OpenShift Container Platform 4.8

OpenShift Virtualization

OpenShift Virtualization 安装、使用和发行注记

Last Updated: 2023-04-19

OpenShift Container Platform 4.8 OpenShift Virtualization

OpenShift Virtualization 安装、使用和发行注记

法律通告

摘要

本文档提供有关如何在 OpenShift Container Platform 中使用 OpenShift Virtualization 的信息。

目录

第 1 章 关于 OPENSHIFT VIRTUALIZATION **4**
1.1. OPENSHIFT VIRTUALIZATION 的作用 4

第 2 章 从 OPENSHIFT VIRTUALIZATION 开始 **5**
2.1. 集群管理员 5
2.2. 虚拟化管理员 5
2.3. 虚拟机管理员/开发人员 5

第 3 章 OPENSHIFT VIRTUALIZATION 发行注记 **7**
3.1. 关于 RED HAT OPENSHIFT VIRTUALIZATION 7
3.2. 使开源包含更多 7
3.3. 新增和改变的功能 7
3.4. 弃用和删除的功能 8
3.5. 主要的技术变化 9
3.6. 技术预览功能 9
3.7. 已知问题 9

第 4 章 安装 OPENSHIFT VIRTUALIZATION **13**
4.1. 为 OPENSHIFT VIRTUALIZATION 准备集群 13
4.2. 为 OPENSHIFT VIRTUALIZATION 组件指定节点 16
4.3. 使用 WEB 控制台卸载 OPENSHIFT VIRTUALIZATION 22
4.4. 安装 VIRTCTL 客户端 25
4.5. 使用 WEB 控制台卸载 OPENSHIFT VIRTUALIZATION 27
4.6. 使用 CLI 卸载 OPENSHIFT VIRTUALIZATION 28

第 5 章 更新 OPENSHIFT VIRTUALIZATION **30**
5.1. 关于升级 OPENSHIFT VIRTUALIZATION 30
5.2. 手动批准待处理的 OPERATOR 更新 31
5.3. 监控升级状态 31
5.4. 其他资源 32

第 6 章 为 KUBEVIRT-CONTROLLER 和 VIRT-LAUNCHER 授予额外的安全权限 **33**
6.1. 为 VIRT-LAUNCHER POD 扩展 SELINUX 策略 33
6.2. KUBEVIRT-CONTROLLER 服务帐户的其他 OPENSHIFT CONTAINER PLATFORM 安全性上下文约束和 LINUX 功能 33
6.3. 其他资源 34

第 7 章 使用 CLI 工具 **35**
7.1. 先决条件 35
7.2. VIRTCTL 客户端命令 35
7.3. OPENSHIFT CONTAINER PLATFORM 客户端命令 36

第 8 章 虚拟机 **38**
8.1. 创建虚拟机 38
8.2. 编辑虚拟机 49
8.3. 编辑引导顺序 54
8.4. 删除虚拟机 56
8.5. 管理虚拟机实例 57
8.6. 控制虚拟机状态 59
8.7. 访问虚拟机控制台 61
8.8. 解决故障节点来触发虚拟机故障切换 69
8.9. 在虚拟机上安装 QEMU 客户机代理 70
8.10. 查看虚拟机的 QEMU 客户机代理信息 72

8.11. 在虚拟机中管理配置映射、SECRET 和服务帐户 73
8.12. 在现有 WINDOWS 虚拟机上安装 VIRTIO 驱动程序 74
8.13. 在新 WINDOWS 虚拟机上安装 VIRTIO 驱动程序 77
8.14. 高级虚拟机管理 80
8.15. 导入虚拟机 107
8.16. 克隆虚拟机 144
8.17. 虚拟机网络 155
8.18. 虚拟机磁盘 175

第 9 章 虚拟机模板 222
9.1. 创建虚拟机模板 222
9.2. 编辑虚拟机模板 231
9.3. 为虚拟机模板启用专用资源 233
9.4. 删除虚拟机模板 234

第 10 章 实时迁移 235
10.1. 虚拟机实时迁移 235
10.2. 实时迁移限制和超时 235
10.3. 迁移虚拟机实例到另一节点 237
10.4. 监控虚拟机实例的实时迁移 238
10.5. 取消虚拟机实例的实时迁移 239
10.6. 配置虚拟机驱除策略 239

第 11 章 节点维护 241
11.1. 关于节点维护 241
11.2. 将节点设置为维护模式 241
11.3. 从维护模式恢复节点 244
11.4. 自动续订 TLS 证书 245
11.5. 为过时的 CPU 型号管理节点标签 246
11.6. 防止节点协调 249

第 12 章 节点网络 250
12.1. 观察节点网络状态 250
12.2. 更新节点网络配置 251
12.3. 对节点网络配置进行故障排除 261

第 13 章 日志记录、事件和监控 266
13.1. 查看虚拟机日志 266
13.2. 查看事件 266
13.3. 使用事件和条件诊断数据卷 267
13.4. 查看有关虚拟机工作负载的信息 269
13.5. 监控虚拟机健康状况 270
13.6. 使用 OPENSHIFT CONTAINER PLATFORM DASHBOARD 获取集群信息 274
13.7. OPENSHIFT CONTAINER PLATFORM 集群监控、日志记录和遥测技术 275
13.8. PROMETHEUS 对虚拟资源的查询 277
13.9. 为红帽支持收集数据 281

