prepare a 5 GiB blank disk for the database server. Use the `ocs-external-storagecluster-ceph-rbd` storage class, which provides fast block storage. Attach the disk to the VM by using a `virtio` interface.

   2.1. Stop the VM 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` VM and then click **Actions** > **Stop** to stop it. Click **Stop** to confirm the operation. Wait for the machine to stop.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2.2. Navigate to the **Disks** tab and then click **Add disk**. Complete the form using the following information.

After you have completed the form, notice the `Access mode: ReadWriteMany / Volume mode: Block` message at the bottom of the form. This message confirms that the volume uses block storage.

> **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.

Click **Add** to create and attach the disk to the VM.

Field	Value
Source	`Blank (creates PVC)`
Name	`dbdata`
Size	5 GiB
Type	`Disk`
Interface	`virtio`
StorageClass	`ocs-external-storagecluster-ceph-rbd`

2.3. Click **Actions** > **Start** to start the VM.

2.4. Log in to the VM console and confirm that a new 5 GiB device is available. Navigate 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
```

2.5. Log out of the VM console.

```
[root@vm1 ~]# logout
```

3. Prepare a 10 GiB blank disk for the web server. Use the `nfs-storage` storage class. Attach the disk to the VM by using a `virtio` interface.

3.1. Navigate to **Virtualization** > **VirtualMachines** and then select the `vm2` VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3.2. Stop the VM so that you can use the `virtio` interface to attach the new disk. Click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

3.3. Navigate to the **Disks** tab and then click **Add disk**. Complete the form using the following information. At the bottom of the form, notice that the volume mode is set to `Filesystem` by default. Do not change that parameter.

Click **Add** after you have completed the form.

Field	Value
Source	`Blank (creates PVC)`
Name	`staticdata`
Size	10 GiB
Type	`Disk`
Interface	`virtio`
StorageClass	`nfs-storage`

3.4. Click **Actions** > **Start** to start the VM.

3.5. Log in to the VM console and confirm that a new 10 GiB device is available. Navigate 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
```

3.6. Log out of the VM console.

```
[root@vm2 ~]# logout
```

4. Detach and then delete the additional disk from the `vm2` VM.

4.1. Click **Actions** > **Stop** to stop the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

4.2. Navigate to the **Disks** tab. At the end of the row of the `staticdata` disk, click the vertical ellipsis menu and then click **Delete**. In the confirmation window, ensure that **Delete Data Volume and PVC** is selected and then click **Detach**.

4.3. Click **Actions** > **Start** to start the VM.

4.4. Log in to the VM console and confirm that the device does not exist anymore. Navigate 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 does not report the `vdc` device anymore.

```
[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
```

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 storage-intro
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Attaching and Accessing Disks on Virtual Machines

Objectives

- Attach disks to a VM.

Connect a Virtual Machine to Multiple Disks

You can attach several `DataVolume` 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 interface that you select when you attach the data volume:

- When you choose `scsi`, the disk shows as a standard SCSI device that most operating systems recognize. Linux systems give these disks a name in the `/dev/sd`*X* format.

- When you choose `virtio`, the operating system uses a driver that communicates with the virtualization layer for optimal performance. Some operating systems do not provide that driver by default. Linux systems include the driver and give these disks a name in the `/dev/vd`*X* format.

Use the OpenShift web console to create a data volume and attach it to a VM as an additional disk. Navigate to **Virtualization** > **VirtualMachines**, select the VM, and then navigate to the **Disks** tab. Click **Add disk** and then complete the form.

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.

Select a Data Volume Source

The source section of a `DataVolume` resource provides the details of the disk image to inject into the *persistent volume (PV)*. The following screen capture shows that you can select the source from the **Add disk** window when using the OpenShift web console.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 5.4: Select the data source

Blank (creates PVC)
: The new *Persistent Volume Claim (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.

Import via URL (creates PVC)
: Red Hat OpenShift Virtualization downloads the virtual disk image from the URL that you provide and then extracts it into the new volume. The virtual disk image must be in the raw format or 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 the changes you make across VM restarts.

Use an existing PVC
: OpenShift Virtualization attaches the PVC you provide as a disk. That PVC might come from a data volume 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 allows pods and VMs to concurrently access volumes in `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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Clone existing PVC (creates PVC)
: OpenShift Virtualization copies the PVC 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 it.

Import via Registry (creates PVC)
: OpenShift Virtualization downloads the given container image from a container registry. The container image contains a disk image in the raw or qcow2 format under the `/disk/` directory. OpenShift Virtualization extracts that disk image and copies it into the volume.

  This source option is similar to importing disks from URLs, but it is useful when you already have a container registry 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 only manage images, you must store your disk images in container images under the `/disk/` directory. These images are not meant to be run as containers, and they usually only contain the `/disk/` directory and the disk image.

Container (ephemeral)
: OpenShift Virtualization downloads the given 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 preceding option except that OpenShift Virtualization does not create `DataVolume`, PVC, or PV resources and therefore does not consume space on your back-end storage.

  You can mount the disk read/write into the VM, but the data you write are not persistent. Every time 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.

Detach a Disk from a Virtual Machine

You can only detach disks from stopped VMs.

After you have stopped the VM, navigate to the **Disks** tab, use the vertical ellipsis to the right of the disk that you want to detach, and then click **Delete**.

A message asks you whether to delete the associated volume and PVC resources or not. If you delete these resources, then all the data are lost. If you choose to keep the resources, then you can reuse the data by attaching the PVC to another VM.

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 it to the same VM).

To attach the disk, stop the VM, navigate to the **Disks** tab, click **Add disk**, and then complete the form:

1. Select **Use an existing PVC** for the source.
2. Select the PVC to attach to the VM. OpenShift Virtualization constructs the name of the PVCs from the name of the initial machine for which you created the disk and from the name you gave to the disk.

> **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.

Preallocate Disk Space for Improved Performances

When OpenShift Virtualization copies a disk image into a PV, it only allocates the space that the image actually uses.

For example, when you create 10 GiB blank disk, OpenShift Virtualization does not reserve that space with the back-end storage. Instead, it progressively allocates the space as you start writing data to the disk. This mechanism allows better utilization of your back-end storage because it only consumes the space that you really 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 `DataVolume` resource file, use the `preallocation: true` option:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: datadisk
spec:
  preallocation: true
  storage:
    resources:
      requests:
        storage: 10Gi
    storageClassName: ocs-external-storagecluster-ceph-rbd
  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 you request even if you do not enable preallocation.

Resize a Virtual Machine Disk

Red Hat OpenShift Container Platform (RHOCP) allows resizing of PVCs. When you change the size of a PVC, RHOCP automatically resizes the associated PV. Because most VM disks are based on PVCs, you resize a disk by resizing its PVC.

> **Note**
> You can only expand PVCs. Shrinking a PVC is not possible.

Confirm that the Storage Class Allows Expansion

Not all back-end storage allows resizing. When resizing is available, the storage class resource has the `allowVolumeExpansion` parameter set to true:

```
[user@host ~]$ oc get storageclass ocs-external-storagecluster-ceph-rbd -o yaml
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    description: Provides RWO Filesystem volumes, and RWO and RWX Block volumes
  creationTimestamp: "2022-02-14T21:14:15Z"
  name: ocs-external-storagecluster-ceph-rbd
...output omitted...
```

To get that information by using the OpenShift web console, navigate to **Storage** > **StorageClasses**, select the storage class, and then navigate to the **YAML** tab.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Expand a Persistent Volume Claim

You can expand a PVC by editing its resource file:

```
[user@host ~]$ oc edit pvc vm1-staticassets-e6s3s
...output omitted...
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 15Gi
  storageClassName: ocs-external-storagecluster-ceph-rbd
  volumeMode: Block
  volumeName: pvc-8dfd1346-822e-4098-bf8d-8aeba16638d4
...output omitted...
```

To expand the PVC by using the OpenShift web console, navigate to **Storage** > **PersistentVolumeClaim**, select the PVC to edit, click **Actions** > **Expand PVC**, and then provide the new size.

Expand the Virtual Machine File System

Resizing a PVC does not require stopping the VM that is using it. However, the VM only detects the new disk size after a restart.

Most operating systems report the new size but do not automatically expand the partitions and the file systems. On Linux systems that enable the `cloud-init` service, these operations might be automatically performed 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://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/storage/index#expanding-persistent-volumes

KubeVirt Data Volumes
https://github.com/kubevirt/containerized-data-importer/blob/main/doc/datavolumes.md

KubeVirt Data Volume preallocation
https://github.com/kubevirt/containerized-data-importer/blob/main/doc/preallocation.md

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Attaching and Accessing Disks on Virtual Machines

A new MariaDB VM is provisioned with a boot disk and a single 15 GB data disk. To ensure performance, you must separate database data files, transaction logs, and undo files onto their own disks. You will add a new disk for each file type, point MariaDB to the appropriate disk for each type, and then delete the old disk to reclaim your storage.

Outcomes

- Create disks with specific requirements
- Attach disks to existing VMs
- Detach disks from VMs
- Delete disks

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It 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 then navigate 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 redhat \
  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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1.3. 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 the `redhat` password when prompted.

1.4. Navigate to **Virtualization** > **VirtualMachines**. Confirm that the `mariadb-server` VM is running.

2. Stop the `mariadb-server` VM to prepare it for disk addition. Review the attached disks.

2.1. From the OpenShift web console, select the `mariadb-server` VM and then click **Actions** > **Stop** to stop it. Click **Stop** to confirm the operation.

2.2. Navigate to the **Disks** tab. 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 for its data.

3. Create and attach three 5 GiB disks to the `mariadb-server` VM. Use the fast `ocs-external-storagecluster-ceph-rbd` storage class in `Block` mode. Preallocate the space for improving write performances. Use the `virtio` interface to connect the disks to the VM.

3.1. Click **Add disk** to create and attach the first disk. You configure that disk for the MariaDB Server data files in another step. Complete the form using the following information.

> **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.

Click **Add** to create and attach the disk to the VM.

Field	Value
Source	`Blank (creates PVC)`
Name	`dbdata`
Size	5 GiB
Type	`Disk`
Interface	`virtio`
StorageClass	`ocs-external-storagecluster-ceph-rbd`
Enable preallocation	Selected

3.2. Repeat the preceding step but use `dbredo` for the disk name. You configure that disk for the MariaDB Server redo log in another step.

3.3. Repeat the process one last time for the `dbundo` disk. You configure that disk for the MariaDB Server undo table spaces in another step.

3.4. Click **Actions** > **Start** to start the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3.5. Log in to the VM console and confirm that three new 5 GiB devices are available. Navigate 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 three 5 GiB 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
vdd    252:48   0    5G  0 disk
vde    252:64   0    5G  0 disk
vdf    252:80   0    5G  0 disk
```

4. The `lab` command has prepared the `/root/dbmove.sh` script to format the three disks and then 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...
```

4.2. Confirm that you can access the database:

```
[root@mariadb-server ~]# mysql --user=devuser --password=developer
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...
MariaDB [(none)]> USE sakila;
...output omitted...
Database changed.
MariaDB [sakila]> SHOW TABLES;
+------------------+
| Tables_in_sakila |
+------------------+
| actors           |
| actor_info       |
| address          |
| category         |
...output omitted...
MariaDB [sakila]> quit
Bye
[root@mariadb-server ~]#
```

4.3. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5. Detach and then delete the `dbroot` disk. The MariaDB Server stored its data on that disk before you moved them to the three new disks.

5.1. Click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

5.2. Navigate to the **Disks** tab. Click the vertical ellipsis to the right of the `dbroot` disk and then click **Delete**. In the confirmation window, ensure that **Delete dbroot Data Volume and PVC** is selected and then click **Detach**.

5.3. Click **Actions** > **Start** to start the VM.

5.4. Log in to the VM console and confirm that the 15 GiB device does not show anymore. Navigate 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 deleted 15 GiB block device is not listed anymore.

```
[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    5G  0 disk /mariadb/data
vdd    252:48   0    5G  0 disk /mariadb/redo
vde    252:64   0    5G  0 disk /mariadb/undo
```

5.5. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

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 storage-multiple
```

Connecting Virtual Machines to External Storage

Objectives

- Connect a VM to external storage using Multus.

Prepare External Storage

When you create a *Persistent Volume Claim (PVC)* to request a volume for your application, Kubernetes prepares a *Persistent Volume (PV)*, and then binds it to the PVC.

With dynamic provisioning, Kubernetes creates persistent volumes on demand. It uses a *provisioner* to control the back-end storage.

Not all storage types have a provisioner available. In some organizations, dynamic provisioning is not allowed 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. Your storage administrators prepare several volumes with the back-end storage, and then your Kubernetes administrators declare these volumes in PV resources. When you create the PVC, Kubernetes binds one available PV to the claim.

Connect a Virtual Machines to External Storage

You have several options to connect VMs 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 is to connect the storage directly from inside the VM by using the VM operating systems tools. 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 your 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 on your VM that is connected to the storage network. The Kubernetes NMState Operator and Multus are explained in greater 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 in the first option, your cluster administrators can use the Kubernetes NMState Operator to configure the cluster nodes for network storage that you access through a dedicated network.

The following paragraphs show how to create the PV and the PVC.

Create a Persistent Volume

A PV represents a piece of storage. It provides all the details that the cluster nodes need to connect to that storage.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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-staticimgs
    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 prepared by your storage administrators.
2. The available access modes depend on the storage type. For iSCSI, only `ReadWriteOnce` and `ReadOnlyMany` are allowed. 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 when possible. For iSCSI PVs, you can choose between `Block` and `Filesystem` modes. Some other PV types, such as NFS, only support 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 under 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)
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
  volumeMode: Filesystem  ❷
  claimRef:
    name: websrv1-logs
    namespace: vm-project
  nfs:  ❸
    path: /exports-ocp/vm135
    server: 10.20.42.42
```

❶ NFS PVs support the `ReadWriteMany` access mode. That mode allows VM live migration.

❷ For NFS PVs, Kubernetes only support the `Filesystem` mode. Because the default value for the `volumeMode` parameter is `Filesystem`, you could omit the parameter altogether. If you use the PV for a VM disk, then OpenShift Virtualization creates a disk image file in the NFS share.

❸ 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 volume. Using Fibre Channel implies that the cluster nodes have *host bus adapters (HBAs)* connected to a Fibre Channel storage array.

```
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:
      - "50060e801049cfd1"
    lun: 0
```

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 `oc apply -f resourcefile.yaml` command as a cluster administrator.

You can also use the OpenShift web console to create PVs. Log in to the web console as an administrator, navigate to **Storage** > **PersistentVolumes**, click **Create PersistentVolume**, and then write the resource file into the YAML editor. Click **Create** to create the resource.

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:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: websrv1-staticimgs
  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 `oc apply -f resourcefile.yaml` command.

You can also use the OpenShift web console to create PVCs. Navigate to **Storage** > **PersistentVolumeClaims**, click **Create PersistentVolumeClaim** > **With Form**, and then complete the form. 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. Click **Create** to create the resource.

Note

Kubernetes does not delete the static PV when you delete the PVC. If you do not need the PV anymore, then ask your cluster administrator to delete it. If you want to create a PVC that reuses the PV, then your cluster administrator must release the PV first. To release the PV, your 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 of 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 creating the PVC, the associated volume is empty. If you attach it as a VM disk, then you get an empty disk that you can partition, format, and mount.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

However, if you have a disk image in the RAW 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-staticimgs \
  --image-path=./webimgfs.qcow2 --no-create
Using existing PVC vm-project/websrv1-staticimgs
Waiting for PVC websrv1-staticimgs 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
```

With the OpenShift web console, you inject the disk image at the same time that you create the PVC. Navigate to **Storage** > **PersistentVolumeClaims**, click **Create PersistentVolumeClaim** > **With Data upload form**, and then complete the form.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 5.5: 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. Navigate to **Virtualization** > **VirtualMachines**, select the VM, and then navigate to the **Disks** tab. Click **Add disk** and complete the form:

- Select **Use an existing PVC** in the **Source** field.
- Select your PVC in the **PersistentVolumeClaim** field.

Click **Add** to attach the disk.

Remember that you must stop the VM to use the `virtio` interface.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

References

Knowledgebase: "How to manually reclaim and reuse OpenShift Persistent volumes that are Released"
https://access.redhat.com/solutions/4651451

For more information about the access modes in the available per volume plug-in, refer to the *Access Modes* section in the *Red Hat OpenShift Container Platform Storage* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/storage/index#pv-access-modes_understanding-persistent-storage

For more information about injecting a disk image into a volume, refer to the *Uploading Local Disk Images by Using the Web Console* section in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#virt-uploading-local-disk-images-web

Kubernetes List of Provisioner
https://kubernetes.io/docs/concepts/storage/storage-classes/#provisioner

Kubernetes Persistent Volume Reference
https://kubernetes.io/docs/reference/kubernetes-api/config-and-storage-resources/persistent-volume-v1/

Guided Exercise

Connecting Virtual Machines to External Storage

Add a new data disk to a MariaDB VM over iSCSI to accommodate a new database. Use a dedicated data disk from your SAN with iSCSI.

Outcomes

- Create a Persistent Volume to declare an iSCSI target.
- Create a new 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.

This command ensures that the cluster API is reachable. It 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 then navigate 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 redhat \
  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 navigate to `https://console-openshift-console.apps.ocp4.example.com`. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

   1.4. Navigate to **Virtualization** > **VirtualMachines**. Confirm that the `mariadb-server` VM is running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. Create the PV resource that declares the iSCSI target to connect.

2.1. From the terminal on the `workstation` machine, review the `/home/student/DO316/labs/storage-external/pv.yaml` resource file prepared by the `lab` command.

```
[student@workstation ~]$ cat ~/DO316/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 available in the classroom environment.

2.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/storage-external/pv.yaml
persistentvolume/iscsi-pv created
```

If you prefer the OpenShift web console, navigate to **Storage** > **PersistentVolumes**, click **Create PersistentVolume**, and then copy and paste the `~/DO316/labs/storage-external/pv.yaml` resource file into the YAML editor. Click **Create** to create the resource.

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 terminal, review the `/home/student/DO316/labs/storage-external/pvc.yaml` resource file that the `lab` command has prepared.

```
[student@workstation ~]$ cat ~/DO316/labs/storage-external/pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb-binlog
  namespace: storage-external
spec:
  resources:
    requests:
      storage: 1Gi
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
```

Notice that the `name`, `namespace`, `storage`, `accessModes`, and `volumeMode` parameters match the parameters of the PV resource.

3.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/storage-external/pvc.yaml
persistentvolumeclaim/mariadb-binlog created
```

If you prefer the OpenShift web console, navigate to **Storage** > **PersistentVolumeClaims**, select the `storage-external` project, click **Create PersistentVolumeClaim** > **With Form**, and then complete the form using the following information.

Field	Value
StorageClass	Keep the default provided class
PersistentVolumeClaim name	`mariadb-binlog`
Access mode	`Single user (RWO)`
Size	1 GiB
Volume mode	`Block`

Click **Create** to create the resource.

3.3. Run the `oc get pvc` command to confirm that Kubernetes binds your PVC to the `iscsi-pv` PV.

```
[student@workstation ~]$ oc get pvc -n storage-external
NAME            STATUS  VOLUME    CAPACITY  ACCESS MODES  STORAGECLASS  AGE
mariadb-binlog  Bound   iscsi-pv  1Gi       RWO           nfs-storage   10m
...output omitted...
```

The command must report the status as `Bound` and the volume as `iscsi-pv`.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 `virtio` interface.

4.1. Use the OpenShift web console to stop the VM 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.

Navigate to **Virtualization** > **VirtualMachines**, select the `mariadb-server` VM, and then click **Actions** > **Stop**. Click **Stop** to confirm the operation. Wait for the machine to stop.

4.2. Navigate to the **Disks** tab and then click **Add disk**. Complete the form using the following information.

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.

Click **Add** to create and attach the disk to the VM.

Field	Value
Source	`Use an existing PVC`
PersistentVolumeClaim	`mariadb-binlog`
Name	`binlogdisk`
Type	`Disk`
Interface	`virtio`

4.3. Click **Actions** > **Start** to start the VM.

4.4. Log in to the VM console and confirm that a new 1 GiB device is available. Navigate 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 `vdc` block device.

```
[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    1G  0 disk
```

4.5. Log out of the VM console.

```
[root@mariadb-server ~]# logout
```

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 storage-external
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Configuring Storage for Virtual Machines

Manage storage and disks for VMs in Red Hat OpenShift.

Outcomes

- Attach and detach disks from 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `storage-review` namespace and starts two virtual machines in that namespace: the `vm1` and the `vm2` VMs.

```
[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 `admin` using `redhat` as the password. The 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 two VMs are running under the `storage-review` project.

2. A disk named `webroot` is connected to the `vm1` VM. Detach the disk and then connect it to the `vm2` VM. Use the `virtio` interface when you connect the disk to the VM. Make sure to preserve the data when you detach the disk.

   Ensure that the `vm1` VM is running after you have completed your work.

   To verify the disk, log in to the console of the `vm2` VM as `root` using `redhat` as the password, and then run the `lsblk` command. The command output should list the new `vdb` disk.

3. Create the PV resource named `nfs-pv` that declares an external NFS share. 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 has prepared the `/home/student/DO316/labs/storage-review/nfspv.yaml` resource file that you can use as a model. If you decide to use that file, then you need to adapt it to the requirements.

4. Create the `weblog` PVC in the `storage-review` namespace with the following parameters:

Parameter	Value
PVC name	`weblog`
Access mode	`ReadWriteMany (RWX)`
Size	5 GiB
Volume mode	`Filesystem`

The `lab` command has prepared the `/home/student/DO316/labs/storage-review/nfspvc.yaml` resource file that you can use as a model. If you decide to use that file, then you need to 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` VM by using a `virtio` interface.

   Ensure that the `vm2` VM is running after you have completed your work.

   To verify the disk, log in to the console of the `vm2` VM as `root` using `redhat` as the password, and then run the `lsblk` command. The command output should list a new 5 GiB disk named `vdc`.

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

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 storage-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Configuring Storage for Virtual Machines

Manage storage and disks for VMs in Red Hat OpenShift.

Outcomes

- Attach and detach disks from 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `storage-review` namespace and starts two virtual machines in that namespace: the `vm1` and the `vm2` VMs.

```
[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 `admin` using `redhat` as the password. The 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 two VMs are running under the `storage-review` project.

   1.1. From a terminal, log in to your OpenShift cluster as `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 the `redhat` password when prompted.

   1.3. Navigate to **Virtualization** > **VirtualMachines** and then select the `storage-review` project. Confirm that the `vm1` and the `vm2` VMs are running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. A disk named `webroot` is connected to the `vm1` VM. Detach the disk and then connect it to the `vm2` VM. Use the `virtio` interface when you connect the disk to the VM. Make sure to preserve the data when you detach the disk.

Ensure that the `vm1` VM is running after you have completed your work.

To verify the disk, log in to the console of the `vm2` VM as `root` using `redhat` as the password, and then run the `lsblk` command. The command output should list the new `vdb` disk.

2.1. Use the OpenShift web console to stop the `vm1` VM. Select the `vm1` VM and then click **Actions** > **Stop**. Click **Stop** to confirm the operation. Wait for the machine to stop.

2.2. Navigate to the **Disks** tab. At the end of the row of the `webroot` disk, click the vertical ellipsis menu and then click **Delete**. In the confirmation window, clear the **Delete Data Volume and PVC** checkbox and then click **Detach**.

2.3. Click **Actions** > **Start** to start the VM.

2.4. Navigate to **Virtualization** > **VirtualMachines**, select the `vm2` VM, and then click **Actions** > **Stop**. Click **Stop** to confirm the operation. Wait for the machine to stop.

2.5. Navigate to the **Disks** tab and then click **Add disk**. Complete the form using the following information.

> **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.

Click **Add** to attach the disk to the VM.

Field	Value
Source	`Use an existing PVC`
PersistentVolumeClaim	`webroot`
Name	`webroot`
Type	`Disk`
Interface	`virtio`

2.6. Click **Actions** > **Start** to start the VM.

2.7. Log in to the VM console and confirm that a new 1 GiB device is available.

Navigate 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 `vdb` 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    1G  0 disk
```

2.8. Log out of the VM console.

```
[root@vm2 ~]# logout
```

3. Create the PV resource named `nfs-pv` that declares an external NFS share. 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 has prepared the `/home/student/DO316/labs/storage-review/nfspv.yaml` resource file that you can use as a model. If you decide to use that file, then you need to adapt it to the requirements.

3.1. From the terminal on the `workstation` machine, edit the `/home/student/DO316/labs/storage-review/nfspv.yaml` resource file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  volumeMode: Filesystem
  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 ~/DO316/labs/storage-review/nfspv.yaml
persistentvolume/nfs-pv created
```

If you prefer the OpenShift web console, navigate to **Storage** > **PersistentVolumes**, click **Create PersistentVolume**, and then copy and paste the resource file into the YAML editor. Click **Create** to create the resource.

3.3. Run the `oc get pv` command to verify your work.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get pv
NAME    CAPACITY ACCESS MODES  RECLAIM POLICY STATUS    CLAIM                 ...
nfs-pv 5Gi       RWX           Retain         Available storage-review/weblog ...
...output omitted...
```

4. Create the `weblog` PVC in the `storage-review` namespace with the following parameters:

Parameter	Value
PVC name	`weblog`
Access mode	`ReadWriteMany (RWX)`
Size	5 GiB
Volume mode	`Filesystem`

The `lab` command has prepared the `/home/student/DO316/labs/storage-review/nfspvc.yaml` resource file that you can use as a model. If you decide to use that file, then you need to adapt it to the requirements.

Confirm that Kubernetes binds the `weblog` PVC to the PV named `nfs-pv`.

4.1. From the terminal, edit the `/home/student/DO316/labs/storage-review/nfspvc.yaml` resource file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: weblog
  namespace: storage-review
spec:
  resources:
    requests:
      storage: 5Gi
  accessModes:
    - ReadWriteMany
  volumeMode: Filesystem
```

4.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/storage-review/nfspvc.yaml
persistentvolumeclaim/weblog created
```

If you prefer the OpenShift web console, navigate to **Storage** > **PersistentVolumeClaims**, select the `storage-review` project, click **Create PersistentVolumeClaim** > **With Form**, and then complete the form using the following information.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Field	Value
StorageClass	Keep the default provided class
PersistentVolumeClaim name	`weblog`
Access mode	`Shared access (RWX)`
Size	5 GiB
Volume mode	`Filesystem`

Click **Create** to create the resource.

4.3. Run the `oc get pvc` command to confirm that Kubernetes binds your PVC to the `nfs-pv` persistent volume.

