OpenShift Container Platform 4.6

OpenShift Virtualization

OpenShift Virtualization installation, usage, and release notes
OpenShift Container Platform 4.6 OpenShift Virtualization

OpenShift Virtualization installation, usage, and release notes
Abstract

This document provides information about how to use OpenShift Virtualization in OpenShift Container Platform.
# Table of Contents

- **CHAPTER 1. ABOUT OPENSHIFT VIRTUALIZATION** .......................................................... 12
  - 1.1. WHAT YOU CAN DO WITH OPENSHIFT VIRTUALIZATION ............................... 12
    - 1.1.1. OpenShift Virtualization supported cluster version ................................. 12

- **CHAPTER 2. OPENSHIFT VIRTUALIZATION RELEASE NOTES** ........................................ 13
  - 2.1. ABOUT RED HAT OPENSHIFT VIRTUALIZATION ................................................. 13
    - 2.1.1. OpenShift Virtualization supported cluster version ................................. 13
  - 2.2. NEW AND CHANGED FEATURES ............................................................................. 13
    - 2.2.1. Networking ..................................................................................................... 14
    - 2.2.2. Storage ......................................................................................................... 14
    - 2.2.3. Web console .................................................................................................. 14
  - 2.3. NOTABLE TECHNICAL CHANGES ......................................................................... 15
  - 2.4. KNOWN ISSUES ................................................................................................. 15

- **CHAPTER 3. OPENSHIFT VIRTUALIZATION INSTALLATION** .......................................... 18
  - 3.1. CONFIGURING YOUR CLUSTER FOR OPENSHIFT VIRTUALIZATION .................. 18
  - 3.2. INSTALLING OPENSHIFT VIRTUALIZATION USING THE WEB CONSOLE ............ 18
    - 3.2.1. Prerequisites .................................................................................................... 19
    - 3.2.2. Subscribing to the OpenShift Virtualization catalog ..................................... 19
    - 3.2.3. Deploying OpenShift Virtualization .............................................................. 20
    - 3.2.4. Next steps ....................................................................................................... 20
    - 3.2.5. Prerequisites .................................................................................................... 20
    - 3.2.6. Subscribing to the OpenShift Virtualization catalog by using the CLI .......... 21
    - 3.2.7. Deploying the OpenShift Virtualization Operator by using the CLI .............. 21
    - 3.2.8. Next steps ....................................................................................................... 22
  - 3.3. INSTALLING THE VIRTCTL CLIENT ..................................................................... 22
    - 3.3.1. Installing the virtctl client from the web console .......................................... 23
    - 3.3.2. Enabling OpenShift Virtualization repositories ............................................. 23
    - 3.3.3. Installing the virtctl client ................................................................................. 24
    - 3.3.4. Additional resources ....................................................................................... 24
  - 3.4. UNINSTALLING OPENSHIFT VIRTUALIZATION USING THE WEB CONSOLE ....... 24
    - 3.4.1. Prerequisites .................................................................................................... 24
    - 3.4.2. Deleting the OpenShift Virtualization Operator Deployment custom resource .. 24
    - 3.4.3. Deleting the OpenShift Virtualization catalog subscription ............................ 25
    - 3.4.4. Deleting a namespace using the web console .................................................. 25
  - 3.5. UNINSTALLING OPENSHIFT VIRTUALIZATION USING THE CLI ..................... 26
    - 3.5.1. Prerequisites .................................................................................................... 26
    - 3.5.2. Deleting OpenShift Virtualization ................................................................. 26

- **CHAPTER 4. UPGRADING OPENSHIFT VIRTUALIZATION** ............................................ 28
  - 4.1. ABOUT UPGRADING OPENSHIFT VIRTUALIZATION ........................................ 28
    - 4.1.1. How OpenShift Virtualization upgrades work .............................................. 28
    - 4.1.2. How OpenShift Virtualization upgrades affect your cluster ........................... 28
  - 4.2. UPGRADE PATHWAYS FOR MINOR RELEASES ................................................. 28
    - 4.2.1. Upgrading from 2.4.3 to 2.5.8 ...................................................................... 29
    - 4.2.2. Upgrading from 2.4.4 or 2.4.5 to 2.5.8 .......................................................... 29
  - 4.3. UPGRADING OPENSHIFT VIRTUALIZATION TO THE NEXT MINOR VERSION ... 29
  - 4.4. MONITORING UPGRADE STATUS ..................................................................... 30
  - 4.5. ADDITIONAL RESOURCES ............................................................................... 30

**CHAPTER 5. ADDITIONAL SECURITY PRIVILEGES GRANTED FOR KUBEVIRT-CONTROLLER AND VIRT-**
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6.1. Starting a virtual machine</td>
<td>57</td>
</tr>
<tr>
<td>7.6.2. Restarting a virtual machine</td>
<td>58</td>
</tr>
<tr>
<td>7.6.3. Stopping a virtual machine</td>
<td>58</td>
</tr>
<tr>
<td>7.6.4. Unpausing a virtual machine</td>
<td>59</td>
</tr>
<tr>
<td>7.7. ACCESSING VIRTUAL MACHINE CONSOLES</td>
<td>59</td>
</tr>
<tr>
<td>7.7.1. About virtual machine console sessions</td>
<td>60</td>
</tr>
<tr>
<td>7.7.2. Connecting to the virtual machine with the web console</td>
<td>60</td>
</tr>
<tr>
<td>7.7.2.1. Connecting to the terminal</td>
<td>60</td>
</tr>
<tr>
<td>7.7.2.2. Connecting to the serial console</td>
<td>60</td>
</tr>
<tr>
<td>7.7.2.3. Connecting to the VNC console</td>
<td>61</td>
</tr>
<tr>
<td>7.7.2.4. Connecting to the RDP console</td>
<td>61</td>
</tr>
<tr>
<td>7.7.3. Accessing virtual machine consoles by using CLI commands</td>
<td>62</td>
</tr>
<tr>
<td>7.7.3.1. Accessing a virtual machine instance via SSH</td>
<td>62</td>
</tr>
<tr>
<td>7.7.3.2. Accessing the serial console of a virtual machine instance</td>
<td>63</td>
</tr>
<tr>
<td>7.7.3.3. Accessing the graphical console of a virtual machine instances with VNC</td>
<td>63</td>
</tr>
<tr>
<td>7.7.3.4. Connecting to a Windows virtual machine with an RDP console</td>
<td>63</td>
</tr>
<tr>
<td>7.8. TRIGGERING VIRTUAL MACHINE FAILOVER BY RESOLVING A FAILED NODE</td>
<td>65</td>
</tr>
<tr>
<td>7.8.1. Prerequisites</td>
<td>65</td>
</tr>
<tr>
<td>7.8.2. Deleting nodes from a bare metal cluster</td>
<td>65</td>
</tr>
<tr>
<td>7.8.3. Verifying virtual machine failover</td>
<td>66</td>
</tr>
<tr>
<td>7.8.3.1. Listing all virtual machine instances using the CLI</td>
<td>66</td>
</tr>
<tr>
<td>7.9. INSTALLING THE QEMU GUEST AGENT ON VIRTUAL MACHINES</td>
<td>66</td>
</tr>
<tr>
<td>7.9.1. Installing QEMU guest agent on a Linux virtual machine</td>
<td>66</td>
</tr>
<tr>
<td>7.9.2. Installing QEMU guest agent on a Windows virtual machine</td>
<td>67</td>
</tr>
<tr>
<td>7.9.2.1. Installing VirtIO drivers on an existing Windows virtual machine</td>
<td>67</td>
</tr>
<tr>
<td>7.9.2.2. Installing VirtIO drivers during Windows installation</td>
<td>67</td>
</tr>
<tr>
<td>7.10. VIEWING THE QEMU GUEST AGENT INFORMATION FOR VIRTUAL MACHINES</td>
<td>68</td>
</tr>
<tr>
<td>7.10.1. Prerequisites</td>
<td>68</td>
</tr>
<tr>
<td>7.10.2. About the QEMU guest agent information in the web console</td>
<td>68</td>
</tr>
<tr>
<td>7.10.3. Viewing the QEMU guest agent information in the web console</td>
<td>68</td>
</tr>
<tr>
<td>7.11. MANAGING CONFIG MAPS, SECRETS, AND SERVICE ACCOUNTS IN VIRTUAL MACHINES</td>
<td>69</td>
</tr>
<tr>
<td>7.11.1. Adding a secret, config map, or service account to a virtual machine</td>
<td>69</td>
</tr>
<tr>
<td>7.11.2. Removing a secret, config map, or service account from a virtual machine</td>
<td>70</td>
</tr>
<tr>
<td>7.11.3. Additional resources</td>
<td>71</td>
</tr>
<tr>
<td>7.12. INSTALLING VIRTIO DRIVER ON AN EXISTING WINDOWS VIRTUAL MACHINE</td>
<td>71</td>
</tr>
<tr>
<td>7.12.1. Understanding VirtIO drivers</td>
<td>71</td>
</tr>
<tr>
<td>7.12.2. Supported VirtIO drivers for Microsoft Windows virtual machines</td>
<td>71</td>
</tr>
<tr>
<td>7.12.3. Adding VirtIO drivers container disk to a virtual machine</td>
<td>72</td>
</tr>
<tr>
<td>7.12.4. Installing VirtIO drivers on an existing Windows virtual machine</td>
<td>73</td>
</tr>
<tr>
<td>7.12.5. Removing the VirtIO container disk from a virtual machine</td>
<td>73</td>
</tr>
<tr>
<td>7.13. INSTALLING VIRTIO DRIVER ON A NEW WINDOWS VIRTUAL MACHINE</td>
<td>74</td>
</tr>
<tr>
<td>7.13.1. Prerequisites</td>
<td>74</td>
</tr>
<tr>
<td>7.13.2. Understanding VirtIO drivers</td>
<td>74</td>
</tr>
<tr>
<td>7.13.3. Supported VirtIO drivers for Microsoft Windows virtual machines</td>
<td>74</td>
</tr>
<tr>
<td>7.13.4. Adding VirtIO drivers container disk to a virtual machine</td>
<td>75</td>
</tr>
<tr>
<td>7.13.5. Installing VirtIO drivers during Windows installation</td>
<td>76</td>
</tr>
<tr>
<td>7.13.6. Removing the VirtIO container disk from a virtual machine</td>
<td>76</td>
</tr>
<tr>
<td>7.14. ADVANCED VIRTUAL MACHINE MANAGEMENT</td>
<td>77</td>
</tr>
<tr>
<td>7.14.1. Automating management tasks</td>
<td>77</td>
</tr>
<tr>
<td>7.14.1.2. Automating virtual machine creation</td>
<td>77</td>
</tr>
<tr>
<td>7.14.1.3. Example: Ansible Playbook for creating virtual machines</td>
<td>78</td>
</tr>
<tr>
<td>7.14.2. Configuring PXE booting for virtual machines</td>
<td>79</td>
</tr>
</tbody>
</table>
7.14.2.1. Prerequisites
7.14.2.2. OpenShift Virtualization networking glossary
7.14.2.3. PXE booting with a specified MAC address
7.14.2.4. Template: Virtual machine instance configuration file for PXE booting
7.14.3. Managing guest memory
7.14.3.1. Configuring guest memory overcommitment
7.14.3.2. Disabling guest memory overhead accounting
7.14.4. Using huge pages with virtual machines
7.14.4.1. Prerequisites
7.14.4.2. What huge pages do
7.14.4.3. Configuring huge pages for virtual machines
7.14.5. Enabling dedicated resources for virtual machines
7.14.5.1. About dedicated resources
7.14.5.2. Prerequisites
7.14.5.3. Enabling dedicated resources for a virtual machine
7.14.6. Scheduling virtual machines
7.14.6.1. Understanding policy attributes
7.14.6.2. Setting a policy attribute and CPU feature
7.14.6.3. Scheduling virtual machines with the supported CPU model
7.14.6.4. Scheduling virtual machines with the host model
7.15. IMPORTING VIRTUAL MACHINES
7.15.1. TLS certificates for data volume imports
7.15.1.1. Adding TLS certificates for authenticating data volume imports
7.15.1.2. Example: Config map created from a TLS certificate
7.15.2. Importing virtual machine images with data volumes
7.15.2.1. Prerequisites
7.15.2.2. CDI supported operations matrix
7.15.2.3. About data volumes
7.15.2.4. Importing a virtual machine image into a persistent volume claim by using a data volume
7.15.3. Importing virtual machine images to block storage with data volumes
7.15.3.1. Prerequisites
7.15.3.2. About data volumes
7.15.3.3. About block persistent volumes
7.15.3.4. Creating a local block persistent volume
7.15.3.5. Importing a virtual machine image to a block persistent volume using data volumes
7.15.3.6. CDI supported operations matrix
7.15.4. Importing a single Red Hat Virtualization virtual machine
7.15.4.1. OpenShift Virtualization storage feature matrix
7.15.4.2. Prerequisites for importing a virtual machine
7.15.4.3. Importing a virtual machine with the VM Import wizard
    Virtual machine wizard fields
    Networking fields
    Storage fields
    Advanced storage settings
7.15.4.4. Importing a virtual machine with the CLI
7.15.4.4.1. Creating a config map for importing a VM
7.15.4.5. Troubleshooting a virtual machine import
7.15.4.5.1. Logs
7.15.4.5.2. Error messages
7.15.4.5.3. Known issues
7.15.5. Importing a single VMware virtual machine or template
7.15.5.1. OpenShift Virtualization storage feature matrix
7.15.5.2. Preparing a VDDK image
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>7.15.5.2.1. Configuring an internal image registry</td>
</tr>
<tr>
<td>113</td>
<td>Changing the image registry's management state</td>
</tr>
<tr>
<td>113</td>
<td>Configuring registry storage for bare metal and other manual installations</td>
</tr>
<tr>
<td>114</td>
<td>Accessing registry directly from the cluster</td>
</tr>
<tr>
<td>116</td>
<td>Exposing a secure registry manually</td>
</tr>
<tr>
<td>117</td>
<td>7.15.5.2.2. Configuring an external image registry</td>
</tr>
<tr>
<td>117</td>
<td>Adding certificate authorities to the cluster</td>
</tr>
<tr>
<td>118</td>
<td>Allowing pods to reference images from other secured registries</td>
</tr>
<tr>
<td>118</td>
<td>7.15.5.2.3. Creating and using a VDDK image</td>
</tr>
<tr>
<td>120</td>
<td>7.15.5.3. Importing a virtual machine with the VM Import wizard</td>
</tr>
<tr>
<td>122</td>
<td>Virtual machine wizard fields</td>
</tr>
<tr>
<td>124</td>
<td>Cloud-init fields</td>
</tr>
<tr>
<td>124</td>
<td>Networking fields</td>
</tr>
<tr>
<td>125</td>
<td>Storage fields</td>
</tr>
<tr>
<td>125</td>
<td>Advanced storage settings</td>
</tr>
<tr>
<td>126</td>
<td>7.15.5.3.1. Updating the NIC name of an imported virtual machine</td>
</tr>
<tr>
<td>126</td>
<td>7.15.5.4. Troubleshooting a virtual machine import</td>
</tr>
<tr>
<td>126</td>
<td>7.15.5.4.1. Logs</td>
</tr>
<tr>
<td>127</td>
<td>7.15.5.4.2. Error messages</td>
</tr>
<tr>
<td>127</td>
<td>7.16. CLONING VIRTUAL MACHINES</td>
</tr>
<tr>
<td>127</td>
<td>7.16.1. Enabling user permissions to clone data volumes across namespaces</td>
</tr>
<tr>
<td>127</td>
<td>7.16.1.1. Prerequisites</td>
</tr>
<tr>
<td>128</td>
<td>7.16.1.2. About data volumes</td>
</tr>
<tr>
<td>128</td>
<td>7.16.1.3. Creating RBAC resources for cloning data volumes</td>
</tr>
<tr>
<td>129</td>
<td>7.16.2. Cloning a virtual machine disk into a new data volume</td>
</tr>
<tr>
<td>129</td>
<td>7.16.2.1. Prerequisites</td>
</tr>
<tr>
<td>129</td>
<td>7.16.2.2. About data volumes</td>
</tr>
<tr>
<td>129</td>
<td>7.16.2.3. Cloning the persistent volume claim of a virtual machine disk into a new data volume</td>
</tr>
<tr>
<td>130</td>
<td>7.16.2.4. Template: Data volume clone configuration file</td>
</tr>
<tr>
<td>131</td>
<td>7.16.2.5. CDI supported operations matrix</td>
</tr>
<tr>
<td>131</td>
<td>7.16.3. Cloning a virtual machine by using a data volume template</td>
</tr>
<tr>
<td>132</td>
<td>7.16.3.1. Prerequisites</td>
</tr>
<tr>
<td>132</td>
<td>7.16.3.2. About data volumes</td>
</tr>
<tr>
<td>132</td>
<td>7.16.3.3. Creating a new virtual machine from a cloned persistent volume claim by using a data volume template</td>
</tr>
<tr>
<td>133</td>
<td>7.16.3.4. Template: Data volume virtual machine configuration file</td>
</tr>
<tr>
<td>134</td>
<td>7.16.3.5. CDI supported operations matrix</td>
</tr>
<tr>
<td>135</td>
<td>7.16.4. Cloning a virtual machine disk into a new block storage data volume</td>
</tr>
<tr>
<td>135</td>
<td>7.16.4.1. Prerequisites</td>
</tr>
<tr>
<td>135</td>
<td>7.16.4.2. About data volumes</td>
</tr>
<tr>
<td>135</td>
<td>7.16.4.3. About block persistent volumes</td>
</tr>
<tr>
<td>135</td>
<td>7.16.4.4. Creating a local block persistent volume</td>
</tr>
<tr>
<td>136</td>
<td>7.16.4.5. Cloning the persistent volume claim of a virtual machine disk into a new data volume</td>
</tr>
<tr>
<td>138</td>
<td>7.16.4.6. CDI supported operations matrix</td>
</tr>
<tr>
<td>139</td>
<td>7.17. VIRTUAL MACHINE NETWORKING</td>
</tr>
<tr>
<td>139</td>
<td>7.17.1. Using the default pod network for virtual machines</td>
</tr>
<tr>
<td>139</td>
<td>7.17.1.1. Configuring masquerade mode from the command line</td>
</tr>
<tr>
<td>140</td>
<td>7.17.1.2. Selecting binding method</td>
</tr>
<tr>
<td>140</td>
<td>7.17.1.2.1. Networking fields</td>
</tr>
<tr>
<td>140</td>
<td>7.17.1.3. Virtual machine configuration examples for the default network</td>
</tr>
<tr>
<td>140</td>
<td>7.17.1.3.1. Template: Virtual machine configuration file</td>
</tr>
<tr>
<td>141</td>
<td>7.17.1.3.2. Template: Windows virtual machine instance configuration file</td>
</tr>
<tr>
<td>142</td>
<td>7.17.1.4. Creating a service from a virtual machine</td>
</tr>
</tbody>
</table>
7.17.2. Attaching a virtual machine to multiple networks
  7.17.2.1. OpenShift Virtualization networking glossary
  7.17.2.2. Creating a network attachment definition
    7.17.2.2.1. Prerequisites
    7.17.2.2.2. Creating a Linux bridge network attachment definition in the web console
    7.17.2.2.3. Creating a Linux bridge network attachment definition in the CLI
  7.17.2.3. Attaching the virtual machine to the additional network
    7.17.2.3.1. Creating a NIC for a virtual machine in the web console
    7.17.2.3.2. Networking fields
    7.17.2.3.3. Attaching a virtual machine to a secondary network in the CLI
  7.17.2.4. Additional resources
  7.17.3. Configuring IP addresses for virtual machines
    7.17.3.1. Configuring an IP address for a new virtual machine using cloud-init
  7.17.4. Configuring an SR-IOV network device for virtual machines
    7.17.4.1. Prerequisites
    7.17.4.2. Automated discovery of SR-IOV network devices
      7.17.4.2.1. Example SrovNetworkNodeState object
      7.17.4.3. Configuring SR-IOV network devices
    7.17.4.4. Next steps
  7.17.5. Defining an SR-IOV network
    7.17.5.1. Prerequisites
    7.17.5.2. Configuring SR-IOV additional network
    7.17.5.3. Next steps
  7.17.6. Attaching a virtual machine to an SR-IOV network
    7.17.6.1. Prerequisites
    7.17.6.2. Attaching a virtual machine to an SR-IOV network
  7.17.7. Viewing the IP address of NICs on a virtual machine
    7.17.7.1. Viewing the IP address of a virtual machine interface in the CLI
    7.17.7.2. Viewing the IP address of a virtual machine interface in the web console
  7.17.8. Using a MAC address pool for virtual machines
    7.17.8.1. About KubeMacPool
    7.17.8.2. Enabling a MAC address pool for a namespace in the CLI
    7.17.8.3. Disabling a MAC address pool for a namespace in the CLI
  7.18. VIRTUAL MACHINE DISKS
  7.18.1. Storage features
    7.18.1.1. OpenShift Virtualization storage feature matrix
  7.18.2. Configuring local storage for virtual machines
    7.18.2.1. About the hostpath provisioner
    7.18.2.2. Configuring SELinux for the hostpath provisioner on Red Hat Enterprise Linux CoreOS (RHCOS) 8
  7.18.3. Configuring CDI to work with namespaces that have a compute resource quota
    7.18.3.1. About CPU and memory quotas in a namespace
    7.18.3.2. Editing the CDIConfig object to override CPU and memory defaults
    7.18.3.3. Additional resources
  7.18.4. Uploading local disk images by using the web console
    7.18.4.1. Prerequisites
    7.18.4.2. CDI supported operations matrix
    7.18.4.3. Uploading an image file using the web console
  7.18.5. Uploading local disk images by using the virtctl tool
    7.18.5.1. Prerequisites
    7.18.5.2. About data volumes
7.18.15. Re-using persistent volumes
7.18.15.1. About reclaiming statically provisioned persistent volumes
7.18.15.2. Reclaiming statically provisioned persistent volumes
7.18.16. Deleting data volumes
7.18.16.1. About data volumes
7.18.16.2. Listing all data volumes
7.18.16.3. Deleting a data volume

CHAPTER 8. VIRTUAL MACHINE TEMPLATES .................................................. 200
8.1. CREATING VIRTUAL MACHINE TEMPLATES ................................. 200
8.1.1. Creating a virtual machine template with the interactive wizard in the web console
8.1.2. Virtual machine template interactive wizard fields
  8.1.2.1. Virtual machine template wizard fields
  8.1.2.2. Cloud-init fields
  8.1.2.3. Networking fields
  8.1.2.4. Storage fields
    Advanced storage settings
8.2. EDITING VIRTUAL MACHINE TEMPLATES ................................... 204
8.2.1. Editing a virtual machine template in the web console
8.2.2. Editing virtual machine template YAML configuration in the web console
8.2.3. Adding a virtual disk to a virtual machine template
8.2.4. Adding a network interface to a virtual machine template
8.2.5. Editing CD-ROMs for Virtual Machine Templates
8.3. ENABLING DEDICATED RESOURCES FOR VIRTUAL MACHINE TEMPLATES
  8.3.1. About dedicated resources
  8.3.2. Prerequisites
  8.3.3. Enabling dedicated resources for a virtual machine template
8.4. DELETING A VIRTUAL MACHINE TEMPLATE .................................... 207
  8.4.1. Deleting a virtual machine template in the web console

CHAPTER 9. LIVE MIGRATION ................................................................. 208
9.1. VIRTUAL MACHINE LIVE MIGRATION .......................................... 208
  9.1.1. Understanding live migration
  9.1.2. Updating access mode for live migration
9.2. LIVE MIGRATION LIMITS AND TIMEOUTS .................................... 208
  9.2.1. Configuring live migration limits and timeouts
  9.2.2. Cluster-wide live migration limits and timeouts
9.3. MIGRATING A VIRTUAL MACHINE INSTANCE TO ANOTHER NODE .... 210
  9.3.1. Initiating live migration of a virtual machine instance in the web console
  9.3.2. Initiating live migration of a virtual machine instance in the CLI
9.4. MONITORING LIVE MIGRATION OF A VIRTUAL MACHINE INSTANCE .... 211
  9.4.1. Monitoring live migration of a virtual machine instance in the web console
  9.4.2. Monitoring live migration of a virtual machine instance in the CLI
9.5. CANCELLING THE LIVE MIGRATION OF A VIRTUAL MACHINE INSTANCE ...
  9.5.1. Cancelling live migration of a virtual machine instance in the web console
  9.5.2. Cancelling live migration of a virtual machine instance in the CLI
9.6. CONFIGURING VIRTUAL MACHINE EVICTION STRATEGY ............. 213
  9.6.1. Configuring custom virtual machines with the LiveMigration eviction strategy

CHAPTER 10. NODE MAINTENANCE ......................................................... 214
10.1. AUTOMATIC RENEWAL OF TLS CERTIFICATES .......................... 214
10.1.1. Automatic renewal of TLS certificates
10.2. MANAGING NODE LABELING FOR OBSOLETE CPU MODELS ......... 214
  10.2.1. Understanding node labeling for obsolete CPU models
10.2.2. Configuring a config map for obsolete CPU models
10.3. NODE MAINTENANCE MODE
  10.3.1. Understanding node maintenance mode
10.4. SETTINGS A NODE TO MAINTENANCE MODE
  10.4.1. Understanding node maintenance mode
  10.4.2. Setting a node to maintenance mode in the web console
  10.4.3. Setting a node to maintenance mode in the CLI
10.5. RESUMING A NODE FROM MAINTENANCE MODE
  10.5.1. Resuming a node from maintenance mode in the web console
  10.5.2. Resuming a node from maintenance mode in the CLI

CHAPTER 11. NODE NETWORKING ................................................................. 221
  11.1. OBSERVING NODE NETWORK STATE
    11.1.1. About nmstate
    11.1.2. Viewing the network state of a node
  11.2. UPDATE NODE NETWORK CONFIGURATION
    11.2.1. About nmstate
    11.2.2. Creating an interface on nodes
      Additional resources
    11.2.3. Confirming node network policy updates on nodes
    11.2.4. Removing an interface from nodes
    11.2.5. Example policy configurations for different interfaces
      11.2.5.1. Example: Linux bridge interface node network configuration policy
      11.2.5.2. Example: VLAN interface node network configuration policy
      11.2.5.3. Example: Bond interface node network configuration policy
      11.2.5.4. Example: Ethernet interface node network configuration policy
      11.2.5.5. Example: Multiple interfaces in the same node network configuration policy
    11.2.6. Examples: IP management
      11.2.6.1. Static
      11.2.6.2. No IP address
      11.2.6.3. Dynamic host configuration
      11.2.6.4. DNS
      11.2.6.5. Static routing
  11.3. TROUBLESHOOTING NODE NETWORK CONFIGURATION
    11.3.1. Troubleshooting an incorrect node network configuration policy configuration

CHAPTER 12. LOGGING, EVENTS, AND MONITORING ..................................... 238
  12.1. VIEWING VIRTUAL MACHINE Logs
    12.1.1. Understanding virtual machine logs
    12.1.2. Viewing virtual machine logs in the CLI
    12.1.3. Viewing virtual machine logs in the web console
  12.2. VIEWING EVENTS
    12.2.1. Understanding virtual machine events
    12.2.2. Viewing the events for a virtual machine in the web console
    12.2.3. Viewing namespace events in the CLI
    12.2.4. Viewing resource events in the CLI
  12.3. DIAGNOSING DATA VOLUMES USING EVENTS AND CONDITIONS
    12.3.1. About conditions and events
    12.3.2. Analyzing data volumes using conditions and events
  12.4. VIEWING INFORMATION ABOUT VIRTUAL MACHINE WORKLOADS
    12.4.1. About the Virtual Machines dashboard
  12.5. MONITORING VIRTUAL MACHINE HEALTH
    12.5.1. About readiness and liveness probes
12.5.2. Defining an HTTP readiness probe
12.5.3. Defining a TCP readiness probe
12.5.4. Defining an HTTP liveness probe
12.5.5. Template: Virtual machine instance configuration file for defining health checks
12.5.6. Additional resources

12.6. USING THE OPENSHIFT CONTAINER PLATFORM DASHBOARD TO GET CLUSTER INFORMATION
12.6.1. About the OpenShift Container Platform dashboards page

12.7. OPENSHIFT CONTAINER PLATFORM CLUSTER MONITORING, LOGGING, AND TELEMETRY
12.7.1. About OpenShift Container Platform monitoring
12.7.2. About cluster logging components
12.7.3. About Telemetry
   12.7.3.1. Information collected by Telemetry
12.7.4. CLI troubleshooting and debugging commands

12.8. COLLECTING OPENSHIFT VIRTUALIZATION DATA FOR RED HAT SUPPORT
12.8.1. About the must-gather tool
12.8.2. About collecting OpenShift Virtualization data
12.8.3. Gathering data about specific features
CHAPTER 1. ABOUT OPENSHIFT VIRTUALIZATION

Learn about OpenShift Virtualization’s capabilities and support scope.

1.1. WHAT YOU CAN DO WITH OPENSHIFT VIRTUALIZATION

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

OpenShift 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
- 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.

OpenShift Virtualization is tested with OpenShift Container Storage (OCS) and designed to use with OCS features for the best experience.

You can use OpenShift Virtualization with either the OVN-Kubernetes or the OpenShiftSDN default Container Network Interface (CNI) network provider.

1.1.1. OpenShift Virtualization supported cluster version

OpenShift Virtualization 2.5 is supported for use on OpenShift Container Platform 4.6 clusters.
CHAPTER 2. OPENSHIFT VIRTUALIZATION RELEASE NOTES

2.1. ABOUT RED HAT OPENSHIFT VIRTUALIZATION

Red Hat OpenShift Virtualization enables you to bring traditional virtual machines (VMs) into OpenShift Container Platform where they run alongside containers, and are managed as native Kubernetes objects.

OpenShift Virtualization is represented by the logo.

You can use OpenShift Virtualization with either the OVN-Kubernetes or the OpenShiftSDN default Container Network Interface (CNI) network provider.

Learn more about what you can do with OpenShift Virtualization.

2.1.1. OpenShift Virtualization supported cluster version

OpenShift Virtualization 2.5 is supported for use on OpenShift Container Platform 4.6 clusters.

2.1.2. Supported guest operating systems

OpenShift Virtualization guests can use the following operating systems:

- Red Hat Enterprise Linux 6, 7, and 8.
- Microsoft Windows 10.

Other operating system templates shipped with OpenShift Virtualization are not supported.

2.2. NEW AND CHANGED FEATURES

- OpenShift Virtualization is certified in Microsoft’s Windows Server Virtualization Validation Program (SVVP) to run Windows Server workloads. The SVVP Certification applies to:
  - Red Hat Enterprise Linux CoreOS 8 workers. In the Microsoft SVVP Catalog, they are named Red Hat OpenShift Container Platform 4 on RHEL CoreOS 8.
  - Intel and AMD CPUs.

- OpenShift Virtualization 2.5 adds three new virtctl commands to manage QEMU guest agent data:
  - `virtctl fslist <vmi_name>` returns a full list of file systems available on the guest machine.
  - `virtctl guestosinfo <vmi_name>` returns guest agent information about the operating system.
  - `virtctl userlist <vmi_name>` returns a full list of logged-in users on the guest machine.

- You can now download the virtctl client from the Command Line Tools page in the web console.
2.2.1. Networking

- The supported bond modes with nmstate now includes `mode=2 balance-xor` and `mode=4 802.3ad`.

2.2.2. Storage

- The Containerized Data Importer (CDI) can now import container disk storage volumes from the container image registry at a faster speed and allocate storage capacity more efficiently. CDI can pull a container disk image from the registry in about the same amount of time as it would take to import from an HTTP endpoint. You can import the disk into a persistent volume claim (PVC) equal in size to the disk image to use the underlying storage more efficiently.

- It is now easier to diagnose and troubleshoot issues when preparing virtual machine (VM) disks that are managed by DataVolumes:
  - For asynchronous image upload, if the virtual size of the disk image is larger than the size of the target DataVolume, an error message is returned before the connection is closed.
  - You can use the `oc describe dv` command to monitor changes in the PersistentVolumeClaim (PVC) Bound conditions or transfer failures. If the value of the Status:Phase field is Succeeded, then the DataVolume is ready to be used.

- You can create, restore, and delete virtual machine (VM) snapshots in the CLI for VMs that are powered off (offline). OpenShift Virtualization supports offline VM snapshots on:
  - Red Hat OpenShift Container Storage
  - Any other storage provider with the Container Storage Interface (CSI) driver that supports the Kubernetes Volume Snapshot API

- You can now clone virtual disks efficiently and quickly using smart-cloning. Smart-cloning occurs automatically when you create a DataVolume with a PersistentVolumeClaim (PVC) source. Your storage provider must support the CSI Snapshots API to use smart-cloning.

2.2.3. Web console

- If the virtual machine is running, changes made to the following fields and tabs in the web console will not take effect until you restart the virtual machine:
  - Boot Order and Flavor in the Details tab
  - The Network Interfaces tab
  - The Disks tab
  - The Environment tab
    The Pending Changes banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

- You can now open a virtual machine console in a separate window.

- You can now create default OS images and automatically upload them using the OpenShift
Container Platform web console. A default OS image is a bootable disk containing an operating system and all of the operating system’s configuration settings, such as drivers. You use a default OS image to create bootable virtual machines with specific configurations.

- You can now upload a virtual machine image file to a new persistent volume claim by using the web console.
- When the QEMU guest agent runs on the virtual machine, you can use the web console to view information about the virtual machine, users, file systems, and secondary networks.

### 2.3. NOTABLE TECHNICAL CHANGES

- When you install or upgrade OpenShift Virtualization, you select an Update Channel. There is a new Update Channel option that is named stable. Select the stable channel to ensure that you install or upgrade to the version of OpenShift Virtualization that is compatible with your OpenShift Container Platform version.
- You can now import VMs with block-based storage into OpenShift Virtualization.
- The HyperConverged Operator (HCO), Containerized Data Importer (CDI), Hostpath Provisioner (HPP), and VM import custom resources have moved to API version v1beta1. The respective API version for these components is now:
  - hco.kubevirt.io/v1beta1
  - cdi.kubevirt.io/v1beta1
  - hostpathprovisioner.kubevirt.io/v1beta1
  - v2v.kubevirt.io/v1beta1
- The default cloud-init user password is now auto-generated for virtual machines that are created from templates.
- When using host-assisted cloning, you can now clone virtual machine disks at a faster speed because of a more efficient compression algorithm.
- When a node fails in user-provisioned installations of OpenShift Container Platform on bare metal deployments, the virtual machine does not automatically restart on another node. Automatic restart is supported only for installer-provisioned installations that have machine health checks enabled. Learn more about configuring your cluster for OpenShift Virtualization.

### 2.4. KNOWN ISSUES

- If your OpenShift Container Platform cluster uses OVN-Kubernetes as the default Container Network Interface (CNI) provider, you cannot attach a Linux bridge or bonding to the default interface of a host because of a change in the host network topology of OVN-Kubernetes. As a workaround, you can use a secondary network interface connected to your host or switch to the OpenShift SDN default CNI provider. (BZ#1887456)
- If you add a VMware Virtual Disk Development Kit (VDDK) image to the openshift-cnv/v2v-vmware config map by using the web console, a Managed resource error message displays. You can safely ignore this error. Save the config map by clicking Save. (BZ#1884538)
- When nodes are evicted, for example, when they are placed in maintenance mode during an OpenShift Container Platform cluster upgrade, virtual machines are migrated twice instead of just once. (BZ#1888790)
- Following an upgrade, there might be more than one template per operating system workload. When creating a Microsoft Windows virtual machine from a cloned PVC using the default
operating system (OS) images feature, the OS must have the correct workload value defined. Selecting an incorrect Workload value does not allow you to use a default OS image, even though the (Source available) label displays in the web console. The default OS image is attached to the newer template but the wizard might use the old template, which is not configured to support default OS images. Windows 2010 systems only support a workload value of Desktop, while Windows 2012, Windows 2016, and Windows 2019 only support a workload value of Server. (BZ#1907183)

- If you enable a MAC address pool for a namespace by applying the KubeMacPool label and using the \texttt{io} attribute for virtual machines in that namespace, the \texttt{io} attribute configuration is not retained for the VMs. As a workaround, do not use the \texttt{io} attribute for VMs. Alternatively, you can disable KubeMacPool for the namespace. (BZ#1869527)

- If you upgrade to OpenShift Virtualization 2.5, both older and newer versions of common templates are available for each combination of operating system, workload, and flavor. When you create a virtual machine by using a common template, you must use the newer version of the template. Disregard the older version to avoid issues. (BZ#1859235)

- Running virtual machines that cannot be live migrated might block an OpenShift Container Platform cluster upgrade. This includes virtual machines that use hostpath-provisioner storage or SR-IOV network interfaces. (BZ#1858777)
  As a workaround, you can reconfigure the virtual machines so that they can be powered off during a cluster upgrade. In the \texttt{spec} section of the virtual machine configuration file:

  1. Remove the \texttt{evictionStrategy: LiveMigrate} field. See Configuring virtual machine eviction strategy for more information on how to configure eviction strategy.

  2. Set the \texttt{runStrategy} field to \texttt{Always}.

- For unknown reasons, memory consumption for the \texttt{containerDisk} volume type might gradually increase until it exceeds the memory limit. To resolve this issue, restart the VM. (BZ#1855067)

- Sometimes, when attempting to edit the subscription channel of the OpenShift Virtualization Operator in the web console, clicking the \texttt{Channel} button of the Subscription Overview results in a JavaScript error. (BZ#1796410)
  - As a workaround, trigger the upgrade process to OpenShift Virtualization 2.5 from the CLI by running the following \texttt{oc} patch command:

  ```bash
  $ export TARGET_NAMESPACE=openshift-cnv CNV_CHANNEL=2.5 && oc patch -n "$\{TARGET_NAMESPACE\}\" $(oc get subscription -n "$\{TARGET_NAMESPACE\}" --no-headers -o name) --type='json' -p='["op": "replace", "path": "/spec/channel", "value": "$\{CNV_CHANNEL\}\""}, {"op": "replace", "path": "/spec/installPlanApproval", "value": "Automatic"]'
  ```

  This command points your subscription to upgrade channel 2.5 and enables automatic updates.

- Live migration fails when nodes have different CPU models. Even in cases where nodes have the same physical CPU model, differences introduced by microcode updates have the same effect. This is because the default settings trigger host CPU passthrough behavior, which is incompatible with live migration. (BZ#1760028)
  - As a workaround, set the default CPU model in the \texttt{kubevirt-config} ConfigMap, as shown in the following example:
NOTE

You must make this change before starting the virtual machines that support live migration.

1. Open the **kubevirt-config** ConfigMap for editing by running the following command:

   ```bash
   $ oc edit configmap kubevirt-config -n openshift-cnv
   ```

2. Edit the ConfigMap:

   ```yaml
   kind: ConfigMap
   metadata:
     name: kubevirt-config
   data:
     default-cpu-model: "<cpu-model>"  
   ```

   Replace `<cpu-model>` with the actual CPU model value. You can determine this value by running `oc describe node <node>` for all nodes and looking at the `cpu-model-<name>` labels. Select the CPU model that is present on all of your nodes.

- OpenShift Virtualization cannot reliably identify node drains that are triggered by running either `oc adm drain` or `kubectl drain`. Do not run these commands on the nodes of any clusters where OpenShift 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**.

- If the OpenShift Virtualization storage PV is not suitable for importing a RHV VM, the progress bar remains at 10% and the import does not complete. The VM Import Controller Pod log displays the following error message: **Failed to bind volumes: provisioning failed for PVC.** *(BZ#1857784)*

- If you enter the wrong credentials for the RHV Manager while importing a RHV VM, the Manager might lock the admin user account because the **vm-import-operator** tries repeatedly to connect to the RHV API. *(BZ#1887140)*

  To unlock the account, log in to the Manager and enter the following command:

  ```bash
  $ ovirt-aaa-jdbc-tool user unlock admin
  ```

- If you are logged in to the OpenShift Container Platform cluster as a user with **basic-user** privileges, retrieving guest agent information by running `virtctl guestosinfo <vmi_name>` fails. As a workaround, you can fetch a subset of the guest agent data by running the `oc describe vmi` command. *(BZ#2000464)*
CHAPTER 3. OPENShift VIRTUALIZATION INSTALLATION

3.1. CONFIGURING YOUR CLUSTER FOR OPENSHEET VIRTUALIZATION

Before you install OpenShift Virtualization, ensure that your OpenShift Container Platform cluster meets the following requirements:

- Your cluster must be installed on **bare metal** infrastructure with Red Hat Enterprise Linux CoreOS workers.

- Additionally, your cluster must use **installer-provisioned infrastructure** and deploy machine **health checks** to maintain high availability (HA) of virtual machines.

  **NOTE**

  In OpenShift Virtualization clusters installed using installer-provisioned infrastructure and with MachineHealthCheck properly configured, if a node fails the MachineHealthCheck and becomes unavailable to the cluster, it is recycled. What happens next with VMs that ran on the failed node depends on a series of conditions. See *About RunStrategies for virtual machines* for more detailed information about the potential outcomes and how RunStrategies affect those outcomes.

- Shared storage is required to enable live migration.

- You must manage your Compute nodes according to the number and size of the virtual machines that you want to host in the cluster.