第 1 章 关于 OPENSHIFT VIRTUALIZATION

OpenShift Virtualization 的功能与支持范围。

1.1. OPENSHIFT VIRTUALIZATION 的作用

OpenShift 虚拟化（OpenShift virtualization）是 OpenShift Container Platform 的一个附加组件，可用于运行和管理虚拟机工作负载以及容器工作负载。

OpenShift Virtualization 通过 Kubernetes 自定义资源添加新对象至 OpenShift Container Platform 集群中，以启用虚拟化任务。这些任务包括：

- 创建和管理 Linux 和 Windows 虚拟机
- 通过各种控制台和 CLI 工具连接至虚拟机
- 导入和克隆现有虚拟机
- 管理虚拟机上附加的网络接口控制器和存储磁盘
- 在节点间实时迁移虚拟机

增强版 web 控制台提供了一个图形化的门户界面 来管理虚拟化资源以及 OpenShift Container Platform 集群容器和基础架构。

OpenShift Virtualization 与 OpenShift Container Storage（OCS）进行了测试，它旨在与 OCS 功能一起使用以获得最佳体验。

您可以将 OpenShift Virtualization 与 OVN-Kubernetes、OpenShiftSDN或认证的 OpenShift CNI 插件 中列出的其他默认 Container Network Interface (CNI) 网络供应商一起使用。

1.1.1. OpenShift Virtualization 支持的集群版本

OpenShift Virtualization 4.8 支持在 OpenShift Container Platform 4.8 集群中使用。要使用 OpenShift Virtualization 的最新 z-stream 版本，您必须首先升级到 OpenShift Container Platform 的最新版本。

第 2 章 从 OPENSHIFT VIRTUALIZATION 开始

使用下表查找内容以帮助您了解和使用 OpenShift Virtualization。

2.1. 集群管理员

学习	计划	部署	其他资源
了解 OpenShift Virtualization	为 OpenShift Virtualization 配置集群	更新节点网络配置	获得支持
了解有关 OpenShift Container Platform 的更多信息	为虚拟机磁盘计划存储	配置 CSI 卷	
了解虚拟机实时迁移		使用 OpenShift Virtualization 控制台或 CLI安装 OpenShift Virtualization。	
了解节点维护			

2.2. 虚拟化管理员

学习	部署	管理	使用
了解 OpenShift Virtualization	将虚拟机连接至虚拟机的默认 pod 网络以及外部网络	安装 **virtctl** 客户端	使用容器的 Migration Toolkit 导入虚拟机
了解虚拟机磁盘的存储功能	自定义存储配置集	使用 CLI 工具	使用实时迁移
	创建引导源并将其附加到模板	查看日志和事件	
	更新引导源模板	监控虚拟机健康状况	

2.3. 虚拟机管理员/开发人员

学习	使用	管理	其他资源
了解 OpenShift Virtualization	安装 **virtctl** 客户端	查看日志和事件	获得支持
	创建虚拟机	监控虚拟机健康状况	

学习	使用	管理	其他资源
	管理虚拟机实例	创建和管理虚拟机快照	
	控制虚拟机状态		
	访问虚拟机控制台		
	使用 secrets、配置映射和服务帐户将配置数据传递给虚拟机		

第 3 章 OPENSHIFT VIRTUALIZATION 发行注记

3.1. 关于 RED HAT OPENSHIFT VIRTUALIZATION

Red Hat OpenShift Virtualization 可让您将传统虚拟机（VM）放入 OpenShift Container Platform 中，与容器一同运行，并作为原生 Kubernetes 对象进行管理。

OpenShift Virtualization 由  徽标表示。

OpenShift Virtualization 可以与 OVN-Kubernetes 或 OpenShiftSDN 默认 Container Network Interface（CNI）网络供应商一起使用。

了解更多有关 OpenShift Virtualization 的作用。

3.1.1. OpenShift Virtualization 支持的集群版本

OpenShift Virtualization 4.8 支持在 OpenShift Container Platform 4.8 集群中使用。要使用 OpenShift Virtualization 的最新 z-stream 版本，您必须首先升级到 OpenShift Container Platform 的最新版本。

3.1.2. 支持的客户端操作系统

OpenShift Virtualization 客户端可使用以下操作系统：

- Red Hat Enterprise Linux 6、7 和 8。
- Microsoft Windows Server 2012 R2、2016 和 2019。
- Microsoft Windows 10.

不支持 OpenShift Virtualization 附带的其他操作系统模板。

3.2. 使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始： master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

3.3. 新增和改变的功能

- OpenShift Virtualization 已在 Microsoft 的 Windows Server Virtualization Validation Program (SVVP) 中认证来运行 Windows Server 的工作负载。
  SVVP 认证适用于：
  - Red Hat Enterprise Linux CoreOS worker。在 Microsoft SVVP Catalog 中，它们名为 *Red Hat OpenShift Container Platform 4 on RHEL CoreOS*。
  - Intel 和 AMD CPU。
- Containerized Data Importer（CDI）现在使用 OpenShift Container Platform 集群范围的代理配置。

- OpenShift Virtualization 现在支持由红帽认证的第三方 Container Network Interface (CNI) 插件以用于 OpenShift Container Platform。
- OpenShift Virtualization 现在提供用于监控集群中如何使用基础架构资源的指标。您可以使用 OpenShift Container Platform 监控仪表板来查询以下资源的指标:
  - vCPU
  - Network
  - 存储
  - 虚拟机内存交换
- OpenShift Virtualization 现在提供了一个统一的 API 来配置证书轮转。
- 如果从模板创建 Windows 虚拟机或预定义的 Hyper-V 功能，则它现在只能调度到支持 Hyper-V 的节点。
- **virtctl vnc** 命令的 **--proxy-only** 选项允许您使用任何 VNC viewer 通过虚拟网络客户端（VNC）连接手动连接到虚拟机实例。

