OpenShift Container Platform 3.11

Container-native Virtualization User’s Guide

Container-native Virtualization User's Guide

Last Updated: 2019-08-13

OpenShift Container Platform 3.11 Container-native Virtualization User's Guide

Container-native Virtualization User's Guide

Legal Notice

Abstract

Learn to use CNV

Table of Contents

CHAPTER 1. USING CONTAINER-NATIVE VIRTUALIZATION

1.1. PRODUCT OVERVIEW

1.1.1. Introduction to Container-native Virtualization

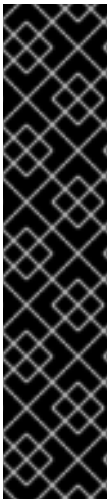
Container-native Virtualization is an add-on to OpenShift Container Platform that allows virtual machine workloads to run and be managed alongside container workloads. You can create virtual machines from disk images imported using the containerized data importer (CDI) controller, or from scratch within OpenShift Container Platform.

Container-native Virtualization introduces two new objects to OpenShift Container Platform:

- **Virtual Machine**: The virtual machine in OpenShift Container Platform
- **Virtual Machine Instance**: A running instance of the virtual machine

With the Container-native Virtualization add-on, virtual machines run in pods and have the same network and storage capabilities as standard pods.

Existing virtual machine disks are imported into persistent volumes (PVs), which are made accessible to Container-native Virtualization virtual machines using persistent volume claims (PVCs). In OpenShift Container Platform, the virtual machine object can be modified or replaced as needed, without affecting the persistent data stored on the PV.



IMPORTANT

Container-native Virtualization is currently a Technology Preview feature. For details about Red Hat support for Container-native Virtualization, see the Container-native Virtualization - Technology Preview Support Policy.

Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for 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 on Red Hat Technology Preview features support scope, see https://access.redhat.com/support/offerings/techpreview/.

1.2. WEB CONSOLE OPERATIONS

1.2.1. Managing virtual machines

1.2.1.1. Creating a virtual machine with the interactive wizard

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**.

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** > The **Results** screen displays the JSON configuration file for the virtual machine.

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

1.2.1.2. Creating a virtual machine using a YAML configuration file

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**.

1.2.1.3. Editing a virtual machine

You can edit some values of a virtual machine in the web console, either by editing the YAML directly, or from the **Virtual Machine Overview** screen.

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 updated values are shown after the operation is processed.

1.2.1.4. Editing the YAML of a virtual machine

You can edit the YAML configuration of a virtual machine directly within 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.
    a. (Optional) You can click **Download** to download the YAML file locally in its current state.
4. Edit the file and click **Save**.

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

1.2.1.5. Viewing the events of a virtual machine

You can view the stream events for a running a virtual machine from the **Virtual Machine Details** screen 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.

1.2.1.6. Deleting a virtual machine

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. Click **Workloads → Virtual Machines** from the side menu.
2. Click the ⋮ button of the virtual machine to delete and select **Delete Virtual Machine**
    a. 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.

1.2.2. Controlling virtual machines

1.2.2.1. Starting a virtual machine

Virtual machines can be started using the ⋮ button of each virtual machine in the **Workloads → Virtual Machines** list.

These same control operations can be done using the **Actions** control of the **Virtual Machine Details** screen.

Procedure

1. Click **Workloads → Virtual Machine** from the side menu.
2. Click the ⋮ button of the virtual machine and select **Start Virtual Machine**
   a. Alternatively, click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** and select **Start Virtual Machine**
3. In the confirmation pop-up window, click **Start** to start the virtual machine.

NOTE

When a virtual machine provisioned from a **URL** source is started for the first time, the virtual machine will be in the **Importing** state while Container-native Virtualization imports the container from the URL endpoint. This may take several minutes depending on the size of the image.

1.2.2.2. Stopping a virtual machine

A running virtual machines can be stopped using the ⋮ button of each virtual machine in the **Workloads → Virtual Machines** list.

These same control operations can be done using the **Actions** control of the **Virtual Machine Details** screen.

Procedure

1. Click **Workloads → Virtual Machine** from the side menu.
2. Click the ⋮ button of the virtual machine and select **Stop Virtual Machine**.
   a. Alternatively, click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** and select **Stop Virtual Machine**.
3. In the confirmation pop-up window, click **Stop** to stop the virtual machine.

1.2.2.3. Restarting a virtual machine

A running virtual machines can be restarted using the ⋮ button of each virtual machine in the **Workloads → Virtual Machines** list.