- To deploy OpenShift Virtualization in a disconnected environment, you must **configure Operator Lifecycle Manager on restricted networks**.

- When using a disconnected cluster on a restricted network, you must **configure proxy support in Operator Lifecycle Manager** to access the Red Hat-provided OperatorHub. Using a proxy allows the cluster to fetch the OpenShift Virtualization Operator.

- If your cluster uses worker nodes from multiple CPU vendors, live migration failures can occur. For example, a virtual machine with an AMD CPU might attempt to live-migrate to a node with an Intel CPU and likely fail migration. To avoid this, label nodes with a vendor-specific label, such as `Vendor=Intel` or `Vendor=AMD`, and set node affinity on your virtual machines to ensure successful migration. See *Configuring a required node affinity rule* for more information.

OpenShift Virtualization works with OpenShift Container Platform by default, but the following installation configurations are recommended:

- Configure **Monitoring** in the cluster.

  **NOTE**

  To obtain an evaluation version of OpenShift Container Platform, download a trial from the OpenShift Container Platform home page.

3.2. INSTALLING OPENSHEET VIRTUALIZATION USING THE WEB CONSOLE
Install OpenShift Virtualization to add virtualization functionality to your OpenShift Container Platform cluster.

You can use the OpenShift Container Platform 4.6 web console to subscribe to and deploy the OpenShift Virtualization Operators.

### 3.2.1. Prerequisites

- Install OpenShift Container Platform 4.6 on your cluster.
- Log in as a user with `cluster-admin` permissions.

### 3.2.2. Subscribing to the OpenShift Virtualization catalog

Before you install OpenShift Virtualization, subscribe to the OpenShift Virtualization catalog from the OpenShift Container Platform web console. Subscribing gives the `openshift-cnv` namespace access to the OpenShift Virtualization Operators.

**Procedure**

1. Open a browser window and log in to the OpenShift Container Platform web console.
2. Navigate to the **Operators → OperatorHub** page.
3. Search for **OpenShift Virtualization** and then select it.
4. Read the information about the Operator and click **Install**.
5. On the **Install Operator** page:
   
   a. For **Installed Namespace**, ensure that the **Operator recommended namespace** option is selected. This installs the Operator in the mandatory `openshift-cnv` namespace, which is automatically created if it does not exist.

   ![WARNING]
   
   Attempting to install the OpenShift Virtualization Operator in a namespace other than `openshift-cnv` causes the installation to fail.

   b. Select **stable** from the list of available **Update Channel** options. This ensures that you install the version of OpenShift Virtualization that is compatible with your OpenShift Container Platform version.

   c. For **Approval Strategy**, ensure that **Automatic**, which is the default value, is selected. OpenShift Virtualization automatically updates when a new z-stream release is available.

6. Click **Install** to make the Operator available to the `openshift-cnv` namespace.

   On the **Installed Operators** screen, the **Status** displays **Succeeded** when OpenShift Virtualization finishes installation.
3.2.3. Deploying OpenShift Virtualization

After subscribing to the OpenShift Virtualization catalog, create the OpenShift Virtualization Operator Deployment custom resource to deploy OpenShift Virtualization.

Prerequisites

- Subscribe to the OpenShift Virtualization catalog in the openshift-cnv namespace.

Procedure

1. Navigate to the Operators → Installed Operators page.
2. Click OpenShift Virtualization.
3. Click the OpenShift Virtualization Operator Deployment tab and click Create HyperConverged Cluster.

   WARNING

   To avoid deployment errors, do not rename the custom resource. Before you proceed to the next step, ensure that the custom resource is named the default kubevirt-hyperconverged.

4. Click Create to launch OpenShift Virtualization.
5. Navigate to the Workloads → Pods page and monitor the OpenShift Virtualization pods until they are all Running. After all the pods display the Running state, you can access OpenShift Virtualization.

3.2.4. Next steps

You might want to additionally configure the following components:

- The KubeMacPool component provides a MAC address pool service for virtual machine NICs in designated namespaces. Enable a MAC address pool in a namespace by applying the KubeMacPool label to that namespace.
- The hostpath provisioner is a local storage provisioner designed for OpenShift Virtualization. If you want to configure local storage for virtual machines, you must enable the hostpath provisioner first.

Install OpenShift Virtualization to add virtualization functionality to your OpenShift Container Platform cluster. You can subscribe to and deploy the OpenShift Virtualization Operators by using the command line to apply manifests to your cluster.

3.2.5. Prerequisites

- Install OpenShift Container Platform 4.6 on your cluster.
• Install the OpenShift CLI (`oc`).
• Log in as a user with `cluster-admin` privileges.

3.2.6. Subscribing to the OpenShift Virtualization catalog by using the CLI

Before you install OpenShift Virtualization, you must subscribe to the OpenShift Virtualization catalog. Subscribing gives the `openshift-cnv` namespace access to the OpenShift Virtualization Operators.

To subscribe, configure **Namespace**, **OperatorGroup**, and **Subscription** objects by applying a single manifest to your cluster.

**Procedure**

1. Create a YAML file that contains the following manifest:

   ```yaml
   apiVersion: v1
   kind: Namespace
   metadata:
     name: openshift-cnv
   ---
   apiVersion: operators.coreos.com/v1
   kind: OperatorGroup
   metadata:
     name: kubevirt-hyperconverged-group
     namespace: openshift-cnv
   spec:
     targetNamespaces:
     - openshift-cnv
   ---
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     name: hco-operatorhub
     namespace: openshift-cnv
   spec:
     source: redhat-operators
     sourceNamespace: openshift-marketplace
     name: kubevirt-hyperconverged
     startingCSV: kubevirt-hyperconverged-operator.v2.5.8
     channel: "stable" ①
   ```

   ① Using the `stable` channel ensures that you install the version of OpenShift Virtualization that is compatible with your OpenShift Container Platform version.

2. Create the required **Namespace**, **OperatorGroup**, and **Subscription** objects for OpenShift Virtualization by running the following command:

   ```bash
   $ oc apply -f <file name>.yaml
   ```

3.2.7. Deploying the OpenShift Virtualization Operator by using the CLI

You can deploy the OpenShift Virtualization Operator by using the `oc` CLI.
Prerequisites

- An active subscription to the OpenShift Virtualization catalog in the `openshift-cnv` namespace.

Procedure

1. Create a YAML file that contains the following manifest:

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

2. Deploy the OpenShift Virtualization Operator by running the following command:

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

Verification

- Ensure that OpenShift Virtualization deployed successfully by watching the PHASE of the cluster service version (CSV) in the `openshift-cnv` namespace. Run the following command:

```
$ watch oc get csv -n openshift-cnv
```

The following output displays if deployment was successful:

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>DISPLAY</th>
<th>VERSION</th>
<th>REPLACES</th>
<th>PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubevirt-hyperconverged-operator.v2.5.8</td>
<td>OpenShift Virtualization</td>
<td>2.5.8</td>
<td></td>
<td>Succeeded</td>
</tr>
</tbody>
</table>

3.2.8. Next steps

You might want to additionally configure the following components:

- The `KubeMacPool` component provides a MAC address pool service for virtual machine NICs in designated namespaces. Enable a MAC address pool in a namespace by applying the `KubeMacPool` label to that namespace.

- The `hostpath provisioner` is a local storage provisioner designed for OpenShift Virtualization. If you want to configure local storage for virtual machines, you must enable the hostpath provisioner first.

3.3. INSTALLING THE VIRTCTL CLIENT

The `virtctl` client is a command-line utility for managing OpenShift Virtualization resources. It is available for Linux, macOS, and Windows distributions.
You can install the `virtctl` client from the OpenShift Virtualization web console or by enabling the OpenShift Virtualization repository and installing the `kubevirt-virtctl` package.

### 3.3.1. Installing the virtctl client from the web console

You can download the `virtctl` client from the Red Hat Customer Portal, which is linked to in your OpenShift Virtualization web console in the **Command Line Tools** page.

**Prerequisites**

- You must have an activated OpenShift Container Platform subscription to access the download page on the Customer Portal.

**Procedure**

1. Access the Customer Portal by clicking the ![icon] icon, which is in the upper-right corner of the web console, and selecting **Command Line Tools**.

2. Ensure you have the appropriate version for your cluster selected from the **Version:** list.

3. Download the `virtctl` client for your distribution. All downloads are in **.tar.gz** format.

4. Extract the tarball. The following CLI command extracts it into the same directory as the tarball and is applicable for all distributions:

   ```
   $ tar -xvf <virtctl-version-distribution.arch>.tar.gz
   $ chmod +x <virtctl-file-name>
   $ echo $PATH
   ```

5. For Linux and macOS:
   a. Navigate the extracted folder hierarchy and make the `virtctl` binary executable:

   ```
   $ chmod +x <virtctl-file-name>
   ```
   b. Move the `virtctl` binary to a directory on your PATH.
      i. To check your path, run:

   ```
   $ echo $PATH
   ```

6. For Windows users:
   a. Navigate the extracted folder hierarchy and double-click the `virtctl` executable file to install the client.

### 3.3.2. Enabling OpenShift Virtualization repositories

Red Hat offers OpenShift Virtualization repositories for both Red Hat Enterprise Linux 8 and Red Hat Enterprise Linux 7:

- Red Hat Enterprise Linux 8 repository: `cnv-2.5-for-rhel-8-x86_64-rpms`
- Red Hat Enterprise Linux 7 repository: `rhel-7-server-cnv-2.5-rpms`

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

- Enable the appropriate OpenShift Virtualization repository for your system by running the following command:

  # subscription-manager repos --enable <repository>

3.3.3. Installing the virtctl client

Install the virtctl client from the kubevirt-virtctl package.

Procedure

- Install the kubevirt-virtctl package:

  # yum install kubevirt-virtctl

3.3.4. Additional resources

- Using the CLI tools for OpenShift Virtualization.

3.4. UNINSTALLING OPENSSHIFT VIRTUALIZATION USING THE WEB CONSOLE

You can uninstall OpenShift Virtualization by using the OpenShift Container Platform web console.

3.4.1. Prerequisites

- You must have OpenShift Virtualization 2.5 installed.
- You must delete all virtual machines, virtual machine instances, and data volumes.

**IMPORTANT**

Attempting to uninstall OpenShift Virtualization without deleting these objects results in failure.

3.4.2. Deleting the OpenShift Virtualization Operator Deployment custom resource

To uninstall OpenShift Virtualization, you must first delete the OpenShift Virtualization Operator Deployment custom resource.

Prerequisites

- Create the OpenShift Virtualization Operator Deployment custom resource.

Procedure

1. From the OpenShift Container Platform web console, select openshift-cnv from the Projects list.

2. Navigate to the Operators → Installed Operators page.
3. Click **OpenShift Virtualization**.

4. Click the **OpenShift Virtualization Operator Deployment** tab.

5. Click the Options menu in the row containing the **kubevirt-hyperconverged** custom resource. In the expanded menu, click **Delete HyperConverged Cluster**.

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 resources by running the following command:

   ```bash
   $ oc delete apiservices v1alpha3.subresources.kubevirt.io -n openshift-cnv
   ```

### 3.4.3. Deleting the OpenShift Virtualization catalog subscription

To finish uninstalling OpenShift Virtualization, delete the **OpenShift Virtualization** catalog subscription.

**Prerequisites**

- An active subscription to the **OpenShift Virtualization** catalog

**Procedure**

1. Navigate to the **Operators → OperatorHub** page.

2. Search for **OpenShift Virtualization** and then select it.

3. Click **Uninstall**.

   **NOTE**

   You can now delete the **openshift-cnv** namespace.

### 3.4.4. Deleting a namespace using the web console

You can delete a namespace by using the OpenShift Container Platform web console.

   **NOTE**

   If you do not have permissions to delete the namespace, the **Delete Namespace** option is not available.

**Procedure**

1. Navigate to **Administration → Namespaces**.

2. Locate the namespace that you want to delete in the list of namespaces.
3. On the far right side of the namespace listing, select **Delete Namespace** from the Options menu.

4. When the **Delete Namespace** pane opens, enter the name of the namespace that you want to delete in the field.

5. Click **Delete**.

### 3.5. Uninstalling OpenShift Virtualization Using the CLI

You can uninstall OpenShift Virtualization by using the OpenShift Container Platform CLI.

#### 3.5.1. Prerequisites

- You must have OpenShift Virtualization 2.5 installed.
- You must delete all virtual machines, virtual machine instances, and data volumes.

**IMPORTANT**

Attempting to uninstall OpenShift Virtualization without deleting these objects results in failure.

#### 3.5.2. Deleting OpenShift Virtualization

You can delete OpenShift Virtualization by using the CLI.

**Prerequisites**

- Install the OpenShift CLI (**oc**).
- Access to a OpenShift Virtualization cluster using an account with **cluster-admin** permissions.

**NOTE**

When you delete the subscription of the OpenShift Virtualization operator in the OLM by using the CLI, the **ClusterServiceVersion** (CSV) object is not deleted from the cluster. To completely uninstall OpenShift Virtualization, you must explicitly delete the CSV.

**Procedure**

1. Delete the **HyperConverged** custom resource:

   ```
   $ oc delete HyperConverged kubevirt-hyperconverged -n openshift-cnv
   ```

2. Delete the subscription of the OpenShift Virtualization operator in the Operator Lifecycle Manager (OLM):

   ```
   $ oc delete subscription kubevirt-hyperconverged -n openshift-cnv
   ```

3. Set the cluster service version (CSV) name for OpenShift Virtualization as an environment variable:
$ CSV_NAME=$(oc get csv -n openshift-cnv -o=custom-columns=:metadata.name)

4. Delete the CSV from the OpenShift Virtualization cluster by specifying the CSV name from the previous step:

$ oc delete csv $CSV_NAME -n openshift-cnv

OpenShift Virtualization is uninstalled when a confirmation message indicates that the CSV was deleted successfully:

**Example output**

clusterserviceversion.operators.coreos.com "kubevirt-hyperconverged-operator.v2.5.8" deleted
CHAPTER 4. UPGRADING OPENSIFT VIRTUALIZATION

You can manually upgrade to the next minor version of OpenShift Virtualization and monitor the status of an update by using the web console.

4.1. ABOUT UPGRADING OPENSIFT VIRTUALIZATION

4.1.1. How OpenShift Virtualization upgrades work

- You can upgrade to the next minor version of OpenShift Virtualization by using the OpenShift Container Platform web console to change the channel of your Operator subscription.
- You can enable automatic z-stream updates during OpenShift Virtualization installation.
- Updates are delivered via the Marketplace Operator, which is deployed during OpenShift Container Platform installation. The Marketplace Operator makes external Operators available to your cluster.
- The amount of time an update takes to complete depends on your network connection. Most automatic updates complete within fifteen minutes.

4.1.2. How OpenShift Virtualization upgrades affect your cluster

- Upgrading does not interrupt virtual machine workloads.
  - Virtual machine pods are not restarted or migrated during an upgrade. If you need to update the virt-launcher pod, you must restart or live migrate the virtual machine.

  **NOTE**
  Each virtual machine has a **virt-launcher** pod that runs the virtual machine instance. The **virt-launcher** pod runs an instance of **libvirt**, which is used to manage the virtual machine process.

- Upgrading does not interrupt network connections.
- Data volumes and their associated persistent volume claims are preserved during upgrade.

  **IMPORTANT**
  If you have virtual machines running that cannot be live migrated, they might block an OpenShift Container Platform cluster upgrade. This includes virtual machines that use hostpath provisioner storage or SR-IOV network interfaces.

  As a workaround, you can reconfigure the virtual machines so that they can be powered off automatically during a cluster upgrade. Remove the evictionStrategy: LiveMigrate field and set the runStrategy field to **Always**.

4.2. UPGRADE PATHWAYS FOR MINOR RELEASES

The upgrade pathway is different depending on the 2.4.z version of OpenShift Virtualization that you have installed.
4.2.1. Upgrading from 2.4.3 to 2.5.8

You must first upgrade to 2.5.0 before you can upgrade the z-stream. You can then upgrade from 2.5.0 to 2.5.1 and then to 2.5.2 and so on.

If your Approval Strategy is Automatic, which is the default, OpenShift Virtualization upgrades the z-stream automatically after you have upgraded to 2.5.0.

4.2.2. Upgrading from 2.4.4 or 2.4.5 to 2.5.8

You can upgrade from 2.4.4 or 2.4.5 directly to 2.5.2. You can then upgrade from 2.5.2 to 2.5.3 and so on.

If your Approval Strategy is Automatic, which is the default, OpenShift Virtualization upgrades the z-stream automatically after you have upgraded to 2.5.2.

4.3. UPGRADING OPENSHIFT VIRTUALIZATION TO THE NEXT MINOR VERSION

You can manually upgrade OpenShift Virtualization to the next minor version by using the OpenShift Container Platform web console to change the channel of your Operator subscription.

Prerequisites

- Log in to the cluster as a user with the cluster-admin role.

Procedure

1. Access the OpenShift Container Platform web console and navigate to Operators → Installed Operators.

2. Click OpenShift Virtualization to open the Operator Details page.

3. Click the Subscription tab to open the Subscription Overview page.

4. In the Channel pane, click the pencil icon on the right side of the version number to open the Change Subscription Update Channel window.

5. Select stable. This ensures that you install the version of OpenShift Virtualization that is compatible with your OpenShift Container Platform version.

6. Click Save.

7. Check the status of the upgrade by navigating to Operators → Installed Operators. You can also check the status by running the following oc command:

   `$ oc get csv -n openshift-cnv`
4.4. MONITORING UPGRADE STATUS

The best way to monitor OpenShift Virtualization upgrade status is to watch the cluster service version (CSV) **PHASE**. You can also monitor the CSV conditions in the web console or by running the command provided here.

**NOTE**

The **PHASE** and conditions values are approximations that are based on available information.

**Prerequisites**

- Log in to the cluster as a user with the `cluster-admin` role.
- Install the OpenShift CLI (`oc`).

**Procedure**

1. Run the following command:

   ```bash
   $ oc get csv
   ```

2. Review the output, checking the **PHASE** field. For example:

   **Example output**
   ```
   VERSION  REPLACES                                      PHASE
   2.5.0    kubevirt-hyperconverged-operator.v2.4.3       Installing
   2.4.3    Replacing
   ```

3. Optional: Monitor the aggregated status of all OpenShift Virtualization component conditions by running the following command:

   ```bash
   $ oc get hco -n openshift-cnv kubevirt-hyperconverged -o=jsonpath='{range .status.conditions[\*]}{.type}\t{.status}\t{.message}\t
   ```

   A successful upgrade results in the following output:
   ```
   ReconcileComplete True Reconcile completed successfully
   Available True Reconcile completed successfully
   Progressing False Reconcile completed successfully
   Degraded False Reconcile completed successfully
   Upgradeable True Reconcile completed successfully
   ```

4.5. ADDITIONAL RESOURCES

- Cluster service versions (CSVs)
- Configuring virtual machine eviction strategy
CHAPTER 5. ADDITIONAL SECURITY PRIVILEGES GRANTED FOR KUBEVIRT-CONTROLLER AND VIRT-LAUNCHER

The kubevirt-controller and virt-launcher pods are granted some SELinux policies and Security Context Constraints privileges that are in addition to typical pod owners. These privileges enable virtual machines to use OpenShift Virtualization features.

5.1. EXTENDED SELINUX POLICIES FOR VIRT-LAUNCHER PODS

The container_t SELinux policy for virt-launcher pods is extended with the following rules:

- allow process self (tun_socket (relabelfrom relabelto attach_queue))
- allow process sysfs_t (file (write))
- allow process hugetlbfs_t (dir (add_name create write remove_name rmdir setattr))
- allow process hugetlbfs_t (file (create unlink))

These rules enable the following virtualization features:

- Relabel and attach queues to its own TUN sockets, which is required to support network multi-queue. Multi-queue enables network performance to scale as the number of available vCPUs increases.
- Allows virt-launcher pods to write information to sysfs (/sys) files, which is required to enable Single Root I/O Virtualization (SR-IOV).
- Read/write hugetlbfs entries, which is required to support huge pages. Huge pages are a method of managing large amounts of memory by increasing the memory page size.

5.2. ADDITIONAL OPENSFiGHT CONTAINER PLATFORM SECURITY CONTEXT CONSTRAINTS AND LINUX CAPABILITIES FOR THE KUBEVIRT-CONTROLLER SERVICE ACCOUNT

Security context constraints (SCCs) control permissions for pods. These permissions include actions that a pod, a collection of containers, can perform and what resources it can access. You can use SCCs to define a set of conditions that a pod must run with in order to be accepted into the system.

The kubevirt-controller is a cluster controller that creates the virt-launcher pods for virtual machines in the cluster. These virt-launcher pods are granted permissions by the kubevirt-controller service account.

5.2.1. Additional SCCs granted to the kubevirt-controller service account

The kubevirt-controller service account is granted additional SCCs and Linux capabilities so that it can create virt-launcher pods with the appropriate permissions. These extended permissions allow virtual machines to take advantage of OpenShift Virtualization features that are beyond the scope of typical pods.

The kubevirt-controller service account is granted the following SCCs:

- scc.AllowHostDirVolumePlugin = true
  This allows virtual machines to use the hostpath volume plug-in.
- **scc.AllowPrivilegedContainer** = false
  
  This ensures the virt-launcher pod is not run as a privileged container.

- **scc.AllowedCapabilities** = [corev1.Capability("NET_ADMIN", "NET_RAW", "SYS_NICE")]
  
  This provides the following additional Linux capabilities NET_ADMIN, NET_RAW, and SYS_NICE.

5.2.2. Viewing the SCC and RBAC definitions for the kubevirt-controller

You can view the SecurityContextConstraints definition for the kubevirt-controller by using the oc tool:

```shell
$ oc get scc kubevirt-controller -o yaml
```

You can view the RBAC definition for the kubevirt-controller clusterrole by using the oc tool:

```shell
$ oc get clusterrole kubevirt-controller -o yaml
```

5.3. ADDITIONAL RESOURCES

- The Red Hat Enterprise Linux Virtualization Tuning and Optimization Guide has more information on network multi-queue and huge pages.

- The capabilities man page has more information on the Linux capabilities.

- The sysfs(5) man page has more information on sysfs.

- The OpenShift Container Platform Authentication guide has more information on Security Context Constraints.
CHAPTER 6. USING THE CLI TOOLS

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

- The OpenShift Virtualization **virtctl** client
- The OpenShift Container Platform **oc** client

6.1. PREREQUISITES

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

6.2. VRTCTL CLIENT COMMANDS

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

To view a list of options that you can use with a command, run it with the `-h` or `--help` flag. For example:

```
$ virtctl image-upload -h
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>virtctl start &lt;vm_name&gt;</strong></td>
<td>Start a virtual machine.</td>
</tr>
<tr>
<td><strong>virtctl stop &lt;vm_name&gt;</strong></td>
<td>Stop a virtual machine.</td>
</tr>
<tr>
<td>**virtctl pause vm</td>
<td>vmi &lt;object_name&gt;**</td>
</tr>
<tr>
<td>**virtctl unpause vm</td>
<td>vmi &lt;object_name&gt;**</td>
</tr>
<tr>
<td><strong>virtctl migrate &lt;vm_name&gt;</strong></td>
<td>Migrate a virtual machine.</td>
</tr>
<tr>
<td><strong>virtctl restart &lt;vm_name&gt;</strong></td>
<td>Restart a virtual machine.</td>
</tr>
<tr>
<td><strong>virtctl expose &lt;vm_name&gt;</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>virtctl console &lt;vmi_name&gt;</strong></td>
<td>Connect to a serial console of a virtual machine instance.</td>
</tr>
<tr>
<td><strong>virtctl vnc &lt;vmi_name&gt;</strong></td>
<td>Open a VNC connection to a virtual machine instance.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>virtctl image-upload dv</code>&lt;datavolume_name&gt; --image-path=&lt;path/to/image&gt; --no-create</td>
<td>Upload a virtual machine image to a data volume that already exists.</td>
</tr>
<tr>
<td><code>virtctl image-upload dv</code>&lt;datavolume_name&gt; --size=&lt;datavolume_size&gt; --image-path=&lt;path/to/image&gt;</td>
<td>Upload a virtual machine image to a new data volume.</td>
</tr>
<tr>
<td><code>virtctl version</code></td>
<td>Display the client and server version information.</td>
</tr>
<tr>
<td><code>virtctl help</code></td>
<td>Display a descriptive list of <code>virtctl</code> commands.</td>
</tr>
<tr>
<td><code>virtctl fslist &lt;vmi_name&gt;</code></td>
<td>Return a full list of file systems available on the guest machine.</td>
</tr>
<tr>
<td><code>virtctl guestosinfo &lt;vmi_name&gt;</code></td>
<td>Return guest agent information about the operating system.</td>
</tr>
<tr>
<td><code>virtctl userlist &lt;vmi_name&gt;</code></td>
<td>Return a full list of logged-in users on the guest machine.</td>
</tr>
</tbody>
</table>

### 6.3. OPENSHIFT CONTAINER PLATFORM CLIENT COMMANDS

The OpenShift Container Platform `oc` client is a command-line utility for managing OpenShift Container Platform resources, including the `VirtualMachine` (vm) and `VirtualMachineInstance` (vmi) object types.

**NOTE**

You can use the `-n <namespace>` flag to specify a different project.

Table 6.2. oc commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oc login -u &lt;user_name&gt;</code></td>
<td>Log in to the OpenShift Container Platform cluster as <code>&lt;user_name&gt;</code>.</td>
</tr>
<tr>
<td><code>oc get &lt;object_type&gt;</code></td>
<td>Display a list of objects for the specified object type in the current project.</td>
</tr>
<tr>
<td><code>oc describe &lt;object_type&gt;</code>&lt;resource_name&gt;</td>
<td>Display details of the specific resource in the current project.</td>
</tr>
<tr>
<td><code>oc create -f &lt;object_config&gt;</code></td>
<td>Create a resource in the current project from a file name or from stdin.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>oc edit &lt;object_type&gt; &lt;resource_name&gt;</code></td>
<td>Edit a resource in the current project.</td>
</tr>
<tr>
<td><code>oc delete &lt;object_type&gt; &lt;resource_name&gt;</code></td>
<td>Delete a resource in the current project.</td>
</tr>
</tbody>
</table>

For more comprehensive information on `oc` client commands, see the [OpenShift Container Platform CLI tools](#) documentation.
CHAPTER 7. VIRTUAL MACHINES

7.1. 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

**WARNING**

Do not create virtual machines in `openshift-*` namespaces. Instead, create a new namespace or use an existing namespace without the `openshift` prefix.

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

The web console features an interactive wizard that guides you through General, Networking, Storage, Advanced, and Review steps to simplify the process of creating virtual machines. All required fields are marked by a *. When the required fields are completed, you can review and create your virtual machine.

Network interface controllers (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 Source in the General step, 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 (RWX) PVCs.

**Procedure**

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Click Create Virtual Machine and select New with Wizard.
4. Fill in all required fields in the **General** step. Selecting a **Template** automatically fills in these fields.

5. Click **Next** to progress to the **Networking** step. A **nic0** NIC is attached by default.
   a. Optional: Click **Add Network Interface** to create additional NICs.

   b. Optional: You can remove any or all NICs by clicking the Options menu and selecting **Delete**. A virtual machine does not need a NIC attached to be created. NICs can be created after the virtual machine has been created.

6. Click **Next** to progress to the **Storage** screen.
   a. Optional: Click **Add Disk** to create additional disks. These disks can be removed by clicking the Options menu and selecting **Delete**.

   b. Optional: Click the Options menu to edit the disk and save your changes.

7. Click **Review and Create**. The **Results** screen displays the JSON configuration file for the virtual machine.

The virtual machine is listed in the **Virtual Machines** tab.

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

### 7.1.1.1. Virtual machine wizard fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td></td>
<td>Template from which to create the virtual machine. Selecting a template will automatically complete other fields.</td>
</tr>
<tr>
<td>Source</td>
<td>PXE</td>
<td>Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.</td>
</tr>
<tr>
<td></td>
<td>URL</td>
<td>Provision virtual machine from an image available from an HTTP or S3 endpoint.</td>
</tr>
<tr>
<td></td>
<td>Container</td>
<td>Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: <em>kubevirt/cirros-registry-disk-demo</em>.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Disk</td>
<td>Provision virtual machine from a disk.</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>The primary operating system that is selected for the virtual machine.</td>
<td></td>
</tr>
<tr>
<td>Flavor</td>
<td>small, medium, large, tiny, Custom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Presets that determine the amount of CPU and memory allocated to the virtual machine. The presets displayed for Flavor are determined by the operating system.</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>Size in GiB of the memory allocated to the virtual machine.</td>
<td></td>
</tr>
<tr>
<td>CPUs</td>
<td>The amount of CPU allocated to the virtual machine.</td>
<td></td>
</tr>
<tr>
<td>Workload Profile</td>
<td>High Performance</td>
<td>A virtual machine configuration that is optimized for high-performance workloads.</td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>A profile optimized to run server workloads.</td>
</tr>
<tr>
<td></td>
<td>Desktop</td>
<td>A virtual machine configuration for use on a desktop.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name can contain lowercase 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 uppercase letters, spaces, periods (.), or special characters.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional description field.</td>
<td></td>
</tr>
<tr>
<td>Start virtual machine on creation</td>
<td>Select to automatically start the virtual machine upon creation.</td>
<td></td>
</tr>
</tbody>
</table>

**7.1.1.2. Cloud-init fields**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Sets a specific hostname for the virtual machine.</td>
</tr>
<tr>
<td>Authenticated SSH Keys</td>
<td>The user’s public key that is copied to ~/.ssh/authorized_keys on the virtual machine.</td>
</tr>
<tr>
<td>Custom script</td>
<td>Replaces other options with a field in which you paste a custom cloud-init script.</td>
</tr>
</tbody>
</table>

### 7.1.1.3. CD-ROM fields

<table>
<thead>
<tr>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Specify the container path. For example: kubevirt/fedora-registry-disk: latest.</td>
</tr>
<tr>
<td>URL</td>
<td>Specify the URL path and size in GiB. Then, select the storage class for this URL from the drop-down list.</td>
</tr>
<tr>
<td>Attach Disk</td>
<td>Select the virtual machine disk that you want to attach.</td>
</tr>
</tbody>
</table>

### 7.1.1.4. Networking fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are e1000e and virtio.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
<tr>
<td>Type</td>
<td>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.</td>
</tr>
<tr>
<td>MAC Address</td>
<td>MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.</td>
</tr>
</tbody>
</table>

### 7.1.1.5. Storage fields
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Select a blank disk for the virtual machine or choose from the options available: URL, Container, Attach Cloned Disk, or Attach Disk. To select an existing disk and attach it to the virtual machine, choose Attach Cloned Disk or Attach Disk from a list of available persistent volume claims (PVCs).</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the disk. The name can contain lowercase 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 uppercase letters, spaces, or special characters.</td>
</tr>
<tr>
<td>Size (GiB)</td>
<td>Size, in GiB, of the disk.</td>
</tr>
<tr>
<td>Interface</td>
<td>Type of disk device. Supported interfaces are virtIO, SATA, and SCSI.</td>
</tr>
<tr>
<td>Storage Class</td>
<td>The storage class that is used to create the disk.</td>
</tr>
<tr>
<td>Advanced → Volume Mode</td>
<td>Defines whether the persistent volume uses a formatted file system or raw block state. Default is Filesystem.</td>
</tr>
<tr>
<td>Advanced → Access Mode</td>
<td>Access mode of the persistent volume. Supported access modes are ReadWriteOnce, ReadOnlyMany, and ReadWriteMany.</td>
</tr>
</tbody>
</table>

**Advanced storage settings**
The following advanced storage settings are available for Blank, Import via URL, and Clone existing PVC disks. These parameters are optional. If you do not specify these parameters, the system uses the default values from the kubevirt-storage-class-defaults config map.

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Mode</td>
<td>Filesystem</td>
<td>Stores the virtual disk on a file system-based volume.</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>Stores the virtual disk directly on the block volume. Only use Block if the underlying storage supports it.</td>
</tr>
<tr>
<td>Access Mode</td>
<td>Single User (RWO)</td>
<td>The disk can be mounted as read/write by a single node.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Shared Access (RWX)</td>
<td>The disk can be mounted as read/write by many nodes.</td>
<td></td>
</tr>
<tr>
<td>Read Only (ROX)</td>
<td>The disk can be mounted as read-only by many nodes.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

This is required for some features, such as live migration of virtual machines between nodes.

For more information on the `kubevirt-storage-class-defaults` config map, see Storage defaults for data volumes.

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

Create a virtual machine by writing or pasting a YAML configuration file. 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 → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Click **Create Virtual Machine** and select **New from YAML**.

4. 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.

5. Optional: Click **Download** to download the YAML configuration file in its present state.

6. Click **Create** to create the virtual machine.

The virtual machine is listed in the **Virtual Machines** tab.

### 7.1.3. Using the CLI to create a virtual machine

**Procedure**
The **spec** object of the virtual machine 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 OpenShift Virtualization.

<table>
<thead>
<tr>
<th>Table 7.1. Domain settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setting</strong></td>
</tr>
<tr>
<td>Cores</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Disks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7.2. Volume settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setting</strong></td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>PersistentVolumeClaim</td>
</tr>
</tbody>
</table>

### 7.1.4. Virtual machine storage volume types

<table>
<thead>
<tr>
<th>Storage volume type</th>
<th>Description</th>
</tr>
</thead>
</table>

### Storage volume type

<table>
<thead>
<tr>
<th>Storage volume type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ephemeral</td>
<td>A local copy-on-write (COW) image that uses a network volume as a read-only backing store. The backing volume must be a <code>PersistentVolumeClaim</code>. 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.</td>
</tr>
</tbody>
</table>
| 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                  | Data volumes 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.  
Specify `type: dataVolume` or `type: ""`. If you specify any other value for `type`, such as `persistentVolumeClaim`, a warning is displayed, and the virtual machine does not start. |
| 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

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 attached to the virtual machine as a disk when the virtual machine is launched.

A containerDisk volume is not limited to a single virtual machine and is 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.

NOTE

A containerDisk volume is ephemeral. It is discarded when the virtual machine is stopped, restarted, or deleted. A containerDisk volume is useful for read-only file systems such as CD-ROMs or for disposable virtual machines.

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.

7.1.5. About RunStrategies for virtual machines

A RunStrategy for virtual machines determines a virtual machine instance’s (VMI) behavior, depending on a series of conditions. The spec.runStrategy setting exists in the virtual machine configuration process as an alternative to the spec.running setting. The spec.runStrategy setting allows greater flexibility for how VMIs are created and managed, in contrast to the spec.running setting with only true or false responses. However, the two settings are mutually exclusive. Only either spec.running or spec.runStrategy can be used. An error occurs if both are used.
There are four defined RunStrategies.

**Always**

A VMI is always present when a virtual machine is created. A new VMI is created if the original stops for any reason, which is the same behavior as `spec.running: true`.

**RerunOnFailure**

A VMI is re-created if the previous instance fails due to an error. The instance is not re-created if the virtual machine stops successfully, such as when it shuts down.

**Manual**

The `start`, `stop`, and `restart` virtctl client commands can be used to control the VMI's state and existence.

**Halted**

No VMI is present when a virtual machine is created, which is the same behavior as `spec.running: false`.

Different combinations of the `start`, `stop` and `restart` virtctl commands affect which RunStrategy is used.

The following table follows a VM’s transition from different states. The first column shows the VM’s initial RunStrategy. Each additional column shows a virtctl command and the new RunStrategy after that command is run.

<table>
<thead>
<tr>
<th>Initial RunStrategy</th>
<th>start</th>
<th>stop</th>
<th>restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always</td>
<td>-</td>
<td>Halted</td>
<td>Always</td>
</tr>
<tr>
<td>RerunOnFailure</td>
<td>-</td>
<td>Halted</td>
<td>RerunOnFailure</td>
</tr>
<tr>
<td>Halted</td>
<td>Always</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**NOTE**

In OpenShift Virtualization clusters installed using installer-provisioned infrastructure, when a node fails the MachineHealthCheck and becomes unavailable to the cluster, VMs with a RunStrategy of `Always` or `RerunOnFailure` are rescheduled on a new node.

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
spec:
  RunStrategy: Always
  template:
...
```

1 The VMI’s current RunStrategy setting.

**7.1.6. Additional resources**
• The **VirtualMachineSpec** definition in the [KubeVirt v0.34.1 API Reference](https://kubevirt.io/v0.34.1) provides broader context for the parameters and hierarchy of the virtual machine specification.

**NOTE**

The KubeVirt API Reference is the upstream project reference and might contain parameters that are not supported in OpenShift Virtualization.

• **Prepare a container disk** before adding it to a virtual machine as a **containerDisk** volume.

• See [Deploying machine health checks](https://www.openshift.com/docs/) for further details on deploying and enabling machine health checks.

• See [Installer-provisioned infrastructure overview](https://www.openshift.com/docs/) for further details on installer-provisioned infrastructure.

### 7.2. EDITING VIRTUAL MACHINES

You can update a virtual machine configuration using either the YAML editor in the web console or the OpenShift client on the command line. You can also update a subset of the parameters in the **Virtual Machine Overview** of the web console.

#### 7.2.1. Editing a virtual machine in the web console

Edit select values of a virtual machine in the **Virtual Machine Overview** screen of the web console by clicking on the pencil icon next to the relevant field. Other values can be edited using the CLI.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.
2. Click the **Virtual Machines** tab.
3. Select a virtual machine to open the **Virtual Machine Overview** screen.
4. Click the **Details** tab.
5. Click the pencil icon to make a field editable.
6. Make the relevant changes and click **Save**.

**NOTE**

If the virtual machine is running, changes to **Boot Order** or **Flavor** will not take effect until you restart the virtual machine.

You can view pending changes by clicking **View Pending Changes** on the right side of the relevant field. The **Pending Changes** banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

#### 7.2.2. Editing a virtual machine YAML configuration using the web console

Using the web console, edit the YAML configuration of a virtual machine.
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 → Virtualization** from the side menu.
2. Click the **Virtual Machines** tab.
3. Select a virtual machine to open the **Virtual Machine Overview** screen.
4. Click the **YAML** tab to display the editable configuration.
5. Optional: You can click **Download** to download the YAML file locally in its current state.
6. 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.

### 7.2.3. Editing a virtual machine YAML configuration using the CLI

Use this procedure to edit a virtual machine YAML configuration using the CLI.

**Prerequisites**

- You configured a 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:

   ```bash
   $ oc edit <object_type> <object_ID>
   ```

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:
     ```bash
     $ oc apply <object_type> <object_ID>
     ```
7.2.4. Adding a virtual disk to a virtual machine

Use this procedure to add a virtual disk to a virtual machine.

Procedure

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Select a virtual machine to open the Virtual Machine Overview screen.
4. Click the Disks tab.
5. Click Add Disk to open the Add Disk window.
6. In the Add Disk window, specify the Source, Name, Size, Interface, Type, and Storage Class.
   a. Optional: In the Advanced list, specify the Volume Mode and Access Mode for the virtual disk. If you do not specify these parameters, the system uses the default values from the kubevirt-storage-class-defaults config map.
7. Click Add.

**NOTE**

If the virtual machine is running, the new disk is in the pending restart state and will not be attached until you restart the virtual machine.

The Pending Changes banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

For more information on the kubevirt-storage-class-defaults config map, see Storage defaults for data volumes.

### 7.2.4.1. Storage fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Select a blank disk for the virtual machine or choose from the options available: URL, Container, Attach Cloned Disk, or Attach Disk. To select an existing disk and attach it to the virtual machine, choose Attach Cloned Disk or Attach Disk from a list of available persistent volume claims (PVCs).</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the disk. The name can contain lowercase 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 uppercase letters, spaces, or special characters.</td>
</tr>
</tbody>
</table>
### size (GiB)
Size, in GiB, of the disk.

### Interface
Type of disk device. Supported interfaces are **virtIO**, **SATA**, and **SCSI**.

### Storage Class
The storage class that is used to create the disk.

### Advanced → Volume Mode
Defines whether the persistent volume uses a formatted file system or raw block state. Default is **Filesystem**.

### Advanced → Access Mode
Access mode of the persistent volume. Supported access modes are **ReadWriteOnce**, **ReadOnlyMany**, and **ReadWriteMany**.

**Advanced storage settings**
The following advanced storage settings are available for **Blank**, **Import via URL**, and **Clone existing PVC** disks. These parameters are optional. If you do not specify these parameters, the system uses the default values from the **kubevirt-storage-class-defaults** config map.

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Mode</td>
<td>Filesystem</td>
<td>Stores the virtual disk on a file system-based volume.</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>Stores the virtual disk directly on the block volume. Only use <strong>Block</strong> if the underlying storage supports it.</td>
</tr>
<tr>
<td>Access Mode</td>
<td>Single User (RWO)</td>
<td>The disk can be mounted as read/write by a single node.</td>
</tr>
<tr>
<td></td>
<td>Shared Access (RWX)</td>
<td>The disk can be mounted as read/write by many nodes.</td>
</tr>
<tr>
<td></td>
<td>Read Only (ROX)</td>
<td>The disk can be mounted as read-only by many nodes.</td>
</tr>
</tbody>
</table>

**NOTE**
This is required for some features, such as live migration of virtual machines between nodes.

### 7.2.5. Adding a network interface to a virtual machine
Use this procedure to add a network interface to a virtual machine.

