



OpenShift Container Platform 4.1

Container-native virtualization

Container-native virtualization installation, usage, and release notes

OpenShift Container Platform 4.1 Container-native virtualization

Container-native virtualization installation, usage, and release notes

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered 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 and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. 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 an official 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 OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides information about how to use container-native virtualization in OpenShift Container Platform 4.1

Table of Contents

CHAPTER 1. CONTAINER-NATIVE VIRTUALIZATION INSTALLATION	6
1.1. ABOUT CONTAINER-NATIVE VIRTUALIZATION	6
1.1.1. What you can do with container-native virtualization	6
1.1.2. Container-native virtualization support	6
1.2. PREPARING YOUR CLUSTER FOR CONTAINER-NATIVE VIRTUALIZATION	6
1.3. INSTALLING CONTAINER-NATIVE VIRTUALIZATION	7
1.3.1. Preparing to install container-native virtualization	7
1.3.2. Subscribing to the KubeVirt HyperConverged Cluster Operator catalog	8
1.3.3. Deploying container-native virtualization	8
1.4. INSTALLING THE VIRTCTL CLIENT	9
1.4.1. Enabling container-native virtualization repositories	9
1.4.2. Installing the virtctl client	9
1.5. UNINSTALLING CONTAINER-NATIVE VIRTUALIZATION	10
1.5.1. Deleting the KubeVirt HyperConverged custom resource	10
1.5.2. Deleting the KubeVirt HyperConverged Cluster Operator catalog subscription	11
1.5.3. Deleting a project using the web console	11
CHAPTER 2. CONTAINER-NATIVE VIRTUALIZATION USER'S GUIDE	12
2.1. CREATING VIRTUAL MACHINES	12
2.1.1. Running the virtual machine wizard to create a virtual machine	12
2.1.1.1. Virtual machine wizard fields	13
2.1.1.2. Cloud-init fields	14
2.1.1.3. Networking fields	15
2.1.1.4. Storage fields	15
2.1.2. Pasting in a pre-configured YAML file to create a virtual machine	16
2.1.3. Using the CLI to create a virtual machine	16
2.1.4. Virtual machine storage volume types	17
2.2. TLS CERTIFICATES FOR DATAVOLUME IMPORTS	18
2.2.1. Adding TLS certificates for authenticating DataVolume imports	18
2.2.2. Example: ConfigMap created from a TLS certificate	18
2.3. IMPORTING A VMWARE VIRTUAL MACHINE OR TEMPLATE WITH THE VIRTUAL MACHINE WIZARD	19
2.3.1. Uploading the VMware Virtual Disk Development Kit	19
2.3.2. Importing the VMware virtual machine or template	20
2.3.3. Updating the imported virtual machine's NIC name	21
2.3.4. Virtual machine wizard fields	22
2.3.4.1. Virtual machine wizard fields	22
2.3.4.2. Cloud-init fields	23
2.3.4.3. Networking fields	24
2.3.4.4. Storage fields	24
2.4. IMPORTING VIRTUAL MACHINE IMAGES WITH DATAVOLUMES	25
2.4.1. CDI supported operations matrix	25
2.4.2. About DataVolumes	26
2.4.3. Importing a virtual machine image into a container-native virtualization object with DataVolumes	26
2.4.4. Template: DataVolume virtual machine configuration file	28
2.4.5. Template: DataVolume import configuration file	29
2.5. EDITING VIRTUAL MACHINES	30
2.5.1. Using the web console to edit a virtual machine	30
2.5.2. Editing the virtual machine YAML configuration	30
2.5.3. Using the CLI to edit a virtual machine	31
2.6. DELETING VIRTUAL MACHINES	31
2.6.1. Deleting a virtual machine using the web console	32

2.6.2. Deleting a virtual machine and PVCs using the CLI	32
2.7. CONTROLLING VIRTUAL MACHINES STATES	32
2.7.1. Controlling virtual machines from the web console	33
2.7.1.1. Starting a virtual machine	33
2.7.1.2. Restarting a virtual machine	33
2.7.1.3. Stopping a virtual machine	34
2.7.2. CLI reference for controlling virtual machines	34
2.7.2.1. start	34
2.7.2.2. restart	34
2.7.2.3. stop	35
2.7.2.4. list	35
2.8. ACCESSING VIRTUAL MACHINE CONSOLES	35
2.8.1. Virtual machine console sessions	35
2.8.1.1. Connecting to a virtual machine with the web console	36
2.8.1.2. Connecting to the serial console	36
2.8.1.3. Connecting to the VNC console	36
2.8.1.4. Connecting to the RDP console	37
2.8.2. Accessing virtual machine consoles by using CLI commands	37
2.8.2.1. Accessing a virtual machine instance via SSH	37
2.8.2.2. Accessing the serial console of a virtual machine instance	38
2.8.2.3. Accessing the graphical console of a virtual machine instances with VNC	38
2.8.2.4. Connecting to a Windows virtual machine with an RDP console	39
2.9. USING THE CLI TOOLS	40
2.9.1. Virtctl client commands	40
2.9.2. OpenShift Container Platform client commands	41
2.10. AUTOMATING MANAGEMENT TASKS	41
2.10.1. About Red Hat Ansible Automation	41
2.10.2. Automating virtual machine creation with Red Hat Ansible Automation	42
2.10.3. Example: Ansible Playbook for creating virtual machines	43
2.11. USING THE DEFAULT POD NETWORK WITH CONTAINER-NATIVE VIRTUALIZATION	44
2.11.1. Configuring masquerade mode from the command line	44
2.11.2. Web console	45
2.11.2.1. Networking fields	45
2.11.3. Configuration file examples	45
2.11.3.1. Template: virtual machine configuration file	45
2.11.3.2. Template: Windows virtual machine instance configuration file	46
2.12. ATTACHING A VIRTUAL MACHINE TO MULTIPLE NETWORKS	47
2.12.1. Container-native virtualization networking glossary	47
2.12.2. Connecting a resource to a bridge-based network	48
2.12.3. Creating a NIC for a virtual machine	49
2.12.4. Networking fields	50
2.13. INSTALLING THE QEMU GUEST AGENT ON VIRTUAL MACHINES	50
2.13.1. Installing QEMU guest agent on a Linux virtual machine	50
2.14. VIEWING THE IP ADDRESS OF VNICs ON A VIRTUAL MACHINE	51
2.14.1. Viewing the IP address of a virtual machine interface in the CLI	51
2.14.2. Viewing the IP address of a virtual machine interface in the web console	52
2.15. CONFIGURING PXE BOOTING FOR VIRTUAL MACHINES	52
2.15.1. Container-native virtualization networking glossary	52
2.15.2. PXE booting with a specified MAC address	53
2.15.3. Template: virtual machine instance configuration file for PXE booting	55
2.16. MANAGING GUEST MEMORY	56
2.16.1. Configuring guest memory overcommitment	56
2.16.2. Disabling guest memory overhead accounting	57

2.17. CREATING VIRTUAL MACHINE TEMPLATES	58
2.17.1. Creating a virtual machine template with the interactive wizard in the web console	58
2.17.2. Virtual machine template interactive wizard fields	59
2.17.2.1. Virtual machine template wizard fields	59
2.17.2.2. Cloud-init fields	60
2.17.2.3. Networking fields	60
2.17.2.4. Storage fields	61
2.18. EDITING A VIRTUAL MACHINE TEMPLATE	61
2.18.1. Editing a virtual machine template in the web console	62
2.19. DELETING A VIRTUAL MACHINE TEMPLATE	62
2.19.1. Deleting a virtual machine template in the web console	62
2.20. CLONING A VIRTUAL MACHINE DISK INTO A NEW DATAVOLUME	62
2.20.1. About DataVolumes	63
2.20.2. Cloning the PersistentVolumeClaim of a virtual machine disk into a new DataVolume	63
2.20.3. Template: DataVolume clone configuration file	64
2.20.4. CDI supported operations matrix	64
2.21. CLONING A VIRTUAL MACHINE BY USING A DATAVOLUMETEMPLATE	65
2.21.1. About DataVolumes	65
2.21.2. Creating a new virtual machine from a cloned PersistentVolumeClaim by using a DataVolumeTemplate	65
2.21.3. Template: DataVolume virtual machine configuration file	67
2.21.4. CDI supported operations matrix	68
2.22. UPLOADING LOCAL DISK IMAGES BY USING THE VIRTCTL TOOL	68
2.22.1. CDI supported operations matrix	69
2.22.2. Uploading a local disk image to a new PersistentVolumeClaim	69
2.23. EXPANDING VIRTUAL STORAGE BY ADDING BLANK DISK IMAGES	70
2.23.1. About DataVolumes	70
2.23.2. Creating a blank disk image with DataVolumes	70
2.23.3. Template: DataVolume configuration file for blank disk images	71
2.24. PREPARING CDI SCRATCH SPACE	71
2.24.1. About DataVolumes	71
2.24.2. Understanding scratch space	71
Manual provisioning	72
2.24.3. Defining a StorageClass in the CDI configuration	72
2.24.4. CDI operations that require scratch space	72
2.24.5. CDI supported operations matrix	73
2.25. VIRTUAL MACHINE LIVE MIGRATION	74
2.25.1. Understanding live migration	74
2.26. LIVE MIGRATION LIMITS AND TIMEOUTS	74
2.26.1. Configuring live migration limits and timeouts	74
2.26.2. Cluster-wide live migration limits and timeouts	75
2.27. MIGRATING A VIRTUAL MACHINE INSTANCE TO ANOTHER NODE	75
2.27.1. Initiating live migration of a virtual machine instance in the web console	75
2.27.2. Initiating live migration of a virtual machine instance in the CLI	76
2.28. MONITORING LIVE MIGRATION OF A VIRTUAL MACHINE INSTANCE	76
2.28.1. Monitoring live migration of a virtual machine instance in the web console	77
2.28.2. Monitoring live migration of a virtual machine instance in the CLI	77
2.29. CANCELLING THE LIVE MIGRATION OF A VIRTUAL MACHINE INSTANCE	77
2.29.1. Cancelling live migration of a virtual machine instance in the web console	77
2.29.2. Cancelling live migration of a virtual machine instance in the CLI	78
2.30. NODE MAINTENANCE MODE	78
2.30.1. Understanding node maintenance mode	78
2.31. CONFIGURING VIRTUAL MACHINE EVICTION STRATEGY	79

2.31.1. Configuring custom virtual machines with the LiveMigration eviction strategy	79
2.32. SETTING A NODE TO MAINTENANCE MODE	79
2.32.1. Understanding node maintenance mode	79
2.32.2. Setting a node to maintenance mode in the web console	80
2.32.3. Setting a node to maintenance mode in the CLI	80
2.33. RESUMING A NODE FROM MAINTENANCE MODE	81
2.33.1. Resuming a node from maintenance mode in the web console	81
2.33.2. Resuming a node from maintenance mode in the CLI	81
2.34. INSTALLING VIRTIO DRIVER ON AN EXISTING WINDOWS VIRTUAL MACHINE	82
2.34.1. Understanding VirtIO drivers	82
2.34.2. Supported VirtIO drivers for Microsoft Windows virtual machines	82
2.34.3. Adding VirtIO drivers container disk to a virtual machine	83
2.34.4. Installing VirtIO drivers on an existing Windows virtual machine	84
2.34.5. Removing the VirtIO container disk from a virtual machine	84
2.35. INSTALLING VIRTIO DRIVER ON A NEW WINDOWS VIRTUAL MACHINE	85
2.35.1. Understanding VirtIO drivers	85
2.35.2. Supported VirtIO drivers for Microsoft Windows virtual machines	85
2.35.3. Adding VirtIO drivers container disk to a virtual machine	86
2.35.4. Installing VirtIO drivers during Windows installation	87
2.35.5. Removing the VirtIO container disk from a virtual machine	87
2.36. VIEWING LOGS	88
2.36.1. Understanding logs	88
2.36.2. Viewing virtual machine logs in the CLI	88
2.36.3. Viewing virtual machine logs in the web console	88
2.37. VIEWING EVENTS	89
2.37.1. Understanding events	89
2.37.2. Viewing the events for a virtual machine in the web console	89
2.37.3. Viewing namespace events in the CLI	89
2.37.4. Viewing resource events in the CLI	89
2.38. OPENSIFT CONTAINER PLATFORM CLUSTER MONITORING, LOGGING, AND TELEMETRY	90
2.38.1. About OpenShift Container Platform cluster monitoring	90
2.38.2. About cluster logging	90
2.38.3. About Telemetry	90
2.38.3.1. What information is collected	90
2.38.4. CLI troubleshooting and debugging commands	91
CHAPTER 3. CONTAINER-NATIVE VIRTUALIZATION 2.0 RELEASE NOTES	92
3.1. CONTAINER-NATIVE VIRTUALIZATION 2.0 RELEASE NOTES	92
3.1.1. About container-native virtualization	92
3.1.1.1. What you can do with container-native virtualization	92
3.1.1.2. Container-native virtualization support	92
3.1.2. New and changed features	92
3.1.2.1. Supported binding methods	92
3.1.2.2. Web console improvements	93
3.1.3. Resolved issues	93
3.1.4. Known issues	93

CHAPTER 1. CONTAINER-NATIVE VIRTUALIZATION INSTALLATION

1.1. ABOUT CONTAINER-NATIVE VIRTUALIZATION

Learn about container-native virtualization's capabilities and support scope.

1.1.1. What you can do with container-native virtualization

Container-native virtualization is an add-on to OpenShift Container Platform that allows you to run and manage virtual machine workloads alongside container workloads.

Container-native virtualization adds new objects into your OpenShift Container Platform cluster via Kubernetes custom resources to enable virtualization tasks. These tasks include:

- Creating and managing Linux and Windows virtual machines
- Connecting to virtual machines through a variety of consoles and CLI tools
- Importing and cloning existing virtual machines, including VMware virtual machines
- Managing network interface controllers and storage disks attached to virtual machines
- Live migrating virtual machines between nodes

An enhanced web console provides a graphical portal to manage these virtualized resources alongside the OpenShift Container Platform cluster containers and infrastructure.

1.1.2. Container-native virtualization support



IMPORTANT

container-native virtualization is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see <https://access.redhat.com/support/offerings/techpreview/>.

1.2. PREPARING YOUR CLUSTER FOR CONTAINER-NATIVE VIRTUALIZATION

Container-native virtualization 2.0 works with OpenShift Container Platform by default, however the following installation configurations are recommended:

- The OpenShift Container Platform cluster is installed on [bare metal](#). Manage your Compute nodes in accordance with the number and size of the virtual machines to host in the cluster.
- [Monitoring](#) is configured in the cluster.

1.3. INSTALLING CONTAINER-NATIVE VIRTUALIZATION

Install container-native virtualization to add virtualization functionality to your OpenShift Container Platform cluster.

Before you deploy container-native virtualization, you must create two Custom Resource Definition (CRD) objects:

- **kind: OperatorGroup**
- **kind: CatalogSource**

You can create both objects by running a single command.

To finish installing container-native virtualization, use the OpenShift Container Platform 4.1 [web console](#) to subscribe to and deploy the container-native virtualization Operators.



IMPORTANT

container-native virtualization is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see <https://access.redhat.com/support/offerings/techpreview/>.

1.3.1. Preparing to install container-native virtualization

Before deploying container-native virtualization:

- Create a namespace called **kubevirt-hyperconverged**.
- Create **OperatorGroup** and **CatalogSource** Custom Resource Definition objects (CRDs) in the **kubevirt-hyperconverged** namespace.

Prerequisites

- OpenShift Container Platform 4.1
- User with **cluster-admin** privileges
- The OpenShift Container Platform Command-line Interface (CLI), commonly known as **oc**

Procedure

1. Create the **kubevirt-hyperconverged** namespace by running the following command:

```
$ oc new-project kubevirt-hyperconverged
```

2. Create the **OperatorGroup** and **CatalogSource** in the **kubevirt-hyperconverged** namespace by running the following command:

```
cat <<EOF | oc apply -f -
```

```
apiVersion: operators.coreos.com/v1alpha2
kind: OperatorGroup
metadata:
  name: hco-operatorgroup
  namespace: kubevirt-hyperconverged
---
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: hco-catalogsource
  namespace: openshift-operator-lifecycle-manager
  imagePullPolicy: Always
spec:
  sourceType: grpc
  image: registry.redhat.io/container-native-virtualization/hco-bundle-registry:v2.0.0
  displayName: KubeVirt HyperConverged
  publisher: Red Hat
EOF
```

1.3.2. Subscribing to the KubeVirt HyperConverged Cluster Operator catalog

Before you install container-native virtualization, subscribe to the **KubeVirt HyperConverged Cluster Operator** catalog from the OpenShift Container Platform web console. Subscribing gives the **kubevirt-hyperconverged** namespace access to the container-native virtualization Operators.

Prerequisites

- **OperatorGroup** and **CatalogSource** Custom Resource Definition objects (CRDs), both created in the **kubevirt-hyperconverged** namespace

Procedure

1. Open a browser window and navigate to the OpenShift Container Platform web console.
2. Select the **kubevirt-hyperconverged** project from the **Projects** list.
3. Navigate to the **Catalog → Operator Management** page.
4. In the **Operator Catalogs** tab, locate **KubeVirt HyperConverged Cluster Operator** and click **Create Subscription**.
5. Click **Create** to launch the container-native virtualization Operators.