3.3.1. 快速启动

- 有几个 OpenShift Virtualization 功能提供快速入门导览。要查看导览，请点击 OpenShift Virtualization 控制台标题菜单栏中的 **Help** 图标 **?**，然后选择 **Quick Starts**。您可以通过在 **Filter** 字段中输入 **virtualization** 关键字来过滤可用的导览。

3.3.2. 网络

- 您可以使用 Kubernetes NMstate Operator 配置和管理集群节点上的 IP 地址。
- OpenShift Virtualization 现在支持实时迁移附加到 SR-IOV 网络接口的虚拟机，如果 **HyperConverged** 自定义资源（CR）中启用了 **sriovLiveMigration** 功能门。

3.3.3. 存储

- 现在，在使用支持 Container Storage Interface（CSI）快照的存储时，将数据卷克隆到不同命名空间中的速度和效率提高。Containerized Data Importer（CDI）使用 CSI 快照（当它们可用时）来提高从模板创建虚拟机时的性能。
- 在虚拟磁盘上运行 **fstrim** 或 **blkdiscard** 命令时，丢弃请求将传递到底层存储设备。如果存储供应商支持 Pass Discard 功能，丢弃请求会释放存储容量。
- 现在，您可以使用存储 API 指定数据卷。存储 API 与 PVC API 不同，允许系统在分配存储时优化 **accessMode**、**volumeMode** 和存储容量。
- 现在，如果虚拟机有内容类型 **kubevirt**，您可以在不同的数据卷模式间克隆虚拟机磁盘。例如，您可以将带有 **volumeMode: Block** 的持久性卷 (PV) 克隆到带有 **volumeMode: Filesystem.** 的 PV 中。
- 您可以通过在 OpenShift Virtualization 控制台中运行向导来创建自定义磁盘镜像作已定义源的任何模板的引导源。

3.4. 弃用和删除的功能

3.4.1. 已弃用的功能

弃用的功能包括在当前发行版本中并被支持。但是，它们将在以后的发行版本中删除，且不建议用于新部署。

- 从 Red Hat Virtualization（RHV）或 VMware 导入单个虚拟机已在当前发行版本中弃用，并将在 OpenShift Virtualization 4.9 中删除。此功能由虚拟化的 Migration Toolkit 替代。

3.5. 主要的技术变化

- 当在启用了双栈网络的集群中运行时，OpenShift Virtualization 现在会配置 IPv6 地址。如果底层 OpenShift Container Platform 集群启用了双栈网络，您可以创建使用 IPv4、IPv6 或两个 IP 地址系列的服务。
- 现在，在安装 OpenShift Virtualization 时 KubeMacPool 会被默认启用。您可以通过将 **mutatevirtualmachines.kubemacpool.io=ignore** 标签添加到命名空间来在命名空间中禁用一个 MAC 地址池。通过删除标签为命名空间重新启用 KubeMacPool。
- **HyperConverged** 自定义资源 (CR) 现在是 OpenShift Virtualization 配置的中心。通过编辑 **HyperConverged** CR，您可以：
  - 配置实时迁移限制和超时
  - 定义 VMware Virtual Disk Development Kit (VDDK) 镜像的位置
  - 配置过时的 CPU 型号
  - 为容器 registry 禁用 TLS
  - 为涂销空间配置存储类
  - 配置存储工作负载的资源要求

3.6. 技术预览功能

这个版本中的一些功能当前还处于技术预览状态。它们并不适用于在生产环境中使用。请参阅红帽门户网站中关于对技术预览功能支持范围的信息：

技术预览功能支持范围

- 现在，对于热插拔虚拟磁盘，可以在不停止虚拟机实例的情况下把它们添加到虚拟机或从虚拟机中删除它们。

3.7. 已知问题

- 更新至 OpenShift Virtualization 4.8.7 会导致一些虚拟机(VM) 处于实时迁移循环中。如果虚拟机清单中的 **spec.volumes.containerDisk.path** 字段设置为相对路径，会出现这种情况。
  - 作为临时解决方案，删除并重新创建 VM 清单，将 **spec.volumes.containerDisk.path** 字段的值设置为绝对路径。然后您可以更新 OpenShift Virtualization。
- 如果您最初部署了 OpenShift Virtualization 版本 2.4.z 或更早版本，升级到 4.8 版本会失败并显示以下信息：

```
risk of data loss updating hyperconvergeds.hco.kubevirt.io: new CRD removes
version v1alpha1 that is listed as a stored version on the existing CRD
```

此程序错误不会影响最初在 2.5.0 或更高版本部署 OpenShift Virtualization 的集群。(BZ#1986989)

- 作为临时解决方案，从 **HyperConverged** 自定义资源定义 (CRD) 中删除 **v1alpha1** 版本，并恢复升级过程：

  1. 运行以下命令，打开集群的代理连接：

     ```
     $ oc proxy &
     ```

  2. 运行以下命令，从 **HyperConverged** CRD 上的 **.status.storedVersions** 中删除 **v1alpha1** 版本：

     ```
     $ curl --header "Content-Type: application/json-patch+json" --request PATCH
     http://localhost:8001/apis/apiextensions.k8s.io/v1/customresourcedefinitions/hyperconve
     rgeds.hco.kubevirt.io/status --data '[{"op": "replace", "path": "/status/storedVersions",
     "value":["v1beta1"]}]'
     ```