**Procedure**
Procedure

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Select a virtual machine to open the Virtual Machine Overview screen.
4. Click the Network Interfaces tab.
5. Click Add Network Interface.
6. In the Add Network Interface window, specify the Name, Model, Network, Type, and MAC Address of the network interface.
7. Click Add.

**NOTE**

If the virtual machine is running, the new network interface is in the pending restart state and changes will not take effect until you restart the virtual machine.

The Pending Changes banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

### 7.2.5.1. Networking fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are <strong>e1000e</strong> and <strong>virtio</strong>.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
<tr>
<td>Type</td>
<td>List of available binding methods. For the default pod network, <strong>masquerade</strong> is the only recommended binding method. For secondary networks, use the <strong>bridge</strong> binding method. The <strong>masquerade</strong> method is not supported for non-default networks.</td>
</tr>
<tr>
<td>MAC Address</td>
<td>MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.</td>
</tr>
</tbody>
</table>

### 7.2.6. Editing CD-ROMs for Virtual Machines

Use the following procedure to configure CD-ROMs for virtual machines.

Procedure
1. From the Virtual Machines tab, select your virtual machine.

2. Select the Overview tab.

3. To add or edit a CD-ROM configuration, click the pencil icon to the right of the CD-ROMs label. The Edit CD-ROM window opens.
   - If CD-ROMs are unavailable for editing, the following message displays: The virtual machine doesn’t have any CD-ROMs attached.
   - If there are CD-ROMs available, you can remove a CD-ROM by clicking -.

4. In the Edit CD-ROM window, do the following:
   a. Select the type of CD-ROM configuration from the drop-down list for Media Type. CD-ROM configuration types are Container, URL, and Persistent Volume Claim.
   b. Complete the required information for each Type.
   c. When all CD-ROMs are added, click Save.

7.3. EDITING BOOT ORDER

You can update the values for a boot order list by using the web console or the CLI.

With Boot Order in the Virtual Machine Overview page, you can:

- Select a disk or network interface controller (NIC) and add it to the boot order list.
- Edit the order of the disks or NICs in the boot order list.
- Remove a disk or NIC from the boot order list, and return it back to the inventory of bootable sources.

7.3.1. Adding items to a boot order list in the web console

Add items to a boot order list by using the web console.

Procedure

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Select a virtual machine to open the Virtual Machine Overview screen.
4. Click the Details tab.
5. Click the pencil icon that is located on the right side of Boot Order. If a YAML configuration does not exist, or if this is the first time that you are creating a boot order list, the following message displays: No resource selected. VM will attempt to boot from disks by order of appearance in YAML file.
6. Click Add Source and select a bootable disk or network interface controller (NIC) for the virtual machine.
7. Add any additional disks or NICs to the boot order list.
8. Click **Save**.

**NOTE**

If the virtual machine is running, changes to **Boot Order** will not take effect until you restart the virtual machine.

You can view pending changes by clicking **View Pending Changes** on the right side of the **Boot Order** field. The **Pending Changes** banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

### 7.3.2. Editing a boot order list in the web console

Edit the boot order list in the web console.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** screen.

4. Click the **Details** tab.

5. Click the pencil icon that is located on the right side of **Boot Order**.

6. Choose the appropriate method to move the item in the boot order list:
   - If you do not use a screen reader, hover over the arrow icon next to the item that you want to move, drag the item up or down, and drop it in a location of your choice.
   - If you use a screen reader, press the Up Arrow key or Down Arrow key to move the item in the boot order list. Then, press the **Tab** key to drop the item in a location of your choice.

7. Click **Save**.

**NOTE**

If the virtual machine is running, changes to the boot order list will not take effect until you restart the virtual machine.

You can view pending changes by clicking **View Pending Changes** on the right side of the **Boot Order** field. The **Pending Changes** banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

### 7.3.3. Editing a boot order list in the YAML configuration file

Edit the boot order list in a YAML configuration file by using the CLI.

**Procedure**

1. Open the YAML configuration file for the virtual machine by running the following command:

   ```bash
   $ oc edit vm example
   ```
2. Edit the YAML file and modify the values for the boot order associated with a disk or network interface controller (NIC). For example:

```yaml
disks:
- bootOrder: 1
  disk:
    bus: virtio
    name: containerdisk
- disk:
    bootOrder: 2
    bus: virtio
    name: cloudinitdisk
- cdrom:
    bootOrder: 1
    bus: virtio
    name: cd-drive-1
interfaces:
- macAddress: '02:96:c4:00:00'
  masquerade: {}
  name: default
```

1. The boot order value specified for the disk.

2. The boot order value specified for the network interface controller.

3. Save the YAML file.

4. Click **reload the content** to apply the updated boot order values from the YAML file to the boot order list in the web console.

### 7.3.4. Removing items from a boot order list in the web console

Remove items from a boot order list by using the web console.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** screen.

4. Click the **Details** tab.

5. Click the pencil icon that is located on the right side of **Boot Order**.

6. Click the **Remove** icon next to the item. The item is removed from the boot order list and saved in the list of available boot sources. If you remove all items from the boot order list, the following message displays: **No resource selected. VM will attempt to boot from disks by order of appearance in YAML file.**
If the virtual machine is running, changes to **Boot Order** will not take effect until you restart the virtual machine.

You can view pending changes by clicking **View Pending Changes** on the right side of the **Boot Order** field. The **Pending Changes** banner at the top of the page displays a list of all changes that will be applied when the virtual machine restarts.

### 7.4. DELETING VIRTUAL MACHINES

You can delete a virtual machine from the web console or by using the `oc` command-line interface.

#### 7.4.1. Deleting a virtual machine using the web console

Deleting a virtual machine permanently removes it from the cluster.

**NOTE**

When you delete a virtual machine, the data volume it uses is automatically deleted.

**Procedure**

1. In the OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.
2. Click the **Virtual Machines** tab.
3. Click the ⋮ button of the virtual machine that you want to delete and select **Delete Virtual Machine**.
   - Alternatively, click the virtual machine name to open the **Virtual Machine Overview** screen and click **Actions → Delete Virtual Machine**
4. In the confirmation pop-up window, click **Delete** to permanently delete the virtual machine.

#### 7.4.2. Deleting a virtual machine by using the CLI

You can delete a virtual machine by using the `oc` command-line interface (CLI). The `oc` client enables you to perform actions on multiple virtual machines.

**NOTE**

When you delete a virtual machine, the data volume it uses is automatically deleted.

**Prerequisites**

- Identify the name of the virtual machine that you want to delete.

**Procedure**

- Delete the virtual machine by running the following command:

```
$ oc delete vm <vm_name>
```
NOTE

This command only deletes objects that exist in the current project. Specify the `-n <project_name>` option if the object you want to delete is in a different project or namespace.

7.5. MANAGING VIRTUAL MACHINE INSTANCES

If you have standalone virtual machine instances (VMIs) that were created independently outside of the OpenShift Virtualization environment, you can manage them by using the web console or the command-line interface (CLI).

7.5.1. About virtual machine instances

A virtual machine instance (VMI) is a representation of a running virtual machine (VM). When a VMI is owned by a VM or by another object, you manage it through its owner in the web console or by using the `oc` command-line interface (CLI).

A standalone VMI is created and started independently with a script, through automation, or by using other methods in the CLI. In your environment, you might have standalone VMIs that were developed and started outside of the OpenShift Virtualization environment. You can continue to manage those standalone VMIs by using the CLI. You can also use the web console for specific tasks associated with standalone VMIs:

- List standalone VMIs and their details.
- Edit labels and annotations for a standalone VMI.
- Delete a standalone VMI.

When you delete a VM, the associated VMI is automatically deleted. You delete a standalone VMI directly because it is not owned by VMs or other objects.

NOTE

Before you uninstall OpenShift Virtualization, list and view the standalone VMIs by using the CLI or the web console. Then, delete any outstanding VMIs.

7.5.2. Listing all virtual machine instances using the CLI

You can list all virtual machine instances (VMIs) in your cluster, including standalone VMIs and those owned by virtual machines, by using the `oc` command-line interface (CLI).

Procedure

- List all VMIs by running the following command:

  $ oc get vmis

7.5.3. Listing standalone virtual machine instances using the web console

Using the web console, you can list and view standalone virtual machine instances (VMIs) in your cluster that are not owned by virtual machines (VMs).
NOTE

VMIs that are owned by VMs or other objects are not displayed in the web console. The web console displays only standalone VMIs. If you want to list all VMIs in your cluster, you must use the CLI.

Procedure

- Click Workloads → Virtualization from the side menu. A list of VMs and standalone VMIs displays. You can identify standalone VMIs by the dark colored badges that display next to the virtual machine instance names.

7.5.4. Editing a standalone virtual machine instance using the web console

You can edit annotations and labels for a standalone virtual machine instance (VMI) using the web console. Other items displayed in the Details page for a standalone VMI are not editable.

Procedure

1. Click Workloads → Virtualization from the side menu. A list of virtual machines (VMs) and standalone VMIs displays.
2. Click the name of a standalone VMI to open the Virtual Machine Instance Overview screen.
3. Click the Details tab.
4. Click the pencil icon that is located on the right side of Annotations.
5. Make the relevant changes and click Save.

NOTE

To edit labels for a standalone VMI, click Actions and select Edit Labels. Make the relevant changes and click Save.

7.5.5. Deleting a standalone virtual machine instance using the CLI

You can delete a standalone virtual machine instance (VMI) by using the oc command-line interface (CLI).

Prerequisites

- Identify the name of the VMI that you want to delete.

Procedure

- Delete the VMI by running the following command:

  ```
  $ oc delete vmi <vmi_name>
  ```

7.5.6. Deleting a standalone virtual machine instance using the web console

Delete a standalone virtual machine instance (VMI) from the web console.
Procedure

1. In the OpenShift Container Platform web console, click Workloads → Virtualization from the side menu.

2. Click the ⋮ button of the standalone virtual machine instance (VMI) that you want to delete and select Delete Virtual Machine Instance
   - Alternatively, click the name of the standalone VMI. The Virtual Machine Instance Overview page displays.

3. Select Actions → Delete Virtual Machine Instance

4. In the confirmation pop-up window, click Delete to permanently delete the standalone VMI.

7.6. CONTROLLING VIRTUAL MACHINE STATES

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

NOTE
To control virtual machines from the command-line interface (CLI), use the virtctl client.

7.6.1. Starting a virtual machine

You can start a virtual machine from the web console.

Procedure

1. Click Workloads → Virtualization from the side menu.

2. Click the Virtual Machines tab.

3. Find the row that contains the virtual machine that you want to start.

4. Navigate to the appropriate menu for your use case:
   - To stay on this page, where you can perform actions on multiple virtual machines:
     a. Click the Options menu located at the far right end of the row.
   - To view comprehensive information about the selected virtual machine before you start it:
     a. Access the Virtual Machine Overview screen by clicking the name of the virtual machine.
     b. Click Actions.

5. Select Start Virtual Machine

6. 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 has a status of Importing while OpenShift Virtualization imports the container from the URL endpoint. Depending on the size of the image, this process might take several minutes.

7.6.2. Restarting a virtual machine

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

IMPORTANT

To avoid errors, do not restart a virtual machine while it has a status of Importing.

Procedure

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Find the row that contains the virtual machine that you want to restart.
4. Navigate to the appropriate menu for your use case:
   a. To stay on this page, where you can perform actions on multiple virtual machines:
      i. Click the Options menu located at the far right end of the row.
   b. To view comprehensive information about the selected virtual machine before you restart it:
      i. Access the Virtual Machine Overview screen by clicking the name of the virtual machine.
      ii. Click Actions.
5. Select Restart Virtual Machine
6. In the confirmation window, click Restart to restart the virtual machine.

7.6.3. Stopping a virtual machine

You can stop a virtual machine from the web console.

Procedure

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Find the row that contains the virtual machine that you want to stop.
4. Navigate to the appropriate menu for your use case:
   a. To stay on this page, where you can perform actions on multiple virtual machines:
a. Click the Options menu located at the far right end of the row.

- To view comprehensive information about the selected virtual machine before you stop it:
  a. Access the Virtual Machine Overview screen by clicking the name of the virtual machine.
  b. Click Actions.

5. Select Stop Virtual Machine

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

7.6.4. Unpausing a virtual machine

You can unpause a paused virtual machine from the web console.

Prerequisites

- At least one of your virtual machines must have a status of Paused.

   NOTE

   You can pause virtual machines by using the virtctl client.

Procedure

1. Click Workloads → Virtualization from the side menu.

2. Click the Virtual Machines tab.

3. Find the row that contains the virtual machine that you want to unpause.

4. Navigate to the appropriate menu for your use case:
   - To stay on this page, where you can perform actions on multiple virtual machines:
     a. In the Status column, click Paused.
   - To view comprehensive information about the selected virtual machine before you unpause it:
     a. Access the Virtual Machine Overview screen by clicking the name of the virtual machine.
     b. Click the pencil icon that is located on the right side of Status.

5. In the confirmation window, click Unpause to unpause the virtual machine.

7.7. ACCESSING VIRTUAL MACHINE CONSOLES

OpenShift 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.
7.7.1. About virtual machine console sessions

You can connect to the VNC and serial consoles of a running virtual machine from the **Console** tab on the **Virtual Machine Details** page of the web console.

The **VNC Console** opens by default when you navigate to the **Console** tab. You can open a connection to the serial console by clicking the **VNC Console** drop-down list and selecting **Serial Console**.

Console sessions remain active in the background unless they are disconnected. To ensure that only one console session is open at a time, click the **Disconnect before switching** check box before switching consoles.

You can open the active console session in a detached window by clicking **Open Console in New Window** or by clicking **Actions → Open Console**.

**Options for the VNC Console**

- Send key combinations to the virtual machine by clicking **Send Key**.

**Options for the Serial Console**

- Manually disconnect the **Serial Console** session from the virtual machine by clicking **Disconnect**.
- Manually open a **Serial Console** session to the virtual machine by clicking **Reconnect**.

7.7.2. Connecting to the virtual machine with the web console

7.7.2.1. Connecting to the terminal

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 **Workloads → Virtualization** from the side menu.

3. Click the **Virtual Machines** tab.

4. Select a virtual machine to open the **Virtual Machine Overview** screen.

5. In the **Details** tab, click the **virt-launcher-<vm-name>** pod.

6. Click the **Terminal** tab. If the terminal is blank, select the terminal and press any key to initiate connection.

7.7.2.2. Connecting to the serial console

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

**Procedure**
1. In the OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** page.

4. Click **Console**. The VNC console opens by default.

5. Click the **VNC Console** drop-down list and select **Serial Console**.

6. Optional: Open the serial console in a separate window by clicking **Open Console in New Window**.

### 7.7.2.3. Connecting to the VNC console

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

**Procedure**

1. In the OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** page.

4. Click the **Console** tab. The VNC console opens by default.

5. Optional: Open the VNC console in a separate window by clicking **Open Console in New Window**.

### 7.7.2.4. 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 NIC 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 OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a Windows virtual machine to open the **Virtual Machine Overview** screen.
4. Click the Console tab.

5. In the Console list, select Desktop Viewer.

6. In the Network Interface list, select the layer-2 NIC.

7. Click Launch Remote Desktop to download the console.rdp file.

8. Open an RDP client and reference the console.rdp file. For example, using remmina:

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

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

### 7.7.3. Accessing virtual machine consoles by using CLI commands

#### 7.7.3.1. Accessing a virtual machine instance via SSH

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

The **virtctl expose** command forwards a virtual machine instance (VMI) port to a node port and creates a service for enabled access. The following example creates the `fedora-vm-ssh` service that forwards traffic from a specific port of cluster nodes to port 22 of the `<fedora-vm>` virtual machine.

**Prerequisites**

- You must be in the same project as the VMI.
- The VMI you want to access must be connected to the default pod network by using the masquerade binding method.
- The VMI you want to access must be running.
- Install the OpenShift CLI (`oc`).

**Procedure**

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

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

   `<fedora-vm>` is the name of the VM that you run the `fedora-vm-ssh` service on.

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

   ```bash
   $ oc get svc
   ```

**Example output**

```
NAME            TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)           AGE
fedora-vm-ssh   NodePort   127.0.0.1      <none>        22:32551/TCP   6s
```

+ In this example, the service acquired the **32551** port.
1. Log in to the VMI via SSH. Use the `ipAddress` of any of the cluster nodes and the port that you found in the previous step:

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

### 7.7.3.2. 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>
```

### 7.7.3.3. 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
```

### 7.7.3.4. 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 NIC 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 NIC 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 OpenShift 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>
   ```

Example output

```json
...

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.

7.8. TRIGGERING VIRTUAL MACHINE FAILOVER BY RESOLVING A FAILED NODE

If a node fails and machine health checks are not deployed on your cluster, virtual machines (VMs) with RunStrategy: Always configured are not automatically relocated to healthy nodes. To trigger VM failover, you must manually delete the Node object.

**NOTE**

If you installed your cluster by using installer-provisioned infrastructure and you properly configured machine health checks:

- Failed nodes are automatically recycled.
- Virtual machines with RunStrategy set to Always or RerunOnFailure are automatically scheduled on healthy nodes.

7.8.1. Prerequisites

- A node where a virtual machine was running has the NotReady condition.
- The virtual machine that was running on the failed node has RunStrategy set to Always.
- You have installed the OpenShift CLI (oc).

7.8.2. Deleting nodes from a bare metal cluster

When you delete a node using the CLI, the node object is deleted in Kubernetes, but the pods that exist on the node are not deleted. Any bare pods not backed by a replication controller become inaccessible to OpenShift Container Platform. Pods backed by replication controllers are rescheduled to other available nodes. You must delete local manifest pods.

**Procedure**

Delete a node from an OpenShift Container Platform cluster running on bare metal by completing the following steps:

1. Mark the node as unschedulable:

   ```
   $ oc adm cordon <node_name>
   ```

2. Drain all pods on the node:

   ```
   $ oc adm drain <node_name> --force=true
   ```

   This step might fail if the node is offline or unresponsive. Even if the node does not respond, it might still be running a workload that writes to shared storage. To avoid data corruption, power down the physical hardware before you proceed.

3. Delete the node from the cluster:

   ```
Although the node object is now deleted from the cluster, it can still rejoin the cluster after reboot or if the kubelet service is restarted. To permanently delete the node and all its data, you must decommission the node.

4. If you powered down the physical hardware, turn it back on so that the node can rejoin the cluster.

7.8.3. Verifying virtual machine failover

After all resources are terminated on the unhealthy node, a new virtual machine instance (VMI) is automatically created on a healthy node for each relocated VM. To confirm that the VMI was created, view all VMIs by using the oc CLI.

7.8.3.1. Listing all virtual machine instances using the CLI

You can list all virtual machine instances (VMIs) in your cluster, including standalone VMIs and those owned by virtual machines, by using the oc command-line interface (CLI).

Procedure

- List all VMIs by running the following command:

  $ oc get vmis

7.9. INSTALLING THE QEMU GUEST AGENT ON VIRTUAL MACHINES

The QEMU guest agent is a daemon that runs on the virtual machine and passes information to the host about the virtual machine, users, file systems, and secondary networks.

7.9.1. 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. Ensure the service is persistent and start it:

   $ systemctl enable --now qemu-guest-agent

You can also install and start the QEMU guest agent by using the custom script field in the cloud-init section of the wizard when creating either virtual machines or virtual machines templates in the web console.
7.9.2. Installing QEMU guest agent on a Windows virtual machine

For Windows virtual machines, the QEMU guest agent is included in the VirtIO drivers, which can be installed using one of the following procedures:

7.9.2.1. 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 installation documentation for your version of Windows for specific installation steps.

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. 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.

7.9.2.2. Installing VirtIO drivers during Windows installation

Install the VirtIO drivers from the attached SATA CD drive 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.

---

**7.10. VIEWING THE QEMU GUEST AGENT INFORMATION FOR VIRTUAL MACHINES**

When the QEMU guest agent runs on the virtual machine, you can use the web console to view information about the virtual machine, users, file systems, and secondary networks.

### 7.10.1. Prerequisites

- Install the **QEMU guest agent** on the virtual machine.

### 7.10.2. About the QEMU guest agent information in the web console

When the QEMU guest agent is installed, the **Details** pane within the **Virtual Machine Overview** tab and the **Details** tab display information about the hostname, operating system, time zone, and logged in users.

The **Virtual Machine Overview** shows information about the guest operating system installed on the virtual machine. The **Details** tab displays a table with information for logged in users. The **Disks** tab displays a table with information for file systems.

**NOTE**

If the QEMU guest agent is not installed, the **Virtual Machine Overview** tab and the **Details** tab display information about the operating system that was specified when the virtual machine was created.

### 7.10.3. Viewing the QEMU guest agent information in the web console

You can use the web console to view information for virtual machines that is passed by the QEMU guest agent to the host.

**Procedure**

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

2. Click the **Virtual Machines** tab.
3. Select a virtual machine name to open the **Virtual Machine Overview** screen and view the **Details** pane.

4. Click **Logged in users** to view the **Details** tab that shows information for users.

5. Click the **Disks** tab to view information about the file systems.

### 7.11. MANAGING CONFIG MAPS, SECRETS, AND SERVICE ACCOUNTS IN VIRTUAL MACHINES

You can use secrets, config maps, and service accounts to pass configuration data to virtual machines. For example, you can:

- Give a virtual machine access to a service that requires credentials by adding a secret to the virtual machine.

- Store non-confidential configuration data in a config map so that a pod or another object can consume the data.

- Allow a component to access the API server by associating a service account with that component.

**NOTE**

OpenShift Virtualization exposes secrets, config maps, and service accounts as virtual machine disks so that you can use them across platforms without additional overhead.

### 7.11.1. Adding a secret, config map, or service account to a virtual machine

Add a secret, config map, or service account to a virtual machine by using the OpenShift Container Platform web console.

**Prerequisites**

- The secret, config map, or service account that you want to add must exist in the same namespace as the target virtual machine.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** screen.

4. Click the **Environment** tab.

5. Click **Select a resource** and select a secret, config map, or service account from the list. A six character serial number is automatically generated for the selected resource.

6. Click **Save**.

7. Optional. Add another object by clicking **Add Config Map, Secret or Service Account**.
NOTE

a. You can reset the form to the last saved state by clicking **Reload**.

b. The **Environment** resources are added to the virtual machine as disks. You can mount the secret, config map, or service account as you would mount any other disk.

c. If the virtual machine is running, changes will not take effect until you restart the virtual machine. The newly added resources are marked as pending changes for both the **Environment** and **Disks** tab in the **Pending Changes** banner at the top of the page.

**Verification**

1. From the **Virtual Machine Overview** page, click the **Disks** tab.

2. Check to ensure that the secret, config map, or service account is included in the list of disks.

3. Optional. Choose the appropriate method to apply your changes:
   a. If the virtual machine is running, restart the virtual machine by clicking **Actions → Restart Virtual Machine**.
   b. If the virtual machine is stopped, start the virtual machine by clicking **Actions → Start Virtual Machine**.

You can now mount the secret, config map, or service account as you would mount any other disk.

7.11.2. Removing a secret, config map, or service account from a virtual machine

Remove a secret, config map, or service account from a virtual machine by using the OpenShift Container Platform web console.

**Prerequisites**

- You must have at least one secret, config map, or service account that is attached to a virtual machine.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** screen.

4. Click the **Environment** tab.

5. Find the item that you want to delete in the list, and click **Remove** on the right side of the item.

6. Click **Save**.
NOTE
You can reset the form to the last saved state by clicking Reload.

Verification

1. From the Virtual Machine Overview page, click the Disks tab.
2. Check to ensure that the secret, config map, or service account that you removed is no longer included in the list of disks.

7.11.3. Additional resources

- Providing sensitive data to pods
- Understanding and creating service accounts
- Understanding config maps

7.12. INSTALLING VIRTIO DRIVER ON AN EXISTING WINDOWS VIRTUAL MACHINE

7.12.1. Understanding VirtIO drivers

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

The container-native-virtualization/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 container-native-virtualization/virtio-win container disk can be removed from the virtual machine.

See also: Installing Virtio drivers on a new Windows virtual machine.

7.12.2. Supported VirtIO drivers for Microsoft Windows virtual machines

Table 7.3. Supported drivers

<table>
<thead>
<tr>
<th>Driver name</th>
<th>Hardware ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>viostor</td>
<td>VEN_1AF4&amp;DEV_1001</td>
<td>The block driver. Sometimes displays as an SCSI Controller in the Other devices group.</td>
</tr>
<tr>
<td></td>
<td>VEN_1AF4&amp;DEV_1042</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>viorng</td>
<td>VEN_1AF4&amp;DEV_1005</td>
<td>The entropy source driver. Sometimes displays as a PCI Device in the Other devices group.</td>
</tr>
<tr>
<td></td>
<td>VEN_1AF4&amp;DEV_1044</td>
<td></td>
</tr>
</tbody>
</table>
### 7.12.3. Adding VirtIO drivers container disk to a virtual machine

OpenShift Virtualization distributes VirtIO drivers for Microsoft Windows as a container disk, which is available from the Red Hat Ecosystem Catalog. To install these drivers to a Windows virtual machine, attach the `container-native-virtualization/virtio-win` container disk to the virtual machine as a SATA CD drive in the virtual machine configuration file.

**Prerequisites**

- Download the `container-native-virtualization/virtio-win` container disk from the Red Hat Ecosystem Catalog. This is not mandatory, because the container disk will be downloaded from the Red Hat registry if it is not already present in the cluster, but it can reduce installation time.

**Procedure**

1. Add the `container-native-virtualization/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.

```yaml
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2
          cdrom:
            bus: sata
      volumes:
        - containerDisk:
            image: container-native-virtualization/virtio-win
            name: virtiocontainerdisk
```

OpenShift 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 `container-native-virtualization/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.

### 7.12.4. 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 installation documentation for your version of Windows for specific installation steps.

**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. 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.

### 7.12.5. Removing the VirtIO container disk from a virtual machine

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

**Procedure**

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

```bash
$ oc edit vm <vm-name>
```
2. Reboot the virtual machine for the changes to take effect.

7.13. INSTALLING VIRTIO DRIVER ON A NEW WINDOWS VIRTUAL MACHINE

7.13.1. 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.

7.13.2. Understanding VirtIO drivers

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

The container-native-virtualization/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 container-native-virtualization/virtio-win container disk can be removed from the virtual machine.

See also: Installing VirtIO driver on an existing Windows virtual machine.

7.13.3. Supported VirtIO drivers for Microsoft Windows virtual machines

Table 7.4. Supported drivers

<table>
<thead>
<tr>
<th>Driver name</th>
<th>Hardware ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>viostor</td>
<td>VEN_1AF4&amp;DEV_1001</td>
<td>The block driver. Sometimes displays as a SCSI Controller in the Other devices group.</td>
</tr>
<tr>
<td></td>
<td>VEN_1AF4&amp;DEV_1042</td>
<td></td>
</tr>
<tr>
<td>viorng</td>
<td>VEN_1AF4&amp;DEV_1005</td>
<td>The entropy source driver. Sometimes displays as a PCI Device in the Other devices group.</td>
</tr>
<tr>
<td></td>
<td>VEN_1AF4&amp;DEV_1044</td>
<td></td>
</tr>
</tbody>
</table>
### 7.13.4. Adding VirtIO drivers container disk to a virtual machine

OpenShift Virtualization distributes VirtIO drivers for Microsoft Windows as a container disk, which is available from the Red Hat Ecosystem Catalog. To install these drivers to a Windows virtual machine, attach the `container-native-virtualization/virtio-win` container disk to the virtual machine as a SATA CD drive in the virtual machine configuration file.

**Prerequisites**

- Download the `container-native-virtualization/virtio-win` container disk from the Red Hat Ecosystem Catalog. This is not mandatory, because the container disk will be downloaded from the Red Hat registry if it is not already present in the cluster, but it can reduce installation time.

**Procedure**

1. Add the `container-native-virtualization/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.

```yaml
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2
          cdrom:
            bus: sata
      volumes:
        - containerDisk:
            image: container-native-virtualization/virtio-win
            name: virtiocontainerdisk
```

OpenShift 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 `container-native-virtualization/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>`.

<table>
<thead>
<tr>
<th>Driver name</th>
<th>Hardware ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetKVM</td>
<td>VEN_1AF4&amp;DEV_1000 VEN_1AF4&amp;DEV_1041</td>
<td>The network driver. Sometimes displays as an Ethernet Controller in the Other devices group. Available only if a VirtIO NIC is configured.</td>
</tr>
</tbody>
</table>
After the virtual machine has started, the VirtIO drivers can be installed from the attached SATA CD drive.

### 7.13.5. 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.

### 7.13.6. Removing the VirtIO container disk from a virtual machine

After installing all required VirtIO drivers to the virtual machine, the `container-native-virtualization/virtio-win` container disk no longer needs to be attached to the virtual machine. Remove the `container-native-virtualization/virtio-win` container disk from the virtual machine configuration file.

**Procedure**

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

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

spec:
  domain:
    devices:
      disks:
      - name: virtiocontainerdisk
        bootOrder: 2
        cdrom:
          bus: sata
      volumes:
```
2. Reboot the virtual machine for the changes to take effect.

7.14. ADVANCED VIRTUAL MACHINE MANAGEMENT

7.14.1. Automating management tasks

You can automate OpenShift Virtualization management tasks by using Red Hat Ansible Automation Platform. Learn the basics by using an Ansible Playbook to create a new virtual machine.


Ansible is an automation tool used to configure systems, deploy software, and perform rolling updates. Ansible includes support for OpenShift Virtualization, 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 OpenShift Virtualization 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.

7.14.1.2. Automating virtual machine creation

You can use the `kubevirt_vm` Ansible Playbook to create virtual machines in your OpenShift Container Platform cluster using Red Hat Ansible Automation Platform.

**Prerequisites**

- Red Hat Ansible Engine version 2.8 or newer

**Procedure**

1. Edit an Ansible Playbook YAML file so that it includes the `kubevirt_vm` task:

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

   **NOTE**

   This snippet only includes the `kubevirt_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:

```
kubevirt_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
```

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

```
kubevirt_vm:
  namespace: default
  name: vm1
  state: running
  cpu_cores: 1
```

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

4. Run the ansible-playbook command, using your playbook’s file name as the only argument:

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

5. Review the output to determine if the play was successful:

```
Example output

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

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

6. 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.

7.14.1.3. Example: Ansible Playbook for creating virtual machines
You can use the `kubevirt_vm` Ansible Playbook to automate virtual machine creation.

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

```yaml
---
- name: Ansible Playbook 1
  hosts: localhost
  connection: local
  tasks:
    - name: Create my first VM
      kubevirt_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](#)

### 7.14.2. Configuring PXE booting for virtual machines

PXE booting, or network booting, is available in OpenShift 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.

#### 7.14.2.1. Prerequisites

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

#### 7.14.2.2. OpenShift Virtualization networking glossary

OpenShift Virtualization provides advanced networking functionality by using custom resources and plug-ins.

The following terms are used throughout OpenShift Virtualization documentation:

**Container Network Interface (CNI)**

A Cloud Native Computing Foundation project, focused on container network connectivity. OpenShift 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.

Network attachment definition

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.

7.14.2.3. 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 network attachment definition 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 network attachment definition file for PXE network `pxe-net-conf`:

```yaml
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",
        "vlan": 1
      },
      {
        "type": "cnv-tuning"
      }
    ]
  }

```
Optional: The VLAN tag.

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 network attachment definition 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>**:

   ```yaml
   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**:

   ```yaml
   devices:
     disks:
       - disk:
           bus: virtio
           name: containerdisk
           bootOrder: 2
   
   c. Specify that the network is connected to the previously created network attachment definition. In this scenario, **<pxe-net>** is connected to the network attachment definition called **<pxe-net-conf>**:

   ```yaml
   networks:
     - name: default
       pod: {}
   ```
4. Create the virtual machine instance:

$ oc create -f vmi-pxe-boot.yaml

Example output

virtualmachineinstance.kubevirt.io "vmi-pxe-boot" created

1. Wait for the virtual machine instance to run:

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

2. View the virtual machine instance using VNC:

$ virtctl vnc vmi-pxe-boot

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

4. Log in to the virtual machine instance:

$ virtctl console vmi-pxe-boot

5. 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

Example output

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

7.14.2.4. 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:
  pxe-boot:
    domain:
    devices:
7.14.3. 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. OpenShift Virtualization allows you to configure guest memory overcommitment and disable guest memory overhead accounting.

```yaml
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: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: kubervirt/fedora-cloud-container-disk-demo
- cloudInitNoCloud:
  userData:
    #!/bin/bash
    echo "fedora" | passwd fedora --stdin
  name: cloudinitdisk
status: {}

7.14.3. 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. OpenShift Virtualization allows you to configure guest memory overcommitment and disable guest memory overhead accounting.

⚠️ WARNING

The following procedures increase the chance that virtual machine processes will be killed due to memory pressure. Proceed only if you understand the risks.

7.14.3.1. 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 (VMIs). Enabling memory overcommitment means that 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 (VMs) 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.

**IMPORTANT**

Memory overcommitment increases the potential for virtual machine processes to be killed due to memory pressure (OOM killed).

The potential for a VM to be OOM killed varies based on your specific configuration, node memory, available swap space, virtual machine memory consumption, the use of kernel same-page merging (KSM), and other factors.

**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.

```yaml
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
```

**7.14.3.2. Disabling guest memory overhead accounting**

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.

**IMPORTANT**

Disabling guest memory overhead accounting increases the potential for virtual machine processes to be killed due to memory pressure (OOM killed).

The potential for a VM to be OOM killed varies based on your specific configuration, node memory, available swap space, virtual machine memory consumption, the use of kernel same-page merging (KSM), and other factors.

**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.

   ```yaml
   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:

   
   ```bash
   $ oc create -f <file_name>.yaml
   ```

**7.14.4. Using huge pages with virtual machines**

You can use huge pages as backing memory for virtual machines in your cluster.

**7.14.4.1. Prerequisites**

- Nodes must have pre-allocated huge pages configured.

**7.14.4.2. What huge pages do**

Memory is managed in blocks known as pages. On most systems, a page is 4Ki. 1Mi of memory is equal to 256 pages; 1Gi of memory is 256,000 pages, and so on. CPUs have a built-in memory management unit that manages a list of these pages in hardware. The Translation Lookaside Buffer (TLB) is a small hardware cache of virtual-to-physical page mappings. If the virtual address passed in a hardware instruction can be found in the TLB, the mapping can be determined quickly. If not, a TLB miss occurs, and the system falls back to slower, software-based address translation, resulting in performance issues. Since the size of the TLB is fixed, the only way to reduce the chance of a TLB miss is to increase the page size.
A huge page is a memory page that is larger than 4Ki. On x86_64 architectures, there are two common huge page sizes: 2Mi and 1Gi. Sizes vary on other architectures. In order to use huge pages, code must be written so that applications are aware of them. Transparent Huge Pages (THP) attempt to automate the management of huge pages without application knowledge, but they have limitations. In particular, they are limited to 2Mi page sizes. THP can lead to performance degradation on nodes with high memory utilization or fragmentation due to defragmenting efforts of THP, which can lock memory pages. For this reason, some applications may be designed to (or recommend) usage of pre-allocated huge pages instead of THP.

In OpenShift Virtualization, virtual machines can be configured to consume pre-allocated huge pages.

### 7.14.4.3. Configuring huge pages for virtual machines

You can configure virtual machines to use pre-allocated huge pages by including the `memory.hugepages.pageSize` and `resources.requests.memory` parameters in your virtual machine configuration.

The memory request must be divisible by the page size. For example, you cannot request 500Mi memory with a page size of 1Gi.

**NOTE**

The memory layouts of the host and the guest OS are unrelated. Huge pages requested in the virtual machine manifest apply to QEMU. Huge pages inside the guest can only be configured based on the amount of available memory of the virtual machine instance.

If you edit a running virtual machine, the virtual machine must be rebooted for the changes to take effect.

**Prerequisites**

- Nodes must have pre-allocated huge pages configured.

**Procedure**

1. In your virtual machine configuration, add the `resources.requests.memory` and `memory.hugepages.pageSize` parameters to the `spec.domain`. The following configuration snippet is for a virtual machine that requests a total of 4Gi memory with a page size of 1Gi:

```yaml
kind: VirtualMachine
...
spec:
  domain:
    resources:
      requests:
        memory: "4Gi"  # 1
      memory:
        hugepages:
          pageSize: "1Gi"  # 2
...
```

1. The total amount of memory requested for the virtual machine. This value must be divisible by the page size.
The size of each huge page. Valid values for x86_64 architecture are 1Gi and 2Mi. The page size must be smaller than the requested memory.

2. Apply the virtual machine configuration:

```bash
$ oc apply -f <virtual_machine>.yaml
```

7.14.5. Enabling dedicated resources for virtual machines

To improve performance, you can dedicate node resources, such as CPU, to a virtual machine.

7.14.5.1. About dedicated resources

When you enable dedicated resources for your virtual machine, your virtual machine’s workload is scheduled on CPUs that will not be used by other processes. By using dedicated resources, you can improve the performance of the virtual machine and the accuracy of latency predictions.

7.14.5.2. Prerequisites

- The CPU Manager must be configured on the node. Verify that the node has the `cpumanager = true` label before scheduling virtual machine workloads.
- The virtual machine must be powered off.

7.14.5.3. Enabling dedicated resources for a virtual machine

You can enable dedicated resources for a virtual machine in the Virtual Machine Overview page of the web console.

Procedure

1. Click Workloads → Virtual Machines from the side menu.
2. Select a virtual machine to open the Virtual Machine Overview page.
3. Click the Details tab.
4. Click the pencil icon to the right of the Dedicated Resources field to open the Dedicated Resources window.
5. Select Schedule this workload with dedicated resources (guaranteed policy)
6. Click Save.

7.14.6. Scheduling virtual machines

You can schedule a virtual machine (VM) on a node by ensuring that the VM’s CPU model and policy attribute are matched for compatibility with the CPU models and policy attributes supported by the node.

7.14.6.1. Understanding policy attributes
You can schedule a virtual machine (VM) by specifying a policy attribute and a CPU feature that is matched for compatibility when the VM is scheduled on a node. A policy attribute specified for a VM determines how that VM is scheduled on a node.

<table>
<thead>
<tr>
<th>Policy attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>force</td>
<td>The VM is forced to be scheduled on a node. This is true even if the host CPU does not support the VM’s CPU.</td>
</tr>
<tr>
<td>require</td>
<td>Default policy that applies to a VM if the VM is not configured with a specific CPU model and feature specification. If a node is not configured to support CPU node discovery with this default policy attribute or any one of the other policy attributes, VMs are not scheduled on that node. Either the host CPU must support the VM’s CPU or the hypervisor must be able to emulate the supported CPU model.</td>
</tr>
<tr>
<td>optional</td>
<td>The VM is added to a node if that VM is supported by the host’s physical machine CPU.</td>
</tr>
<tr>
<td>disable</td>
<td>The VM cannot be scheduled with CPU node discovery.</td>
</tr>
<tr>
<td>forbid</td>
<td>The VM is not scheduled even if the feature is supported by the host CPU and CPU node discovery is enabled.</td>
</tr>
</tbody>
</table>

7.14.6.2. Setting a policy attribute and CPU feature

You can set a policy attribute and CPU feature for each virtual machine (VM) to ensure that it is scheduled on a node according to policy and feature. The CPU feature that you set is verified to ensure that it is supported by the host CPU or emulated by the hypervisor.

Procedure

- Edit the `domain` spec of your VM configuration file. The following example sets the CPU feature and the `require` policy for a virtual machine instance (VMI):

  ```yaml
  apiVersion: v1
  kind: VirtualMachine
  metadata:
    name: myvmi
  spec:
    domain:
      cpu:
        features:
        - name: apic
        policy: require
  ```

  1. Name of the CPU feature for the VM or VMI.
  2. Policy attribute for the VM or VMI.

7.14.6.3. Scheduling virtual machines with the supported CPU model
You can configure a CPU model for a virtual machine (VM) or a virtual machine instance (VMI) to schedule it on a node where its CPU model is supported.

**Procedure**

- Edit the `domain` spec of your virtual machine configuration file. The following example shows a specific CPU model defined for a VMI:

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  name: myvmi
spec:
domain:
cpu:
  model: Conroe
```

CPU model for the VMI.

### 7.14.6.4. Scheduling virtual machines with the host model

When the CPU model for a virtual machine (VM) is set to `host-model`, the VM inherits the CPU model of the node where it is scheduled.

**Procedure**

- Edit the `domain` spec of your VM configuration file. The following example shows `host-model` being specified for the virtual machine instance (VMI):

```yaml
apiVersion: kubevirt/v1alpha3
kind: VirtualMachineInstance
metadata:
  name: myvmi
spec:
domain:
cpu:
  model: host-model
```

The VM or VMI that inherits the CPU model of the node where it is scheduled.

### 7.15. IMPORTING VIRTUAL MACHINES

#### 7.15.1. TLS certificates for data volume imports

##### 7.15.1.1. Adding TLS certificates for authenticating data volume imports

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

Create the config map by referencing the relative file path for the TLS certificate.
Procedure

1. Ensure you are in the correct namespace. The config map can only be referenced by data volumes if it is in the same namespace.

   $ oc get ns

2. Create the config map:

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

7.15.1.2. Example: Config map created from a TLS certificate

The following example is of a config map created from `ca.pem` TLS certificate.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: tls-certs
data:
  ca.pem: |
    -----BEGIN CERTIFICATE-----
    ...
    ...
    -----END CERTIFICATE-----
```

7.15.2. Importing virtual machine images with data volumes

Use the Containerized Data Importer (CDI) to import a virtual machine image into a persistent volume claim (PVC) by using a data volume. You can attach a data volume to a virtual machine for persistent storage.