1.3.3. Deploying container-native virtualization

After subscribing to the **KubeVirt HyperConverged Cluster Operator** catalog, create the **KubeVirt HyperConverged Cluster Operator Deployment** custom resource to deploy container-native virtualization.

Prerequisites

- An active subscription to the **KubeVirt HyperConverged Cluster Operator** catalog in the **kubevirt-hyperconverged** namespace

Procedure

1. Navigate to the **Catalog** → **Installed Operators** page.
2. Click **KubeVirt HyperConverged Cluster Operator**.
3. Click the **KubeVirt HyperConverged Cluster Operator Deployment** tab and click **Create HyperConverged**.
4. Click **Create** to launch container-native virtualization.
5. Navigate to the **Workloads** → **Pods** page and monitor the container-native virtualization Pods until they are all **Running**. After all the Pods display the **Running** state, you can access container-native virtualization.



NOTE

You can verify the installation by navigating to the web console at **kubevirt-web-ui.your.app.subdomain.host.com**. Log in by using your OpenShift Container Platform credentials.

1.4. INSTALLING THE VIRTCTL CLIENT

The **virtctl** client is a command-line utility for managing container-native virtualization resources.

Install the client to your system by enabling the container-native virtualization repository and installing the **kubevirt-virtctl** package.

1.4.1. Enabling container-native virtualization repositories

Red Hat offers container-native virtualization repositories for both Red Hat Enterprise Linux 8 and Red Hat Enterprise Linux 7:

- Red Hat Enterprise Linux 8 repository: **cnv-2.0-for-rhel-8-x86_64-rpms**
- Red Hat Enterprise Linux 7 repository: **rhel-7-server-cnv-2.0-rpms**

The process for enabling the repository in **subscription-manager** is the same in both platforms.

Procedure

- Use **subscription manager** to enable the appropriate container-native virtualization repository for your system:

```
# subscription-manager repos --enable <repository>
```

1.4.2. Installing the virtctl client

Install the **virtctl** client from the **kubevirt-virtctl** package.

Procedure

- Install the **kubevirt-virtctl** package:

```
# yum install kubevirt-virtctl
```

See also: [Using the CLI tools](#) for container-native virtualization.

1.5. UNINSTALLING CONTAINER-NATIVE VIRTUALIZATION

You can uninstall container-native virtualization by using the OpenShift Container Platform 4.1 [web console](#). First, delete the custom resource you created during deployment. Then, delete the **KubeVirt HyperConverged Cluster Operator** catalog subscription.

Prerequisites

- Container-native virtualization 2.0


1.5.1. Deleting the KubeVirt HyperConverged custom resource

To uninstall container-native virtualization, you must delete the custom resource that you created during deployment.

Prerequisites

- An active **KubeVirt HyperConverged Cluster Operator Deployment** custom resource

Procedure

1. From the OpenShift Container Platform web console, select **kubevirt-hyperconverged** from the **Projects** list.
2. Navigate to the **Catalog → Installed Operators** page.
3. Click **KubeVirt HyperConverged Cluster Operator**.
4. Click the **KubeVirt HyperConverged Cluster Operator Deployment** tab.
5. Click the Options menu  in the row containing the **kubevirt-hyperconverged** custom resource. In the expanded menu, click **Delete HyperConverged**.
6. Click **Delete** in the confirmation window.
7. Navigate to the **Workloads → Pods** page to verify that only the Operator Pods are running.
8. Open a terminal window and clean up the remaining KubeVirt resources by running the following command:

```
$ oc delete apiservices v1alpha3.subresources.kubevirt.io -n kubevirt-hyperconverged
```



NOTE

Because some KubeVirt resources are currently improperly retained, you must manually remove them. These resources will be removed automatically after [\(BZ1712429\)](#) is resolved.


1.5.2. Deleting the KubeVirt HyperConverged Cluster Operator catalog subscription

To finish uninstalling container-native virtualization, delete your KubeVirt HyperConverged Cluster Operator catalog subscription.

Prerequisites

- An active **KubeVirt HyperConverged Cluster Operator** catalog subscription

Procedure

1. From the OpenShift Container Platform web console, select **kubevirt-hyperconverged** from the **Projects** list.
2. Navigate to the **Catalog** → **Operator Management** page.
3. Click the **Operator Subscriptions** tab.
4. Click the Options menu  in the row that contains the **hco-subscription** subscription. In the expanded menu, click **Remove Subscription**.
5. Ensure that the **Also completely remove the Operator from the selected namespace** check box is selected. If it is not selected, Operator Pods will remain in **Workloads** → **Pods** when the subscription is removed.
6. Confirm that you want to remove the subscription by clicking **Remove** again.



NOTE

You can now delete the **kubevirt-hyperconverged** namespace.

1.5.3. Deleting a project using the web console

Procedure

1. Navigate to **Home** → **Projects**.
2. Locate the project that you want to delete from the list of projects.
3. On the far right side of the project listing, select **Delete Project** from the menu. If you do not have permissions to delete the project, the **Delete Project** option is grayed out and the option is not clickable.

CHAPTER 2. CONTAINER-NATIVE VIRTUALIZATION USER'S GUIDE

2.1. CREATING VIRTUAL MACHINES

Use one of these procedures to create a virtual machine:

- Running the virtual machine wizard
- Pasting a pre-configured YAML file with the virtual machine wizard
- Using the CLI
- Importing a VMware virtual machine or template with the virtual machine wizard

2.1.1. Running the virtual machine wizard to create a virtual machine

The web console features an interactive wizard that guides you through **Basic Settings**, **Networking**, and **Storage** screens to simplify the process of creating virtual machines. All required fields are marked by a *. The wizard prevents you from moving to the next screen until the required fields have been completed.

NICs and storage disks can be created and attached to virtual machines after they have been created.

Bootable Disk

If either **URL** or **Container** are selected as the **Provision Source** in the **Basic Settings** screen, a **rootdisk** disk is created and attached to the virtual machine as the **Bootable Disk**. You can modify the **rootdisk** but you cannot remove it.



A **Bootable Disk** is not required for virtual machines provisioned from a **PXE** source if there are no disks attached to the virtual machine. If one or more disks are attached to the virtual machine, you must select one as the **Bootable Disk**.

Prerequisites

- When you create your virtual machine using the wizard, your virtual machine's storage medium must support Read-Write-Many (RWM) PVCs.

Procedure

1. Click **Workloads** → **Virtual Machines** from the side menu.
2. Click **Create Virtual Machine** and select **Create with Wizard**
3. Fill in all required **Basic Settings**. Selecting a **Template** automatically fills in these fields.
4. Click **Next** to progress to the **Networking** screen. A **nic0** NIC is attached by default.
 - a. (Optional) Click **Create NIC** to create additional NICs.
 - b. (Optional) You can remove any or all NICs by clicking the **:** button and selecting **Remove NIC**. A virtual machine does not need a NIC attached to be created. NICs can be created after the virtual machine has been created.

5. Click **Next** to progress to the **Storage** screen.
 - a. (Optional) Click **Create Disk** to create additional disks. These disks can be removed by clicking the  button and selecting **Remove Disk**.
 - b. (Optional) Click on a disk to modify available fields. Click the  button to save the update.
 - c. (Optional) Click **Attach Disk** to choose an available disk from the **Select Storage** drop-down list.
6. Click **Create Virtual Machine** \triangleright The **Results** screen displays the JSON configuration file for the virtual machine.

The virtual machine is listed in **Workloads** \rightarrow **Virtual Machines**.

Refer to the virtual machine wizard fields section when running the web console wizard.

2.1.1.1. Virtual machine wizard fields

Name	Parameter	Description
Name		The name can contain lower-case letters (a-z), numbers (0-9), and hyphens (-), up to a maximum of 253 characters. The first and last characters must be alphanumeric. The name must not contain upper-case letters, spaces, periods (.), or special characters.
Description		Optional description field.
Template		Template from which to create the virtual machine. Selecting a template will automatically complete other fields.
Provision Source	PXE	Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.
	URL	Provision virtual machine from an image available from an HTTP or S3 endpoint.
	Container	Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: kubevirt/cirros-registry-disk-demo .
	Cloned Disk	Provision source is a cloned disk.

Name	Parameter	Description
	Import	Import virtual machine from a supported provider.
Operating System		A list of operating systems available in the cluster. This is the primary operating system for the virtual machine. If you select Import as the Provider Source , the operating system is filled in automatically, based on the operating system of the VMware virtual machine being imported.
Flavor	small, medium, large, tiny, Custom	Presets that determine the amount of CPU and memory allocated to the virtual machine.
Workload Profile	desktop	A virtual machine configuration for use on a desktop.
	generic	A virtual machine configuration that balances performance and compatibility for a broad range of workloads.
	high performance	A virtual machine configuration that is optimized for high-performance loads.
Start virtual machine on creation		Select to automatically start the virtual machine upon creation.
Use cloud-init		Select to enable the cloud-init fields.

2.1.1.2. Cloud-init fields

Name	Description
Hostname	Sets a specific host name for the virtual machine.
Authenticated SSH Keys	The user's public key that is copied to <code>~/.ssh/authorized_keys</code> on the virtual machine.
Use custom script	Replaces other options with a field in which you paste a custom cloud-init script.

2.1.1.3. Networking fields

Name	Description
Create NIC	Create a new NIC for the virtual machine.
NIC NAME	Name for the NIC.
MAC ADDRESS	MAC address for the network interface. If a MAC address is not specified, an ephemeral address is generated for the session.
NETWORK CONFIGURATION	List of available NetworkAttachmentDefinition objects.
BINDING METHOD	List of available binding methods. For the default Pod network, masquerade is the only recommended binding method. For secondary networks, use the bridge binding method. The masquerade method is not supported for non-default networks.
PXE NIC	List of PXE-capable networks. Only visible if PXE has been selected as the Provision Source .

2.1.1.4. Storage fields

Name	Description
Create Disk	Create a new disk for the virtual machine.
Attach Disk	Select an existing disk, from a list of available PVCs, to attach to the virtual machine.
DISK NAME	Name of the disk. The name can contain lower-case letters (a-z), numbers (0-9), hyphens (-), and periods (.), up to a maximum of 253 characters. The first and last characters must be alphanumeric. The name must not contain upper-case letters, spaces, or special characters.
SIZE (GB)	Size, in GB, of the disk.
STORAGE CLASS	Name of the underlying StorageClass .
Bootable Disk	List of available disks from which the virtual machine will boot. This is locked to rootdisk if the Provision Source of the virtual machine is URL or Container .

2.1.2. Pasting in a pre-configured YAML file to create a virtual machine

Create a virtual machine by writing or pasting a YAML configuration file in the web console in the **Workloads → Virtual Machines** screen. A valid **example** virtual machine configuration is provided by default whenever you open the YAML edit screen.

If your YAML configuration is invalid when you click **Create**, an error message indicates the parameter in which the error occurs. Only one error is shown at a time.



NOTE

Navigating away from the YAML screen while editing cancels any changes to the configuration you have made.

Procedure

1. Click **Workloads → Virtual Machines** from the side menu.
2. Click **Create Virtual Machine** and select **Create from YAML**.
3. Write or paste your virtual machine configuration in the editable window.
 - a. Alternatively, use the **example** virtual machine provided by default in the YAML screen.
4. (Optional) Click **Download** to download the YAML configuration file in its present state.
5. Click **Create** to create the virtual machine.

The virtual machine is listed in **Workloads → Virtual Machines**.

2.1.3. Using the CLI to create a virtual machine

Procedure

The **spec** object of the VirtualMachine configuration file references the virtual machine settings, such as the number of cores and the amount of memory, the disk type, and the volumes to use.

1. Attach the virtual machine disk to the virtual machine by referencing the relevant PVC **claimName** as a volume.
2. To create a virtual machine with the OpenShift Container Platform client, run this command:

```
$ oc create -f <vm.yaml>
```

3. Since virtual machines are created in a **Stopped** state, run a virtual machine instance by starting it.



NOTE

A [ReplicaSet](#)'s purpose is often used to guarantee the availability of a specified number of identical Pods. ReplicaSet is not currently supported in container-native virtualization.

Table 2.1. Domain settings

Setting	Description
Cores	The number of cores inside the virtual machine. Must be a value greater than or equal to 1.
Memory	The amount of RAM that is allocated to the virtual machine by the node. Specify a value in M for Megabyte or Gi for Gigabyte.
Disks: name	The name of the volume that is referenced. Must match the name of a volume.

Table 2.2. Volume settings

Setting	Description
Name	The name of the volume, which must be a DNS label and unique within the virtual machine.
PersistentVolumeClaim	The PVC to attach to the virtual machine. The claimName of the PVC must be in the same project as the virtual machine.

Virtual machine storage volume types are listed here, as well as domain and volume settings. See the [kubevirt API Reference](#) for a definitive list of virtual machine settings.

2.1.4. Virtual machine storage volume types

ephemeral	A local copy-on-write (COW) image that uses a network volume as a read-only backing store. The backing volume must be a PersistentVolumeClaim . The ephemeral image is created when the virtual machine starts and stores all writes locally. The ephemeral image is discarded when the virtual machine is stopped, restarted, or deleted. The backing volume (PVC) is not mutated in any way.
persistentVolumeClaim	Attaches an available PV to a virtual machine. Attaching a PV allows for the virtual machine data to persist between sessions. Importing an existing virtual machine disk into a PVC by using CDI and attaching the PVC to a virtual machine instance is the recommended method for importing existing virtual machines into OpenShift Container Platform. There are some requirements for the disk to be used within a PVC.
dataVolume	DataVolumes build on the persistentVolumeClaim disk type by managing the process of preparing the virtual machine disk via an import, clone, or upload operation. VMs that use this volume type are guaranteed not to start until the volume is ready.
cloudInitNoCloud	Attaches a disk that contains the referenced cloud-init NoCloud data source, providing user data and metadata to the virtual machine. A cloud-init installation is required inside the virtual machine disk.

containerDisk	<p>References an image, such as a virtual machine disk, that is stored in the container image registry. The image is pulled from the registry and embedded in a volume when the virtual machine is created. A containerDisk volume is ephemeral. It is discarded when the virtual machine is stopped, restarted, or deleted.</p> <p>Container disks are not limited to a single virtual machine and are useful for creating large numbers of virtual machine clones that do not require persistent storage.</p> <p>Only RAW and QCOW2 formats are supported disk types for the container image registry. QCOW2 is recommended for reduced image size.</p>
emptyDisk	<p>Creates an additional sparse QCOW2 disk that is tied to the life-cycle of the virtual machine interface. The data survives guest-initiated reboots in the virtual machine but is discarded when the virtual machine stops or is restarted from the web console. The empty disk is used to store application dependencies and data that otherwise exceeds the limited temporary file system of an ephemeral disk.</p> <p>The disk capacity size must also be provided.</p>

2.2. TLS CERTIFICATES FOR DATAVOLUME IMPORTS

2.2.1. Adding TLS certificates for authenticating DataVolume imports

TLS certificates for registry or HTTPS endpoints must be added to a ConfigMap in order to import data from these sources. This ConfigMap must be present in the namespace of the destination DataVolume.

Create the ConfigMap by referencing the relative file path for the TLS certificate.