```
[student@workstation ~]$ oc get pvc -n storage-review
NAME      STATUS  VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS  AGE
...output omitted...
weblog    Bound   nfs-pv   5Gi        RWX            nfs-storage   24s
...output omitted...
```

5. Attach the `weblog` PVC as a disk to the `vm2` VM by using a `virtio` interface.

Ensure that the `vm2` VM is running after you have completed your work.

To verify the disk, log in to the console of the `vm2` VM as `root` using `redhat` as the password, and then run the `lsblk` command. The command output should list a new 5 GiB disk named `vdc`.

5.1. Use the OpenShift web console to stop the `vm2` VM. Navigate to **Virtualization** > **VirtualMachines**, select the `vm2` VM, and then click **Actions** > **Stop**. Click **Stop** to confirm the operation. Wait for the machine to stop.

5.2. Navigate to the **Disks** tab and then click **Add disk**. Complete the form using the following information.

> **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.

Click **Add** to attach the disk to the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Field	Value
Source	`Use an existing PVC`
PersistentVolumeClaim	`weblog`
Name	`weblog`
Type	`Disk`
Interface	`virtio`

5.3. Click **Actions** > **Start** to start the VM.

5.4. Log in to the VM console and confirm that a new 5 GiB device is available.

Navigate 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@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    1G  0 disk
vdc    252:32   0    5G  0 disk
```

5.5. Log out of the VM console.

```
[root@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

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 storage-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- *Persistent Volume (PV)* resources represent a piece of storage.
- Developers request and manage persistent storage for their pods and VMs by creating *Persistent Volume Claim (PVC)* resources.
- VM disks use PVCs as their back-end storage.
- Red Hat OpenShift Virtualization adds the *Data Volume (DV)* resource type.
- A Data Volume resource simplifies the creation of VM disks by grouping a PVC definition and the disk image to inject into the volume.
- Administrators can manually create static PVs that describe external storage.
- Developers can create PVCs to use these static PVs. Developers can then attach these PVCs to their VMs so that the VMs can access the external storage.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 6

Virtual Machine Template Management

Goal

Create and manage templates to provision virtual machines.

Objectives

- Use and modify preconfigured templates to provision VMs.
- Generate custom templates to provision VMs.

Sections

- Virtual Machine Template Usage (and Guided Exercise)
- Create Custom Templates for Virtual Machines (and Guided Exercise)

Lab

- Virtual Machine Template Management

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Virtual Machine Template Usage

Objectives

- Use and modify preconfigured templates to provision VMs.

Virtual Machine Templates

Red Hat OpenShift Container Platform (RHOCP) provides resources that facilitate the creation and administration of workload objects in a cluster. One of these resources is preconfigured templates. Templates contain default editable parameters that you can customize according to your needs. RHOCP provides templates for several resource types, including network interfaces, *persistent volumes (PV)*, and *virtual machines (VM)*.

Red Hat OpenShift Virtualization provides a catalog of preconfigured templates that you can use 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 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, and flavor, which specifies CPU and memory resources.

Attributes of a Template

Preconfigured templates include fields that can be changed according to the needs of your VM. The default parameters suggest a basic configuration that works with any VM. However, if you customize the VM configuration, there are some important considerations:

Boot Source

This field indicates the source of the image used to create your VM and includes the following options:

- **PVC (creates PVC)**: Clone an existing PVC available in the cluster to create a new PVC.
- **Registry (creates PVC)**: Create a new PVC by importing content from a container registry.
- **URL (creates PVC)**: Create a new PVC by importing content from a URL with a HTTP or S3 endpoint.
- **PXE (network boot - adds network interface)**: Boots an operating system stored on a server on the network (a PXE bootable network attachment definition is required).

Flavor

This field indicates the size of your VM in terms of CPU and memory and includes the following options:

- **Tiny**: Creates a VM with 1 CPU and 1 GiB memory; recommended for testing VM creation.
- **Small**: Creates a VM with 1 CPU and 2 GiB memory; this is the default option for any preconfigured template.
- **Medium**: Creates a VM with 1 CPU and 4 GiB memory; appropriate for code testing or to store basic application resources.

- **Large**: Creates a VM with 2 CPU and 8 GiB memory; recommended for systems that require a significant consumption of resources.
- **Custom**: Specify custom values of CPU and memory as needed for your VM.

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 guaranteed VM performance. VMs with this configuration are recommended for use in the OpenShift web console.
- **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 guaranteed VM performance over VM density.

Storage

A preconfigured Linux-based template has two partitions by default, cloud-init and root disk. However, it is possible to 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 desired performance of your VM. The available options are *virtio*, *SATA*, or *SCSI*.
- **Storage Class**: 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, overriding the default storage profile settings. The available access modes are **Single user (RWO)**, **Shared access (RWX)**, or **Read only (ROX)**.
- **Volume mode**: You can choose between **Filesystem** and **Block** storage volume modes for your VM, depending upon the selected storage class.

> **Note**
> VMs must have a PVC with a shared *ReadWriteMany (RWX)* access mode to enable live migration.

Networking

Red Hat provides templates that are set with a default network interface connected to the pod network. You can configure additional network interfaces for your VM. For each field, you have several options available 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 best performance and is supported by most Linux distributions. Additional drivers are needed for Windows VMs.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The `e1000e` model is supported by most operating systems, including Windows, but offers slower performance compared to 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 available 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.

Cloud-init

Available on compatible systems, you can use the `cloud-init` service to configure the guest operating system post-installation. You can use this service to install tools and packages, configure users and passwords, and manage system applications.

> **Note**
> For more information on the `cloud-init` modules, refer to *Red Hat support for cloud-init* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_cloud-init_for_rhel_8/red-hat-support-for-cloud-init_cloud-content

CDI and Data Volumes

The *Containerized Data Importer (CDI)* is an add-on that manages persistent storage for VMs in OpenShift Virtualization. CDI provides the ability to import, upload, and clone existing PVCs for your VMs. CDI provides a *custom resource definition (CRD)* for `DataVolume` objects that orchestrate the import, clone, and upload operations associated with an underlying PVC. Data volumes are integrated into your OpenShift cluster when you install the OpenShift Virtualization operator. Data volumes prevent a VM from starting until its underlying PVC is ready.

OpenShift Virtualization allows you to create a new VM from a data volume, including cloning an existing PVC as a new data volume or uploading a local disk image.

You can use a data volume template to create a new VM from an existing PVC by using the OpenShift web console or from the command line. Cloning the PVC of an existing VM in a different namespace produces the following configuration for a new VM:

> **Note**
> Additional permissions are required to clone data volumes across namespaces. For more information about the required permissions, refer to *Enabling user permissions to clone data volumes across namespaces* at https://docs.openshift.com/container-platform/4.10/virt/virtual_machines/cloning_vms/virt-enabling-user-permissions-to-clone-datavolumes.html

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-clone-pvc    (1)
  labels:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
    kubevirt.io/vm: vm-clone-pvc
  namespace: backup-vms  ❷
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-clone-pvc
    spec:
      domain:
       cpu:
          cores: 1
        devices:
          disks:
          - disk:
              bus: virtio
            name: root-disk
        resources:
          requests:
            memory: 2Gi
      terminationGracePeriodSeconds: 0
      volumes:
        - dataVolume:
            name: new-pvc-from-clone  ❸
          name: root-disk
  dataVolumeTemplates:  ❹
    - apiVersion: cdi.kubevirt.io/v1beta1
      metadata:
        name: new-pvc-from-clone  ❺
      spec:
        pvc:
          accessModes:
          - ReadWriteOnce
          resources:
            requests:
              storage: 10Gi  ❻
        source:
          pvc:
            namespace: storage-vm  ❼
            name: database-storage-pvc  ❽
```

❶ Name of the new VM.

❷ Namespace where the new VM is created.

❸❺ Name of the new PVC that is created from the original PVC.

❹ Data volume template to specify the new PVC to create.

❻ If your PVC uses the *Filesystem* volume mode, the requested storage must be larger than the storage source to account for file system overhead data.

❼ Source namespace where the original VM and PVC exists.

❽ Name of the source PVC that is used to create the new VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

You can apply the completed YAML manifest to you cluster with the `oc create -f` or `oc apply -f` commands.

```
[user@host ~]$ oc create -f virtualmachine-vm-example-clone.yaml
virtualmachine.kubevirt.io/vm-clone-pvc created
```

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. The following example shows how to create a Red Hat Enterprise Linux 8.0 VM from the web console by using a preconfigured template:

From the OpenShift web console left panel, navigate to **Virtualization** > **Templates**.

Figure 6.1: OpenShift Virtualization menu

Locate the *Red Hat Enterprise Linux 8.0+ VM* template and click **Create VM**.

Figure 6.2: Red Hat Enterprise Linux 8.0+ VM template

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Click **Customize virtual machine** and then provide a name for your VM. Select the **URL (creates PVC)** option in the **Boot Source** list. Click **RHEL download page**, log in with your credentials, and then copy the URL of the *KVM Guest image* download. Leave the default values for **Flavor** (`Small`) and the **Workload Type** (`Server`). Click **Next**.

Figure 6.3: Customizing the VM template

Confirm that the *default* interface is connected to *Pod Networking* with a *masquerade* bonding method, and then click **Next**.

Figure 6.4: Network interfaces

Leave the disks at their default values and click **Next**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 6.5: Storage configuration

Leave the **Cloud-init** form fields at the default values and click **Next**.

Figure 6.6: Cloud-init and advanced configuration

> **Note**
> A username and password are created by default. After creating the VM, you can examine the credentials generated by the `cloud-init` service by navigating to **VirtualMachines** > *VM-Name*. Click the **Console** tab and then click **Guest login credentials** located above the console of your VM.

Review the settings of your VM and click **Create Virtual Machine**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 6.7: Reviewing the VM

Navigate to **Virtualization** > **VirtualMachines** and click the name of your VM to see its status.

Figure 6.8: Status of the VM

References

For more information, refer to the *Red Hat OpenShift Virtualization* documentation at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Virtual Machine Template Usage

Define the attributes of a product template based on a predefined image to create a medium sized VM containing a web server.

Outcomes

- Create a VM by using a product template.
- Define the attributes of a product template. Define the boot source as a predefined image and define the size, the workload type, and custom access credentials.
- Access the VM to confirm that the web server is up and running.
- Delete the VM.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `template-intro` namespace and exercise resources.

The OpenShift Virtualization operator auto-imports default boot images for Red Hat provided templates. To enable custom boot sources for these templates, the `lab` command ensures that the `enableCommonBootImageImport` feature is disabled at the start of the exercise.

```
[student@workstation ~]$ lab start template-intro
```

Instructions

1. As the `admin` user, confirm that the `web-server-image` data volume is available in the `vm-images` namespace. Locate and then navigate to the Red Hat OpenShift web console.

   1.1. From a terminal, log in to your Red Hat OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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. Confirm that the corresponding PVC's capacity is 10 GiB and has a `Bound` status.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get datavolume -n vm-images
NAME                PHASE       PROGRESS   RESTARTS   AGE
web-server-image    Succeeded   100.0%                69s
[student@workstation ~]$ oc get pvc -n vm-images
NAME                STATUS   VOLUME            CAPACITY   ACCESS MODES ...
web-server-image    Bound    pvc-e32c...89b5   10Gi       RWX          ...
```

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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

2. Create the `helloworld` VM in the `template-intro` project by using the *Red Hat Enterprise Linux 8.0+ VM* template. Define the boot source as the `web-server-image` PVC in the `vm-images` project.

2.1. From the web console left panel, navigate to **Virtualization** > **Templates**. Select the project `template-intro` and then locate the row containing the `Red Hat Enterprise Linux 8.0+ VM` template. Click **Create VM**.

2.2. Select the option `PVC (creates PVC)` under the **Boot source type** option.

2.3. Specify the `vm-images` namespace under the **Persistent Volume Claim project** option.

2.4. Select the `web-server-image` PVC under the **Persistent Volume Claim name** option.

Figure 6.9: Select the boot source

2.5. Click **Customize virtual machine**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2.6. Specify `helloworld` as the VM name and select `Medium` as the **Flavor**. Confirm that `Server` is selected for **Workload Type** and then click **Next**.

2.7. Confirm that the network interface type is `masquerade` and click **Next**.

2.8. Confirm that the attached disks include `cloudinitdisk` and `rootdisk`, and then click **Next**.

2.9. Define the `developer` user and the `developer` password in the **Cloud-init** form. Click **Review and confirm**.

2.10. Review the defined settings and click **Create Virtual Machine**.

3. Observe the status of the `helloworld` VM and its resources by using the command line.

3.1. In your terminal window, observe the progress of the VM's data volume.

```
[student@workstation ~]$ oc get datavolume -n template-intro
NAME                    PHASE            PROGRESS   RESTARTS   AGE
helloworld-rootdisk...  CloneInProgress  70.0%                 25s
[student@workstation ~]$ oc get datavolume -n template-intro
NAME                    PHASE            PROGRESS   RESTARTS   AGE
helloworld-rootdisk...  Succeeded                             45s
```

3.2. Observe the status of the `helloworld` VM.

```
[student@workstation ~]$ oc get vm -n template-intro
NAME         AGE   STATUS    READY
helloworld   2m    Running   True
```

4. Connect to the console of the `helloworld` VM by using the `virtctl console` command. Log in as the `developer` user with the `developer` password that you defined in a previous step.

4.1. In your terminal, 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 -n template-intro
Successfully connected to helloworld console. The escape sequence is ^]

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

helloworld login:
```

4.2. Log in as the `developer` user with the `developer` password.

```
hellworld login: developer
Password: developer
[developer@helloworld ~]$
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5. Confirm that the Apache HTTP server is running and available.

   5.1. 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; vendor prese>
   Active: active (running) since Tue 2022-04-12 17:16:52 EDT; 5min ago
     Docs: man:httpd.service(8)
 Main PID: 947 (httpd)
   Status: "Running, listening on: port 80"
...output omitted...
```

   5.2. Use the `curl` command to confirm that web server responds to requests. The command returns some HTML content.

```
[developer@helloworld ~]$ curl localhost
<!DOCTYPE html
...output omitted...
</html>
```

6. Shut down and delete the `helloworld` VM.

   6.1. Shut down the `helloworld` VM.

```
[developer@helloworld ~]$ sudo systemctl poweroff
...output omitted...
[ 2172.699653] reboot: Power down

You were disconnected from the console. This has one of the following reasons:
...output omitted...
```

   6.2. Delete the `helloworld` VM with the `oc` command.

```
[student@workstation ~]$ oc delete vm helloworld -n template-intro
virtualmachine.kubevirt.io "helloworld" deleted
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish template-intro
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Create Custom Templates for Virtual Machines

Objectives

- Generate custom templates to provision VMs.

Customized Template Creation

You can create custom templates to provision *virtual machines (VMs)* in Red Hat OpenShift Virtualization. In the OpenShift web console, navigate to the **Virtualization** > **Templates** to create and manage cluster templates. You can browse through and select from the catalog of available operating systems for cluster templates, or define new ones.

You cannot access custom templates within a specific project from other projects. To make a custom template available to all projects, create the template in the `openshift` namespace, where the cluster templates reside. Use the `oc get templates -n openshift` command command to view the list of templates available to all projects.

Click **Create** and select the method to use for template creation. Click **With Wizard** to start the template creation wizard to customize template parameters. Click **With YAML** to display the YAML editor where you can edit the template parameters manually, or upload a file written in either YAML or JSON.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Customizable Template Parameters

When creating custom templates, many parameters are available to provide the proper information, definitions, and supporting descriptions. When you use the template creation wizard, the following parameters are displayed, noting which are mandatory.

Virtual Machine Template General Parameters

Parameter	Required	Description
`Name`	Yes	Display name.
`Template Provider`	Yes	Specifies the author.
`Template Support`	No	Denotes support level.
`Description`	No	Helpful context information regarding the environment and intended usage of VMs created from the template.
`Operating System`	Yes	List of available VM operating systems; selecting an operating system automatically configures the default flavor and workload type for the VM.
`Boot Source`	Yes	Manner of access to the operating system image file, such as a registry or URL.
`Flavor`	Yes	Selection of machine resource configurations for both memory and processing requirements.
`Workload Type`	Yes	Purpose for the VM, such as workstation or server.

In the wizard, provide the needed selections for the new template and click **Next**. Review and customize other parameters, as necessary for the new template. Review each category to view the default selections or further customize those options. These categories store the template definitions that define cluster resources:

Networking
: Add, edit, or remove network interfaces

Storage
: Add, edit, or remove disks

Advanced
: `cloud-init`, `sysprep`, SSH access, and hardware device configurations

Review
: Overview of template definition prior to finalization

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

After the template customization is complete, select **Review and confirm** followed by **Create Virtual Machine template** to finish creating the template.

YAML Template Definitions

The following YAML template includes the parameters required for a manifest that you can either paste into the YAML template creation editor in the web console or add by using the `oc` command-line client.

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: vm-template-example (1)
  namespace: default (2)
  annotations:
    name.os.template.kubevirt.io/rhel8.5: Red Hat Enterprise Linux 8.0 or higher
    description: VM template example
    iconClass: icon-rhel
    template.kubevirt.ui/parent-support-level: Full (3)
    template.kubevirt.ui/parent-provider: Red Hat (4)
    template.kubevirt.ui/parent-provider-url: 'https://www.redhat.com' (5)
  labels:
    os.template.kubevirt.io/rhel8.5: 'true' (6)
    flavor.template.kubevirt.io/tiny: 'true' (7)
    workload.template.kubevirt.io/server: 'true' (8)
    vm.kubevirt.io/template: rhel8-server-tiny
    vm.kubevirt.io/template.namespace: openshift
    template.kubevirt.io/type: vm
parameters:
  - description: VM name
    from: rhel8-[a-z0-9]{16}
    generate: expression
    name: NAME (9)
  - description: Name of the DataSource to clone
    name: DATA_SOURCE_NAME (10)
    value: rhel8
  - description: Namespace of the DataSource
    name: DATA_SOURCE_NAMESPACE (11)
    value: openshift-virtualization-os-images
  - description: Randomized password for the cloud-init user cloud-user
    from: '[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}'
    generate: expression
    name: CLOUD_USER_PASSWORD (12)
objects:
  - apiVersion: kubevirt.io/v1
    kind: VirtualMachine
    metadata:
      annotations:
        vm.kubevirt.io/validations: |
          [
            {
              "name": "minimal-required-memory",
              "path": "jsonpath::.spec.domain.resources.requests.memory",
              "rule": "integer",
              "message": "This VM requires more memory.",
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
              "min": 1610612736
            }
          ]
      labels:
        app: '${NAME}'
        vm.kubevirt.io/template: rhel8-server-tiny
        vm.kubevirt.io/template.revision: '1'
        vm.kubevirt.io/template.version: v0.19.3
      name: '${NAME}'
    spec:
      running: false
      template:
        metadata:
          annotations:
            vm.kubevirt.io/flavor: tiny
            vm.kubevirt.io/os: rhel8
            vm.kubevirt.io/workload: server
          labels:
            kubevirt.io/domain: '${NAME}'
            kubevirt.io/size: tiny
        spec:
          domain:
            cpu:
              cores: 1
              sockets: 1
              threads: 1
            devices:
              disks: 13
                - name: containerdisk
                  bootOrder: 1
                  disk:
                    bus: virtio
                - disk:
                    bus: virtio
                  name: cloudinitdisk
              interfaces: 14
                - masquerade: {}
                  name: default
                  model: virtio
              networkInterfaceMultiqueue: true
              rng: {}
            machine:
              type: pc-q35-rhel8.4.0
            resources:
              requests:
                memory: 1.5Gi
          evictionStrategy: LiveMigrate
          networks:
            - name: default
              pod: {}
          terminationGracePeriodSeconds: 180
          volumes:
            - name: containerdisk
              containerDisk:
                image: registry.redhat.io/rhel8/rhel-guest-image
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
            - name: cloudinitdisk
              cloudInitNoCloud: ⓯
                userData: |
                  #cloud-config
                  user: cloud-user
                  password: 1n4u-4qfw-wxsg
                  chpasswd:
                    expire: false
        hostname: '${NAME}'
```

General parameters found in a template manifest:

1. Template name
2. Project namespace
3. Support level
4. Template author
5. Boot source
6. Operating system
7. Flavor
8. Workload
9. Random generated name
10. Data source name
11. Data source namespace
12. Random generated password
13. Storage volumes
14. Networking interfaces
15. Cloud-init

Creating a VM from a Customized Template

After you create a custom template, it appears among the list of available virtual machine templates. From the web console, navigate to **Virtualization** > **Templates** and search or browse through the list of templates. To refine your search, sort the list using the title headers for each column.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

The template row contains the name, provider, boot source, and other details that can be further customized before launching a virtual machine. Click **Create VM** on the row for the preferred template to create a virtual machine using that template definition.

References

https://cloud.redhat.com/blog/customizing-virtual-machine-templates-in-openshift-1
[Red Hat Hybrid Cloud Blog]

For more information, refer to the *Creating virtual machine templates* chapter in the *Red Hat OpenShift Container Platform Virtualization Guide* at https://docs.openshift.com/container-platform/4.10/virt/vm_templates/virt-creating-vm-template.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Create Custom Templates for Virtual Machines

Create a Red Hat Enterprise Linux 8 template based on a predefined image containing a MariaDB server instance. You will then use the custom template to create a VM containing a database with preconfigured data.

Outcomes

- Create a template using the `mariadb-server` image by defining the attributes.
- Create a VM from the customized template and access it to confirm that the MariaDB service is up and running.
- Delete the VM and the template.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `template-create` namespace and ensures that all resources are available for this exercise.

```
[student@workstation ~]$ lab start template-create
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console.

   1.1. From a terminal, log in to your Red Hat OpenShift cluster as the `admin` user.

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

   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 navigate to https://console-openshift-console.apps.ocp4.example.com.

   1.4. Click **htpasswd_provider** and log in as the `admin` user with the `redhat` password.

2. Create a template named `template-create` in the `template-create` namespace. Configure the template to use a `mariadb-server` image hosted on a website. The

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

template must use the `Medium` flavor and the `Server` workload type. Configure the `cloud-init` service to create a `developer` user with the `developer` password.

2.1. Navigate to **Virtualization** > **Templates**. Select the `template-create` project from the **Projects** list.

2.2. Click **Create** > **With Wizard** to open the **Create Virtual Machine template** wizard.

2.3. Enter `template-create` in the **Name** field and enter `Red Hat Training` in the **Template provider** field. Select `Red Hat Enterprise Linux 8.0 or higher` from the **Operating System** list, and select `URL (creates PVC)` from the **Boot Source** list.

Figure 6.12: Template data

2.4. Enter http://utility.lab.example.com:8080/openshift4/images/mariadb-server.qcow2 in the **URL** field. Select `Medium` from the **Flavor** list and confirm that `Server (default)` is selected for the **Workload Type**.

Figure 6.13: URL entry

2.5. Click **Next** to further customize the template. Confirm that `nic-0` is on the `Pod Networking` network and then click **Next**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2.6. Review the storage settings to confirm that the `rootdisk` volume storage size is `30 GiB` and that the disk uses the `nfs-storage` storage class. Click **Next** to open the **Cloud-init** form.

2.7. In the **Cloud-init** form, enter `developer` in the **User** field. Enter `developer` in the **Password** field.

Figure 6.14: Cloud-init form

2.8. Click **Next** and review the custom template settings. Ensure that the data is correct and then click **Create Virtual Machine template**.

3. Create a VM from the `template-create` template.

3.1. Navigate to **Virtualization** > **Templates**. Select `template-create` from **Projects** list, if necessary.

3.2. Locate the row containing the `template-create` template and then click **Create VM** on the right side of the row.

Figure 6.15: Create VM

3.3. Enter `template-create-vm` in the **Virtual Machine Name** field.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 6.16: Name VM

3.4. Click **Create virtual machine** to begin the operation.

4. Access the `template-create-vm` VM.

4.1. Navigate to **Virtualization** > **VirtualMachines**.

4.2. Wait for the `template-create-vm` VM to provision and display a `Running` status.

4.3. Click the `template-create-vm` link to access the **Overview** tab for the VM.

Figure 6.17: VM overview

4.4. Navigate to the **Console** tab. Wait for the console to display the `login` prompt.

4.5. Use the console to log in as the `developer` user with the `developer` password.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
template-create-vm login: developer
Password: developer
[developer@template-create-vm ~]$
```

4.6. Confirm that the preconfigured database, `sakila`, is accessible. Use the `mysql` command to log in as the `devuser` with the `developer` password.

```
[developer@template-create-vm ~]$ mysql sakila -u devuser -p
Enter password: developer
Welcome to the MariaDB Monitor. Command end with ; or \g.
...output omitted...
MariaDB [sakila]>
```

4.7. Exit the `sakila` database and the `template-create-vm` VM.

```
MariaDB [sakila]> EXIT;
Bye
[developer@template-create-vm ~]$ exit
```

5. Delete the `template-create-vm` VM and the `template-create` template.

5.1. Navigate to **Virtualization** > **VirtualMachines**.

5.2. Click the vertical ellipsis to the right of the `template-create-vm` VM and select **Delete Virtual Machine**. Click **Delete** in the confirmation window to delete the VM.

Figure 6.18: Delete VM

5.3. Navigate to **Virtualization** > **Templates**.

5.4. Click the vertical ellipsis to the right of the `template-create` template row and select **Delete Template**. Click **Delete** in the confirmation window to delete the template.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 6.19: Delete template

Finish

On the `workstation` machine, change to `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish template-create
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Virtual Machine Template Management

Create two Red Hat Enterprise Linux 8 templates based on predefined images that contain a MariaDB server and an Apache HTTP server. You will then create two VMs from the templates and confirm that the VMs contain a database with preloaded data and a configured web server.