The virtual machine image can be hosted at an HTTP or HTTPS endpoint, or built into a container disk and stored in a container registry.

**IMPORTANT**

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 virtual machine. Refer to the operating system documentation for details.

7.15.2.1. Prerequisites

- If the endpoint requires a TLS certificate, the certificate must be included in a config map in the same namespace as the data volume and referenced in the data volume configuration.

- To import a container disk:
  - You might need to prepare a container disk from a virtual machine image and store it in your container registry before importing it.
If the container registry does not have TLS, you must add the registry to the `cdi-insecure-registries` config map before you can import a container disk from it.

- You might need to define a storage class or prepare CDI scratch space for this operation to complete successfully.

### 7.15.2.2. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2**</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2*</td>
<td>✓ QCOW2*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
</tbody>
</table>

- Supported operation
- Unsupported operation
- * Requires scratch space
- ** Requires scratch space if a custom certificate authority is required

### 7.15.2.3. About data volumes

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

### 7.15.2.4. Importing a virtual machine image into a persistent volume claim by using a data volume

You can import a virtual machine image into a persistent volume claim (PVC) by using a data volume.

The virtual machine image can be hosted at an HTTP or HTTPS endpoint, or the image can be built into a container disk and stored in a container registry.

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

**Prerequisites**

- You have installed the OpenShift CLI (**oc**).
- Your cluster has at least one available persistent volume.
To import a virtual machine image you must have the following:

- 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. For example: `http://www.example.com/path/to/data`

To import a container disk you must have the following:

- A container disk built from a virtual machine image stored in your container image registry, along with any authentication credentials needed to access the data source. For example: `docker://registry.example.com/container-image`

**Procedure**

1. Optional: If your data source requires authentication credentials, edit the `endpoint-secret.yaml` file, and apply the updated configuration to the cluster:

   ```yaml
   apiVersion: v1
   kind: Secret
   metadata:
     name: <endpoint-secret>
     labels:
       app: containerized-data-importer
       type: Opaque
   data:
     accessKeyId: ""  # 1
     secretKey: ""  # 2
   
   $ oc apply -f endpoint-secret.yaml
   ``

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

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

   ```yaml
   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:
   ```
- The source type to import the image from. This example uses an HTTP endpoint. To import a container disk from a registry, replace `http` with `registry`.

- The source of the virtual machine image you want to import. This example references a virtual machine image at an HTTP endpoint. An example of a container registry endpoint is `url: "docker://kubevirt/fedora-cloud-container-disk-demo:latest"`.

- The `secretRef` parameter is optional.

- The `certConfigMap` is required for communicating with servers that use self-signed certificates or certificates not signed by the system CA bundle. The referenced config map must be in the same namespace as the data volume.

- Specify `type: dataVolume` or `type: ""`. If you specify any other value for `type`, such as `persistentVolumeClaim`, a warning is displayed, and the virtual machine does not start.

3. Create the virtual machine:
NOTE

The `oc create` command creates the data volume 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 data volume status changes to **Succeeded**, and the virtual machine is allowed to start.

Data volume 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.

**Verification**

1. The importer pod downloads the virtual machine image or container disk from the specified URL and stores it on the provisioned PV. View the status of the importer pod by running the following command:

   ```
   $ oc get pods
   ```

2. Monitor the data volume status until it shows **Succeeded** by running the following command:

   ```
   $ oc describe dv <datavolume-name>
   ```

   ① The name of the data volume as specified under `dataVolumeTemplates.metadata.name` in the virtual machine configuration file. In the example configuration above, this is `fedora-dv`.

3. To verify that provisioning is complete and that the VMI has started, try accessing its serial console by running the following command:

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

### 7.15.3. Importing virtual machine images to block storage with data volumes

You can import an existing virtual machine image into your OpenShift Container Platform cluster. OpenShift Virtualization uses data volumes to automate the import of data and the creation of an underlying persistent volume claim (PVC).

**IMPORTANT**

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 that is installed on the virtual machine. Refer to the operating system documentation for details.

### 7.15.3.1. Prerequisites

- `$ oc create -f vm-<name>-datavolume.yaml`
If you require scratch space according to the CDI supported operations matrix, you must first define a storage class or prepare CDI scratch space for this operation to complete successfully.

### 7.15.3.2. About data volumes

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

### 7.15.3.3. About block persistent volumes

A block persistent volume (PV) is a PV that is backed by a raw block device. These volumes do not have a file system and can provide performance benefits for virtual machines by reducing overhead.

Raw block volumes are provisioned by specifying `volumeMode: Block` in the PV and persistent volume claim (PVC) specification.

### 7.15.3.4. Creating a local block persistent volume

Create a local block persistent volume (PV) on a node by populating a file and mounting it as a loop device. You can then reference this loop device in a PV manifest as a **Block** volume and use it as a block device for a virtual machine image.

**Procedure**

1. Log in as **root** to the node on which to create the local PV. This procedure uses **node01** for its examples.

2. Create a file and populate it with null characters so that it can be used as a block device. The following example creates a file **loop10** with a size of 2Gb (20 100Mb blocks):

   ```
   $ dd if=/dev/zero of=<loop10> bs=100M count=20
   ```

3. Mount the **loop10** file as a loop device.

   ```
   $ losetup </dev/loop10>d3 <loop10> ① ②
   ``

   ① File path where the loop device is mounted.

   ② The file created in the previous step to be mounted as the loop device.

4. Create a **PersistentVolume** manifest that references the mounted loop device.

   ```yaml
   kind: PersistentVolume
   apiVersion: v1
   metadata:
     name: <local-block-pv10>
   annotations:
   spec:
     local:
       path: </dev/loop10> ①
     capacity:
       storage: <2Gi>
   ```
The path of the loop device on the node.

Specifies it is a block PV.

Optional: Set a storage class for the PV. If you omit it, the cluster default is used.

The node on which the block device was mounted.

5. Create the block PV.

The file name of the persistent volume created in the previous step.

7.15.3.5. Importing a virtual machine image to a block persistent volume using data volumes

You can import an existing virtual machine image into your OpenShift Container Platform cluster. OpenShift Virtualization uses data volumes to automate the importing data and the creation of an underlying persistent volume claim (PVC). You can then reference the data volume in a virtual machine manifest.

Prerequisites

- A virtual machine disk image, in RAW, ISO, or QCOW2 format, optionally compressed by using xz or gz
- An HTTP or s3 endpoint where the image is hosted, along with any authentication credentials needed to access the data source
- At least one available block PV.

Procedure

1. If your data source requires authentication credentials, edit the endpoint-secret.yaml file, and apply the updated configuration to the cluster.

   a. Edit the endpoint-secret.yaml file with your preferred text editor:

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

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

b. Update the secret by running the following command:

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

2. Create a **DataVolume** manifest that specifies the data source for the image you want to import and **volumeMode: Block** so that an available block PV is used.

```yaml
apiVersion: cdi.kubevirt.io/v1beta1  
kind: DataVolume  
metadata:  
  name: <import-pv-datavolume>  
spec:  
  storageClassName: local  
  source:  
    http:  
      url: <http://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora-Cloud-Base-28-1.1.x86_64.qcow2>  
      secretRef: <endpoint-secret>  
  pvc:  
    volumeMode: Block  
    accessModes:  
      - ReadWriteOnce  
    resources:  
      requests:  
        storage: <2Gi>
```

1. The name of the data volume.  
2. Optional: Set the storage class or omit it to accept the cluster default.  
3. The **HTTP** source of the image to import.  
4. Only required if the data source requires authentication.  
5. Required for importing to a block PV.  

3. Create the data volume to import the virtual machine image by running the following command:
7.15.3.6. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCrow2)</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2*</td>
<td>✓ QCOW2*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
</tr>
<tr>
<td></td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
</tr>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
</tr>
</tbody>
</table>

✓ Supported operation

☐ Unsupported operation

* Requires scratch space

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

7.15.4. Importing a single Red Hat Virtualization virtual machine

You can import a single Red Hat Virtualization (RHV) virtual machine into OpenShift Virtualization by using the VM Import wizard or the CLI.

7.15.4.1. OpenShift Virtualization storage feature matrix

The following table describes the OpenShift Virtualization storage types that support VM import.

<table>
<thead>
<tr>
<th></th>
<th>RHV VM import</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenShift Container Storage: RBD block-mode volumes</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenShift Virtualization hostpath provisioner</td>
<td>No</td>
</tr>
<tr>
<td>Other multi-node writable storage</td>
<td>Yes [1]</td>
</tr>
<tr>
<td>Other single-node writable storage</td>
<td>Yes [2]</td>
</tr>
</tbody>
</table>
1. PVCs must request a ReadWriteMany access mode.

2. PVCs must request a ReadWriteOnce access mode.

7.15.4.2. Prerequisites for importing a virtual machine

Importing a virtual machine from Red Hat Virtualization (RHV) into OpenShift Virtualization has the following prerequisites:

- You must have admin user privileges.

- Storage:
  - The OpenShift Virtualization local and shared persistent storage classes must support VM import.
  - If you are using Ceph RBD block-mode volumes, the storage must be large enough to accommodate the virtual disk. If the disk is too large for the available storage, the import process fails and the PV that is used to copy the virtual disk is not released.

- Networks:
  - The RHV and OpenShift Virtualization networks must either have the same name or be mapped to each other.
  - The RHV VM network interface must be **e1000**, **rtl8139**, or **virtio**.

- VM disks:
  - The disk interface must be **sata**, **virtio_scsi**, or **virtio**.
  - The disk must not be configured as a direct LUN.
  - The disk status must not be **illegal** or **locked**.
  - The storage type must be **image**.
  - SCSI reservation must be disabled.
  - **ScsiGenericIO** must be disabled.

- VM configuration:
  - If the VM uses GPU resources, the nodes providing the GPUs must be configured.
  - The VM must not be configured for vGPU resources.
  - The VM must not have snapshots with disks in an **illegal** state.
  - The VM must not have been created with OpenShift Container Platform and subsequently added to RHV.
  - The VM must not be configured for USB devices.
  - The watchdog model must not be **diag288**.

7.15.4.3. Importing a virtual machine with the VM Import wizard
You can import a single virtual machine with the VM Import wizard.

**Procedure**

1. In the web console, click **Workloads → Virtual Machines.**
2. Click **Create Virtual Machine** and select **Import with Wizard.**
3. Select **Red Hat Virtualization (RHV)** from the **Provider** list.
4. Select **Connect to New Instance** or a saved RHV instance.
   - If you select **Connect to New Instance**, fill in the following fields:
     - **API URL**: For example, `https://<RHV_Manager_FQDN>/ovirt-engine/api`
     - **CA certificate**: Click **Browse** to upload the RHV Manager CA certificate or paste the CA certificate into the field. View the CA certificate by running the following command:
       ```bash
       $ openssl s_client -connect <RHV_Manager_FQDN>:443 -showcerts < /dev/null
       ```
       The CA certificate is the second certificate in the output.
     - **Username**: RHV Manager user name, for example, `ocpadmin@internal`
     - **Password**: RHV Manager password
   - If you select a saved RHV instance, the wizard connects to the RHV instance using the saved credentials.
5. Click **Check and Save** and wait for the connection to complete.

**NOTE**

The connection details are stored in a secret. If you add a provider with an incorrect URL, user name, or password, click **Workloads → Secrets** and delete the provider secret.

6. Select a cluster and a virtual machine.
7. Click **Next.**
8. In the **Review** screen, review your settings.
9. Optional: You can select **Start virtual machine on creation**
10. Click **Edit** to update the following settings:
    - **General → Name**: The VM name is limited to 63 characters.
    - **General → Description**: Optional description of the VM.
    - **Storage Class**: Select **NFS** or **ocs-storagecluster-ceph-rbd**. If you select **ocs-storagecluster-ceph-rbd**, you must set the **Volume Mode** of the disk to **Block**.
• Advanced → Volume Mode: Select Block.

• Advanced → Volume Mode: Select Block.

• Networking → Network: You can select a network from a list of available network attachment definition objects.

II. Click Import or Review and Import, if you have edited the import settings. A Successfully created virtual machine message and a list of resources created for the virtual machine are displayed. The virtual machine appears in Workloads → Virtual Machines.

### Virtual machine wizard fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td></td>
<td>Template from which to create the virtual machine. Selecting a template will automatically complete other fields.</td>
</tr>
<tr>
<td>Source</td>
<td>PXE</td>
<td>Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.</td>
</tr>
<tr>
<td></td>
<td>URL</td>
<td>Provision virtual machine from an image available from an HTTP or S3 endpoint.</td>
</tr>
<tr>
<td>Container</td>
<td></td>
<td>Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: kubevirt/cirros-registry-disk-demo.</td>
</tr>
<tr>
<td>Disk</td>
<td></td>
<td>Provision virtual machine from a disk.</td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
<td>The primary operating system that is selected for the virtual machine.</td>
</tr>
<tr>
<td>Flavor</td>
<td>small, medium, large, tiny, Custom</td>
<td>Presets that determine the amount of CPU and memory allocated to the virtual machine. The presets displayed for Flavor are determined by the operating system.</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td>Size in GiB of the memory allocated to the virtual machine.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CPUs</td>
<td></td>
<td>The amount of CPU allocated to the virtual machine.</td>
</tr>
<tr>
<td>Workload Profile</td>
<td>High Performance</td>
<td>A virtual machine configuration that is optimized for high-performance workloads.</td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>A profile optimized to run server workloads.</td>
</tr>
<tr>
<td></td>
<td>Desktop</td>
<td>A virtual machine configuration for use on a desktop.</td>
</tr>
</tbody>
</table>

**Name**

- The name can contain lowercase 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 uppercase letters, spaces, periods (.), or special characters.

**Description**

Optional description field.

**Start virtual machine on creation**

Select to automatically start the virtual machine upon creation.

---

### Networking fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are e1000e and virtio.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
<tr>
<td>Type</td>
<td>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.</td>
</tr>
</tbody>
</table>
### MAC Address

MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.

### Storage fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Select a blank disk for the virtual machine or choose from the options available: URL, Container, Attach Cloned Disk, or Attach Disk. To select an existing disk and attach it to the virtual machine, choose Attach Cloned Disk or Attach Disk from a list of available persistent volume claims (PVCs).</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the disk. The name can contain lowercase 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 uppercase letters, spaces, or special characters.</td>
</tr>
<tr>
<td>Size (GiB)</td>
<td>Size, in GiB, of the disk.</td>
</tr>
<tr>
<td>Interface</td>
<td>Type of disk device. Supported interfaces are virtIO, SATA, and SCSI.</td>
</tr>
<tr>
<td>Storage Class</td>
<td>The storage class that is used to create the disk.</td>
</tr>
<tr>
<td>Advanced → Volume Mode</td>
<td>Defines whether the persistent volume uses a formatted file system or raw block state. Default is Filesystem.</td>
</tr>
</tbody>
</table>

### Advanced storage settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Mode</td>
<td>Filesystem</td>
<td>Stores the virtual disk on a file system-based volume.</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>Stores the virtual disk directly on the block volume. Only use Block if the underlying storage supports it.</td>
</tr>
<tr>
<td>Access Mode</td>
<td>Single User (RWO)</td>
<td>The disk can be mounted as read/write by a single node.</td>
</tr>
</tbody>
</table>
### Table: Access modes for RHV VMs

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Access (RWX)</td>
<td>The disk can be mounted as read/write by many nodes.</td>
<td></td>
</tr>
<tr>
<td>Read Only (ROX)</td>
<td>The disk can be mounted as read-only by many nodes.</td>
<td></td>
</tr>
</tbody>
</table>

1. You can change the access mode by using the command line interface.

### 7.15.4.4. Importing a virtual machine with the CLI

You can import a virtual machine with the CLI by creating the `Secret` and `VirtualMachineImport` custom resources (CRs). The `Secret` CR stores the RHV Manager credentials and CA certificate. The `VirtualMachineImport` CR defines the parameters of the VM import process.

Optional: You can create a `ResourceMapping` CR that is separate from the `VirtualMachineImport` CR. A `ResourceMapping` CR provides greater flexibility, for example, if you import additional RHV VMs.

**IMPORTANT**

The default target storage class must be NFS. Cinder does not support RHV VM import.

**Procedure**

1. Create the `Secret` CR by running the following command:

   ```bash
   $ cat <<EOF | oc create -f -
   apiVersion: v1
   kind: Secret
   metadata:
     name: rhv-credentials
     namespace: default
   type: Opaque
   stringData:
     ovirt: |
     apiUrl: <api_endpoint>
     username: ocpadmin@internal
     password: 
     caCert: |
     -----BEGIN CERTIFICATE-----
     ----END CERTIFICATE-----
   EOF
   ```

   1. Optional. You can specify a different namespace in all the CRs.
   2. Specify the API endpoint of the RHV Manager, for example, "https://www.example.com:8443/ovirt-engine/api"
3. Specify the password for `ocpadmin@internal`.

4. Specify the RHV Manager CA certificate. You can obtain the CA certificate by running the following command:

   ```bash
   $ openssl s_client -connect :443 -showcerts < /dev/null
   ```

2. Optional: Create a `ResourceMapping` CR if you want to separate the resource mapping from the `VirtualMachineImport` CR by running the following command:

   ```bash
   $ cat <<EOF | kubectl create -f -
   apiVersion: v2v.kubevirt.io/v1alpha1
   kind: ResourceMapping
   metadata:
     name: resourcemapping_example
     namespace: default
   spec:
     ovirt:
       networkMappings:
         - source:
           name: <rhv_logical_network>/<vnic_profile> 1
           target:
             name: <target_network> 2
             type: pod
       storageMappings: 3
         - source:
           name: <rhv_storage_domain> 4
           target:
             name: <target_storage_class> 5
             volumeMode: <volume_mode> 6
   EOF
   ```

3. Create the `VirtualMachineImport` CR by running the following command:

   ```bash
   $ cat <<EOF | oc create -f -
   apiVersion: v2v.kubevirt.io/v1beta1
   kind: VirtualMachineImport
   metadata:
     name: vm-import
   EOF
   ```

1. Specify the RHV logical network and vNIC profile.

2. Specify the OpenShift Virtualization network.

3. If storage mappings are specified in both the `ResourceMapping` and the `VirtualMachineImport` CRs, the `VirtualMachineImport` CR takes precedence.

4. Specify the RHV storage domain.

5. Specify `nfs` or `ocs-storagecluster-ceph-rbd`.

6. If you specified the `ocs-storagecluster-ceph-rbd` storage class, you must specify `Block` as the volume mode.
If you create a `ResourceMapping` CR, uncomment the `resourceMapping` section.

Specify the target VM name.

Specify the source VM ID, for example, `80554327-0569-496b-bdeb-fcbbf52b827b`. You can obtain the VM ID by entering `https://www.example.com/ovirt-engine/api/vms/` in a web browser on the Manager machine to list all VMs. Locate the VM you want to import and its corresponding VM ID. You do not need to specify a VM name or cluster name.

If you specify the source VM name, you must also specify the source cluster. Do not specify the source VM ID.

If you specify the source cluster, you must also specify the source VM name. Do not specify the source VM ID.

If you create a `ResourceMapping` CR, comment out the `mappings` section.
Specify the logical network and vNIC profile of the source VM.

Specify the OpenShift Virtualization network.

If storage mappings are specified in both the ResourceMapping and the VirtualMachineImport CRs, the VirtualMachineImport CR takes precedence.

Specify the source storage domain.

Specify the target storage class.

Specify **ReadWriteOnce**, **ReadWriteMany**, or **ReadOnlyMany**. If no access mode is specified, `{virt}` determines the correct volume access mode based on the Host → Migration mode setting of the RHV VM or on the virtual disk access mode:

- If the RHV VM migration mode is *Allow manual and automatic migration*, the default access mode is **ReadWriteMany**.

- If the RHV virtual disk access mode is **ReadOnly**, the default access mode is **ReadOnlyMany**.

- For all other settings, the default access mode is **ReadWriteOnce**.

Specify the source VM disk ID, for example, `8181ecc1-5db8-4193-9c92-3ddab3be7b05`. You can obtain the disk ID by entering `https://www.example.com/ovirt-engine/api/vms/vm23` in a web browser on the Manager machine and reviewing the VM details.

Specify the target storage class.

4. Follow the progress of the virtual machine import to verify that the import was successful:

```bash
$ oc get vmimports vm-import -n default
```

The output indicating a successful import resembles the following example:

**Example output**

```bash
... status:
  conditions:
  - lastHeartbeatTime: "2020-07-22T08:58:52Z"
    lastTransitionTime: "2020-07-22T08:58:52Z"
    message: Validation completed successfully
    reason: ValidationCompleted
    status: "True"
    type: Valid
  - lastHeartbeatTime: "2020-07-22T08:58:52Z"
    lastTransitionTime: "2020-07-22T08:58:52Z"
    message: 'VM specifies IO Threads: 1, VM has NUMA tune mode specified: interleave'
    reason: MappingRulesVerificationReportedWarnings
    status: "True"
    type: MappingRulesVerified
  - lastHeartbeatTime: "2020-07-22T08:58:56Z"
    lastTransitionTime: "2020-07-22T08:58:52Z"
```
7.15.4.4.1. Creating a config map for importing a VM

You can create a config map to map the Red Hat Virtualization (RHV) virtual machine operating system to an OpenShift Virtualization template if you want to override the default vm-import-controller mapping or to add additional mappings.

The default vm-import-controller config map contains the following RHV operating systems and their corresponding common OpenShift Virtualization templates.

Table 7.6. Operating system and template mapping

<table>
<thead>
<tr>
<th>RHV VM operating system</th>
<th>OpenShift Virtualization template</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhel_6_9_plus_ppc64</td>
<td>rhel6.9</td>
</tr>
<tr>
<td>rhel_6_ppc64</td>
<td>rhel6.9</td>
</tr>
<tr>
<td>rhel_6</td>
<td>rhel6.9</td>
</tr>
<tr>
<td>rhel_6x64</td>
<td>rhel6.9</td>
</tr>
<tr>
<td>rhel_7_ppc64</td>
<td>rhel7.7</td>
</tr>
<tr>
<td>rhel_7_s390x</td>
<td>rhel7.7</td>
</tr>
<tr>
<td>rhel_7x64</td>
<td>rhel7.7</td>
</tr>
<tr>
<td>rhel_8x64</td>
<td>rhel8.1</td>
</tr>
<tr>
<td>sles_11_ppc64</td>
<td>opensuse15.0</td>
</tr>
<tr>
<td>sles_11</td>
<td>opensuse15.0</td>
</tr>
<tr>
<td>sles_12_s390x</td>
<td>opensuse15.0</td>
</tr>
<tr>
<td>ubuntu_12_04</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>ubuntu_12_10</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>ubuntu_13_04</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>ubuntu_13_10</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>RHV VM operating system</td>
<td>OpenShift Virtualization template</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>ubuntu_14_04_ppc64</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>ubuntu_14_04</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>ubuntu_16_04_s390x</td>
<td>ubuntu18.04</td>
</tr>
<tr>
<td>windows_10</td>
<td>win10</td>
</tr>
<tr>
<td>windows_10x64</td>
<td>win10</td>
</tr>
<tr>
<td>windows_2003</td>
<td>win10</td>
</tr>
<tr>
<td>windows_2003x64</td>
<td>win10</td>
</tr>
<tr>
<td>windows_2008R2x64</td>
<td>win2k8</td>
</tr>
<tr>
<td>windows_2008</td>
<td>win2k8</td>
</tr>
<tr>
<td>windows_2008x64</td>
<td>win2k8</td>
</tr>
<tr>
<td>windows_2012R2x64</td>
<td>win2k12r2</td>
</tr>
<tr>
<td>windows_2012x64</td>
<td>win2k12r2</td>
</tr>
<tr>
<td>windows_2016x64</td>
<td>win2k16</td>
</tr>
<tr>
<td>windows_2019x64</td>
<td>win2k19</td>
</tr>
<tr>
<td>windows_7</td>
<td>win10</td>
</tr>
<tr>
<td>windows_7x64</td>
<td>win10</td>
</tr>
<tr>
<td>windows_8</td>
<td>win10</td>
</tr>
<tr>
<td>windows_8x64</td>
<td>win10</td>
</tr>
<tr>
<td>windows_xp</td>
<td>win10</td>
</tr>
</tbody>
</table>

**Procedure**

1. In a web browser, identify the REST API name of the RHV VM operating system by navigating to [URL](http://<RHV_Manager_FQDN>/ovirt-engine/api/vms/<VM_ID>). The operating system name appears in the `<os>` section of the XML output, as shown in the following example:

```xml
...<os>
```
2. View a list of the available OpenShift Virtualization templates:

   $ oc get templates --show-labels | tr ',' 'n' | grep os.template.kubevirt.io | sed -r 's#os.template.kubevirt.io/(.*)=.*#\1#' | sort -u

   **Example output**

   fedora31
   fedora32
   ...
   rhel8.1
   rhel8.2
   ...

3. If an OpenShift Virtualization template that matches the RHV VM operating system does not appear in the list of available templates, create a template with the OpenShift Virtualization web console.

4. Create a config map to map the RHV VM operating system to the OpenShift Virtualization template:

   $ cat <<EOF | oc create -f -
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: os-configmap
     namespace: default
   data:
     guestos2common: |
     "Red Hat Enterprise Linux Server": "rhel"
     "CentOS Linux": "centos"
     "Fedora": "fedora"
     "Ubuntu": "ubuntu"
     "openSUSE": "opensuse"
   osinfo2common: |
     "<rhv-operating-system>": "<vm-template>"
   EOF

   **Optional:** You can change the value of the **namespace** parameter.

   **Specify the REST API name of the RHV operating system and its corresponding VM template as shown in the following example.**

   **Config map example**

   $ cat <<EOF | oc apply -f -
   apiVersion: v1
   kind: ConfigMap
   metadata:
   EOF
5. Verify that the custom config map was created:

$ oc get cm -n default os-configmap -o yaml

6. Patch the `vm-import-controller-config` config map to apply the new config map:

$ oc patch configmap vm-import-controller-config -n openshift-cnv --patch '{
  "data": {
    "osConfigMap.name": "os-configmap",
    "osConfigMap.namespace": "default"
  }
}'

1. Update the namespace if you changed it in the config map.

7. Verify that the template appears in the OpenShift Virtualization web console:
   a. Click **Workloads → Virtualization** from the side menu.
   b. Click the **Virtual Machine Templates** tab and find the template in the list.

### 7.15.4.5. Troubleshooting a virtual machine import

#### 7.15.4.5.1. Logs

You can check the VM Import Controller pod log for errors.

**Procedure**

1. View the VM Import Controller pod name by running the following command:

   $ oc get pods -n <namespace> | grep import

   1. Specify the namespace of your imported virtual machine.

   **Example output**

   vm-import-controller-f66f7d-zqkz7 1/1  Running 0 4h49m

2. View the VM Import Controller pod log by running the following command:

   $ oc logs <vm-import-controller-f66f7d-zqkz7> -f -n <namespace>

   1. Specify the VM Import Controller pod name and the namespace.
7.15.4.5.2. Error messages

The following error message might appear:

- The following error message is displayed in the VM Import Controller pod log and the progress bar stops at 10% if the OpenShift Virtualization storage PV is not suitable:

  Failed to bind volumes: provisioning failed for PVC

  You must use a compatible storage class. The Cinder storage class is not supported.

7.15.4.5.3. Known issues

- If you are using Ceph RBD block-mode volumes and the available storage space is too small for the virtual disk, the import process bar stops at 75% for more than 20 minutes and the migration does not succeed. No error message is displayed in the web console. **BZ#1910019**

7.15.5. Importing a single VMware virtual machine or template

You can import a VMware vSphere 6.5, 6.7, or 7.0 VM or VM template into OpenShift Virtualization by using the VM Import wizard.

If you import a VM template, OpenShift Virtualization creates a virtual machine based on the template.

7.15.5.1. OpenShift Virtualization storage feature matrix

The following table describes the OpenShift Virtualization storage types that support VM import.

**Table 7.7. OpenShift Virtualization storage feature matrix**

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>VMware VM import</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenShift Container Storage: RBD block-mode volumes</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenShift Virtualization hostpath provisioner</td>
<td>Yes</td>
</tr>
<tr>
<td>Other multi-node writable storage</td>
<td>Yes [1]</td>
</tr>
<tr>
<td>Other single-node writable storage</td>
<td>Yes [2]</td>
</tr>
</tbody>
</table>

1. PVCs must request a ReadWriteMany access mode.
2. PVCs must request a ReadWriteOnce access mode.

7.15.5.2. Preparing a VDDK image

The import process uses the VMware Virtual Disk Development Kit (VDDK) to copy the VMware virtual disk.

You can download the VDDK SDK, create a VDDK image, upload the image to an image registry, and add it to the **v2v-vmware** config map.
You can configure either an internal OpenShift Container Platform image registry or a secure external image registry for the VDDK image. The registry must be accessible to your OpenShift Virtualization environment.

**NOTE**

Storing the VDDK image in a public registry might violate the terms of the VMware license.

### 7.15.5.2.1. Configuring an internal image registry

You can configure the internal OpenShift Container Platform image registry on bare metal by updating the Image Registry Operator configuration.

You can access the registry directly, from within the OpenShift Container Platform cluster, or externally, by exposing the registry with a route.

**Changing the image registry’s management state**

To start the image registry, you must change the Image Registry Operator configuration’s `managementState` from **Removed** to **Managed**.

**Procedure**

- Change `managementState` Image Registry Operator configuration from **Removed** to **Managed**. For example:

  ```bash
  $ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
  ```

**Configuring registry storage for bare metal and other manual installations**

As a cluster administrator, following installation you must configure your registry to use storage.

**Prerequisites**

- Cluster administrator permissions.
- A cluster that uses manually-provisioned Red Hat Enterprise Linux CoreOS (RHCOS) nodes, such as bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

**IMPORTANT**

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

**Procedure**

1. To configure your registry to use storage, change the `spec.storage.pvc` in the `configs.imageregistry/cluster` resource.
2. Verify that you do not have a registry pod:

   $ oc get pod -n openshift-image-registry

3. Check the registry configuration:

   $ oc edit configs.imageregistry.operator.openshift.io

   **Example output**

   ```yaml
   storage:
   pvc:
   claim:
   ```

   Leave the `claim` field blank to allow the automatic creation of an `image-registry-storage` PVC.

4. Check the `clusterooperator` status:

   $ oc get clusterooperator image-registry

5. Ensure that your registry is set to managed to enable building and pushing of images.

   - Run:

     $ oc edit configs.imageregistry/cluster

     Then, change the line

     ```yaml
     managementState: Removed
     ```

     to

     ```yaml
     managementState: Managed
     ```

**Accessing registry directly from the cluster**

You can access the registry from inside the cluster.

**Procedure**

Access the registry from the cluster by using internal routes:

1. Access the node by getting the node’s address:
2. To enable access to tools such as `oc` and `podman` on the node, run the following command:

   ```sh
   sh-4.2# chroot /host
   ```

3. Log in to the container image registry by using your access token:

   ```sh
   sh-4.2# oc login -u kubeadmin -p <password_from_install_log> https://api-int.<cluster_name>.<base_domain>:6443
   sh-4.2# podman login -u kubeadmin -p $(oc whoami -t) image-registry.openshift-image-registry.svc:5000
   ```

   You should see a message confirming login, such as:

   ```sh
   Login Succeeded!
   ```

   **NOTE**

   You can pass any value for the user name; the token contains all necessary information. Passing a user name that contains colons will result in a login failure.

   Since the Image Registry Operator creates the route, it will likely be similar to `default-route-openshift-image-registry.<cluster_name>`.

4. Perform `podman pull` and `podman push` operations against your registry:

   **IMPORTANT**

   You can pull arbitrary images, but if you have the `system:registry` role added, you can only push images to the registry in your project.

   In the following examples, use:

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;registry_ip&gt;</td>
<td>172.30.124.220</td>
</tr>
<tr>
<td>&lt;port&gt;</td>
<td>5000</td>
</tr>
<tr>
<td>&lt;project&gt;</td>
<td>openshift</td>
</tr>
<tr>
<td>&lt;image&gt;</td>
<td>image</td>
</tr>
<tr>
<td>&lt;tag&gt;</td>
<td>omitted (defaults to latest)</td>
</tr>
</tbody>
</table>
a. Pull an arbitrary image:

```
sh-4.2# podman pull name.io/image
```

b. Tag the new image with the form `<registry_ip>:<port>/<project>/<image>`. The project name must appear in this pull specification for OpenShift Container Platform to correctly place and later access the image in the registry:

```
sh-4.2# podman tag name.io/image image-registry.openshift-image-registry.svc:5000/openshift/image
```

**NOTE**

You must have the `system:image-builder` role for the specified project, which allows the user to write or push an image. Otherwise, the `podman push` in the next step will fail. To test, you can create a new project to push the image.

c. Push the newly tagged image to your registry:

```
sh-4.2# podman push image-registry.openshift-image-registry.svc:5000/openshift/image
```

**Exposing a secure registry manually**

Instead of logging in to the OpenShift Container Platform registry from within the cluster, you can gain external access to it by exposing it with a route. This allows you to log in to the registry from outside the cluster using the route address, and to tag and push images to an existing project by using the route host.

**Prerequisites:**

- The following prerequisites are automatically performed:
  - Deploy the Registry Operator.
  - Deploy the Ingress Operator.

**Procedure**

You can expose the route by using `DefaultRoute` parameter in the `configs.imageregistry.operator.openshift.io` resource or by using custom routes.

To expose the registry using `DefaultRoute`:

1. Set `DefaultRoute` to `True`:

   ```
   $ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":
   {"defaultRoute":true}}' --type=merge
   ```

2. Log in with `podman`:

   ```
   $ HOST=$(oc get route default-route -n openshift-image-registry --template="{{ .spec.host }}")
   $ podman login -u kubeadmin -p $(oc whoami -t) --tls-verify=false $HOST
   ```
--tls-verify=false is needed if the cluster’s default certificate for routes is untrusted. You can set a custom, trusted certificate as the default certificate with the Ingress Operator.

To expose the registry using custom routes:

1. Create a secret with your route’s TLS keys:

   ```bash
   $ oc create secret tls public-route-tls 
   -n openshift-image-registry 
   --cert=/path/to/tls.crt 
   --key=/path/to/tls.key
   
   This step is optional. If you do not create a secret, the route uses the default TLS configuration from the Ingress Operator.
   
2. On the Registry Operator:

   ```yaml
   spec:
   routes:
   - name: public-routes
     hostname: myregistry.mycorp.organization
     secretName: public-route-tls
   ...```

   **NOTE**
   
   Only set `secretName` if you are providing a custom TLS configuration for the registry’s route.

### 7.15.5.2.2. Configuring an external image registry

If you use an external image registry for the VDDK image, you can add the external image registry’s certificate authorities to the OpenShift Container Platform cluster.

Optionally, you can create a pull secret from your Docker credentials and add it to your service account.

**Adding certificate authorities to the cluster**

You can add certificate authorities (CA) to the cluster for use when pushing and pulling images with the following procedure.

**Prerequisites**

- You must have cluster administrator privileges.
- You must have access to the public certificates of the registry, usually a `hostname/ca.crt` file located in the `/etc/docker/certs.d/` directory.

**Procedure**

1. Create a ConfigMap in the `openshift-config` namespace containing the trusted certificates for the registries that use self-signed certificates. For each CA file, ensure the key in the ConfigMap is the hostname of the registry in the `hostname[..port]` format:
Allowing pods to reference images from other secured registries

The `.dockercfg` file for Docker clients is a Docker credentials file that stores your authentication information if you have previously logged into a secured or insecure registry.

To pull a secured container image that is not from OpenShift Container Platform's internal registry, you must create a pull secret from your Docker credentials and add it to your service account.