Procedure

1. Ensure you are in the correct namespace. The ConfigMap can only be referenced by DataVolumes if it is in the same namespace.

```
$ oc get ns
```

2. Create the ConfigMap:

```
$ oc create configmap <configmap-name> --from-file=</path/to/file/ca.pem>
```

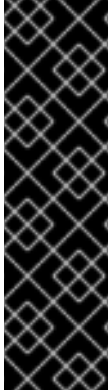
2.2.2. Example: ConfigMap created from a TLS certificate

The following example is of a ConfigMap created from **ca.pem** TLS certificate.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: tls-certs
data:
  ca.pem: |
    -----BEGIN CERTIFICATE-----
    ... <base64 encoded cert> ...
    -----END CERTIFICATE-----
```

2.3. IMPORTING A VMWARE VIRTUAL MACHINE OR TEMPLATE WITH THE VIRTUAL MACHINE WIZARD

You can import a VMware virtual machine or template. If you import a template, a virtual machine based on the template is created.



IMPORTANT

Importing a VMware virtual machine or template is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see <https://access.redhat.com/support/offerings/techpreview/>.

Prerequisites

- The VMware virtual machine is powered off.
- The VMware Virtual Disk Development Kit (VDDK) has been uploaded to the namespace in which you are importing the virtual machine.

2.3.1. Uploading the VMware Virtual Disk Development Kit

Procedure

1. In a browser, navigate to [VMware Documentation](#) and log in.
2. Click **VMware SDK & API Product Documentation** to expand.
3. Click **VMware Virtual Disk Development Kit (VDDK)**
4. Click **Latest Releases** and select the latest VDDK release.
5. Click **Download SDKs** to download the VDDK archive file.
6. Save the VDDK archive file in an HTTP-accessible location.
7. Switch to the namespace where you will import the virtual machine:

```
$ oc project <namespace>
```

8. Upload VDDK to the namespace:

```
$ cat <<EOF | oc apply -f -
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: "vddk-pvc"
spec:
```

```

source:
  http:
    url: "<http://www.example.com/VDDK_archive_file>" 1
  contentType: "archive"
  pvc:
    storageClassName: nfs-sc
    accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: "200Mi"
EOF

```

1 <url> is the path of the VDDK archive file.

2.3.2. Importing the VMware virtual machine or template

Procedure

1. Click **Workloads** → **Virtual Machines** from the side menu.
2. Click **Create Virtual Machine** and select **Create with Wizard**
3. Perform the following procedure in the **Basic Settings** screen:
 - a. Click **Provision Source** and select **Import**.
 - b. Click **Provider** and select **VMware**.
 - c. Click **vCenter Instance** and select **Connect to New Instance** or a saved vCenter instance.
 - If you select **Connect to New Instance**, provide values in the following fields and click **Check and Save**
 - **vCenter Hostname**
 - **vCenter User Name**
 - **vCenter Password**

The wizard connects to the vCenter instance and saves the credentials.

If you clear the **Save as New vCenter Instance** check box and click **Check**, the wizard connects to the vCenter instance without saving the credentials.
 - If you select a saved vCenter instance, the wizard connects to it by using the saved credentials.
 - d. Click **VM to import** and select a virtual machine or template.

The **Name** and **Operating System** are filled in automatically with the attributes of the selected virtual machine or template.
 - e. Specify the **Memory (GB)** and number of **CPUs** if you use the default flavor, **Custom**. Optionally, you can select an existing **Flavor**.
 - f. Select a **Workload Profile**.

- g. Click **Next**.
4. Perform the following procedure in the **Networking** screen:
 - a. Click **NETWORK CONFIGURATION** and select **Pod Networking**.
 - b. Select a **BINDING METHOD**.
 - c. Click the ✓ button to save the update.
 - d. Click **Next**.
5. Perform the following procedure in the **Storage** screen:
 - a. Click each disk and select a **STORAGE CLASS**. If you do not select a storage class, container-native virtualization uses the default storage class to create the virtual machine.
 - b. Click the ✓ button to save the update.
 - c. Select a **Bootable Disk** if you have more than one bootable disk.
 - d. Click **Create Virtual Machine >**
The **Result** screen displays the resources created for the virtual machine.
6. Click **Close**.

To view the progress of the import:

1. Click **Workloads** → **Pods** from the side menu.
2. Click the conversion pod, **kubevirt-v2v-conversion-*<id>***, to view the pod details.
3. In the **Overview** tab, click **ANNOTATIONS** for the **v2vConversionProgress** value.

If an error occurs, you can check the conversion pod log:

1. Click **Workloads** → **Pods** from the side menu.
2. Click the conversion pod, **kubevirt-v2v-conversion-*<id>***, to view the pod details.
3. Click **Logs** to view the conversion pod log.

Refer to the [virtual machine wizard fields](#) section for more information on the wizard fields.

2.3.3. Updating the imported virtual machine's NIC name

Update the imported virtual machine's NIC name to conform to container-native virtualization naming conventions.

Procedure

1. Log in to the imported virtual machine.
2. Go to the **/etc/sysconfig/network-scripts** directory.
3. Rename the network configuration file with the new NIC name (a single NIC is called **eth0**):

```
$ mv original_NIC ifcfg-ethx
```

- Update the following parameters in the network configuration file:

```
NAME=ethx
DEVICE=ethx
```

- Restart the network:

```
$ systemctl restart network
```

2.3.4. Virtual machine wizard fields

2.3.4.1. Virtual machine wizard fields

Name	Parameter	Description
Name		The name can contain lower-case letters (a-z), numbers (0-9), and hyphens (-), up to a maximum of 253 characters. The first and last characters must be alphanumeric. The name must not contain upper-case letters, spaces, periods (.), or special characters.
Description		Optional description field.
Template		Template from which to create the virtual machine. Selecting a template will automatically complete other fields.
Provision Source	PXE	Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.
	URL	Provision virtual machine from an image available from an HTTP or S3 endpoint.
	Container	Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: kubevirt/cirros-registry-disk-demo .
	Cloned Disk	Provision source is a cloned disk.

Name	Parameter	Description
	Import	Import virtual machine from a supported provider.
Operating System		A list of operating systems available in the cluster. This is the primary operating system for the virtual machine. If you select Import as the Provider Source , the operating system is filled in automatically, based on the operating system of the VMware virtual machine being imported.
Flavor	small, medium, large, tiny, Custom	Presets that determine the amount of CPU and memory allocated to the virtual machine.
Workload Profile	desktop	A virtual machine configuration for use on a desktop.
	generic	A virtual machine configuration that balances performance and compatibility for a broad range of workloads.
	high performance	A virtual machine configuration that is optimized for high-performance loads.
Start virtual machine on creation		Select to automatically start the virtual machine upon creation.
Use cloud-init		Select to enable the cloud-init fields.

2.3.4.2. Cloud-init fields

Name	Description
Hostname	Sets a specific host name for the virtual machine.
Authenticated SSH Keys	The user's public key that is copied to <code>~/.ssh/authorized_keys</code> on the virtual machine.

Name	Description
Use custom script	Replaces other options with a field in which you paste a custom cloud-init script.

2.3.4.3. Networking fields

Name	Description
Create NIC	Create a new NIC for the virtual machine.
NIC NAME	Name for the NIC.
MAC ADDRESS	MAC address for the network interface. If a MAC address is not specified, an ephemeral address is generated for the session.
NETWORK CONFIGURATION	List of available NetworkAttachmentDefinition objects.
BINDING METHOD	List of available binding methods. For the default Pod network, masquerade is the only recommended binding method. For secondary networks, use the bridge binding method. The masquerade method is not supported for non-default networks.
PXE NIC	List of PXE-capable networks. Only visible if PXE has been selected as the Provision Source .

2.3.4.4. Storage fields

Name	Description
Create Disk	Create a new disk for the virtual machine.
Attach Disk	Select an existing disk, from a list of available PVCs, to attach to the virtual machine.
DISK NAME	Name of the disk. The name can contain lower-case letters (a-z), numbers (0-9), hyphens (-), and periods (.), up to a maximum of 253 characters. The first and last characters must be alphanumeric. The name must not contain upper-case letters, spaces, or special characters.
SIZE (GB)	Size, in GB, of the disk.

Name	Description
STORAGE CLASS	Name of the underlying StorageClass .
Bootable Disk	List of available disks from which the virtual machine will boot. This is locked to rootdisk if the Provision Source of the virtual machine is URL or Container .

2.4. IMPORTING VIRTUAL MACHINE IMAGES WITH DATAVOLUMES

You can import an existing virtual machine image into your OpenShift Container Platform cluster. Container-native virtualization uses DataVolumes to automate the import of data and the creation of an underlying PersistentVolumeClaim (PVC).

CAUTION

When you import a disk image into a PVC, the disk image is expanded to use the full storage capacity that is requested in the PVC. To use this space, the disk partitions and file system(s) in the virtual machine might need to be expanded.

The resizing procedure varies based on the operating system installed on the VM. Refer to the operating system documentation for details.

Prerequisites

- If the endpoint requires a TLS certificate, the certificate must be [included in a ConfigMap](#) in the same namespace as the DataVolume and referenced in the DataVolume configuration.
- You may need to [define a StorageClass or prepare CDI scratch space](#) for this operation to complete successfully.

2.4.1. CDI supported operations matrix

This matrix shows the supported CDI operations for content types against endpoints, and which of these operations requires scratch space.

Content types	HTTP	HTTPS	HTTP basic auth	Registry	S3 Bucket	Upload
KubeVirt(QCOW2)	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2** ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* ✓ GZ* ✓ XZ*
Archive+	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR

✓ Supported operation

□ Unsupported operation

* Requires scratch space

** Requires scratch space if a custom certificate authority is required

+ Archive does not support block mode DVs

2.4.2. About DataVolumes

DataVolume objects are custom resources that are provided by the Containerized Data Importer (CDI) project. DataVolumes orchestrate import, clone, and upload operations that are associated with an underlying PersistentVolumeClaim (PVC). DataVolumes are integrated with KubeVirt, and they prevent a virtual machine from being started before the PVC has been prepared.

2.4.3. Importing a virtual machine image into a container-native virtualization object with DataVolumes

To create a virtual machine from an imported image, specify the image location in the **VirtualMachine** configuration file before you create the virtual machine.

Prerequisites

- Install the OpenShift Container Platform Command-line Interface (CLI), commonly known as **oc**
- A virtual machine disk image, in RAW, ISO, or QCOW2 format, optionally compressed by using **xz** or **gz**
- An **HTTP** endpoint where the image is hosted, along with any authentication credentials needed to access the data source
- At least one available PersistentVolume

Procedure

1. Identify an **HTTP** file server that hosts the virtual disk image that you want to import. You need the complete URL in the correct format:
 - <http://www.example.com/path/to/data>
2. If your data source requires authentication credentials, edit the **endpoint-secret.yaml** file, and apply the updated configuration to the cluster:

```
apiVersion: v1
kind: Secret
metadata:
  name: <endpoint-secret>
  labels:
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: "" 1
  secretKey: "" 2
```

- 1 Optional: your key or user name, base64 encoded
- 2 Optional: your secret or password, base64 encoded

```
$ oc apply -f endpoint-secret.yaml
```

3. Edit the virtual machine configuration file, specifying the data source for the image you want to import. In this example, a Fedora image is imported:

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  creationTimestamp: null
  labels:
    kubevirt.io/vm: vm-fedora-datavolume
  name: vm-fedora-datavolume
spec:
  dataVolumeTemplates:
  - metadata:
    creationTimestamp: null
    name: fedora-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 2Gi
        storageClassName: local
      source:
        http:
          url:
            https://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora
            -Cloud-Base-28-1.1.x86_64.qcow2 1
          secretRef: "" 2
          certConfigMap: "" 3
        status: {}
      running: false
    template:
      metadata:
        creationTimestamp: null
        labels:
          kubevirt.io/vm: vm-fedora-datavolume
      spec:
        domain:
          devices:
            disks:
            - disk:
                bus: virtio
                name: datavolumedisk1
          machine:
            type: ""
          resources:
            requests:
```

```

memory: 64M
terminationGracePeriodSeconds: 0
volumes:
- dataVolume:
  name: fedora-dv
  name: datavolumedisk1
status: {}

```

- 1 The **HTTP** source of the image you want to import.
- 2 The **secretRef** parameter is optional.
- 3 The **certConfigMap** is only required if the endpoint requires authentication. The referenced ConfigMap must be in the same namespace as the DataVolume.

4. Create the virtual machine:

```
$ oc create -f vm-<name>-datavolume.yaml
```



NOTE

The **oc create** command creates the DataVolume and the virtual machine. The CDI controller creates an underlying PVC with the correct annotation, and the import process begins. When the import completes, the DataVolume status changes to **Succeeded**, and the virtual machine is allowed to start.

DataVolume provisioning happens in the background, so there is no need to monitor it. You can start the virtual machine, and it will not run until the import is complete.

Optional verification steps

1. Run **oc get pods** and look for the importer Pod. This Pod downloads the image from the specified URL and stores it on the provisioned PV.
2. Monitor the DataVolume status until it shows **Succeeded**.

```
$ oc describe dv <data-label> 1
```

- 1 The data label for the DataVolume specified in the virtual machine configuration file.
3. To verify that provisioning is complete and that the VMI has started, try accessing its serial console:

```
$ virtctl console <vm-fedora-datavolume>
```

2.4.4. Template: DataVolume virtual machine configuration file

example-dv-vm.yaml

```

apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine

```



```

metadata:
  labels:
    kubevirt.io/vm: example-vm
  name: example-vm
spec:
  dataVolumeTemplates:
  - metadata:
    name: example-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 1G
        source:
          http:
            url: "" 1
  running: false
template:
  metadata:
    labels:
      kubevirt.io/vm: example-vm
  spec:
    domain:
      cpu:
        cores: 1
    devices:
      disks:
      - disk:
        bus: virtio
        name: example-dv-disk
    machine:
      type: q35
    resources:
      requests:
        memory: 1G
    terminationGracePeriodSeconds: 0
  volumes:
  - dataVolume:
    name: example-dv
    name: example-dv-disk

```

1 The **HTTP** source of the image you want to import, if applicable.

2.4.5. Template: DataVolume import configuration file

example-import-dv.yaml

```

apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: "example-import-dv"
spec:

```

```

source:
  http:
    url: "" 1
    secretRef: "" 2
pvc:
  accessModes:
  - ReadWriteOnce
  resources:
  requests:
  storage: "1G"

```

- 1 The **HTTP** source of the image you want to import.
- 2 The **secretRef** parameter is optional.

2.5. EDITING VIRTUAL MACHINES

Edit a virtual machine by completing one of the following tasks:

- Using the web console
- Editing the virtual machine YAML configuration
- Using the CLI

2.5.1. Using the web console to edit a virtual machine

Edit select values of a virtual machine in the **Virtual Machine Overview** screen of the web console. Other values can be edited using the CLI. Changes do not display until you reboot the virtual machine.

Prerequisites

- When editing from the **Virtual Machine Overview** screen, the virtual machine must be **Off**.

Procedure

1. Click **Workloads** → **Virtual Machines** from the side menu.
2. Select a Virtual Machine.
3. Click **Edit** to make editable fields available.
4. You can change the **Flavor** but only to **Custom**, which provides additional fields for **CPU** and **Memory**.
5. Click **Save**.
The values and virtual machine are updated after the operation is processed.

If the virtual machine is running, changes will not display until the virtual machine is rebooted.

2.5.2. Editing the virtual machine YAML configuration

Edit the YAML configuration of a virtual machine the web console.

Not all parameters can be updated. If you edit values that cannot be changed and click **Save**, an error message indicates the parameter that was not able to be updated.

The YAML configuration can be edited while the virtual machine is **Running**, however the changes will only take effect after the virtual machine has been stopped and started again.



NOTE

Navigating away from the YAML screen while editing cancels any changes to the configuration you have made.

Procedure

1. Click **Workloads** → **Virtual Machine** from the side menu.
2. Select a virtual machine.
3. Click the **YAML** tab to display the editable configuration.
4. Optional: You can click **Download** to download the YAML file locally in its current state.
5. Edit the file and click **Save**.

A confirmation message shows that the modification has been successful and includes the updated version number for the object.

2.5.3. Using the CLI to edit a virtual machine

Prerequisites

- You configured your virtual machine with a YAML object configuration file.
- You installed the oc CLI.

Procedure

1. Run the following command to update the virtual machine configuration.

```
oc edit
```

2. Open the object configuration.
3. Edit the YAML.
4. If you edit a running virtual machine, you need to do one of the following:
 - Restart the virtual machine
 - Run the following command for the new configuration to take effect.

```
oc apply
```

2.6. DELETING VIRTUAL MACHINES

Use one of these procedures to delete a virtual machine:

- Using the web console
- Using the CLI

2.6.1. Deleting a virtual machine using the web console

Deleting a virtual machine permanently removes it from the cluster.

Delete a virtual machine using the **:** button of the virtual machine in the **Workloads → Virtual Machines** list, or using the **Actions** control of the **Virtual Machine Details** screen.

Procedure

1. In the container-native virtualization console, click **Workloads → Virtual Machines** from the side menu.
2. Click the **:** button of the virtual machine to delete and select **Delete Virtual Machine**
 - Alternatively, click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions → Delete Virtual Machine**
3. In the confirmation pop-up window, click **Delete** to permanently delete the virtual machine.

2.6.2. Deleting a virtual machine and PVCs using the CLI

When you delete a virtual machine, the PVC it uses is unbound.

If you do not plan to bind this PVC to a different VM, it is best practice to delete it, in order to maintain a clean environment and avoid possible confusion.

Procedure

Run these commands to delete the virtual machine and the PVC.



NOTE

You can delete objects only in the project you are currently working in, unless you specify the **-n <project_name>** option, for project name.

1. Run the following command to delete the virtual machine:

```
$ oc delete vm <fedora-vm>
```

2. Run the following command to delete the PVC associated with the virtual machine.

```
$ oc delete pvc <fedora-vm-pvc>
```

2.7. CONTROLLING VIRTUAL MACHINES STATES

With container-native virtualization, you can stop, start, and restart virtual machines from both the web console and the command-line interface (CLI).

2.7.1. Controlling virtual machines from the web console


You can also stop, start, and restart virtual machines from the web console.

2.7.1.1. Starting a virtual machine

You can start a virtual machine from the web console.

Procedure

1. In the container-native virtualization console, click **Workloads → Virtual Machines**.
2. Start the virtual machine from this screen, which makes it easier to perform actions on multiple virtual machines in the one screen, or from the **Virtual Machine Details** screen where you can view comprehensive details of the selected virtual machine:

- Click the Options menu  at the end of virtual machine and select **Start Virtual Machine**.
- Click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** and select **Start Virtual Machine**

3. In the confirmation window, click **Start** to start the virtual machine.

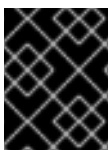


NOTE

When you start virtual machine that is provisioned from a **URL** source for the first time, the virtual machine is in the **Importing** state while container-native virtualization imports the container from the URL endpoint. Depending on the size of the image, this process might take several minutes.

2.7.1.2. Restarting a virtual machine

You can restart a running virtual machine from the web console.




IMPORTANT

Do not restart a virtual machine while it has a status of **Importing**. Restarting the virtual machine causes an error for it.

Procedure

1. In the container-native virtualization console, click **Workloads → Virtual Machines**.
2. Restart the virtual machine from this screen, which makes it easier to perform actions on multiple virtual machines in the one screen, or from the **Virtual Machine Details** screen where you can view comprehensive details of the selected virtual machine:

- Click the Options menu  at the end of virtual machine and select **Restart Virtual Machine**.