  3. 运行以下命令恢复升级过程：

     ```
     $ curl --header "Content-Type: application/json-patch+json" --request PATCH
     http://localhost:8001/apis/operators.coreos.com/v1alpha1/namespaces/openshift-
     cnv/installplans/$(oc get installplan -n openshift-cnv | grep kubevirt-hyperconverged-
     operator.v4.8.0 | cut -d' ' -f1)/status --data '[{"op": "remove", "path":
     "/status/conditions"},{"op": "remove", "path": "/status/message"},{"op": "replace",
     "path": "/status/phase", "value": "Installing"}]'
     ```

  4. 运行以下命令来终止 **oc proxy** 进程：

     ```
     $ kill $(ps -C "oc proxy" -o pid=)
     ```

  5. 可选：运行以下命令来监控升级状态：

     ```
     $ oc get csv
     ```

- 如果您删除版本 4.8 或更高版本中的 OpenShift Virtualization 提供的模板，OpenShift Virtualization Operator 会自动重新创建模板。但是，如果您删除版本 4.8 之前创建的 OpenShift Virtualization 提供的模板，这些以前的模板不会在删除后自动重新创建。因此，引用之前模板的虚拟机的任何编辑或更新都将失败。

- 如果在源可用前启动克隆操作，则操作会无限期停止。这是因为克隆授权在克隆操作启动前过期。(BZ#1855182)

  - 作为临时解决方案，删除正在请求克隆的 **DataVolume** 对象。当源可用时，重新创建您删除的 **DataVolume** 对象，以便克隆操作可以成功完成。

- 如果您的 OpenShift Container Platform 集群使用 OVN-Kubernetes 作为默认 Container Network Interface（CNI）供应商，则无法将 Linux 网桥或绑定附加到主机的默认接口，因为 OVN-Kubernetes 的主机网络拓扑发生了变化。(BZ#1885605)

  - 作为临时解决方案，您可以使用连接到主机的二级网络接口，或切换到 OpenShift SDN 默认 CNI 供应商。

- 运行无法实时迁移的虚拟机可能会阻止 OpenShift Container Platform 集群升级。这包括使用 hostpath-provisioner 存储或 SR-IOV 网络接口的虚拟机。(BZ#1858777)
  - 作为临时解决方案，您可以重新配置虚拟机以便在集群升级过程中关闭它们。在虚拟机配置文件的 **spec** 部分中：
    1. 删除 **evictionStrategy: LiveMigrate 字段**。有关如何配置驱除策略的更多信息，请参阅配置虚拟机驱除策略。
    2. 将 **runStrategy** 字段设置为 **Always**。
- 当节点具有不同的 CPU 型号时，实时迁移会失败。即使节点具有相同的物理 CPU 型号，微代码更新引入的差异也会产生同样的问题。这是因为默认设置触发了主机 CPU 透传行为，这与实时迁移不兼容。(BZ#1760028)
  - 作为临时解决方案，请运行以下命令设置默认 CPU 模型：

注意

您必须在启动支持实时迁移的虚拟机前进行此更改。

```
$ oc annotate --overwrite -n openshift-cnv hyperconverged kubevirt-hyperconverged
kubevirt.kubevirt.io/jsonpatch='[
 {
    "op": "add",
    "path": "/spec/configuration/cpuModel",
    "value": "<cpu_model>" 1
 }
]'
```

1 将 **<cpu_model>** 替换为实际 CPU 型号值。要确定此值，您可以为所有节点运行 **oc describe node <node>** 并查看 **cpu-model-<name>** 标签。选择所有节点上出现的 CPU 型号。

- 在导入 RHV 虚拟机的过程中，如果您为 RHV Manager 输入了错误的凭据，Manager 可能会锁定 admin 用户帐户，因为 **vm-import-operator** 会尝试多次连接到 RHV API。(BZ#1887140)
  - 要解锁帐户，请登录到 Manager 并输入以下命令：

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

- 如果您使用 OpenShift Container Platform 4.8 运行 OpenShift Virtualization 2.6.5，则会出现各种问题。您可以通过将 OpenShift Virtualization 升级到 4.8 版本来避免这些问题。
  - 在 web 控制台中，如果您导航到 **Virtualization** 页面并选择 **Create → With YAML**，会显示以下错误消息：

```
The server doesn't have a resource type "kind: VirtualMachine, apiVersion:
kubevirt.io/v1"
```

    - 作为临时解决方案，编辑 **VirtualMachine** 清单，使 **apiVersion** 为 **kubevirt.io/v1alpha3**。例如：

```
apiVersion: kubevirt.io/v1alpha3
```

```
kind: VirtualMachine
metadata:
  annotations:
...
```

(BZ#1979114)

- 如果使用 **Customize** 向导来创建虚拟机，则会显示以下错误消息：

```
Error creating virtual machine
```

  - 作为临时解决方案，复制清单并从 CLI 创建虚拟机。
(BZ#1979116)

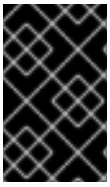
- 当使用 OpenShift Virtualization web 控制台连接到 VNC 控制台时，VNC 控制台总是无法响应。

  - 作为临时解决方案，请通过 CLI 创建虚拟机或升级到 OpenShift Virtualization 4.8。
(BZ#1977037)