Outcomes

- Create two templates using the `mariadb-server` and `web-server` images by defining the attributes.
- Create a VM from the customized `mariadb-server` template and access it to confirm that the MariaDB service is up and running.
- Create a VM from the customized `web-server` template and access it to confirm that the web server is up and running.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that that the cluster API is reachable. It also creates the `template-review` namespace and creates the `mariadb-server` and `web-server` persistent volume claims in that namespace.

```
[student@workstation ~]$ lab start template-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 the `redhat` 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. Create a template named `mariadb-server` in the `template-review` namespace for VMs with a `Red Hat Enterprise Linux 8.0 or higher` operating system. Define the boot source as the `mariadb-server` PVC in the `template-review` namespace. Select the `Medium` flavor and the `Server` workload type for VM's created by the template. Define a 20 GiB `rootdisk` vbolume that uses the `ocs-external-storagecluster-ceph-rbd` storage class with a `ReadWriteMany` access mode and a `Block` volume mode. Configure the `cloud-init` service to create a `dbadmin` user with the `redhat` password on VMs created from the template.

3. Create a `mariadb-server-vm` VM from the custom `mariadb-server` template in the `template-review` namespace.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

4. Connect to the console of the `mariadb-server-vm` VM. Log in as the `dbadmin` user with the `redhat` password that you defined in a previous step. Connect to the MariaDB instance as the `devuser` with the `developer` password by using the `mysql -u devuser -p` command. Use the `SHOW DATABASES;` command to confirm that the preconfigured `sakila` database is available. Exit the MariaDB instance with the `EXIT;` command and then log out of the VM.

5. Create a `web-server` custom template in the `template-review` namespace for VMs with a `Red Hat Enterprise Linux 8.0 or higher` operating system. Define the boot source as the `web-server` PVC located within the `template-review` namespace. Select the `Medium` flavor and the `Server` workload type for VM's created by the template. Define a 15 GiB `rootdisk` volume that uses the `ocs-external-storagecluster-ceph-rbd` storage class with a `ReadWriteMany` access mode and a `Block` volume mode. Attach an additional 5 GiB disk named `tmpdata` that uses the `ocs-external-storagecluster-cephfs` storage class with `ReadWriteMany` access mode and a `Filesystem` volume mode. Configure the `cloud-init` service to create a `webadmin` user with the `redhat` password on VMs created from the template.

6. Create a `web-server-vm` VM from the `web-server` custom template in the `template-review` namespace.

7. Connect to the VNC console of the `web-server-vm` VM by using the `virtctl vnc` command. Log in as the `webadmin` user with the `redhat` password that 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. Afterwards, log out of the VM and then disconnect from the VNC console.

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 template-review
```

Finish

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

```
[student@workstation ~]$ lab finish template-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Virtual Machine Template Management

Create two Red Hat Enterprise Linux 8 templates based on predefined images that contain a MariaDB server and an Apache HTTP server. You will then create two VMs from the templates and confirm that the VMs contain a database with preloaded data and a configured web server.

Outcomes

- Create two templates using the `mariadb-server` and `web-server` images by defining the attributes.
- Create a VM from the customized `mariadb-server` template and access it to confirm that the MariaDB service is up and running.
- Create a VM from the customized `web-server` template and access it to confirm that the web server is up and running.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that that the cluster API is reachable. It also creates the `template-review` namespace and creates the `mariadb-server` and `web-server` persistent volume claims in that namespace.

```
[student@workstation ~]$ lab start template-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 the `redhat` 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 a terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 the `redhat` password when prompted.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. Create a template named `mariadb-server` in the `template-review` namespace for VMs with a `Red Hat Enterprise Linux 8.0 or higher` operating system. Define the boot source as the `mariadb-server` PVC in the `template-review` namespace. Select the `Medium` flavor and the `Server` workload type for VM's created by the template. Define a 20 GiB `rootdisk` vbolume that uses the `ocs-external-storagecluster-ceph-rbd` storage class with a `ReadWriteMany` access mode and a `Block` volume mode. Configure the `cloud-init` service to create a `dbadmin` user with the `redhat` password on VMs created from the template.

   2.1. Navigate to **Virtualization** > **Templates**. Select the `template-review` project from the **Projects** list.

   2.2. Click **Create** > **With Wizard** to open the **Create Virtual Machine template** wizard.

   2.3. Enter `mariadb-server` in the **Name** field and then enter `Red Hat Training` in the **Template provider** field. Select `Red Hat Enterprise Linux 8.0 or higher` from the **Operating System** list.

   2.4. Select `PVC (creates PVC)` from the **Boot Source** list. Select the `template-review` from the **Persistent Volume Claim project** list. Select the `PVC mariadb-server` from the **Persistent Volume Claim name** list.

   2.5. Select `Medium` from the **Flavor** list and `Server (default)` from the **Workload Type** list. Click **Next**.

   2.6. Confirm that the `nic-0` network interface is on the `Pod Networking` network and then click **Next**.

   2.7. Click the vertical ellipsis to the right of the `rootdisk` volume and then click **Edit**. In the **Size** field, modify the disk size to `20 GiB` and then confirm that the disk uses the `ocs-external-storagecluster-ceph-rbd` storage class. Confirm that the **Apply optimized StorageProfile settings** option is selected to configure the disk with a `ReadWriteMany` access mode and a `Block` volume mode. Click **Save** to apply the changes and then click **Next**.

   2.8. Use the **Cloud-init** form to define the `dbadmin` user with the `redhat` password. Click **Next** to review the custom template configuration.

   2.9. Review the template configuration and then click **Create Virtual Machine template**.

3. Create a `mariadb-server-vm` VM from the custom `mariadb-server` template in the `template-review` namespace.

   3.1. Navigate to **Virtualization** > **Templates**. Locate the row containing the `mariadb-server` template and then click **Create VM** on the right side of the row.

   3.2. Enter `mariadb-server-vm` in the **Virtual Machine Name** field. Confirm that **Start this virtual machine after creation** is selected. Click **Create virtual machine**. Wait a few moments for the operation to complete.

4. Connect to the console of the `mariadb-server-vm` VM. Log in as the `dbadmin` user with the `redhat` password that you defined in a previous step. Connect to the MariaDB instance as the `devuser` with the `developer` password by using the `mysql -u devuser -p` command. Use the `SHOW DATABASES;` command to confirm that the preconfigured `sakila` database is available. Exit the MariaDB instance with the `EXIT;` command and then log out of the VM.

   4.1. Navigate to **Virtualization** > **VirtualMachines**. Select the `mariadb-server-vm` to open the **Overview** tab for the VM.

4.2. Click the **Console** tab and then click **Guest login credentials** to confirm that the defined guest login credential is the `dbadmin` user with the `redhat` password.

4.3. Use the console to log in as the `dbadmin` user.

```
mariadb-server-vm login: dbadmin
password: redhat
[dbadmin@mariadb-server-vm ~]$
```

4.4. Connect to the MariaDB instance as the `devuser` with the `developer` password.

```
[dbadmin@mariadb-server-vm ~]$ mysql -u devuser -p
Enter password: developer
Welcome to the MariaDB monitor. Commands end with ; or \g.
...output omitted...
MariaDB [(none)]>
```

4.5. Confirm that the `sakila` database is available. Exit the MariaDB instance and then log out of the VM.

```
MariaDB [(none)]> SHOW DATABASES;
---------------------
| Database           |
---------------------
| information_schema |
| sakila             |
---------------------
2 rows in set (0.004 sec)

MariaDB [(none)]> EXIT;
Bye
[dbadmin@mariadb-server-vm ~]$ logout
```

5. Create a `web-server` custom template in the `template-review` namespace for VMs with a `Red Hat Enterprise Linux 8.0 or higher` operating system. Define the boot source as the `web-server` PVC located within the `template-review` namespace. Select the `Medium` flavor and the `Server` workload type for VM's created by the template. Define a 15 GiB `rootdisk` volume that uses the `ocs-external-storagecluster-ceph-rbd` storage class with a `ReadWriteMany` access mode and a `Block` volume mode. Attach an additional 5 GiB disk named `tmpdata` that uses the `ocs-external-storagecluster-cephfs` storage class with `ReadWriteMany` access mode and a `Filesystem` volume mode. Configure the `cloud-init` service to create a `webadmin` user with the `redhat` password on VMs created from the template.

5.1. Navigate to **Virtualization** > **Templates**. Select the `template-review` project from the **Projects** list.

5.2. Click **Create** > **With Wizard** to open the **Create Virtual Machine template** wizard.

5.3. Enter `web-server` in the **Name** field and then enter `Red Hat Training` in the **Template provider** field. Select `Red Hat Enterprise Linux 8.0 or higher` from the **Operating System** list.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5.4. Select `PVC (creates PVC)` from the **Boot Source** list. Select `template-review` from the **Persistent Volume Claim project** list. Select `PVC web-server` from the **Persistent Volume Claim name** list.

5.5. Select `Medium` from the **Flavor** list and `Server (default)` from the **Workload Type** list. Click **Next**.

5.6. Confirm that the `nic-0` network interface is on the `Pod Networking` network and then click **Next**.

5.7. Click the vertical ellipsis to the right of the `rootdisk` volume and then click **Edit**. In the **Size** field, modify the disk size to `15 GiB` and then confirm that the disk uses the `ocs-external-storagecluster-ceph-rbd` storage class. Confirm that the **Apply optimized StorageProfile settings** option is selected to configure the disk with a `ReadWriteMany` access mode and a `Block` volume mode. Click **Save** to apply the changes.

5.8. Click **Add disk** to create the `tmpdata` disk. Select `Blank (creates PVC)` from the **Source** list. Enter `tmpdata` in the **Name** field and enter `5 GiB` in the **Size** field. Select the `ocs-external-storagecluster-cephfs` from the **StorageClass** list. Confirm that the **Apply optimized StorageProfile settings** option is selected to configure the disk with a `ReadWriteMany` access mode and a `Filesystem` volume mode. Click **Add** and then click **Next**.

5.9. Use the **Cloud-init** form to define the `webadmin` user with the `redhat` password. Click **Next** to review the custom template configuration.

5.10. Review the template configuration and then click **Create Virtual Machine template**.

6. Create a `web-server-vm` VM from the `web-server` custom template in the `template-review` namespace.

6.1. Navigate to **Virtualization** > **Templates**. Locate the row containing the `web-server` template and then click **Create VM** on the right side of the row.

6.2. Enter `web-server-vm` in the **Virtual Machine Name** field. Confirm that **Start this virtual machine after creation** is selected. Click **Create virtual machine**. Wait a few moments for the operation to complete.

7. Connect to the VNC console of the `web-server-vm` VM by using the `virtctl vnc` command. Log in as the `webadmin` user with the `redhat` password that 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. Afterwards, log out of the VM and then disconnect from the VNC console.

7.1. In your terminal, connect to the console of the `web-server-vm` with `virtctl vnc` command.

```
[student@workstation ~]$ virtctl vnc web-server-vm -n template-review
```

7.2. In the console window, log in as the `webadmin` user with the `redhat` password.

```
web-server-vm login: webadmin
password: redhat
[webadmin@web-server-vm ~]$
```

7.3. Confirm that the `tmpdata` disk is available as the `vdc` block device.

```
[webadmin@web-server-vm ~]$ lsblk
NAME   MAJ:MIN  RM  SIZE RO TYPE
vda    252:0     0    1M  0 disk
vdb    252:16    0   30G  0 disk
...output omitted...
vdc    252:32    0    5G  0 disk
```

7.4. Confirm that the Apache `httpd` service is loaded and running. Press `q` to exit the status log.

```
[webadmin@web-server-vm ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor prese>
   Active: active (running) since Tue 2022-04-26 13:43:42 EDT; 5min ago
     Docs: man:httpd.service(8)
 Main PID: 922 (httpd)
   Status: "Running, listening on: port 80"
...output omitted...
```

7.5. Use the `curl` command to confirm that web server responds to requests. The command returns some HTML content.

```
[webadmin@web-server-vm ~]$ curl localhost
<!DOCTYPE html
...output omitted...
</html>
```

7.6. Log out of the VM and then close the VNC console.

```
[webadmin@web-server-vm ~]$ 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 template-review
```

Finish

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

```
[student@workstation ~]$ lab finish template-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- Red Hat OpenShift Virtualization provides virtual machine templates that are preconfigured with the operating system image, default settings for the operating system, workload type, and flavor, which specifies CPU and memory resources.
- When creating a custom template, you can configure default parameters for the template, such as the boot source, workload type, flavor, storage, network interfaces, and post-installation tasks. You can also provide a description of the template, include a message indicating how to use the template, and define the template support level and provider.
- To view the list of templates from the command line, use the `oc get templates -n openshift` command.
- *Containerized Data Importer (CDI)* provides the `DataVolume` *Custom Resource Definition(CRD)* to orchestrate the import, clone, and upload operations of PVCs for your VMs.
- OpenShift Virtualization allows you to create a new VM from a data volume, including cloning an existing PVC as a new data volume or uploading a local disk image.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 7

Advanced Virtual Machine Management

Goal

Snapshot, clone, and live migrate a virtual machine and initiate node maintenance.

Objectives

- Describe VM snapshots and execute snapshot management tasks.
- Clone an existing VM's PVC as a new VM and as a new data volume to provision additional VMs.
- Describe and manage VM live migrations.
- Set a node into maintenance mode and describe how the OpenShift Virtualization update process affects VMs.

Sections

- Managing Virtual Machine Snapshots (and Guided Exercise)
- Cloning Virtual Machines (and Guided Exercise)
- Performing Virtual Machine Live Migrations (and Guided Exercise)
- Node Maintenance and OpenShift Virtualization Updates (and Guided Exercise)

Lab

- Advanced Virtual Machine Management

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Managing Virtual Machine Snapshots

Objectives

- Describe VM snapshots and execute snapshot management tasks.

Take Snapshots of Virtual Machines

A *snapshot* is a view of a *Virtual Machine (VM)* 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 making changes to it. The snapshot can protect against errors that might have unintended consequences. If there is a problem, then the developer can revert the state of the VM recorded in the snapshot.

Red Hat OpenShift Virtualization allows you to 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 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 QEMU guest agent is available for Linux and Microsoft Windows systems.

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 must also support the snapshot feature. A link to the list of drivers and their snapshot support is provided in the references section.

If one VM disk relies on back-end storage that does not support snapshots, then OpenShift Virtualization displays a warning message when you take the snapshot. When you revert the snapshot, OpenShift Virtualization does not restore the data for that specific disk, which stays unchanged after the revert.

Red Hat does not support taking a live snapshot of a VM that has hot-plugged disks. Hot-plugged disks are disks that you attach to a running VM. You must stop the VM before taking the snapshot.

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 the snapshot, then OpenShift Virtualization restores the original configuration when you revert the snapshot.

Create Snapshots

Use the OpenShift web console to create a VM snapshot. Navigate to **Virtualization** > **VirtualMachines** and then select the VM to snapshot.

If you do not install the QEMU guest agent in the VM, then Red Hat recommends that you stop the VM before taking the snapshot. Click **Actions** > **Stop** and wait for the machine to stop.

Navigate to the **Snapshots** tab and then click **Take Snapshot** to initiate the snapshot. Complete the form and then click **Save**. Consider providing a meaningful name and description to facilitate snapshot identification.

A warning message displays if the VM includes disks that do not support snapshots, as shown in the following screen capture. The `nfs1` disk uses the storage class named `nfs-storage` that does not support snapshots.

Figure 7.1: 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.

You create a snapshot from the command line by declaring 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-2022-3-23` for the `mariadb-server` VM.

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineSnapshot
metadata:
  name: mariadb-server-2022-3-23
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: mariadb-server
```

Restore Snapshots

Before you can restore a snapshot, you must stop the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

After you stop the machine, navigate to the **Snapshots** tab and then locate the snapshot that you want to restore. You can click a snapshot name to access the description you provided at creation time.

To initiate the restoration process from the OpenShift web console, click **Restore** to the right of the snapshot. You can restart the VM after the restoration 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-2022-3-23 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 initiate the process. The following resource file restores the `mariadb-server-2022-3-23` 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-2022-3-23
```

Delete Snapshots

Snapshots consume disk space on back-end storage. To preserve disk space, delete the snapshots that you are no longer using.

To delete a snapshot, navigate to the **Snapshots** tab, click the vertical ellipsis to the right of the snapshot to delete, and then click **Delete VirtualMachineSnapshot**. You do not need to stop the VM to delete its snapshots.

From the command line, use the `oc delete` command. The following example deletes the `mariadb-server-2022-3-23` snapshot.

```
[user@host ~]$ oc delete virtualmachinesnapshot mariadb-server-2022-3-23
virtualmachinesnapshot.snapshot.kubevirt.io "mariadb-server-2022-3-23" deleted
```

Update the Virtual Machine Configuration

A VM snapshot includes the VM configuration. If you change the VM disk configuration after you have taken a snapshot, review the following considerations before restoring the snapshot:

- If you removed a disk, then OpenShift Virtualization recreates 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 present at the time you took the snapshot. These data are lost.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

- If you add a disk, then OpenShift Virtualization removes the disk when you restore the snapshot. If you used a data volume resource to create the associated PVC, by using the OpenShift web console, for example, then OpenShift Virtualization deletes the PVC and all its data.

Copy Kubernetes Volume Snapshots

In addition to restoring a complete VM from a snapshot, you can also 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 navigate to **Storage** > **VolumeSnapshots**. You can also run the `oc get volumesnapshots` command:

```
[user@host ~]$ oc get volumesnapshots
NAME                                                SOURCEPVC
vmsnapshot-0575...ce32-volume-dbdata          ...  mariadb-server-dbdata-dv01d
vmsnapshot-0575...ce32-volume-dbredo          ...  mariadb-server-dbredo-6zt45
vmsnapshot-0575...ce32-volume-dbundo          ...  mariadb-server-dbundo-etpi5
vmsnapshot-0575...ce32-volume-mariadb-server  ...  mariadb-server
```

The preceding output lists the Kubernetes volume snapshots and their associated PVC. With this information, you can identify which snapshot corresponds to which 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 to inspect the disk's content or to use the same initial data with several VMs.

To restore a Kubernetes volume snapshot and attach it to a VM, use the following steps:

- Navigate 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 to the right of the snapshot and then click **Restore as new PVC**. Complete the form and then click **Restore**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 7.2: Restoring a volume snapshot as a PVC

- Navigate to **Virtualization** > **VirtualMachines**, select the VM you want to attach the PVC, and then click **Actions** > **Stop** to stop the VM.
- Navigate to the **Disks** tab and then 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 **Add** to attach the disk.
- Click **Actions** > **Start** to start the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

References

For more information, refer to the *Installing VirtIO Driver on an Existing Windows Virtual Machine* and *Managing Virtual Machine Snapshots* sections in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

For more information about volume snapshots, refer to the *CSI Volume Snapshots* section in the *Red Hat OpenShift Container Platform Storage* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/storage/index#persistent-storage-csi-snapshots

Live Snapshots in OpenShift Virtualization
https://cloud.redhat.com/blog/live-snapshots-in-openshift-virtualization

CSI Drivers and Their Snapshot Support Status
https://kubernetes-csi.github.io/docs/drivers.html

Guided Exercise

Managing Virtual Machine Snapshots

Create a VM snapshot to restore a VM to a previous state. You will then use the snapshot to create a new VM.

Outcomes

- Take a VM snapshot
- Revert a snapshot
- Use volume snapshots as root file systems for new VMs

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `advanced-snapshot` namespace and starts the `vm1` virtual machine in that namespace.

```
[student@workstation ~]$ lab start advanced-snapshot
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console. Confirm that the 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 redhat \
  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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

   1.4. Navigate to **Virtualization** > **VirtualMachines**. Confirm that the `vm1` VM is running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. Confirm that the `vm1` VM runs the QEMU guest agent, review the machine configuration, and then take a live snapshot of the VM.

2.1. Select the `vm1` VM. In the **Overview** tab, under the **Status** section, notice that the status of the guest agent is `OK`. That status means that the agent is installed and running inside the VM.

2.2. Navigate to the **Details** tab. Under the **Scheduling and resources requirements** section, locate the **Flavor** subsection. Notice that the machine is assigned 1 CPU and 2 GiB of memory.

2.3. Navigate to the **Disks** tab. 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, navigate to the **Snapshots** tab and then click **Take Snapshot**. Enter `rhel-vm1-snap1` in the **Snapshot Name** field. Notice the warning message regarding the `cloudinitdisk` volume that Red Hat OpenShift Virtualization cannot snapshot. Select **I am aware of this warning and wish to proceed** and then click **Save**. Wait for the snapshot to report the status as `Ready`.

3. 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. You perform these file system updates as a test to confirm that OpenShift Virtualization restores the initial state of the VM when you revert the snapshot in a later step.

3.1. Navigate to the **Console** tab and then log in as the `root` user with the `redhat` password.

3.2. Review the current content of the `/etc/motd` file. The file is empty.

```
[root@vm1 ~]# cat /etc/motd
[root@vm1 ~]#
```

3.3. Add some content to the file.

```
[root@vm1 ~]# echo OpenShift Virtualization VM >> /etc/motd
```

3.4. Create the `/root/test.txt` file and then delete the `/etc/httpd/conf/magic` file.

```
[root@vm1 ~]# echo Hello World > /root/test.txt
[root@vm1 ~]# rm -f /etc/httpd/conf/magic
```

3.5. Log out of the VM console.

```
[root@vm1 ~]# logout
```

4. Configure the `vm1` VM with 2 CPUs and 4 GiB of memory. After reverting the snapshot, you will notice that OpenShift Virtualization restores the initial configuration.

4.1. Click **Actions** > **Stop**. Click **Stop** to confirm the operation. Wait for the machine to stop.

4.2. Navigate to the **Details** tab. Under the **Scheduling and resources requirements** section, locate the **Flavor** subsection and then click **Small: 1 CPU | 2 GiB Memory**.

4.3. Select `Custom` from the **Flavor** field. Set the **CPUs** field to 2 and set the **Memory** field to 4 GiB. Click **Save**.

5. Restore the VM snapshot and then confirm that OpenShift Virtualization reverts the VM to its initial state.

5.1. Navigate to the **Snapshots** tab and then click **Restore**. Click **Restore** to confirm the operation.

5.2. Click **Actions** > **Start** to start the VM.

5.3. Navigate to the **Console** tab and then log in as the `root` user with `redhat` as the password.

5.4. Review the content of the `/etc/motd` file. The file is empty and no longer contains your content. This was the state of the file when you took the snapshot.

```
[root@vm1 ~]# cat /etc/motd
[root@vm1 ~]#
```

5.5. List the content of the `/root` directory. The `test.txt` file does not exist anymore.

```
[root@vm1 ~]# ls /root
[root@vm1 ~]#
```

5.6. List the content of the `/etc/httpd/conf` directory. The `magic` file is restored.

```
[root@vm1 ~]# ls /etc/httpd/conf
httpd.conf  magic
```

5.7. Log out of the VM console.

```
[root@vm1 ~]# logout
```

5.8. Navigate to the **Details** tab. Under the **Scheduling and resources requirements** section, locate the **Flavor** subsection. Notice that the machine is assigned the initial 1 CPU and 2 GiB of memory.

6. Use a volume snapshot to copy the `vm1` disk and then use that copy as a root disk for a new VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Note

The following instructions demonstrate how to use a snapshot to make a copy of a volume.

Using that copy to create a new VM is not the best approach to duplicating a VM. OpenShift Virtualization provides a clone feature for that purpose. VM cloning is explained in greater detail elsewhere in this course.

6.1. Navigate to **Storage** > **VolumeSnapshots**. Click the vertical ellipsis to right of the snapshot where the source is the `vm1` PVC and then click **Restore as new PVC**. Complete the form by using the following information.

Click **Restore** when done.

Field	Value
Name	`root-copy`
StorageClass	`ocs-external-storagecluster-ceph-rbd`
Access mode	`Shared access (RWX)`

6.2. The `lab` command has prepared the `/home/student/DO316/labs/advanced-snapshot/vm2.yaml` file that 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: containerdisk
          persistentVolumeClaim:
            claimName: root-copy
```

6.3. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/advanced-snapshot/vm2.yaml
virtualmachine.kubevirt.io/vm2 created
```

If you prefer the OpenShift web console, navigate to **Virtualization** > **VirtualMachines** and then click **Create** > **With YAML**. Copy and then paste the `/home/student/DO316/labs/advanced-snapshot/vm2.yaml` file into the editor. Set the `claimName` parameter to `root-copy` and then click **Create**.

6.4. Navigate to **Virtualization** > **VirtualMachines** and then select the `vm2` VM. Navigate to the **Console** tab and then log in as the `root` user with the `redhat` password.

6.5. Run the `hostname` command and notice that the machine name is `vm1`. The machine has the same system configuration as the `vm1` VM because it uses an exact copy of its root disk.

```
[root@vm1 ~]# hostname
vm1
```

6.6. Log out of the VM console.

```
[root@vm1 ~]# logout
```

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 advanced-snapshot
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Cloning Virtual Machines

Objectives

- Clone an existing VM's PVC as a new VM and as a new data volume to provision additional VMs.

Clone Virtual Machines

A clone is a copy of a *virtual machine (VM)*. Usually, you carefully prepare a VM that you 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.

Be aware that a clone will likely have 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.

To prevent the clones from having these same identifying settings, you usually seal the original VM before using it to create clones. Sealing the VM is a process where you clear the machine-specific information.

An alternative to using clones is using templates and then performing the VM configuration with tools such as `cloud-init` or Ansible. This method starts with an initial operating system installation that you configure by using automated tools.

Prepare Virtual Machines for Cloning

After installing and configuring the VM that you want to use as a model for clones, remove all information unique to the VM. Unique information includes hardware information 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 there might be tools available to seal the image for you.

For Red Hat Enterprise Linux systems, use the `virt-sysprep` command. The command directly edits the disk block device to remove all information unique to the VM. You do not run the command from inside the VM itself but from a Linux system that has access to the VM disk block devices. You must stop the VM before using the `virt-sysprep` command.

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.