- Click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** and select **Restart Virtual Machine**
3. In the confirmation window, click **Restart** to restart the virtual machine.

2.7.1.3. Stopping a virtual machine

You can stop a virtual machine from the web console.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. Stop the virtual machine from this screen, which makes it easier to perform actions on multiple virtual machines in the one screen, or from the **Virtual Machine Details** screen where you can view comprehensive details of the selected virtual machine:

- Click the Options menu  at the end of virtual machine and select **Stop Virtual Machine**.
- Click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** and select **Stop Virtual Machine**

3. In the confirmation window, click **Stop** to stop the virtual machine.

2.7.2. CLI reference for controlling virtual machines

Use the following **virtctl** client utility and **oc** commands to change the state of the virtual machines and display lists of the virtual machines and the virtual machine instances that represent them.



NOTE

When you run **virtctl** commands, you modify the virtual machines themselves, not the virtual machine instances that represent them in the web console.

2.7.2.1. start

Start a virtual machine.

Example: Start a virtual machine in the current project

```
$ virtctl start <example-vm>
```

Example: Start a virtual machine in a specific project

```
$ virtctl start <example-vm> -n <project_name>
```

2.7.2.2. restart

Restart a running virtual machine.

Example: Restart a virtual machine in the current project

```
$ virtctl restart <example-vm>
```

Example: Restart a virtual machine in a specific project

```
$ virtctl restart <example-vm> -n <project_name>
```

2.7.2.3. stop

Stop a running virtual machine.

Example: Stop a virtual machine in the current project

```
$ virtctl stop <example-vm>
```

Example: Stop a virtual machine in a specific project

```
$ virtctl stop <example-vm> -n <project_name>
```

2.7.2.4. list

List the virtual machines or virtual machine instances in a project. The virtual machine instances are abstractions that represent the virtual machines themselves.

Example: List the virtual machines in the current project

```
$ oc get vm
```

Example: List the virtual machines in a specific project

```
$ oc get vm -n <project_name>
```

Example: List the running virtual machine instances in the current project

```
$ oc get vmi
```

Example: List the running virtual machine instances in a specific project

```
$ oc get vmi -n <project_name>
```

2.8. ACCESSING VIRTUAL MACHINE CONSOLES

Container-native virtualization provides different virtual machine consoles that you can use to accomplish different product tasks. You can access these consoles through the web console and by using CLI commands.

2.8.1. Virtual machine console sessions

You can connect to the VNC and serial consoles of a running virtual machine from the **Consoles** tab in the **Virtual Machine Details** screen of the web console.

There are two consoles available: the graphical **VNC Console** and the **Serial Console**. The **VNC Console** opens by default whenever you navigate to the **Consoles** tab. You can switch between the consoles using the **VNC Console Serial Console** list.

Console sessions remain active in the background unless they are disconnected. When the **Disconnect before switching** checkbox is active and you switch consoles, the current console session is disconnected and a new session with the selected console connects to the virtual machine. This ensures only one console session is open at a time.

Options for the VNC Console

The **Send Key** button lists key combinations to send to the virtual machine.

Options for the Serial Console

Use the **Disconnect** button to manually disconnect the **Serial Console** session from the virtual machine. Use the **Reconnect** button to manually open a **Serial Console** session to the virtual machine.

2.8.1.1. Connecting to a virtual machine with the web console

You can connect to a virtual machine by using the web console.

Procedure

1. Ensure you are in the correct project. If not, click the **Project** list and select the appropriate project.
2. Click **Applications** → **Virtual Machines** to display the virtual machines in the project.
3. Select a virtual machine.
4. In the **Overview** tab, click the **virt-launcher-`<vm-name>`** pod.
5. Click the **Terminal** tab. If the terminal is blank, click the terminal and press any key to initiate connection.

2.8.1.2. Connecting to the serial console

Connect to the **Serial Console** of a running virtual machine from the **Consoles** tab in the **Virtual Machine Details** screen of the web console.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. Select a virtual machine.
3. Click **Consoles**. The VNC console opens by default.
4. Click the **VNC Console** drop-down list and select **Serial Console**.

2.8.1.3. Connecting to the VNC console

Connect to the VNC console of a running virtual machine from the **Consoles** tab in the **Virtual Machine Details** screen of the web console.

Procedure

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. Select a virtual machine.
3. Click **Consoles**. The VNC console opens by default.

2.8.1.4. Connecting to the RDP console

The desktop viewer console, which utilizes the Remote Desktop Protocol (RDP), provides a better console experience for connecting to Windows virtual machines.

To connect to a Windows virtual machine with RDP, download the **console.rdp** file for the virtual machine from the **Consoles** tab in the **Virtual Machine Details** screen of the web console and supply it to your preferred RDP client.

Prerequisites

- A running Windows virtual machine with the QEMU guest agent installed. The **qemu-guest-agent** is included in the VirtIO drivers.
- A layer 2 vNIC attached to the virtual machine.
- An RDP client installed on a machine on the same network as the Windows virtual machine.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. Select a Windows virtual machine.
3. Click the **Consoles** tab.
4. Click the **Consoles** list and select **Desktop Viewer**.
5. In the **Network Interface** list, select the layer 2 vNIC.
6. Click **Launch Remote Desktop** to download the **console.rdp** file.
7. Open an RDP client and reference the **console.rdp** file. For example, using **remmina**:

```
$ remmina --connect /path/to/console.rdp
```

8. Enter the **Administrator** user name and password to connect to the Windows virtual machine.

2.8.2. Accessing virtual machine consoles by using CLI commands

2.8.2.1. Accessing a virtual machine instance via SSH

You can use SSH to access a virtual machine after you expose port 22 on it.

The **virtctl expose** command forwards a virtual machine instance port to a node port and creates a service for enabled access. The following example creates the **fedora-vm-ssh** service that forwards port 22 of the **<fedora-vm>** virtual machine to a port on the node:

Prerequisites

- The virtual machine instance you want to access must be running.
- Install the OpenShift Command-line Interface (CLI), commonly known as **oc**.

Procedure

1. Run the following command to create the **fedora-vm-ssh** service:

```
$ virtctl expose vm <fedora-vm> --port=20022 --target-port=22 --name=fedora-vm-ssh --
type=NodePort 1
```

- 1** **<fedora-vm>** is the name of the virtual machine that you run the **fedora-vm-ssh** service on.

2. Check the service to find out which port the service acquired:

```
$ oc get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
fedora-vm-ssh NodePort      127.0.0.1     <none>       20022:32551/TCP 6s
```

In this example, the service acquired the **32551** port.

3. Log in to the virtual machine instance via SSH. Use the **ipAddress** of the node and the port that you found in the previous step:

```
$ ssh username@<node_IP_address> -p 32551
```

2.8.2.2. Accessing the serial console of a virtual machine instance

The **virtctl console** command opens a serial console to the specified virtual machine instance.

Prerequisites

- The **virt-viewer** package must be installed.
- The virtual machine instance you want to access must be running.

Procedure

- Connect to the serial console with **virtctl**:

```
$ virtctl console <VMI>
```

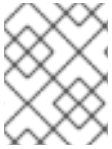
2.8.2.3. Accessing the graphical console of a virtual machine instances with VNC

The **virtctl** client utility can use the **remote-viewer** function to open a graphical console to a running virtual machine instance. This capability is included in the **virt-viewer** package.

Prerequisites

- The **virt-viewer** package must be installed.

- The virtual machine instance you want to access must be running.



NOTE

If you use **virtctl** via SSH on a remote machine, you must forward the X session to your machine.

Procedure

1. Connect to the graphical interface with the **virtctl** utility:

```
$ virtctl vnc <VMI>
```

2. If the command failed, try using the **-v** flag to collect troubleshooting information:

```
$ virtctl vnc <VMI> -v 4
```

2.8.2.4. Connecting to a Windows virtual machine with an RDP console

The Remote Desktop Protocol (RDP) provides a better console experience for connecting to Windows virtual machines.

To connect to a Windows virtual machine with RDP, specify the IP address of the attached L2 vNIC to your RDP client.

Prerequisites

- A running Windows virtual machine with the QEMU guest agent installed. The **qemu-guest-agent** is included in the VirtIO drivers.
- A layer 2 vNIC attached to the virtual machine.
- An RDP client installed on a machine on the same network as the Windows virtual machine.

Procedure

1. Log in to the container-native virtualization cluster through the **oc** CLI tool as a user with an access token.

```
$ oc login -u <user> https://<cluster.example.com>:8443
```

2. Use **oc describe vmi** to display the configuration of the running Windows virtual machine.

```
$ oc describe vmi <windows-vmi-name>
```

```
...
spec:
  networks:
  - name: default
  pod: {}
  - multus:
    networkName: cnv-bridge
    name: bridge-net
```

```

...
status:
  interfaces:
  - interfaceName: eth0
    ipAddress: 198.51.100.0/24
    ipAddresses:
      198.51.100.0/24
    mac: a0:36:9f:0f:b1:70
    name: default
  - interfaceName: eth1
    ipAddress: 192.0.2.0/24
    ipAddresses:
      192.0.2.0/24
      2001:db8::/32
    mac: 00:17:a4:77:77:25
    name: bridge-net
...

```

3. Identify and copy the IP address of the layer 2 network interface. This is **192.0.2.0** in the above example, or **2001:db8::** if you prefer IPv6.
4. Open an RDP client and use the IP address copied in the previous step for the connection.
5. Enter the **Administrator** user name and password to connect to the Windows virtual machine.

2.9. USING THE CLI TOOLS

The two primary CLI tools used for managing resources in the cluster are:

- The container-native virtualization **virtctl** client
- The OpenShift Container Platform **oc** client

Prerequisites

- You must [install the virtctl client](#).

2.9.1. Virtctl client commands

The **virtctl** client is a command-line utility for managing container-native virtualization resources. The following table contains the **virtctl** commands used throughout the container-native virtualization documentation.

Table 2.3. virtctl client commands

Command	Description
virtctl start <vm>	Start a virtual machine.
virtctl stop <vm>	Stop a virtual machine.
virtctl restart <vm>	Restart a virtual machine.

Command	Description
virtctl expose <vm>	Create a service that forwards a designated port of a virtual machine or virtual machine instance and expose the service on the specified port of the node.
virtctl console <vmi>	Connect to a serial console of a virtual machine instance.
virtctl vnc <vmi>	Open a VNC connection to a virtual machine instance.
virtctl image-upload <...>	Upload a virtual machine image to a PersistentVolumeClaim.

2.9.2. OpenShift Container Platform client commands

The OpenShift Container Platform **oc** client is a command-line utility for managing OpenShift Container Platform resources. The following table contains the **oc** commands used throughout the container-native virtualization documentation.

Table 2.4. **oc** commands

Command	Description
oc login -u <user_name>	Log in to the OpenShift Container Platform cluster as <user_name> .
oc get <object_type>	Display a list of objects for the specified object type in the project.
oc describe <object_type> <resource_name>	Display details of the specific resource in the project.
oc create -f <object_config>	Create a resource in the project from a filename or from stdin.
oc edit <object_type> <resource_name>	Edit a resource in the project.
oc delete <object_type> <resource_name>	Delete a resource in the project.

For more comprehensive information on **oc** client commands, see the [OpenShift Container Platform CLI reference](#).

2.10. AUTOMATING MANAGEMENT TASKS

You can automate {ProductName} management tasks by using Red Hat Ansible Automation. Learn the basics by using an Ansible Playbook to create a new virtual machine.

2.10.1. About Red Hat Ansible Automation

[Ansible](#) is an automation tool used to configure systems, deploy software, and perform rolling updates. Ansible includes support for {ProductName}, and Ansible modules enable you to automate cluster management tasks such as template, persistent volume claim, and virtual machine operations.

Ansible provides a way to automate {ProductName} management, which you can also accomplish by using the **oc** CLI tool or APIs. Ansible is unique because it allows you to integrate [KubeVirt modules](#) with other Ansible modules.

2.10.2. Automating virtual machine creation with Red Hat Ansible Automation

You can use the **kubvirt_vm** Ansible Playbook to create virtual machines in your OpenShift Container Platform cluster.

Prerequisites

- [Red Hat Ansible Engine](#) version 2.8 or newer

Procedure

1. Edit an Ansible Playbook YAML file so that it includes the **kubvirt_vm** task:

```
kubvirt_vm:
  namespace:
  name:
  cpu_cores:
  memory:
  disks:
  - name:
    volume:
      containerDisk:
        image:
  disk:
  bus:
```



NOTE

This snippet only includes the **kubvirt_vm** portion of the playbook.

2. Edit the values to reflect the virtual machine you want to create, including the **namespace**, the number of **cpu_cores**, the **memory**, and the **disks**. For example:

```
kubvirt_vm:
  namespace: default
  name: vm1
  cpu_cores: 1
  memory: 64Mi
  disks:
  - name: containerdisk
    volume:
      containerDisk:
        image: kubvirt/cirros-container-disk-demo:latest
  disk:
  bus: virtio
```

- If you want the virtual machine to boot immediately after creation, add **state: running** to the YAML file. For example:

```
kubvirt_vm:
  namespace: default
  name: vm1
  state: running ❶
  cpu_cores: 1
```

- Changing this value to **state: absent** deletes the virtual machine, if it already exists.

- Run the **ansible-playbook** command, using your playbook's file name as the only argument:

```
$ ansible-playbook create-vm.yaml
```

- Review the output to determine if the play was successful:

```
(...)
TASK [Create my first VM] *****
changed: [localhost]

PLAY RECAP
*****
localhost      : ok=2  changed=1  unreachable=0  failed=0  skipped=0
rescued=0  ignored=0
```

- If you did not include **state: running** in your playbook file and you want to boot the VM now, edit the file so that it includes **state: running** and run the playbook again:

```
$ ansible-playbook create-vm.yaml
```

To verify that the virtual machine was created, try to [access the VM console](#).

2.10.3. Example: Ansible Playbook for creating virtual machines

You can use the **kubvirt_vm** Ansible Playbook to automate virtual machine creation.

The following YAML file is an example of the **kubvirt_vm** playbook. It includes sample values that you must replace with your own information if you run the playbook.

```
---
- name: Ansible Playbook 1
  hosts: localhost
  connection: local
  tasks:
    - name: Create my first VM
      kubvirt_vm:
        namespace: default
        name: vm1
        cpu_cores: 1
        memory: 64Mi
        disks:
          - name: containerdisk
```

```

volume:
  containerDisk:
    image: kubevirt/cirros-container-disk-demo:latest
disk:
  bus: virtio

```

Additional information

- [Intro to Playbooks](#)
- [Tools for Validating Playbooks](#)

2.11. USING THE DEFAULT POD NETWORK WITH CONTAINER-NATIVE VIRTUALIZATION

You can use the default Pod network with container-native virtualization. To do so, you must use the **masquerade** binding method. It is the only recommended binding method for use with the default Pod network. Do not use **masquerade** mode with non-default networks.



NOTE

For secondary networks, use the **bridge** binding method.

2.11.1. Configuring masquerade mode from the command line

You can use masquerade mode to hide a virtual machine's outgoing traffic behind the Pod IP address. Masquerade mode uses Network Address Translation (NAT) to connect virtual machines to the Pod network backend through a Linux bridge.

Enable masquerade mode and allow traffic to enter the virtual machine by editing your virtual machine configuration file.

Prerequisites

- The virtual machine must be configured to use DHCP to acquire IPv4 addresses. The examples below are configured to use DHCP.

Procedure

1. Edit the **interfaces** spec of your virtual machine configuration file:

```

kind: VirtualMachine
spec:
  domain:
    devices:
      interfaces:
        - name: red
          masquerade: {} 1
        ports:
          - port: 80 2
      networks:
        - name: red
          pod: {}

```


- 1 Connect using masquerade mode
- 2 Allow incoming traffic on port 80

2. Create the virtual machine:

```
$ oc create -f <vm-name>.yaml
```

2.11.2. Web console

If you create a virtual machine from the container-native virtualization [web console wizard](#), select the required binding method from the **Networking** screen.

2.11.2.1. Networking fields

Name	Description
Create NIC	Create a new NIC for the virtual machine.
NIC NAME	Name for the NIC.
MAC ADDRESS	MAC address for the network interface. If a MAC address is not specified, an ephemeral address is generated for the session.
NETWORK CONFIGURATION	List of available NetworkAttachmentDefinition objects.
BINDING METHOD	List of available binding methods. For the default Pod network, masquerade is the only recommended binding method. For secondary networks, use the bridge binding method. The masquerade method is not supported for non-default networks.
PXE NIC	List of PXE-capable networks. Only visible if PXE has been selected as the Provision Source .