These same control operations can be done using the **Actions** control of the **Virtual Machine Details** screen.

IMPORTANT

Do not restart a virtual machine while it has a status of **Importing**. This will result in an error for the virtual machine and is a known issue.

Procedure

1. Click **Workloads → Virtual Machine** from the side menu.
2. Click the ⋮ button of the virtual machine and select **Restart Virtual Machine**
   a. Alternatively, click the virtual machine name to open the **Virtual Machine Details** screen and click **Actions** and select **Restart Virtual Machine**

3. In the confirmation pop-up window, click **Restart** to restart the virtual machine.

1.2.3. Accessing virtual machine consoles

1.2.3.1. 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** drop-down 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.

1.2.3.2. 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

1. Click **Workloads → Virtual Machines** from the side menu.
2. Select a virtual machine.
3. Click **Consoles**. The VNC console opens by default.

1.2.3.3. 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. Click **Workloads → Virtual Machines** from the side menu.
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**.

1.2.4. Managing virtual machine NICs

1.2.4.1. Creating a NIC for a virtual machine

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

Procedure

1. Click **Workloads → Virtual Machines** from the side menu.
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**, and **MAC ADDRESS** details for the new NIC.
6. Click the ✓ button to save and attach the NIC to the virtual machine.

1.2.4.2. Deleting a NIC from a virtual machine

Deleting a NIC from a virtual machine detaches and permanently deletes the NIC.

Procedure

1. Click **Workloads → Virtual Machines** from the side menu.
2. Select a virtual machine.
3. Click **Network Interfaces** to display the NICs already attached to the virtual machine.
4. Click the ⋮ button of the NIC you wish to delete and select **Delete**.
5. In the confirmation pop-up window, click **Delete** to detach and delete the NIC.

1.2.5. Managing virtual machine disks

1.2.5.1. Creating a disk for a virtual machine

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

Procedure

1. Click **Workloads → Virtual Machines** from the side menu.
2. Select a virtual machine.
3. Click **Disks** to display the disks already attached to the virtual machine.
4. Click **Create Disk** to create a new slot in the list.
5. Fill in the **NAME**, **SIZE**, and optional **STORAGE CLASS** details for the new disk.
6. Click the ✓ button to save and attach the disk to the virtual machine.

1.2.5.2. Deleting a disk from a virtual machine

Deleting a disk from a virtual machine detaches and permanently deletes the disk.

Procedure

1. Click **Workloads → Virtual Machines** from the side menu.
2. Select a virtual machine.
3. Click **Disks** to display the disks already attached to the virtual machine.
4. Click the ⋮ button of the disk you wish to delete and select **Delete**.
5. Click **Confirm** to detach and delete the disk.

1.2.6. Managing virtual machine templates

1.2.6.1. Creating a virtual machine template with the interactive wizard

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

The web console features an interactive wizard that guides you through **Basic Settings**, **Networking**, and **Storage** screens to simplify the process of creating virtual machine templates. 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 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**.

Procedure

1. Click **Workloads → Virtual Machine Templates** from the side menu.
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. An **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**. Virtual machines created from a template do not need a NIC attached. NICs can be created after a virtual machine has been created.

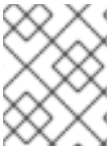
5. Click **Next** to progress to the Storage screen.
    a. (Optional) Click **Create Disk** to create additional disks. These disks can be removed by clicking the ⋮ button and selecting **Remove Disk**.
    b. (Optional) Click on a disk to modify available fields. Click the ✓ button to save the update.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

The template is listed in **Workloads → Virtual Machine Templates**.

1.2.6.2. Editing the YAML of a virtual machine template

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

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



NOTE

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

Procedure

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

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

1.2.6.3. Deleting a virtual machine template

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

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

Procedure

1. Click **Workloads → Virtual Machine Templates** from the side menu.
2. Click the ⋮ button of the template to delete and select **Delete Template**.

a. Alternatively, click the template name to open the **Virtual Machine Template Details** screen and click **Actions → Delete Template**.

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

1.3. CLI OPERATIONS

1.3.1. Before you begin

1.3.1.1. OpenShift Container Platform client commands

The **oc** client is a command-line utility for managing OpenShift Container Platform resources. The following table contains the **oc** commands that you use with Container-native Virtualization.

Table 1.1. oc commands

Command	Description
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.
oc create -f <config>	Create a resource from a filename or from stdin.
oc process -f <config>	Process a template into a configuration file. Templates have "parameters", which are either generated on creation or set by the user, as well as metadata describing the template.
oc apply -f <file>	Apply a configuration to a resource by filename or stdin.