**Procedure**

- If you already have a `.dockercfg` file for the secured registry, you can create a secret from that file by running:
  
  ```bash
  $ oc create secret generic <pull_secret_name> 
  --from-file=.dockercfg=<path/to/.dockercfg> 
  --type=kubernetes.io/dockercfg
  
  $ oc create secret generic <pull_secret_name> 
  --from-file=.dockerconfigjson=<path/to/.docker/config.json> 
  --type=kubernetes.io/dockerconfigjson
  
  $ oc create secret docker-registry <pull_secret_name> 
  --docker-server=<registry_server> 
  --docker-username=<user_name> 
  --docker-password=<password> 
  --docker-email=<email>
  
  $ oc secrets link default <pull_secret_name> --for=pull
  ``

Prerequisites

You can download the VMware Virtual Disk Development Kit (VDDK), build a VDDK image, and push the VDDK image to your image registry. You then add the VDDK image to the `v2v-vmware` config map.

7.15.5.2.3. Creating and using a VDDK image

You can download the VMware Virtual Disk Development Kit (VDDK), build a VDDK image, and push the VDDK image to your image registry. You then add the VDDK image to the `v2v-vmware` config map.
You must have access to an OpenShift Container Platform internal image registry or a secure external registry.

Procedure

1. Create and navigate to a temporary directory:

   ```
   $ mkdir /tmp/<dir_name> && cd /tmp/<dir_name>
   ```

2. In a browser, navigate to VMware code and click SDKs.

3. Under Compute Virtualization, click Virtual Disk Development Kit (VDDK)

4. Select the VDDK version that corresponds to your VMware vSphere version, for example, VDDK 7.0 for vSphere 7.0, click Download, and then save the VDDK archive in the temporary directory.

5. Extract the VDDK archive:

   ```
   $ tar -xzf VMware-vix-disklib-<version>.x86_64.tar.gz
   ```

6. Create a Dockerfile:

   ```
   $ cat > Dockerfile <<EOF
   FROM busybox:latest
   COPY vmware-vix-disklib-distrib /vmware-vix-disklib-distrib
   RUN mkdir -p /opt
   ENTRYPOINT ["cp", "-r", "/vmware-vix-disklib-distrib", "/opt"]
   EOF
   ```

7. Build the image:

   ```
   $ podman build . -t <registry_route_or_server_path>/vddk:<tag>  
   ```

   Specify your image registry:

   - For an internal OpenShift Container Platform registry, use the internal registry route, for example, `image-registry.openshift-image-registry.svc:5000/openshift/vddk:<tag>`.
   - For an external registry, specify the server name, path, and tag, for example, `server.example.com:5000/vddk:<tag>`.

8. Push the image to the registry:

   ```
   $ podman push <registry_route_or_server_path>/vddk:<tag>
   ```

9. Ensure that the image is accessible to your OpenShift Virtualization environment.

10. Edit the `v2v-vmware` config map in the `openshift-cnv` project:

    ```
    $ oc edit configmap v2v-vmware -n openshift-cnv
    ```
7.15.5.3. Importing a virtual machine with the VM Import wizard

You can import a single virtual machine with the VM Import wizard.

You can also import a VM template. If you import a VM template, OpenShift Virtualization creates a virtual machine based on the template.

Prerequisites

- You must have admin user privileges.
- The VMware Virtual Disk Development Kit (VDDK) image must be in an image registry that is accessible to your OpenShift Virtualization environment.
- The VDDK image must be added to the `v2v-vmware` config map.
- The VM must be powered off.
- Virtual disks must be connected to IDE or SCSI controllers. If virtual disks are connected to a SATA controller, you can change them to IDE controllers and then migrate the VM.
- The OpenShift Virtualization local and shared persistent storage classes must support VM import.
- The OpenShift Virtualization storage must be large enough to accommodate the virtual disk.

**WARNING**

If you are using Ceph RBD block-mode volumes, the storage must be large enough to accommodate the virtual disk. If the disk is too large for the available storage, the import process fails and the PV that is used to copy the virtual disk is not released. You will not be able to import another virtual machine or to clean up the storage because there are insufficient resources to support object deletion. To resolve this situation, you must add more object storage devices to the storage back end.

- The OpenShift Virtualization egress network policy must allow the following traffic:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Protocol</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware ESXi hosts</td>
<td>TCP</td>
<td>443</td>
</tr>
<tr>
<td>VMware ESXi hosts</td>
<td>TCP</td>
<td>902</td>
</tr>
</tbody>
</table>
### Procedure

1. In the web console, click **Workloads → Virtual Machines**.

2. Click **Create Virtual Machine** and select **Import with Wizard**.

3. Select **VMware** from the **Provider** list.

4. Select **Connect to New Instance** or a saved vCenter instance.
   - If you select **Connect to New Instance**, enter the **vCenter hostname**, **Username**, and **Password**.
   - If you select a saved vCenter instance, the wizard connects to the vCenter instance using the saved credentials.

5. Click **Check and Save** and wait for the connection to complete.

   **NOTE**

   The connection details are stored in a secret. If you add a provider with an incorrect hostname, user name, or password, click **Workloads → Secrets** and delete the provider secret.

6. Select a virtual machine or a template.

7. Click **Next**.

8. In the **Review** screen, review your settings.

9. Click **Edit** to update the following settings:
   - **General**:
     - Description
     - Operating System
     - Flavor
     - Memory
     - CPUs
     - Workload Profile
   - **Networking**:
     - Name
     - Model
- Network
- Type
- MAC Address

- **Storage**: Click the Options menu of the VM disk and select Edit to update the following fields:
  - Name
  - **Source**: For example, Import Disk.
  - Size
  - Interface
  - **Storage Class**: Select NFS or ocs-storagecluster-ceph-rbd (ceph-rbd).
    If you select ocs-storagecluster-ceph-rbd, you must set the **Volume Mode** of the disk to **Block**.
    Other storage classes might work, but they are not officially supported.
  - **Advanced → Volume Mode**: Select Block.
  - **Advanced → Access Mode**

- **Advanced → Cloud-init**
  - **Form**: Enter the **Hostname** and **Authenticated SSH Keys**
  - **Custom script**: Enter the **cloud-init** script in the text field.

- **Advanced → Virtual Hardware**: You can attach a virtual CD-ROM to the imported virtual machine.

10. Click **Import** or **Review and Import**, if you have edited the import settings.
    A **Successfully created virtual machine** message and a list of resources created for the virtual machine are displayed. The virtual machine appears in **Workloads → Virtual Machines**.

### Virtual machine wizard fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td></td>
<td>Template from which to create the virtual machine. Selecting a template will automatically complete other fields.</td>
</tr>
<tr>
<td>Source</td>
<td>PXE</td>
<td>Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>URL</td>
<td></td>
<td>Provision virtual machine from an image available from an HTTP or S3 endpoint.</td>
</tr>
<tr>
<td>Container</td>
<td></td>
<td>Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: kubevirt/cirros-registry-disk-demo.</td>
</tr>
<tr>
<td>Disk</td>
<td></td>
<td>Provision virtual machine from a disk.</td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
<td>The primary operating system that is selected for the virtual machine.</td>
</tr>
<tr>
<td>Flavor</td>
<td>small, medium, large, tiny, Custom</td>
<td>Presets that determine the amount of CPU and memory allocated to the virtual machine. The presets displayed for Flavor are determined by the operating system.</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td>Size in GiB of the memory allocated to the virtual machine.</td>
</tr>
<tr>
<td>CPUs</td>
<td></td>
<td>The amount of CPU allocated to the virtual machine.</td>
</tr>
<tr>
<td>Workload Profile</td>
<td>High Performance</td>
<td>A virtual machine configuration that is optimized for high-performance workloads.</td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td>A profile optimized to run server workloads.</td>
</tr>
<tr>
<td>Desktop</td>
<td></td>
<td>A virtual machine configuration for use on a desktop.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name can contain lowercase 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 uppercase letters, spaces, periods (.), or special characters.</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td>Optional description field.</td>
</tr>
<tr>
<td>Start virtual machine on creation</td>
<td></td>
<td>Select to automatically start the virtual machine upon creation.</td>
</tr>
</tbody>
</table>

**Cloud-init fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Sets a specific hostname for the virtual machine.</td>
</tr>
<tr>
<td>Authenticated SSH Keys</td>
<td>The user's public key that is copied to ~/.ssh/authorized_keys on the virtual machine.</td>
</tr>
<tr>
<td>Custom script</td>
<td>Replaces other options with a field in which you paste a custom cloud-init script.</td>
</tr>
</tbody>
</table>

**Networking fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are e1000e and virtio.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
<tr>
<td>Type</td>
<td>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.</td>
</tr>
<tr>
<td>MAC Address</td>
<td>MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.</td>
</tr>
</tbody>
</table>
### Storage fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Select a blank disk for the virtual machine or choose from the options available: <strong>URL, Container, Attach Cloned Disk, or Attach Disk</strong>. To select an existing disk and attach it to the virtual machine, choose <strong>Attach Cloned Disk or Attach Disk</strong> from a list of available persistent volume claims (PVCs).</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the disk. The name can contain lowercase 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 uppercase letters, spaces, or special characters.</td>
</tr>
<tr>
<td>Size (GiB)</td>
<td>Size, in GiB, of the disk.</td>
</tr>
<tr>
<td>Interface</td>
<td>Type of disk device. Supported interfaces are <strong>virtIO</strong>, <strong>SATA</strong>, and <strong>SCSI</strong>.</td>
</tr>
<tr>
<td>Storage Class</td>
<td>The storage class that is used to create the disk.</td>
</tr>
<tr>
<td>Advanced → Volume Mode</td>
<td>Defines whether the persistent volume uses a formatted file system or raw block state. Default is <strong>Filesystem</strong>.</td>
</tr>
<tr>
<td>Advanced → Access Mode</td>
<td>Access mode of the persistent volume. Supported access modes are <strong>ReadWriteOnce</strong>, <strong>ReadOnlyMany</strong>, and <strong>ReadWriteMany</strong>.</td>
</tr>
</tbody>
</table>

### Advanced storage settings

The following advanced storage settings are available for **Blank, Import via URL, and Clone existing PVC disks**. These parameters are optional. If you do not specify these parameters, the system uses the default values from the **kubevirt-storage-class-defaults** config map.

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Mode</td>
<td>Filesystem</td>
<td>Stores the virtual disk on a file system-based volume.</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>Stores the virtual disk directly on the block volume. Only use <strong>Block</strong> if the underlying storage supports it.</td>
</tr>
<tr>
<td>Access Mode</td>
<td>Single User (RWO)</td>
<td>The disk can be mounted as read/write by a single node.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Shared Access (RWX)</td>
<td></td>
<td>The disk can be mounted as read/write by many nodes.</td>
</tr>
<tr>
<td>Read Only (ROX)</td>
<td></td>
<td>The disk can be mounted as read-only by many nodes.</td>
</tr>
</tbody>
</table>

**NOTE**

This is required for some features, such as live migration of virtual machines between nodes.

7.15.5.3.1. Updating the NIC name of an imported virtual machine

You must update the NIC name of a virtual machine that you imported from VMware to conform to OpenShift Virtualization naming conventions.

**Procedure**

1. Log in to the virtual machine.

2. Navigate to the `/etc/sysconfig/network-scripts` directory.

3. Rename the network configuration file:

   ```bash
   $ mv vmnic0 ifcfg-eth0
   ```

   ![](image) The first network configuration file is named `ifcfg-eth0`. Additional network configuration files are numbered sequentially, for example, `ifcfg-eth1`, `ifcfg-eth2`.

4. Update the `NAME` and `DEVICE` parameters in the network configuration file:

   ```bash
   NAME=eth0
   DEVICE=eth0
   ```

5. Restart the network:

   ```bash
   $ systemctl restart network
   ```

7.15.5.4. Troubleshooting a virtual machine import

7.15.5.4.1. Logs

You can check the V2V Conversion pod log for errors.

**Procedure**

1. View the V2V Conversion pod name by running the following command:
Specify the namespace of your imported virtual machine.

Example output

```
kubevirt-v2v-conversion-f66f7d-zqkz7  1/1     Running     0     4h49m
```

2. View the V2V Conversion pod log by running the following command:

```
$ oc logs <kubevirt-v2v-conversion-f66f7d-zqkz7> -f -n <namespace>
```

Specify the VM Conversion pod name and the namespace.

7.15.5.4.2. Error messages

The following error messages might appear:

- If the VMware VM is not shut down before import, the imported virtual machine displays the error message, **Readiness probe failed** in the OpenShift Container Platform console and the V2V Conversion pod log displays the following error message:

  ```
  INFO - have error: ('virt-v2v error: internal error: invalid argument: libvirt domain 'v2v_migration_vm_1' is running or paused. It must be shut down in order to perform virt-v2v conversion'),)
  ```

- The following error message is displayed in the OpenShift Container Platform console if a non-admin user tries to import a VM:

  ```
  Could not load config map vmware-to-kubevirt-os in kube-public namespace
  Restricted Access: configmaps "vmware-to-kubevirt-os" is forbidden: User cannot get resource "configmaps" in API group "" in the namespace "kube-public"
  ```

Only an admin user can import a VM.

7.16. CLONING VIRTUAL MACHINES

7.16.1. Enabling user permissions to clone data volumes across namespaces

The isolating nature of namespaces means that users cannot by default clone resources between namespaces.

To enable a user to clone a virtual machine to another namespace, a user with the **cluster-admin** role must create a new cluster role. Bind this cluster role to a user to enable them to clone virtual machines to the destination namespace.

7.16.1.1. Prerequisites

- Only a user with the **cluster-admin** role can create cluster roles.
7.16.1.2. About data volumes

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

7.16.1.3. Creating RBAC resources for cloning data volumes

Create a new cluster role that enables permissions for all actions for the datavolumes resource.

Procedure

1. Create a ClusterRole manifest:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <datavolume-cloner>  # 1
rules:
- apiGroups: ["cdi.kubevirt.io"]
  resources: ["datavolumes/source"]
  verbs: ["*"]
```

1. Unique name for the cluster role.

2. Create the cluster role in the cluster:

```
$ oc create -f <datavolume-cloner.yaml>  # 1
```

1. The file name of the ClusterRole manifest created in the previous step.

3. Create a RoleBinding manifest that applies to both the source and destination namespaces and references the cluster role created in the previous step.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: <allow-clone-to-user>  # 1
  namespace: <Source namespace>  # 2
subjects:
- kind: ServiceAccount
  name: default
  namespace: <Destination namespace>  # 3
roleRef:
  kind: ClusterRole
  name: datavolume-cloner  # 4
  apiGroup: rbac.authorization.k8s.io
```

1. Unique name for the role binding.

2. The namespace for the source data volume.
3. The namespace to which the data volume is cloned.

4. The name of the cluster role created in the previous step.

4. Create the role binding in the cluster:

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

   The file name of the **RoleBinding** manifest created in the previous step.

---

### 7.16.2. Cloning a virtual machine disk into a new data volume

You can clone the persistent volume claim (PVC) of a virtual machine disk into a new data volume by referencing the source PVC in your data volume configuration file.

**WARNING**

Cloning operations between different volume modes are not supported. The `volumeMode` values must match in both the source and target specifications.

For example, if you attempt to clone from a persistent volume (PV) with `volumeMode: Block` to a PV with `volumeMode: Filesystem`, the operation fails with an error message.

---

#### 7.16.2.1. Prerequisites

- Users need **additional permissions** to clone the PVC of a virtual machine disk into another namespace.

#### 7.16.2.2. About data volumes

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

#### 7.16.2.3. Cloning the persistent volume claim of a virtual machine disk into a new data volume

You can clone a persistent volume claim (PVC) of an existing virtual machine disk into a new data volume. The new data volume can then be used for a new virtual machine.

**NOTE**

When a data volume is created independently of a virtual machine, the lifecycle of the data volume is independent of the virtual machine. If the virtual machine is deleted, neither the data volume 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 CLI (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 data volume that specifies the name of the new data volume, the name and namespace of the source PVC, and the size of the new data volume. For example:

   ```yaml
   apiVersion: cdi.kubevirt.io/v1beta1
   kind: DataVolume
   metadata:
     name: <cloner-datavolume>  
   spec:
     source:
       pvc:
         namespace: "<source-namespace>"  
         name: "<my-favorite-vm-disk>"  
       pvc:
         accessModes:
           - ReadWriteOnce
         resources:
           requests:
             storage: <2Gi>  
   
1. The name of the new data volume.
2. The namespace where the source PVC exists.
3. The name of the source PVC.
4. The size of the new data volume. You must allocate enough space, or the cloning operation fails. The size must be the same as or larger than the source PVC.

3. Start cloning the PVC by creating the data volume:

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

   **NOTE**

   Data volumes prevent a virtual machine from starting before the PVC is prepared, so you can create a virtual machine that references the new data volume while the PVC clones.

7.16.2.4. Template: Data volume clone configuration file
7.16.2.5. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2**</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2*</td>
<td>✓ QCOW2*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ XZ*</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>□ XZ</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ XZ</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>□ XZ</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
</tbody>
</table>

✓ Supported operation

☐ Unsupported operation

* Requires scratch space

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

7.16.3. Cloning a virtual machine by using a data volume template

You can create a new virtual machine by cloning the persistent volume claim (PVC) of an existing VM. By including a `dataVolumeTemplate` in your virtual machine configuration file, you create a new data volume from the original PVC.
WARNING

Cloning operations between different volume modes are not supported. The `volumeMode` values must match in both the source and target specifications.

For example, if you attempt to clone from a persistent volume (PV) with `volumeMode: Block` to a PV with `volumeMode: Filesystem`, the operation fails with an error message.

7.16.3.1. Prerequisites

- Users need additional permissions to clone the PVC of a virtual machine disk into another namespace.

7.16.3.2. About data volumes

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

7.16.3.3. Creating a new virtual machine from a cloned persistent volume claim by using a data volume template

You can create a virtual machine that clones the persistent volume claim (PVC) of an existing virtual machine into a data volume. Reference a `dataVolumeTemplate` in the virtual machine manifest and the `source` PVC is cloned to a data volume, which is then automatically used for the creation of the virtual machine.

**NOTE**

When a data volume is created as part of the data volume template of a virtual machine, the lifecycle of the data volume is then dependent on the virtual machine. If the virtual machine is deleted, the data volume 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 CLI (`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` data volume called `favorite-clone` is created from `my-favorite-vm-disk`. 
For example:

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm-dv-clone
name: vm-dv-clone
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
                resources:
                  requests:
                    storage: 2Gi
                source:
                  pvc:
                    namespace: "source-namespace"
                    name: "my-favorite-vm-disk"
            - dataVolume:
                name: favorite-clone
        dataVolumeTemplates:
          - metadata:
              name: favorite-clone
            spec:
              pvc:
                accessModes:
                -ReadWriteOnce
              resources:
                requests:
                  storage: 2Gi
```

3. Create the virtual machine with the PVC-cloned data volume:

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

### 7.16.3.4. Template: Data volume virtual machine configuration file

**example-dv-vm.yaml**

```yaml
apiVersion: kubevirt.io/v1alpha3
```
7.16.3.5. CDI supported operations matrix

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

The **HTTP** source of the image you want to import, if applicable.
<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2 ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2** ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2 ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2* □ GZ □ XZ</td>
<td>✓ QCOW2* ✓ GZ* ✓ XZ*</td>
</tr>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW ✓ GZ ✓ XZ</td>
<td>✓ RAW ✓ GZ ✓ XZ</td>
<td>✓ RAW ✓ GZ ✓ XZ</td>
<td>✓ RAW* □ GZ □ XZ</td>
<td>✓ RAW* ✓ GZ* ✓ XZ*</td>
</tr>
</tbody>
</table>

Supported operation

Unsupported operation

* Requires scratch space

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

### 7.16.4. Cloning a virtual machine disk into a new block storage data volume

You can clone the persistent volume claim (PVC) of a virtual machine disk into a new block data volume by referencing the source PVC in your data volume configuration file.

**WARNING**

Cloning operations between different volume modes are not supported. The `volumeMode` values must match in both the source and target specifications.

For example, if you attempt to clone from a persistent volume (PV) with `volumeMode: Block` to a PV with `volumeMode: Filesystem`, the operation fails with an error message.

### 7.16.4.1. Prerequisites

- Users need additional permissions to clone the PVC of a virtual machine disk into another namespace.

### 7.16.4.2. About data volumes

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

### 7.16.4.3. About block persistent volumes
A block persistent volume (PV) is a PV that is backed by a raw block device. These volumes do not have a file system and can provide performance benefits for virtual machines by reducing overhead.

Raw block volumes are provisioned by specifying `volumeMode: Block` in the PV and persistent volume claim (PVC) specification.

### 7.16.4.4. Creating a local block persistent volume

Create a local block persistent volume (PV) on a node by populating a file and mounting it as a loop device. You can then reference this loop device in a PV manifest as a `Block` volume and use it as a block device for a virtual machine image.

**Procedure**

1. Log in as `root` to the node on which to create the local PV. This procedure uses `node01` for its examples.

2. Create a file and populate it with null characters so that it can be used as a block device. The following example creates a file `loop10` with a size of 2Gb (20 100Mb blocks):

   ```
   $ dd if=/dev/zero of=loop10 bs=100M count=20
   $$
   $$
   ```

3. Mount the `loop10` file as a loop device.

   ```
   $ losetup /dev/loop10 d3 loop10
   $$
   ```

   1. **File path where the loop device is mounted.**
   2. **The file created in the previous step to be mounted as the loop device.**

4. Create a `PersistentVolume` manifest that references the mounted loop device.

   ```yaml
   kind: PersistentVolume
   apiVersion: v1
   metadata:
     name: <local-block-pv10>
     annotations:
   spec:
     local:
       path: /dev/loop10
     capacity:
       storage: <2Gi>
     volumeMode: Block
     storageClassName: local
     accessModes:
     - ReadWriteOnce
     persistentVolumeReclaimPolicy: Delete
   nodeAffinity:
     required:
       nodeSelectorTerms:
       - matchExpressions:
         - key: kubernetes.io/hostname
   ```
The path of the loop device on the node.

2. Specifies it is a block PV.

3. Optional: Set a storage class for the PV. If you omit it, the cluster default is used.

4. The node on which the block device was mounted.

5. Create the block PV.

   # oc create -f <local-block-pv10.yaml>

1. The file name of the persistent volume created in the previous step.

**7.16.4.5. Cloning the persistent volume claim of a virtual machine disk into a new data volume**

You can clone a persistent volume claim (PVC) of an existing virtual machine disk into a new data volume. The new data volume can then be used for a new virtual machine.

**NOTE**

When a data volume is created independently of a virtual machine, the lifecycle of the data volume is independent of the virtual machine. If the virtual machine is deleted, neither the data volume 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 CLI (`oc`).
- At least one available block persistent volume (PV) that is the same size as or larger than the source PVC.

**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 data volume that specifies the name of the new data volume, the name and namespace of the source PVC, `volumeMode: Block` so that an available block PV is used, and the size of the new data volume.
   
   For example:
   ```yaml
   apiVersion: cdi.kubevirt.io/v1beta1
   kind: DataVolume
   metadata:
   ```
The name of the new data volume.

The namespace where the source PVC exists.

The name of the source PVC.

The size of the new data volume. You must allocate enough space, or the cloning operation fails. The size must be the same as or larger than the source PVC.

Specifies that the destination is a block PV

3. Start cloning the PVC by creating the data volume:

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

**NOTE**

Data volumes prevent a virtual machine from starting before the PVC is prepared, so you can create a virtual machine that references the new data volume while the PVC clones.

### 7.16.4.6. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2**</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2*</td>
<td>✓ QCOW2*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
</tbody>
</table>
GREEN SUPPORTED OPERATION

RED UNSUPPORTED OPERATION

* Requires scratch space

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

7.17. VIRTUAL MACHINE NETWORKING

7.17.1. Using the default pod network for virtual machines

You can use the default pod network with OpenShift Virtualization. To do so, you must use the \textit{masquerade} binding method. It is the only recommended binding method for use with the default pod network. Do not use \textit{masquerade} mode with non-default networks.

For secondary networks, use the \textit{bridge} binding method.

\begin{quote}
\textbf{NOTE}

The \textit{KubeMacPool} component provides a MAC address pool service for virtual machine NICs in designated namespaces. It is not enabled by default. \textit{Enable a MAC address pool in a namespace} by applying the KubeMacPool label to that namespace.
\end{quote}

7.17.1.1. 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.

\textbf{Prerequisites}

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

\textbf{Procedure}

1. Edit the \texttt{interfaces} spec of your virtual machine configuration file:

\begin{verbatim}
kind: VirtualMachine
spec:
  domain:
    devices:
      interfaces:
        - name: red
          masquerade: {}
          ports:
            - port: 80
        networks:
          - name: red
          pod: {}
\end{verbatim}
1. Connect using masquerade mode
2. Allow incoming traffic on port 80

2. Create the virtual machine:

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

### 7.17.1.2. Selecting binding method

If you create a virtual machine from the OpenShift Virtualization web console wizard, select the required binding method from the Networking screen.

### 7.17.1.2.1. Networking fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are e1000e and virtio.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
<tr>
<td>Type</td>
<td>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.</td>
</tr>
<tr>
<td>MAC Address</td>
<td>MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.</td>
</tr>
</tbody>
</table>

### 7.17.1.3. Virtual machine configuration examples for the default network

#### 7.17.1.3.1. Template: Virtual machine configuration file

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

OpenShift Container Platform 4.6 OpenShift Virtualization
7.17.1.3.2. Template: Windows virtual machine instance configuration file

```yaml
apiVersion: kubevirt.io/v1alpha3
gkind: VirtualMachineInstance
metadata:
  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: {}

CHAPTER 7. VIRTUAL MACHINES

7.17.1.3.2. Template: Windows virtual machine instance configuration file

```
7.17.1.4. Creating a service from a virtual machine

Create a service from a running virtual machine by first creating a Service object to expose the virtual machine.

The ClusterIP service type exposes the virtual machine internally, within the cluster. The NodePort or LoadBalancer service types expose the virtual machine externally, outside of the cluster.

This procedure presents an example of how to create, connect to, and expose a Service object of type: ClusterIP as a virtual machine-backed service.

NOTE

ClusterIP is the default service type, if the service type is not specified.

Procedure

1. Edit the virtual machine YAML as follows:

```yaml
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
```

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: vm-ephemeral
  namespace: example-namespace
spec:
  running: false
  template:
    metadata:
      labels:
```

```yaml
```
Add the label `special: key` in the `spec.template.metadata.labels` section.

```
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
```

1. Add the label `special: key` in the `spec.template.metadata.labels` section.

**NOTE**

Labels on a virtual machine are passed through to the pod. The labels on the `VirtualMachine` configuration, for example `special: key`, must match the labels in the `Service YAML` selector attribute, which you create later in this procedure.

2. Save the virtual machine YAML to apply your changes.

3. Edit the `Service YAML` to configure the settings necessary to create and expose the `Service` object:

```
apiVersion: v1
kind: Service
metadata:
  name: vmservice 1
  namespace: example-namespace 2
spec:
  ports:
    - port: 27017
      protocol: TCP
    targetPort: 22 3
```

1. Add the label `special: key` in the `spec.template.metadata.labels` section.

2. **NOTE**

Labels on a virtual machine are passed through to the pod. The labels on the `VirtualMachine` configuration, for example `special: key`, must match the labels in the `Service YAML` selector attribute, which you create later in this procedure.

3. Save the virtual machine YAML to apply your changes.

4. Edit the `Service YAML` to configure the settings necessary to create and expose the `Service` object:
1. Specify the **name** of the service you are creating and exposing.

2. Specify **namespace** in the **metadata** section of the **Service** YAML that corresponds to the **namespace** you specify in the virtual machine YAML.

3. Add **targetPort: 22**, exposing the service on SSH port **22**.

4. In the **spec** section of the **Service** YAML, add **special: key** to the **selector** attribute, which corresponds to the **labels** you added in the virtual machine YAML configuration file.

5. In the **spec** section of the **Service** YAML, add **type: ClusterIP** for a ClusterIP service. To create and expose other types of services externally, outside of the cluster, such as **NodePort** and **LoadBalancer**, replace **type: ClusterIP** with **type: NodePort** or **type: LoadBalancer**, as appropriate.

4. Save the **Service** YAML to store the service configuration.

5. Create the **ClusterIP** service:

   ```bash
   $ oc create -f <service_name>.yaml
   ```

6. Start the virtual machine. If the virtual machine is already running, restart it.

7. Query the **Service** object to verify it is available and is configured with type **ClusterIP**.

**Verification**

- Run the **oc get service** command, specifying the **namespace** that you reference in the virtual machine and **Service** YAML files.

  ```bash
  $ oc get service -n example-namespace
  ```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmservice</td>
<td>ClusterIP</td>
<td>172.30.3.149</td>
<td>&lt;none&gt;</td>
<td>27017/TCP</td>
<td>2m</td>
</tr>
</tbody>
</table>

- As shown from the output, **vmservice** is running.
- The **TYPE** displays as **ClusterIP**, as you specified in the **Service** YAML.

8. Establish a connection to the virtual machine that you want to use to back your service. Connect from an object inside the cluster, such as another virtual machine.

   a. Edit the virtual machine YAML as follows:

   ```yaml
   apiVersion: kubevirt.io/v1alpha3
   kind: VirtualMachine
   metadata:
     name: vm-connect
   ```
Run the `oc create` command to create a second virtual machine, where `file.yaml` is the name of the virtual machine YAML:

```yaml
namespace: example-namespace
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
```

b. Run the `oc create` command to create a second virtual machine, where `file.yaml` is the name of the virtual machine YAML:

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

c. Start the virtual machine.

d. Connect to the virtual machine by running the following `virtctl` command:

```
$ virtctl -n example-namespace console <new-vm-name>
```

NOTE

For service type `LoadBalancer`, use the `vinagre` client to connect your virtual machine by using the public IP and port. External ports are dynamically allocated when using service type `LoadBalancer`.

e. Run the `ssh` command to authenticate the connection, where `172.30.3.149` is the ClusterIP of the service and `fedora` is the user name of the virtual machine:

```
$ ssh fedora@172.30.3.149 -p 27017
```
Verification

- You receive the command prompt of the virtual machine backing the service you want to expose. You now have a service backed by a running virtual machine.

Additional resources

- Configuring external IPs

7.17.2. Attaching a virtual machine to multiple networks

You can attach virtual machines to multiple networks by using Linux bridges. You can also import virtual machines with existing workloads that depend on access to multiple interfaces.

To attach a virtual machine to an additional network:

1. Configure a bridge network attachment definition for a namespace in the web console or CLI.

2. Attach the virtual machine to the network attachment definition by using either the web console or the CLI:
   - In the web console, create a NIC for a new or existing virtual machine.
   - In the CLI, include the network information in the virtual machine configuration.

NOTE

The KubeMacPool component provides a MAC address pool service for virtual machine NICs in designated namespaces. It is not enabled by default. Enable a MAC address pool in a namespace by applying the KubeMacPool label to that namespace.

7.17.2.1. OpenShift Virtualization networking glossary

OpenShift Virtualization provides advanced networking functionality by using custom resources and plug-ins.

The following terms are used throughout OpenShift Virtualization documentation:

**Container Network Interface (CNI)**

- a Cloud Native Computing Foundation project, focused on container network connectivity. OpenShift 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.

**Network attachment definition**

- 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.

7.17.2.2. Creating a network attachment definition

7.17.2.2.1. Prerequisites

- A Linux bridge must be configured and attached on every node. See the node networking section for more information.

![WARNING]

Configuring ipam in a network attachment definition for virtual machines is not supported.

7.17.2.2.2. Creating a Linux bridge network attachment definition in the web console

The network attachment definition is a custom resource that exposes layer-2 devices to a specific namespace in your OpenShift Virtualization cluster.

Network administrators can create network attachment definitions to provide existing layer-2 networking to pods and virtual machines.

Procedure

1. In the web console, click Networking → Network Attachment Definitions.
2. Click Create Network Attachment Definition.
3. Enter a unique Name and optional Description.
4. Click the Network Type list and select CNV Linux bridge.
5. Enter the name of the bridge in the Bridge Name field.
6. Optional: If the resource has VLAN IDs configured, enter the ID numbers in the VLAN Tag Number field.
7. Click Create.

7.17.2.2.3. Creating a Linux bridge network attachment definition in the CLI

As a network administrator, you can configure a network attachment definition of type cnv-bridge to provide Layer-2 networking to pods and virtual machines.

![NOTE]

The network attachment definition must be in the same namespace as the pod or virtual machine.
Procedure

1. Create a network attachment definition manifest. The manifest must have the following contents, modified to match your configuration:

```yaml
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: <a-bridge-network>  
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/<bridge-interface>
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "<a-bridge-network>",
    "plugins": [
      {"type": "cnv-bridge",
       "bridge": "<bridge-interface>",
       "vlan": 1
      },
      {"type": "cnv-tuning"
      }
    ]
  }'

1 The name for the NetworkAttachmentDefinition object.
2 Optional: Annotation key-value pair for node selection, where bridge-interface is the name of a bridge configured on some nodes. If you add this annotation to your network attachment definition, your virtual machine instances will only run on the nodes that have the bridge-interface bridge connected.
3 The name for the configuration. It is recommended to match the configuration name to the name value of the network attachment definition.
4 The actual name of the Container Network Interface (CNI) plug-in that provides the network for this network attachment definition. Do not change this field unless you want to use a different CNI.
5 The name of the Linux bridge configured on the node.
6 Optional: The VLAN tag.
7 The CNI plug-in that allows the MAC pool manager to assign a unique MAC address to the connection.

2. Create the network attachment definition:

```
$ oc create -f <network-attachment-definition.yaml>  
```

1 Where <network-attachment-definition.yaml> is the file name of the network attachment definition manifest.
Verification

- Verify that the network attachment definition is created by running either `oc get network-attachment-definition <name>` or `oc get net-attach-def <name>`. For example:

  ```
  $ oc get network-attachment-definition <a-bridge-network>  
  ```

  Where `<a-bridge-network>` is the name specified in the network attachment definition config.

7.17.2.3. Attaching the virtual machine to the additional network

7.17.2.3.1. Creating a NIC for a virtual machine in the web console

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

Procedure

1. In the correct project in the OpenShift Virtualization console, click Workloads → Virtualization from the side menu.

2. Click the Virtual Machines tab.

3. Select a virtual machine to open the Virtual Machine Overview screen.

4. Click Network Interfaces to display the NICs already attached to the virtual machine.

5. Click Add Network Interface to create a new slot in the list.

6. Use the Network drop-down list to select the network attachment definition for the additional network.

7. Fill in the Name, Model, Type, and MAC Address for the new NIC.

8. Click Add to save and attach the NIC to the virtual machine.

7.17.2.3.2. Networking fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are e1000e and virtio.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>List of available binding methods. For the default pod network, <strong>masquerade</strong> is the only recommended binding method. For secondary networks, use the <strong>bridge</strong> binding method. The <strong>masquerade</strong> method is not supported for non-default networks.</td>
</tr>
<tr>
<td>MAC Address</td>
<td>MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.</td>
</tr>
</tbody>
</table>

### 7.17.2.3.3. Attaching a virtual machine to a secondary network in the CLI

Attach a virtual machine to a secondary network by adding a bridge interface and specifying a network attachment definition in the virtual machine configuration.

This procedure uses a YAML file to demonstrate editing the configuration and applying the updated file to the cluster. You can alternatively use the **oc edit <object> <name>** command to edit an existing virtual machine.

#### Prerequisites

- Shut down the virtual machine before editing the configuration. If you edit a running virtual machine, you must restart the virtual machine for the changes to take effect.

#### Procedure

1. Create or edit a virtual machine configuration that you want to connect to the bridge network. Add the bridge interface to the **spec.template.spec.domain.devices.interfaces** list and the network attachment definition to the **spec.template.spec.networks** list. The name of the **interfaces** entry must be the same as the **networks** entry. The following example adds a bridge interface called **bridge-net** that connects to the **a-bridge-network** network attachment definition:

```yaml
apiVersion: v1
kind: VirtualMachine
metadata:
  name: example-vm
spec:
template:
spec:
domain:
devices:
  interfaces:
    - masquerade: {}
      name: default
    - bridge: {}
      name: bridge-net
... networks:
  - name: default
```
The name of the bridge interface. This value must match the name value of the corresponding spec.template.spec.networks entry.

3. The name of the network attachment definition.

**NOTE**

The example virtual machine is connected to both the default pod network and bridge-net, which is defined by a network attachment definition named a-bridge-network.

2. Apply the configuration:

   $ oc apply -f <example-vm.yaml>

3. Optional: If you edited a running virtual machine, you must restart it for the changes to take effect.

### 7.17.2.4. Additional resources

- Configuring IP addresses for virtual machines
- Configuring PXE booting for virtual machines
- Attaching a virtual machine to an SR-IOV network

### 7.17.3. Configuring IP addresses for virtual machines

You can configure either dynamically or statically provisioned IP addresses for virtual machines.

**Prerequisites**

- The virtual machine can connect to a secondary network
- To configure a dynamic IP for the virtual machine, you must have a DHCP server available on the secondary network.

#### 7.17.3.1. Configuring an IP address for a new virtual machine using cloud-init

You can use cloud-init to configure an IP address when you create a virtual machine. The IP address can be dynamically or statically provisioned.

**Procedure**

- Create a virtual machine configuration and include the cloud-init network details in the spec.volumes.cloudInitNoCloud.networkData field of the virtual machine configuration:
a. To configure a dynamic IP, specify the interface name and the `dhcp4` boolean:

```yaml
kind: VirtualMachine
spec:
  ...  
  volumes:
  - cloudInitNoCloud:
      networkData:
        version: 2
        ethernets:
          eth1: 1
          dhcp4: true 2
```

1. The interface name.
2. Uses DHCP to provision an IPv4 address.

b. To configure a static IP, specify the interface name and the IP address:

```yaml
kind: VirtualMachine
spec:
  ...  
  volumes:
  - cloudInitNoCloud:
      networkData:
        version: 2
        ethernets:
          eth1: 1
          addresses:
            - 10.10.10.14/24 2
```

1. The interface name.
2. The static IP address for the virtual machine.

### 7.17.4. Configuring an SR-IOV network device for virtual machines

You can configure a Single Root I/O Virtualization (SR-IOV) device for virtual machines in your cluster. This process is similar but not identical to configuring an SR-IOV device for OpenShift Container Platform.

**NOTE**

Live migration is not supported for virtual machines that are attached to an SR-IOV network interface.

### 7.17.4.1. Prerequisites

- You must have installed the SR-IOV Operator.
- You must have configured the SR-IOV Operator.
7.17.4.2. Automated discovery of SR-IOV network devices

The SR-IOV Network Operator searches your cluster for SR-IOV capable network devices on worker nodes. The Operator creates and updates a SriovNetworkNodeState custom resource (CR) for each worker node that provides a compatible SR-IOV network device.

The CR is assigned the same name as the worker node. The `status.interfaces` list provides information about the network devices on a node.

IMPORTANT

Do not modify a `SriovNetworkNodeState` object. The Operator creates and manages these resources automatically.

7.17.4.2.1. Example SriovNetworkNodeState object

The following YAML is an example of a `SriovNetworkNodeState` object created by the SR-IOV Network Operator:

An `SriovNetworkNodeState` object

```yaml
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodeState
metadata:
  name: node-25
  namespace: openshift-sriov-network-operator
  ownerReferences:
    - apiVersion: sriovnetwork.openshift.io/v1
      blockOwnerDeletion: true
      controller: true
      kind: SriovNetworkNodePolicy
      name: default
spec:
  dpConfigVersion: "39824"
status:
  interfaces:
    - deviceID: "1017"
      driver: mlx5_core
      mtu: 1500
      name: ens785f0
      pciAddress: "0000:18:00.0"
      totalvfs: 8
      vendor: 15b3
    - deviceID: "1017"
      driver: mlx5_core
      mtu: 1500
      name: ens785f1
      pciAddress: "0000:18:00.1"
      totalvfs: 8
      vendor: 15b3
    - deviceID: 158
      driver: i40e
      mtu: 1500
      name: ens817f0
      pciAddress: 0000:81:00.0
```
The value of the name field is the same as the name of the worker node.

The interfaces stanza includes a list of all of the SR-IOV devices discovered by the Operator on the worker node.

7.17.4.3. Configuring SR-IOV network devices


NOTE

When applying the configuration specified in a SriovNetworkNodePolicy object, the SR-IOV Operator might drain the nodes, and in some cases, reboot nodes.

It might take several minutes for a configuration change to apply.

Prerequisites

- You installed the OpenShift CLI (oc).
- You have access to the cluster as a user with the cluster-admin role.
- You have installed the SR-IOV Network Operator.
- You have enough available nodes in your cluster to handle the evicted workload from drained nodes.
- You have not selected any control plane nodes for SR-IOV network device configuration.

Procedure

1. Create an SriovNetworkNodePolicy object, and then save the YAML in the <name>-sriov-node-network.yaml file. Replace <name> with the name for this configuration.
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: <name>  
  namespace: openshift-sriov-network-operator  
spec:
  resourceName: <sriov_resource_name>  
nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true"  
priority: <priority>  
mtu: <mtu>  
umVfs: <num>  
nicSelector: 
  vendor: "<vendor_code>"  
  deviceID: "<device_id>"  
  pfNames: ["<pf_name>", ...]  
  rootDevices: ["<pci_bus_id>", "..."]  
deviceType: vfio-pci  
isRdma: false

1. Specify a name for the CR object.
2. Specify the namespace where the SR-IOV Operator is installed.
3. Specify the resource name of the SR-IOV device plug-in. You can create multiple SriovNetworkNodePolicy objects for a resource name.
4. Specify the node selector to select which nodes are configured. Only SR-IOV network devices on selected nodes are configured. The SR-IOV Container Network Interface (CNI) plug-in and device plug-in are deployed only on selected nodes.
5. Optional: Specify an integer value between 0 and 99. A smaller number gets higher priority, so a priority of 10 is higher than a priority of 99. The default value is 99.
6. Optional: Specify a value for the maximum transmission unit (MTU) of the virtual function. The maximum MTU value can vary for different NIC models.
7. Specify the number of the virtual functions (VF) to create for the SR-IOV physical network device. For an Intel network interface controller (NIC), the number of VFs cannot be larger than the total VFs supported by the device. For a Mellanox NIC, the number of VFs cannot be larger than 128.
8. The nicSelector mapping selects the Ethernet device for the Operator to configure. You do not need to specify values for all the parameters. It is recommended to identify the Ethernet adapter with enough precision to minimize the possibility of selecting an Ethernet device unintentionally. If you specify rootDevices, you must also specify a value for vendor, deviceID, or pfNames. If you specify both pfNames and rootDevices at the same time, ensure that they point to an identical device.
9. Optional: Specify the vendor hex code of the SR-IOV network device. The only allowed values are either 8086 or 15b3.
10. Optional: Specify the device hex code of SR-IOV network device. The only allowed values are 158b, 1015, 1017.
Optional: The parameter accepts an array of one or more physical function (PF) names for the Ethernet device.

The parameter accepts an array of one or more PCI bus addresses for the physical function of the Ethernet device. Provide the address in the following format: `0000:02:00.1`.

The `vfio-pci` driver type is required for virtual functions in OpenShift Virtualization.

Optional: Specify whether to enable remote direct memory access (RDMA) mode. For a Mellanox card, set `isRdma` to `false`. The default value is `false`.

NOTE

If `isRDMA` flag is set to `true`, you can continue to use the RDMA enabled VF as a normal network device. A device can be used in either mode.

1. Optional: Label the SR-IOV capable cluster nodes with `SriovNetworkNodePolicy.Spec.NodeSelector` if they are not already labeled. For more information about labeling nodes, see "Understanding how to update labels on nodes".

2. Create the `SriovNetworkNodePolicy` object:

   ```bash
   $ oc create -f <name>-sriov-node-network.yaml
   
   where `<name>` specifies the name for this configuration.
   
   After applying the configuration update, all the pods in `sriov-network-operator` namespace transition to the `Running` status.

3. To verify that the SR-IOV network device is configured, enter the following command. Replace `<node_name>` with the name of a node with the SR-IOV network device that you just configured.