2.11.3. Configuration file examples

2.11.3.1. Template: virtual machine configuration file

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: example-vm
  namespace: default
spec:
  running: false
```

```
template:
spec:
  domain:
  devices:
    disks:
      - name: containerdisk
        disk:
          bus: virtio
      - name: cloudinitdisk
        disk:
          bus: virtio
    interfaces:
      - masquerade: {}
        name: default
  resources:
    requests:
      memory: 1024M
  networks:
    - name: default
      pod: {}
  volumes:
    - name: containerdisk
      containerDisk:
        image: kubevirt/fedora-cloud-container-disk-demo
    - name: cloudinitdisk
      cloudInitNoCloud:
        userData: |
          #!/bin/bash
          echo "fedora" | passwd fedora --stdin
```

2.11.3.2. Template: Windows virtual machine instance configuration file

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  labels:
    special: vmi-windows
  name: vmi-windows
spec:
  domain:
    clock:
      timer:
        hpet:
          present: false
        hyperv: {}
        pit:
          tickPolicy: delay
        rtc:
          tickPolicy: catchup
        utc: {}
    cpu:
      cores: 2
  devices:
    disks:
      - disk:
```

```

    bus: sata
    name: pvcdisk
  interfaces:
  - masquerade: {}
    model: e1000
    name: default
  features:
    acpi: {}
    apic: {}
    hyperv:
      relaxed: {}
    spinlocks:
      spinlocks: 8191
    vapic: {}
  firmware:
    uuid: 5d307ca9-b3ef-428c-8861-06e72d69f223
  machine:
    type: q35
  resources:
    requests:
      memory: 2Gi
  networks:
  - name: default
    pod: {}
  terminationGracePeriodSeconds: 0
  volumes:
  - name: pvcdisk
    persistentVolumeClaim:
      claimName: disk-windows

```

2.12. ATTACHING A VIRTUAL MACHINE TO MULTIPLE NETWORKS

Container-native virtualization provides Layer-2 networking capabilities that allow you to connect virtual machines to multiple networks. You can import virtual machines with existing workloads that depend on access to multiple interfaces. You can also configure a PXE network so that you can boot machines over the network.

To get started, a network administrator configures a NetworkAttachmentDefinition of type **cnv-bridge**. Then, users can attach Pods, virtual machine instances, and virtual machines to the bridge network. From the container-native virtualization web console, you can create a vNIC that refers to the bridge network.

2.12.1. Container-native virtualization networking glossary

Container-native virtualization provides advanced networking functionality by using custom resources and plug-ins.

The following terms are used throughout container-native virtualization documentation:

Container Network Interface (CNI)

a [Cloud Native Computing Foundation](#) project, focused on container network connectivity. Container-native virtualization uses CNI plug-ins to build upon the basic Kubernetes networking functionality.

Multus

a "meta" CNI plug-in that allows multiple CNIs to exist so that a Pod or virtual machine can use the interfaces it needs.

Custom Resource Definition (CRD)

a [Kubernetes](#) API resource that allows you to define custom resources, or an object defined by using the CRD API resource.

NetworkAttachmentDefinition

a CRD introduced by the Multus project that allows you to attach Pods, virtual machines, and virtual machine instances to one or more networks.

Preboot eXecution Environment (PXE)

an interface that enables an administrator to boot a client machine from a server over the network. Network booting allows you to remotely load operating systems and other software onto the client.

2.12.2. Connecting a resource to a bridge-based network

As a network administrator, you can configure a NetworkAttachmentDefinition of type **cnv-bridge** to provide Layer-2 networking to Pods and virtual machines.

Prerequisites

- Container-native virtualization 2.0 or newer
- A Linux bridge must be configured and attached to the correct Network Interface Card on every node.
- If you use VLANs, **vlan_filtering** must be enabled on the bridge.
- The NIC must be tagged to all relevant VLANs.
 - For example: **bridge vlan add dev bond0 vid 1-4095 master**

Procedure

1. Create a new file for the NetworkAttachmentDefinition in any local directory. The file must have the following contents, modified to match your configuration:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: a-bridge-network
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/br0 1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "a-bridge-network", 2
    "type": "cnv-bridge", 3
    "bridge": "br0", 4
    "isGateway": true,
    "ipam": {}
  }'
```

- 1** If you add this annotation to your NetworkAttachmentDefinition, your virtual machine instances will only run on nodes that have the **br0** bridge connected.

- 2 The **name** value is part of the annotation you will use in the next step.
- 3 The actual name of the Container Network Interface (CNI) plug-in that provides the network for this NetworkAttachmentDefinition. Do not change this field unless you want to use a different CNI.
- 4 You must substitute the actual name of the bridge, if it is not **br0**.

```
$ oc create -f <resource_spec.yaml>
```

2. Edit the configuration file of a virtual machine or virtual machine instance that you want to connect to the bridge network:

```
apiVersion: v1
kind: VirtualMachine
metadata:
  name: example-vm
  annotations:
    k8s.v1.cni.cncf.io/networks: a-bridge-network 1
spec:
  ...
```

- 1 You must substitute the actual **name** value from the NetworkAttachmentDefinition.

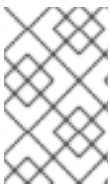
In this example, the NetworkAttachmentDefinition and Pod are in the same namespace.

To specify a different namespace, use the following syntax:

```
...
annotations:
  k8s.v1.cni.cncf.io/networks: <namespace>/a-bridge-network
...
```

3. Apply the configuration file to the resource:

```
$ oc create -f <local/path/to/network-attachment-definition.yaml>
```



NOTE

When defining the vNIC in the next section, ensure that the **NETWORK** value is the bridge network name from the NetworkAttachmentDefinition you created in the previous section.

2.12.3. Creating a NIC for a virtual machine

Create and attach additional NICs to a virtual machine from the web console.

Procedure

1. In the correct project in the container-native virtualization console, click **Workloads → Virtual Machines**.

2. Select a virtual machine template.
3. Click **Network Interfaces** to display the NICs already attached to the virtual machine.
4. Click **Create NIC** to create a new slot in the list.
5. Fill in the **NAME**, **NETWORK**, **MAC ADDRESS**, and **BINDING METHOD** for the new NIC.
6. Click the ✓ button to save and attach the NIC to the virtual machine.

2.12.4. Networking fields

Name	Description
Create NIC	Create a new NIC for the virtual machine.
NIC NAME	Name for the NIC.
MAC ADDRESS	MAC address for the network interface. If a MAC address is not specified, an ephemeral address is generated for the session.
NETWORK CONFIGURATION	List of available NetworkAttachmentDefinition objects.
BINDING METHOD	List of available binding methods. For the default Pod network, masquerade is the only recommended binding method. For secondary networks, use the bridge binding method. The masquerade method is not supported for non-default networks.
PXE NIC	List of PXE-capable networks. Only visible if PXE has been selected as the Provision Source .

Install the optional [QEMU guest agent](#) on the virtual machine so that the host can display relevant information about the additional networks.

2.13. INSTALLING THE QEMU GUEST AGENT ON VIRTUAL MACHINES

The QEMU guest agent is a daemon that runs on the virtual machine. The agent passes network information on the virtual machine, notably the IP address of additional networks, to the host.

2.13.1. Installing QEMU guest agent on a Linux virtual machine

The **qemu-guest-agent** is widely available and available by default in Red Hat virtual machines. Install the agent and start the service

Procedure

1. Access the virtual machine command line through one of the consoles or by SSH.

2. Install the QEMU guest agent on the virtual machine:

```
$ yum install -y qemu-guest-agent
```

3. Start the QEMU guest agent service:

```
$ systemctl start qemu-guest-agent
```

4. Ensure the service is persistent:

```
$ systemctl enable qemu-guest-agent
```

You can also install and start the QEMU guest agent using the **cloud-init**:*Use custom script* field of the wizard when creating either virtual machines or virtual machines templates in the web console.

For Windows virtual machines, the QEMU guest agent is included in the VirtIO drivers, which can be installed on [an existing Windows virtual machine](#) or [during the installation of Windows on the virtual machine](#).

2.14. VIEWING THE IP ADDRESS OF VNICS ON A VIRTUAL MACHINE

The QEMU guest agent runs on the virtual machine and passes the IP address of attached vNICs to the host, allowing you to view the IP address from both the web console and the **oc** client.

Prerequisites

- The QEMU guest agent must be [installed and running](#) on the virtual machine.

2.14.1. Viewing the IP address of a virtual machine interface in the CLI

The network interface configuration is included in the **oc describe vmi <vmi_name>** command.

You can also view the IP address information by running **ip addr** on the virtual machine, or by running **oc get vmi <vmi_name> -o yaml**.

Procedure

- Use the **oc describe** command to display the virtual machine interface configuration:

```
$ oc describe vmi <vmi_name>

...
Interfaces:
  Interface Name: eth0
  Ip Address:    10.244.0.37/24
  Ip Addresses:
    10.244.0.37/24
    fe80::858:aff:fef4:25/64
  Mac:          0a:58:0a:f4:00:25
  Name:         default
  Interface Name: v2
  Ip Address:   1.1.1.7/24
  Ip Addresses:
```

```
1.1.1.7/24
fe80::f4d9:70ff:fe13:9089/64
Mac:      f6:d9:70:13:90:89
Interface Name: v1
Ip Address: 1.1.1.1/24
Ip Addresses:
1.1.1.1/24
1.1.1.2/24
1.1.1.4/24
2001:de7:0:f101::1/64
2001:db8:0:f101::1/64
fe80::1420:84ff:fe10:17aa/64
Mac:      16:20:84:10:17:aa
```

2.14.2. Viewing the IP address of a virtual machine interface in the web console

The IP information displays in the **Virtual Machine Overview** screen for the virtual machine.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. Click the virtual machine name to open the **Virtual Machine Overview** screen.

The information for each attached vNIC is displayed under **IP ADDRESSES**.

2.15. CONFIGURING PXE BOOTING FOR VIRTUAL MACHINES

PXE booting, or network booting, is available in container-native virtualization. Network booting allows a computer to boot and load an operating system or other program without requiring a locally attached storage device. For example, you can use it to choose your desired OS image from a PXE server when deploying a new host.

Prerequisites

- A Linux bridge must be [connected](#).
- The PXE server must be connected to the same VLAN as the bridge.

2.15.1. Container-native virtualization networking glossary

Container-native virtualization provides advanced networking functionality by using custom resources and plug-ins.

The following terms are used throughout container-native virtualization documentation:

Container Network Interface (CNI)

a [Cloud Native Computing Foundation](#) project, focused on container network connectivity. Container-native virtualization uses CNI plug-ins to build upon the basic Kubernetes networking functionality.

Multus

a "meta" CNI plug-in that allows multiple CNIs to exist so that a Pod or virtual machine can use the interfaces it needs.

Custom Resource Definition (CRD)

a [Kubernetes](#) API resource that allows you to define custom resources, or an object defined by using the CRD API resource.

NetworkAttachmentDefinition

a CRD introduced by the Multus project that allows you to attach Pods, virtual machines, and virtual machine instances to one or more networks.

Preboot eXecution Environment (PXE)

an interface that enables an administrator to boot a client machine from a server over the network. Network booting allows you to remotely load operating systems and other software onto the client.

2.15.2. PXE booting with a specified MAC address

As an administrator, you can boot a client over the network by first creating a NetworkAttachmentDefinition object for your PXE network. Then, reference the NetworkAttachmentDefinition in your virtual machine instance configuration file before you start the virtual machine instance. You can also specify a MAC address in the virtual machine instance configuration file, if required by the PXE server.

Prerequisites

- A Linux bridge must be connected.
- The PXE server must be connected to the same VLAN as the bridge.

Procedure

1. Configure a PXE network on the cluster:
 - a. Create the NetworkAttachmentDefinition file for PXE network **pxe-net-conf**:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: pxe-net-conf
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "pxe-net-conf",
    "plugins": [
      {
        "type": "cnv-bridge",
        "bridge": "br1",
        "ipam": {}
      },
      {
        "type": "cnv-tuning" 1
      }
    ]
  }'
```

- 1** The **cnv-tuning** plug-in provides support for custom MAC addresses.

**NOTE**

The virtual machine instance will be attached to the bridge **br1** through an access port with the requested VLAN.

2. Create the NetworkAttachmentDefinition object by using the file you created in the previous step:

```
$ oc create -f pxe-net-conf.yaml
```

3. Edit the virtual machine instance configuration file to include the details of the interface and network.
 - a. Specify the network and MAC address, if required by the PXE server. If the MAC address is not specified, a value is assigned automatically. However, note that at this time, MAC addresses assigned automatically are not persistent. Ensure that **bootOrder** is set to **1** so that the interface boots first. In this example, the interface is connected to a network called **<pxe-net>**:

```
interfaces:
- masquerade: {}
  name: default
- bridge: {}
  name: pxe-net
  macAddress: de:00:00:00:00:de
  bootOrder: 1
```

**NOTE**

Boot order is global for interfaces and disks.

- b. Assign a boot device number to the disk to ensure proper booting after operating system provisioning.

Set the disk **bootOrder** value to **2**:

```
devices:
  disks:
  - disk:
    bus: virtio
    name: containerdisk
    bootOrder: 2
```

- c. Specify that the network is connected to the previously created NetworkAttachmentDefinition. In this scenario, **<pxe-net>** is connected to the NetworkAttachmentDefinition called **<pxe-net-conf>**:

```
networks:
- name: default
  pod: {}
- name: pxe-net
  multus:
    networkName: pxe-net-conf
```

4. Create the virtual machine instance:

```
$ oc create -f vmi-pxe-boot.yaml
virtualmachineinstance.kubevirt.io "vmi-pxe-boot" created
```

5. Wait for the virtual machine instance to run:

```
$ oc get vmi vmi-pxe-boot -o yaml | grep -i phase
phase: Running
```

6. View the virtual machine instance using VNC:

```
$ virtctl vnc vmi-pxe-boot
```

7. Watch the boot screen to verify that the PXE boot is successful.

8. Log in to the virtual machine instance:

```
$ virtctl console vmi-pxe-boot
```

9. Verify the interfaces and MAC address on the virtual machine and that the interface connected to the bridge has the specified MAC address. In this case, we used **eth1** for the PXE boot, without an IP address. The other interface, **eth0**, got an IP address from OpenShift Container Platform.

```
$ ip addr
...
3. eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen
1000
link/ether de:00:00:00:00:de brd ff:ff:ff:ff:ff:ff
```

2.15.3. Template: virtual machine instance configuration file for PXE booting

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  creationTimestamp: null
  labels:
    special: vmi-pxe-boot
  name: vmi-pxe-boot
spec:
  domain:
  devices:
    disks:
      - disk:
          bus: virtio
          name: containerdisk
          bootOrder: 2
      - disk:
          bus: virtio
          name: cloudinitdisk
    interfaces:
      - masquerade: {}
```

```

    name: default
  - bridge: {}
    name: pxe-net
    macAddress: de:00:00:00:00:de
    bootOrder: 1
  machine:
    type: ""
  resources:
    requests:
      memory: 1024M
  networks:
  - name: default
    pod: {}
  - multus:
      networkName: pxe-net-conf
      name: pxe-net
  terminationGracePeriodSeconds: 0
  volumes:
  - name: containerdisk
    containerDisk:
      image: kubevirt/fedora-cloud-container-disk-demo
  - cloudInitNoCloud:
      userData: |
        #!/bin/bash
        echo "fedora" | passwd fedora --stdin
      name: cloudinitdisk
  status: {}

```

2.16. MANAGING GUEST MEMORY

If you want to adjust guest memory settings to suit a specific use case, you can do so by editing the guest's YAML configuration file. Container-native virtualization allows you to configure guest memory overcommitment and disable guest memory overhead accounting.

Both of these procedures carry some degree of risk. Proceed only if you are an experienced administrator.

2.16.1. Configuring guest memory overcommitment

If your virtual workload requires more memory than available, you can use memory overcommitment to allocate all or most of the host's memory to your virtual machine instances. Enabling memory overcommitment means you can maximize resources that are normally reserved for the host.

For example, if the host has 32 GB RAM, you can use memory overcommitment to fit 8 virtual machines with 4 GB RAM each. This allocation works under the assumption that the virtual machines will not use all of their memory at the same time.

Procedure

1. To explicitly tell the virtual machine instance that it has more memory available than was requested from the cluster, edit the virtual machine configuration file and set **spec.domain.memory.guest** to a higher value than **spec.domain.resources.requests.memory**. This process is called memory overcommitment.

In this example, **1024M** is requested from the cluster, but the virtual machine instance is told that it has **2048M** available. As long as there is enough free memory available on the node, the virtual machine instance will consume up to 2048M.

```
kind: VirtualMachine
spec:
  template:
    domain:
      resources:
        requests:
          memory: 1024M
        memory:
          guest: 2048M
```



NOTE

The same eviction rules as those for Pods apply to the virtual machine instance if the node is under memory pressure.

2. Create the virtual machine:

```
$ oc create -f <file name>.yaml
```

2.16.2. Disabling guest memory overhead accounting



WARNING

This procedure is only useful in certain use-cases and must only be attempted by advanced users.

A small amount of memory is requested by each virtual machine instance in addition to the amount that you request. This additional memory is used for the infrastructure that wraps each **VirtualMachineInstance** process.

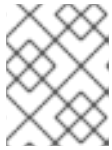
Though it is not usually advisable, it is possible to increase the virtual machine instance density on the node by disabling guest memory overhead accounting.

Procedure

1. To disable guest memory overhead accounting, edit the YAML configuration file and set the **overcommitGuestOverhead** value to **true**. This parameter is disabled by default.