See the OpenShift Container Platform CLI Reference Guide , or run the **oc --help** command, for definitive information on the OpenShift Container Platform client.

1.3.1.2. Virtctl 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 1.2. Virtctl client

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

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 disk from a client machine to the cluster.

1.3.1.3. Ensure correct OpenShift Container Platform project

Before you modify objects using the shell or web console, ensure you use the correct project. In the shell, use the following commands:

Command	Description
oc projects	List all available projects. The current project is marked with an asterisk.
oc project <project_name>	Switch to another project.
oc new-project <project_name>	Create a new project.

In the Web Console click the **Project** list and select the appropriate project or create a new one.

1.3.2. Configuring networking for virtual machines

1.3.2.1. Viewing guest IP addresses

You can view the IP addresses that are assigned to your Linux virtual machines by installing a QEMU guest agent on the virtual machine. When the guest agent is running, you can view the virtual machine IP addresses for each interface by checking the VMI status on the command line.

Procedure

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

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

2. Start the QEMU guest agent service:

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

3. View the IP address information for a VMI:

```
$ 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
```

NOTE

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

1.3.2.2. Configuring masquerade mode

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.

Procedure

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

```
kind: VM
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

1.3.3. Importing and uploading virtual machines and disk images

1.3.3.1. Uploading a local disk image to a new PVC

You can use **virtctl image-upload** to upload a virtual machine disk image from a client machine to your OpenShift Container Platform cluster. This creates a PVC that can be associated with a virtual machine after the upload has completed.

Prerequisites

- A virtual machine disk image, in RAW or QCOW2 format. It can be compressed using **xz** or **gzip**.
- **kubevirt-virtctl** must be installed on the client machine.
- The client machine must be configured to trust the OpenShift router's certificate.

Procedure

1. Identify the following items:
   - File location of the VM disk image you want to upload
   - Name and size desired for the resulting PVC
2. Remove the existing passthrough route:

```
$ oc delete route -n cdi cdi-uploadproxy-route
```

3. Create a secured route using re-encryption termination:

```
$ oc get secret -n cdi cdi-upload-proxy-ca-key -o=jsonpath="{.data['tls\.crt']}" | base64 -d > ca.pem

$ oc create route reencrypt cdi-uploadproxy-route -n cdi --service=cdi-uploadproxy --dest-ca-cert=ca.pem
```

4. Use the **virtctl image-upload** command to upload your VM image, making sure to include your chosen parameters. For example:

```
$ virtctl image-upload --uploadproxy-url=https://$(oc get route cdi-uploadproxy-route -n cdi -o=jsonpath='{.status.ingress[0].host}') --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.

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

```
$ oc get pvc
```

Next, you can create a virtual machine object to bind to the PVC.

1.3.3.2. Importing an existing virtual machine image with DataVolumes

DataVolume objects provide orchestration of import, clone, and upload operations associated with an underlying PVC. DataVolumes are integrated with KubeVirt and they can prevent a virtual machine from being started before the PVC has been prepared.

CAUTION

When you import a disk image into a PersistentVolumeClaim, 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

- The virtual machine disk can be RAW or QCOW2 format and can be compressed using **xz** or **gz**.
- The disk image must be available at either an **HTTP** or **S3** endpoint.

Procedure

1. Identify an **HTTP** or **S3** 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*
   - *s3://bucketName/fileName*
2. If your data source requires authentication credentials, edit the ***endpoint-secret.yaml*** file and apply it to the cluster:

```
apiVersion: v1
kind: Secret
metadata:
  name: <endpoint-secret>
  labels:
```

```
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: ""  # <optional: your key or user name, base64 encoded>
  secretKey:    "" # <optional: your secret or password, base64 encoded>
```

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

3. Edit the VM configuration file, optionally including the **secretRef** parameter. In our example, we used a Fedora image:

```
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
          secretRef: "" # Optional
    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: {}
```

4. Create the virtual machine:

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

The virtual machine and a DataVolume will now be created. The CDI controller creates an underlying PVC with the correct annotation and begins the import process. When the import completes, the DataVolume status changes to **Succeeded** and the virtual machine will be allowed to start.