Red Hat OpenShift Virtualization provides the `virtctl guestfs` command to start a container that includes the `virt-sysprep` tool. The container also attaches the VM's PVC as the `/dev/vda` block device. The following example shows the execution of the `virt-sysprep` command to seal 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.
[root@libguestfs-tools-dsk1 /]# virt-sysprep -a /dev/vda
[   0.0] Examining the guest ...
[   3.4] Performing "abrt-data" ...
[   3.4] Performing "backup-files" ...
[   4.1] Performing "bash-history" ...
...output omitted...
[   4.7] Setting a random seed
[   4.7] Setting the machine ID in /etc/machine-id
[   4.9] Performing "lvm-uuids" ...
[root@libguestfs-tools-dsk1 /]# exit
exit
[user@host ~]$
```

Note

The first time the VM boots after sealing, the operating system recreates many of the items that the sealing tools stripped out.

As such, never start a VM that you have sealed. If you accidentally start the VM, then you have to go through the process of sealing the image again.

Clone Virtual Machines by Using the OpenShift Web Console

Use the OpenShift web console to create a VM clone:

- Navigate to **Virtualization** > **VirtualMachines**, select the VM you want to clone, and then stop the VM. OpenShift Virtualization can only clone stopped VMs.
- Click **Actions** > **Clone Virtual Machine** and then complete the form.
- Click **Clone Virtual Machine**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1. Name of the VM to create
2. Namespace of the new VM. To create the VM in a different namespace than the source VM, your cluster administrators must enable permissions to clone VMs across namespaces. The reference section provides a link that describes how to perform that configuration.
3. Summary of the VM configuration.

Clone VM Disks by Using Data Volumes

When OpenShift Virtualization clones a VM, it creates a `VirtualMachine` resource with the same characteristics as the source VM. The new `VirtualMachine` resource declares a `DataVolume` resource for each disk in the source VM. These `DataVolume` resources orchestrate the cloning operation.

The following example shows the `dataVolumeTemplates` section of the `VirtualMachine` resource for a cloned VM with two disks:

```
[user@host ~]$ oc get VirtualMachine front1 -o yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...output omitted...
spec:
  dataVolumeTemplates:  (1)
  - metadata:
      creationTimestamp: null
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
      name: front1-vm1-bvle4
    spec:
      pvc:  ❷
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 10Gi
        storageClassName: ocs-external-storagecluster-ceph-rbd  ❸
        volumeMode: Block
      source:  ❹
        pvc:
          name: vm1
          namespace: golden-vms
  - metadata:
      creationTimestamp: null
      name: front1-webroot-diaki
    spec:
      pvc:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: ocs-external-storagecluster-ceph-rbd
        volumeMode: Block
      source:
        pvc:
          name: webroot
          namespace: golden-vms
...output omitted...
```

❶ The `dataVolumeTemplates` section lists the data volumes for the VM disks. In the example, the section declares two disks.

❷ The `pvc` section provides the characteristics of the disk. These characteristics are the same as the disk from the source VM.

❸ The `storageClassName` parameter specifies the storage class of the new volume. OpenShift Virtualization creates a `StorageProfile` resource for each storage class. `StorageProfile` resources provide optimized parameters for VM disks.

❹ The `source` section specifies how to copy a disk image into the persistent volume. The `pvc` method instructs OpenShift Virtualization to clone the source PVC. The `name` and `namespace` parameters indicate the source PVC to clone.

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 into the destination volume. That copy method is usually slower than delegating the cloning process to the back-end storage.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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.

`snapshot`
: The cloning process relies on the back-end storage snapshot feature.

`copy`
: The cloning process does not rely on back-end storage. OpenShift Virtualization starts Kubernetes pods to perform the copy.

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.

`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
  source:  (2)
    pvc:
      name: documentroot
      namespace: golden-vms
```

1. The `storage` section provides the new disk characteristics. The disk size must be the same or larger than the source PVC.
2. The `source` section provides the details of the PVC to clone.

Run the `oc apply -f` *`resourcefile.yaml`* command to create the resource. If the VM using the source PVC is running, then the `DataVolume` resource stays in the `CloneScheduled` phase:

```
[user@host ~]$ oc apply -f documenrootDV.yaml
datavolume.cdi.kubevirt.io/documentroot-clone1 created
[user@host ~]$ oc get DataVolume
NAME                  PHASE            PROGRESS   RESTARTS   AGE
documentroot-clone1   CloneScheduled                         10m
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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

Knowledgebase: "Steps to Make a Clean VM for Use as a Template or Clone"
https://access.redhat.com/solutions/198693

For more information, refer to the *Creating a container using virtctl guestfs* and the *Cloning virtual machines* sections in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Cloning Virtual Machines

Clone an existing VM's PVC into new data volumes and then use the data volumes to create additional VMs.

Outcomes

- Clone VMs
- 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `advanced-clone` namespace and starts two virtual machines in that namespace: `golden-rhel` and `vm1`.

```
[student@workstation ~]$ lab start advanced-clone
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console. Confirm that the two VMs are 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 redhat \
  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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

   1.4. Navigate to **Virtualization** > **VirtualMachines**. Confirm that the `golden-rhel` and `vm1` VMs are running.

2. Clone the `golden-rhel` VM to a new VM named `web1`.

2.1. The cloning process requires that you stop the source machine.

Select the `golden-rhel` VM and then click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

2.2. Click **Actions** > **Clone**. Set the VM name to `web1`, select **Start virtual machine on clone**, and then click **Clone**.

2.3. Navigate to **Virtualization** > **VirtualMachines** and confirm that the new `web1` VM is running.

3. List the `DataVolume` resources associated with the `web1` VM. Delete the VM and then confirm that *Red Hat OpenShift Virtualization* also deletes the `DataVolume` resources and the underlying PVCs.

3.1. From the terminal on the `workstation` machine, use the `oc` command to list the `DataVolume` resources in the `advanced-clone` namespace.

```
[student@workstation ~]$ oc get DataVolume -n advanced-clone
NAME                      PHASE        PROGRESS   RESTARTS   AGE
golden-rhel               Succeeded    100.0%                19m
vm1                       Succeeded    100.0%                18m
web1-golden-rhel-doyls    Succeeded                          4m20s
web1-webroot-woern        Succeeded                          4m20s
webroot                   Succeeded    100.0%                19m
```

The cloning process prefixes the `DataVolume` resources with the VM name, `web1` in the preceding output.

3.2. List the PVCs. PVCs have the same name as the associated `DataVolume` resources.

```
[student@workstation ~]$ oc get pvc -n advanced-clone
NAME                      STATUS   VOLUME            CAPACITY   ACCESS MODES ...
golden-rhel               Bound    pvc-f874...367d   10Gi       RWX  ...
vm1                       Bound    pvc-de86...5ebb   10Gi       RWX  ...
web1-golden-rhel-doyls    Bound    pvc-de62...eac8   10Gi       RWX  ...
web1-webroot-woern        Bound    pvc-31d1...f37e   1Gi        RWX  ...
webroot                   Bound    pvc-2588...9f7d   1Gi        RWX  ...
```

3.3. Use the OpenShift web console to delete the `web1` VM. Navigate to **Virtualization** > **VirtualMachines**, select the `web1` VM, and then click **Actions** > **Delete Virtual Machine**. Select **Delete Disks (2x)** and then click **Delete**.

3.4. From the terminal, use the `oc` command to list the `DataVolume` and PVC resources. Notice that the `DataVolume` and PVC resources for the `web1` VM no longer exist.

```
[student@workstation ~]$ oc get DataVolume -n advanced-clone
NAME          PHASE        PROGRESS   RESTARTS   AGE
golden-rhel   Succeeded    100.0%                29m
vm1           Succeeded    100.0%                27m
webroot       Succeeded    100.0%                29m
[student@workstation ~]$ oc get pvc -n advanced-clone
NAME                      STATUS   VOLUME            CAPACITY   ACCESS MODES ...
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
golden-rhel              Bound    pvc-f874...367d   10Gi       RWX   ...
vm1                      Bound    pvc-de86...5ebb   10Gi       RWX   ...
webroot                  Bound    pvc-2588...9f7d   1Gi        RWX   ...
```

4. Clone the `webroot` PVC. The `golden-rhel` VM uses that PVC as an additional disk to store contents for the web server running inside the VM.

4.1. From the terminal, edit the `/home/student/DO316/labs/advanced-clone/dv.yaml` file that the `lab` command has prepared. The file defines a `DataVolume` resource named `documentroot-clone1`. Under 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: advanced-clone
spec:
  storage:
    resources:
      requests:
        storage: 1Gi
    storageClassName: ocs-external-storagecluster-ceph-rbd
  source:
    pvc:
      name: webroot
      namespace: advanced-clone
```

4.2. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/advanced-clone/dv.yaml
datavolume.cdi.kubevirt.io/documentroot-clone1 created
```

If you prefer the OpenShift web console, navigate to **Home** > **Search**, click **Resources**, and then select **(DV) DataVolume**. Click **Create DataVolume**. Copy and paste the `~/DO316/labs/advanced-clone/dv.yaml` resource file into the YAML editor. Under the `.spec.source.pvc` section, set the `name` parameter to `webroot`. Click **Create** to create the resource.

4.3. Use the `oc` command to confirm that OpenShift Virtualization creates the `DataVolume` and PVC resources.

```
[student@workstation ~]$ oc get DataVolume -n advanced-clone
NAME                  PHASE        PROGRESS   RESTARTS   AGE
documentroot-clone1   Succeeded                          30s
...output omitted...
[student@workstation ~]$ oc get pvc -n advanced-clone
NAME                  STATUS   VOLUME            CAPACITY   ACCESS MODES ...
documentroot-clone1   Bound    pvc-7bc3...f58d   1Gi        RWX   ...
...output omitted...
```

5. Attach the `documentroot-clone1` PVC to the `vm1` VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5.1. Navigate to **Virtualization** > **VirtualMachines**, select the `vm1` VM, and then click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

5.2. Navigate to the **Disks** tab and then click **Add disk**. Complete the form using the following information.

> **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.

Click **Add** to attach the disk to the VM.

Field	Value
Source	`Use an existing PVC`
PersistentVolumeClaim	`documentroot-clone1`
Name	`docroot`
Type	`Disk`
Interface	`virtio`

5.3. Click **Actions** > **Start** to start the VM.

5.4. Log in to the VM console and confirm that a new 1 GiB device is available. Navigate 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 `vdb` 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    1G  0 disk
```

5.5. Log out of the VM console.

```
[root@vm1 ~]# logout
```

6. 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.

6.1. Click **Actions** > **Delete Virtual Machine**. Select **Delete Disks (1x)** and then click **Delete**.

6.2. From the terminal, use the `oc` command to list the `DataVolume` and `PVC` resources. Notice that the `documentroot-clone1` data volume and the PVC still exist.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get DataVolume -n advanced-clone
NAME                  PHASE       PROGRESS   RESTARTS   AGE
documentroot-clone1   Succeeded                         59s
...output omitted...
[student@workstation ~]$ oc get pvc -n advanced-clone
NAME                  STATUS   VOLUME            CAPACITY   ACCESS MODES ...
documentroot-clone1   Bound    pvc-7bc3...f58d   1Gi        RWX  ...
...output omitted...
```

6.3. Delete the `documentroot-clone1` data volume and then confirm that OpenShift Virtualization automatically deletes the underlying PVC.

```
[student@workstation ~]$ oc delete DataVolume documentroot-clone1 \
  -n advanced-clone
datavolume.cdi.kubevirt.io "documentroot-clone1" deleted
[student@workstation ~]$ oc get pvc -n advanced-clone
NAME                      STATUS   VOLUME            CAPACITY   ACCESS MODES ...
golden-rhel               Bound    pvc-f874...367d   10Gi       RWX  ...
webroot                   Bound    pvc-2588...9f7d   1Gi        RWX  ...
```

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 advanced-clone
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Performing Virtual Machine Live Migrations

Objectives

- Describe and manage VM live migrations.

Describing Live Migrations

During maintenance or troubleshooting, you might need to migrate a *virtual machine (VM)* to a different cluster node. Live migration is the process of moving a running *virtual machine instance (VMI)* to a different cluster node without disrupting access or the virtual workload.

To perform a live migration, the VM must meet the following conditions:

- The underlying *persistent volume claim (PVC)* must use *ReadWriteMany (RWX)* access mode.
- The pod network is not 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.

During installation, the Red Hat OpenShift Virtualization operator creates a `kubevirt-storage-class-defaults` configuration map in the `openshift-cnv` namespace. You can modify the configuration map to configure the default access and volume modes for a particular storage class.

Use the following command to set the RWX access mode for a storage class:

```
[user@host ~]$ oc patch -n openshift-cnv cm kubevirt-storage-class-defaults \
-p '{"data":{"'$storage class'.accessMode":"ReadWriteMany"}}'
```

> **Note**
> Modifications to the `kubevirt-storage-class-defaults` configuration map affect new PVCs and do not impact existing PVCs. You cannot modify the access mode of an existing PVC.

You must enable the `sriovLiveMigration` feature gate in the `HyperConverged` *custom resource (CR)* to perform a live migration for VMs attached to an SR-IOV network interface.

Live Migrations Limitations

Default settings for live migration limits and timeouts can be adjusted 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 then save the file.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Key	Description	Default Value
`parallelMigrationsPerCluster`	The maximum number of migrations that can run in parallel in the cluster.	5
`parallelOutboundMigrationsPerNode`	The maximum number of parallel outbound migrations per node.	2
`bandwidthPerMigration`	Limits the bandwidth, in MiB/s, used for each migration.	64
`completionTimeoutPerGiB`	If the migration exceeds the defined time, in GiB/s of memory, the migration is canceled. The size of the migrating disks is included in the calculation if you use `BlockMigration` as the migration method.	800
`progressTimeout`	The migration is canceled if the memory copy fails to make progress in this time (in seconds).	150

Virtualization Node Placement

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 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 outlined below.

`nodeSelector`

VMs are scheduled on nodes labeled with the key-value pairs specified in this field. The node's labels must exactly match all of the key-value pairs listed.

For example, if you want a VM to run on a node that includes the two `example-key-1=example-value-1` and `example-key-2=example-value-2` labels, use the following metadata in your VM manifest:

```
metadata:
  name: example-vm-node-selector
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  template:
    spec:
      nodeSelector:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        example-key-1: example-value-1
        example-key-2: example-value-2
...output omitted...
```

In the event that the node does not meet all requirements, the VM is not scheduled.

`affinity`
: Use node affinity rules to schedule a VM to run on a group of nodes. You can also specify anti-affinity rules to prevent a VM from running on a particular group of nodes. To allow more flexibility during scheduling, you can specify if the affinity rule is required for the VM to run, or preferred but not required.

> **Note**
> Affinity rules only apply during scheduling; running VMs are not rescheduled even if the constraints are no longer met.

In this example, the affinity rule specifies that the VM must run on a node that has the `example.io/example-key=example-value-1` label or 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, but it can ignore the constraint in the event that none of the nodes have the `example-node-label-key=example-node-label-value` label.

```
metadata:
  name: example-vm-node-affinity
apiVersion: kubevirt.io/v1
kind: VirtualMachine
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...
```

`tolerations`
: A toleration specifies that a VM can be scheduled on a node if the VM's *taint* matches the taints on the node. However, if a VM is configured to tolerate a taint, then it is not required to schedule the VM onto a node configured with that taint.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

In this example, the nodes reserved for VM workloads are labeled with the `key=virtualization:NoSchedule` taint. The VM is configured with a matching toleration and is thus able to run on the nodes with that taint.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-tolerations
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "virtualization"
    effect: "NoSchedule"
...output omitted...
```

Starting, Monitoring, and Canceling Live Migrations

As a cluster administrator, you can also manually migrate *virtual machine instances (VMIs)*.

To start a live migration by using the Red Hat OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select your project from the **Project** list. Locate the row containing the VM to migrate and then click the vertical ellipsis menu to the right of the row. Click **Migrate Node to Node** and then click **Migrate Node to Node** to confirm the operation and begin the live migration.

You can also click the VM's name to navigate to the VM's **Overview** tab. Click **Actions** > **Migrate Node to Node** and then click **Migrate Node to Node** to confirm and begin migration.

To perform a migration from the command line, 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 web-servers` command to initiate the live migration of the VMI specified in the manifest. The status of a live migration can be viewed in the web console by navigating to the **Overview** tab or the **Details** tab of the VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

You can use the command line to monitor the status of a VM's live migration with the `oc describe vmi` *`vmi-name`* command or with the `oc get vmim` command.

To cancel a live migration in the web console, navigate to **Virtualization** > **VirtualMachines**. Select your project from the **Project** list. Click the vertical ellipsis to the right of the VM you want to migrate and then click **Cancel Virtual Machine Migration**. Click **Cancel Migration** to confirm the operation.

To cancel a live migration from the command line, you can delete the `VirtualMachineInstanceMigration` object associated with the migration with the `oc delete vmim` *`migration-job`* command.

References

For more information, refer to the *Virtual Machine Live Migration* section of the *Red Hat OpenShift Container Platform Virtualization Guide* at https://docs.openshift.com/container-platform/4.10/virt/live_migration/virt-live-migration.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Performing Virtual Machine Live Migrations

Live migrate a VM to another node in the cluster.

Outcomes

- Observe and set the access mode for a VM's storage class.
- Initiate a live migration through the web console and command line.
- Monitor the progress of live migrations.
- 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `advanced-migrate` namespace and starts the `vm1` virtual machine in that namespace.

```
[student@workstation ~]$ lab start advanced-migrate
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console. Confirm that the 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 redhat \
  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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

   1.4. Navigate to **Virtualization** > **VirtualMachines**. Select the `advanced-migrate` project from the **Project** list. Confirm that the `vm1` VM is running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. Attempt a live migration of the `vm1` VM. Determine the storage class of the VM's empty `tempdata` disk.

2.1. Click the vertical ellipsis to the right of the `vm1` VM and then click **Migrate Node to Node**. Click **Migrate Node to Node** to confirm the operation.

The live migration fails because the empty `tempdata` PVC is not configured with a `ReadWriteMany` access mode. Click **Cancel** to end the operation.

2.2. Identify the storage class for the `tempdata` disk. Select the `vm1` VM. Navigate to the **Disks** tab. Notice that the `tempdata` disk uses the `nfs-storage` storage class.

3. Retrieve the current default storage class settings for Red Hat OpenShift Virtualization. Use the `oc patch` command to configure the `nfs-storage` storage class to use `ReadWriteMany` access mode.

3.1. From a terminal, execute the `oc get` command to retrieve the `kubevirt-storage-class-defaults` configuration map in the `openshift-cnv` namespace. Notice that the `nfs-storage` storage class defaults to a `ReadWriteOnce` access mode.

You can copy and then paste the command from the `~/DO316/labs/advanced-migrate/commands.txt` file that the `lab` command prepared.

```
[student@workstation ~]$ oc get cm kubevirt-storage-class-defaults \
  -n openshift-cnv -o yaml
apiVersion: v1
data:
  accessMode: ReadWriteOnce
  local-sc.accessMode: ReadWriteOnce
  local-sc.volumeMode: Filesystem
  nfs-storage.accessMode: ReadWriteOnce
  ocs-storagecluster-ceph-rbd.accessMode: ReadWriteMany
  ocs-storagecluster-ceph-rbd.volumeMode: Block
  volumeMode: Filesystem
kind: ConfigMap
metadata:
...output omitted...
```

3.2. Change the `nfs-storage` access mode to `ReadWriteMany` by using the `oc patch` command.

You can copy and then paste the command from the `~/DO316/labs/advanced-migrate/commands.txt` file.

```
[student@workstation ~]$ oc patch -n openshift-cnv cm \
  kubevirt-storage-class-defaults -p \
  '{"data":{"nfs-storage.accessMode":"ReadWriteMany"}}'
configmap/kubevirt-storage-class-defaults patched
```

3.3. Confirm that the `kubevirt-storage-class-defaults` configuration map is updated with the correct access mode.

```
[student@workstation ~]$ oc get cm kubevirt-storage-class-defaults \
  -n openshift-cnv -o yaml
apiVersion: v1
data:
  accessMode: ReadWriteOnce
  local-sc.accessMode: ReadWriteOnce
  local-sc.volumeMode: Filesystem
  nfs-storage.accessMode: ReadWriteMany
  ocs-storagecluster-ceph-rbd.accessMode: ReadWriteMany
  ocs-storagecluster-ceph-rbd.volumeMode: Block
  volumeMode: Filesystem
kind: ConfigMap
...output omitted...
```

4. Delete the empty `tempdata` disk from the `vm1` VM. Attach a new 2 GiB blank disk named `newtempdata` that uses the `nfs-storage` storage class.

4.1. You must stop the `vm1` VM before you can modify the VM's disks. In the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `vm1` VM and then click **Actions** > **Stop**. Click **Stop** and wait for the VM to stop.

4.2. Navigate to the **Disks** tab. Click the vertical ellipsis to the right of the `tempdata` disk and then click **Delete**. Confirm that the **Delete tempdata Data Volume and PVC** option is selected and then click **Detach** to confirm the operation.

4.3. Click **Add disk** to create a new disk. Select **Blank (creates PVC)** as the **Source** and then enter `newtempdata` for the **Name**. Set the **Size** to `2 GiB` and leave `virtio` as the **Interface**. Confirm that the **StorageClass** is set to **nfs-storage** and click **Add** to add the disk.

4.4. Click **Actions** > **Start** to power on the VM. Wait a few moments for the operation to complete.

5. Live migrate the `vm1` VM in the OpenShift web console. Afterwards, initiate and then cancel another live migration for the VM.

5.1. In the web console, click the **Overview** tab. In the **Details** section, make note of the **Node** that is hosting the VM.

5.2. Click **Actions** > **Migrate Node to Node**. Click **Migrate Node to Node** to confirm the operation. Wait a few moments for the migration to complete.

5.3. In the **Overview** tab, confirm that the **Details** section lists a different **Node** than identified in an earlier step.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5.4. Initiate another live migration for the VM. Click **Actions** > **Migrate Node to Node**. Click **Migrate Node to Node** to confirm the operation.

5.5. Cancel the live migration. Click **Actions** > **Cancel Virtual Machine Migration**. Click **Cancel Migration** to confirm the operation.

6. Initiate another live migration for the `vm1` VM by using a `VirtualMachineInstanceMigration` manifest.

6.1. In a text editor, modify the `migrate-vmi.yml` file located in the `~/DO316/labs/advanced-migrate/` directory.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstanceMigration
metadata:
  name: vm1-migration-job
spec:
  vmiName: vm1
```

6.2. In your terminal, use the `oc create` command to initiate the live migration with the `migrate-vmi.yml` manifest.

```
[student@workstation ~]$ oc create -f \
  ~/DO316/labs/advanced-migrate/migrate-vmi.yml -n advanced-migrate
virtualmachineinstancemigration.kubevirt.io/vm1-migration-job created
```

6.3. Monitor the live migration with the `oc describe vmi` command. Confirm that `vm1` is migrated to another node and make note of the node's name. You might have to wait a few minutes and rerun the command several times.

```
[student@workstation ~]$ oc describe vmi vm1 -n advanced-migrate
...output omitted...
Events:
  Type    Reason          Age   From          Message
  ----    ------          ----  ----          -------
...output omitted...
  Normal  PreparingTarget 57s   virt-handler  Migration Target is listening at
 10.9.0.37, on ports: 45195,36331,33211
  Normal  PreparingTarget 57s   virt-handler  VirtualMachineInstance Migration
 Target Prepared.
  Normal  Deleted         53s   virt-handler  Signaled Deletion
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
  Normal  Migrating       53s   virt-handler  VirtualMachineInstance is migrating.
  Normal  Migrated        53s   virt-handler  The VirtualMachineInstance migrated
 to node master03.
...output omitted...
```

7. Use the command line to initiate and then cancel a live migration for the `vm1` VM.

   7.1. In your terminal, delete the previous `vm1-migration-job` with the `oc delete vmim` command.

```
[student@workstation ~]$ oc delete vmim vm1-migration-job -n advanced-migrate
virtualmachineinstancemigration.kubevirt.io "vm1-migration-job" deleted
```

   7.2. 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 -n advanced-migrate
virtualmachineinstancemigration.kubevirt.io/vm1-migration-job created
```

   7.3. Cancel the live migration by deleting the `VirtualMachineInstanceMigration` object.

```
[student@workstation ~]$ oc delete vmim vm1-migration-job -n advanced-migrate
virtualmachineinstancemigration.kubevirt.io "vm1-migration-job" deleted
```

   7.4. Confirm that the VM is not migrated to another node.

```
[student@workstation ~]$ oc describe vmi vm1 -n advanced-migrate
...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.
  Warning  Migrated        82s   virt-handler  VirtualMachineInstance migration
 uid 3d72c630-91a1-443c-9904-29e410a0d165 failed. reason:Live migration aborted
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ lab finish advanced-migrate
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Node Maintenance and OpenShift Virtualization Updates

Objectives

- Set a node into maintenance mode and describe how the OpenShift Virtualization update process affects VMs.

Move a Node into Maintenance Mode

To prepare a node for maintenance, cluster administrators must perform two operations:

- *Cordon* off the node to prevent the cluster from deploying new workloads on the node.
- *Drain* the node to move the current workload, including pods and virtual machines (VMs), to the remaining nodes.

Create a Node Maintenance Resource

As a cluster administrator, you set a node in maintenance mode and move its workload to other nodes by creating a `NodeMaintenance` resource.

The following example shows the resource file that sets the `node2` node in maintenance mode:

```
apiVersion: nodemaintenance.kubevirt.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.

If you prefer the OpenShift web console, navigate to **Home** > **Search**, click **Resources**, and then select **(NM) NodeMaintenance**. Click **Create Node Maintenance**. Write the resource definition into the YAML editor and then click **Create**.

You can follow the process by using the `oc describe NodeMaintenance` command:

```
[user@host ~]$ oc describe NodeMaintenance maintenance-node2
...output omitted...
Status:
  Eviction Pods:  29
  Phase:          Succeeded
  Totalpods:      47
...output omitted...
```

To verify that the node does not accept new workloads, confirm that the node has the `SchedulingDisabled` status:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[user@host ~]$ oc get nodes
NAME    STATUS                     ROLES           AGE   VERSION
...output omitted...
node1   Ready                      worker          13d   v1.23.3+e419edf
node2   Ready,SchedulingDisabled   worker          13d   v1.23.3+e419edf
```

To get that status from the OpenShift web console, navigate to **Compute** > **Nodes**. The node displays a **Scheduling disable** status.

To remove the node from maintenance mode and make it ready to accept new workloads, delete the `NodeMaintenance` resource:

```
[user@host ~]$ oc delete NodeMaintenance maintenance-node2
nodemaintenance.nodemaintenance.kubevirt.io "maintenance-node2" deleted
```

Red Hat OpenShift Container Platform (RHOCP) does not move the pods and *virtual machines (VMs)* that were running on the node before maintenance back to the original node.

Move a Node into Maintenance from the Command Line

As an alternative to creating a `NodeMaintenance` resource, you can use the `oc` command.

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
```

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, RHOCP restarts the pod or VM on the other nodes with a new empty volume.
2. With the `--ignore-daemonsets` option, RHOCP skips moving the Daemon Set pods.
3. The `--force` option prevents the drain operation from failing when some pods that RHOCP does not manage are running on the node.