   ```bash
   $ oc get sriovnetworknodestates -n openshift-sriov-network-operator <node_name> -o jsonpath='{.status.syncStatus}'
   
   7.17.4.4. Next steps
   
   - Configuring an SR-IOV network attachment for virtual machines

7.17.5. Defining an SR-IOV network

You can create a network attachment for a Single Root I/O Virtualization (SR-IOV) device for virtual machines.

After the network is defined, you can attach virtual machines to the SR-IOV network.

7.17.5.1. Prerequisites

- You must have configured an SR-IOV device for virtual machines.

7.17.5.2. Configuring SR-IOV additional network
You can configure an additional network that uses SR-IOV hardware by creating a **SriovNetwork** object. When you create a **SriovNetwork** object, the SR-IOV Operator automatically creates a **NetworkAttachmentDefinition** object.

Users can then attach virtual machines to the SR-IOV network by specifying the network in the virtual machine configurations.

**NOTE**

Do not modify or delete a **SriovNetwork** object if it is attached to any pods or virtual machines in the **running** state.

**Prerequisites**

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

**Procedure**

1. Create the following **SriovNetwork** object, and then save the YAML in the `<name>-sriov-network.yaml` file. Replace `<name>` with a name for this additional network.

   ```yaml
   apiVersion: sriovnetwork.openshift.io/v1
   kind: SriovNetwork
   metadata:
     name: <name>  # 1
     namespace: openshift-sriov-network-operator  # 2
   spec:
     resourceName: <sriov_resource_name>  # 3
     networkNamespace: <target_namespace>  # 4
     vlan: <vlan>  # 5
     spoofChk: "<spoof_check>"  # 6
     linkState: <link_state>  # 7
     maxTxRate: <max_tx_rate>  # 8
     minTxRate: <min_rx_rate>  # 9
     vlanQoS: <vlan_qos>  # 10
     trust: "<trust_vf>"  # 11
     capabilities: <capabilities>  # 12
   ```

   **1** Replace `<name>` with a name for the object. The SR-IOV Network Operator creates a **NetworkAttachmentDefinition** object with same name.

   **2** Specify the namespace where the SR-IOV Network Operator is installed.

   **3** Replace `<sriov_resource_name>` with the value for the `.spec.resourceName` parameter from the **SriovNetworkNodePolicy** object that defines the SR-IOV hardware for this additional network.

   **4** Replace `<target_namespace>` with the target namespace for the SriovNetwork. Only pods or virtual machines in the target namespace can attach to the SriovNetwork.

   **5** Optional: Replace `<vlan>` with a Virtual LAN (VLAN) ID for the additional network. The integer value must be from 0 to 4095. The default value is 0.
Optional: Replace `<spoof_check>` with the spoof check mode of the VF. The allowed values are the strings "on" and "off".

**IMPORTANT**

You must enclose the value you specify in quotes or the CR is rejected by the SR-IOV Network Operator.

Optional: Replace `<link_state>` with the link state of virtual function (VF). Allowed value are `enable`, `disable` and `auto`.

Optional: Replace `<max_tx_rate>` with a maximum transmission rate, in Mbps, for the VF.

Optional: Replace `<min_tx_rate>` with a minimum transmission rate, in Mbps, for the VF. This value should always be less than or equal to Maximum transmission rate.

**NOTE**

Intel NICs do not support the `minTxRate` parameter. For more information, see BZ#1772847.

Optional: Replace `<vlan_qos>` with an IEEE 802.1p priority level for the VF. The default value is `0`.

Optional: Replace `<trust_vf>` with the trust mode of the VF. The allowed values are the strings "on" and "off".

**IMPORTANT**

You must enclose the value you specify in quotes or the CR is rejected by the SR-IOV Network Operator.

Optional: Replace `<capabilities>` with the capabilities to configure for this network.

2. To create the object, enter the following command. Replace `<name>` with a name for this additional network.

   ```bash
   $ oc create -f <name>-sriov-network.yaml
   ```

3. Optional: To confirm that the `NetworkAttachmentDefinition` object associated with the `SriovNetwork` object that you created in the previous step exists, enter the following command. Replace `<namespace>` with the namespace you specified in the `SriovNetwork` object.

   ```bash
   $ oc get net-attach-def -n <namespace>
   ```

### 7.17.5.3. Next steps

- Attaching a virtual machine to an SR-IOV network.

### 7.17.6. Attaching a virtual machine to an SR-IOV network

You can attach a virtual machine to use a Single Root I/O Virtualization (SR-IOV) network as a secondary network.
7.17.6.1. Prerequisites

- You must have configured an SR-IOV device for virtual machines.
- You must have defined an SR-IOV network.

7.17.6.2. Attaching a virtual machine to an SR-IOV network

You can attach the virtual machine to the SR-IOV network by including the network details in the virtual machine configuration.

Procedure

1. Include the SR-IOV network details in the `spec.domain.devices.interfaces` and `spec.networks` of the virtual machine configuration:

   ```yaml
   kind: VirtualMachine
   ...
   spec:
     domain:
       devices:
         interfaces:
           - name: <default>  
             masquerade: {}
           - name: <nic1>  
             sriov: {}
         networks:
           - name: <default>  
             pod: {}
           - name: <nic1>  
             multus:
               networkName: <sriov-network>
   ...
   ```

   - **1**: A unique name for the interface that is connected to the pod network.
   - **2**: The `masquerade` binding to the default pod network.
   - **3**: A unique name for the SR-IOV interface.
   - **4**: The name of the pod network interface. This must be the same as the `interfaces.name` that you defined earlier.
   - **5**: The name of the SR-IOV interface. This must be the same as the `interfaces.name` that you defined earlier.
   - **6**: The name of the SR-IOV network attachment definition.

2. Apply the virtual machine configuration:

   ```bash
   $ oc apply -f <vm-sriov.yaml>  
   ```

   - **1**: The name of the virtual machine YAML file.
7.17.7. Viewing the IP address of NICs on a virtual machine

You can view the IP address for a network interface controller (NIC) by using the web console or the oc client. The QEMU guest agent displays additional information about the virtual machine’s secondary networks.

7.17.7.1. 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
  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
  ```

7.17.7.2. 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 OpenShift Virtualization console, click Workloads → Virtualization from the side menu.
2. Click the **Virtual Machines** tab.

3. Select a virtual machine name to open the **Virtual Machine Overview** screen.

The information for each attached NIC is displayed under **IP Address**.

### 7.17.8. Using a MAC address pool for virtual machines

The **KubeMacPool** component provides a MAC address pool service for virtual machine NICs in designated namespaces. Enable a MAC address pool in a namespace by applying the KubeMacPool label to that namespace.

#### 7.17.8.1. About KubeMacPool

If you enable the KubeMacPool component for a namespace, virtual machine NICs in that namespace are allocated MAC addresses from a MAC address pool. This ensures that the NIC is assigned a unique MAC address that does not conflict with the MAC address of another virtual machine.

Virtual machine instances created from that virtual machine retain the assigned MAC address across reboots.

**NOTE**

KubeMacPool does not handle virtual machine instances created independently from a virtual machine.

KubeMacPool is disabled by default. Enable a MAC address pool for a namespace by applying the KubeMacPool label to the namespace.

#### 7.17.8.2. Enabling a MAC address pool for a namespace in the CLI

Enable a MAC address pool for virtual machines in a namespace by applying the `mutatevirtualmachines.kubemacpool.io=allocate` label to the namespace.

**Procedure**

- Add the KubeMacPool label to the namespace. The following example adds the KubeMacPool label to two namespaces, `<namespace1>` and `<namespace2>`:

  ```bash
  $ oc label namespace <namespace1> <namespace2>
  mutatevirtualmachines.kubemacpool.io=allocate
  ```

#### 7.17.8.3. Disabling a MAC address pool for a namespace in the CLI

Disable a MAC address pool for virtual machines in a namespace by removing the `mutatevirtualmachines.kubemacpool.io` label.

**Procedure**

- Remove the KubeMacPool label from the namespace. The following example removes the KubeMacPool label from two namespaces, `<namespace1>` and `<namespace2>`:
7.18. VIRTUAL MACHINE DISKS

7.18.1. Storage features

Use the following table to determine feature availability for local and shared persistent storage in OpenShift Virtualization.

7.18.1.1. OpenShift Virtualization storage feature matrix

Table 7.8. OpenShift Virtualization storage feature matrix

<table>
<thead>
<tr>
<th>Feature</th>
<th>Virtual machine live migration</th>
<th>Host-assisted virtual machine disk cloning</th>
<th>Storage-assisted virtual machine disk cloning</th>
<th>Virtual machine snapshots</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenShift Container Storage: RBD block-mode volumes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenShift Virtualization hostpath provisioner</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Other single-node writable storage</td>
<td>No</td>
<td>Yes</td>
<td>Yes [2]</td>
<td>Yes [2]</td>
</tr>
</tbody>
</table>

1. PVCs must request a ReadWriteMany access mode.

2. Storage provider must support both Kubernetes and CSI snapshot APIs

NOTE

You cannot live migrate virtual machines that use:

- A storage class with ReadWriteOnce (RWO) access mode
- Passthrough features such as SR-IOV and GPU

Do not set the `evictionStrategy` field to `LiveMigrate` for these virtual machines.

7.18.2. Configuring local storage for virtual machines

You can configure local storage for your virtual machines by using the hostpath provisioner feature.

7.18.2.1. About the hostpath provisioner

$ oc label namespace <namespace1> <namespace2> mutatevirtualmachines.kubemacpool.io-
The hostpath provisioner is a local storage provisioner designed for OpenShift Virtualization. If you want to configure local storage for virtual machines, you must enable the hostpath provisioner first.

When you install the OpenShift Virtualization Operator, the hostpath provisioner Operator is automatically installed. To use it, you must:

- Configure SELinux:
  - If you use Red Hat Enterprise Linux CoreOS (RHCOS) 8 workers, you must create a MachineConfig object on each node.
  - Otherwise, apply the SELinux label container_file_t to the persistent volume (PV) backing directory on each node.
- Create a HostPathProvisioner custom resource.
- Create a StorageClass object for the hostpath provisioner.

The hostpath provisioner Operator deploys the provisioner as a DaemonSet on each node when you create its custom resource. In the custom resource file, you specify the backing directory for the persistent volumes that the hostpath provisioner creates.

### 7.18.2.2. Configuring SELinux for the hostpath provisioner on Red Hat Enterprise Linux CoreOS (RHCOS) 8

You must configure SELinux before you create the HostPathProvisioner custom resource. To configure SELinux on Red Hat Enterprise Linux CoreOS (RHCOS) 8 workers, you must create a MachineConfig object on each node.

#### Prerequisites

- Create a backing directory on each node for the persistent volumes (PVs) that the hostpath provisioner creates.

  **IMPORTANT**

  The backing directory must not be located in the filesystem’s root directory because the / partition is read-only on RHCOS. For example, you can use /var/<directory_name> but not /<directory_name>.

#### Procedure

1. Create the MachineConfig file. For example:

   ```
   $ touch machineconfig.yaml
   ```

2. Edit the file, ensuring that you include the directory where you want the hostpath provisioner to create PVs. For example:

   ```yaml
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     name: 50-set-selinux-for-hostpath-provisioner
   labels:
     machineconfiguration.openshift.io/role: worker
   ```
1 Specify the backing directory where you want the provisioner to create PVs. This directory must not be located in the filesystem’s root directory (/).

3. Create the **MachineConfig** object:

```bash
$ oc create -f machineconfig.yaml -n <namespace>
```

### 7.18.2.3. Using the hostpath provisioner to enable local storage

To deploy the hostpath provisioner and enable your virtual machines to use local storage, first create a **HostPathProvisioner** custom resource.

#### Prerequisites

- Create a backing directory on each node for the persistent volumes (PVs) that the hostpath provisioner creates.

**IMPORTANT**

The backing directory must not be located in the filesystem’s root directory because the / partition is read-only on Red Hat Enterprise Linux CoreOS (RHCOS). For example, you can use `/var/<directory_name>` but not `/<directory_name>`.

- Apply the SELinux context **container_file_t** to the PV backing directory on each node. For example:

  ```bash
  $ sudo chcon -t container_file_t -R <backing_directory_path>
  ```

**NOTE**

If you use Red Hat Enterprise Linux CoreOS (RHCOS) 8 workers, you must configure SELinux by using a **MachineConfig** manifest instead.
Procedure

1. Create the `HostPathProvisioner` custom resource file. For example:

   ```bash
   $ touch hostpathprovisioner_cr.yaml
   ```

2. Edit the file, ensuring that the `spec.pathConfig.path` value is the directory where you want the hostpath provisioner to create PVs. For example:

   ```yaml
   apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
   kind: HostPathProvisioner
   metadata:
     name: hostpath-provisioner
   spec:
     imagePullPolicy: IfNotPresent
     pathConfig:
       path: "<backing_directory_path>"  # Specify the backing directory where you want the provisioner to create PVs. This directory must not be located in the filesystem's root directory (/).
       useNamingPrefix: false  # Change this value to `true` if you want to use the name of the persistent volume claim (PVC) that is bound to the created PV as the prefix of the directory name.
   ```

3. Create the custom resource in the `openshift-cnv` namespace:

   ```bash
   $ oc create -f hostpathprovisioner_cr.yaml -n openshift-cnv
   ```

7.18.2.4. Creating a storage class

When you create a storage class, you set parameters that affect the dynamic provisioning of persistent volumes (PVs) that belong to that storage class.

**IMPORTANT**

When using OpenShift Virtualization with OpenShift Container Platform Container Storage, specify RBD block mode persistent volume claims (PVCs) when creating virtual machine disks. With virtual machine disks, RBD block mode volumes are more efficient and provide better performance than Ceph FS or RBD filesystem-mode PVCs.

To specify RBD block mode PVCs, use the `ocs-storagecluster-ceph-rbd` storage class and **VolumeMode: Block**.
NOTE

You cannot update a **StorageClass** object’s parameters after you create it.

**Procedure**

1. Create a YAML file for defining the storage class. For example:

   ```bash
   $ touch storageclass.yaml
   ```

2. Edit the file. For example:

   ```yaml
   apiVersion: storage.k8s.io/v1
   kind: StorageClass
   metadata:
     name: hostpath-provisioner
   provisioner: kubevirt.io/hostpath-provisioner
   reclaimPolicy: Delete
   volumeBindingMode: WaitForFirstConsumer
   ```

   You can optionally rename the storage class by changing this value.

   The two possible **reclaimPolicy** values are **Delete** and **Retain**. If you do not specify a value, the storage class defaults to **Delete**.

   The **volumeBindingMode** value determines when dynamic provisioning and volume binding occur. Specify **WaitForFirstConsumer** to delay the binding and provisioning of a PV until after a pod that uses the persistent volume claim (PVC) is created. This ensures that the PV meets the pod’s scheduling requirements.

3. Create the **StorageClass** object:

   ```bash
   $ oc create -f storageclass.yaml
   ```

**NOTE**

Virtual machines use data volumes that are based on local PVs. Local PVs are bound to specific nodes. While the disk image is prepared for consumption by the virtual machine, it is possible that the virtual machine cannot be scheduled to the node where the local storage PV was previously pinned.

To solve this problem, use the Kubernetes pod scheduler to bind the PVC to a PV on the correct node. By using **StorageClass** with **volumeBindingMode** set to **WaitForFirstConsumer**, the binding and provisioning of the PV is delayed until a **Pod** is created using the PVC.

**Additional resources**

- **Storage classes**

7.18.3. Configuring CDI to work with namespaces that have a compute resource quota
You can use the Containerized Data Importer (CDI) to import, upload, and clone virtual machine disks into namespaces that are subject to CPU and memory resource restrictions.

### 7.18.3.1. About CPU and memory quotas in a namespace

A resource quota, defined by the `ResourceQuota` object, imposes restrictions on a namespace that limit the total amount of compute resources that can be consumed by resources within that namespace.

The `CDIConfig` object defines the user configuration for the Containerized Data Importer (CDI). The CPU and memory request and limit values for the `CDIConfig` object are set to a default value of 0. This ensures that pods created by CDI that make no compute resource requirements are given the default values and are allowed to run in a namespace that is restricted with a quota.

### 7.18.3.2. Editing the `CDIConfig` object to override CPU and memory defaults

Modify the default settings for CPU and memory requests and limits for your use case by editing the `spec` attribute of the `CDIConfig` object.

**Prerequisites**
- Install the OpenShift CLI (`oc`).

**Procedure**

1. Edit the `cdiconfig/config` by running the following command:

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

2. Change the default CPU and memory requests and limits by editing the `spec`:
   - `podResourceRequirements` property of the `CDIConfig` object:

   ```yaml
   apiVersion: cdi.kubevirt.io/v1beta1
   kind: CDIConfig
   metadata:
     labels:
       app: containerized-data-importer
       cdi.kubevirt.io: ""
   name: config
   spec:
     podResourceRequirements:
     limits:
       cpu: "4"
       memory: "1Gi"
     requests:
       cpu: "1"
       memory: "250Mi"
   ...
   ```

3. Save and exit the editor to update the `CDIConfig` object.

**Verification**
- View the `CDIConfig` status and verify your changes by running the following command:
7.18.3.3. Additional resources

- Resource quotas per project

7.18.4. Uploading local disk images by using the web console

You can upload a locally stored disk image file by using the web console.

7.18.4.1. Prerequisites

- You must have a virtual machine image file in IMG, ISO, or QCOW2 format.
- If you require scratch space according to the CDI supported operations matrix, you must first define a storage class or prepare CDI scratch space for this operation to complete successfully.

7.18.4.2. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2 ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2** ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2 ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2* □ GZ □ XZ</td>
<td>✓ QCOW2* ✓ GZ* ✓ XZ*</td>
</tr>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW ✓ GZ ✓ XZ</td>
<td>✓ RAW ✓ GZ ✓ XZ</td>
<td>✓ RAW ✓ GZ ✓ XZ</td>
<td>✓ RAW* □ GZ □ XZ</td>
<td>✓ RAW* ✓ GZ* ✓ XZ*</td>
</tr>
</tbody>
</table>

✓ Supported operation

□ Unsupported operation

* Requires scratch space

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

7.18.4.3. Uploading an image file using the web console

Use the web console to upload an image file to a new persistent volume claim (PVC). You can later use this PVC to attach the image to new virtual machines.

Prerequisites

- You must have one of the following:
  - A raw virtual machine image file in either ISO or IMG format.

$ oc get cdiconfig config -o yaml
A virtual machine image file in QCOW2 format.

- For best results, compress your image file according to the following guidelines before you upload it:
  - Compress a raw image file by using `xz` or `gzip`.

**NOTE**

Using a compressed raw image file results in the most efficient upload.

- Compress a QCOW2 image file by using the method that is recommended for your client:
  - If you use a Linux client, **sparsify** the QCOW2 file by using the `virt-sparsify` tool.
  - If you use a Windows client, compress the QCOW2 file by using `xz` or `gzip`.

**Procedure**

1. From the side menu of the web console, click **Storage → Persistent Volume Claims**
2. Click the **Create Persistent Volume Claim** drop-down list to expand it.
3. Click **With Data Upload Form** to open the **Upload Data to Persistent Volume Claim** page.
4. Click **Browse** to open the file manager and select the image that you want to upload, or drag the file into the **Drag a file here or browse to upload** field.
5. Optional: Set this image as the default image for a specific operating system.
   a. Select the **Attach this data to a virtual machine operating system** check box.
   b. Select an operating system from the list.
6. The **Persistent Volume Claim Name** field is automatically filled with a unique name and cannot be edited. Take note of the name assigned to the PVC so that you can identify it later, if necessary.
7. Select a storage class from the **Storage Class** list.
8. In the **Size** field, enter the size value for the PVC. Select the corresponding unit of measurement from the drop-down list.

**WARNING**

The PVC size must be larger than the size of the uncompressed virtual disk.

9. Select an **Access Mode** that matches the storage class that you selected.
10. Click **Upload**.
7.18.5. Uploading local disk images by using the virtctl tool

You can upload a locally stored disk image to a new or existing data volume by using the virtctl command-line utility.

7.18.5.1. Prerequisites

- Install the kubevirt-virtctl package.
- If you require scratch space according to the CDI supported operations matrix, you must first define a storage class or prepare CDI scratch space for this operation to complete successfully.

7.18.5.2. About data volumes

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

7.18.5.3. Creating an upload data volume

You can manually create a data volume with an upload data source to use for uploading local disk images.

Procedure

1. Create a data volume configuration that specifies spec: source: upload:

   ```yaml
   apiVersion: cdi.kubevirt.io/v1beta1
   kind: DataVolume
   metadata:
     name: <upload-datavolume>  
   spec:
     source:
       upload: {}
     pvc:
       accessModes:
         -ReadWriteOnce
       resources:
         requests:
           storage: <2Gi>
   ```

   1. The name of the data volume.
   2. The size of the data volume. Ensure that this value is greater than or equal to the size of the disk that you upload.

2. Create the data volume by running the following command:

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

7.18.5.4. Uploading a local disk image to a data volume
You can use the `virtctl` CLI utility to upload a local disk image from a client machine to a data volume (DV) in your cluster. You can use a DV that already exists in your cluster or create a new DV during this procedure.

**NOTE**

After you upload a local disk image, you can add it to a virtual machine.

**Prerequisites**

- You must have one of the following:
  - A raw virtual machine image file in either ISO or IMG format.
  - A virtual machine image file in QCOW2 format.
- For best results, compress your image file according to the following guidelines before you upload it:
  - Compress a raw image file by using `xz` or `gzip`.

**NOTE**

Using a compressed raw image file results in the most efficient upload.

- Compress a QCOW2 image file by using the method that is recommended for your client:
  - If you use a Linux client, `sparsify` the QCOW2 file by using the `virt-sparsify` tool.
  - If you use a Windows client, compress the QCOW2 file by using `xz` or `gzip`.
- 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:
   - The name of the upload data volume that you want to use. If this data volume does not exist, it is created automatically.
   - The size of the data volume, if you want it to be created during the upload procedure. The size must be greater than or equal to the size of the disk image.
   - The file location of the virtual machine disk image that you want to upload.

2. Upload the disk image by running the `virtctl image-upload` command. Specify the parameters that you identified in the previous step. For example:

   ```
   $ virtctl image-upload dv <datavolume_name> \
   --size=<datavolume_size> \
   --image-path=<path/to/image>
   ```
1. The name of the data volume.

2. The size of the data volume. For example: `--size=500Mi, --size=1G`

3. The file path of the virtual machine disk image.

**NOTE**

- If you do not want to create a new data volume, omit the `--size` parameter and include the `--no-create` flag.
- When uploading a disk image to a PVC, the PVC size must be larger than the size of the uncompressed virtual disk.
- 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. Optional. To verify that a data volume was created, view all data volumes by running the following command:

   ```bash
   $ oc get dvs
   ```

### 7.18.5.5. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QCOW2</td>
<td>QCOW2**</td>
<td>QCOW2</td>
<td>QCOW2*</td>
<td>QCOW2*</td>
</tr>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2**</td>
<td>✓ QCOW2</td>
<td>✓ QCOW2*</td>
<td>✓ QCOW2*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>✓ GZ*</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>✓ XZ*</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
<tr>
<td>KubeVirt(RAW)</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>□ GZ</td>
<td>□ GZ</td>
</tr>
<tr>
<td></td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>□ XZ</td>
<td>□ XZ</td>
</tr>
</tbody>
</table>

- ✓ Supported operation
- □ Unsupported operation
- * Requires scratch space
- ** Requires scratch space if a custom certificate authority is required

### 7.18.6. Uploading a local disk image to a block storage data volume

You can upload a local disk image into a block data volume by using the `virtctl` command-line utility.
In this workflow, you create a local block device to use as a persistent volume, associate this block
volume with an upload data volume, and use `virtctl` to upload the local disk image into the data volume.

### 7.18.6.1. Prerequisites

- Install the `kubevirt-virtctl` package.
- If you require scratch space according to the CDI supported operations matrix, you must first
define a storage class or prepare CDI scratch space for this operation to complete successfully.

### 7.18.6.2. About data volumes

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

### 7.18.6.3. About block persistent volumes

A block persistent volume (PV) is a PV that is backed by a raw block device. These volumes do not have
a file system and can provide performance benefits for virtual machines by reducing overhead.

Raw block volumes are provisioned by specifying `volumeMode: Block` in the PV and persistent volume
claim (PVC) specification.

### 7.18.6.4. Creating a local block persistent volume

Create a local block persistent volume (PV) on a node by populating a file and mounting it as a loop
device. You can then reference this loop device in a PV manifest as a `Block` volume and use it as a block
device for a virtual machine image.

**Procedure**

1. Log in as **root** to the node on which to create the local PV. This procedure uses `node01` for its examples.

2. Create a file and populate it with null characters so that it can be used as a block device. The
following example creates a file **loop10** with a size of 2Gb (20 100Mb blocks):

   ```
   $ dd if=/dev/zero of=<loop10> bs=100M count=20
   ```

3. Mount the **loop10** file as a loop device.

   ```
   $ losetup </dev/loop10>d3 <loop10> ① ②
   
   ① File path where the loop device is mounted.
   ② The file created in the previous step to be mounted as the loop device.
   ```

4. Create a **PersistentVolume** manifest that references the mounted loop device.

   ```
   kind: PersistentVolume
   apiVersion: v1
   ```
The path of the loop device on the node.

Specifies it is a block PV.

Optional: Set a storage class for the PV. If you omit it, the cluster default is used.

The node on which the block device was mounted.

5. Create the block PV.

```bash
# oc create -f <local-block-pv10.yaml>
```

The file name of the persistent volume created in the previous step.

### 7.18.6.5. Creating an upload data volume

You can manually create a data volume with an upload data source to use for uploading local disk images.

**Procedure**

1. Create a data volume configuration that specifies `spec: source: upload`:

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <upload-datavolume>
spec:
  source:
    upload: {}
pvc:
```
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: <2Gi>  

1. The name of the data volume.
2. The size of the data volume. Ensure that this value is greater than or equal to the size of the disk that you upload.

2. Create the data volume by running the following command:

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

### 7.18.6.6. Uploading a local disk image to a data volume

You can use the `virtctl` CLI utility to upload a local disk image from a client machine to a data volume (DV) in your cluster. You can use a DV that already exists in your cluster or create a new DV during this procedure.

**NOTE**

After you upload a local disk image, you can add it to a virtual machine.

#### Prerequisites

- You must have one of the following:
  - A raw virtual machine image file in either ISO or IMG format.
  - A virtual machine image file in QCOW2 format.
- For best results, compress your image file according to the following guidelines before you upload it:
  - Compress a raw image file by using `xz` or `gzip`.

**NOTE**

Using a compressed raw image file results in the most efficient upload.

- Compress a QCOW2 image file by using the method that is recommended for your client:
  - If you use a Linux client, `sparsify` the QCOW2 file by using the `virt-sparsify` tool.
  - If you use a Windows client, compress the QCOW2 file by using `xz` or `gzip`.
- 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:

- The name of the upload data volume that you want to use. If this data volume does not exist, it is created automatically.
- The size of the data volume, if you want it to be created during the upload procedure. The size must be greater than or equal to the size of the disk image.
- The file location of the virtual machine disk image that you want to upload.

2. Upload the disk image by running the `virtctl image-upload` command. Specify the parameters that you identified in the previous step. For example:

   ```
   $ virtctl image-upload dv <datavolume_name> \1
   --size=<datavolume_size> \2
   --image-path=</path/to/image> \3
   
   1 The name of the data volume.
   2 The size of the data volume. For example: --size=500Mi, --size=1G
   3 The file path of the virtual machine disk image.
   ```

   **NOTE**

   - If you do not want to create a new data volume, omit the `--size` parameter and include the `--no-create` flag.
   - When uploading a disk image to a PVC, the PVC size must be larger than the size of the uncompressed virtual disk.
   - 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. Optional. To verify that a data volume was created, view all data volumes by running the following command:

   ```
   $ oc get dvs
   ```

### 7.18.6.7. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✓ QCOW2 ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2** ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2 ✓ GZ* ✓ XZ*</td>
<td>✓ QCOW2* □ GZ □ XZ</td>
<td>✓ QCOW2* ✓ GZ* ✓ XZ*</td>
</tr>
</tbody>
</table>
## 7.18.7. Managing offline virtual machine snapshots

You can create, restore, and delete virtual machine (VM) snapshots for VMs that are powered off (offline). OpenShift Virtualization supports offline VM snapshots on:

- Red Hat OpenShift Container Storage
- Any other storage provider with the Container Storage Interface (CSI) driver that supports the Kubernetes Volume Snapshot API

### 7.18.7.1. About virtual machine snapshots

A snapshot represents the state and data of a virtual machine (VM) at a specific point in time. You can use a snapshot to restore an existing VM to a previous state (represented by the snapshot) for backup and disaster recovery or to rapidly roll back to a previous development version.

An offline VM snapshot is created from a VM that is powered off (Stopped state). The snapshot stores a copy of each Container Storage Interface (CSI) volume attached to the VM and a copy of the VM specification and metadata. Snapshots cannot be changed after creation.

With the offline VM snapshots feature, cluster administrators and application developers can:

- Create a new snapshot
- List all snapshots attached to a specific VM
- Restore a VM from a snapshot
- Delete an existing VM snapshot

### 7.18.7.1.1. Virtual machine snapshot controller and custom resource definitions (CRDs)

The VM snapshot feature introduces three new API objects defined as CRDs for managing snapshots:

- **VirtualMachineSnapshot**: Represents a user request to create a snapshot. It contains information about the current state of the VM.

### Content types

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt (RAW)</td>
<td>✓ RAW</td>
<td>✓ RAW</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
<td>✓ RAW*</td>
</tr>
<tr>
<td></td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>✓ GZ</td>
<td>□ GZ</td>
<td>□ GZ*</td>
</tr>
<tr>
<td></td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ</td>
<td>✓ XZ*</td>
</tr>
</tbody>
</table>

✓ Supported operation

☐ Unsupported operation

* Requires scratch space

** Requires scratch space if a custom certificate authority is required
• **VirtualMachineSnapshotContent**: Represents a provisioned resource on the cluster (a snapshot). It is created by the VM snapshot controller and contains references to all resources required to restore the VM.

• **VirtualMachineRestore**: Represents a user request to restore a VM from a snapshot.

The VM snapshot controller binds a **VirtualMachineSnapshotContent** object with the **VirtualMachineSnapshot** object for which it was created, with a one-to-one mapping.

### 7.18.7.2. Creating an offline virtual machine snapshot in the CLI

You can create a virtual machine (VM) snapshot for an offline VM by creating a **VirtualMachineSnapshot** object.

**Prerequisites**

- Ensure that the persistent volume claims (PVCs) are in a storage class that supports Container Storage Interface (CSI) volume snapshots.

- Install the OpenShift CLI (**oc**).

- Power down the VM for which you want to create a snapshot.

**Procedure**

1. Create a YAML file to define a **VirtualMachineSnapshot** object that specifies the name of the new **VirtualMachineSnapshot** and the name of the source VM. For example:

   ```yaml
   apiVersion: snapshot.kubevirt.io/v1alpha1
   kind: VirtualMachineSnapshot
   metadata:
     name: my-vmsnapshot
   spec:
     source:
       apiGroup: kubevirt.io
       kind: VirtualMachine
       name: my-vm
   ```

   1. The name of the new **VirtualMachineSnapshot** object.

   2. The name of the source VM.

2. Create the **VirtualMachineSnapshot** resource. The snapshot controller creates a **VirtualMachineSnapshotContent** object, binds it to the **VirtualMachineSnapshot** and updates the **status** and **readyToUse** fields of the **VirtualMachineSnapshot** object.

   ```bash
   $ oc create -f <my-vmsnapshot>.yaml
   ```

**Verification**

1. Verify that the **VirtualMachineSnapshot** object is created and bound with **VirtualMachineSnapshotContent**. The **readyToUse** flag must be set to **true**.
The **status** field of the **Progressing** condition specifies if the snapshot is still being created.

2. The **status** field of the **Ready** condition specifies if the snapshot creation process is complete.

3. Specifies if the snapshot is ready to be used.

4. Specifies that the snapshot is bound to a **VirtualMachineSnapshotContent** object created by the snapshot controller.

2. Check the **spec:volumeBackups** property of the **VirtualMachineSnapshotContent** resource to verify that the expected PVCs are included in the snapshot.
7.18.7.3. Restoring a virtual machine from a snapshot in the CLI

You can restore an existing virtual machine (VM) to a previous configuration by using a VM snapshot.

**Prerequisites**

- Install the OpenShift CLI (`oc`).
- Power down the VM you want to restore to a previous state.

**Procedure**

1. Create a YAML file to define a `VirtualMachineRestore` object that specifies the name of the VM you want to restore and the name of the snapshot to be used as the source.
   
   For example:
   ```yaml
   apiVersion: snapshot.kubevirt.io/v1alpha1
   kind: VirtualMachineRestore
   metadata:
     name: my-vmrestore
   spec:
     target:
       apiGroup: kubevirt.io
       kind: VirtualMachine
       name: my-vm
     virtualMachineSnapshotName: my-vmsnapshot
   ```
   
   1. The name of the new `VirtualMachineRestore` object.
   2. The name of the target VM you want to restore.
   3. The name of the `VirtualMachineSnapshot` object to be used as the source.

2. Create the `VirtualMachineRestore` resource. The snapshot controller updates the status fields of the `VirtualMachineRestore` object and replaces the existing VM configuration with the snapshot content.

   ```bash
   $ oc create -f <my-vmrestore>.yaml
   ```

**Verification**

- Verify that the VM is restored to the previous state represented by the snapshot. The `complete` flag must be set to `true`.

   ```bash
   $ oc get vmrestore <my-vmrestore>
   ```

**Example output**

```yaml
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineRestore
metadata:
  creationTimestamp: "2020-09-30T14:46:27Z"
generation: 5
```
name: my-vmrestore
namespace: default
ownerReferences:
- apiVersion: kubevirt.io/v1alpha3
  blockOwnerDeletion: true
controller: true
kind: VirtualMachine
name: my-vm
uid: 355897f3-73a0-4ec4-83d3-3c2df9486f4f
resourceVersion: "5512"
selfLink: /apis/snapshot.kubevirt.io/v1alpha1/namespaces/default/virtualmachinerestores/my-vmrestore
uid: 71c679a8-136e-46b0-b9b5-f57175a6a041
spec:
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: my-vm
virtualMachineSnapshotName: my-vmsnapshot
status:
  complete: true
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2020-09-30T14:46:28Z"
    reason: Operation complete
    status: "False"
    type: Progressing
  - lastProbeTime: null
    lastTransitionTime: "2020-09-30T14:46:28Z"
    reason: Operation complete
    status: "True"
    type: Ready
deletedDataVolumes:
- test-dv1
restoreTime: "2020-09-30T14:46:28Z"
restores:
- dataVolumeName: restore-71c679a8-136e-46b0-b9b5-f57175a6a041-datavolumedisk1
  persistentVolumeClaim: restore-71c679a8-136e-46b0-b9b5-f57175a6a041-datavolumedisk1
  volumeName: datavolumedisk1
  volumeSnapshotName: vmsnapshot-28eedf08-5d6a-42c1-969c-2eda58e2a78d-volume-datavolumedisk1

1. Specifies if the process of restoring the VM to the state represented by the snapshot is complete.

2. The status field of the Progressing condition specifies if the VM is still being restored.

3. The status field of the Ready condition specifies if the VM restoration process is complete.

7.18.7.4. Deleting a virtual machine snapshot in the CLI
You can delete an existing virtual machine (VM) snapshot by deleting the appropriate `VirtualMachineSnapshot` object.

**Prerequisites**

- Install the OpenShift CLI (`oc`).

**Procedure**

- Delete the `VirtualMachineSnapshot` object. The snapshot controller deletes the `VirtualMachineSnapshot` along with the associated `VirtualMachineSnapshotContent` object.

  ```
  $ oc delete vmsnapshot <my-vmsnapshot>
  ```

**Verification**

- Verify that the snapshot is deleted and no longer attached to this VM:

  ```
  $ oc get vmsnapshot
  ```

### 7.18.7.5. Additional resources

- [CSI Volume Snapshots](#)

### 7.18.8. Moving a local virtual machine disk to a different node

Virtual machines that use local volume storage can be moved so that they run on a specific node.

You might want to move the virtual machine to a specific node for the following reasons:

- The current node has limitations to the local storage configuration.
- The new node is better optimized for the workload of that virtual machine.

To move a virtual machine that uses local storage, you must clone the underlying volume by using a data volume. After the cloning operation is complete, you can edit the virtual machine configuration so that it uses the new data volume, or add the new data volume to another virtual machine.

**NOTE**

Users without the `cluster-admin` role require additional user permissions in order to clone volumes across namespaces.

### 7.18.8.1. Cloning a local volume to another node

You can move a virtual machine disk so that it runs on a specific node by cloning the underlying persistent volume claim (PVC).

To ensure the virtual machine disk is cloned to the correct node, you must either create a new persistent volume (PV) or identify one on the correct node. Apply a unique label to the PV so that it can be referenced by the data volume.
NOTE
The destination PV must be the same size or larger than the source PVC. If the destination PV is smaller than the source PVC, the cloning operation fails.

Prerequisites

- The virtual machine must not be running. Power down the virtual machine before cloning the virtual machine disk.

Procedure

1. Either create a new local PV on the node, or identify a local PV already on the node:

   - Create a local PV that includes the `nodeAffinity.nodeSelectorTerms` parameters. The following manifest creates a `10Gi` local PV on `node01`:

     ```yaml
     kind: PersistentVolume
     apiVersion: v1
     metadata:
       name: <destination-pv>  # 1
       annotations:
       spec:
         accessModes:
           - ReadWriteOnce
         capacity:
           storage: 10Gi  # 2
         local:
           path: /mnt/local-storage/local/disk1  # 3
         nodeAffinity:
           required:
             nodeSelectorTerms:
               - matchExpressions:
                 - key: kubernetes.io/hostname
                   operator: In
                   values:
                     - node01  # 4
           persistentVolumeReclaimPolicy: Delete
       storageClassName: local
       volumeMode: Filesystem
     
     1 The name of the PV.
     2 The size of the PV. You must allocate enough space, or the cloning operation fails. The size must be the same as or larger than the source PVC.
     3 The mount path on the node.
     4 The name of the node where you want to create the PV.

   - Identify a PV that already exists on the target node. You can identify the node where a PV is provisioned by viewing the `nodeAffinity` field in its configuration:

     ```bash
     $ oc get pv <destination-pv> -o yaml
     ```
The following snippet shows that the PV is on **node01**:

### Example output

```yaml
... 
spec: 
  nodeAffinity: 
    required: 
      nodeSelectorTerms: 
        - matchExpressions: 
          - key: kubernetes.io/hostname 
            operator: In 
            values: 
              - node01
... 
```

1. The `kubernetes.io/hostname` key uses the node hostname to select a node.
2. The hostname of the node.

2. Add a unique label to the PV:

   ```bash
   $ oc label pv <destination-pv> node=node01
   ```

3. Create a data volume manifest that references the following:
   - The PVC name and namespace of the virtual machine.
   - The label you applied to the PV in the previous step.
   - The size of the destination PV.

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <clone-datavolume>  
spec:
  source:
    pvc:
      name: "<source-vm-disk>"  
      namespace: "<source-namespace>"
    pvc:
      accessModes:
        - ReadWriteOnce
      selector:
        matchLabels:
          node: node01
      resources:
        requests:
          storage: <10Gi>
```

1. The name of the new data volume.
The name of the source PVC. If you do not know the PVC name, you can find it in the virtual machine configuration: `spec.volumes.persistentVolumeClaim.claimName`.

The namespace where the source PVC exists.

The label that you applied to the PV in the previous step.

The size of the destination PV.

4. Start the cloning operation by applying the data volume manifest to your cluster:

```bash
$ oc apply -f <clone-datavolume.yaml>
```

The data volume clones the PVC of the virtual machine into the PV on the specific node.

### 7.18.9. 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 OpenShift Virtualization.

#### 7.18.9.1. About data volumes

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

#### 7.18.9.2. Creating a blank disk image with data volumes

You can create a new blank disk image in a persistent volume claim by customizing and deploying a data volume configuration file.

**Prerequisites**

- At least one available persistent volume.
- Install the OpenShift CLI (`oc`).

**Procedure**

1. Edit the data volume configuration file:

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
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
```
Create the blank disk image by running the following command:

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

### 7.18.9.3. Template: Data volume configuration file for blank disk images

**blank-image-datavolume.yaml**

```yaml
apiVersion: cdi.kubevirt.io/v1beta1
crd: 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
  requests:
    storage: 500Mi
```

### 7.18.10. Cloning a data volume using smart-cloning

Smart-cloning is a built-in feature of OpenShift Container Platform Storage (OCS), designed to enhance performance of the cloning process. Clones created with smart-cloning are faster and more efficient than host-assisted cloning.

You do not need to perform any action to enable smart-cloning, but you need to ensure your storage environment is compatible with smart-cloning to use this feature.

When you create a data volume with a persistent volume claim (PVC) source, you automatically initiate the cloning process. You always receive a clone of the data volume, if your environment supports smart-cloning or not. However, you will only receive the performance benefits of smart cloning if you storage provider supports smart-cloning.

#### 7.18.10.1. Understanding smart-cloning

When a data volume is smart-cloned, the following occurs:

1. A snapshot of the source persistent volume claim (PVC) is created.
2. A PVC is created from the snapshot.
3. The snapshot is deleted.

#### 7.18.10.2. Cloning a data volume

```yaml
resources:
  requests:
    storage: 500Mi
```
**Prerequisites**

For smart-cloning to occur, the following conditions are required.

- Your storage provider must support snapshots.
- The source and target PVCs must be defined to the same namespace.
- The source and target PVCs must be defined to the same storage class.
- The **VolumeSnapshotClass** object must reference the storage class defined to both the source and target PVCs.

If any of these prerequisites are not met, host-assisted cloning automatically occurs when you create a data volume with a PVC source.

**Procedure**

To initiate cloning of a data volume:

1. Create a YAML file for a **DataVolume** object that specifies the name of the new data volume, the name and namespace of the source PVC, and the size of the new data volume. This example clones a source PVC in block mode, so **volumeMode: Block** is used:

   ```yaml
   apiVersion: cdi.kubevirt.io/v1beta1
   kind: DataVolume
   metadata:
     name: <cloner-datavolume>  
   spec:
     source:
       pvc:
         namespace: "<source-namespace>",
         name: "<my-favorite-vm-disk>"
       accessModes:
         - ReadWriteMany
       resources:
         requests:
           storage: <2Gi>
       volumeMode: Block
   ``

   - **1** The name of the new data volume.
   - **2** The namespace where the source PVC exists.
   - **3** The name of the source PVC.
   - **4** The size of the new data volume. You must allocate enough space, or the cloning operation fails. The size must be the same as or larger than the source PVC.
   - **5** Specifies that the destination is a block PV.

2. Start cloning the PVC by creating the data volume:

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

Data volumes prevent a virtual machine from starting before the PVC is prepared, so you can create a virtual machine that references the new data volume while the PVC clones.

7.18.10.3. Additional resources

- Cloning the persistent volume claim of a virtual machine disk into a new data volume

7.18.11. Storage defaults for data volumes

The `kubevirt-storage-class-defaults` config map provides access mode and volume mode defaults for data volumes. You can edit or add storage class defaults to the config map in order to create data volumes in the web console that better match the underlying storage.