第 4 章 安装 OPENSHIFT VIRTUALIZATION

4.1. 为 OPENSHIFT VIRTUALIZATION 准备集群

在安装 OpenShift Virtualization 前，参阅这个部分以确保集群满足要求。



重要

您可以使用任何安装方法（包括用户置备的、安装程序置备或辅助安装程序）来部署 OpenShift Container Platform。但是，安装方法和集群拓扑可能会影响 OpenShift Virtualization 功能，如快照或实时迁移。

FIPS 模式

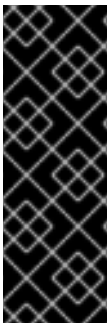
如果使用 FIPS 模式安装集群，则 OpenShift Virtualization 不需要额外的设置。

4.1.1. 硬件和操作系统要求

查看 OpenShift Virtualization 的以下硬件和操作系统要求。

支持的平台

- 内部裸机服务器
- Amazon Web Services 裸机实例



重要

在 AWS 裸机实例上安装 OpenShift Virtualization 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

- **不支持由其他云供应商提供的裸机实例或服务器。**

CPU 要求

- Red Hat Enterprise Linux(RHEL)8 支持
- 支持 Intel 64 或 AMD64 CPU 扩展
- 启用 Intel VT 或 AMD-V 硬件虚拟化扩展
- 启用 NX（无执行）标记

存储要求

- OpenShift Container Platform 支持

操作系统要求

- 在 worker 节点上安装的 Red Hat Enterprise Linux CoreOS(RHCOS)

注意

不支持 RHEL worker 节点。

其他资源

- 关于 RHCOS
- 用于支持的 CPU 的红帽生态系统目录
- 支持的存储

4.1.2. 物理资源开销要求

OpenShift Virtualization 是 OpenShift Container Platform 的一个附加组件，它会带来额外的开销。除了 OpenShift Container Platform 要求外，每个集群机器都必须满足以下开销要求。覆盖集群中的物理资源可能会影响性能。

重要

本文档中给出的数字基于红帽的测试方法和设置。这些数字会根据您自己的设置和环境而有所不同。

4.1.2.1. 内存开销

使用以下因素计算 OpenShift Virtualization 的内存开销值。

集群内存开销

```
Memory overhead per infrastructure node ≈ 150 MiB
```

```
Memory overhead per worker node ≈ 360 MiB
```

另外，OpenShift Virtualization 环境资源需要总计 2179 MiB 的内存，分布到所有基础架构节点。

虚拟机内存开销

```
Memory overhead per virtual machine ≈ (1.002 * requested memory) + 146 MiB  \
              + 8 MiB * (number of vCPUs) \ 1
             + 16 MiB * (number of graphics devices) 2
```

1 虚拟机请求的虚拟 CPU 数量

2 虚拟机请求的虚拟图形卡数

如果您的环境包含单一根 I/O 虚拟化（SR-IOV）网络设备或图形处理单元（GPU），请为每个设备分配 1 GiB 额外的内存开销。

4.1.2.2. CPU 开销

使用以下内容计算 OpenShift Virtualization 的集群处理器开销要求。每个虚拟机的 CPU 开销取决于您的单独设置。

集群 CPU 开销

> CPU overhead for infrastructure nodes ≈ 4 cores

OpenShift Virtualization 增加集群级别服务的整体使用，如日志记录、路由和监控。要考虑这个工作负载，请确保托管基础架构组件的节点分配了用于不同节点的 4 个额外内核（4000 毫秒）的容量。

> CPU overhead for worker nodes ≈ 2 cores + CPU overhead per virtual machine

除了虚拟机工作负载所需的 CPU 外，每个托管虚拟机的 worker 节点都必须有 2 个额外内核（2000 毫秒）用于 OpenShift Virtualization 管理工作负载。

虚拟机 CPU 开销

如果请求专用 CPU，则会对集群 CPU 开销要求有 1:1 影响。否则，没有有关虚拟机所需 CPU 数量的具体规则。

4.1.2.3. 存储开销

使用以下指南来估算 OpenShift Virtualization 环境的存储开销要求。

集群存储开销

> Aggregated storage overhead per node ≈ 10 GiB

10 GiB 在安装 OpenShift Virtualization 时，集群中每个节点的磁盘存储影响估计值。

虚拟机存储开销

每个虚拟机的存储开销取决于虚拟机内的具体资源分配请求。该请求可能用于集群中其他位置托管的节点或存储资源的临时存储。OpenShift Virtualization 目前不会为正在运行的容器本身分配任何额外的临时存储。

4.1.2.4. 示例

作为集群管理员，如果您计划托管集群中的 10 个虚拟机，每个虚拟机都有 1 GiB RAM 和 2 个 vCPU，集群中的内存影响为 11.68 GiB。集群中每个节点的磁盘存储影响估算为 10 GiB，托管虚拟机工作负载的 worker 节点的 CPU 影响最小 2 个内核。

4.1.3. 对象最大值

在规划集群时，您必须考虑以下测试的对象最大值：

- OpenShift Container Platform 对象最大值
- OpenShift Virtualization 对象最大值

4.1.4. 受限网络环境

如果在没有互联网连接的受限环境中安装 OpenShift Virtualization，您必须为受限网络配置 Operator Lifecycle Manager。

如果您拥有有限的互联网连接，您可以在 Operator Lifecycle Manager 中配置代理支持 以访问红帽提供的 OperatorHub。

4.1.5. 实时迁移

实时迁移有以下要求：

- 使用 **ReadWriteMany** (RWX)访问模式的共享存储
- 足够的 RAM 和网络带宽
- worker 节点上具有足够容量的适当 CPU。如果 CPU 具有不同的容量，实时迁移可能会非常慢或失败。