To remove the node from maintenance mode, use the `oc adm uncordon` command:

```
[user@host ~]$ oc adm uncordon node2
node/node2 uncordoned
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Live Migrate Virtual Machine During Node Drain

When you create a VM by using a template, *Red Hat OpenShift Virtualization* sets the `evictionStrategy` resource parameter to `LiveMigrate`:

```
[user@host ~]$ oc get vm mariadb-server -o yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...output omitted...
spec:
  ...output omitted...
  template:
    ...output omitted...
    spec:
      ...output omitted...
      evictionStrategy: LiveMigrate
      hostname: mariadb-server
...output omitted...
```

During node drain, OpenShift Virtualization performs live migration for these VMs. For VMs without the `evictionStrategy` parameter, OpenShift Virtualization shuts down the VMs and then restarts them on another node.

Important

Even though the `evictionStrategy` parameter is set, some VMs might not support live migration. For example, OpenShift Virtualization cannot perform live migration for VMs using storage that does not support the `ReadWriteMany` (RWX) access mode.

These VMs prevents the drain process from completing. The operation is blocked.

To verify whether a VM supports live migration, navigate to **Virtualization** > **VirtualMachines** and then select the VM. A warning message displays in the **Overview** tab under the **Status** section when the VM does not support live migration.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 7.6: Verifying the VM live migration status

For these VMs, remove the `evictionStrategy` parameter to prevent the drain process from blocking. To remove the parameter, stop the VM, navigate to the **Details** tab, and then scroll down to the **Scheduling and resource requirements** section. Remove the **LiveMigrate** flag from the **Eviction Strategy** parameter.

Update Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform 4 adds a new software distribution system that provides the best upgrade path to update your cluster and the underlying operating system. This new distribution system enables clusters to upgrade directly from the internet. It provides all the resources necessary to update a cluster to a particular version. It 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. CVO provides the current component versions, and OSUS computes and returns an update path.

OSUS computes the update path from the *channel* you select for your cluster. A channel corresponds to a specific RHOCP version.

Navigate to **Administration** > **Cluster Settings** to manage your cluster updates. The following screen capture shows the path to update the current 4.9.10 cluster to RHOCP 4.10. After the cluster administrator selects the `stable-4.10` channel, CVO automatically contacts OSUS to get the update path. The path shows that to migrate from 4.9.10, which is the current cluster version, to 4.10, you must first update your cluster to version 4.9.27.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Figure 7.7: Updating from 4.9.10 to 4.10

The *Machine Config Operator (MCO)* updates the operating system on the cluster nodes during the update process. During the update, MCO cordons off and drains the nodes to keep the pods and VMs running.

Update Red Hat OpenShift Virtualization

For Red Hat to support your installation, the version of the OpenShift Virtualization operator must match the version of RHOCP. By default, RHOCP automatically updates the operator.

Navigate to **Operators** > **Installed Operators** and then click **OpenShift Virtualization** to access the operator details. From the **Subscription** tab, review the **Update approval** section to confirm that RHOCP automatically applies the updates.

Figure 7.8: Confirming operator automatic update

Red Hat recommends using the automatic update process.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 performs a live migration of the VMs. During the live migration, OpenShift Virtualization starts new updated `virt-launcher` pods.

If a VM does not support live migration, then OpenShift Virtualization does not update the pod. OpenShift Virtualization will update the pod the next time you restart the VM.

References

For more information about node maintenance, refer to the *Using the Node Maintenance Operator to Place Nodes in Maintenance Mode* section in the *Red Hat OpenShift Container Platform Nodes* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#node-maintenance-operator

For more information about virtual machines and node maintenance, refer to the *Node Maintenance* chapter in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#node-maintenance

For more information about updating RHOCP, refer to the *Red Hat OpenShift Container Platform Updating Clusters* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/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://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#updating-openshift-virtualization

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Node Maintenance and OpenShift Virtualization Updates

Set worker nodes into maintenance mode, drain the workloads from those nodes, and observe the maintenance mode 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 system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `advanced-maintenance` namespace and starts the `vm1` virtual machine in that namespace.

```
[student@workstation ~]$ lab start advanced-maintenance
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console. Confirm that the 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 redhat \
  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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1.4. Navigate to **Virtualization** > **VirtualMachines**. Confirm that the `vm1` VM is running.

2. To prepare for maintenance mode, verify that *Red Hat OpenShift Virtualization* can live migrate the `vm1` VM. Retrieve the name of the node that runs the VM.

2.1. Select the `vm1` VM. In the **Overview** tab, under the **Status** section, notice that there are no warning events stating that the VM is not migratable.

Figure 7.9: Verifying that OpenShift Virtualization can live migrate the VM

2.2. Under the **Details** section, retrieve the name of the node currently running the VM. It should be `worker01`, `worker02`, or one of the master node.

3. Cordon off the node running the VM.

3.1. From the terminal on the `workstation` machine, use the `oc adm cordon` command to disable scheduling. In the following command, replace the `worker02` node with the node's name running your VM.

```
[student@workstation ~]$ oc adm cordon worker02
node/worker02 cordoned
```

If you prefer the OpenShift web console, then navigate to **Compute** > **Nodes**, click the vertical ellipsis to the right of the node, and then click **Mark as unschedulable**. Click **Mark unschedulable** to confirm the operation.

3.2. Confirm that the node has the `SchedulingDisabled` status.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get nodes
NAME       STATUS                     ROLES           AGE   VERSION
master01   Ready                      master,worker   12d   v1.23.3+e419edf
master02   Ready                      master,worker   12d   v1.23.3+e419edf
master03   Ready                      master,worker   12d   v1.23.3+e419edf
worker01   Ready                      worker          12d   v1.23.3+e419edf
worker02   Ready,SchedulingDisabled   worker          12d   v1.23.3+e419edf
```

If you prefer the OpenShift web console, navigate to **Compute** > **Nodes** and confirm that the node has the **Scheduling disabled** status.

4. Drain the node of its workload.

4.1. From the terminal, run the `oc adm drain` command. In the following command, replace the `worker02` node with the node's name running your VM. The command might take a few minutes to complete.

You can copy and then paste the command from the `/home/student/DO316/labs/advanced-maintenance/commands.txt` file that the `lab` command has prepared for you.

```
[student@workstation ~]$ oc adm drain worker02 --delete-emptydir-data \
  --ignore-daemonsets --force
node/worker02 already cordoned
WARNING: ignoring DaemonSet-managed Pods: openshift-cluster-node-tuning-operator/
tuned-chfvb, openshift-cnv/bridge-marker-q9gkr, openshift-cnv/kube-cni-linux-
bridge-plugin-w7k7z, openshift-cnv/nmstate-handler-4pmc4, openshift-cnv/virt-
handler-nvvpk, openshift-dns/dns-default-bcd9w, openshift-dns/node-resolver-
tjgmt, openshift-image-registry/node-ca-sxm9q, openshift-ingress-canary/ingress-
canary-wmb2t, openshift-machine-config-operator/machine-config-daemon-qs76k,
 openshift-monitoring/node-exporter-7c6kx, openshift-multus/multus-additional-cni-
plugins-snlsq, openshift-multus/multus-q2c7z, openshift-multus/network-metrics-
daemon-qqvf4, openshift-network-diagnostics/network-check-target-4l5d4, openshift-
sdn/sdn-rzwt4, openshift-storage/csi-cephfsplugin-r44dh, openshift-storage/csi-
rbdplugin-bg2vb
...output omitted...
pod/virt-api-84b8c9df4b-57hbd evicted
evicting pod storage-intro/virt-launcher-vm1-d2bbv
error when evicting pods/"virt-launcher-vm1-d2bbv" -n "storage-intro" (will retry
 after 5s): Cannot evict pod as it would violate the pod's disruption budget.
evicting pod storage-intro/virt-launcher-vm1-d2bbv
error when evicting pods/"virt-launcher-vm1-d2bbv" -n "storage-intro" (will retry
 after 5s): Cannot evict pod as it would violate the pod's disruption budget.
evicting pod storage-intro/virt-launcher-vm1-d2bbv
pod/virt-launcher-vm1-d2bbv evicted
pod/cdi-uploadproxy-7568555574-ww4v6 evicted
pod/router-default-699bc7764c-k55lp evicted
node/worker02 drained
```

Ignore the error messages regarding the failed eviction of the `virt-launcher` pod. OpenShift Virtualization cannot delete the pod until its migration is complete and a new `virt-launcher` pod is running on another node.

4.2. Confirm that the VM is now running on another node.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get vmi -o wide -n advanced-maintenance
NAME    AGE    PHASE     IP           NODENAME    READY    LIVE-MIGRATABLE    PAUSED
vm1     163m   Running   10.11.0.40   worker01    True     True
```

If your VM was initially running on the `worker01` node, then it should now run on the `worker02` node or one of the master nodes.

5. Resume the node from maintenance mode.

5.1. From the terminal, 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 you set in maintenance mode.

```
[student@workstation ~]$ oc adm uncordon worker02
node/worker02 uncordoned
```

If you prefer the OpenShift web console, then navigate to **Compute** > **Nodes**, click the vertical ellipsis to the right of the node, and then click **Mark as schedulable**.

5.2. Confirm that the node is not in maintenance mode anymore.

```
[student@workstation ~]$ oc get nodes
NAME        STATUS    ROLES           AGE   VERSION
master01    Ready     master,worker   12d   v1.23.3+e419edf
master02    Ready     master,worker   12d   v1.23.3+e419edf
master03    Ready     master,worker   12d   v1.23.3+e419edf
worker01    Ready     worker          12d   v1.23.3+e419edf
worker02    Ready     worker          12d   v1.23.3+e419edf
```

If you prefer the OpenShift web console, navigate to **Compute** > **Nodes** and confirm that the node does not have the **Scheduling disabled** status anymore.

6. As an alternative to the `oc adm cordon` and `oc adm drain` commands that you used in the preceding steps, create a `NodeMaintenance` resource to set the `worker02` node in maintenance mode and drain its workload.

6.1. From the terminal, edit the `/home/student/DO316/labs/advanced-maintenance/nm.yaml` file that the `lab` command has prepared. The file defines a `NodeMaintenance` resource named `maintenance-node`. Under the `spec` section, update the `nodeName` parameter to specify the `worker02` node.

```
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
metadata:
  name: maintenance-node
spec:
  nodeName: worker02
  reason: "Maintenance required for the node"
```

6.2. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/advanced-maintenance/nm.yaml
nodemaintenance.nodemaintenance.kubevirt.io/maintenance-node created
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

If you prefer the OpenShift web console, navigate to **Home** > **Search**, click **Resources**, and then select **(NM) NodeMaintenance**. Click **Create Node Maintenance**. Copy and paste the `~/DO316/labs/advanced-maintenance/nm.yaml` resource file into the YAML editor. Under the `spec` section, set the `nodeName` parameter to `worker02`. Click **Create** to create the resource.

6.3. Confirm that the node has the `SchedulingDisabled` status.

```
[student@workstation ~]$ oc get nodes
NAME       STATUS                     ROLES           AGE   VERSION
master01   Ready                      master,worker   12d   v1.23.3+e419edf
master02   Ready                      master,worker   12d   v1.23.3+e419edf
master03   Ready                      master,worker   12d   v1.23.3+e419edf
worker01   Ready                      worker          12d   v1.23.3+e419edf
worker02   Ready,SchedulingDisabled   worker          12d   v1.23.3+e419edf
```

If you prefer the OpenShift web console, navigate to **Compute** > **Nodes** and confirm that the node has the **Scheduling disabled** status.

6.4. Monitor the `NodeMaintenance` resource named `maintenance-node` to confirm that the drain operation completes. You might have to wait a few minutes and rerun the command several times to get the `Succeeded` status.

```
[student@workstation ~]$ oc get NodeMaintenance maintenance-node -o yaml
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
...output omitted...
status:
  evictionPods: 4
  phase: Succeeded
  totalpods: 22
```

6.5. Delete the `NodeMaintenance` resource to resume the node from maintenance mode.

```
[student@workstation ~]$ oc delete NodeMaintenance maintenance-node
nodemaintenance.nodemaintenance.kubevirt.io "maintenance-node" deleted
```

6.6. Confirm that the node is not in maintenance mode anymore.

```
[student@workstation ~]$ oc get nodes
NAME       STATUS    ROLES           AGE   VERSION
master01   Ready     master,worker   12d   v1.23.3+e419edf
master02   Ready     master,worker   12d   v1.23.3+e419edf
master03   Ready     master,worker   12d   v1.23.3+e419edf
worker01   Ready     worker          12d   v1.23.3+e419edf
worker02   Ready     worker          12d   v1.23.3+e419edf
```

If you prefer the OpenShift web console, navigate to **Compute** > **Nodes** and confirm that the node does not have the **Scheduling disabled** status anymore.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 advanced-maintenance
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Advanced Virtual Machine Management

Create a VM snapshot to deploy a new VM, create a virtual machine by cloning an existing PVC, perform live migrations, and initiate node maintenance.

Outcomes

- Clone VMs.
- Take snapshots of VMs.
- Live migrate VMs.
- Set cluster nodes in maintenance mode.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `advanced-review` namespace and starts the `golden-rhel` virtual machine in that namespace.

```
[student@workstation ~]$ lab start advanced-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 the `redhat` 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 `golden-rhel` VM is running under the `advanced-review` project.

2. Clone the `golden-rhel` VM to a new `www1` VM. Start the `www1` VM.

   If you need to access the VM's console, then log in as the `root` user with the `redhat` password.

3. Take a snapshot of the `www1` VM. The name of the snapshot must be `www1-snap1`.

4. The snapshot that you take in the preceding step creates a volume snapshot for each VM disk. Restore the volume snapshot of the `www1-golden-rhel-`*`xxxxx`* PVC and then use that new PVC as the root disk for a new VM.

   Use the following information when restoring the volume snapshot as a new PVC:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Field	Value
Name	`root-copy`
StorageClass	`ocs-external-storagecluster-ceph-rbd`
Access mode	`Shared access (RWX)`

The `lab` command prepared the `/home/student/DO316/labs/advanced-review/www2.yaml` file that declares a `VirtualMachine` resource. Edit the file and set the `root-copy` PVC as the root disk. Create the `www2` VM from the resource file.

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 only contains temporary data that you can safely discard).

6. Set the `worker02` cluster node in maintenance mode and move its workload to the remaining nodes.

   The `lab` command has prepared the `/home/student/DO316/labs/advanced-review/nm.yaml` resource file that you can use as a model. If you decide to use that file, then you need to adapt it to the requirements.

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

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 advanced-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Advanced Virtual Machine Management

Create a VM snapshot to deploy a new VM, create a virtual machine by cloning an existing PVC, perform live migrations, and initiate node maintenance.

Outcomes

- Clone VMs.
- Take snapshots of VMs.
- Live migrate VMs.
- Set cluster nodes in maintenance mode.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `advanced-review` namespace and starts the `golden-rhel` virtual machine in that namespace.

```
[student@workstation ~]$ lab start advanced-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 the `redhat` 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 `golden-rhel` VM is running under the `advanced-review` project.

   1.1. From a terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 the `redhat` password when prompted.

   1.3. Navigate to **Virtualization** > **VirtualMachines** and then select the `advanced-review` project. Confirm that the `golden-rhel` VM is running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. Clone the `golden-rhel` VM to a new `www1` VM. Start the `www1` VM.

If you need to access the VM's console, then log in as the `root` user with the `redhat` password.

2.1. The cloning process requires that you stop the source machine.

Select the `golden-rhel` VM and then click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

2.2. Click **Actions** > **Clone**. Set the VM name to `www1`, select **Start virtual machine on clone**, and then click **Clone**.

2.3. Navigate to **Virtualization** > **VirtualMachines** and confirm that the new `www1` VM is running.

3. Take a snapshot of the `www1` VM. The name of the snapshot must be `www1-snap1`.

Select the `www1` VM, navigate to the **Snapshots** tab, and then click **Take Snapshot**.

Enter `www1-snap1` in the **Snapshot Name** field. Select **I am aware of this warning and wish to proceed** and then click **Save**.

Wait for the snapshot to report a `Ready` status.

4. The snapshot that you take in the preceding step creates a volume snapshot for each VM disk. Restore the volume snapshot of the `www1-golden-rhel-`*`xxxxx`* PVC and then use that new PVC as the root disk for a new VM.

Use the following information when restoring the volume snapshot as a new PVC:

Field	Value
Name	`root-copy`
StorageClass	`ocs-external-storagecluster-ceph-rbd`
Access mode	`Shared access (RWX)`

The `lab` command prepared the `/home/student/DO316/labs/advanced-review/www2.yaml` file that declares a `VirtualMachine` resource. Edit the file and set the `root-copy` PVC as the root disk. Create the `www2` VM from the resource file.

4.1. Navigate to **Storage** > **VolumeSnapshots**. Click the vertical ellipsis to the right of the snapshot where the source is the `www1-golden-rhel-`*`xxxxx`* PVC and then click **Restore as new PVC**.

> **Note**
> If you cannot identify the source PVC from the list of volume snapshots, select a volume snapshot from the list to open the **Details** tab for the snapshot. Locate the `Source` field to confirm that the identified source is the `www1-golden-rhel-`*`xxxxx`* PVC and then click **Actions** > **Restore as new PVC**.

Complete the form by using the information given in the instructions. Click **Restore** when done.

4.2. From the terminal on the `workstation` machine, edit the `~/DO316/labs/advanced-review/www2.yaml` file to specify the PVC to use as the root disk:

```
...output omitted...
      volumes:
        - name: rootdisk
          persistentVolumeClaim:
            claimName: root-copy
```

4.3. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/advanced-review/www2.yaml
virtualmachine.kubevirt.io/www2 created
```

If you prefer the OpenShift web console, navigate to **Virtualization** > **VirtualMachines** and then click **Create** > **With YAML**. Copy and then paste the `/home/student/DO316/labs/advanced-review/www2.yaml` file into the editor. Set the `claimName` parameter to `root-copy` and then click **Create**.

4.4. Use the `oc` command to confirm that the `www2` VM is running.

```
[student@workstation ~]$ oc get vmi -n advanced-review
NAME   AGE     PHASE     IP           NODENAME   READY
www1   3m19s   Running   10.9.0.50    worker02   True
www2   10m     Running   10.11.0.45   worker01   True
```

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 only contains temporary data that you can safely discard).

5.1. To test live migration, navigate to **Virtualization** > **VirtualMachines**, select the `www1` VM, and then click **Actions** > **Migrate Node to Node**. Click **Migrate Node to Node** to confirm the operation.

The error message indicates that the `www1-tempdata-`*`xxxxx`* PVC is not using the `ReadWriteMany` access mode. Click **Cancel**

5.2. Navigate to the **Disks** tab. You need to stop the VM before you can detach and then delete the disk that uses the `www1-tempdata-`*`xxxxx`* PVC. Click **Actions** > **Stop** and then click **Stop** to confirm the operation. Wait for the machine to stop.

5.3. Click the vertical ellipsis to the right of the `tempdata` disk and then click **Delete**. Select `Delete www1-tempdata-`*`xxxxx`* `Data Volume and PVC` and then click **Detach** to confirm the operation.

5.4. Click **Actions** > **Start** to restart the VM. Wait for the VM to start.

5.5. Perform a live migration of the VM. Click **Actions** > **Migrate Node to Node** and then click **Migrate Node to Node** to confirm the operation. Wait for the migration to complete.

6. Set the `worker02` cluster node in maintenance mode and move its workload to the remaining nodes.

The `lab` command has prepared the `/home/student/DO316/labs/advanced-review/nm.yaml` resource file that you can use as a model. If you decide to use that file, then you need to adapt it to the requirements.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

6.1. From the terminal on the `workstation` machine, edit the `~/DO316/labs/advanced-review/nm.yaml` file:

```
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
metadata:
  name: node-maintenance
spec:
  nodeName: worker02
  reason: "Node maintenance"
```

6.2. Use the `oc` command to create the resource:

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/advanced-review/nm.yaml
nodemaintenance.nodemaintenance.kubevirt.io/node-maintenance created
```

If you prefer the OpenShift web console, navigate to **Home** > **Search**, click **Resources**, and then select **(NM) NodeMaintenance**. Click **Create Node Maintenance**. Copy and paste the `~/DO316/labs/advanced-review/nm.yaml` resource file into the YAML editor. Under the `spec` section, set the `nodeName` parameter to `worker02`. Click **Create** to create the resource.

6.3. Confirm that the node has the `SchedulingDisabled` status.

```
[student@workstation ~]$ oc get nodes
NAME       STATUS                     ROLES           AGE   VERSION
master01   Ready                      master,worker   12d   v1.23.3+e419edf
master02   Ready                      master,worker   12d   v1.23.3+e419edf
master03   Ready                      master,worker   12d   v1.23.3+e419edf
worker01   Ready                      worker          12d   v1.23.3+e419edf
worker02   Ready,SchedulingDisabled   worker          12d   v1.23.3+e419edf
```

If you prefer the OpenShift web console, navigate to **Compute** > **Nodes** and confirm that the node has the **Scheduling disabled** status.

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

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 advanced-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- A snapshot is a view of a VM at a given time.
- A snapshot includes the VM's configuration and an image of the VM's disks.
- Live snapshots rely on the QEMU guest agent running inside the VM to quiesce the file systems.
- Disk snapshots rely on the back-end storage. If the back-end storage does not support snapshots, then the snapshot only includes the VM's configuration.
- 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.
- `DataVolume` resources orchestrate the cloning process.
- Live migration requires the VM's PVCs to use the `ReadWriteMany` (RWX) access mode.
- Setting a cluster node in maintenance mode requires cordoning off the node and draining the node's workload.
- During the draining process, VMs that support live migration are moved without interruption. Other VMs are stopped and then restarted on a different node.
- You create a `NodeMaintenance` resource to set a node in 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 8

Configuring Kubernetes High Availability for Virtual Machines

Goal

Configure Kubernetes resources to implement high availability for virtual machines.

Objectives

- Demonstrate VM load balancing with Kubernetes networking resources.
- Configure health probes and a Watchdog device to monitor the health and responsiveness of a VM and its services.
- Configure Kubernetes resources to help VMs fail over to another cluster node when node failure is detected.

Sections

- Virtual Machine Load Balancing with Kubernetes Networking Resources (and Guided Exercise)
- Configure Health Probes for Virtual Machines (and Guided Exercise)
- Surviving Node Failure with Virtual Machines (and Guided Exercise)

Lab

- Configuring Kubernetes High Availability for Virtual Machines

Virtual Machine Load Balancing with Kubernetes Networking Resources

Objectives

- Demonstrate VM load balancing with Kubernetes networking resources.

Manage Virtual Machines with a Hypervisor

A Hypervisor helps you to provision *Virtual Machines (VMs)* on a host computer by sharing the compute resources of the host with the virtualized environments. There are two kinds of hypervisors:

Bare metal hypervisor
: The bare metal hypervisor executes on the host's hardware as a layer of a lightweight operating system. You cannot use a host running a bare metal hypervisor for anything except virtualization.

Hosted hypervisor
: The hosted hypervisor executes on the host's operating system, like any other software, allowing you to use the host for purposes besides virtualization.

High Availability with Hypervisors

High Availability (HA) is the ability to maintain continuous operations and to recover from unexpected failures in the shortest time possible.

HA in traditional hypervisors 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.

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.

A standalone hypervisor does not include the software technology to load balance VMs by default. Depending on the hypervisor provider, you might be able to implement a load balancer to work with the hypervisor. By implementing hypervisor agents, you can also add node management actions, such as `fence` and `restart`. You must configure the load balancer, readiness/liveness probes, and watchdog functions for health monitoring in addition to 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Kubernetes Features for Virtual Machines

Kubernetes provides a set of features for VM management that traditional standalone hypervisors do not offer by default. These features include probes to monitor the health of your VMs and applications, load balancing of network traffic for high availability, and sticky sessions for stateful application traffic. Additional features, such as VM run strategies and watchdog devices, are available only when the KubeVirt operator is configured in a Kubernetes cluster.

You can implement Kubevirt in your *Red Hat OpenShift Container Platform (RHOCP)* cluster by installing the Red Hat OpenShift Virtualization operator.

Load balancing

You can configure a load balancer service to allow 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)*, but is still a single machine for initial load balancing.

Readiness and Liveness probes

Developers can configure probes for monitoring their applications. A probe is a periodic check that monitors application health. You can use probes for applications running 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.

Sticky sessions

Sticky sessions enable stateful application traffic by ensuring all traffic hits the same endpoint, but if the endpoint pod terminates, the statefulness can disappear.

Watchdog devices

You can configure a watchdog device inside a VM to check the state of the operating system (OS) and act according to the run strategy. Watchdogs only monitor operating systems and do not detect application failures.

Machine health checks

Machine health checks automatically remediate an unhealthy machine (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 check, the label for the set of hosts to monitor, and the remediation process to use.

Live migration

A live migration ensures that the VM is not interrupted if the node is placed into maintenance or drained.

VM run strategies

A VM's run strategy determines the behavior of a Virtual Machine Instance (VMI) based on a series of conditions:

- `.spec.running`: The `.spec.running` Boolean specifies whether the VM should be running or not. When set to `true`, Red Hat OpenShift Virtualization ensures that the VM is always running. When set to `false`, OpenShift Virtualization ensures that the VM is not running.
- `.spec.runStrategy`: This setting in the virtual machine configuration process provides an alternative to the `spec.running` setting. The `spec.runStrategy` setting allows

greater flexibility for creating and managing VMIs. These two settings are mutually exclusive.

Fencing nodes

Fencing is a remediation method that reboots/deletes `Machine` CRDs to solve problems with automatically provisioned nodes.

When the `MachineHealthCheck` controller detects that a node is `NotReady` it removes the associated `Machine` resource, causing the node to be deleted from the pool host.

References

For more information about route configuration, refer to the *Route configuration _ section in the _Red Hat OpenShift Container Platform* guide at https://docs.openshift.com/container-platform/4.10/networking/routes/route-configuration.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Virtual Machine Load Balancing with Kubernetes Networking Resources

Create a service and route with a custom cookie for two web servers hosting the `helloworld` application to confirm session stickiness and service load balancing.

Outcomes

- Create a `helloworld` service for the web servers and verify 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 resiliency and automatic load balancing provided by Kubernetes.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `ha-loadbalance` namespace and starts the `server1` and `server2` VMs in that namespace.

```
[student@workstation ~]$ lab start ha-loadbalance
```

Instructions

1. As the `admin` user, locate and then navigate to the OpenShift web console. Confirm that the `server1` and `server2` VMs are 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 redhat \
  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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

1.4. Navigate to **Virtualization** > **VirtualMachines**. Select the `ha-loadbalance` namespace from the **Projects** list. Confirm that the `server1` and `server2` VMs are running.

2. Create a `helloworld` service and verify that the service endpoints resolve to the *Virtual Machine Instances (VMI)* of the `server1` and `server2` VMs.

2.1. From the terminal on the `workstation` machine, review the `/home/student/DO316/labs/ha-loadbalance/svc.yml` resource file prepared by the `lab` command.

Notice that the resource file defines a `helloworld` service in the `ha-loadbalance` namespace. The resource file also defines the service selector as the `app: helloworld` label. The `server1` and `server2` VMIs contain the `app: helloworld` label.

```
[student@workstation ~]$ cat ~/DO316/labs/ha-loadbalance/svc.yml
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 create -f ~/DO316/labs/ha-loadbalance/svc.yml
service/helloworld created
```

2.3. Verify that the `helloworld` service endpoints resolve to the IP addresses of the `server1` and `server2` VMIs.

```
[student@workstation ~]$ oc get endpoints -n ha-loadbalance
NAME         ENDPOINTS                   AGE
helloworld   10.11.0.46:80,10.8.2.31:80  15s
[student@workstation ~]$ oc get vmi -n ha-loadbalance
NAME     AGE  PHASE    IP          NODENAME  READY
server1  10m  Running  10.8.2.31   worker01  True
server2  8m   Running  10.11.0.46  worker02  True
```

3. Create a `helloworld` route for the `helloworld` service.

3.1. From the terminal, review the `/home/student/DO316/labs/ha-loadbalance/route.yml` resource file prepared by the `lab` command.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ cat ~/DO316/labs/ha-loadbalance/route.yml
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. Annotating a route with the `router.openshift.io/cookie_name="`*`cookie_name`*`"` annotation creates a custom-named cookie for the route.