DataVolume provisioning happens in the background, so there is no need to monitor it. You can start the VM 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 VirtualMachine 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>
```

1.3.3.3. Importing a virtual machine disk to a PVC

The process of importing a virtual machine disk is handled by the CDI controller. When a PVC is created with special **cdi.kubevirt.io/storage.import** annotations, the controller creates a short-lived import pod that attaches to the PV and downloads the virtual disk image into the PV.

You cannot import images from insecure registries using the Containerized Data Importer, even if you mark a registry as insecure with the **openshift_docker_insecure_registries** attribute in your OpenShift Container Platform inventory file.

CAUTION

When you import a disk image into a PersistentVolumeClaim, 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

- The virtual machine disk can be RAW or QCOW2 format and can be compressed using **xz** or **gzip**.
- The disk image must be available at either an **HTTP** or **S3** endpoint.

NOTE

For locally provisioned storage, the PV needs to be created before the PVC. This is not required for OpenShift Container Storage, for which the PVs are created dynamically.

Procedure

1. Identify an **HTTP** or **S3** file server hosting the virtual disk image that you want to import. You need the complete URL, in either format:
   - *http://www.example.com/path/to/data*
   - *s3://bucketName/fileName*
     Use this URL as the **cdi.kubevirt.io/storage.import.endpoint** annotation value in your PVC configuration file.

     For example: **cdi.kubevirt.io/storage.import.endpoint: https://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora-Cloud-Base-28-1.1.x86_64.qcow2**
2. If the file server requires authentication credentials, edit the ***endpoint-secret.yaml*** file:

```
apiVersion: v1
kind: Secret
metadata:
  name: endpoint-secret
  labels:
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: ""  # <optional: your key or user name, base64 encoded>
  secretKey:    "" # <optional: your secret or password, base64 encoded>
```

   a. Save the value of **metadata.name** to use with the **cdi.kubevirt.io/storage.import.secret** annotation in your PVC configuration file.
   For example: **cdi.kubevirt.io/storage.import.secret: endpoint-secret**
3. Apply ***endpoint-secret.yaml*** to the cluster:

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

4. Edit the PVC configuration file, making sure to include the required annotations. For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: "example-vmdisk-volume"
  labels:
    app: containerized-data-importer
  annotations:
    cdi.kubevirt.io/storage.import.endpoint:
"https://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora-Cloud-Base-28-1.1.x86_64.qcow2" (1)
    cdi.kubevirt.io/storage.import.secret: "endpoint-secret" (2)
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

(1) Endpoint annotation for the import image URL

(2) Endpoint annotation for the authorization secret

5. Create the PVC using the **oc** CLI:

```
$ oc create -f <pvc.yaml> (1)
```

(1) The PersistentVolumeClaim file name.

After the disk image is successfully imported into the PV, the import pod expires, and you can bind the PVC to a virtual machine object within OpenShift Container Platform.

Next, create a virtual machine object to bind to the PVC.

1.3.3.4. Cloning an existing PVC and creating a virtual machine using a dataVolumeTemplate

You can create a virtual machine that clones the 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 will also be deleted.

Prerequisites

- A PVC of an existing virtual machine disk. The associated virtual machine must be powered down, or the clone process will be queued until the PVC is available.

Procedure

1. Examine the DataVolume 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, **<vm-dv-clone>**, clones **<my-favorite-vm-disk>** (located in the **<source-namespace>** namespace) and creates the **2Gi <favorite-clone>** DataVolume, referenced in the virtual machine as the **<favorite-clone>** volume.
   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"
```

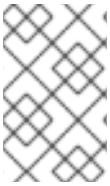
1 The virtual machine to create.

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

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

1.3.3.5. Cloning the PVC of an existing virtual machine disk

You can clone a 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 will be deleted.

Prerequisites

- A PVC of an existing virtual machine disk. The associated virtual machine must be powered down, or the clone process will be queued until the PVC is available.

Procedure

1. Examine the DataVolume 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 following parameters:

metadata: name	The name of the new DataVolume.
source: pvc: namespace	The namespace in which the source PVC exists.
source: pvc: name	The name of the source PVC.

storage	The size of the new DataVolume. Be sure to allocate enough space or the cloning operation fails. The size must be the same or larger as the source PVC.
	
	For example:
	
	`apiVersion: cdi.kubevirt.io/v1alpha1`
	`kind: DataVolume`
	`metadata:`
	`  name: cloner-datavolume`
	`spec:`
	`  source:`
	`  pvc:`
	`  namespace: "<source-namespace>"`
	`  name: "<my-favorite-vm-disk>"`
	`  pvc:`
	`  accessModes:`
	`  - ReadWriteOnce`
	`  resources:`
	`  requests:`
	`  storage: 2Gi`

3. Start the PVC clone by creating the DataVolume:

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

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.