4.1.6. 快照和克隆

有关快照和克隆要求，请参阅 OpenShift Virtualization 存储功能。

4.1.7. 集群高可用性选项

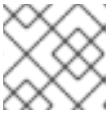
您可以为集群配置以下高可用性(HA)选项之一：

- 通过部署机器健康检查，可以使用安装程序置备的基础架构 (IPI)自动高可用性。

注意

在使用安装程序置备的基础架构安装并正确配置 MachineHealthCheck 的 OpenShift Container Platform 集群中，如果节点上的 MachineHealthCheck 失败且对集群不可用，则该节点可以被回收使用。在故障节点上运行的虚拟机之后会发生什么，这取决于一系列条件。如需了解更多有关潜在结果以及 RunStrategies 如何影响这些结果的信息，请参阅虚拟机的 RunStrategies。

- 任何平台的高可用性可通过使用监控系统或合格的人类监控节点可用性来实现。当节点丢失时，关闭并运行 **oc delete node <lost_node>**。



注意

如果没有外部监控系统或合格的人类监控节点运行状况，虚拟机就失去高可用性。

4.2. 为 OPENSHIFT VIRTUALIZATION 组件指定节点

通过配置节点放置规则来指定要部署 OpenShift Virtualization Operator、工作负载和控制器的节点。

注意

您可以在安装 OpenShift Virtualization 后为一些组件配置节点放置，但如果要为工作负载配置节点放置，则一定不能存在虚拟机。

4.2.1. 关于虚拟化组件的节点放置

您可能想要自定义 OpenShift Virtualization 在什么位置部署其组件，以确保：

- 虚拟机仅部署到设计为用于虚拟化工作负载的节点上。

- Operator 仅在基础架构节点上部署。
- 某些节点不会受到 OpenShift Virtualization 的影响。例如，您有与集群中运行的虚拟化不相关的工作负载，希望这些工作负载与 OpenShift Virtualization 分离。

4.2.1.1. 如何将节点放置规则应用到虚拟化组件

您可以通过直接编辑对应对象或使用 Web 控制台为组件指定节点放置规则。

- 对于 Operator Lifecycle Manager（OLM）部署的 OpenShift Virtualization Operator，直接编辑 OLM **Subscription** 对象。目前，您无法使用 Web 控制台为 **Subscription** 对象配置节点放置规则。
- 对于 OpenShift Virtualization Operator 部署的组件,直接编辑 **HyperConverged** 对象，或在 OpenShift Virtualization 安装过程中使用 Web 控制台进行配置。
- 对于 hostpath 置备程序，直接编辑 **HostPathProvisioner** 对象，或使用 web 控制台进行配置。

警告

您必须将 hostpath 置备程序和虚拟化组件调度到同一节点上。否则，使用 hostpath 置备程序的虚拟化 pod 无法运行。

根据对象，您可以使用以下一个或多个规则类型：

nodeSelector

允许将 Pod 调度到使用您在此字段中指定的键值对标记的节点上。节点必须具有与所有列出的对完全匹配的标签。

关联性

可让您使用更宽松的语法来设置与 pod 匹配的规则。关联性也允许在规则应用方面更加精细。例如，您可以指定规则是首选项，而不是硬要求，因此如果不满足该规则，仍可以调度 pod。

容限（tolerations）

允许将 pod 调度到具有匹配污点的节点。如果某个节点有污点（taint），则该节点只接受容许该污点的 pod。

4.2.1.2. 放置在 OLM 订阅对象中的节点

要指定 OLM 部署 OpenShift Virtualization Operator 的节点，在 OpenShift Virtualization 安装过程中编辑 **Subscription** 对象。您可以在 **spec.config** 字段中包含节点放置规则，如下例所示：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
```

```
  startingCSV: kubevirt-hyperconverged-operator.v4.8.7
  channel: "stable"
  config: 1
```

1 **config** 字段支持 **nodeSelector** 和 **tolerations**，但它不支持 **关联性**。

4.2.1.3. HyperConverged 对象中的节点放置

要指定 OpenShift Virtualization 部署其组件的节点，您可以在 OpenShift Virtualization 安装过程中创建的 HyperConverged Cluster 自定义资源（CR）文件中包含 **nodePlacement** 对象。您可以在 **spec.infra** 和 **spec.workloads** 字段中包含 **nodePlacement**，如下例所示：

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement: 1
    ...
  workloads:
    nodePlacement:
    ...
```

1 **nodePlacement** 字段支持 **nodeSelector**、**affinity** 和 **tolerations** 字段。

4.2.1.4. HostPathProvisioner 对象中的节点放置

您可以在安装 hostpath 置备程序时创建的 **HostPathProvisioner** 对象的 **spec.workload** 字段中配置节点放置规则。

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  pathConfig:
    path: "</path/to/backing/directory>"
    useNamingPrefix: false
  workload: 1
```

1 **workload** 字段支持 **nodeSelector**、**affinity** 和 **tolerations** 字段。

4.2.1.5. 其他资源

- 为虚拟机指定节点
- 使用节点选择器将 pod 放置到特定节点

- 使用节点关联性规则控制节点上的 pod 放置
- 使用节点污点控制 pod 放置
- 使用 CLI 安装 OpenShift Virtualization
- 使用 Web 控制台卸载 OpenShift Virtualization
- 为虚拟机配置本地存储

4.2.2. 清单示例

以下示例 YAML 文件使用 **nodePlacement**、**affinity（关联性）** 和 **tolerations（容限）**对象为 OpenShift Virtualization 组件自定义节点放置。

4.2.2.1. Operator Lifecycle Manager Subscription 对象

4.2.2.1.1. 示例：在 OLM 订阅对象中使用 nodeSelector 的节点放置

在本例中，配置了 **nodeSelector**，OLM 将 OpenShift Virtualization Operator 放置到标记为 **example.io/example-infra-key = example-infra-value** 的节点上。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.8.7
  channel: "stable"
  config:
    nodeSelector:
      example.io/example-infra-key: example-infra-value
```