(3) The `.spec.host` object defines a custom URL for the route. If the `.spec.host` object is not included, 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 create -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 -n ha-loadbalance
NAME        HOST/PORT                                    PATH  SERVICES    ......
helloworld  hello-ha-loadbalance.apps.ocp4.example.com        helloworld  ......
```

4. Use the `curl` command to confirm that the `helloworld` route is accessible. Confirm that the route provides the custom-named cookie, `hello`, and observe the session stickiness provided by 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. Notice that the connection is load balanced between the two web servers.

```
[student@workstation ~]$ curl hello-ha-loadbalance.apps.ocp4.example.com
<!DOCTYPE html>
<html>
    <head>
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        <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. Afterwards, confirm that the `hello` cookie exists in 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>
[student@workstation ~]$ cat /tmp/cookie_jar
# Netscape HTTP Cookie File
# https://curl.haxx.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
 a67340bf83c97312c11b32f1aa386722
```

4.3. 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. 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>
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        <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>
```

4.4. Use the `curl` command to connect to the `helloworld` route without the `hello` cookie. Observe that session stickiness does not occur without 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 and the IP addresses for the `server1` and `server2` VMIs.

```
[student@workstation ~]$ oc get endpoints -n ha-loadbalance
NAME         ENDPOINTS                   AGE
helloworld   10.11.0.46:80,10.8.2.31:80  15m
[student@workstation ~]$ oc get vmi -n ha-loadbalance
NAME     AGE  PHASE    IP          NODENAME  READY
server1  10m  Running  10.8.2.31   worker01  True
server2  8m   Running  10.11.0.46  worker02  True
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5.2. Use the `virtctl` client to shut down the `server1` VM.

```
[student@workstation ~]$ virtctl stop server1 -n ha-loadbalance
VM server1 was scheduled to stop.
```

5.3. Confirm that the `helloworld` service endpoint is updated to the IP address of the `server2` VMI and does not contain the IP address of the `server1` VMI.

```
[student@workstation ~]$ oc get endpoints -n ha-loadbalance
NAME         ENDPOINTS                   AGE
helloworld   10.11.0.46:80                17m
[student@workstation ~]$ oc get vmi -n ha-loadbalance
NAME     AGE  PHASE    IP          NODENAME  READY
server2  12m  Running  10.11.0.46  worker02  True
```

5.4. 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 -n ha-loadbalance
VM server1 was scheduled to start
```

6.2. Confirm that Kubernetes updated the `helloworld` service endpoint to include the IP address of the `server1` VMI.

```
[student@workstation ~]$ oc get endpoints -n ha-loadbalance
NAME         ENDPOINTS                    AGE
helloworld   10.11.0.48:80,10.8.2.24:80   20m
[student@workstation ~]$ oc get vmi -n ha-loadbalance
NAME     AGE    PHASE     IP          NODENAME   READY
server1  2m33s  Running   10.11.0.48  worker01   True
server2  25m    Running   10.11.0.46  worker02   True
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

6.3. 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. Kubernetes load balances the requests between the two web servers.

```
[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

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-loadbalance
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Configure Health Probes for Virtual Machines

Objectives

- Configure health probes and a Watchdog device to monitor the health and responsiveness of a VM and its services.

Configure the Run Strategy

Virtual machines instances (VMI) have settings that control the running state of the virtual machine (VM).

`.spec.running`
: The `.spec.running` Boolean specifies whether the VM should be running or not. When set to `true`, Red Hat OpenShift Virtualization ensures that the VM is always running. OpenShift Virtualization restarts the VM when it fails. OpenShift Virtualization also restarts the VM when you stop it by gracefully shutting down the operating system or deleting the VMI resource.

  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 the Red Hat OpenShift web console. When you run the `virtctl stop` command or click **Actions** > **Stop** in the OpenShift web console, OpenShift Virtualization switches the `.spec.running` Boolean to `false`. When you run the `virtctl start` command or click **Actions** > **Start** in the OpenShift web console, 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; OpenShift Virtualization detects that the VM is not running and then restarts it.

  The `.spec.runStrategy` parameter accepts one of the following values:

  - `RerunOnFailure`: Restarts the VM when it fails. If you gracefully shut down the operating system from inside the VM, OpenShift Virtualization does not restart the VM.
  - `Always`: Ensures that the VM is always running. Using this value is similar to setting the `.spec.running` Boolean to `true`.
  - `Halted`: Ensures that the VM is not running. Using this value is similar to 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` and `virtctl stop` commands or the OpenShift web console.

The `.spec.running` and `.spec.runStrategy` are mutually exclusive. If you specify both, then OpenShift Virtualization returns an error.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Update the Run Strategy Parameters

To set the `.spec.running` or `.spec.runStrategy` parameter, 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...
```

You can also use the OpenShift web console:

- Navigate to **Virtualization** > **VirtualMachines**.
- Select the VM to edit.
- Navigate to the **YAML** tab.
- Use the YAML editor to the set `.spec.running` or `.spec.runStrategy` parameter.
- Click **Save**.

Configure Health Probes

There are many reasons why applications become unreliable:

- 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 running in pods or VMs.

For monitoring applications running 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 some time to complete. The application might also have to run long batch jobs from time to time, making it temporarily unavailable to clients.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Kubernetes continues running the probe even after the application fails. When the application is available again, Kubernetes adds the VM's IP address back 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 cannot recover without a complete restart of the VM.

Kubernetes provides two 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.

TCP socket

Some applications, such as database servers, file servers, or application servers, do not use HTTP but instead provide their service through TCP ports. 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.

Add Probes to Virtual Machines

You add probes by editing the VM resource. Use the `oc edit` command to add new `readinessProbe` and `livenessProbe` sections to the VM resource. 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)
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        timeoutSeconds: 10   (4)
        failureThreshold: 3   (5)
        successThreshold: 3   (6)
...output omitted...
```

1. The `httpGet` section provides the parameters for the HTTP request. The `path` parameter provides the URL path to the probe endpoint.
2. Specifies how long to wait after the VM starts before probing the application. By default, Kubernetes immediately starts probing the VM.
3. Specifies the frequency of the check. By default, Kubernetes performs the check every 10 seconds.
4. Determines how long to wait for the probe to finish. If the probe exceeds this time, then Kubernetes assumes that it failed. By default, Kubernetes waits one second for the check to complete.
5. Specifies the number of consecutive failures for Kubernetes to consider that the probe failed. By default, Kubernetes tries three times.
6. Specifies the minimum consecutive successes after a probe failed for Kubernetes to consider the probe successful. By default, Kubernetes sets the probe as successful after only 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...
```

1. The `tcpSocket` section provides the parameters of the TCP socket. The `port` parameter provides the TCP socket port to test.

You can also use the OpenShift web console to add readiness and liveness probes:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

- Navigate to **Virtualization** > **VirtualMachines**.
- Select the VM to edit.
- Navigate to the **YAML** tab.
- Use the YAML editor to declare the probes.
- Click **Save**.

Configure a Watchdog Device

The watchdog feature detects and restarts unresponsive operating systems. The feature relies on two components:

- A hardware component that sets a 30-seconds timer (your seconds value might be different). 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.

- Run by the operating system, a software component regularly resets the hardware timer to prevent it from expiring. When the operating system hangs, the software component also hangs and cannot refresh the timer. The timer expires and the system restarts.

Watchdogs only monitor operating systems and do not detect application failures. For example, the watchdog does not trigger when the application hangs, but the operating system is still responding.

Do not use watchdogs for applications that you can monitor by using liveness probes. A liveness probe indirectly detects an unresponsive operating system because all the applications running in an unresponsive operating system also become unresponsive. Use watchdog monitoring only when you have no other way to test your application.

Activate Watchdog Monitoring

Add the emulated watchdog device to your VM resource to activate watchdog monitoring. Use the `oc edit` command to add a new `device` section to the VM resource:

```
[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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
            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 only emulate the Intel 6300ESB chipset.

(2) Action to perform when the watchdog triggers. The `poweroff` action stops the VM; OpenShift Virtualization restarts the VM according to the `.spec.running` or `.spec.runStrategy` parameter. Other options include the `reset` and `shutdown` actions. The references section at the end of this section provides more details.

You can also use the OpenShift web console to activate watchdog monitoring:

- Navigate to **Virtualization** > **VirtualMachines**.
- Select the VM to edit.
- Navigate to the **YAML** tab.
- Use the YAML editor to create the watchdog device.
- Click **Save**.

Watchdog monitoring only works for operating systems that support the Intel 6300ESB chipset. Red Hat Enterprise Linux automatically loads the kernel module that supports the device.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

References

`watchdog`(8) man page

For more information about run strategies, refer to the *About RunStrategies for Virtual Machines* section in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#virt-about-runstrategies-vms_virt-create-vms

For more information about OpenShift Virtualization probes, refer to the *Monitoring Virtual Machine Health* section in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/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 Building Applications* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/building_applications/index#application-health

For more information about configuring a watchdog device, refer to the *Configuring a Watchdog* section in the *Red Hat OpenShift Container Platform Virtualization* guide at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/virtualization/index#virt-configuring-a-watchdog

Guided Exercise

Configure Health Probes for Virtual Machines

Create health probes for VMs behind a service to enable Kubernetes to detect an application's availability and modify service networking accordingly.

Outcomes

- Declare readiness and liveness probes.
- Add watchdog devices to VMs.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. The `lab` command creates the `ha-probe` namespace and starts four virtual machines in that namespace: the `www1` and `www2` VMs, which host a web application; the `watch1` VM, a test VM; and the `mariadb-server` VM, which hosts a MariaDB database.

The `lab` command also creates a service that dispatches the client requests between the two web servers. Finally, the command creates a route resource so that the clients can reach the web application by using the http://www-ha-probe.apps.ocp4.example.com URL.

```
[student@workstation ~]$ lab start ha-probe
```

Instructions

1. As the `admin` user, locate and then navigate to the Red Hat OpenShift web console. Confirm that the four VMs are 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 redhat \
  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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

1.3. 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 the `redhat` password when prompted.

1.4. Navigate to **Virtualization** > **VirtualMachines** and then select the `ha-probe` project. Confirm that the `mariadb-server`, `watch1`, `www1`, and `www2` VMs are running.

2. Review the service and the route that the `lab` command deployed. Confirm that Kubernetes dispatches the requests to the http://www-ha-probe.apps.ocp4.example.com URL between the `www1` and `www2` VMs.

2.1. From the terminal on the `workstation` machine, use the `oc` command to list the VMI resources in the `ha-probe` namespace. Notice the IP addresses of the `www1` and `www2` VM instances.

```
[student@workstation ~]$ oc get vmi -n ha-probe
NAME             AGE   PHASE     IP           NODENAME   READY
mariadb-server   75m   Running   10.9.0.49    master03   True
watch1           76m   Running   10.11.0.47   worker01   True
www1             79m   Running   10.11.0.45   worker01   True
www2             78m   Running   10.8.2.36    worker02   True
```

The IP addresses are probably different in your environment.

2.2. List the service and its endpoints. Notice that the service uses the IP addresses of the two VMs for its endpoints.

```
[student@workstation ~]$ oc get service -n ha-probe
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
front   ClusterIP   172.30.150.44   <none>        80/TCP    76m
[student@workstation ~]$ oc get endpoints -n ha-probe
NAME    ENDPOINTS                     AGE
front   10.11.0.45:80,10.8.2.36:80   76m
```

2.3. List the route resource. Notice that the DNS name for the route is `www-ha-probe.apps.ocp4.example.com`. This name points to the `front` service.

```
[student@workstation ~]$ oc get route -n ha-probe
NAME        HOST/PORT                          PATH   SERVICES   PORT ...
www-front   www-ha-probe.apps.ocp4.example.com         front      80   ...
```

2.4. 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-ha-probe.apps.ocp4.example.com
Welcome to www2
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www2
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www1
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
Welcome to www1
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www2
```

3. The web servers running inside the `www1` and `www2` VMs expose the `/health` endpoint that you can use 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. From the terminal on the `workstation` machine, use the `oc edit vm` command to add the readiness probe to the `www1` VM.

   You can copy and paste the `readinessProbe` section from the `/home/student/DO316/labs/ha-probe/readiness.yaml` file that the `lab` command prepared.

```
[student@workstation ~]$ oc edit vm www1 -n ha-probe
# 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:
          path: /health
          port: 80
        initialDelaySeconds: 10
        periodSeconds: 5
        timeoutSeconds: 2
        failureThreshold: 2
        successThreshold: 1
      evictionStrategy: LiveMigrate
      hostname: www1
...output omitted...
```

   If you prefer the OpenShift web console, then navigate to **Virtualization** > **VirtualMachines**, select the `www1` VM, and then navigate to the **YAML** tab. Use the YAML editor to declare the probe and then click **Save**.

   3.2. You must delete the `www1` VMI resource before Red Hat OpenShift Virtualization recognizes the new probe. OpenShift Virtualization restarts a new instance from the VM resource that you modified.

   The following `oc delete` command might take a few minutes to complete.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc delete vmi www1 -n ha-probe
virtualmachineinstance.kubevirt.io "www1" deleted
```

3.3. Confirm that a new VM instance is running. You might have to wait a few minutes and run the command several times.

```
[student@workstation ~]$ oc get vmi www1 -n ha-probe
NAME   AGE   PHASE     IP           NODENAME   READY
www1   95s   Running   10.11.0.48   worker01   True
```

3.4. Repeat this step to add the readiness probe to the `www2` VM.

4. Test the probe by stopping the `httpd` service running inside the `www1` VM.

4.1. From the terminal on the `workstation` machine, run the `/home/student/DO316/labs/ha-probe/loop.sh` script that the `lab` command prepared. This script runs the `curl` command in an infinite loop.

Leave the command running.

```
[student@workstation ~]$ ~/DO316/labs/ha-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. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `www1` VM and then navigate to the **Console** tab.

4.3. Log in as the `root` user with the `redhat` password.

4.4. Stop the `httpd` service.

```
[root@www1 ~]# systemctl stop httpd
[root@www1 ~]#
```

Do not close the console.

4.5. Switch back to the terminal on the `workstation` machine. After a few seconds, the output of the `loop.sh` script shows that Kubernetes sends the requests only to the `www2` VM.

Press `Ctrl+C` to stop the script.

```
...output omitted...
Welcome to www2
Welcome to www2
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
Welcome to www1
Welcome to www1
Welcome to www2
Welcome to www2
Welcome to www2
Welcome to www2
Welcome to www2
Welcome to www2
^C
```

4.6. Use the `oc` command to confirm that the `front` service has only one endpoint remaining.

```
[student@workstation ~]$ oc get endpoints -n ha-probe
NAME    ENDPOINTS      AGE
front   10.8.2.46:80   122m
```

4.7. Switch back to the `www1` VM console running inside the web browser and restart the `httpd` service. Log out of the console when done.

```
[root@www1 ~]# systemctl start httpd
[root@www1 ~]# logout
```

4.8. Return to the terminal on the `workstation` machine. Run the `curl` command several times to confirm that Kubernetes dispatches the requests between the two VMs again.

```
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www2
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www2
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www1
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www1
[student@workstation ~]$ curl http://www-ha-probe.apps.ocp4.example.com
Welcome to www2
```

4.9. Use the `oc` command to confirm that the `front` service has two endpoints.

```
[student@workstation ~]$ oc get endpoints -n ha-probe
NAME    ENDPOINTS                     AGE
front   10.11.0.48:80,10.8.2.46:80    126m
```

5. The MariaDB database 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. From the terminal on the `workstation` machine, use the `oc edit vm` command to add the liveness probe to the `mariadb-server` VM.

You can copy and paste the `livenessProbe` section from the `/home/student/DO316/labs/ha-probe/liveness.yaml` file that the `lab` command prepared.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc edit vm mariadb-server -n ha-probe
# 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:
          port: 3306
        initialDelaySeconds: 10
        periodSeconds: 5
      evictionStrategy: LiveMigrate
      hostname: mariadb-server
...output omitted...
```

If you prefer the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `mariadb-server` VM, and then navigate to the **YAML** tab. Use the YAML editor to declare the probe and then click **Save**.

5.2. You must delete the `mariadb-server` VMI resource before Red Hat OpenShift Virtualization recognizes the new probe. OpenShift Virtualization restarts a new instance from the VM resource that you modified.

The following `oc delete` command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi mariadb-server -n ha-probe
virtualmachineinstance.kubevirt.io "mariadb-server" deleted
```

5.3. Confirm that a new VM instance is running. You might have to wait a few minutes and run the command several times.

```
[student@workstation ~]$ oc get vmi mariadb-server -n ha-probe
NAME             AGE   PHASE     IP          NODENAME   READY
mariadb-server   14s   Running   10.9.0.50   master03   True
```

6. Test the probe by stopping the `mysql` service running inside the `mariadb-server` VM.

6.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `mariadb-server` VM and then navigate to the **Console** tab.

6.2. Log in as the `root` user with the `redhat` password.

6.3. Stop the `mysql` service.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[root@mariadb-server ~]# systemctl stop mysql
[root@mariadb-server ~]#
```

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. From the terminal on the `workstation` machine, use the `oc edit vm` command to add a watchdog device to the `watch1` VM.

You can copy and paste the `watchdog` section from the `/home/student/DO316/labs/ha-probe/watchdog.yaml` file that the `lab` command prepared.

```
[student@workstation ~]$ oc edit vm watch1 -n ha-probe
# 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: 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-rhel8.4.0
...output omitted...
```

If you prefer the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `watch1` VM, and then navigate to the **YAML** tab. Use the YAML editor to declare the watchdog device and then click **Save**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

7.2. You must delete the `watch1` VMI resource before OpenShift Virtualization recognizes the new device. OpenShift Virtualization restarts a new instance from the VM resource that you modified.

The following `oc delete` command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi watch1 -n ha-probe
virtualmachineinstance.kubevirt.io "watch1" deleted
```

7.3. Confirm that a new VM instance is running. You might have to wait a few minutes and run the command several times.

```
[student@workstation ~]$ oc get vmi watch1 -n ha-probe
NAME     AGE   PHASE     IP           NODENAME   READY
watch1   28s   Running   10.11.0.50   worker01   True
```

8. Test the watchdog device.

8.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `watch1` VM and then navigate to the **Console** tab.

8.2. Log in as the `root` user with the `redhat` 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 `cat` command to simulate a process opening the `/dev/watchdog` device file in write mode. The kernel stops refreshing the timer. After 30 seconds, the Intel 6300ESB emulated chipset restarts the VM.

```
[root@watch1 ~]# cat >> /dev/watchdog
```

The preceding `cat` command does not return output and does not exit.

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 ha-probe
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Surviving Node Failure with Virtual Machines

Objectives

- Configure Kubernetes resources to help VMs fail over to another cluster node when node failure is detected.

Eviction Strategies to Survive Cluster Node Loss

Nodes fail for many reasons, from insufficient disk space to package issues. When a node enters a failing state, whether the node is a control plane or worker node, 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 a `NotReady` state after failing health checks, the remaining pods on the node are rescheduled to healthy nodes with a `Ready` status.

Eviction strategies determine if VMs on the failed node are moved to another node or terminated:

- Live migration: Perform a live migration to ensure that the VM is not interrupted if the node is placed into maintenance or drained.
- Not defined: VMs are terminated if the node is placed into maintenance or drained.

VMs with a live migration strategy must meet the following requirements:

- Have a *persistent volume claim (PVC)* with a shared *ReadWriteMany (RWX)* access mode.
- The pod network binding must not use the `bridge` interface type.
- If the VM uses a host model CPU, live migration is only supported between nodes that also support the VM's host model CPU.

> **Note**
> Versions 4.8 and newer of *Red Hat OpenShift Virtualization* support live migration for virtual machines that are attached to an SR-IOV network interface.

VM Scheduling

VMs are scheduled to healthy nodes based on the eviction strategy and the node placement rules:

Node selector
: VMs are scheduled on nodes that are labeled with the key-value pairs specified in the `.spec.nodeSelector` field of the VM's manifest.

Affinity
: Pod affinity, pod anti-affinity, and node affinity rules are supported for VM scheduling. Affinity rules use more expressive syntax to schedule VMs on nodes, such as specifying if a rule is a hard requirement or a preference. For example, if you specify that a rule is a hard requirement, a VM is not scheduled if the rule is not satisfied. If a rule is a preference, a VM is still scheduled on node even if the constraints of the rule are not met.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

VMs with node affinity rules are scheduled on nodes that contain the specified key-value labels in the `.spec.affinity.nodeAffinity` field of the VM's manifest. If the specified key-value labels are not present on a node, the VM is not scheduled to that node. With node affinity, the key-value labels of other pods does not affect the scheduling of VMs on a node.

In contrast to node affinity, pod affinity and anti-affinity rules rely on the key-value labels of other pods to determine VM scheduling. For example, you can create a pod anti-affinity rule to prevent a VM from scheduling on a node if any pods on that node contain a specified key-value label. Pod affinity works for VMs because the `VirtualMachine` workload type is based on the `Pod` object.

> **Note**
> Affinity rules only apply during scheduling. If a constraint is no longer met, pod and VM workloads are not rescheduled to other nodes.

Tolerations and taints

Pods and VMs are scheduled on nodes that have matching taints. VMs must have a toleration that matches the taint of a node, otherwise, the node will refuse to schedule the workload.

You can also specify a scheduler profile to control how pods are scheduled onto nodes. There are three scheduler profiles available:

- `LowNodeUtilization`: Attempts to spread pods evenly across nodes to get low resource usage per node.
- `HighNodeUtilization`: Attempts to place as many pods on to as few nodes as possible. This approach minimizes node count and creates high resource usage per node.
- `NoScoring`: A low latency profile that strives for the quickest scheduling cycle by disabling all score plug-ins.

A scheduler profile is not a node placement rule.

Node Failures

The `virt-handler` is a daemonset that communicates with `libvirtd` instance to define VMs.

VMs are not scheduled on nodes that have not had a heartbeat from the `virt-handler` daemonset. It might take up to five minutes for `virt-handler` and Kubernetes to detect the failure. After that time, the control plane nodes (master nodes) mark the node as unschedulable and then migrate the node's workloads according to the resource's node placement and scheduling rules.

If the `virt-handler` daemonset loses the connection to the cluster's API server, the node cannot communicate its status. The node enters a failed state, and the remaining VMs cannot migrate to the healthy nodes.

Initiate VM Rescheduling with Monitoring Tools

By default, *Red Hat OpenShift Container Platform* (RHOCP) includes a set of monitoring tools that periodically check the status of cluster services and created resources, thus providing a way to monitor 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Watchdog Monitoring

You can configure a watchdog device to check the state of the VM's guest operating system (OS) and act according to the run strategy. The VM's watchdog service monitors only for guest OS failures and does not detect application failures.

A watchdog has three ways to proceed when the guest OS is unresponsive:

power off
: The VM powers down immediately. If `spec.running` is set to `true` or if `spec.runStrategy` 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 down by stopping all services.

Using the reset option is discouraged because the reboot can cause liveness probes to time out and mark the VM's pod as failed. Depending on the run strategy, a VM can be deleted or restarted on the same node or on another node.

> **Note**
> The run strategies and the installation of the watchdog device is explained in greater detail elsewhere in this course.

Health Checks for VM Applications and Nodes

You can monitor the status of a VM's application and perform necessary actions with health checks. A health check periodically checks the availability of a VM's application by using readiness and liveness probes. You can use health checks individually or combined.

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.

Liveness health check
: 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 then redeploys a new instance.

Watchdog device
: A watchdog device detects and restarts unresponsive operating systems and does not directly detect application failures. For example, a watchdog does not trigger when a VM's application is unresponsive but the operating system is still responding.

Do not use watchdogs for applications that you can monitor by using liveness probes. A liveness probe indirectly detects an unresponsive operating system because all the applications running in an unresponsive operating system also become unresponsive. Use watchdog monitoring only when you have no other way to test your application.

> **Note**
> The liveness, readiness, and watchdog device health checks are explained in greater detail elsewhere in this course.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Machine Health Checks

Machine health checks automatically remediate an unhealthy machine (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 check, the label for the set of hosts to monitor, and the remediation process to use. The bare metal machine controller that observes a `MachineHealthCheck` resource checks for the defined condition; if the condition is detected, the controller initiates the remediation process.

> **Note**
> Machine health checks are only available for clusters that are installed as bare-metal *installer-provisioned infrastructure (IPI)*.

Depending on the condition, the host might be removed or only rebooted.

Standard Remediation for Failed Nodes

If a host fails the health check, the bare metal machine controller drains the remaining VMs on the node and reschedules the workloads to healthy nodes owned by the machine set, following the scheduled rules and eviction strategies configured for the VMs. After all the VMs are drained and the node reregisters itself with the cluster, the bare metal machine controller restores the annotations and labels that existed on the unhealthy node to the new node.

To limit the disruptive impact of the host deletion, the controller drains and deletes nodes one node at a time. If multiple unhealthy hosts exceed the `maxUnhealthy` threshold specified for the targeted pool of hosts, then remediation stops and requires manual intervention.

There are some limitations to consider when using machine health checks:

- Control plane hosts are not currently supported and are not remediated if they are unhealthy.
- Only hosts owned by a machine set are remediated by a machine health check.
- If the node for a host is removed from the cluster, the machine health check identifies that the host is unhealthy and remediates it immediately.
- A host with the master role cannot be remediated.
- A host is remediated immediately if the `Machine` resource enters a `Failed` status.
- 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 Container Platform 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:

- Allows the recovery of control plane nodes.
- Reduces the risk of data loss in hyperconverged environments.
- Reduces the downtime associated with recovering physical hosts.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

If the power operations do not complete, the bare metal machine controller triggers the reprovisioning of the unhealthy node, unless this is a control plane node or an externally-provisioned node.

> **Note**
> If the node has a *Baseboard Management Controller (BMC)*, BMC credentials and network access to the BMC interface on the node are required.

Poison Pill Operator

Available in non-IPI bare metal clusters, the Poison Pill Operator runs on the cluster nodes and reboots nodes that are identified as unhealthy. This remediation strategy minimizes downtime for stateful applications and ReadWriteOnce (RWO) volumes, and restores compute capacity if transient failures occur.

The operator uses the `MachineHealthCheck` controller to detect the health of a node in the cluster. If a node is unhealthy, the `MachineHealthCheck` resource creates the `PoisonPillRemediation` *custom resource (CR)*, to trigger the Poison Pill Operator, but this CR is not editable. Instead, the Poison Pill Operator creates a `PoisonPillConfig` CR in the operator's namespace that is editable. If the `PoisionPillConfig` CR detects any changes, it recreates the Poison Pill daemonset.

When an unhealthy node is detected according to the `PoisonPillConfig` CR, the `MachineHealthCheck` controller creates the `PoisonPillRemediation` CR to trigger the Poison Pill Operator. The operator uses the `PoisionPillRemediationTemplate` to define the remediation strategy for the unhealthy nodes.

You can install the Poison Pill Operator from the command line by creating a subscription in the `openshift-operators` namespace, or via the web console through the `OperatorHub`.

> **Note**
> For more information about the Poison Pill Operator, refer to the *Remediating nodes with the Poison Pill Operator* section in the *Red Hat OpenShift Container Platform* guide at https://docs.openshift.com/container-platform/4.10/nodes/nodes/eco-poison-pill-operator.html

Manually Recovering VMs from a Failed Node

If a node fails and machine health checks and the Poison Pill Operator are not deployed on the cluster, VMs configured with `RunStrategy: Always` are not automatically relocated to healthy nodes. Instead, you must manually remove the node by deleting the node object. However, if the cluster was installed with IPI and the machine health checks are correctly configured, then failed nodes are automatically recycled and the VMs with `RunStrategy: Always` or `RerunOnFailure` are automatically scheduled on healthy nodes.

Deleting a Node

Deleting a node deletes the node object in Kubernetes, but the pods that reside on the node are not deleted. If the pods on the node are not backed by a replication controller, the pods become inaccessible. Only pods backed by replication controllers are rescheduled to other available nodes.

You can delete a node from your cluster by completing the following steps:

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Mark the node as not schedulable with the `oc adm cordon` *`node-name`* command.