1.3.4. Creating new blank disk images

1.3.4.1. Creating a blank disk image with a DataVolume manifest

You can use blank disks to increase your storage capacity or create new data partitions. You can create a new blank disk image in a **PersistentVolumeClaim** with a DataVolume manifest file.

Prerequisites

- Container-native Virtualization 1.4
- At least one available **PersistentVolume**

Procedure

1. Create the DataVolume manifest 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. Deploy the DataVolume manifest to create the blank disk image:

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

1.3.4.2. Creating a blank disk image with a PVC manifest

You can use blank disks to increase your storage capacity or create new data partitions. You can create a new blank disk image in a **PersistentVolumeClaim** with a PVC manifest file.

Prerequisites

- Container-native Virtualization 1.4
- At least one available **PersistentVolume**

Procedure

1. Create the PVC manifest file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: "blank-image-pvc"
  labels:
    app: containerized-data-importer
  annotations:
    cdi.kubevirt.io/storage.import.source: "none"
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 4Gi
  # Optional: Set the storage class or omit to accept the default
  # storageClassName: local
```

2. Deploy the PVC manifest to create the blank disk image:

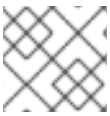
```
$ oc create -f blank-image-pvc.yaml
```

1.3.5. Managing virtual machines

1.3.5.1. Creating a new virtual machine from the CLI

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.

Attach the virtual machine disk to the virtual machine by referencing the relevant PVC **claimName** as a volume.



NOTE

ReplicaSet is not currently supported in Container-native Virtualization.

See the Reference section for information about volume types and sample configuration files.

Table 1.3. 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 allocated to the virtual machine by the node. Specify the denomination with **M** for Megabyte or **Gi** for Gigabyte.
disks: name	The Name of the volume which is referenced. Must match the name of a volume.

Table 1.4. Volume settings

Setting	Description
name	The Name of the volume. 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.

See the kubevirt API Reference for a definitive list of virtual machine settings.

To create a virtual machine with the OpenShift Container Platform client:

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

Virtual machines are created in a **Stopped** state. Run a virtual machine instance by starting it.

1.3.5.2. Deleting virtual machines and virtual machine PVCs

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, delete it, too.

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

```
$ oc delete vm fedora-vm
```

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

1.3.6. Controlling virtual machines

1.3.6.1. Controlling virtual machines

You can start, stop, or restart a virtual machine, depending on its current state.

Use the **virtctl** client utility to change the state of the virtual machine, open virtual console sessions with the virtual machines, and expose virtual machine ports as services.

The **virtctl** syntax is: **virtctl <action> <vm_name> <options>**

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

Examples:

```
$ virtctl start example-vm
```

```
$ virtctl restart example-vm
```

```
$ virtctl stop example-vm
```

oc get vm lists the virtual machines in the project. **oc get vmi** lists running virtual machine instances.

1.3.7. Accessing virtual machine consoles

1.3.7.1. Accessing the serial console of a VMI

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

Prerequisites

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

Procedure

1. Connect to the serial console with **virtctl**:

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

1.3.7.2. Accessing the graphical console of a VMI with VNC

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

Prerequisites

- **virt-viewer** 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 for this procedure to work.

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
```

1.3.7.3. Accessing a virtual machine instance via SSH

You can use SSH to access a virtual machine, but first you must expose port 22 on the VM.

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 which 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.

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
```

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
```

3. Log into the virtual machine instance via SSH, using the **ipAddress** of the node and the port that you found in Step 2:

```
$ ssh username@<node IP> -p 32551
```

1.3.8. Events, logs, errors, and metrics

1.3.8.1. Events

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

To retrieve the **events** for the project, run:

```
$ oc get events
```

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

```
$ oc describe <resource_type> <resource_name>
$ oc describe vm <fedora-vm>
$ oc describe vmi <fedora-vm>
$ oc describe pod virt-launcher-fedora-vm-<random>
```

Resource descriptions also include configuration, scheduling, and status details.

1.3.8.2. Logs

Logs are collected for OpenShift Container Platform builds, deployments, and pods. Virtual machine logs can be retrieved from the virtual machine launcher pod.

```
$ oc logs virt-launcher-fedora-vm-zzftf
```

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 being referenced.

1.3.8.3. Metrics

OpenShift Container Platform Metrics collects memory, CPU, and network performance information for nodes, components, and containers in the cluster. The specific information collected depends on how the Metrics subsystem is configured. For more information on configuring Metrics, see the OpenShift Container Platform Configuring Clusters Guide.