4.2.2.1.2. 示例：将容限放置在 OLM 订阅对象中

在本例中，为 OLM 部署 OpenShift Virtualization Operator 保留的节点使用 **key=virtualization:NoSchedule** 污点标记。只有具有与容限匹配的 pod 才会调度到这些节点。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.8.7
  channel: "stable"
  config:
```

```
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "virtualization"
    effect: "NoSchedule"
```

4.2.2.2. HyperConverged 对象

4.2.2.2.1. 示例： 在 HyperConverged Cluster CR 中使用 nodeSelector 进行节点放置

在本例中，配置了 **nodeSelector**，将基础架构资源放置在带有 **example.io/example-infra-key = example-infra-value = example-infra-value** 的节点上，把工作负载放置在带有 **example.io/example-workloads-key = example-workloads-value** 的节点上。

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement:
      nodeSelector:
        example.io/example-infra-key: example-infra-value
  workloads:
    nodePlacement:
      nodeSelector:
        example.io/example-workloads-key: example-workloads-value
```

4.2.2.2.2. 示例：在 HyperConverged Cluster CR 中使用关联性进行节点放置

在本例中，配置了 **affinity**，将基础架构资源放置在带有 **example.io/example-infra-key = example-value** 的节点上，把工作负载放置在带有 **example.io/example-workloads-key = example-workloads-value** 的节点上。对于工作负载，最好使用八个以上 CPU 的节点，但如果它们不可用，仍可调度 pod。

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  infra:
    nodePlacement:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: example.io/example-infra-key
                operator: In
                values:
                - example-infra-value
  workloads:
    nodePlacement:
```

```
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: example.io/example-workloads-key
              operator: In
              values:
              - example-workloads-value
        preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
            - key: example.io/num-cpus
              operator: Gt
              values:
              - 8
```

4.2.2.2.3. 示例：在 HyperConverged Cluster CR 中使用容限进行节点放置

在本例中，为 OpenShift Virtualization 组件保留的节点使用 **key=virtualization:NoSchedule** 污点标记。只有具有与容限匹配的 pod 才会调度到这些节点。

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  workloads:
    nodePlacement:
      tolerations:
      - key: "key"
        operator: "Equal"
        value: "virtualization"
        effect: "NoSchedule"
```

4.2.2.3. HostPathProvisioner 对象

4.2.2.3.1. 示例： HostPathProvisioner 对象中的 nodeSelector 的节点放置

在本例中，配置了 **nodeSelector**，以便将工作负载放置到带有 **example.io/example-workloads-key = example-workloads-value** 的节点上。

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  pathConfig:
    path: "</path/to/backing/directory>"
    useNamingPrefix: false
```

```
workload:
  nodeSelector:
    example.io/example-workloads-key: example-workloads-value
```

4.3. 使用 WEB 控制台卸载 OPENSHIFT VIRTUALIZATION

安装 OpenShift Virtualization 以便在 OpenShift Container Platform 集群中添加虚拟化功能。

您可以使用 OpenShift Container Platform 4.8 web 控制台来订阅和部署 OpenShift Virtualization Operator。

4.3.1. 安装 OpenShift Virtualization Operator

您可以从 OpenShift Container Platform Web 控制台安装 OpenShift Virtualization Operator。

先决条件

- 在集群上安装 OpenShift Container Platform 4.8。
- 以具有 **cluster-admin** 权限的用户身份登录到 OpenShift Container Platform web 控制台。

流程

1. 从 **Administrator** 视角中，点 **Operators → OperatorHub**。
2. 在 **Filter by keyword** 字段中输入 **OpenShift Virtualization**。
3. 选择 **OpenShift Virtualization** 标题。
4. 阅读 Operator 信息并单击 **Install**。
5. 在 **Install Operator** 页面中：
   a. 从可用 **Update Channel** 选项列表中选择 **stable**。这样可确保安装与 OpenShift Container Platform 版本兼容的 OpenShift Virtualization 版本。
   b. **对于安装的命名空间**，请确保选择了 **Operator 推荐的命名空间**选项。这会在 **openshift-cnv** 命名空间中安装 Operator，该命名空间在不存在时自动创建。

警告

尝试在 **openshift-cnv** 以外的命名空间中安装 OpenShift Virtualization Operator 会导致安装失败。

   c. 对于 **Approval Strategy**，强烈建议您选择 **Automatic** （默认值），以便在 **stable** 更新频道中提供新版本时 OpenShift Virtualization 会自动更新。
   虽然可以选择 **Manual** 批准策略，但这不可取，因为它会给集群提供支持和功能带来高风险。只有在您完全了解这些风险且无法使用 **Automatic** 时，才选择 **Manual**。

警告

因为 OpenShift Virtualization 只在与对应的 OpenShift Container Platform 版本搭配使用时被支持，所以缺少的 OpenShift Virtualization 更新可能会导致您的集群不被支持。

6. 点击 **Install** 使 Operator 可供 **openshift-cnv** 命名空间使用。
7. 当 Operator 成功安装时，点 **Create HyperConverged**。
8. 可选：为 OpenShift Virtualization 组件配置 **Infra** 和 **Workloads** 节点放置选项。
9. 点击 **Create** 启动 OpenShift Virtualization。

验证

- 导航到 **Workloads → Pods** 页面，并监控 OpenShift Virtualization Pod，直至全部处于 **Running** 状态。在所有 pod 都处于 **Running** 状态后，您可以使用 OpenShift Virtualization。

4.3.2. 后续步骤

您可能还需要额外配置以下组件：

- hostpath 置备程序是设计用于 OpenShift Virtualization 的本地存储置备程序。如果要为虚拟机配置本地存储，您必须首先启用 hostpath 置备程序。

安装 OpenShift Virtualization 以便在 OpenShift Container Platform 集群中添加虚拟化功能。您可以使用命令行将清单应用到集群，以订阅和部署 OpenShift Virtualization Operator。

注意

要指定 OpenShift Virtualization 安装其组件的节点，请配置节点放置规则。

4.3.3. 先决条件

- 在集群上安装 OpenShift Container Platform 4.8。
- 安装 OpenShift CLI (**oc**)。
- 以具有 **cluster-admin** 特权的用户身份登录。

4.3.4. 使用 CLI 订阅 OpenShift virtualization 目录

在安装 OpenShift Virtualization 前，需要订阅到 OpenShift Virtualization catalog。订阅会授予 OpenShift virtualization Operator 对 **openshift-cnv** 命名空间的访问权限。

为了订阅，在您的集群中应用一个单独的清单（manifest）来配置 **Namespace**、**OperatorGroup** 和 **Subscription** 对象。

流程

1. 创建一个包含以下清单的 YAML 文件：

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-cnv
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: kubevirt-hyperconverged-group
  namespace: openshift-cnv
spec:
  targetNamespaces:
    - openshift-cnv
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: hco-operatorhub
  namespace: openshift-cnv
spec:
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  name: kubevirt-hyperconverged
  startingCSV: kubevirt-hyperconverged-operator.v4.8.7
  channel: "stable" (1)
```