```
[user@host ~]$ oc adm cordon worker01
node/worker01 cordoned
```

Drain all pods and VMs on the node with the `oc adm drain` *`node-name`* command.

```
[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 down the physical hardware before you proceed.

Delete the node from the cluster with the `oc delete node` *`node-name`* command.

```
[user@host ~]$ oc delete node worker01
node "worker01" deleted
```

Note

The node can still rejoin the cluster after 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 rescheduled on healthy nodes with the `oc get vmis` command.

```
[user@host ~]$ oc get vmis --all-namespaces
NAMESPACE          NAME               AGE    PHASE     IP          NODENAME    READY
virtual-machinesQA   testing-piranha   4h6m   Running   10.9.0.47   worker01
 True
```

References

For more information, refer to the *Working with nodes* chapter in the *Red Hat OpenShift Container Platform Guide* at https://docs.openshift.com/container-platform/4.10/nodes/nodes/nodes-nodes-working.html

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Guided Exercise

Surviving Node Failure with Virtual Machines

Configure a virtual machine to automatically fail over to another cluster node if the node it runs on becomes unresponsive.

Outcomes

- Monitor the connectivity to a web application provided by two VMs that reside on a failed node.
- Identify and drain the failed node hosting 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 system for this exercise.

This command ensures that the cluster API is reachable. The `lab` command creates the `ha-node` namespace and starts two virtual machines, `web1` and `web2`, which host the web application.

It also creates a service that load balances client requests between the two VMs and a route resource for clients to access the web application.

```
[student@workstation ~]$ lab start ha-node
```

Instructions

1. As the `admin` user, locate and then navigate to Red Hat OpenShift web console. Confirm that the two VMs are running.

   1.1. From a terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 navigate to https://console-openshift-console.apps.ocp4.example.com. Select **htpasswd_provider** and log in as the `admin` user with the `redhat` password when prompted.

1.4. Navigate to **Virtualization** > **VirtualMachines**. Select the `ha-node` project from the **Projects** list to confirm that the `web1` and `web2` VMs are running.

2. Check the eviction and run strategies of the `web1` and `web2` VMs.

2.1. Select the `web1` VM, and then navigate 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. Then, navigate to the `spec.template.spec.evictionStrategy` object and confirm that the `LiveMigrate` value is set.

```
...output omitted...
spec:
...output omitted...
  runStrategy: RerunOnFailure
  template:
    spec:
...output omitted...
      evictionStrategy: LiveMigrate
...output omitted...
```

2.2. Navigate to **Virtualization** > **VirtualMachines** and select the `web2` VM.

Click the **Details** tab and navigate to the **Scheduling and resource requirements** section. Locate the `Eviction Strategy` subsection and confirm that an eviction strategy is not configured for the VM.

Then, click the **YAML** tab to open the VM's manifest in the YAML editor. Use the YAML editor to locate the `spec.runStrategy` object and confirm that the `RerunOnFailure` value is set.

```
...output omitted...
spec:
...output omitted...
  runStrategy: RerunOnFailure
...output omitted...
```

3. Confirm that the `www-web` service endpoints resolve to the IP addresses of the `web1` and `web2` VMIs. Identify and monitor which nodes are running the `web1` and `web2` VMIs.

3.1. On the `workstation` machine, use the terminal to confirm that the `www-web` service has active endpoints that resolve to the IP addresses of the `web1` and `web2` VMIs.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get endpoints -n ha-node
NAME      ENDPOINTS                    AGE
www-web   10.11.0.9:80,10.11.0.11:80  8m

[student@workstation ~]$ oc get vmi -n ha-node
NAME   AGE   PHASE     IP           NODENAME   READY
web1   11m   Running   10.11.0.9  worker01   True
web2   10m   Running   10.11.0.11 worker01   True
```

Notice the node hosting the VMIs.

3.2. Open a new terminal window and execute the `loop.sh` file located in the `/home/student/DO316/labs/ha-node/` directory. The `loop.sh` file executes the `curl` command against the \web-ha-node.apps.ocp4.example.com route.

```
[student@workstation ~]$ sh /home/student/DO316/labs/ha-node/loop.sh
Welcome to web1
Welcome to web2
Welcome to web1
...output omitted...
```

3.3. Open a new terminal window and execute the `watch oc get vmi -n ha-node` command to monitor the VMI's availability during this exercise.

```
[student@workstation ~]$ watch oc get vmi -n ha-node
Every 2.0s: oc get vmi -n ha-node                workstation.lab.example.com:....
NAME   AGE   PHASE     IP           NODENAME   READY
web1   15m   Running   10.11.0.9    worker01   True
web2   14m   Running   10.11.0.11   worker01   True
```

4. Developers are reporting that resources residing on the `worker01` node are experiencing performance and connectivity issues. These issues do not affect resources residing on other cluster nodes.

As the cluster administrator, you suspect that the node is failing due to an incorrect configuration on the `worker01` node.

Prevent new workloads from running on the `worker01` node 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 `worker01` node and delete it from the cluster.

4.1. On the `workstation` machine, open a new terminal 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 a `Ready,SchedulingDisabled` status.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get node worker01
NAME       STATUS                     ROLES    AGE   VERSION
worker01   Ready,SchedulingDisabled   worker   3d17h   v1.23.3+e419edf
```

4.3. Drain the node of its workloads. During the node drain, monitor the terminal windows executing the `loop.sh` and the `watch oc get vmi -n ha-node` commands. Observe the high availability of the web application and the status of the `web1` and `web2` VMIs. Wait for the node drain to complete before you proceed to the next step.

```
[student@workstation ~]$ oc adm drain worker01 --ignore-daemonsets=true \
  --delete-emptydir-data --force
node/worker01 already cordoned
...output omitted...
node/worker01 drained
[student@workstation ~]$ sh /home/student/DO316/labs/ha-node/loop.sh
Welcome to web1
Welcome to web1
Welcome to web1
Welcome to web1
Welcome to web1
Welcome to web1
...output omitted...
[student@workstation ~]$ watch oc get vmi -n ha-node
Every 2.0s: oc get vmi -n ha-node                workstation.lab.example.com:....
NAME   AGE   PHASE      IP           NODENAME   READY
web1   21m   Running    10.8.2.155   worker02   True
web2   19m   Succeeded  10.11.0.11   worker01   False
```

Notice that the phase of the `web2` VMI is set to `Succeeded` and that the VMI has a `False` ready status. This is because Kubernetes shut down the `web2` VMI. Kubernetes cannot automatically relocate the `web2` VM to a healthy node in the cluster because the `web2` VMI is not configured with an eviction strategy and machine health checks are not configured in the cluster.

4.4. Configure the `web2` VM with the `LiveMigrate` eviction strategy and then manually recover the VM from the failed node.

From the web console, navigate to **Virtualization** > **VirtualMachines**. Select the `web2` VMI and click the **Details** tab. Navigate to the **Scheduling and resource requirements** section and click **No Eviction Strategy** in the `Eviction Strategy` subsection. Click the **LiveMigrate** flag and click **Save**. Click **Actions** > **Start** to power on and reschedule the VM on another node in the cluster.

4.5. Monitor the terminal window executing the `loop.sh` and observe the `web1` VMI is serving the requests until `web2` enters a `Running` status.

Afterwards, move to the terminal where you have the `watch oc get vmi -n ha-node` command running and notice that the `web1` VMI node host is now `worker02` and the `master02` node host for the `web2` VMI.

```
[student@workstation ~]$ sh /home/student/DO316/labs/ha-node/loop.sh
Welcome to web1
Welcome to web1
Welcome to web1
Welcome to web1
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
Welcome to web2
Welcome to web2
Welcome to web1
...output omitted...

[student@workstation ~]$ watch oc get vmi -n ha-node
Every 2.0s: oc get vmi -n ha-node                workstation.lab.example.com:....
NAME   AGE   PHASE     IP           NODENAME   READY
web1   28m   Running   10.8.2.155   worker02   True
web2   5m    Running   10.8.0.56    master02   True
```

4.6. 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 a `stopped` status before proceeding.

4.7. Return to the terminal 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
```

4.8. 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    master,worker   19d   v1.23.3+e419edf
master02   Ready    master,worker   19d   v1.23.3+e419edf
master03   Ready    master,worker   19d   v1.23.3+e419edf
worker02   Ready    worker          19d   v1.23.3+e419edf
```

5. Instruct the deleted node to rejoin the cluster.

5.1. 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 an `active` status before proceeding.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

5.2. From a terminal on the `workstation` machine, list the nodes to confirm that the deleted node has rejoined the cluster. It might take a few minutes for the node to display a `Ready` status.

```
[student@workstation ~]$ oc get nodes
NAME       STATUS   ROLES           AGE     VERSION
master01   Ready    master,worker   19d     v1.23.3+e419edf
master02   Ready    master,worker   19d     v1.23.3+e419edf
master03   Ready    master,worker   19d     v1.23.3+e419edf
worker01   Ready    worker          2m25s   v1.23.3+e419edf
worker02   Ready    worker          19d     v1.23.3+e419edf
```

5.3. In the terminal that is executing the `loop.sh` file, press `Ctrl+C` to stop the command. Close the terminal window.

5.4. In the terminal that is executing the `watch oc get vmi -n ha-node` command, press `Ctrl+C` to stop the command. Close the terminal window.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-node
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Configuring Kubernetes High Availability for Virtual Machines

Configure Kubernetes resources to provide high availability to virtual machines and their applications.

Outcomes

- Configure load balancing for a web application residing on two virtual machines.
- Configure health probes and watchdog devices on virtual machines.
- Configure VMs to automatically recover from a failed node.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `ha-review` namespace and starts the three virtual machines in that namespace.

```
[student@workstation ~]$ lab start ha-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 the `redhat` 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`, `www2`, and `mariadb-server` VMs are running under the `ha-review` project.

2. Configure Kubernetes networking resources to provide high availability to the web application hosted on the `www1` and `www2` VMs.

   Create a `web` service in the `ha-review` namespace that uses the `app: web` selector. 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 prepared the `service.yaml` and `route.yaml` files in the `/home/student/DO316/labs/ha-review/` directory for you. Edit and then apply the files to create the `web` service and the `www` route.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3. Confirm that the `www` route is accessible to clients at the `web-ha-review.apps.ocp4.example.com` host by using the `curl` command. Confirm that the web application is load balanced 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.

4. The web servers running inside the `www1` and `www2` VMs expose the `/health` endpoint that you can use 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 prepared the `/home/student/DO316/labs/ha-review/readiness.yaml` file for you. Reference this file to create the readiness probe for the `www1` VM.

   After configuring the readiness probe, restart the VM to regenerate the `www1` VMI.

   Confirm that Kubernetes removes the `www1` VMI as a `web` service endpoint if the readiness probe fails. Log in to the VM console as the `root` user with the `redhat` password. Stop the `httpd` service and confirm that the `www1` VMI is removed from the `web` service endpoints.

   Then, start the `httpd` service and confirm that the `web` service endpoints include the `www1` VMI.

5. Add an `i6300esb` watchdog device named `testwatchdog` on the `www2` 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 prepared the `/home/student/DO316/labs/ha-review/watchdog.yaml` file for your reference. Restart the VM to apply the changes.

   Then, log in to the `www2` VM as the `root` user with the `redhat` password. Confirm that the watchdog device works by using the `cat >> /dev/watchdog` command.

6. The MariaDB database 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.

   The `lab` command prepared the `/home/student/DO316/labs/ha-review/liveness.yaml` file for you. Reference this file to create the liveness probe on the `mariadb-server` VM.

   After configuring the liveness probe, restart the VM to regenerate the `mariadb-server` VMI.

   Confirm that Kubernetes automatically restarts the `mariadb-server` VM if the liveness probe fails. Log in to the VM console as the `root` user with the `redhat` password. Stop the `mysql` service and confirm that the VM is restarted by Kubernetes.

7. Monitor the availability of the web application during a node failure. Open a new terminal window and then execute the `loop.sh` file in the `/home/student/DO316/labs/ha-review/` directory that the `lab` command prepared for you.

   Prevent new workloads from scheduling on the node that is hosting the `www2` VMI and then drain the node of its workloads.

8. 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 is responding 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, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Configuring Kubernetes High Availability for Virtual Machines

Configure Kubernetes resources to provide high availability to virtual machines and their applications.

Outcomes

- Configure load balancing for a web application residing on two virtual machines.
- Configure health probes and watchdog devices on virtual machines.
- Configure VMs to automatically recover from a failed node.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `ha-review` namespace and starts the three virtual machines in that namespace.

```
[student@workstation ~]$ lab start ha-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 the `redhat` 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`, `www2`, and `mariadb-server` VMs are running under the `ha-review` project.

   1.1. From a terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 the `redhat` password when prompted.

   1.3. Navigate to **Virtualization** > **VirtualMachines** and then select the `ha-review` project. Confirm that the `www1`, `www2`, and `mariadb-server` VMs are running.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2. Configure Kubernetes networking resources to provide high availability to the web application hosted on the `www1` and `www2` VMs.

Create a `web` service in the `ha-review` namespace that uses the `app: web` selector. 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 prepared the `service.yaml` and `route.yaml` files in the `/home/student/DO316/labs/ha-review/` directory for you. Edit and then apply the files to create the `web` service and the `www` route.

2.1. From the terminal on the `workstation` machine, edit the `~/DO316/labs/ha-review/service.yaml` file to configure the `web` service in the `ha-review` namespace 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. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/ha-review/service.yaml
service/web created
```

If you prefer the OpenShift web console, navigate to **Networking** > **Services** and then click **Create Service**. Drag and drop the `/home/student/DO316/labs/ha-review/service.yaml` file into the editor. Edit the service by using the information given in the instructions. Click **Create** when done.

2.3. Use the `oc` command to confirm that the `web` service endpoints resolve to the IP addresses of the `www1` and `www2` VMIs.

```
[student@workstation ~]$ oc get endpoints -n ha-review
NAME   ENDPOINTS                    AGE
web    10.8.2.48:80,10.8.2.50:80   1m39s
[student@workstation ~]$ oc get vmi -n ha-review
NAME             AGE   PHASE     IP          NODENAME   READY
mariadb-server   14m   Running   10.8.2.52   worker01   True
www1             17m   Running   10.8.2.48   worker01   True
www2             16m   Running   10.8.2.50   worker01   True
```

2.4. Expose the `web` service for external access by creating a route. Edit the `~/DO316/labs/ha-review/route.yaml` file to configure the `www` route in the `ha-review` namespace. Specify the host as `web-ha-review.apps.ocp4.example.com` 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
```

2.5. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/ha-review/route.yaml
route.route.openshift.io/wwww created
```

If you prefer the OpenShift web console, navigate to **Networking** > **Routes** and then click **Create Route**. Click **Edit YAML** and then drag and drop the `/home/student/DO316/labs/ha-review/route.yaml` file into the editor. Edit the route by using the information given in the instructions. Click **Create** when done.

2.6. Use the `oc` command to verify the `www` route settings.

```
[student@workstation ~]$ oc get route -n ha-review
NAME    HOST/PORT                              PATH    SERVICES    PORT    TERMINATION
  WILDCARD
www     web-ha-review.apps.ocp4.example.com            web         80
  None
```

3. Confirm that the `www` route is accessible to clients at the `web-ha-review.apps.ocp4.example.com` host by using the `curl` command. Confirm that the web application is load balanced 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. From the terminal on the `workstation` machine, 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
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

3.2. Use the `curl` command to send five subsequent requests to the `www` route. Confirm that the web application is load balanced between the `www1` and `www2` VMs.

```
[student@workstation ~]$ curl web-ha-review.apps.ocp4.example.com/?[1-5]
[1/5]: web-ha-review.apps.ocp4.example.com/?1 --> <stdout>
_-_curl_--web-ha-review.apps.ocp4.example.com/?1
Welcome to www2

[2/5]: web-ha-review.apps.ocp4.example.com/?2 --> <stdout>
--_curl_--web-ha-review.apps.ocp4.example.com/?2
Welcome to www1

[3/5]: web-ha-review.apps.ocp4.example.com/?3 --> <stdout>
--_curl_--web-ha-review.apps.ocp4.example.com/?3
Welcome to www2

[4/5]: web-ha-review.apps.ocp4.example.com/?4 --> <stdout>
--_curl_--web-ha-review.apps.ocp4.example.com/?4
Welcome to www1

[5/5]: web-ha-review.apps.ocp4.example.com/?5 --> <stdout>
--_curl_--web-ha-review.apps.ocp4.example.com/?5
Welcome to www2
```

3.3. Use the `curl` command to save the `web` cookie to the `workstation` machine.

```
[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` with the `web` cookie to access the `www` route.

```
[student@workstation ~]$ curl web-ha-review.apps.ocp4.example.com
 -b /tmp/cookie_jar
Welcome to www1
```

4. The web servers running inside the `www1` and `www2` VMs expose the `/health` endpoint that you can use 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 prepared the `/home/student/DO316/labs/ha-review/readiness.yaml` file for you. Reference this file to create the readiness probe for the `www1` VM.

After configuring the readiness probe, restart the VM to regenerate the `www1` VMI.

Confirm that Kubernetes removes the `www1` VMI as a `web` service endpoint if the readiness probe fails. Log in to the VM console as the `root` user with the `redhat` password. Stop the `httpd` service and confirm that the `www1` VMI is removed from the `web` service endpoints.

Then, start the `httpd` service and confirm that the `web` service endpoints include the `www1` VMI.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

4.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `www1` VM and then click the **YAML** tab. Use the YAML editor to create a readiness probe for the VM.

You can reference the `readinessProbe` section in the `/home/student/DO316/labs/ha-review/readiness.yaml` file that the `lab` command prepared.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
...output omitted...
spec:
  ...output omitted...
  template:
    metadata:
    ...output omitted...
    spec:
      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...
```

Click **Save** to save the changes.

4.2. Restart the VM to regenerate the `www1` VMI. Click **Actions** > **Restart** and then click **Restart** to confirm the operation.

4.3. Click the **Console** tab and then log in to the console of the `www1` VM as the `root` user with the `redhat` password. Use the `systemctl stop` command to stop the `httpd` service on the VM.

```
...output trimmed...
www1 login: root
Password: redhat
[root@www1 ~]# systemctl stop httpd
```

4.4. 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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get endpoints -n ha-review
NAME   ENDPOINTS                    AGE
web    10.8.2.50:80  20m06s
[student@workstation ~]$ oc get vmi -n ha-review
NAME             AGE   PHASE     IP          NODENAME   READY
mariadb-server   14m   Running   10.8.2.52   worker01   True
www1             17m   Running   10.11.0.64  worker02   True
www2             16m   Running   10.8.2.50   worker01   True
```

4.5. From the `www1` console, use the `systemctl start` command to start the `httpd` service.

```
[root@www1 ~]# systemctl start httpd
```

4.6. Return to the terminal on the `workstation` machine. Use the `oc` command to confirm that the `web` service endpoints include the IP address of the `www1` VMI.

```
[student@workstation ~]$ oc get endpoints -n ha-review
NAME   ENDPOINTS                    AGE
web    10.11.0.64:80,10.8.2.50:80  20m06s
[student@workstation ~]$ oc get vmi -n ha-review
NAME             AGE   PHASE     IP          NODENAME   READY
mariadb-server   14m   Running   10.8.2.52   worker01   True
www1             17m   Running   10.11.0.64  worker02   True
www2             16m   Running   10.8.2.50   worker01   True
```

5. Add an `i6300esb` watchdog device named `testwatchdog` on the `www2` 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 prepared the `/home/student/DO316/labs/ha-review/watchdog.yaml` file for your reference. Restart the VM to apply the changes.

Then, log in to the `www2` VM as the `root` user with the `redhat` password. Confirm that the watchdog device works by using the `cat >> /dev/watchdog` command.

5.1. From the terminal on the `workstation` machine, use the `oc edit vm` command to add a watchdog device to the `www2` VM.

You can copy and paste the `watchdog` section from the `/home/student/DO316/labs/ha-review/watchdog.yaml` file that the `lab` command prepared.

```
[student@workstation ~]$ oc edit vm www2 -n ha-review
# 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:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
    ...output omitted...
    spec:
      domain:
        cpu:
          ...output omitted...
        devices:
          disks:
          - bootOrder: 1
            disk:
              bus: virtio
            name: www2
          - disk:
              bus: virtio
            name: cloudinitdisk
          interfaces:
          - macAddress: 02:7e:be:00:00:02
            masquerade: {}
            name: default
          networkInterfaceMultiqueue: true
          rng: {}
          watchdog:
            i6300esb:
              action: poweroff
            name: testwatchdog
        machine:
          type: pc-q35-rhel8.4.0
...output omitted...
```

If you prefer the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `www2` VM, and then navigate to the **YAML** tab. Use the YAML editor to declare the watchdog device and then click **Save**.

5.2. Restart the VM to regenerate the `www2` VMI. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines** and select the `www2` VM. Click **Actions** > **Restart** and then click **Restart** to confirm the operation.

5.3. Confirm that the watchdog device works on the `www2` VM. Click the **Console** tab and log in as the `root` user with the `redhat` password. Afterwards, use the `cat >> /dev/watchdog` command to simulate an unresponsive operating system. Confirm that the VM is restarted.

```
...output trimmed...
www2 login: root
Password: redhat
[root@www2 ~]# cat >> /dev/watchdog
```

The preceding `cat` command does not return output and does not exit until the `www2` VM is automatically restarted.

6. The MariaDB database 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.

The `lab` command prepared the `/home/student/DO316/labs/ha-review/liveness.yaml` file for you. Reference this file to create the liveness probe on the `mariadb-server` VM.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

After configuring the liveness probe, restart the VM to regenerate the `mariadb-server` VMI.

Confirm that Kubernetes automatically restarts the `mariadb-server` VM if the liveness probe fails. Log in to the VM console as the `root` user with the `redhat` password. Stop the `mysql` service and confirm that the VM is restarted by Kubernetes.

6.1. From the terminal on the `workstation` machine, use the `oc edit vm` command to add the liveness probe to the `mariadb-server` VM.

You can copy and paste the `livenessProbe` section from the `/home/student/DO316/labs/ha-review/liveness.yaml` file that the `lab` command prepared.

```
[student@workstation ~]$ oc edit vm mariadb-server -n ha-review
# 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:
          port: 3306
        initialDelaySeconds: 10
        periodSeconds: 5
      evictionStrategy: LiveMigrate
      hostname: mariadb-server
...output omitted...
```

If you prefer the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**, select the `mariadb-server` VM, and then navigate to the **YAML** tab. Use the YAML editor to declare the probe and then click **Save**.

6.2. You must delete the `mariadb-server` VMI resource before Red Hat OpenShift Virtualization recognizes the new probe. OpenShift Virtualization restarts a new instance from the VM resource that you modified.

The following `oc delete` command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi mariadb-server -n ha-review
virtualmachineinstance.kubevirt.io "mariadb-server" deleted
```

6.3. Confirm that a new VM instance is running. You might have to wait a few minutes and run the command several times.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get vmi mariadb-server -n ha-review
NAME             AGE   PHASE     IP          NODENAME   READY
mariadb-server   14s   Running   10.8.2.62   worker01   True
```

6.4. Test the probe by stopping the `mysql` service running inside the `mariadb-server` VM. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `mariadb-server` VM and then navigate to the **Console** tab.

Log in as the `root` user with the `redhat` password and then stop the `mysql` service.

```
mariadb-server login: root
Password: redhat
[root@mariadb-server ~]# systemctl stop mysql
[root@mariadb-server ~]#
```

Because the liveness probe fails, the VM restarts after a few seconds.

7. Monitor the availability of the web application during a node failure. Open a new terminal window and then execute the `loop.sh` file in the `/home/student/DO316/labs/ha-review/` directory that the `lab` command prepared for you.

Prevent new workloads from scheduling on the node that is hosting the `www2` VMI and then drain the node of its workloads.

7.1. Open a new terminal window on the `workstation` machine. Execute the `loop.sh` file in the `~/DO316/labs/ha-review/` directory.

```
[student@workstation ~]$ sh ~/DO316/labs/ha-review/loop.sh
Welcome to www2
Welcome to www2
Welcome to www1
Welcome to www1
...output trimmed...
```

7.2. On the `workstation` machine, return to the terminal window where you confirmed the new VMI instance. Identify the node that is hosting the `www2` VMI.

```
[student@workstation ~]$ oc get vmis -n ha-review
NAME             AGE     PHASE     IP          NODENAME   READY
mariadb-server   9m39s   Running   10.8.2.62   worker01   True
www1             16m     Running   10.11.0.64  worker02   True
www2             14m     Running   10.8.2.60   worker01   True
```

Important

The identified node in your lab might be different than in the example output. Be sure to modify your commands accordingly.

7.3. Prevent new workloads from scheduling on the identified node.

```
[student@workstation ~]$ oc adm cordon worker01
node/worker01 cordoned
[student@workstation ~]$ oc get nodes
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
NAME       STATUS                     ROLES           AGE     VERSION
master01   Ready                      master,worker   5d11h   v1.23.3+e419edf
master02   Ready                      master,worker   5d11h   v1.23.3+e419edf
master03   Ready                      master,worker   5d11h   v1.23.3+e419edf
worker01   Ready,SchedulingDisabled   worker          5d10h   v1.23.3+e419edf
worker02   Ready                      worker          5d10h   v1.23.3+e419edf
```

7.4. Drain the identified 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
```

7.5. Return the terminal window that is executing the `loop.sh` file. Notice that only the `www1` VM is responding to requests.