```
kind: VirtualMachine
spec:
  template:
    domain:
      resources:
```

```
overcommitGuestOverhead: true
requests:
  memory: 1024M
```



NOTE

If **overcommitGuestOverhead** is enabled, it adds the guest overhead to memory limits, if present.

2. Create the virtual machine:

```
$ oc create -f <file name>.yaml
```


2.17. CREATING VIRTUAL MACHINE TEMPLATES

Using Virtual machines templates is an easy way to create multiple virtual machines with similar configuration. After a template is created, reference the template when creating virtual machines.

2.17.1. Creating a virtual machine template with the interactive wizard in the web console

The web console features an interactive wizard that guides you through the **Basic Settings**, **Networking**, and **Storage** screens to simplify the process of creating virtual machine templates. All required fields are marked with a *. The wizard prevents you from moving to the next screen until you provide values in the required fields.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machine Templates**
2. Click **Create Template** and select **Create with Wizard**
3. Fill in all required **Basic Settings**.
4. Click **Next** to progress to the **Networking** screen. A NIC that is named **nic0** is attached by default.
 - a. Optional: Click **Create NIC** to create additional NICs.
 - b. Optional: You can remove any or all NICs by clicking the Options menu  and selecting **Remove NIC**. Virtual machines created from a template do not need a NIC attached. NICs can be created after a virtual machine has been created.
5. Click **Next** to progress to the **Storage** screen.
 - a. Optional: Click **Create Disk** to create additional disks.
 - b. Optional: Click a disk to modify available fields. Click the ✓ button to save the changes.
 - c. Optional: Click **Attach Disk** to choose an available disk from the **Select Storage** list.

**NOTE**

If either **URL** or **Container** are selected as the **Provision Source** in the **Basic Settings** screen, a **rootdisk** disk is created and attached to virtual machines as the **Bootable Disk**. You can modify the **rootdisk** but you cannot remove it.

A **Bootable Disk** is not required for virtual machines provisioned from a **PXE** source if there are no disks attached to the virtual machine. If one or more disks are attached to the virtual machine, you must select one as the **Bootable Disk**.

- Click **Create Virtual Machine Template** ▶ The **Results** screen displays the JSON configuration file for the virtual machine template.
The template is listed in **Workloads** → **Virtual Machine Templates**

2.17.2. Virtual machine template interactive wizard fields

The following tables describe the fields for the **Basic Settings**, **Networking**, and **Storage** panes in the **Create Virtual Machine Template** interactive wizard.

2.17.2.1. Virtual machine template wizard fields

Name	Parameter	Description
Name		The name can contain lower-case letters (a-z), numbers (0-9), and hyphens (-), up to a maximum of 253 characters. The first and last characters must be alphanumeric. The name must not contain upper-case letters, spaces, periods (.), or special characters.
Description		Optional description field.
Provision Source	PXE	Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.
	URL	Provision virtual machine from an image available from an HTTP or S3 endpoint.
	Container	Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: kubevirt/cirros-registry-disk-demo .
	Cloned Disk	Provision source is a cloned disk.

Name	Parameter	Description
	Import	Import virtual machine from a supported provider.
Operating System		A list of operating systems available in the cluster. This is the primary operating system for the virtual machine. If you select Import as the Provider Source , the operating system is filled in automatically, based on the operating system of the VMware virtual machine being imported.
Flavor	small, medium, large, tiny, Custom	Presets that determine the amount of CPU and memory allocated to the virtual machine.
Workload Profile	desktop	A virtual machine configuration for use on a desktop.
	generic	A virtual machine configuration that balances performance and compatibility for a broad range of workloads.
	high performance	A virtual machine configuration that is optimized for high-performance loads.
Use cloud-init		Select to enable the cloud-init fields.

2.17.2.2. Cloud-init fields

Name	Description
Hostname	Sets a specific host name for the virtual machine.
Authenticated SSH Keys	The user's public key that is copied to <code>~/.ssh/authorized_keys</code> on the virtual machine.
Use custom script	Replaces other options with a field in which you paste a custom cloud-init script.

2.17.2.3. Networking fields

Name	Description
Create NIC	Create a new NIC for the virtual machine.
NIC NAME	Name for the NIC.
MAC ADDRESS	MAC address for the network interface. If a MAC address is not specified, an ephemeral address is generated for the session.
NETWORK CONFIGURATION	List of available NetworkAttachmentDefinition objects.
BINDING METHOD	List of available binding methods. For the default Pod network, masquerade is the only recommended binding method. For secondary networks, use the bridge binding method. The masquerade method is not supported for non-default networks.
PXE NIC	List of PXE-capable networks. Only visible if PXE has been selected as the Provision Source .

2.17.2.4. Storage fields

Name	Description
Create Disk	Create a new disk for the virtual machine.
Attach Disk	Select an existing disk, from a list of available PVCs, to attach to the virtual machine.
DISK NAME	Name of the disk. The name can contain lower-case letters (a-z), numbers (0-9), hyphens (-), and periods (.), up to a maximum of 253 characters. The first and last characters must be alphanumeric. The name must not contain upper-case letters, spaces, or special characters.
SIZE (GB)	Size, in GB, of the disk.
STORAGE CLASS	Name of the underlying StorageClass .
Bootable Disk	List of available disks from which the virtual machine will boot. This is locked to rootdisk if the Provision Source of the virtual machine is URL or Container .

2.18. EDITING A VIRTUAL MACHINE TEMPLATE

You can edit a virtual machine template in the web console.

2.18.1. Editing a virtual machine template in the web console

You can edit the YAML configuration of a virtual machine template from the web console.

Not all parameters can be modified. If you click **Save** with an invalid configuration, an error message indicates the parameter that cannot be modified.



NOTE

Navigating away from the YAML screen while editing cancels any changes to the configuration that you made.

Procedure

1. In the container-native virtualization console, click **Workloads → Virtual Machine Templates**
2. Select a template.
3. Click the **YAML** tab to display the editable configuration.
4. Edit the file and click **Save**.

A confirmation message, which includes the updated version number for the object, shows the modification has been successful.

2.19. DELETING A VIRTUAL MACHINE TEMPLATE


You can delete a virtual machine template in the web console.

2.19.1. Deleting a virtual machine template in the web console

Deleting a virtual machine template permanently removes it from the cluster.

Procedure

1. In the container-native virtualization console, click **Workloads → Virtual Machine Templates**
2. You can delete the virtual machine template from this pane, which makes it easier to perform actions on multiple templates in the one pane, or from the **Virtual Machine Template Details** pane where you can view comprehensive details of the selected template:

- Click the Options menu  of the template to delete and select **Delete Template**.
- Click the template name to open the **Virtual Machine Template Details** pane and click **Actions → Delete Template**.

3. In the confirmation pop-up window, click **Delete** to permanently delete the template.

2.20. CLONING A VIRTUAL MACHINE DISK INTO A NEW DATAVOLUME

You can clone the PersistentVolumeClaim (PVC) of a virtual machine disk into a new DataVolume by referencing the source PVC in your DataVolume configuration file.

Prerequisites

- You may need to [define a StorageClass or prepare CDI scratch space](#) for this operation to complete successfully. The [CDI supported operations matrix](#) shows the conditions that require scratch space.

2.20.1. About DataVolumes

DataVolume objects are custom resources that are provided by the Containerized Data Importer (CDI) project. DataVolumes orchestrate import, clone, and upload operations that are associated with an underlying PersistentVolumeClaim (PVC). DataVolumes are integrated with KubeVirt, and they prevent a virtual machine from being started before the PVC has been prepared.

2.20.2. Cloning the PersistentVolumeClaim of a virtual machine disk into a new DataVolume

You can clone a PersistentVolumeClaim (PVC) of an existing virtual machine disk into a new DataVolume. The new DataVolume can then be used for a new virtual machine.



NOTE

When a DataVolume is created independently of a virtual machine, the lifecycle of the DataVolume is independent of the virtual machine. If the virtual machine is deleted, neither the DataVolume nor its associated PVC is deleted.

Prerequisites

- Determine the PVC of an existing virtual machine disk to use. You must power down the virtual machine that is associated with the PVC before you can clone it.
- Install the OpenShift Command-line Interface (CLI), commonly known as **oc**.

Procedure

1. Examine the virtual machine disk you want to clone to identify the name and namespace of the associated PVC.
2. Create a YAML file for a DataVolume object that specifies the name of the new DataVolume, the name and namespace of the source PVC, and the size of the new DataVolume.

For example:

```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: cloner-datavolume 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  pvc:
```

```

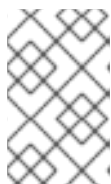
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 500Mi 4

```

- 1 The name of the new DataVolume.
- 2 The namespace where the source PVC exists.
- 3 The name of the source PVC.
- 4 The size of the new DataVolume. You must allocate enough space, or the cloning operation fails. The size must be the same or larger as the source PVC.

3. Start cloning the PVC by creating the DataVolume:

```
$ oc create -f <cloner-datavolume>.yaml
```



NOTE

DataVolumes prevent a virtual machine from starting before the PVC is prepared, so you can create a virtual machine that references the new DataVolume while the PVC clones.

2.20.3. Template: DataVolume clone configuration file

example-clone-dv.yaml

```

apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: "example-clone-dv"
spec:
  source:
    pvc:
      name: source-pvc
      namespace: example-ns
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: "1G"

```

2.20.4. CDI supported operations matrix

This matrix shows the supported CDI operations for content types against endpoints, and which of these operations requires scratch space.

Content types	HTTP	HTTPS	HTTP basic auth	Registry	S3 Bucket	Upload
KubeVirt(QCOW2)	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2** ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* ✓ GZ* ✓ XZ*
Archive+	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR

✓ Supported operation

Unsupported operation

* Requires scratch space

** Requires scratch space if a custom certificate authority is required

+ Archive does not support block mode DVs

2.21. CLONING A VIRTUAL MACHINE BY USING A DATAVOLUMETEMPLATE

You can create a new virtual machine by cloning the PersistentVolumeClaim (PVC) of an existing VM. By including a **dataVolumeTemplate** in your virtual machine configuration file, you create a new DataVolume from the original PVC.

Prerequisites

- You may need to [define a StorageClass or prepare CDI scratch space](#) for this operation to complete successfully. The [CDI supported operations matrix](#) shows the conditions that require scratch space.

2.21.1. About DataVolumes

DataVolume objects are custom resources that are provided by the Containerized Data Importer (CDI) project. DataVolumes orchestrate import, clone, and upload operations that are associated with an underlying PersistentVolumeClaim (PVC). DataVolumes are integrated with KubeVirt, and they prevent a virtual machine from being started before the PVC has been prepared.

2.21.2. Creating a new virtual machine from a cloned PersistentVolumeClaim by using a DataVolumeTemplate

You can create a virtual machine that clones the PersistentVolumeClaim (PVC) of an existing virtual machine into a DataVolume. By referencing a **dataVolumeTemplate** in the virtual machine **spec**, the **source** PVC is cloned to a DataVolume, which is then automatically used for the creation of the virtual machine.



NOTE

When a DataVolume is created as part of the DataVolumeTemplate of a virtual machine, the lifecycle of the DataVolume is then dependent on the virtual machine. If the virtual machine is deleted, the DataVolume and associated PVC are also deleted.

Prerequisites

- Determine the PVC of an existing virtual machine disk to use. You must power down the virtual machine that is associated with the PVC before you can clone it.
- Install the OpenShift Command-line Interface (CLI), commonly known as **oc**.

Procedure

1. Examine the virtual machine you want to clone to identify the name and namespace of the associated PVC.
2. Create a YAML file for a **VirtualMachine** object. The following virtual machine example clones **my-favorite-vm-disk**, which is located in the **source-namespace** namespace. The **2Gi** DataVolume called **favorite-clone** is created from **my-favorite-vm-disk**.
For example:

```

apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm-dv-clone
  name: vm-dv-clone 1
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-dv-clone
    spec:
      domain:
        devices:
          disks:
            - disk:
                bus: virtio
                name: root-disk
          resources:
            requests:
              memory: 64M
          volumes:
            - dataVolume:
                name: favorite-clone
                name: root-disk
      dataVolumeTemplates:
        - metadata:
            name: favorite-clone
          spec:
            pvc:
              accessModes:

```

```

- ReadWriteOnce
resources:
  requests:
    storage: 2Gi
source:
  pvc:
    namespace: "source-namespace"
    name: "my-favorite-vm-disk"

```

1 The virtual machine to create.

3. Create the virtual machine with the PVC-cloned DataVolume:

```
$ oc create -f <vm-clone-datavolumetemplate>.yaml
```

2.21.3. Template: DataVolume virtual machine configuration file

example-dv-vm.yaml

```

apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: example-vm
  name: example-vm
spec:
  dataVolumeTemplates:
  - metadata:
    name: example-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 1G
        source:
          http:
            url: "" 1
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: example-vm
    spec:
      domain:
        cpu:
          cores: 1
      devices:
        disks:
        - disk:
            bus: virtio
            name: example-dv-disk
  machine:

```

```

type: q35
resources:
  requests:
    memory: 1G
terminationGracePeriodSeconds: 0
volumes:
- dataVolume:
  name: example-dv
  name: example-dv-disk

```

- 1 The **HTTP** source of the image you want to import, if applicable.

2.21.4. CDI supported operations matrix

This matrix shows the supported CDI operations for content types against endpoints, and which of these operations requires scratch space.

Content types	HTTP	HTTPS	HTTP basic auth	Registry	S3 Bucket	Upload
KubeVirt(QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* <input type="checkbox"/> GZ <input type="checkbox"/> XZ	✓ QCOW2* ✓ GZ* ✓ XZ*	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* <input type="checkbox"/> GZ <input type="checkbox"/> XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ✓ GZ* ✓ XZ*
Archive+	✓ TAR	✓ TAR	✓ TAR	<input type="checkbox"/> TAR	<input type="checkbox"/> TAR	<input type="checkbox"/> TAR

✓ Supported operation

Unsupported operation

* Requires scratch space

** Requires scratch space if a custom certificate authority is required

+ Archive does not support block mode DVs

2.22. UPLOADING LOCAL DISK IMAGES BY USING THE VIRTCTL TOOL

You can upload a disk image that is stored locally by using the **virtctl** command-line utility.

Prerequisites

- [Install](#) the **kubevirt-virtctl** package
- You may need to [define a StorageClass](#) or [prepare CDI scratch space](#) for this operation to complete successfully.

2.22.1. CDI supported operations matrix

This matrix shows the supported CDI operations for content types against endpoints, and which of these operations requires scratch space.

Content types	HTTP	HTTPS	HTTP basic auth	Registry	S3 Bucket	Upload
KubeVirt(QCOW2)	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2** ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* ✓ GZ* ✓ XZ*
Archive+	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR

✓ Supported operation

Unsupported operation

* Requires scratch space

** Requires scratch space if a custom certificate authority is required

+ Archive does not support block mode DVs

2.22.2. Uploading a local disk image to a new PersistentVolumeClaim

You can use the **virtctl** CLI utility to upload a virtual machine disk image from a client machine to your cluster. Uploading the disk image creates a PersistentVolumeClaim (PVC) that you can associate with a virtual machine.

Prerequisites

- A virtual machine disk image, in RAW, ISO, or QCOW2 format, optionally compressed by using **xz** or **gz**.
- The **kubevirt-virtctl** package must be installed on the client machine.
- The client machine must be configured to trust the OpenShift Container Platform router's certificate.

Procedure

1. Identify the following items:
 - File location of the VM disk image you want to upload
 - Name and size required for the resulting PVC

2. Run the **virtctl image-upload** command to upload your VM image. You must specify the PVC name, PVC size, and file location. For example:

```
$ virtctl image-upload --pvc-name=upload-fedora-pvc --pvc-size=10Gi --image-path=/images/fedora28.qcow2
```

CAUTION

To allow insecure server connections when using HTTPS, use the **--insecure** parameter. Be aware that when you use the **--insecure** flag, the authenticity of the upload endpoint is **not** verified.

3. To verify that the PVC was created, view all PVC objects:

```
$ oc get pvc
```

2.23. EXPANDING VIRTUAL STORAGE BY ADDING BLANK DISK IMAGES

You can increase your storage capacity or create new data partitions by adding blank disk images to container-native virtualization.

2.23.1. About DataVolumes

DataVolume objects are custom resources that are provided by the Containerized Data Importer (CDI) project. DataVolumes orchestrate import, clone, and upload operations that are associated with an underlying PersistentVolumeClaim (PVC). DataVolumes are integrated with KubeVirt, and they prevent a virtual machine from being started before the PVC has been prepared.

2.23.2. Creating a blank disk image with DataVolumes

You can create a new blank disk image in a PersistentVolumeClaim by customizing and deploying a DataVolume configuration file.

Prerequisites

- At least one available PersistentVolume
- Install the OpenShift Command-line Interface (CLI), commonly known as **oc**

Procedure

1. Edit the DataVolume configuration file:

```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: blank-image-datavolume
spec:
  source:
    blank: {}
  pvc:
```

```

# Optional: Set the storage class or omit to accept the default
# storageClassName: "hostpath"
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 500Mi

```

2. Create the blank disk image by running the following command:

```
$ oc create -f <blank-image-datavolume>.yaml
```

2.23.3. Template: DataVolume configuration file for blank disk images

blank-image-datavolume.yaml

```

apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: blank-image-datavolume
spec:
  source:
    blank: {}
  pvc:
    # Optional: Set the storage class or omit to accept the default
    # storageClassName: "hostpath"
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 500Mi

```

2.24. PREPARING CDI SCRATCH SPACE

2.24.1. About DataVolumes

DataVolume objects are custom resources that are provided by the Containerized Data Importer (CDI) project. DataVolumes orchestrate import, clone, and upload operations that are associated with an underlying PersistentVolumeClaim (PVC). DataVolumes are integrated with KubeVirt, and they prevent a virtual machine from being started before the PVC has been prepared.

2.24.2. Understanding scratch space

The Containerized Data Importer (CDI) requires scratch space (temporary storage) to complete some operations, such as importing and uploading virtual machine images. During this process, the CDI provisions a scratch space PVC equal to the size of the PVC backing the destination DataVolume (DV). The scratch space PVC is deleted after the operation completes or aborts.

The CDIConfig object allows you to define which StorageClass to use to bind the scratch space PVC by setting the **scratchSpaceStorageClass** in the **spec:** section of the CDIConfig object.

If the defined StorageClass does not match a StorageClass in the cluster, then the default StorageClass defined for the cluster is used. If there is no default StorageClass defined in the cluster, the StorageClass used to provision the original DV or PVC is used.



NOTE

The CDI requires requesting scratch space with a **file** volume mode, regardless of the PVC backing the origin DataVolume. If the origin PVC is backed by **block** volume mode, you must define a StorageClass capable of provisioning **file** volume mode PVCs.

Manual provisioning

If there are no storage classes, the CDI will use any PVCs in the project that match the size requirements for the image. If there are no PVCs that match these requirements, the CDI import Pod will remain in a **Pending** state until an appropriate PVC is made available or until a timeout function kills the Pod.

2.24.3. Defining a StorageClass in the CDI configuration

Define a StorageClass in the CDI configuration to dynamically provision scratch space for CDI operations.

Procedure

- Use the **oc** client to edit the **cdiconfig/config** and add or edit the **spec: scratchSpaceStorageClass** to match a StorageClass in the cluster.