The **oc** CLI command **adm top** uses the Heapster API to fetch data about the current state of pods and nodes in the cluster.

To retrieve metrics for a pod:

```
$ oc adm top pod <pod_name>
```

To retrieve metrics for the nodes in the cluster:

```
$ oc adm top node
```

The OpenShift Container Platform web console can represent metric information graphically over a time range.

1.4. MANAGING VIRTIO DRIVERS FOR MICROSOFT WINDOWS VIRTUAL MACHINES

1.4.1. VirtIO drivers for Microsoft Windows virtual machines

VirtIO drivers are paravirtualized device drivers required for Microsoft Windows virtual machines to run properly in Container-native Virtualization. The supported drivers are available in the **virtio-win** container disk of the Red Hat Container Catalog.

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

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

1.4.2. Adding VirtIO drivers container disk to a virtual machine

Container-native Virtualization distributes VirtIO drivers for Microsoft Windows as a container disk, available from the Red Hat Container Catalog. To install or add these drivers to a Windows virtual machine, attach the **virtio-win** container disk to the virtual machine as a SATA CD drive.

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 machines disks in the order defined in the **VirtualMachine** configuration file. You can either define other disks for the virtual machine before the **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.

After the virtual machine has been created and started, the VirtIO drivers can be installed from the attached SATA CD drive during Windows installation on the virtual machine, or added to an existing Windows installation.

After the drivers have been installed, you can remove the **virtio-win** container disk from the virtual machine.

1.4.3. Installing VirtIO drivers during Windows installation

Install the VirtIO drivers during Windows installation to the virtual machine. At a minimum, you must install a supported storage driver to select the storage destination for the Windows installation.

Prerequisites

- Virtual machine with VirtIO drivers container disk attached as a SATA CD drive .
- Windows installation media accessible by the virtual machine.

NOTE

This procedure uses a generic approach to the Windows installation and the installation method may 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 recognised until the driver is loaded. Click **Load driver**.
5. The drivers are attached as a SATA CD driver. 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.

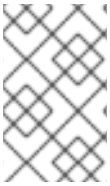
After Windows has been installed, you can remove the VirtIO driver container disk from the virtual machine configuration file.

1.4.4. Installing VirtIO drivers on an existing Windows virtual machine

Install VirtIO drivers on an existing Windows virtual machine.

Prerequisites

- Windows virtual machine with VirtIO drivers container disk attached as a SATA CD drive .



NOTE

This procedure uses a generic approach to adding drivers to Windows. The process may 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 may 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 drop-down 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 location of the driver. The drivers are arranged hierarchically according to their driver type, operating system, and CPU architecture.
6. Click **Next** to install the driver.
7. After the driver installs, click **Close** to close the window.
8. Reboot the virtual machine to complete the driver installation.

After the drivers have been installed, you can remove the disk from the virtual machine configuration file.

1.4.5. Removing the VirtIO container disk from a virtual machine

After you have installed all required VirtIO drivers to the virtual machine, the **virtio-win** container disk no longer needs to be attached to the virtual machine. Remove the **virtio-win** container disk from the virtual machine configuration file.

Procedure

1. Edit the virtual machine configuration in the web console, or edit the configuration file in your preferred editor, and remove the **disk** and the **volume**.

```
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.

1.5. ADVANCED VIRTUAL MACHINE CONFIGURATION

1.5.1. Using an Open vSwitch bridge as the network source for a VM

With Container-native Virtualization, you can connect a virtual machine instance to an Open vSwitch bridge that is configured on the node.

Prerequisites

- A cluster running OpenShift Container Platform 3.11 or newer

Procedure

1. Prepare the cluster host networks (optional).
   If the host network needs additional configuration changes, such as bonding, refer to the Red Hat Enterprise Linux networking guide.

2. Configure interfaces and bridges on all cluster hosts.
   On each node, choose an interface connected to the desired network. Then, create an Open vSwitch bridge and specify the interface you chose as the bridge's port.

   In this example, we create bridge **br1** and connect it to interface **eth1**. This bridge must be configured on all nodes. If it is only available on a subset of nodes, make sure that VMIs have **nodeSelector** constraints in place.

NOTE

```
Any connections to `eth1` are lost once the interface is
assigned to the bridge, so another interface must be present on the host.
```