7.18.11.1. About storage settings for data volumes

Data volumes require a defined access mode and volume mode to be created in the web console. These storage settings are configured by default with a `ReadWriteOnce` access mode and `Filesystem` volume mode.

You can modify these settings by editing the `kubevirt-storage-class-defaults` config map in the `openshift-cnv` namespace. You can also add settings for other storage classes in order to create data volumes in the web console for different storage types.

NOTE

You must configure storage settings that are supported by the underlying storage.

All data volumes that you create in the web console use the default storage settings unless you specify a storage class that is also defined in the config map.

7.18.11.1.1. Access modes

Data volumes support the following access modes:

- **ReadWriteOnce**: The volume can be mounted as read-write by a single node. `ReadWriteOnce` has greater versatility and is the default setting.

- **ReadWriteMany**: The volume can be mounted as read-write by many nodes. `ReadWriteMany` is required for some features, such as live migration of virtual machines between nodes.

`ReadWriteMany` is recommended if the underlying storage supports it.

7.18.11.1.2. Volume modes

The volume mode defines if a volume is intended to be used with a formatted file system or to remain in raw block state. Data volumes support the following volume modes:

- **Filesystem**: Creates a file system on the data volume. This is the default setting.

- **Block**: Creates a block data volume. Only use **Block** if the underlying storage supports it.
7.18.11.2. Editing the kubevirt-storage-class-defaults config map in the web console

Modify the storage settings for data volumes by editing the `kubevirt-storage-class-defaults` config map in the `openshift-cnv` namespace. You can also add settings for other storage classes in order to create data volumes in the web console for different storage types.

**NOTE**

You must configure storage settings that are supported by the underlying storage.

**Procedure**

1. Click `Workloads → Config Maps` from the side menu.
2. In the `Project` list, select `openshift-cnv`.
3. Click `kubevirt-storage-class-defaults` to open the `Config Map Overview`.
4. Click the `YAML` tab to display the editable configuration.
5. Update the `data` values with the storage configuration that is appropriate for your underlying storage:

   ```yaml
   ... 
   data:
     accessMode: ReadWriteOnce 1
     volumeMode: Filesystem 2
     <new>.accessMode: ReadWriteMany 3
     <new>.volumeMode: Block 4
   
   1 The default accessMode is `ReadWriteOnce`.
   2 The default volumeMode is `Filesystem`.
   3 If you add an access mode for a storage class, replace the `<new>` part of the parameter with the storage class name.
   4 If you add a volume mode for a storage class, replace the `<new>` part of the parameter with the storage class name.

6. Click `Save` to update the config map.

7.18.11.3. Editing the kubevirt-storage-class-defaults config map in the CLI

Modify the storage settings for data volumes by editing the `kubevirt-storage-class-defaults` config map in the `openshift-cnv` namespace. You can also add settings for other storage classes in order to create data volumes in the web console for different storage types.

**NOTE**

You must configure storage settings that are supported by the underlying storage.

**Procedure**
1. Edit the config map by running the following command:

```
$ oc edit configmap kubevirt-storage-class-defaults -n openshift-cnv
```

2. Update the data values of the config map:

```
...data:
  accessMode: ReadWriteOnce 1
  volumeMode: Filesystem 2
  <new>.accessMode: ReadWriteMany 3
  <new>.volumeMode: Block 4
```

1. The default accessMode is `ReadWriteOnce`.
2. The default volumeMode is `Filesystem`.
3. If you add an access mode for a storage class, replace the `<new>` part of the parameter with the storage class name.
4. If you add a volume mode for a storage class, replace the `<new>` part of the parameter with the storage class name.

3. Save and exit the editor to update the config map.

### 7.18.11.4. Example of multiple storage class defaults

The following YAML file is an example of a `kubevirt-storage-class-defaults` config map that has storage settings configured for two storage classes, `migration` and `block`.

Ensure that all settings are supported by your underlying storage before you update the config map.

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: kubevirt-storage-class-defaults
  namespace: openshift-cnv
...
data:
  accessMode: ReadWriteOnce
  volumeMode: Filesystem
  nfs-sc.accessMode: ReadWriteMany
  nfs-sc.volumeMode: Filesystem
  block-sc.accessMode: ReadWriteMany
  block-sc.volumeMode: Block
```

### 7.18.12. Creating and using default OS images

A default operating system (OS) image is a bootable disk containing an OS and all of the configuration settings for the OS, such as drivers. You use a default OS image to create bootable virtual machines with specific configurations.

To use default OS images, you must install the latest version of OpenShift Virtualization. You then
create a persistent volume claim (PVC) by using the OpenShift Container Platform web console to upload a default OS image to the PVC. After uploading, you create virtual machines or virtual machine templates from the uploaded images by using the wizard in the web console.

7.18.12.1. Creating persistent volume claims to store default OS images

Follow these steps to create a persistent volume claim (PVC), which you use to upload and save default operating system (OS) images.

Prerequisites

- You must be logged in as a user with the `os-images.kubevirt.io:edit` RBAC role or as an administrator.

Procedure

1. Select a local image of a bootable disk that you want to upload and save as a default OS image.
2. In the OpenShift Container Platform web console, click `Storage > Persistent Volume Claims` in the sidebar menu. The `Persistent Volume Claim` page displays.
3. Click the `Create Persistent Volume Claim` button and select the `With Data upload form` option.
4. Complete the `Upload Data to Persistent Volume Claim` form to create a PVC that is used to upload and save default OS images.
   a. Click `Browse` and locate a bootable image to upload and save as a default OS image.
   b. Select the `Attach this data to a Virtual Machine operating system` check box.
   c. Select the OS of the bootable disk that you want to upload from the `Operating System` list.
   d. Select the storage class you want to use in the `Storage Class` list.
   e. Enter the size of the PVC you are creating in the `Size` fields.
   f. Select the `Access Mode`.
   g. Click `Create` to create the PVC.

The `Persistent Volume Claim Details` screen displays information about the PVC that you created.

7.18.12.2. Creating a virtual machine from a default OS image

After you have created a persistent volume claim (PVC) for default operating system (OS) images, you create a new virtual machine or virtual machine template from a default OS image that you uploaded to the PVC.

**NOTE**

You can create virtual machines and virtual machine templates from default OS images with non-administrator user privileges. You need administrator privileges to upload and save default OS images to PVCs.
Prerequisites

- When you created the PVC, you selected the Attach this data to a Virtual Machine operating system check box.

Procedure

1. In the OpenShift Container Platform web console, click Workloads > Virtualization in the sidebar menu. The Virtualization page displays. Use web console Help or existing documentation to create virtual machine templates.

2. Create a virtual machine or virtual machine template from an uploaded default OS image.
   
   
   b. On the General wizard page, select an OS from the Operating System list that displays the (Source available) label next to the OS and version name. The (Source available) label indicates default OS images are available for this OS.
   
   c. Ensure that the Clone available operating system source check box is selected.
   
   d. Click the Review and Confirm button.
   
   e. On the Review and confirm your settings wizard page, review information about your virtual machine and change them, if required.
   
   f. Click Create Virtual Machine to create your virtual machine. The Successfully created virtual machine page displays with links to See virtual machine details or Go to list to display the list of virtual machines and virtual machine templates.

7.18.12.3. Additional resources

- Uploading local disk images

7.18.13. Using container disks with virtual machines

You can build a virtual machine image into a container disk and store it in your container registry. You can then import the container disk into persistent storage for a virtual machine or attach it directly to the virtual machine for ephemeral storage.

IMPORTANT

If you use large container disks, I/O traffic might increase, impacting worker nodes. This can lead to unavailable nodes. You can resolve this by:

- Pruning DeploymentConfig objects
- Configuring garbage collection

7.18.13.1. About container disks

A container disk is a virtual machine image that is stored as a container image in a container image registry. You can use container disks to deliver the same disk images to multiple virtual machines and to create large numbers of virtual machine clones.
A container disk can either be imported into a persistent volume claim (PVC) by using a data volume that is attached to a virtual machine, or attached directly to a virtual machine as an ephemeral containerDisk volume.

### 7.18.13.1.1. Importing a container disk into a PVC by using a data volume

Use the Containerized Data Importer (CDI) to import the container disk into a PVC by using a data volume. You can then attach the data volume to a virtual machine for persistent storage.

### 7.18.13.1.2. Attaching a container disk to a virtual machine as a containerDisk volume

A containerDisk volume is ephemeral. It is discarded when the virtual machine is stopped, restarted, or deleted. When a virtual machine with a containerDisk volume starts, the container image is pulled from the registry and hosted on the node that is hosting the virtual machine.

Use containerDisk volumes for read-only file systems such as CD-ROMs or for disposable virtual machines.

**IMPORTANT**

Using containerDisk volumes for read-write file systems is not recommended because the data is temporarily written to local storage on the hosting node. This slows live migration of the virtual machine, such as in the case of node maintenance, because the data must be migrated to the destination node. Additionally, all data is lost if the node loses power or otherwise shuts down unexpectedly.

### 7.18.13.2. Preparing a container disk for virtual machines

You must build a container disk with a virtual machine image and push it to a container registry before it can used with a virtual machine. You can then either import the container disk into a PVC using a data volume and attach it to a virtual machine, or you can attach the container disk directly to a virtual machine as an ephemeral containerDisk volume.

The size of a disk image inside a container disk is limited by the maximum layer size of the registry where the container disk is hosted.

**NOTE**

For Red Hat Quay, you can change the maximum layer size by editing the YAML configuration file that is created when Red Hat Quay is first deployed.

**Prerequisites**

- Install podman if it is not already installed.
- The virtual machine image must be either QCOW2 or RAW format.

**Procedure**

1. Create a Dockerfile to build the virtual machine image into a container image. The virtual machine image must be owned by QEMU, which has a UID of 107, and placed in the /disk/directory inside the container. Permissions for the /disk/directory must then be set to 0440.
OpenShift Container Platform 4.6 OpenShift Virtualization

The following example uses the Red Hat Universal Base Image (UBI) to handle these
configuration changes in the first stage, and uses the minimal scratch image in the second
stage to store the result:
$ cat > Dockerfile << EOF
FROM registry.access.redhat.com/ubi8/ubi:latest AS builder
ADD --chown=107:107 <vm_image>.qcow2 /disk/ 1
RUN chmod 0440 /disk/*
FROM scratch
COPY --from=builder /disk/* /disk/
EOF
1

Where <vm_image> is the virtual machine image in either QCOW2 or RAW format.
To use a remote virtual machine image, replace <vm_image>.qcow2 with the complete url
for the remote image.

2. Build and tag the container:
$ podman build -t <registry>/<container_disk_name>:latest .
3. Push the container image to the registry:
$ podman push <registry>/<container_disk_name>:latest
If your container registry does not have TLS you must add it as an insecure registry before you can
import container disks into persistent storage.

7.18.13.3. Disabling TLS for a container registry to use as insecure registry
You can disable TLS (transport layer security) for a container registry by adding the registry to the cdiinsecure-registries config map.
Prerequisites
Log in to the cluster as a user with the cluster-admin role.
Procedure
Add the registry to the cdi-insecure-registries config map in the openshift-cnv namespace.
$ oc patch configmap cdi-insecure-registries -n openshift-cnv \
--type merge -p '{"data":{"mykey": "<insecure-registry-host>:5000"}}' 1
1

Replace <insecure-registry-host> with the registry hostname.

7.18.13.4. Next steps
Import the container disk into persistent storage for a virtual machine .
Create a virtual machine that uses a containerDisk volume for ephemeral storage.

194


7.18.14. Preparing CDI scratch space

7.18.14.1. About data volumes

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

7.18.14.2. 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, CDI provisions a scratch space PVC equal to the size of the PVC backing the destination data volume (DV). The scratch space PVC is deleted after the operation completes or aborts.

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

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

**NOTE**

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

**Manual provisioning**

If there are no storage classes, CDI uses 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 remains in a **Pending** state until an appropriate PVC is made available or until a timeout function kills the pod.

7.18.14.3. CDI operations that require scratch space

<table>
<thead>
<tr>
<th>Type</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry imports</td>
<td>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.</td>
</tr>
<tr>
<td>Upload image</td>
<td>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.</td>
</tr>
<tr>
<td>HTTP imports of archived images</td>
<td>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.</td>
</tr>
</tbody>
</table>
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, CDI downloads the image to scratch space before passing the file to QEMU-IMG.

### 7.18.14.4. Defining a storage class in the CDI configuration

Define a storage class 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 storage class in the cluster.

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

- API Version: `cdi.kubevirt.io/v1beta1`
- kind: `CDIConfig`
- metadata:
  - name: `config`
  - ...
- spec:
  - `scratchSpaceStorageClass: "<storage_class>"`
  - ...

### 7.18.14.5. CDI supported operations matrix

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

<table>
<thead>
<tr>
<th>Content types</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>HTTP basic auth</th>
<th>Registry</th>
<th>Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>KubeVirt(QCOW2)</td>
<td>✔ QCOW2 ✔ GZ* ✔ XZ*</td>
<td>✔ QCOW2** ✔ GZ* ✔ XZ*</td>
<td>✔ QCOW2 ✔ GZ* ✔ XZ*</td>
<td>✔ QCOW2*</td>
<td>✔ QCOW2*</td>
</tr>
<tr>
<td></td>
<td>✔ RAW ✔ GZ ✔ XZ</td>
<td>✔ RAW ✔ GZ ✔ XZ</td>
<td>✔ RAW ✔ GZ ✔ XZ</td>
<td>✔ RAW*</td>
<td>✔ RAW*</td>
</tr>
<tr>
<td>KubeVirt(RAW)</td>
<td>✔ RAW ✔ GZ ✔ XZ</td>
<td>✔ RAW ✔ GZ ✔ XZ</td>
<td>✔ RAW ✔ GZ ✔ XZ</td>
<td>✔ RAW*</td>
<td>✔ RAW*</td>
</tr>
</tbody>
</table>

- ✔ Supported operation
Unsupported operation

* Requires scratch space

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

Additional resources

- See the Dynamic provisioning section for more information on storage classes and how these are defined in the cluster.

7.18.15. Re-using persistent volumes

In order to re-use a statically provisioned persistent volume (PV), you must first reclaim the volume. This involves deleting the PV so that the storage configuration can be re-used.

7.18.15.1. About reclaiming statically provisioned persistent volumes

When you reclaim a persistent volume (PV), you unbind the PV from a persistent volume claim (PVC) and delete the PV. Depending on the underlying storage, you might need to manually delete the shared storage.

You can then re-use the PV configuration to create a PV with a different name.

Statically provisioned PVs must have a reclaim policy of Retain to be reclaimed. If they do not, the PV enters a failed state when the PVC is unbound from the PV.

IMPORTANT

The Recycle reclaim policy is deprecated in OpenShift Container Platform 4.

7.18.15.2. Reclaiming statically provisioned persistent volumes

Reclaim a statically provisioned persistent volume (PV) by unbinding the persistent volume claim (PVC) and deleting the PV. You might also need to manually delete the shared storage.

Reclaiming a statically provisioned PV is dependent on the underlying storage. This procedure provides a general approach that might need to be customized depending on your storage.

Procedure

1. Ensure that the reclaim policy of the PV is set to Retain:
   
   a. Check the reclaim policy of the PV:

   ```
   $ oc get pv <pv_name> -o yaml | grep 'persistentVolumeReclaimPolicy'
   ```

   b. If the persistentVolumeReclaimPolicy is not set to Retain, edit the reclaim policy with the following command:

   ```
   $ oc patch pv <pv_name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
   ```

2. Ensure that no resources are using the PV:
Remove any resources that use the PVC before continuing.

3. Delete the PVC to release the PV:

   $ oc delete pvc <pvc_name>

4. Optional: Export the PV configuration to a YAML file. If you manually remove the shared storage later in this procedure, you can refer to this configuration. You can also use `spec` parameters in this file as the basis to create a new PV with the same storage configuration after you reclaim the PV:

   $ oc get pv <pv_name> -o yaml > <file_name>.yaml

5. Delete the PV:

   $ oc delete pv <pv_name>

6. Optional: Depending on the storage type, you might need to remove the contents of the shared storage folder:

   $ rm -rf <path_to_share_storage>

7. Optional: Create a PV that uses the same storage configuration as the deleted PV. If you exported the reclaimed PV configuration earlier, you can use the `spec` parameters of that file as the basis for a new PV manifest:

   ```
   NOTE
   To avoid possible conflict, it is good practice to give the new PV object a different name than the one that you deleted.
   ```

   $ oc create -f <new_pv_name>.yaml

Additional resources

- Configuring local storage for virtual machines
- The OpenShift Container Platform Storage documentation has more information on Persistent Storage.

### 7.18.16. Deleting data volumes

You can manually delete a data volume by using the `oc` command-line interface.

#### 7.18.16.1. About data volumes

**NOTE**

When you delete a virtual machine, the data volume it uses is automatically deleted.
DataVolume objects are custom resources that are provided by the Containerized Data Importer (CDI) project. Data volumes orchestrate import, clone, and upload operations that are associated with an underlying persistent volume claim (PVC). Data volumes are integrated with OpenShift Virtualization, and they prevent a virtual machine from being started before the PVC has been prepared.

7.18.16.2. Listing all data volumes

You can list the data volumes in your cluster by using the `oc` command-line interface.

**Procedure**

- List all data volumes by running the following command:

  ```
  $ oc get dvs
  ```

7.18.16.3. Deleting a data volume

You can delete a data volume by using the `oc` command-line interface (CLI).

**Prerequisites**

- Identify the name of the data volume that you want to delete.

**Procedure**

- Delete the data volume by running the following command:

  ```
  $ oc delete dv <datavolume_name>
  ```

**NOTE**

This command only deletes objects that exist in the current project. Specify the `-n <project_name>` option if the object you want to delete is in a different project or namespace.
CHAPTER 8. VIRTUAL MACHINE TEMPLATES

8.1. CREATING VIRTUAL MACHINE TEMPLATES

You can use virtual machine templates to create multiple virtual machines that have similar configurations. After a template is created, reference the template when creating virtual machines.

8.1.1. 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 General, Networking, Storage, Advanced, and Review steps 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 step until you provide values in the required fields.

Procedure

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

NOTE

If either URL or Container are selected as the Source in the General step, 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.
7. Click Create Virtual Machine Template > The Results screen displays the JSON configuration file for the virtual machine template. The template is listed in the Virtual Machine Templates tab.

### 8.1.2. 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.

#### 8.1.2.1. Virtual machine template wizard fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>PXE</td>
<td>Provision virtual machine from PXE menu. Requires a PXE-capable NIC in the cluster.</td>
</tr>
<tr>
<td>URL</td>
<td></td>
<td>Provision virtual machine from an image available from an HTTP or S3 endpoint.</td>
</tr>
<tr>
<td>Container</td>
<td></td>
<td>Provision virtual machine from a bootable operating system container located in a registry accessible from the cluster. Example: <code>kubevirt/cirros-registry-disk-demo</code>.</td>
</tr>
<tr>
<td>Disk</td>
<td></td>
<td>Provision virtual machine from a disk.</td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
<td>The primary operating system that is selected for the virtual machine.</td>
</tr>
<tr>
<td>Flavor</td>
<td>small, medium, large, tiny, Custom</td>
<td>Presets that determine the amount of CPU and memory allocated to the virtual machine. The presets displayed for Flavor are determined by the operating system.</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td>Size in GiB of the memory allocated to the virtual machine.</td>
</tr>
<tr>
<td>CPUs</td>
<td></td>
<td>The amount of CPU allocated to the virtual machine.</td>
</tr>
<tr>
<td>Workload Profile</td>
<td>High Performance</td>
<td>A virtual machine configuration that is optimized for high-performance workloads.</td>
</tr>
<tr>
<td>Name</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>A profile optimized to run server workloads.</td>
</tr>
<tr>
<td></td>
<td>Desktop</td>
<td>A virtual machine configuration for use on a desktop.</td>
</tr>
</tbody>
</table>

**Name**

The name can contain lowercase 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 uppercase letters, spaces, periods (.), or special characters.

**Description**

Optional description field.

### 8.1.2.2. Cloud-init fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Sets a specific hostname for the virtual machine.</td>
</tr>
<tr>
<td>Authenticated SSH Keys</td>
<td>The user's public key that is copied to ~/.ssh/authorized_keys on the virtual machine.</td>
</tr>
<tr>
<td>Custom script</td>
<td>Replaces other options with a field in which you paste a custom cloud-init script.</td>
</tr>
</tbody>
</table>

### 8.1.2.3. Networking fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the network interface controller.</td>
</tr>
<tr>
<td>Model</td>
<td>Indicates the model of the network interface controller. Supported values are e1000e and virtio.</td>
</tr>
<tr>
<td>Network</td>
<td>List of available network attachment definitions.</td>
</tr>
</tbody>
</table>
**Type**

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.

**MAC Address**

MAC address for the network interface controller. If a MAC address is not specified, one is assigned automatically.

### 8.1.2.4. Storage fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>Select a blank disk for the virtual machine or choose from the options available: URL, Container, Attach Cloned Disk, or Attach Disk. To select an existing disk and attach it to the virtual machine, choose Attach Cloned Disk or Attach Disk from a list of available persistent volume claims (PVCs).</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Name of the disk. The name can contain lowercase 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 uppercase letters, spaces, or special characters.</td>
</tr>
<tr>
<td><strong>Size (GiB)</strong></td>
<td>Size, in GiB, of the disk.</td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td>Type of disk device. Supported interfaces are virtIO, SATA, and SCSI.</td>
</tr>
<tr>
<td><strong>Storage Class</strong></td>
<td>The storage class that is used to create the disk.</td>
</tr>
<tr>
<td><strong>Advanced → Volume Mode</strong></td>
<td>Defines whether the persistent volume uses a formatted file system or raw block state. Default is Filesystem.</td>
</tr>
<tr>
<td><strong>Advanced → Access Mode</strong></td>
<td>Access mode of the persistent volume. Supported access modes are ReadWriteOnce, ReadOnlyMany, and ReadWriteMany.</td>
</tr>
</tbody>
</table>

**Advanced storage settings**

The following advanced storage settings are available for Blank, Import via URL, and Clone existing PVC disks. These parameters are optional. If you do not specify these parameters, the system uses the default values from the **kubevirt-storage-class-defaults** config map.
<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Mode</td>
<td>Filesystem</td>
<td>Stores the virtual disk on a file system-based volume.</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>Stores the virtual disk directly on the block volume. Only use <strong>Block</strong> if</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the underlying storage supports it.</td>
</tr>
<tr>
<td>Access Mode</td>
<td>Single User (RWO)</td>
<td>The disk can be mounted as read/write by a single node.</td>
</tr>
<tr>
<td></td>
<td>Shared Access (RWX)</td>
<td>The disk can be mounted as read/write by many nodes.</td>
</tr>
<tr>
<td></td>
<td>Read Only (ROX)</td>
<td>The disk can be mounted as read-only by many nodes.</td>
</tr>
</tbody>
</table>

**NOTE**
This is required for some features, such as live migration of virtual machines between nodes.

### 8.2. EDITING VIRTUAL MACHINE TEMPLATES

You can update a virtual machine template in the web console, either by editing the full configuration in the YAML editor or by editing a subset of the parameters in the Virtual Machine Template Overview screen.

#### 8.2.1. Editing a virtual machine template in the web console

Edit select values of a virtual machine template in the Virtual Machine Template Overview screen of the web console by clicking on the pencil icon next to the relevant field. Other values can be edited using the CLI.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.
2. Click the **Virtual Machine Templates** tab.
3. Select a virtual machine template to open the Virtual Machine Template Overview screen.
4. Click the **Details** tab.
5. Click the pencil icon to make a field editable.
6. Make the relevant changes and click **Save**.

Editing a virtual machine template will not affect virtual machines already created from that template.

#### 8.2.2. Editing virtual machine template YAML configuration 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 OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machine Templates** tab.

3. Select a template.

4. Click the **YAML** tab to display the editable configuration.

5. Edit the file and click **Save**.

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

### 8.2.3. Adding a virtual disk to a virtual machine template

Use this procedure to add a virtual disk to a virtual machine template.

**Procedure**

1. Click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machine Templates** tab.

3. Select a virtual machine template to open the **Virtual Machine Template Overview** screen.

4. Click the **Disks** tab.

5. Click **Add Disk** to open the **Add Disk** window.

6. In the **Add Disk** window, specify the **Source**, **Name**, **Size**, **Interface**, **Type**, and **Storage Class**.

   a. Optional: In the **Advanced** list, specify the **Volume Mode** and **Access Mode** for the virtual disk. If you do not specify these parameters, the system uses the default values from the **kubevirt-storage-class-defaults** config map.

7. Click **Add**.

### 8.2.4. Adding a network interface to a virtual machine template

Use this procedure to add a network interface to a virtual machine template.

**Procedure**
1. Click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machine Templates** tab.

3. Select a virtual machine template to open the **Virtual Machine Template Overview** screen.

4. Click the **Network Interfaces** tab.

5. Click **Add Network Interface**.

6. In the **Add Network Interface** window, specify the **Name**, **Model**, **Network**, **Type**, and **MAC Address** of the network interface.

7. Click **Add**.

**8.2.5. Editing CD-ROMs for Virtual Machine Templates**

Use the following procedure to configure CD-ROMs for virtual machines.

**Procedure**

1. From the **Virtual Machine Templates** tab, select your virtual machine template.

2. Select the **Overview** tab.

3. To add or edit a CD-ROM configuration, click the pencil icon to the right of the **CD-ROMs** label. The **Edit CD-ROM** window opens.

   - If CD-ROMs are unavailable for editing, the following message displays: **The virtual machine doesn’t have any CD-ROMs attached.**

   - If there are CD-ROMs available, you can remove a CD-ROM by clicking `-`.

4. In the **Edit CD-ROM** window, do the following:

   a. Select the type of CD-ROM configuration from the drop-down list for **Media Type**. CD-ROM configuration types are **Container**, **URL**, and **Persistent Volume Claim**.

   b. Complete the required information for each **Type**.

   c. When all CD-ROMs are added, click **Save**.

**8.3. ENABLING DEDICATED RESOURCES FOR VIRTUAL MACHINE TEMPLATES**

Virtual machines can have resources of a node, such as CPU, dedicated to them in order to improve performance.

**8.3.1. About dedicated resources**

When you enable dedicated resources for your virtual machine, your virtual machine’s workload is scheduled on CPUs that will not be used by other processes. By using dedicated resources, you can improve the performance of the virtual machine and the accuracy of latency predictions.

**8.3.2. Prerequisites**
The CPU Manager must be configured on the node. Verify that the node has the `cpumanager = true` label before scheduling virtual machine workloads.

8.3.3. Enabling dedicated resources for a virtual machine template

You can enable dedicated resources for a virtual machine template in the Virtual Machine Template Overview page of the web console.

Procedure

1. Click Workloads → Virtual Machine Templates from the side menu.
2. Select a virtual machine template to open the Virtual Machine Template Overview page.
3. Click the Details tab.
4. Click the pencil icon to the right of the Dedicated Resources field to open the Dedicated Resources window.
5. Select Schedule this workload with dedicated resources (guaranteed policy)
6. Click Save.

8.4. DELETING A VIRTUAL MACHINE TEMPLATE

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

8.4.1. Deleting a virtual machine template in the web console

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

Procedure

1. In the OpenShift Virtualization console, click Workloads → Virtualization from the side menu.
2. Click the Virtual Machine Templates tab.
3. 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.
4. In the confirmation pop-up window, click Delete to permanently delete the template.
CHAPTER 9. LIVE MIGRATION

9.1. VIRTUAL MACHINE LIVE MIGRATION

9.1.1. Understanding live migration

Live migration is the process of moving a running virtual machine instance (VMI) to another node in the cluster without interrupting the virtual workload or access. If a VMI uses the `LiveMigrate` eviction strategy, it automatically migrates when the node that the VMI runs on is placed into maintenance mode. You can also manually start live migration by selecting a VMI to migrate.

Virtual machines must have a persistent volume claim (PVC) with a shared ReadWriteMany (RWX) access mode to be live migrated.

**NOTE**

Live migration is not supported for virtual machines that are attached to an SR-IOV network interface.

9.1.2. Updating access mode for live migration

For live migration to function properly, you must use the ReadWriteMany (RWX) access mode. Use this procedure to update the access mode, if needed.

**Procedure**

- To set the RWX access mode, run the following `oc patch` command:

  ```bash
  $ oc patch -n openshift-cnv \
  cm kubevirt-storage-class-defaults \
  -p '{"data":{"$<STORAGE_CLASS>.accessMode":"ReadWriteMany"}}'
  ```

**Additional resources:**

- Migrating a virtual machine instance to another node
- Node maintenance mode
- Live migration limiting
- Storage defaults for data volumes

9.2. 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.

9.2.1. 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 `openshift-cnv` namespace.
Procedure

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

```bash
$ oc edit configmap kubevirt-config -n openshift-cnv
```

Example configuration file

```yaml
apiVersion: v1
data:
default-network-interface: masquerade
feature-gates: DataVolumes,SRIOV,LiveMigration,CPUPermission,CPUNodeDiscovery,Sidecar,Snapshot
migrations: |-
  parallelMigrationsPerCluster: "5"
  parallelOutboundMigrationsPerNode: "2"
  bandwidthPerMigration: "64Mi"
  completionTimeoutPerGiB: "800"
  progressTimeout: "150"
  machine-type: pc-q35-rhel8.3.0
  selinuxLauncherType: virt_launcher.process
smbios: |
  Family: Red Hat
  Product: Container-native virtualization
  Manufacturer: Red Hat
  Sku: 2.6.0
  Version: 2.6.0
kind: ConfigMap
metadata:
creationTimestamp: "2021-03-26T18:01:04Z"
labels:
  app: kubevirt-hyperconverged
  name: kubevirt-config
namespace: openshift-cnv
resourceVersion: "15371295"
selfLink: /api/v1/namespaces/openshift-cnv/configmaps/kubevirt-config
uid: <uuid>
```

### 9.2.2. Cluster-wide live migration limits and timeouts

Table 9.1. Migration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>parallelMigrationsPerCluster</code></td>
<td>Number of migrations running in parallel in the cluster.</td>
<td>5</td>
</tr>
<tr>
<td><code>parallelOutboundMigrationsPerNode</code></td>
<td>Maximum number of outbound migrations per node.</td>
<td>2</td>
</tr>
<tr>
<td><code>bandwidthPerMigration</code></td>
<td>Bandwidth limit of each migration, in MiB/s.</td>
<td>64Mi</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>completionTimeoutPerGiB</td>
<td>The migration is canceled if it has not completed in this time, in seconds per GiB of memory. For example, a virtual machine instance with 6GiB memory times out 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.</td>
<td>800</td>
</tr>
<tr>
<td>progressTimeout</td>
<td>The migration is canceled if memory copy fails to make progress in this time, in seconds.</td>
<td>150</td>
</tr>
</tbody>
</table>

### 9.3. 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.

#### 9.3.1. 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 OpenShift Virtualization console, click Workloads → Virtualization from the side menu.

2. Click the Virtual Machines tab.

3. 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 Overview 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 Overview screen and click Actions → Migrate Virtual Machine

4. Click Migrate to migrate the virtual machine to another node.

#### 9.3.2. 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`:

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

2. Create the object in the cluster by running the following command:

   `$ 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

### 9.4. 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.

#### 9.4.1. 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** tab or in the **Virtual Machine Overview** screen for the migrating virtual machine.

Procedure

1. In the OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. Select a virtual machine to open the **Virtual Machine Overview** screen.

#### 9.4.2. 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:

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

**Example output**

```
...
Status: Complete
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
```

### 9.5. 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.

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

You can cancel a live migration of the virtual machine instance using the Options menu found on each virtual machine in the Virtualization → Virtual Machines tab, or from the Actions menu available on all tabs in the Virtual Machine Overview screen.

**Procedure**

1. In the OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.

2. Click the **Virtual Machines** tab.

3. You can cancel the migration from this screen, which makes it easier to perform actions on multiple virtual machines, or from the Virtual Machine Overview 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**.
Select a virtual machine name to open the **Virtual Machine Overview** screen and click **Actions → Cancel Virtual Machine Migration**.

4. Click **Cancel Migration** to cancel the virtual machine live migration.

### 9.5.2. 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:

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

### 9.6. 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.

#### 9.6.1. 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.template.spec** section in the virtual machine configuration file. This example uses **oc edit** to update the relevant snippet of the **VirtualMachine** configuration file:

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

   ```yaml
   apiVersion: kubevirt.io/v1alpha3
   kind: VirtualMachine
   metadata:
     name: custom-vm
   spec:
     template:
       spec:
         evictionStrategy: LiveMigrate
   ...
   ```

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

   ```bash
   $ virtctl restart <custom-vm> -n <my-namespace>
   ```
CHAPTER 10. NODE MAINTENANCE

10.1. AUTOMATIC RENEWAL OF TLS CERTIFICATES

All TLS certificates for OpenShift Virtualization components are renewed and rotated automatically. You are not required to refresh them manually.

10.1.1. Automatic renewal of TLS certificates

TLS certificates are automatically deleted and replaced according to the following schedule:

- KubeVirt certificates are renewed daily.
- Containerized Data Importer controller (CDI) certificates are renewed every 15 days.
- MAC pool certificates are renewed every year.

Automatic TLS certificate rotation does not disrupt any operations. For example, the following operations continue to function without any disruption:

- Migrations
- Image uploads
- VNC and console connections

10.2. MANAGING NODE LABELING FOR OBSOLETÉ CPU MODELS

You can schedule a virtual machine (VM) on a node where the CPU model and policy attribute of the VM are compatible with the CPU models and policy attributes that the node supports. By specifying a list of obsolete CPU models in a config map, you can exclude them from the list of labels created for CPU models.

10.2.1. Understanding node labeling for obsolete CPU models

To ensure that a node supports only valid CPU models for scheduled VMs, create a config map with a list of obsolete CPU models. When the node-labeller obtains the list of obsolete CPU models, it eliminates those CPU models and creates labels for valid CPU models.

NOTE

If you do not configure a config map with a list of obsolete CPU models, all CPU models are evaluated for labels, including obsolete CPU models that are not present in your environment.

Through the process of iteration, the list of base CPU features in the minimum CPU model are eliminated from the list of labels generated for the node. For example, an environment might have two supported CPU models: Penryn and Haswell.

If Penryn is specified as the CPU model for minCPU, the node-labeller evaluates each base CPU feature for Penryn and compares it with each CPU feature supported by Haswell. If the CPU feature is supported by both Penryn and Haswell, the node-labeller eliminates that feature from the list of CPU features.
features for creating labels. If a CPU feature is supported only by Haswell and not by Penryn, that CPU feature is included in the list of generated labels. The node-labeller follows this iterative process to eliminate base CPU features that are present in the minimum CPU model and create labels.

The following example shows the complete list of CPU features for Penryn which is specified as the CPU model for minCPU:

**Example of CPU features for Penryn**

```
apic
clflush
cmov
cx16
cx8
de
fpu
fxsr
lahf_lm
lm
mca
mce
mmx
msr
mtrr
nx
pae
pat
pge
pni
pse
pse36
sep
sse
sse2
sse4.1
ssse3
syscall
tsc
```

The following example shows the complete list of CPU features for Haswell:

**Example of CPU features for Haswell**

```
aes
apic
avx
avx2
bmi1
bmi2
clflush
cmov
cx16
cx8
de
erms
```
The following example shows the list of node labels generated by the node-labeller after iterating and comparing the CPU features for Penryn with the CPU features for Haswell:

**Example of node labels after iteration**

- aes
- avx
- avx2
- bmi1
- bmi2
- erms
- fma
- fsbs
- hle
- invpcid
- movbe
- msr
- mtrr
- nx
- pae
- pat
- pcid
- pclmulqd
- pge
- pni
- popcnt
- pse
- pse36
- rdtscp
- rtm
- sep
- smep
- sse
- sse2
- sse4.1
- sse4.2
- ssse3
- syscall
- tsc
tsc-deadline
- x2apic
- xsave
10.2.2. Configuring a config map for obsolete CPU models

Use this procedure to configure a config map for obsolete CPU models.

**Procedure**

- Create a `ConfigMap` object, specifying the obsolete CPU models in the `obsoleteCPUs` array. For example:

```yaml
apiVersion: v1
class: ConfigMap
metadata:
  name: cpu-plugin-configmap
  data:
    cpu-plugin-configmap:
      obsoleteCPUs:
        - "486"
        - "pentium"
        - "pentium2"
        - "pentium3"
        - "pentiumpro"
      minCPU: "Penryn"

1. Name of the config map.
2. Configuration data.
3. List of obsolete CPU models.
4. Minimum CPU model that is used for basic CPU features.

10.3. NODE MAINTENANCE MODE

10.3.1. 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.
IMPORTANT
Virtual machines must have a persistent volume claim (PVC) with a shared ReadWriteMany (RWX) access mode to be live migrated.

Additional resources:
- Virtual machine live migration
- Configuring virtual machine eviction strategy

10.4. SETTING A NODE TO MAINTENANCE MODE

10.4.1. 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.

IMPORTANT
Virtual machines must have a persistent volume claim (PVC) with a shared ReadWriteMany (RWX) access mode to be live migrated.

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

10.4.2. 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 OpenShift 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.

### 10.4.3. 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`:

   ```yaml
   apiVersion: nodemaintenance.kubevirt.io/v1beta1
group: kubevirt.io
kind: NodeMaintenance
metadata:
  name: node02-maintenance
spec:
  nodeName: node02
  reason: "Replacing node02"
   
   $ 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](#)

### 10.5. 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.

#### 10.5.1. 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 OpenShift 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.

## 10.5.2. 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:

```bash
$ oc get nodemaintenance
```

2. Optional: Inspect the NodeMaintenance object to ensure it is associated with the correct node:

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

### Example output

```
Name: node02-maintenance
Namespace:
Labels:
Annotations:
API Version: nodemaintenance.kubevirt.io/v1beta1
Kind: NodeMaintenance
...
Spec:
  Node Name: node02
  Reason: Replacing node02
```

3. Delete the NodeMaintenance object:

```bash
$ oc delete nodemaintenance <node02-maintenance>
```
CHAPTER 11. NODE NETWORKING

11.1. OBSERVING NODE NETWORK STATE

Node network state is the network configuration for all nodes in the cluster.

11.1.1. About nmstate

OpenShift Virtualization uses nmstate to report on and configure the state of the node network. This makes it possible to modify network policy configuration, such as by creating a Linux bridge on all nodes, by applying a single configuration manifest to the cluster.

Node networking is monitored and updated by the following objects:

- **NodeNetworkState**: Reports the state of the network on that node.
- **NodeNetworkConfigurationPolicy**: Describes the requested network configuration on nodes. You update the node network configuration, including adding and removing interfaces, by applying a `NodeNetworkConfigurationPolicy` manifest to the cluster.
- **NodeNetworkConfigurationEnactment**: Reports the network policies enacted upon each node.

OpenShift Virtualization supports the use of the following nmstate interface types:

- Linux Bridge
- VLAN
- Bond
- Ethernet

NOTE

If your OpenShift Container Platform cluster uses OVN-Kubernetes as the default Container Network Interface (CNI) provider, you cannot attach a Linux bridge or bonding to the default interface of a host because of a change in the host network topology of OVN-Kubernetes. As a workaround, you can use a secondary network interface connected to your host, or switch to the OpenShift SDN default CNI provider.

11.1.2. Viewing the network state of a node

A `NodeNetworkState` object exists on every node in the cluster. This object is periodically updated and captures the state of the network for that node.

Procedure

1. List all the `NodeNetworkState` objects in the cluster:

   ```
   $ oc get nns
   ```
2. Inspect a **NodeNetworkState** object to view the network on that node. The output in this example has been redacted for clarity:

```bash
$ oc get nns node01 -o yaml
```

**Example output**

```
apiVersion: nmstate.io/v1alpha1
kind: NodeNetworkState
metadata:
  name: node01
status:
  currentState:
    dns-resolver:
    interfaces:
    route-rules:
    routes:
  lastSuccessfulUpdateTime: "2020-01-31T12:14:00Z"
```

1. The name of the **NodeNetworkState** object is taken from the node.
2. The **currentState** contains the complete network configuration for the node, including DNS, interfaces, and routes.
3. Timestamp of the last successful update. This is updated periodically as long as the node is reachable and can be used to evaluate the freshness of the report.

### 11.2. UPDATING NODE NETWORK CONFIGURATION

You can update the node network configuration, such as adding or removing interfaces from nodes, by applying **NodeNetworkConfigurationPolicy** manifests to the cluster.

#### 11.2.1. About nmstate

OpenShift Virtualization uses **nmstate** to report on and configure the state of the node network. This makes it possible to modify network policy configuration, such as by creating a Linux bridge on all nodes, by applying a single configuration manifest to the cluster.

Node networking is monitored and updated by the following objects:

**NodeNetworkState**

- Reports the state of the network on that node.

**NodeNetworkConfigurationPolicy**

- Describes the requested network configuration on nodes. You update the node network configuration, including adding and removing interfaces, by applying a **NodeNetworkConfigurationPolicy** manifest to the cluster.