```
$ oc edit cdiconfig/config
```

```
API Version: cdi.kubevirt.io/v1alpha1
kind: CDIConfig
metadata:
  name: config
  ...
spec:
  scratchSpaceStorageClass: "<storage_class>"
  ...
```

2.24.4. CDI operations that require scratch space

Type	Reason
Registry imports	The CDI must download the image to a scratch space and extract the layers to find the image file. The image file is then passed to QEMU-IMG for conversion to a raw disk.
Upload image	QEMU-IMG does not accept input from STDIN. Instead, the image to upload is saved in scratch space before it can be passed to QEMU-IMG for conversion.

Type	Reason
HTTP imports of archived images	QEMU-IMG does not know how to handle the archive formats CDI supports. Instead, the image is unarchived and saved into scratch space before it is passed to QEMU-IMG.
HTTP imports of authenticated images	QEMU-IMG inadequately handles authentication. Instead, the image is saved to scratch space and authenticated before it is passed to QEMU-IMG.
HTTP imports of custom certificates	QEMU-IMG inadequately handles custom certificates of HTTPS endpoints. Instead, the CDI downloads the image to scratch space before passing the file to QEMU-IMG.

2.24.5. CDI supported operations matrix

This matrix shows the supported CDI operations for content types against endpoints, and which of these operations requires scratch space.

Content types	HTTP	HTTPS	HTTP basic auth	Registry	S3 Bucket	Upload
KubeVirt(QCOW2)	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2** ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2 ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ* 	<ul style="list-style-type: none"> ✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* <input type="checkbox"/> GZ <input type="checkbox"/> XZ 	<ul style="list-style-type: none"> ✓ RAW ✓ GZ ✓ XZ 	<ul style="list-style-type: none"> ✓ RAW* ✓ GZ* ✓ XZ*
Archive+	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> ✓ TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR 	<ul style="list-style-type: none"> <input type="checkbox"/> TAR

✓ Supported operation

Unsupported operation

* Requires scratch space

** Requires scratch space if a custom certificate authority is required

+ Archive does not support block mode DVs

Additional resources

- See the [Dynamic provisioning](#) section for more information on StorageClasses and how these are defined in the cluster.

2.25. VIRTUAL MACHINE LIVE MIGRATION

2.25.1. Understanding live migration

Live migration is the process of moving a running virtual machine instance to another node in the cluster without interruption to the virtual workload or access. This can be a manual process, if you select a virtual machine instance to migrate to another node, or an automatic process, if the virtual machine instance has a **LiveMigrate** eviction strategy and the node on which it is running is placed into maintenance.



NOTE

NFS shared volumes are the only shared data volume type supported for live migration in container-native virtualization 2.0.

Additional resources:

- [Migrating a virtual machine instance to another node](#)
- [Node maintenance mode](#)
- [Live migration limiting](#)

2.26. LIVE MIGRATION LIMITS AND TIMEOUTS

Live migration limits and timeouts are applied so that migration processes do not overwhelm the cluster. Configure these settings by editing the **kubevirt-config** configuration file.

2.26.1. Configuring live migration limits and timeouts

Configure live migration limits and timeouts for the cluster by adding updated key:value fields to the **kubevirt-config** configuration file, which is located in the **kubevirt-hyperconverged** namespace.

Procedure

- Edit the **kubevirt-config** configuration file and add the necessary live migration parameters. The following example shows the default values:

```
$ oc edit configmap kubevirt-config -n kubevirt-hyperconverged
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kubevirt-config
  namespace: kubevirt
  labels:
    kubevirt.io: ""
data:
  feature-gates: "LiveMigration"
  migrations: |-
    parallelMigrationsPerCluster: 5
    parallelOutboundMigrationsPerNode: 2
```

bandwidthPerMigration: 64Mi
 completionTimeoutPerGiB: 800
 progressTimeout: 150

2.26.2. Cluster-wide live migration limits and timeouts

Table 2.5. Migration parameters

Parameter	Description	Default
parallelMigrationsPerCluster	Number of migrations running in parallel in the cluster.	5
parallelOutboundMigrationsPerNode	Maximum number of outbound migrations per node.	2
bandwidthPerMigration	Bandwidth limit of each migration, in MiB/s.	64Mi
completionTimeoutPerGiB	The migration will be canceled if it has not completed in this time, in seconds per GiB of memory. For example, a virtual machine instance with 6GiB memory will timeout if it has not completed migration in 4800 seconds. If the Migration Method is BlockMigration , the size of the migrating disks is included in the calculation.	800
progressTimeout	The migration will be canceled if memory copy fails to make progress in this time, in seconds.	150

2.27. MIGRATING A VIRTUAL MACHINE INSTANCE TO ANOTHER NODE

Manually initiate a live migration of a virtual machine instance to another node using either the web console or the CLI.

2.27.1. Initiating live migration of a virtual machine instance in the web console

Migrate a running virtual machine instance to a different node in the cluster.




NOTE

The **Migrate Virtual Machine** action is visible to all users but only admin users can initiate a virtual machine migration.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. You can initiate the migration from this screen, which makes it easier to perform actions on multiple virtual machines in the one screen, or from the **Virtual Machine Details** screen where you can view comprehensive details of the selected virtual machine:

- Click the Options menu  at the end of virtual machine and select **Migrate Virtual Machine**.
- Click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** → **Migrate Virtual Machine**

3. Click **Migrate** to migrate the virtual machine to another node.

2.27.2. Initiating live migration of a virtual machine instance in the CLI

Initiate a live migration of a running virtual machine instance by creating a **VirtualMachineInstanceMigration** object in the cluster and referencing the name of the virtual machine instance.

Procedure

1. Create a **VirtualMachineInstanceMigration** configuration file for the virtual machine instance to migrate. For example, **vmi-migrate.yaml**:

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstanceMigration
metadata:
  name: migration-job
spec:
  vmiName: vmi-fedora
```

2. Create the object in the cluster:

```
$ oc create -f vmi-migrate.yaml
```

The **VirtualMachineInstanceMigration** object triggers a live migration of the virtual machine instance. This object exists in the cluster for as long as the virtual machine instance is running, unless manually deleted.

Additional resources:

- [Monitoring live migration of a virtual machine instance](#)
- [Cancelling the live migration of a virtual machine instance](#)

2.28. MONITORING LIVE MIGRATION OF A VIRTUAL MACHINE INSTANCE

You can monitor the progress of a live migration of a virtual machine instance from either the web console or the CLI.

2.28.1. Monitoring live migration of a virtual machine instance in the web console

For the duration of the migration, the virtual machine has a status of **Migrating**. This status is displayed in the **Virtual Machines** list or in the **Virtual Machine Details** screen for the migrating virtual machine.

Procedure

- In the container-native virtualization console, click **Workloads** → **Virtual Machines**.

2.28.2. Monitoring live migration of a virtual machine instance in the CLI

The status of the virtual machine migration is stored in the **Status** component of the **VirtualMachineInstance** configuration.

Procedure

- Use the **oc describe** command on the migrating virtual machine instance:

```
$ oc describe vmi vmi-fedora
```

```
+
```

```
...
```

```
Status:
```

```
Conditions:
```

```
  Last Probe Time:  <nil>
```

```
  Last Transition Time: <nil>
```

```
  Status:           True
```

```
  Type:             LiveMigratable
```

```
Migration Method: LiveMigration
```

```
Migration State:
```

```
  Completed:        true
```

```
  End Timestamp:    2018-12-24T06:19:42Z
```

```
  Migration UID:    d78c8962-0743-11e9-a540-fa163e0c69f1
```

```
  Source Node:      node2.example.com
```

```
  Start Timestamp:  2018-12-24T06:19:35Z
```

```
  Target Node:      node1.example.com
```

```
  Target Node Address: 10.9.0.18:43891
```

```
  Target Node Domain Detected: true
```


2.29. CANCELLING THE LIVE MIGRATION OF A VIRTUAL MACHINE INSTANCE

Cancel the live migration so that the virtual machine instance remains on the original node.

You can cancel a live migration from either the web console or the CLI.

2.29.1. Cancelling live migration of a virtual machine instance in the web console




A live migration of the virtual machine instance can be cancelled using the Options menu  found on each virtual machine in the **Workloads → Virtual Machines** screen, or from the **Actions** menu on the **Virtual Machine Details** screen.

Procedure

1. In the container-native virtualization console, click **Workloads → Virtual Machines**.
2. You can cancel the migration from this screen, which makes it easier to perform actions on multiple virtual machines in the one screen, or from the **Virtual Machine Details** screen where you can view comprehensive details of the selected virtual machine:



- Click the Options menu  at the end of virtual machine and select **Cancel Virtual Machine Migration**.
 - Click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions → Cancel Virtual Machine Migration**
3. Click **Cancel Migration** to cancel the virtual machine live migration.

2.29.2. Cancelling live migration of a virtual machine instance in the CLI

Cancel the live migration of a virtual machine instance by deleting the **VirtualMachineInstanceMigration** object associated with the migration.

Procedure

- Delete the **VirtualMachineInstanceMigration** object that triggered the live migration, **migration-job** in this example:

```
$ oc delete vmim migration-job
```

2.30. NODE MAINTENANCE MODE

2.30.1. Understanding node maintenance mode

Placing a node into maintenance marks the node as unschedulable and drains all the virtual machines and pods from it. Virtual machine instances that have a **LiveMigrate** eviction strategy are live migrated to another node without loss of service. This eviction strategy is configured by default in virtual machine created from common templates but must be configured manually for custom virtual machines.

Virtual machine instances without an eviction strategy will be deleted on the node and recreated on another node.



NOTE

NFS shared volumes are the only shared data volume type supported for live migration in container-native virtualization 2.0.

Additional resources:

- [Virtual machine live migration](#)
- [Configuring virtual machine eviction strategy](#)

2.31. CONFIGURING VIRTUAL MACHINE EVICTION STRATEGY

The **LiveMigrate** eviction strategy ensures that a virtual machine instance is not interrupted if the node is placed into maintenance or drained. Virtual machines instances with this eviction strategy will be live migrated to another node.

2.31.1. Configuring custom virtual machines with the LiveMigration eviction strategy

You only need to configure the **LiveMigration** eviction strategy on custom virtual machines. Common templates have this eviction strategy configured by default.

Procedure

1. Add the **evictionStrategy: LiveMigrate** option to the **spec** section in the virtual machine configuration file. This example uses **oc edit** to update the relevant snippet of the **VirtualMachine** configuration file:

```
$ oc edit vm <custom-vm> -n <my-namespace>
```

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: custom-vm
spec:
  terminationGracePeriodSeconds: 30
  evictionStrategy: LiveMigrate
  domain:
    resources:
      requests:
    ...
```

2. Restart the virtual machine for the update to take effect:

```
$ virtctl restart <custom-vm> -n <my-namespace>
```

2.32. SETTING A NODE TO MAINTENANCE MODE

2.32.1. Understanding node maintenance mode

Placing a node into maintenance marks the node as unschedulable and drains all the virtual machines and pods from it. Virtual machine instances that have a **LiveMigrate** eviction strategy are live migrated to another node without loss of service. This eviction strategy is configured by default in virtual machine created from common templates but must be configured manually for custom virtual machines.


Virtual machine instances without an eviction strategy will be deleted on the node and recreated on another node.

**NOTE**

NFS shared volumes are the only shared data volume type supported for live migration in container-native virtualization 2.0.


Place a node into maintenance from either the web console or the CLI.

2.32.2. Setting a node to maintenance mode in the web console

Set a node to maintenance mode using the Options menu  found on each node in the **Compute** → **Nodes** list, or using the **Actions** control of the **Node Details** screen.

Procedure

1. In the container-native virtualization console, click **Compute** → **Nodes**.
2. You can set the node to maintenance from this screen, which makes it easier to perform actions on multiple nodes in the one screen or from the **Node Details** screen where you can view comprehensive details of the selected node:

- Click the Options menu  at the end of the node and select **Start Maintenance**.
- Click the node name to open the **Node Details** screen and click **Actions** → **Start Maintenance**.

3. Click **Start Maintenance** in the confirmation window.

The node will live migrate virtual machine instances that have the **liveMigration** eviction strategy, and the node is no longer schedulable. All other pods and virtual machines on the node are deleted and recreated on another node.

2.32.3. Setting a node to maintenance mode in the CLI

Set a node to maintenance mode by creating a **NodeMaintenance** Custom Resource (CR) object that references the node name and the reason for setting it to maintenance mode.

Procedure

1. Create the node maintenance CR configuration. This example uses a CR that is called **node02-maintenance.yaml**:

```
apiVersion: kubevirt.io/v1alpha1
kind: NodeMaintenance
metadata:
  name: node02-maintenance
spec:
  nodeName: node02
  reason: "Replacing node02"
```

2. Create the **NodeMaintenance** object in the cluster:

```
$ oc apply -f <node02-maintenance.yaml>
```

The node live migrates virtual machine instances that have the **liveMigration** eviction strategy, and taint the node so that it is no longer schedulable. All other pods and virtual machines on the node are deleted and recreated on another node.

Additional resources:

- [Resuming a node from maintenance mode](#)

2.33. RESUMING A NODE FROM MAINTENANCE MODE

Resuming a node brings it out of maintenance mode and schedulable again.


Resume a node from maintenance from either the web console or the CLI.

2.33.1. Resuming a node from maintenance mode in the web console

Resume a node from maintenance mode using the Options menu  found on each node in the **Compute** → **Nodes** list, or using the **Actions** control of the **Node Details** screen.

Procedure

1. In the container-native virtualization console, click **Compute** → **Nodes**.
2. You can resume the node from this screen, which makes it easier to perform actions on multiple nodes in the one screen, or from the **Node Details** screen where you can view comprehensive details of the selected node:

- Click the Options menu  at the end of the node and select **Stop Maintenance**.
- Click the node name to open the **Node Details** screen and click **Actions** → **Stop Maintenance**.

3. Click **Stop Maintenance** in the confirmation window.

The node becomes schedulable, but virtual machine instances that were running on the node prior to maintenance will not automatically migrate back to this node.

2.33.2. Resuming a node from maintenance mode in the CLI

Resume a node from maintenance mode and make it schedulable again by deleting the **NodeMaintenance** object for the node.

Procedure

1. Find the **NodeMaintenance** object:

```
$ oc get nodemaintenance
```

- Optional: Inspect the **NodeMaintenance** object to ensure it is associated with the correct node:

```
$ oc describe nodemaintenance <node02-maintenance>
```

```
Name:      node02-maintenance
Namespace:
Labels:
Annotations:
API Version: kubevirt.io/v1alpha1
Kind:      NodeMaintenance
...
Spec:
  Node Name: node02
  Reason:   Replacing node02
```

- Delete the **NodeMaintenance** object:

```
$ oc delete nodemaintenance <node02-maintenance>
```

2.34. INSTALLING VIRTIO DRIVER ON AN EXISTING WINDOWS VIRTUAL MACHINE

2.34.1. Understanding VirtIO drivers

VirtIO drivers are paravirtualized device drivers required for Microsoft Windows virtual machines to run in container-native virtualization. The supported drivers are available in the **cnv-tech-preview/virtio-win** container disk of the [Red Hat Container Catalog](#).

The **cnv-tech-preview/virtio-win** container disk must be attached to the virtual machine as a SATA CD drive to enable driver installation. You can install VirtIO drivers during Windows installation on the virtual machine or added to an existing Windows installation.

After the drivers are installed, the **cnv-tech-preview/virtio-win** container disk can be removed from the virtual machine.

See also: [Installing Virtio drivers on a new Windows virtual machine](#).

2.34.2. Supported VirtIO drivers for Microsoft Windows virtual machines

Table 2.6. Supported drivers

Driver name	Hardware ID	Description
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	The block driver. Sometimes displays as an SCSI Controller in the Other devices group.
viornng	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	The entropy source driver. Sometimes displays as a PCI Device in the Other devices group.

Driver name	Hardware ID	Description
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	The network driver. Sometimes displays as an Ethernet Controller in the Other devices group. Available only if a VirtIO NIC is configured.

2.34.3. Adding VirtIO drivers container disk to a virtual machine

Container-native virtualization distributes VirtIO drivers for Microsoft Windows as a container disk, which is available from the [Red Hat Container Catalog](#). To install these drivers to a Windows virtual machine, attach the **cnv-tech-preview/virtio-win** container disk to the virtual machine as a SATA CD drive in the virtual machine configuration file.

Prerequisites

- Download the **cnv-tech-preview/virtio-win** container disk from the [Red Hat Container Catalog](#). This is not mandatory, because the container disk will be downloaded from the Red Hat registry if it not already present in the cluster, but it can reduce installation time.

Procedure

1. Add the **cnv-tech-preview/virtio-win** container disk as a **cdrom** disk in the Windows virtual machine configuration file. The container disk will be downloaded from the registry if it is not already present in the cluster.