```
$ ovs-vsctl add-br br1
$ ovs-vsctl add-port br1 eth1
$ ovs-vsctl show
8d004495-ea9a-44e1-b00c-3b65648dae5f
    Bridge br1
        Port br1
            Interface br1
                type: internal
        Port "eth1"
            Interface "eth1"
    ovs_version: "2.8.1"
```

3. Configure the network on the cluster.
   L2 networks are treated as cluster-wide resources. Define the network in a network attachment definition YAML file. You can define the network using the **NetworkAttachmentDefinition** CRD.

The **NetworkAttachmentDefinition** CRD object contains information about pod-to-network attachment. In the following example, there is an attachment to Open vSwitch bridge **br1** and traffic is tagged to VLAN 100.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vlan-100-net-conf
spec:
  config: '{
      "cniVersion": "0.3.1",
      "type": "ovs",
      "bridge": "br1",
      "vlan": 100
    }'
```

NOTE

"vlan" is optional. If omitted, the VMI will be attached through a trunk.

4. Edit the virtual machine instance configuration file to include the details of the interface and network.
   Specify that the network is connected to the previously created **NetworkAttachmentDefinition**. In this scenario, **vlan-100-net** is connected to the **NetworkAttachmentDefinition** called **vlan-100-net-conf**:

```
networks:
- name: default
  pod: {}
- name: vlan-100-net
  multus:
    networkName: vlan-100-net-conf
```

   After you start the VMI, the **eth0** interface connects to the default cluster network and **eth1** connects to VLAN 100 using bridge **br1** on the node running the VMI.

1.5.2. PXE booting with a specified MAC address

PXE booting, or network booting, is supported 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.

The Reference section has a configuration file template for PXE booting.

Prerequisites

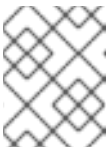
- A cluster running OpenShift Container Platform 3.11 or newer
- A configured interface that allows PXE booting

Procedure

1. Configure a PXE network on the cluster:

a. Create **NetworkAttachmentDefinition** of PXE network **pxe-net-conf**:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: pxe-net-conf
spec:
  config: '{
      "cniVersion": "0.3.1",
      "type": "ovs",
      "bridge": "br1"
    }'
```



NOTE

In this example, the VMI will be attached through a trunk port to the Open vSwitch bridge **<br1>**.

b. Create Open vSwitch bridge **<br1>** and connect it to interface **<eth1>**, which is connected to a network that allows for PXE booting:

```
$ ovs-vsctl add-br br1
$ ovs-vsctl add-port br1 eth1
$ ovs-vsctl show
8d004495-ea9a-44e1-b00c-3b65648dae5f
    Bridge br1
        Port br1
            Interface br1
                type: internal
        Port "eth1"
            Interface "eth1"
    ovs_version: "2.8.1"
```

NOTE

This bridge must be configured on all nodes. If it is only available on a subset of nodes, make sure that VMIs have **nodeSelector** constraints in place.

2. 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 OS 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
```