**NodeNetworkConfigurationEnactment**
Reports the network policies enacted upon each node.

OpenShift Virtualization supports the use of the following nmstate interface types:

- Linux Bridge
- VLAN
- Bond
- Ethernet

**NOTE**

If your OpenShift Container Platform cluster uses OVN-Kubernetes as the default Container Network Interface (CNI) provider, you cannot attach a Linux bridge or bonding to the default interface of a host because of a change in the host network topology of OVN-Kubernetes. As a workaround, you can use a secondary network interface connected to your host, or switch to the OpenShift SDN default CNI provider.

### 11.2.2. Creating an interface on nodes

Create an interface on nodes in the cluster by applying a `NodeNetworkConfigurationPolicy` manifest to the cluster. The manifest details the requested configuration for the interface.

By default, the manifest applies to all nodes in the cluster. To add the interface to specific nodes, add the `spec: nodeSelector` parameter and the appropriate `<key>:<value>` for your node selector.

**Procedure**

1. Create the `NodeNetworkConfigurationPolicy` manifest. The following example configures a Linux bridge on all worker nodes:

```yaml
apiVersion: nmstate.io/v1alpha1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: <br1-eth1-policy>  # 1
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""  # 2
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with eth1 as a port  # 3
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: eth1
```

---

**CHAPTER 11. NODE NETWORKING**

223
1. Name of the policy.

2. Optional: If you do not include the `nodeSelector` parameter, the policy applies to all nodes in the cluster.

3. This example uses the `node-role.kubernetes.io/worker:""` node selector to select all worker nodes in the cluster.


2. Create the node network policy:

   ```bash
   $ oc apply -f <br1-eth1-policy.yaml>  
   1. File name of the node network configuration policy manifest.
   
Additional resources

- Example policy configurations for different interfaces
- Example for creating multiple interfaces in the same policy
- Examples of different IP management methods in policies

11.2.3. Confirming node network policy updates on nodes

A `NodeNetworkConfigurationPolicy` manifest describes your requested network configuration for nodes in the cluster. The node network policy includes your requested network configuration and the status of execution of the policy on the cluster as a whole.

When you apply a node network policy, a `NodeNetworkConfigurationEnactment` object is created for every node in the cluster. The node network configuration enactment is a read-only object that represents the status of execution of the policy on that node. If the policy fails to be applied on the node, the enactment for that node includes a traceback for troubleshooting.

Procedure

1. To confirm that a policy has been applied to the cluster, list the policies and their status:

   ```bash
   $ oc get nncp
   
   2. Optional: If a policy is taking longer than expected to successfully configure, you can inspect the requested state and status conditions of a particular policy:

   ```bash
   $ oc get nncp <policy> -o yaml
   
   3. Optional: If a policy is taking longer than expected to successfully configure on all nodes, you can list the status of the enactments on the cluster:

   ```bash
   $ oc get nnce
   
   4. Optional: To view the configuration of a particular enactment, including any error reporting for a failed configuration:
### 11.2.4. Removing an interface from nodes

You can remove an interface from one or more nodes in the cluster by editing the `NodeNetworkConfigurationPolicy` object and setting the `state` of the interface to `absent`.

Removing an interface from a node does not automatically restore the node network configuration to a previous state. If you want to restore the previous state, you will need to define that node network configuration in the policy.

If you remove a bridge or bonding interface, any node NICs in the cluster that were previously attached or subordinate to that bridge or bonding interface are placed in a `down` state and become unreachable. To avoid losing connectivity, configure the node NIC in the same policy so that it has a status of `up` and either DHCP or a static IP address.

**NOTE**

Deleting the node network policy that added an interface does not change the configuration of the policy on the node. Although a `NodeNetworkConfigurationPolicy` is an object in the cluster, it only represents the requested configuration. Similarly, removing an interface does not delete the policy.

### Procedure

1. Update the `NodeNetworkConfigurationPolicy` manifest used to create the interface. The following example removes a Linux bridge and configures the `eth1` NIC with DHCP to avoid losing connectivity:

   ```yaml
   $ oc get nnce <node>.<policy> -o yaml
   ```

   ```yaml
   apiVersion: nmstate.io/v1alpha1
   kind: NodeNetworkConfigurationPolicy
   metadata:
     name: <br1-eth1-policy>  
   spec:
     nodeSelector:
       node-role.kubernetes.io/worker: ""
     desiredState:
       interfaces:
       - name: br1
         type: linux-bridge
         state: absent
       - name: eth1
         type: ethernet
         state: up
       ipv4:
         dhcp: true
         enabled: true
   ```

   1. Name of the policy.
   2. **Optional:** If you do not include the `nodeSelector` parameter, the policy applies to all nodes in the cluster.
This example uses the node-role.kubernetes.io/worker: "" node selector to select all worker nodes in the cluster.

Changing the state to absent removes the interface.

The name of the interface that is to be unattached from the bridge interface.

The type of interface. This example creates an Ethernet networking interface.

The requested state for the interface.

Optional: If you do not use dhcp, you can either set a static IP or leave the interface without an IP address.

Enables ipv4 in this example.

2. Update the policy on the node and remove the interface:

   $ oc apply -f <br1-eth1-policy.yaml>  

File name of the policy manifest.

11.2.5. Example policy configurations for different interfaces

11.2.5.1. Example: Linux bridge interface node network configuration policy

Create a Linux bridge interface on nodes in the cluster by applying a NodeNetworkConfigurationPolicy manifest to the cluster.

The following YAML file is an example of a manifest for a Linux bridge interface. It includes samples values that you must replace with your own information.

```yaml
apiVersion: nmstate.io/v1alpha1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-policy
spec:
  nodeSelector:
    kubernetes.io/hostname: <node01>
desiredState:
  interfaces:
    - name: br1
      description: Linux bridge with eth1 as a port
      type: linux-bridge
      state: up
      ipv4:
        dhcp: true
        enabled: true
      bridge:
        options:
          stp:
```
Name of the policy.

Optional: If you do not include the `nodeSelector` parameter, the policy applies to all nodes in the cluster.

This example uses a `hostname` node selector.

Name of the interface.

Optional: Human-readable description of the interface.

The type of interface. This example creates a bridge.

The requested state for the interface after creation.

Optional: If you do not use `dhcp`, you can either set a static IP or leave the interface without an IP address.

Enables `ipv4` in this example.

Disables `stp` in this example.

The node NIC to which the bridge attaches.

### 11.2.5.2. Example: VLAN interface node network configuration policy

Create a VLAN interface on nodes in the cluster by applying a `NodeNetworkConfigurationPolicy` manifest to the cluster.

The following YAML file is an example of a manifest for a VLAN interface. It includes samples values that you must replace with your own information.

```yaml
apiVersion: nmstate.io/v1alpha1
class: NodeNetworkConfigurationPolicy
metadata:
  name: vlan-eth1-policy
spec:
  nodeSelector:
    kubernetes.io/hostname: <node01>
  desiredState:
    interfaces:
      - name: eth1.102
        description: VLAN using eth1
        type: vlan
        state: up
        vlan:
          base-iface: eth1
          id: 102
```
Name of the policy.

Optional: If you do not include the `nodeSelector` parameter, the policy applies to all nodes in the cluster.

This example uses a `hostname` node selector.

Name of the interface.

Optional: Human-readable description of the interface.

The type of interface. This example creates a VLAN.

The requested state for the interface after creation.

The node NIC to which the VLAN is attached.

The VLAN tag.

11.2.5.3. Example: Bond interface node network configuration policy

Create a bond interface on nodes in the cluster by applying a `NodeNetworkConfigurationPolicy` manifest to the cluster.

**NOTE**

OpenShift Virtualization only supports the following bond modes:

- `mode=1` active-backup
- `mode=2` balance-xor
- `mode=4` 802.3ad
- `mode=5` balance-tlb
- `mode=6` balance-alb

The following YAML file is an example of a manifest for a bond interface. It includes samples values that you must replace with your own information.

```yaml
apiVersion: nmstate.io/v1alpha1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: bond0-eth1-eth2-policy
spec:
  nodeSelector:
    kubernetes.io/hostname: <node01>
desiredState:
  interfaces:
  - name: bond0
    description: Bond enslaving eth1 and eth2
    type: bond
    state: up
```
### Name of the policy.

1. Optional: If you do not include the `nodeName` parameter, the policy applies to all nodes in the cluster.

2. This example uses a `hostname` node selector.

3. Name of the interface.


5. The type of interface. This example creates a bond.

6. The requested state for the interface after creation.

7. Optional: If you do not use `dhcp`, you can either set a static IP or leave the interface without an IP address.

8. Enables `ipv4` in this example.

9. The driver mode for the bond. This example uses an active backup mode.

10. Optional: This example uses miimon to inspect the bond link every 140ms.

11. The subordinate node NICs in the bond.

12. Optional: The maximum transmission unit (MTU) for the bond. If not specified, this value is set to 1500 by default.

#### 11.2.5.4. Example: Ethernet interface node network configuration policy

Configure an Ethernet interface on nodes in the cluster by applying a `NodeNetworkConfigurationPolicy` manifest to the cluster.

The following YAML file is an example of a manifest for an Ethernet interface. It includes sample values that you must replace with your own information.

```yaml
apiVersion: nmstate.io/v1alpha1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: eth1-policy
spec:
```
nodeSelector: 

kubernetes.io/hostname: <node01>

desiredState:

interfaces:

- name: eth1
  description: Configuring eth1 on node01
  type: ethernet
  state: up
  ipv4:
    dhcp: true
    enabled: true

Name of the policy.

Optional: If you do not include the nodeSelector parameter, the policy applies to all nodes in the cluster.

This example uses a hostname node selector.

Name of the interface.

Optional: Human-readable description of the interface.

The type of interface. This example creates an Ethernet networking interface.

The requested state for the interface after creation.

Optional: If you do not use dhcp, you can either set a static IP or leave the interface without an IP address.

Enables ipv4 in this example.

11.2.5.5. Example: Multiple interfaces in the same node network configuration policy

You can create multiple interfaces in the same node network configuration policy. These interfaces can reference each other, allowing you to build and deploy a network configuration by using a single policy manifest.

The following example snippet creates a bond that is named bond10 across two NICs and a Linux bridge that is named br1 that connects to the bond.

... 

interfaces:

- name: bond10
  description: Bonding eth2 and eth3 for Linux bridge
  type: bond
  state: up
  link-aggregation:
    slaves:
      - eth2
      - eth3
- name: br1
  description: Linux bridge on bond
  type: linux-bridge
11.2.6. Examples: IP management

The following example configuration snippets demonstrate different methods of IP management.

These examples use the `ethernet` interface type to simplify the example while showing the related context in the policy configuration. These IP management examples can be used with the other interface types.

11.2.6.1. Static

The following snippet statically configures an IP address on the Ethernet interface:

```yaml
... interfaces:
- name: eth1
  description: static IP on eth1
  type: ethernet
  state: up
  ipv4:
    address:
      - ip: 192.168.122.250 1
        prefix-length: 24
        enabled: true
...
```

1  Replace this value with the static IP address for the interface.

11.2.6.2. No IP address

The following snippet ensures that the interface has no IP address:

```yaml
... interfaces:
- name: eth1
  description: No IP on eth1
  type: ethernet
  state: up
  ipv4:
    enabled: false
... 
```

11.2.6.3. Dynamic host configuration

The following snippet configures an Ethernet interface that uses a dynamic IP address, gateway address, and DNS:
The following snippet configures an Ethernet interface that uses a dynamic IP address but does not use a dynamic gateway address or DNS:

```yaml
... interfaces:
  - name: eth1
description: DHCP on eth1
type: ethernet
state: up
ipv4:
dhcp: true
enabled: true
...
```

The following snippet sets DNS configuration on the host.

```yaml
... interfaces:
... dns-resolver:
config:
search:
  - example.com
  - example.org
server:
  - 8.8.8.8
...
```

### 11.2.6.4. DNS

The following snippet sets DNS configuration on the host.

### 11.2.6.5. Static routing

The following snippet configures a static route and a static IP on interface `eth1`.

```yaml
... interfaces:
  - name: eth1
description: Static routing on eth1
type: ethernet
state: up
ipv4:
```
11.3. TROUBLESHOOTING NODE NETWORK CONFIGURATION

If the node network configuration encounters an issue, the policy is automatically rolled back and the enactments report failure. This includes issues such as:

- The configuration fails to be applied on the host.
- The host loses connection to the default gateway.
- The host loses connection to the API server.

11.3.1. Troubleshooting an incorrect node network configuration policy configuration

You can apply changes to the node network configuration across your entire cluster by applying a node network configuration policy. If you apply an incorrect configuration, you can use the following example to troubleshoot and correct the failed node network policy.

In this example, a Linux bridge policy is applied to an example cluster that has 3 master nodes and 3 worker nodes. The policy fails to be applied because it references an incorrect interface. To find the error, investigate the available nmstate resources. You can then update the policy with the correct configuration.

Procedure

1. Create a policy and apply it to your cluster. The following example creates a simple bridge on the `ens01` interface:

```yaml
apiVersion: nmstate.io/v1alpha1
description: Linux bridge with the wrong port
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ens01-bridge-testfail
spec:
desiredState:
  interfaces:
    - name: br1
      enabled: true
  routes:
    config:
      destination: 198.51.100.0/24
      metric: 150
      next-hop-address: 192.0.2.1
      next-hop-interface: eth1
      table-id: 254

# Example:

- ip: 192.0.2.251
  prefix-length: 24
  enabled: true
```

1. The static IP address for the Ethernet interface.
2. Next hop address for the node traffic. This must be in the same subnet as the IP address set for the Ethernet interface.
Verify the status of the policy by running the following command:

```
$ oc get nncp
```

The output shows that the policy failed:

```
NAME                    STATUS
ens01-bridge-testfail   FailedToConfigure
```

However, the policy status alone does not indicate if it failed on all nodes or a subset of nodes.

List the node network configuration enactments to see if the policy was successful on any of the nodes. If the policy failed for only a subset of nodes, it suggests that the problem is with a specific node configuration. If the policy failed on all nodes, it suggests that the problem is with the policy.

```
$ oc get nnce
```

The output shows that the policy failed on all nodes:

```
NAME                                   STATUS
master-1.ens01-bridge-testfail         FailedToConfigure
master-2.ens01-bridge-testfail         FailedToConfigure
master-3.ens01-bridge-testfail         FailedToConfigure
worker-1.ens01-bridge-testfail         FailedToConfigure
worker-2.ens01-bridge-testfail         FailedToConfigure
worker-3.ens01-bridge-testfail         FailedToConfigure
```

View one of the failed enactments and look at the traceback. The following command uses the output tool `jsonpath` to filter the output:
This command returns a large traceback that has been edited for brevity:

**Example output**

```
error reconciling NodeNetworkConfigurationPolicy at desired state apply: , failed to execute
nmstatectl set --no-commit --timeout 480: 'exit status 1' "
...
libnmstate.error.NmstateVerificationError:
desired
=======
---
name: br1
type: linux-bridge
state: up
bridge:
  options:
    group-forward-mask: 0
    mac-ageing-time: 300
    multicast-snooping: true
  stp:
    enabled: false
    forward-delay: 15
    hello-time: 2
    max-age: 20
    priority: 32768
port:
  name: ens01
description: Linux bridge with the wrong port
ipv4:
  address: []
  auto-dns: true
  auto-gateway: true
  auto-routes: true
dhcp: true
  enabled: true
ipv6:
  enabled: false
mac-address: 01-23-45-67-89-AB
mtu: 1500

current
=======
---
name: br1
type: linux-bridge
state: up
bridge:
  options:
    group-forward-mask: 0
    mac-ageing-time: 300
    multicast-snooping: true
  stp:
```

---

The `NmstateVerificationError` lists the desired policy configuration, the current configuration of the policy on the node, and the difference highlighting the parameters that do not match. In this example, the port is included in the difference, which suggests that the problem is the port configuration in the policy.

```
enabled: false
forward-delay: 15
hello-time: 2
max-age: 20
priority: 32768
port: []
description: Linux bridge with the wrong port
ipv4:
  address: []
  auto-dns: true
  auto-gateway: true
  auto-routes: true
  dhcp: true
  enabled: true
ipv6:
  enabled: false
mac-address: 01-23-45-67-89-AB
mtu: 1500
```

```
--- desired
+++ current
@@ -13,8 +13,7 @@
    hello-time: 2
    max-age: 20
    priority: 32768
-  port:
-    name: ens01
+  port: []
description: Linux bridge with the wrong port
ipv4:
  address: []
```

The `NmstateVerificationError` lists the desired policy configuration, the current configuration of the policy on the node, and the difference highlighting the parameters that do not match. In this example, the `port` is included in the difference, which suggests that the problem is the port configuration in the policy.

5. To ensure that the policy is configured properly, view the network configuration for one or all of the nodes by requesting the `NodeNetworkState` object. The following command returns the network configuration for the `master-1` node:

```
$ oc get nns master-1 -o yaml
```

The output shows that the interface name on the nodes is `ens1` but the failed policy incorrectly uses `ens01`:

Example output

```
  - ipv4:
    ...
    name: ens1
```
6. Correct the error by editing the existing policy:

```
$ oc edit nncp ens01-bridge-testfail
...
  port:
    - name: ens1
```

Save the policy to apply the correction.

7. Check the status of the policy to ensure it updated successfully:

```
$ oc get nncp
```

**Example output**

```
NAME                    STATUS
ens01-bridge-testfail   SuccessfullyConfigured
```

The updated policy is successfully configured on all nodes in the cluster.
CHAPTER 12. LOGGING, EVENTS, AND MONITORING

12.1. VIEWING VIRTUAL MACHINE LOGS

12.1.1. Understanding virtual machine logs

Logs are collected for OpenShift Container Platform builds, deployments, and pods. In OpenShift 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](image)

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

12.1.2. Viewing virtual machine logs in the CLI

Get virtual machine logs from the virtual machine launcher pod.

**Procedure**

- Use the following command:

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

12.1.3. Viewing virtual machine logs in the web console

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

**Procedure**

1. In the OpenShift Virtualization console, click **Workloads → Virtualization** from the side menu.
2. Click the **Virtual Machines** tab.
3. Select a virtual machine to open the **Virtual Machine Overview** screen.
4. In the **Details** tab, click the **virt-launcher-<name>** pod in the **Pod** section.
5. Click **Logs**.

12.2. VIEWING EVENTS
12.2.1. Understanding virtual machine 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.

OpenShift 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.

12.2.2. 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 Overview panel of the web console.

The ▪ button pauses the events stream.
The ▶ button continues a paused events stream.

Procedure

1. Click Workloads → Virtualization from the side menu.
2. Click the Virtual Machines tab.
3. Select a virtual machine to open the Virtual Machine Overview screen.
4. Click Events to view all events for the virtual machine.

12.2.3. 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

12.2.4. 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>
12.3. DIAGNOSING DATA VOLUMES USING EVENTS AND CONDITIONS

Use the `oc describe` command to analyze and help resolve issues with data volumes.

12.3.1. About conditions and events

Diagnose data volume issues by examining the output of the `Conditions` and `Events` sections generated by the command:

```
$ oc describe dv <DataVolume>
```

There are three **Types** in the `Conditions` section that display:

- **Bound**
- **Running**
- **Ready**

The `Events` section provides the following additional information:

- **Type** of event
- **Reason** for logging
- **Source** of the event
- **Message** containing additional diagnostic information.

The output from `oc describe` does not always contain `Events`.

An event is generated when either **Status**, **Reason**, or **Message** changes. Both conditions and events react to changes in the state of the data volume.

For example, if you misspell the URL during an import operation, the import generates a 404 message. That message change generates an event with a reason. The output in the `Conditions` section is updated as well.

12.3.2. Analyzing data volumes using conditions and events

By inspecting the `Conditions` and `Events` sections generated by the `describe` command, you determine the state of the data volume in relation to persistent volume claims (PVCs), and whether or not an operation is actively running or completed. You might also receive messages that offer specific details about the status of the data volume, and how it came to be in its current state.

There are many different combinations of conditions. Each must be evaluated in its unique context.

Examples of various combinations follow.

- **Bound** – A successfully bound PVC displays in this example. Note that the **Type** is `Bound`, so the **Status** is `True`. If the PVC is not bound, the **Status** is `False`.
When the PVC is bound, an event is generated stating that the PVC is bound. In this case, the **Reason** is *Bound* and **Status** is *True*. The **Message** indicates which PVC owns the data volume.

**Message**, in the **Events** section, provides further details including how long the PVC has been bound (**Age**) and by what resource (**From**), in this case *datavolume-controller*.

### Example output

<table>
<thead>
<tr>
<th>Status:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions:</td>
</tr>
<tr>
<td>Last Heart Beat Time: 2020-07-15T03:58:24Z</td>
</tr>
<tr>
<td>Last Transition Time: 2020-07-15T03:58:24Z</td>
</tr>
<tr>
<td>Message: PVC win10-rootdisk Bound</td>
</tr>
<tr>
<td>Reason: Bound</td>
</tr>
<tr>
<td>Status: True</td>
</tr>
<tr>
<td>Type: Bound</td>
</tr>
</tbody>
</table>

**Events:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Reason</th>
<th>Age</th>
<th>From</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>------</td>
<td>-------</td>
<td>----------------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Bound</td>
<td>24s</td>
<td>datavolume-controller PVC example-dv Bound</td>
</tr>
</tbody>
</table>

- **Running** – In this case, note that **Type** is *Running* and **Status** is *False*, indicating that an event has occurred that caused an attempted operation to fail, changing the Status from *True* to *False*.

  However, note that **Reason** is *Completed* and the **Message** field indicates *Import Complete*.

In the **Events** section, the **Reason** and **Message** contain additional troubleshooting information about the failed operation. In this example, the **Message** displays an inability to connect due to a **404**, listed in the **Events** section’s first **Warning**.

From this information, you conclude that an import operation was running, creating contention for other operations that are attempting to access the data volume:

### Example output

<table>
<thead>
<tr>
<th>Status:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions:</td>
</tr>
<tr>
<td>Message: Import Complete</td>
</tr>
<tr>
<td>Reason: Completed</td>
</tr>
<tr>
<td>Status: False</td>
</tr>
<tr>
<td>Type: Running</td>
</tr>
</tbody>
</table>

**Events:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Reason</th>
<th>Age</th>
<th>From</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning</td>
<td>Error</td>
<td>12s (x2 over 14s)</td>
<td>datavolume-controller</td>
<td>Unable to connect to http data source: expected status code 200, got 404. Status: 404 Not Found</td>
</tr>
</tbody>
</table>

- **Ready** – If **Type** is *Ready* and **Status** is *True*, then the data volume is ready to be used, as in the following example. If the data volume is not ready to be used, the **Status** is *False*:

### Example output

<table>
<thead>
<tr>
<th>Status:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions:</td>
</tr>
<tr>
<td>Message: PVC example-dv Bound</td>
</tr>
<tr>
<td>Reason: Bound</td>
</tr>
<tr>
<td>Status: True</td>
</tr>
<tr>
<td>Type: Bound</td>
</tr>
</tbody>
</table>

**Events:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Reason</th>
<th>Age</th>
<th>From</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>------</td>
<td>-------</td>
<td>----------------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Bound</td>
<td>24s</td>
<td>datavolume-controller PVC example-dv Bound</td>
</tr>
</tbody>
</table>
12.4. VIEWING INFORMATION ABOUT VIRTUAL MACHINE WORKLOADS

You can view high-level information about your virtual machines by using the Virtual Machines dashboard in the OpenShift Container Platform web console.

12.4.1. About the Virtual Machines dashboard

Access virtual machines from the OpenShift Container Platform web console by navigating to the Workloads → Virtualization page. The Workloads → Virtualization page contains two tabs: * Virtual Machines * Virtual Machine Templates

The following cards describe each virtual machine:

- **Details** provides identifying information about the virtual machine, including:
  - Name
  - Namespace
  - Date of creation
  - Node name
  - IP address

- **Inventory** lists the virtual machine’s resources, including:
  - Network interface controllers (NICs)
  - Disks

- **Status** includes:
  - The current status of the virtual machine
  - A note indicating whether or not the QEMU guest agent is installed on the virtual machine

- **Utilization** includes charts that display usage data for:
  - CPU
  - Memory
  - Filesystem
  - Network transfer
NOTE

Use the drop-down list to choose a duration for the utilization data. The available options are 1 Hour, 6 Hours, and 24 Hours.

- Events lists messages about virtual machine activity over the past hour. To view additional events, click View all.

12.5. MONITORING VIRTUAL MACHINE HEALTH

A virtual machine instance (VMI) can become unhealthy due to transient issues such as connectivity loss, deadlocks, or problems with external dependencies. A health check periodically performs diagnostics on a VMI by using any combination of the readiness and liveness probes.

12.5.1. About readiness and liveness probes

Use readiness and liveness probes to detect and handle unhealthy virtual machine instances (VMIs). You can include one or more probes in the specification of the VMI to ensure that traffic does not reach a VMI that is not ready for it and that a new instance is created when a VMI becomes unresponsive.

A readiness probe determines whether a VMI is ready to accept service requests. If the probe fails, the VMI is removed from the list of available endpoints until the VMI is ready.

A liveness probe determines whether a VMI is responsive. If the probe fails, the VMI is deleted and a new instance is created to restore responsiveness.

You can configure readiness and liveness probes by setting the spec.readinessProbe and the spec.livenessProbe fields of the VirtualMachineInstance object. These fields support the following tests:

HTTP GET

The probe determines the health of the VMI by using a web hook. The test is successful if the HTTP response code is between 200 and 399. You can use an HTTP GET test with applications that return HTTP status codes when they are completely initialized.

TCP socket

The probe attempts to open a socket to the VMI. The VMI is only considered healthy if the probe can establish a connection. You can use a TCP socket test with applications that do not start listening until initialization is complete.

12.5.2. Defining an HTTP readiness probe

Define an HTTP readiness probe by setting the spec.readinessProbe.httpGet field of the virtual machine instance (VMI) configuration.

Procedure

1. Include details of the readiness probe in the VMI configuration file.

Sample readiness probe with an HTTP GET test

```yaml
# ...
spec:
  readinessProbe:
```
The HTTP GET request to perform to connect to the VMI.

The port of the VMI that the probe queries. In the above example, the probe queries port 1500.

The path to access on the HTTP server. In the above example, if the handler for the server’s /healthz path returns a success code, the VMI is considered to be healthy. If the handler returns a failure code, the VMI is removed from the list of available endpoints.

The time, in seconds, after the VMI starts before the readiness probe is initiated.

The delay, in seconds, between performing probes. The default delay is 10 seconds. This value must be greater than timeoutSeconds.

The number of seconds of inactivity after which the probe times out and the VMI is assumed to have failed. The default value is 1. This value must be lower than periodSeconds.

The number of times that the probe is allowed to fail. The default is 3. After the specified number of attempts, the pod is marked Unready.

The number of times that the probe must report success, after a failure, to be considered successful. The default is 1.

2. Create the VMI by running the following command:

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

12.5.3. Defining a TCP readiness probe

Define a TCP readiness probe by setting the spec.readinessProbe.tcpSocket field of the virtual machine instance (VMI) configuration.

Procedure

1. Include details of the TCP readiness probe in the VMI configuration file.

Sample readiness probe with a TCP socket test

```yaml
# ...
The time, in seconds, after the VMI starts before the readiness probe is initiated.

The delay, in seconds, between performing probes. The default delay is 10 seconds. This value must be greater than `timeoutSeconds`.

The TCP action to perform.

The port of the VMI that the probe queries.

The number of seconds of inactivity after which the probe times out and the VMI is assumed to have failed. The default value is 1. This value must be lower than `periodSeconds`.

2. Create the VMI by running the following command:

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

### 12.5.4. Defining an HTTP liveness probe

Define an HTTP liveness probe by setting the `spec.livenessProbe.httpGet` field of the virtual machine instance (VMI) configuration. You can define both HTTP and TCP tests for liveness probes in the same way as readiness probes. This procedure configures a sample liveness probe with an HTTP GET test.

Procedure

1. Include details of the HTTP liveness probe in the VMI configuration file.

**Sample liveness probe with an HTTP GET test**

```
# ...
spec:
livenessProbe:
  initialDelaySeconds: 120   # 1
  periodSeconds: 20         # 2
  httpGet:                  # 3
    port: 1500              # 4
    path: /healthz          # 5
    httpHeaders:
      - name: Custom-Header
        value: Awesome
  timeoutSeconds: 10       # 6
# ...
```
The time, in seconds, after the VMI starts before the liveness probe is initiated.

2. The delay, in seconds, between performing probes. The default delay is 10 seconds. This value must be greater than `timeoutSeconds`.

3. The HTTP GET request to perform to connect to the VMI.

4. The port of the VMI that the probe queries. In the above example, the probe queries port 1500. The VMI installs and runs a minimal HTTP server on port 1500 via cloud-init.

5. The path to access on the HTTP server. In the above example, if the handler for the server’s `/healthz` path returns a success code, the VMI is considered to be healthy. If the handler returns a failure code, the VMI is deleted and a new instance is created.

6. The number of seconds of inactivity after which the probe times out and the VMI is assumed to have failed. The default value is 1. This value must be lower than `periodSeconds`.

2. Create the VMI by running the following command:

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

### 12.5.5. Template: Virtual machine instance configuration file for defining health checks

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  labels:
    special: vmi-fedora
  name: vmi-fedora
spec:
  domain:
    devices:
      disks:
      - disk:
          bus: virtio
          name: containerdisk
      - disk:
          bus: virtio
          name: cloudinitdisk
  resources:
    requests:
      memory: 1024M
  readinessProbe:
    httpGet:
      port: 1500
      initialDelaySeconds: 120
      periodSeconds: 20
      timeoutSeconds: 10
      failureThreshold: 3
      successThreshold: 3
  terminationGracePeriodSeconds: 0
volumes:
```

OpenShift Container Platform 4.6 OpenShift Virtualization
12.6. USING THE OPENSHIFT CONTAINER PLATFORM DASHBOARD TO GET CLUSTER INFORMATION

Access the OpenShift Container Platform dashboard, which captures high-level information about the cluster, by clicking Home > Dashboards > Overview from the OpenShift Container Platform web console.

The OpenShift Container Platform dashboard provides various cluster information, captured in individual dashboard cards.

12.6.1. About the OpenShift Container Platform dashboards page

The OpenShift Container Platform dashboard consists of the following cards:

- **Details** provides a brief overview of informational cluster details. Status include ok, error, warning, in progress, and unknown. Resources can add custom status names.
  - Cluster ID
  - Provider
  - Version

- **Cluster Inventory** details number of resources and associated statuses. It is helpful when intervention is required to resolve problems, including information about:
  - Number of nodes
  - Number of pods
  - Persistent storage volume claims
  - Virtual machines (available if OpenShift Virtualization is installed)
Bare metal hosts in the cluster, listed according to their state (only available in metal3 environment).

- Cluster Health summarizes the current health of the cluster as a whole, including relevant alerts and descriptions. If OpenShift Virtualization is installed, the overall health of OpenShift Virtualization is diagnosed as well. If more than one subsystem is present, click See All to view the status of each subsystem.

- Cluster Capacity charts help administrators understand when additional resources are required in the cluster. The charts contain an inner ring that displays current consumption, while an outer ring displays thresholds configured for the resource, including information about:
  - CPU time
  - Memory allocation
  - Storage consumed
  - Network resources consumed

- Cluster Utilization shows the capacity of various resources over a specified period of time, to help administrators understand the scale and frequency of high resource consumption.

- Events lists messages related to recent activity in the cluster, such as pod creation or virtual machine migration to another host.

- Top Consumers helps administrators understand how cluster resources are consumed. Click on a resource to jump to a detailed page listing pods and nodes that consume the largest amount of the specified cluster resource (CPU, memory, or storage).

### 12.7. OPENSIFT CONTAINER PLATFORM CLUSTER MONITORING, LOGGING, AND TELEMETRY

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

#### 12.7.1. About OpenShift Container Platform monitoring

OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that provides monitoring for core platform components. OpenShift Container Platform delivers monitoring best practices out of the box. A set of alerts are included by default that immediately notify cluster administrators about issues with a cluster. Default dashboards in the OpenShift Container Platform web console include visual representations of cluster metrics to help you to quickly understand the state of your cluster.

After installing OpenShift Container Platform 4.6, cluster administrators can optionally enable monitoring for user-defined projects. By using this feature, cluster administrators, developers, and other users can specify how services and pods are monitored in their own projects. You can then query metrics, review dashboards, and manage alerting rules and silences for your own projects in the OpenShift Container Platform web console.

**NOTE**

Cluster administrators can grant developers and other users permission to monitor their own projects. Privileges are granted by assigning one of the predefined monitoring roles.
12.7.2. About cluster logging components

The cluster logging components include a collector deployed to each node in the OpenShift Container Platform cluster that collects all node and container logs and writes them to a log store. You can use a centralized web UI to create rich visualizations and dashboards with the aggregated data.

The major components of cluster logging are:

- **collection** - This is the component that collects logs from the cluster, formats them, and forwards them to the log store. The current implementation is Fluentd.

- **log store** - This is where the logs are stored. The default implementation is Elasticsearch. You can use the default Elasticsearch log store or forward logs to external log stores. The default log store is optimized and tested for short-term storage.

- **visualization** - This is the UI component you can use to view logs, graphs, charts, and so forth. The current implementation is Kibana.

For more information on cluster logging, see the OpenShift Container Platform cluster logging documentation.

12.7.3. About Telemetry

Telemetry sends a carefully chosen subset of the cluster monitoring metrics to Red Hat. The Telemeter Client fetches the metrics values every four minutes and thirty seconds and uploads the data to Red Hat. These metrics are described in this document.

This stream of data is used by Red Hat to monitor the clusters in real-time and to react as necessary to problems that impact our customers. It also allows Red Hat to roll out OpenShift Container Platform upgrades to customers to minimize service impact and continuously improve the upgrade experience.

This debugging information is available to Red Hat Support and Engineering teams with the same restrictions as accessing data reported through support cases. All connected cluster information is used by Red Hat to help make OpenShift Container Platform better and more intuitive to use.

12.7.3.1. Information collected by Telemetry

The following information is collected by Telemetry:

- The unique random identifier that is generated during an installation

- Version information, including the OpenShift Container Platform cluster version and installed update details that are used to determine update version availability

- Update information, including the number of updates available per cluster, the channel and image repository used for an update, update progress information, and the number of errors that occur in an update

- The name of the provider platform that OpenShift Container Platform is deployed on and the data center location

- Sizing information about clusters, machine types, and machines, including the number of CPU cores and the amount of RAM used for each

- The number of running virtual machine instances in a cluster

- The number of etcd members and the number of objects stored in the etcd cluster
The OpenShift Container Platform framework components installed in a cluster and their condition and status

- Usage information about components, features, and extensions
- Usage details about Technology Previews and unsupported configurations
- Information about degraded software
- Information about nodes that are marked as NotReady
- Events for all namespaces listed as "related objects" for a degraded Operator
- Configuration details that help Red Hat Support to provide beneficial support for customers. This includes node configuration at the cloud infrastructure level, hostnames, IP addresses, Kubernetes pod names, namespaces, and services.

- Information about the validity of certificates

Telemetry does not collect identifying information such as user names, or passwords. Red Hat does not intend to collect personal information. If Red Hat discovers that personal information has been inadvertently received, Red Hat will delete such information. To the extent that any telemetry data constitutes personal data, please refer to the Red Hat Privacy Statement for more information about Red Hat’s privacy practices.

### 12.7.4. CLI troubleshooting and debugging commands

For a list of the `oc` client troubleshooting and debugging commands, see the OpenShift Container Platform CLI tools documentation.

### 12.8. COLLECTING OPENSHEET VIRTUALIZATION DATA FOR RED HAT SUPPORT

When opening a support case, it is helpful to provide debugging information about your cluster to Red Hat Support.

The `must-gather` tool enables you to collect diagnostic information about your OpenShift Container Platform cluster, including virtual machines and other data related to OpenShift Virtualization.

For prompt support, supply diagnostic information for both OpenShift Container Platform and OpenShift Virtualization.

### 12.8.1. About the must-gather tool

The `oc adm must-gather` CLI command collects the information from your cluster that is most likely needed for debugging issues, including:

- Resource definitions
- Service logs

By default, the `oc adm must-gather` command uses the default plug-in image and writes into `.must-gather.local`. 
Alternatively, you can collect specific information by running the command with the appropriate arguments as described in the following sections:

- To collect data related to one or more specific features, use the `--image` argument with an image, as listed in a following section.
  For example:

  ```
  $ oc adm must-gather --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.0
  ```

- To collect the audit logs, use the `-- /usr/bin/gather_audit_logs` argument, as described in a following section.
  For example:

  ```
  $ oc adm must-gather -- /usr/bin/gather_audit_logs
  ```

**NOTE**

Audit logs are not collected as part of the default set of information to reduce the size of the files.

When you run `oc adm must-gather`, a new pod with a random name is created in a new project on the cluster. The data is collected on that pod and saved in a new directory that starts with `must-gather.local`. This directory is created in the current working directory.

For example:

```

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>openshift-must-gather-5drcj</td>
<td>must-gather-bklx4</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>72s</td>
</tr>
<tr>
<td>openshift-must-gather-5drcj</td>
<td>must-gather-s8sdsx</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>72s</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

### 12.8.2. About collecting OpenShift Virtualization data

You can use the `oc adm must-gather` CLI command to collect information about your cluster, including features and objects associated with OpenShift Virtualization:

- The Hyperconverged Cluster Operator namespaces (and child objects)
- All namespaces (and their child objects) that belong to any OpenShift Virtualization resources
- All OpenShift Virtualization custom resource definitions (CRDs)
- All namespaces that contain virtual machines
- All virtual machine definitions

To collect OpenShift Virtualization data with `must-gather`, you must specify the OpenShift Virtualization image: `--image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.5.8`.

### 12.8.3. Gathering data about specific features
You can gather debugging information about specific features by using the `oc adm must-gather` CLI command with the `--image` or `--image-stream` argument. The must-gather tool supports multiple images, so you can gather data about more than one feature by running a single command.

**NOTE**

To collect the default must-gather data in addition to specific feature data, add the `--image-stream=openshift/must-gather` argument.

**Prerequisites**

- Access to the cluster as a user with the `cluster-admin` role.
- The OpenShift Container Platform CLI (`oc`) installed.

**Procedure**

1. Navigate to the directory where you want to store the must-gather data.

2. Run the `oc adm must-gather` command with one or more `--image` or `--image-stream` arguments. For example, the following command gathers both the default cluster data and information specific to OpenShift Virtualization:

   ```bash
   $ oc adm must-gather \
   --image-stream=openshift/must-gather \
   --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.5.8
   ```

   **1** The default OpenShift Container Platform must-gather image

   **2** The must-gather image for OpenShift Virtualization

   You can use the must-gather tool with additional arguments to gather data that is specifically related to cluster logging and the Cluster Logging Operator in your cluster. For cluster logging, run the following command:

   ```bash
   $ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator \ 
   -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
   ```

   **Example 12.1. Example must-gather output for cluster logging**

   ```
   ├── cluster-logging
   │   ├── clo
   │   │   └── cluster-logging-operator-74dd5994f-6ttgt
   │   │       └── clusterlogforwarder_cr
   │   │           └── cr
   │   │               └── csv
   │   │                   └── deployment
   │   │                       └── logforwarding_cr
   │   │                           └── collector
   │   │                               └── fluentd-2tr64
   │   │                                   └── curator
   ```
CHAPTER 12. LOGGING, EVENTS, AND MONITORING

```
curator-1596028500-zkz4s
  eo
  csv
  deployment
  elasticsearch-operator-7dc7d97b9d-jb4r4
  es
    cluster-elasticsearch
      aliases
      health
      indices
      latest_documents.json
      nodes
      nodes_stats.json
      thread_pool
      cr
      elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
      logs
      elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
      install
      co_logs
      install_plan
      olmo_logs
      subscription
  kibana
    cr
    kibana-9d69668d4-2rkvz
  cluster-scoped-resources
    core
      nodes
      ip-10-0-146-180.eu-west-1.compute.internal.yaml
      persistentvolumes
      pvc-0a8d65d9-54aa-4c44-9ecc-33d9381e41c1.yaml
    event-filter.html
    gather-debug.log
  namespaces
    openshift-logging
      apps
        daemonsets.yaml
        deployments.yaml
        replicaset.yaml
        statefulset.yaml
      batch
        cronjobs.yaml
        jobs.yaml
      core
        configmaps.yaml
        endpoints.yaml
      events
        curator-1596021300-wn2ks.1626344ef0055a94.yaml
        curator.162638330681bee2.yaml
        elasticsearch-im-app-159602400-gm6nl.1626341a296c16a1.yaml
        elasticsearch-im-audit-1596020400-99n94.1626341a2af81b3d.yaml
        elasticsearch-im-infra-1596020400-v98tk.1626341a2d821069.yaml
        elasticsearch-im-app-1596020400-cc5vc.1626341a3019b2b8.yaml
        elasticsearch-im-audit-1596020400-s8d5f.1626341a31f7b315.yaml
        elasticsearch-im-infra-1596020400-7mgv8.1626341a35ea59ed.yaml
```
3. Create a compressed file from the **must-gather** directory that was just created in your working directory. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/
```

1. Make sure to replace **must-gather-local.5421342344627712289/** with the actual directory name.

4. Attach the compressed file to your support case on the [Red Hat Customer Portal](https://example.com)