```
[student@workstation ~]$ sh ~/DO316/labs/ha-review/loop.sh
Welcome to www1
Welcome to www1
Welcome to www1
Welcome to www1
...output trimmed...
```

7.6. Return to the terminal window where you ran the `oc adm drain` command. Retrieve the status of the `www2` VMI.

```
[student@workstation ~]$ oc get vmi -n ha-review
NAME             AGE   PHASE       IP           NODENAME   READY
mariadb-server   15m   Running     10.11.0.66   worker02   True
www1             25m   Running     10.11.0.64   worker02   True
www2             19m   Succeeded   10.8.2.60    worker01   False
```

The `www2` VM did not automatically fail over to another cluster node.

8. 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 is responding to requests.

8.1. From the OpenShift web console, navigate to **Virtualization** > **VirtualMachines**. Select the `www2` VM and click the **Details** tab. Navigate to the **Scheduling and resource requirements** section and click **No Eviction Strategy** in the `Eviction Strategy` subsection. Click the **LiveMigrate** flag and click **Save**.

8.2. 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.

8.3. From a terminal on the `workstation` machine, confirm that the `www2` VMI is scheduled on another node.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get vmi -n ha-review
NAME              AGE   PHASE       IP           NODENAME   READY
mariadb-server    17m   Running     10.11.0.66   worker02   True
www1              27m   Running     10.11.0.64   worker02   True
www2              22m   Running     10.11.0.68   worker02   True
```

8.4. Return to the terminal window on the `workstation` machine that is executing the `loop.sh` file. Confirm that the `www2` VM is responding to requests. Press `Ctrl+C` to stop the command and then close the terminal window.

```
[student@workstation ~]$ sh ~/DO316/labs/ha-review/loop.sh
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
...output trimmed...
```

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, change to the `student` user home directory and use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ha-review
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Summary

- You can use health checks to monitor the availability of a VM's application and perform necessary actions by using readiness and liveness probes.
- Use the HTTP GET or TCP socket methods for monitoring your application.
- Virtual machines instances have settings that control the running state of the VM. The `spec.running` and `spec.runstrategy` settings are mutually exclusive, so using both returns an error.
- The watchdog feature detects and restarts unresponsive operating systems, but it does not detect application failures.
- Eviction strategies, such as Live Migration, determine the appropriate actions to take with orphaned resources, such as pods, persistent volumes, and VMs.
- When an eviction strategy is not defined, VMs are terminated if the node is placed into maintenance or drained.
- Machine health checks automatically remediate an unhealthy machine if it exists in a particular machine pool.
- Standard remediation defines the draining of the remaining VMs on the failed node and reschedules them to healthy nodes so that the failed node can reregister within the cluster and restore the annotations and labels.
- Power-based remediation uses a power controller to power off an inoperable node.
- Available in non-IPI bare-metal clusters, the Poison Pill Operator runs on the cluster nodes and reboots nodes that are unhealthy.
- VMs configured with `RunStrategy: Always` setting are not automatically relocated to healthy nodes.
- Deleting a node deletes the node object in Kubernetes, but the pods that reside on the node are not deleted.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Chapter 9

Comprehensive Review

Goal

Review tasks from *Managing Virtual Machines with Red Hat OpenShift Virtualization*.

Objectives

- Review tasks from *Managing Virtual Machines with Red Hat OpenShift Virtualization*.

Sections

- Comprehensive Review

Lab

- Deploying OpenShift Virtualization and Configuring External Network
- Using Templates to Deploy Virtual Machines
- Cloning Virtual Machines and Configuring Load Balancing

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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, Introducing Red Hat OpenShift Virtualization

Distinguish Red Hat OpenShift Virtualization from container technologies and traditional virtual machine technologies. Describe the features and use cases of OpenShift Virtualization.

- Describe the features and use cases of OpenShift Virtualization.
- Describe Kubernetes high-level architecture and operators.
- 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.

- Describe and compare virtual machine instance components.
- Provision and access a new virtual machine.
- Monitor VM resources and inspect a VM's configuration.

Chapter 3, Configuring Kubernetes Networking for Virtual Machines

Configure standard Kubernetes network objects and external access for VMs and virtual machine-backed applications.

- Describe VM communication on the Kubernetes SDN, create a ClusterIP service for a VM, and create a network policy for virtual machines.
- Configure external access for virtual machines.

Chapter 4, Connecting Virtual Machines to External Networks

Configure node networking to connect virtual machines and nodes to networks that are outside the cluster.

- Describe the Multus CNI plug-in and use cases.

- Configure multihomed nodes and virtual machines using the NMstate operator and Multus.

Chapter 5, Configuring Kubernetes Storage for Virtual Machines

Manage storage and disks for VMs in Red Hat OpenShift.

- Configure appropriate services for persistent VM storage.
- Attach disks to a VM.
- Connect a VM to external storage using Multus.

Chapter 6, Virtual Machine Template Management

Create and manage templates to provision virtual machines.

- Use and modify preconfigured templates to provision VMs.
- Generate custom templates to provision VMs.

Chapter 7, Advanced Virtual Machine Management

Snapshot, clone, and live migrate a virtual machine and initiate node maintenance.

- Describe VM snapshots and execute snapshot management tasks.
- Clone an existing VM's PVC as a new VM and as a new data volume to provision additional VMs.
- Describe and manage VM live migrations.
- Set a node into maintenance mode and describe how the OpenShift Virtualization update process affects VMs.

Chapter 8, Configuring Kubernetes High Availability for Virtual Machines

Configure Kubernetes resources to implement high availability for virtual machines.

- Demonstrate VM load balancing with Kubernetes networking resources.
- Configure health probes and a Watchdog device to monitor the health and responsiveness of a VM and its services.
- Configure Kubernetes resources to help VMs fail over to another cluster node when node failure is detected.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Deploying OpenShift Virtualization and Configuring External Network

Deploy the Red Hat OpenShift Virtualization operator, configure node networking, and connect a VM to an additional network.

Outcomes

- Deploy the OpenShift Virtualization operator.
- Configure cluster nodes with additional networks.
- Prepare projects for additional networks.
- Deploy virtual machines with multiple network interfaces.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and then creates the `review-cr1` namespace.

```
[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. Use the `admin` user with the `redhat` password to log in to your OpenShift cluster. The `oc` command is available on the `workstation` machine.

- Install the OpenShift Virtualization Operator in the `openshift-cnv` namespace. Use the `stable` channel and configure the operator for automatic update approval. Use the default options when creating the HyperConverged instance.

> **Note**
> If the `kubevirt-hyperconverged` resource already exists, then an error message displays. Skip the step and continue with the exercise.

- Add the `orgnet: true` label to the `worker01` and `worker02` cluster nodes.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

- The `worker01` and `worker02` cluster nodes have a network interface named `ens4` connected to an additional network. Create a Linux bridge named `br1` on each node, and then add the `ens4` interface to the bridge. Configure the bridge interface for DHCP. Use the `orgnet: true` label to select the nodes to configure.

  The `lab` command prepared the `/home/student/DO316/labs/review-cr1/conf-policy.yaml` file that you can use to create the resource.

- 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	`CNV Linux bridge`
Bridge	`br1`

- Create a *virtual machine (VM)* named `web1` in the `review-cr1` namespace from the `Red Hat Enterprise Linux 8.0+ VM` template. Use the http://utility.lab.example.com:8080/openshift4/images/helloworld.qcow2 URL for the boot source. Add a second network interface named `nic-0` connected to the bridge that you declared in the `ext-net` network attachment definition resource.

  The `lab` command prepared the `/home/student/DO316/labs/review-cr1/vm_parameters.txt` file that lists these parameters. You can use the file to copy and then paste the parameters into the OpenShift web console.

- 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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Deploying OpenShift Virtualization and Configuring External Network

Deploy the Red Hat OpenShift Virtualization operator, configure node networking, and connect a VM to an additional network.

Outcomes

- Deploy the OpenShift Virtualization operator.
- Configure cluster nodes with additional networks.
- Prepare projects for additional networks.
- Deploy virtual machines with multiple network interfaces.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and then creates the `review-cr1` namespace.

```
[student@workstation ~]$ lab start review-cr1
```

1. From a terminal on the `workstation` machine, use the `oc` command to log in to your OpenShift cluster as the `admin` user with the `redhat` 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 terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 the `redhat` password when prompted.

2. Install the OpenShift Virtualization operator in the `openshift-cnv` namespace.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2.1. Navigate to **Operators** > **OperatorHub**.

2.2. Type `virtualization` in the **OperatorHub** search bar and then click the **OpenShift Virtualization** card.

2.3. Click **Install**. Verify the default options and then click **Install**.

- **Update channel** is set to **stable**.
- **Installation mode** is set to **A specific namespace on the cluster**.
- **Installed Namespace** is set to `openshift-cnv`.
- **Update approval** is set to **Automatic**.

2.4. After the OpenShift Virtualization operator installation completes, click **Create HyperConverged**. Accept the default options and then click **Create**.

If the `kubevirt-hyperconverged` resource already exists, then an error message is displayed. Click **Cancel** and continue with the exercise.

2.5. Navigate to **Operators** > **Installed Operators** and then wait for the OpenShift Virtualization operator to report the `Succeeded` status.

3. From the terminal on the `workstation` machine, use the `oc` command to add the `orgnet: true` label to the `worker01` and `worker02` cluster nodes.

```
[student@workstation ~]$ oc label node worker01 orgnet=true
node/worker01 labeled
[student@workstation ~]$ oc label node worker02 orgnet=true
node/worker02 labeled
```

4. Configure a Linux bridge named `br1` for each cluster node that has the `orgnet: true` label. Add the `ens4` interface to the bridge.

4.1. Review the `/home/student/DO316/labs/review-cr1/conf-policy.yaml` file that the `lab` command prepared. You do not have to change anything in that file.

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-ens4-policy
spec:
  nodeSelector:
    orgnet: "true"
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with ens4 as a port
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
                enabled: false
            port:
              - name: ens4
```

4.2. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/review-cr1/conf-policy.yaml
nodenetworkconfigurationpolicy.nmstate.io/br1-ens4-policy created
```

4.3. Monitor the progress of the configuration policy. You might have to wait a few minutes and rerun the following command several times for the command to report the `Available` status.

```
[student@workstation ~]$ oc get nnce
NAME                        STATUS
worker01.br1-ens4-policy    Available
worker02.br1-ens4-policy    Available
```

5. Create a network attachment definition resource named `ext-net` in the `review-cr1` namespace.

5.1. From the OpenShift web console, navigate to **Networking** > **NetworkAttachmentDefinitions**, select the `review-cr1` project, and then click **Create network attachment definition**.

5.2. Complete the form using the following information and then click **Create**.

Field	Value
Name	`ext-net`
Network Type	`CNV Linux bridge`
Bridge Name	`br1`

6. Use the `Red Hat Enterprise Linux 8.0+ VM` template to create the `web1` VM in the `review-cr1` namespace. Add a second network interface named `nic-0`.

6.1. Navigate to **Virtualization** > **Templates** and then set the project to `review-cr1`.

6.2. Click **Create VM** to the right of the `Red Hat Enterprise Linux 8.0+ VM` template. Set **Boot source type** to `URL (creates PVC)` and then set **Import URL** to http://utility.lab.example.com:8080/openshift4/images/helloworld.qcow2. Click **Customize virtual machine**.

6.3. Set the **Name** field to `web1` and then click **Next**.

6.4. Click **Add Network Interface**, complete the form using the following information, and then click **Add**.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Field	Value
Name	`nic-0`
Model	`virtio`
Network	`ext-net`
Type	`Bridge`

6.5. Click **Review and confirm** and then click **Create Virtual Machine**.

6.6. Navigate to **Virtualization** > **VirtualMachines** and then wait for the `web1` VM to have the `Running` status.

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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Using Templates to Deploy Virtual Machines

Create a virtual machine template, use that template to deploy a virtual machine, and set a cluster node into maintenance mode.

Outcomes

- Create virtual machine templates.
- Deploy virtual machines from templates.
- Manage user access rights.
- Set cluster nodes into maintenance mode.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and creates the `review-cr2` namespace.

```
[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 the `redhat` password to log in to your OpenShift cluster. The `oc` command is available on the `workstation` machine. During the exercise, you also use the `developer` user account with the `developer` password.

- Create a *virtual machine (VM)* template in the `review-cr2` project with the following characteristics:

Parameter	Value
Project	`review-cr2`
Template name	`dev-web-rhel8`
Template provider	`Red Hat Training`

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Parameter	Value
Operating system	`Red Hat Enterprise Linux 8.0 or higher`
Boot disk URL	http://utility.lab.example.com:8080/openshift4/images/helloworld.qcow2
Flavor	`Tiny`
Workload type	`Server`
Root disk size	10 GiB
Root disk interface	`virtio`
Root disk storage class	`ocs-external-storagecluster-ceph-rbd`

The `lab` command prepared the `/home/student/DO316/labs/review-cr2/template_parameters.txt` file that lists these parameters. You can use the file to copy and then paste the parameters into the OpenShift web console.

- Grant `admin` access to the `developer` user for the `review-cr2` project.
- As the `developer` user, create a VM named `web1` in the `review-cr2` namespace. Use the `dev-web-rhel8` template.
- Set the `worker02` cluster node in maintenance mode and move its workload to the remaining nodes. The `lab` command prepared the `/home/student/DO316/labs/review-cr2/nm.yaml` file that you can use as an example.
- 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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Using Templates to Deploy Virtual Machines

Create a virtual machine template, use that template to deploy a virtual machine, and set a cluster node into maintenance mode.

Outcomes

- Create virtual machine templates.
- Deploy virtual machines from templates.
- Manage user access rights.
- Set cluster nodes into maintenance mode.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and creates the `review-cr2` namespace.

```
[student@workstation ~]$ lab start review-cr2
```

1. From a terminal on the `workstation` machine, use the `oc` command to log in to your OpenShift cluster as the `admin` user with the `redhat` 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 terminal, log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  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 the `redhat` password when prompted.

2. Create the `dev-web-rhel8` VM template in the `review-cr2` project.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

2.1. Navigate to **Virtualization** > **Templates**, set the project to `review-cr2`, and then click **Create** > **With Wizard**.

2.2. Complete the form using the following information and then click **Next**.

Field	Value
Name	`dev-web-rhel8`
Template provider	`Red Hat Training`
Operating System	`Red Hat Enterprise Linux 8.0 or higher`
Boot Source	`URL (creates PVC)`
URL	`http://utility.lab.example.com:8080/openshift4/images/helloworld.qcow2`
Flavor	`Tiny: 1 CPU | 1 GiB Memory`
Workload Type	`Server (default)`

2.3. Do not make any changes to the network configuration. Click **Next**.

2.4. Click the vertical ellipsis to the right of the `rootdisk` disk and then click **Edit**. Complete the form using the following information and then click **Save**.

Field	Value
Size	10 GiB
Interface	`virtio`
StorageClass	`ocs-external-storagecluster-ceph-rbd`

2.5. Click **Review and confirm** and then click **Create Virtual Machine template**.

3. Grant `admin` access to the `developer` user for the `review-cr2` project.

3.1. From the terminal on the `workstation` machine, use the `oc adm policy` command to grant `admin` access to the `developer` user.

```
[student@workstation ~]$ oc adm policy add-role-to-user admin developer \
  -n review-cr2
Warning: User 'developer' not found
clusterrole.rbac.authorization.k8s.io/admin added: "developer"
```

You can ignore the warning message regarding the missing user. Because you have not yet used the account, OpenShift does not have a record of the user.

3.2. Confirm that the `developer` user has `admin` rights.

```
[student@workstation ~]$ oc get rolebindings -n review-cr2 -o wide
NAME   ROLE                AGE   USERS      GROUPS     SERVICEACCOUNTS
admin  ClusterRole/admin   39s   developer
...output omitted...
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

4. As the `developer` user, use the `dev-web-rhel8` template to create the `web1` 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 the `developer` password when prompted.

4.3. Select the **Administrator** perspective from the menu at the top of the menu on the left.

4.4. Navigate to **Virtualization** > **Templates** and then set the project to `review-cr2`.

4.5. Click **Create VM** to the right of the `dev-web-rhel8` template. Set the **Virtual Machine Name** field to `web1` and then click **Create virtual machine**.

4.6. Navigate to **Virtualization** > **VirtualMachines** and then wait for the `web1` VM to have the `Running` status.

5. Set the `worker02` cluster node in maintenance mode and drain its workload.

5.1. From the terminal on the `workstation` machine, edit the `/home/student/DO316/labs/review-cr2/nm.yaml` resource file that the `lab` command prepared.

```
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
metadata:
  name: maintenance-node
spec:
  nodeName: worker02
  reason: "Maintenance required for the node"
```

5.2. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f ~/DO316/labs/review-cr2/nm.yaml
nodemaintenance.nodemaintenance.kubevirt.io/maintenance-node created
```

5.3. Confirm that the node has the `SchedulingDisabled` status.

```
[student@workstation ~]$ oc get nodes
NAME       STATUS                     ROLES           AGE   VERSION
master01   Ready                      master,worker   12d   v1.23.3+e419edf
master02   Ready                      master,worker   12d   v1.23.3+e419edf
master03   Ready                      master,worker   12d   v1.23.3+e419edf
worker01   Ready                      worker          12d   v1.23.3+e419edf
worker02   Ready,SchedulingDisabled   worker          12d   v1.23.3+e419edf
```

5.4. Monitor the `NodeMaintenance` resource named `maintenance-node` to confirm that the drain operation completes. You might have to wait a few minutes and rerun the command several times to get the `Succeeded` status.

```
[student@workstation ~]$ oc get NodeMaintenance maintenance-node -o yaml
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
...output omitted...
```

```
status:
  evictionPods: 27
  lastError: 'error when waiting for pod "router-default-699bc7764c-8cdfx"
 terminating:
    global timeout reached: 30s'
  phase: Succeeded
  totalpods: 45
```

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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Lab

Cloning Virtual Machines and Configuring Load Balancing

Clone a virtual machine and then configure the clones to provide a highly available web application.

Outcomes

- Clone virtual machines.
- Take snapshots of virtual machines.
- Configure readiness probes.
- Attach PVCs as disks to virtual machines.
- Create services of the `ClusterIP` type.
- Link services to virtual machines.
- Create routes.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It 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 with the `redhat` password to log in to your OpenShift cluster. The `oc` command is available on the `workstation` machine. All the resources you create must belong to the `review-cr3` namespace.

- Clone the `golden-web` *virtual machine (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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

- The web application 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 prepared the `/home/student/DO316/labs/review-cr3/readiness.yaml` file that you can use as an example.

- The `lab` command prepared the `web1-documentroot` and `web2-documentroot` PVCs. They contain web content for the `web1` and `web2` VMs. Attach the `web1-documentroot` PVC to the `web1` VM and attach the `web2-documentroot` PVC 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 the `redhat` password. Run the `/root/mount.sh` script that the `lab` command prepared. The script mounts the additional disk under 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` of the `ClusterIP` type. The service must dispatch the web requests between the VMs that have the `tier: front` label. These VMs host a web service that listens on port TCP 80. The `lab` command prepared the `/home/student/DO316/labs/review-cr3/service.yaml` file that 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 http://front-review-cr3.apps.ocp4.example.com.

  You can test the route and the service by using the `curl http://front-review-cr3.apps.ocp4.example.com` command from the terminal 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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Solution

Cloning Virtual Machines and Configuring Load Balancing

Clone a virtual machine and then configure the clones to provide a highly available web application.

Outcomes

- Clone virtual machines.
- Take snapshots of virtual machines.
- Configure readiness probes.
- Attach PVCs as disks to virtual machines.
- Create services of the `ClusterIP` type.
- Link services to virtual machines.
- Create routes.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable. It also creates the `review-cr3` namespace and starts the `golden-web` virtual machine in that namespace.

```
[student@workstation ~]$ lab start review-cr3
```

1. From a terminal on the `workstation` machine, use the `oc` command to log in to your OpenShift cluster as the `admin` user with the `redhat` 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 terminal, log in to your OpenShift cluster as the `admin` user.

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


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

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 the `redhat` password when prompted.

2. Clone the `golden-web` VM to a new VM named `web1`.

2.1. The cloning process requires that you stop the source machine.

Navigate to **Virtualization** > **VirtualMachines**, select the `golden-web` VM, and then click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

2.2. Click **Actions** > **Clone**. Set the VM name to `web1`, select **Start virtual machine on clone**, and then click **Clone**.

2.3. Navigate 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, navigate to the **Snapshots** tab, and then click **Take Snapshot**.

3.2. Enter `web1-snap1` in the **Snapshot Name** field. Select **I am aware of this warning and wish to proceed** and then click **Save**.

3.3. Wait for the snapshot to report a `Ready` status.

4. The web servers running inside the `web1` and `web2` VMs expose the `/cgi-bin/health` endpoint that you can use 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. From the terminal on the `workstation` machine, use the `oc edit vm` command to add the readiness probe to the `web1` VM.

You can copy and then paste the `readinessProbe` section from the `/home/student/DO316/labs/review-cr3/readiness.yaml` file that the `lab` command prepared.

```
[student@workstation ~]$ oc edit vm web1 -n review-cr3
# 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...
```


For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
      readinessProbe:
        httpGet:
          path: /cgi-bin/health
          port: 80
        periodSeconds: 5
        failureThreshold: 2
      evictionStrategy: LiveMigrate
      hostname: web1
...output omitted...
```

If you prefer the OpenShift web console, then navigate to **Virtualization** > **VirtualMachines**, select the `web1` VM, and then navigate to the **YAML** tab. Use the YAML editor to declare the probe and then click **Save**.

4.2. You must delete the `web1` VMI resource before OpenShift Virtualization recognizes the new probe. OpenShift Virtualization restarts a new instance from the VM resource that you modified.

The following `oc delete` command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi web1 -n review-cr3
virtualmachineinstance.kubevirt.io "web1" deleted
```

4.3. Confirm that a new VM instance is running. You might have to wait a few minutes and run the command several times.

```
[student@workstation ~]$ oc get vmi web1 -n review-cr3
NAME   AGE   PHASE     IP           NODENAME   READY
web1   95s   Running   10.11.0.48   worker01   True
```

4.4. Repeat this step to add the readiness probe to the `web2` VM.

5. Attach the `web1-documentroot` PVC as a disk to the `web1` VM by using a `virtio` interface.

5.1. Use the OpenShift web console to stop the `web1` VM. Navigate to **Virtualization** > **VirtualMachines**, select the `web1` VM, and then click **Actions** > **Stop** to stop the VM. Click **Stop** to confirm the operation. Wait for the machine to stop.

5.2. Navigate to the **Disks** tab and then click **Add Disk**. Complete the form using the following information.

> **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.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

Field	Value
Source	`Use an existing PVC`
PersistentVolumeClaim	`web1-documentroot`
Type	`Disk`
Interface	`virtio`

Click **Add** to attach the disk to the VM.

5.3. Click **Actions** > **Start** to start the VM.

5.4. Repeat this step to attach the `web2-documentroot` PVC to the `web2` VM.

6. Log in to the `web1` VM console and then run the `/root/mount.sh` script.

6.1. Navigate to **Virtualization** > **VirtualMachines**, select the `web1` VM, and then navigate to the **Console** tab.

6.2. Log in to the VM console as the `root` user with the `redhat` 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.

```
[root@golden-web ~]# logout
```

6.5. Repeat this step to mount the disk on the `web2` VM.

7. Add the `tier: front` label to the `virt-launcher` pod associated with the `web1` VM.

7.1. From the terminal on the `workstation` machine, use the `oc edit` command to edit the configuration of the `web1` VM.

```
[student@workstation ~]$ oc edit vm web1 -n review-cr3
```

7.2. Add the `tier: front` label under the `.spec.template.metadata.labels` path.

```
...output omitted...
spec:
  dataVolumeTemplates:
  ...output omitted...
  template:
    metadata:
      creationTimestamp: null
      labels:
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
        tier: front
        flavor.template.kubevirt.io/small: "true"
        kubevirt.io/domain: golden-web
        kubevirt.io/size: small
        ...output omitted...
```

7.3. You must delete the `web1` VMI resource before OpenShift Virtualization recognizes the new label. OpenShift Virtualization restarts a new instance from the VM resource that you modified.

The following `oc delete` command might take a few minutes to complete.

```
[student@workstation ~]$ oc delete vmi web1 -n review-cr3
virtualmachineinstance.kubevirt.io "web1" deleted
```

7.4. Confirm that a new VM instance is running. You might have to wait a few minutes and run the command several times.

```
[student@workstation ~]$ oc get vmi web1 -n review-cr3
NAME   AGE   PHASE     IP           NODENAME   READY
web1   95s   Running   10.11.0.48   worker01   True
```

7.5. Repeat this step to add the same label to the `web2` VM.

8. Create a service named `front` of the `ClusterIP` type. 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 terminal on the `workstation` machine, create the `front.yaml` resource definition file 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 prepared the `/home/student/DO316/labs/review-cr3/service.yaml` file as an example.

8.2. Use the `oc apply` command to create the resource.

```
[student@workstation ~]$ oc apply -f front.yaml
service/front created
```

8.3. Confirm that the `front` service has two active endpoints.

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
[student@workstation ~]$ oc get endpoints -n review-cr3
NAME    ENDPOINTS                    AGE
front   10.11.0.48:80,10.8.2.36:80   76m
```

9. Create a route so that you can access the web application running inside the VMs by using the http://front-review-cr3.apps.ocp4.example.com URL.

9.1. Expose the `front` service.

```
[student@workstation ~]$ oc expose service front -n review-cr3
route.route.openshift.io/front exposed
```

9.2. Use the `curl` command from the `workstation` machine to confirm that you can access the web application from outside the cluster.

```
[student@workstation ~]$ curl http://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 http://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>
[student@workstation ~]$ curl http://front-review-cr3.apps.ocp4.example.com
...output omitted...
  <title>Test page web1</title>
...output omitted...
[student@workstation ~]$ curl http://front-review-cr3.apps.ocp4.example.com
...output omitted...
  <title>Test page web1</title>
...output omitted...
[student@workstation ~]$ curl http://front-review-cr3.apps.ocp4.example.com
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.

```
...output omitted...
  <title>Test page web2</title>
...output omitted...
```

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
```

For use by Naresh D naresh.deenadayalan@hcl.com naresh.deenadayalan@hcl.com Copyright © 2024 Red Hat, Inc.