3. Create the virtual machine instance:

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

4. Wait for the virtual machine instance to run:

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

5. View the virtual machine instance using VNC:

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

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

7. Log in to the VMI:

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

8. Verify the interfaces and MAC address on the VM, 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
```

1.5.3. 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 leverage memory overcommitment to fit 8 VMs with 4 GB RAM each. This works under the assumption that the VMs will not use all of their memory at the same time.

Prerequisites

- A cluster running OpenShift Container Platform 3.11 or newer

Procedure

To explicitly tell the VMI that it has more memory available than what has been requested from the cluster, 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 VMI is told that it has **<2048M>** available. As long as there is enough free memory available on the node, the VMI 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 VMI if the node gets under memory pressure.

1.5.4. Disabling guest memory overhead accounting

WARNING

This procedure is only useful in certain use-cases and should 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 wrapping each **VirtualMachineInstance** process.

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

Prerequisites

- A cluster running OpenShift Container Platform 3.11 or newer

Procedure

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).

1.6. REFERENCE

1.6.1. Virtual machine wizard fields

Name	Parameter	Description
Name		Name of the virtual machine. Alphanumeric characters only, up to a maximum of 63.
Description		Optional description field.

Name	Parameter	Description
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***
Operating System		A list of operating systems available in the cluster. This is the primary operating system for the virtual machine.
Flavor	small, medium, large, tiny, Custom	Presets that determine the amount of CPU and memory allocated to the virtual machine.
Workload Profile	generic	A general configuration that balances performance and compatibility for a broad range of workloads.
	highperformance	The virtual machine has a more efficient configuration optimized for high performance loads.
Start virtual machine on creation		Select this checkbox to automatically start the virtual machine upon creation.
cloud-init		Select this checkbox to enable the cloud-init fields.

1.6.2. Virtual machine template wizard fields

Name	Parameter	Description
Name		Name of the virtual machine template. Alphanumeric characters only, up to a maximum of 63.
Description		Optional description field.
Provision Source	PXE	Provision virtual machines from PXE menu. Requires a PXE-capable NIC in the cluster.
	URL	Provision virtual machines from an image available from a **HTTP** or **S3** endpoint.
	Container	Provision virtual machines from a bootable operating system container located in a registry accessible from the cluster. Example: ***kubevirt/cirros-registry-disk-demo***
Operating System		A list of operating systems available in the cluster. This is the primary operating system for the virtual machine.
Flavor	small, medium, large, tiny, Custom	Presets that determine the amount of CPU and memory allocated to the virtual machines.
Workload Profile	generic	A general configuration that balances performance and compatibility for a broad range of workloads.
	highperformance	Virtual machines have a more efficient configuration optimized for high performance loads.
cloud-init		Select this checkbox to enable the cloud-init fields.

1.6.3. Cloud-init fields

Name	Description
Hostname	Sets a specific hostname for the virtual machine.
Authenticated SSH Keys	The user's public key. This will be copied to ***~/.ssh/authorized_keys*** on the virtual machine.
Use custom script	Replaces other options with a textbox into which you can paste a custom cloud-init script.

1.6.4. 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.
PXE NIC	List of PXE-capable networks. Only visible if **PXE** has been selected as the **Provision Source**.

1.6.5. 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.
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**.

1.6.6. Virtual machine actions

Table 1.5. Actions

Action	Available in state	Description
Start Virtual Machine	*Off*	Start the virtual machine.
Stop Virtual Machine	*Running* or *Other*	Stop the virtual machine.
Restart Virtual Machine	*Running* or *Other*	Restart the running virtual machine.
Delete Virtual Machine	*All*	Permanently delete the virtual machine from the cluster.

1.6.7. Supported VirtIO drivers for Microsoft Windows virtual machines in Container-native Virtualization

Table 1.6. Supported drivers

Driver name	Hardware ID	Description
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	The block driver. May represent as an **SCSI Controller** in the **Other devices** group.
viorng	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	The entropy source driver. May represent as an **PCI Device** in the **Other devices** group.
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	The network driver. May represent as an **Ethernet Controller** in the **Other devices** group. Available only if a VirtIO NIC is configured.

1.6.8. Types of storage volumes for virtual machines

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. This allows for the virtual machine data to persist between sessions. Importing an existing virtual machine disk into a PVC 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 using this volume type are guaranteed not to start until the volume is ready.
cloudInitNoCloud	Attaches a disk containing 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	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. 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. Only RAW and QCOW2 formats are supported disk types for the container image registry. QCOW2 is recommended for reduced image size.
emptyDisk	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. The disk **capacity** size must also be provided.

1.6.9. Template: PVC configuration file

pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: "example-vmdisk-volume"
  labels:
   app: containerized-data-importer
  annotations:
    kubevirt.io/storage.import.endpoint: "" # Required. Format: (http||s3)://www.myUrl.com/path/to/data
    kubevirt.io/storage.import.secretName: "" # Optional. The name of the secret containing credentials
for the data source
spec:
  accessModes:
  - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 5Gi
```

1.6.10. Template: VM configuration file

vm.yaml

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  creationTimestamp: null
  labels:
    kubevirt-vm: fedora-vm
  name: fedora-vm
spec:
  running: false
  template:
    metadata:
      creationTimestamp: null
      labels:
        kubevirt.io/domain: fedora-vm
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: containerdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
        machine:
          type: ""
        resources:
          requests:
            memory: 1Gi
      terminationGracePeriodSeconds: 0
      volumes:
      - cloudInitNoCloud:
          userData: |-
            #cloud-config
            password: fedora
            chpasswd: { expire: False }
        name: cloudinitdisk
      - name: containerdisk
        persistentVolumeClaim:
          claimName: example-vmdisk-volume
status: {}
```

1.6.11. Template: Windows VMI configuration file

windows-vmi.yaml

```
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
```

1.6.12. Template: VM configuration file (DataVolume)

example-vm-dv.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:
"https://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora-
Cloud-Base-28-1.1.x86_64.qcow2"
  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
```

1.6.13. 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:
"https://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora-
Cloud-Base-28-1.1.x86_64.qcow2" # Or S3
         secretRef: "" # Optional
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: "1G"
```

1.6.14. 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"
```

1.6.15. Template: VMI configuration file for PXE booting

vmi-pxe-boot.yaml

```
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: {}
```