```
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2 1
          cdrom:
            bus: sata
  volumes:
    - containerDisk:
        image: cnv-tech-preview/virtio-win
        name: virtiocontainerdisk
```

- 1** container-native virtualization boots virtual machine disks in the order defined in the **VirtualMachine** configuration file. You can either define other disks for the virtual machine before the **cnv-tech-preview/virtio-win** container disk or use the optional **bootOrder** parameter to ensure the virtual machine boots from the correct disk. If you specify the **bootOrder** for a disk, it must be specified for all disks in the configuration.

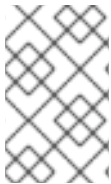
2. The disk is available once the virtual machine has started:
 - If you add the container disk to a running virtual machine, use **oc apply -f <vm.yaml>** in the CLI or reboot the virtual machine for the changes to take effect.

- If the virtual machine is not running, use **virtctl start <vm>**.

After the virtual machine has started, the VirtIO drivers can be installed from the attached SATA CD drive.

2.34.4. Installing VirtIO drivers on an existing Windows virtual machine

Install the VirtIO drivers from the attached SATA CD drive to an existing Windows virtual machine.



NOTE

This procedure uses a generic approach to adding drivers to Windows. The process might differ slightly between versions of Windows. Refer to the documentation for the version of Windows that you are installing.

Procedure

1. Start the virtual machine and connect to a graphical console.
2. Log in to a Windows user session.
3. Open **Device Manager** and expand **Other devices** to list any **Unknown device**.
 - a. You might need to open the **Device Properties** to identify the unknown device. Right-click the device and select **Properties**.
 - b. Click the **Details** tab and select **Hardware Ids** in the **Property** list.
 - c. Compare the **Value** for the **Hardware Ids** with the supported VirtIO drivers.
4. Right-click the device and select **Update Driver Software**.
5. Click **Browse my computer for driver software** and browse to the attached SATA CD drive, where the VirtIO drivers are located. The drivers are arranged hierarchically according to their driver type, operating system, and CPU architecture.
6. Click **Next** to install the driver.
7. Repeat this process for all the necessary VirtIO drivers.
8. After the driver installs, click **Close** to close the window.
9. Reboot the virtual machine to complete the driver installation.

2.34.5. Removing the VirtIO container disk from a virtual machine

After installing all required VirtIO drivers to the virtual machine, the **cnv-tech-preview/virtio-win** container disk no longer needs to be attached to the virtual machine. Remove the **cnv-tech-preview/virtio-win** container disk from the virtual machine configuration file.

Procedure

1. Edit the configuration file and remove the **disk** and the **volume**.

```
$ oc edit vm <vm-name>
```



```

spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2
          cdrom:
            bus: sata
      volumes:
        - containerDisk:
            image: cnv-tech-preview/virtio-win
            name: virtiocontainerdisk

```

2. Reboot the virtual machine for the changes to take effect.

2.35. INSTALLING VIRTIO DRIVER ON A NEW WINDOWS VIRTUAL MACHINE

Prerequisites

- Windows installation media accessible by the virtual machine, such as [importing an ISO into a data volume](#) and attaching it to the virtual machine.

2.35.1. Understanding VirtIO drivers

VirtIO drivers are paravirtualized device drivers required for Microsoft Windows virtual machines to run in container-native virtualization. The supported drivers are available in the **cnv-tech-preview/virtio-win** container disk of the [Red Hat Container Catalog](#).

The **cnv-tech-preview/virtio-win** container disk must be attached to the virtual machine as a SATA CD drive to enable driver installation. You can install VirtIO drivers during Windows installation on the virtual machine or added to an existing Windows installation.

After the drivers are installed, the **cnv-tech-preview/virtio-win** container disk can be removed from the virtual machine.

See also: [Installing VirtIO driver on an existing Windows virtual machine](#).

2.35.2. Supported VirtIO drivers for Microsoft Windows virtual machines

Table 2.7. Supported drivers

Driver name	Hardware ID	Description
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	The block driver. Sometimes displays as an SCSI Controller in the Other devices group.
viornrg	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	The entropy source driver. Sometimes displays as a PCI Device in the Other devices group.

Driver name	Hardware ID	Description
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	The network driver. Sometimes displays as an Ethernet Controller in the Other devices group. Available only if a VirtIO NIC is configured.

2.35.3. Adding VirtIO drivers container disk to a virtual machine

Container-native virtualization distributes VirtIO drivers for Microsoft Windows as a container disk, which is available from the [Red Hat Container Catalog](#). To install these drivers to a Windows virtual machine, attach the **cnv-tech-preview/virtio-win** container disk to the virtual machine as a SATA CD drive in the virtual machine configuration file.

Prerequisites

- Download the **cnv-tech-preview/virtio-win** container disk from the [Red Hat Container Catalog](#). This is not mandatory, because the container disk will be downloaded from the Red Hat registry if it not already present in the cluster, but it can reduce installation time.

Procedure

- Add the **cnv-tech-preview/virtio-win** container disk as a **cdrom** disk in the Windows virtual machine configuration file. The container disk will be downloaded from the registry if it is not already present in the cluster.

```
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2 1
      cdrom:
        bus: sata
  volumes:
    - containerDisk:
        image: cnv-tech-preview/virtio-win
        name: virtiocontainerdisk
```

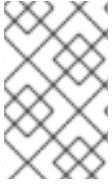
- 1** container-native virtualization boots virtual machine disks in the order defined in the **VirtualMachine** configuration file. You can either define other disks for the virtual machine before the **cnv-tech-preview/virtio-win** container disk or use the optional **bootOrder** parameter to ensure the virtual machine boots from the correct disk. If you specify the **bootOrder** for a disk, it must be specified for all disks in the configuration.

- The disk is available once the virtual machine has started:
 - If you add the container disk to a running virtual machine, use **oc apply -f <vm.yaml>** in the CLI or reboot the virtual machine for the changes to take effect.
 - If the virtual machine is not running, use **virtctl start <vm>**.

After the virtual machine has started, the VirtIO drivers can be installed from the attached SATA CD drive.

2.35.4. Installing VirtIO drivers during Windows installation

Install the VirtIO drivers from the attached SATA CD driver during Windows installation.



NOTE

This procedure uses a generic approach to the Windows installation and the installation method might differ between versions of Windows. Refer to the documentation for the version of Windows that you are installing.

Procedure

1. Start the virtual machine and connect to a graphical console.
2. Begin the Windows installation process.
3. Select the **Advanced** installation.
4. The storage destination will not be recognized until the driver is loaded. Click **Load driver**.
5. The drivers are attached as a SATA CD drive. Click **OK** and browse the CD drive for the storage driver to load. The drivers are arranged hierarchically according to their driver type, operating system, and CPU architecture.
6. Repeat the previous two steps for all required drivers.
7. Complete the Windows installation.

2.35.5. Removing the VirtIO container disk from a virtual machine

After installing all required VirtIO drivers to the virtual machine, the **cnv-tech-preview/virtio-win** container disk no longer needs to be attached to the virtual machine. Remove the **cnv-tech-preview/virtio-win** container disk from the virtual machine configuration file.

Procedure

1. Edit the configuration file and remove the **disk** and the **volume**.

```
$ oc edit vm <vm-name>

spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2
          cdrom:
            bus: sata
  volumes:
```

```
- containerDisk:
  image: cnv-tech-preview/virtio-win
  name: virtiocontainerdisk
```

2. Reboot the virtual machine for the changes to take effect.

2.36. VIEWING LOGS

2.36.1. Understanding logs

Logs are collected for OpenShift Container Platform Builds, Deployments, and Pods. In container-native virtualization, virtual machine logs can be retrieved from the virtual machine launcher Pod in either the web console or the CLI.

The **-f** option follows the log output in real time, which is useful for monitoring progress and error checking.

If the launcher Pod is failing to start, use the **--previous** option to see the logs of the last attempt.



WARNING

ErrImagePull and **ImagePullBackOff** errors can be caused by an incorrect Deployment configuration or problems with the images that are referenced.

2.36.2. Viewing virtual machine logs in the CLI

Get virtual machine logs from the virtual machine launcher Pod.

Procedure

- Use the following command:

```
$ oc logs <virt-launcher-name>
```

2.36.3. Viewing virtual machine logs in the web console

Get virtual machine logs from the associated virtual machine launcher Pod.

Procedure

1. In the container-native virtualization console, click **Workloads** → **Virtual Machines**.
2. Click the virtual machine to open the **Virtual Machine Details** panel.
3. In the **Overview** tab, click the **virt-launcher-<name>** Pod in the **POD** section.
4. Click **Logs**.

2.37. VIEWING EVENTS

2.37.1. Understanding events

OpenShift Container Platform events are records of important life-cycle information in a namespace and are useful for monitoring and troubleshooting resource scheduling, creation, and deletion issues.

Container-native virtualization adds events for virtual machines and virtual machine instances. These can be viewed from either the web console or the CLI.

See also: [Viewing system event information in an OpenShift Container Platform cluster](#) .

2.37.2. Viewing the events for a virtual machine in the web console

You can view the stream events for a running a virtual machine from the **Virtual Machine Details** panel of the web console.

The **||** button pauses the events stream.

The **▶** button continues a paused events stream.

Procedure

1. Click **Workloads** → **Virtual Machines** from the side menu.
2. Select a Virtual Machine.
3. Click **Events** to view all events for the virtual machine.

2.37.3. Viewing namespace events in the CLI

Use the OpenShift Container Platform client to get the events for a namespace.

Procedure

- In the namespace, use the **oc get** command:

```
$ oc get events
```

2.37.4. Viewing resource events in the CLI

Events are included in the resource description, which you can get using the OpenShift Container Platform client.

Procedure

- In the namespace, use the **oc describe** command. The following example shows how to get the events for a virtual machine, a virtual machine instance, and the virt-launcher Pod for a virtual machine:

```
$ oc describe vm <vm>
$ oc describe vmi <vmi>
$ oc describe pod virt-launcher-<name>
```

2.38. OPENSIFT CONTAINER PLATFORM CLUSTER MONITORING, LOGGING, AND TELEMETRY

OpenShift Container Platform provides various resources for monitoring at the cluster level.

2.38.1. About OpenShift Container Platform cluster monitoring

OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that is based on the [Prometheus](#) open source project and its wider eco-system. It provides monitoring of cluster components and includes a set of alerts to immediately notify the cluster administrator about any occurring problems and a set of [Grafana](#) dashboards. The cluster monitoring stack is only supported for monitoring OpenShift Container Platform clusters.



IMPORTANT

To ensure compatibility with future OpenShift Container Platform updates, configuring only the specified monitoring stack options is supported.

2.38.2. About cluster logging

The cluster logging components are based upon Elasticsearch, Fluentd, and Kibana (EFK). The collector, [Fluentd](#), is deployed to each node in the OpenShift Container Platform cluster. It collects all node and container logs and writes them to [Elasticsearch](#) (ES). [Kibana](#) is the centralized, web UI where users and administrators can create rich visualizations and dashboards with the aggregated data.

For more information on cluster logging, see the [OpenShift Container Platform cluster logging](#) documentation.

2.38.3. About Telemetry

Telemetry collects anonymized aggregated information about:

- The size of an OpenShift Container Platform cluster
- The health and status of OpenShift Container Platform components
- Use of OpenShift Container Platform components
- The features in use

This information is used by Red Hat to help make OpenShift Container Platform better and more intuitive to use. None of the information is shared with third parties.

2.38.3.1. What information is collected

Telemetry does not collect, and will never collect, identifying information like user names, passwords, or the names or addresses of user resources.

Primary information collected includes:

- Number of updates available per cluster
- Channel and image repository used for an update
- Number of errors that occurred during an update

- Progress information of an update that is running
- Number of machines per cluster
- Number of CPU cores and size of RAM of the machines
- Number of members in the etcd cluster and number of objects currently stored in the etcd cluster
- Number of CPU cores and RAM used per machine type - infra or master
- Number of CPU cores and RAM used per cluster
- Use of OpenShift Container Platform framework components per cluster
- Version of the OpenShift Container Platform cluster
- Health, condition, and status for any OpenShift Container Platform framework component that is installed on the cluster, for example Cluster Version Operator, Cluster Monitoring, Image Registry, and Elasticsearch for Logging
- A unique random identifier that is generated during installation
- Name of the platform OpenShift Container Platform is deployed on, such as Amazon Web Services

2.38.4. CLI troubleshooting and debugging commands

For a list of the **oc** client troubleshooting and debugging commands, see the [OpenShift Container Platform CLI reference](#) documentation.

CHAPTER 3. CONTAINER-NATIVE VIRTUALIZATION 2.0 RELEASE NOTES

3.1. CONTAINER-NATIVE VIRTUALIZATION 2.0 RELEASE NOTES

3.1.1. About container-native virtualization

3.1.1.1. What you can do with container-native virtualization

Container-native virtualization is an add-on to OpenShift Container Platform that allows you to run and manage virtual machine workloads alongside container workloads.

Container-native virtualization adds new objects into your OpenShift Container Platform cluster via Kubernetes custom resources to enable virtualization tasks. These tasks include:

- Creating and managing Linux and Windows virtual machines
- Connecting to virtual machines through a variety of consoles and CLI tools
- Importing and cloning existing virtual machines, including VMware virtual machines
- Managing network interface controllers and storage disks attached to virtual machines
- Live migrating virtual machines between nodes

An enhanced web console provides a graphical portal to manage these virtualized resources alongside the OpenShift Container Platform cluster containers and infrastructure.

3.1.1.2. Container-native virtualization support



IMPORTANT

container-native virtualization is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see <https://access.redhat.com/support/offerings/techpreview/>.

3.1.2. New and changed features

3.1.2.1. Supported binding methods

- Open vSwitch (OVS) is no longer recommended and should not be used in container-native virtualization 2.0.
- For the default Pod network, **masquerade** is the only recommended binding method. It is not supported for non-default networks.
- For secondary networks, use the **bridge** binding method.

3.1.2.2. Web console improvements

- You can now view all services associated with a virtual machine in the **Virtual Machine Details** screen.

3.1.3. Resolved issues

- Deleting a PVC after a CDI import fails no longer results in the importer Pod getting stuck in a **CrashLoopBackOff** state. The PVC is now deleted normally. ([BZ#1673683](#))

3.1.4. Known issues

- Some KubeVirt resources are improperly retained when removing container-native virtualization. As a workaround, you must manually remove them by running the command **oc delete apiservices v1alpha3.subresources.kubevirt.io -n kubevirt-hyperconverged**. These resources will be removed automatically after ([BZ#1712429](#)) is resolved.
- When using an older version of **virtctl** with container-native virtualization 2.0, **virtctl** cannot connect to the requested virtual machine. On the client, update the **virtctl** RPM package to the latest version to resolve this issue. ([BZ#1706108](#))
- Interfaces connected to the default Pod network lose connectivity after live migration. As a workaround, use an additional **multus**-backed network. ([BZ#1693532](#))
- Container-native virtualization cannot reliably identify node drains that are triggered by running either **oc adm drain** or **kubectrl drain**. Do not run these commands on the nodes of any clusters where container-native virtualization is deployed. The nodes might not drain if there are virtual machines running on top of them. The current solution is to put nodes into maintenance. ([BZ#1707427](#))
- If you create a virtual machine with the Pod network connected in **bridge** mode and use a **cloud-init** disk, the virtual machine will lose its network connectivity after being restarted. As a workaround, remove the **HWADDR** line in the file **/etc/sysconfig/network-scripts/ifcfg-eth0**. ([BZ#1708680](#))
- Masquerade does not currently work with CNV. Due to an upstream issue, you cannot connect a virtual machine to the default Pod network while in Masquerade mode. ([BZ#1725848](#))
- Creating a NIC with Masquerade in the wizard does not allow you to specify the **port** option. ([BZ#1725848](#))
- If a virtual machine uses guaranteed CPUs, it will not be scheduled, because the label **cpumanager=true** is not automatically set on nodes. As a workaround, remove the **CPUManager** entry from the **kubevirt-config** ConfigMap. Then, manually label the nodes with **cpumanager=true** before running virtual machines with guaranteed CPUs on your cluster. ([BZ#1718944](#))
- If you use the web console to create a virtual machine template that has the same name as an existing virtual machine, the operation fails and the message **Name is already used by another virtual machine** is displayed. As a workaround, create the template from the command line. ([BZ#1717930](#))
- ReadWriteMany (RWX) is the only supported storage access mode for live migration, importing VMware virtual machines, and creating virtual machines by using the wizard. ([BZ#1724654](#))