(1) 使用 **stable** 频道可确保您安装与 OpenShift Container Platform 版本兼容的 OpenShift Virtualization 版本。

2. 运行以下命令,为 OpenShift Virtualization 创建所需的 **Namespace**、**OperatorGroup** 和 **Subscription**对象：

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

注意

您可以在 YAML 文件中配置证书轮转参数。

4.3.5. 使用 CLI 部署 OpenShift Virtualization Operator

您可以使用 **oc** CLI 部署 OpenShift Virtualization Operator。

先决条件

- 在 **openshift-cnv** 命名空间中的一个有效的 OpenShift virtualization 目录订阅。

流程

1. 创建一个包含以下清单的 YAML 文件：

```
apiVersion: hco.kubevirt.io/v1beta1
```

```
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
```

2. 运行以下命令来部署 OpenShift Virtualization Operator:

```
$ oc apply -f <file_name>.yaml
```

验证

- 通过观察 **openshift-cnv** 命名空间中集群服务版本（CSV）的 **PHASE** 来确保 OpenShift Virtualization 已被成功部署。运行以下命令：

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

如果部署成功，则会显示以下输出：

输出示例

```
NAME                                       DISPLAY                    VERSION   REPLACES   PHASE
kubevirt-hyperconverged-operator.v4.8.7    OpenShift Virtualization   4.8.7
Succeeded
```

4.3.6. 后续步骤

您可能还需要额外配置以下组件：

- hostpath 置备程序是设计用于 OpenShift Virtualization 的本地存储置备程序。如果要为虚拟机配置本地存储，您必须首先启用 hostpath 置备程序。

4.4. 安装 VIRTCTL 客户端

virtctl 客户端是用于管理 OpenShift Virtualization 资源的命令行实用程序。它适用于 Linux、macOS 和 Windows 发行版。

您可从 OpenShift Virtualization web 控制台或启用 OpenShift Virtualization 仓库并安装 **kubevirt-virtctl** 软件包来安装 **virtctl** 客户端。

4.4.1. 从 web 控制台安装 virtctl 客户端

您可从红帽客户门户网站下载 **virtctl** 客户端，OpenShift Virtualization web 控制台的 **Command Line Tools** 页面中包括了该门户网站的链接 。

先决条件

- 您必须具有有效的 OpenShift Container Platform 订阅才能访问客户门户网站的下载页面。

流程

1. 点击位于 web 控制台右上角的  图标访问客户门户网站，并选择 **Command Line Tools**。

2. 确保您有从 **Version:** 列表中选择的集群的适当版本。
3. 为您的发行版本下载 **virtctl** 客户端。所有下载都是 **tar.gz** 格式。
4. 解压 tarball。以下 CLI 命令将其解压到 tarball 所在的目录中，并适用于所有发行版本：

```
$ tar -xvf <virtctl-version-distribution.arch>.tar.gz
```

5. 对于 Linux 和 macOS:
   a. 进入解压的目录，使 **virtctl** 二进制可执行文件：

```
$ chmod +x <virtctl-file-name>
```

   b. 将 **virtctl** 二进制文件移到 PATH 的目录中。
      i. 要查看路径，请运行：

```
$ echo $PATH
```

6. 对于 Windows 用户：
   a. 进入解压的目录中，双击 **virtctl** 可执行文件来安装客户端。

4.4.2. 启用 OpenShift Virtualization 仓库

红帽为 Red Hat Enterprise Linux 8 和 Red Hat Enterprise Linux 7 提供 OpenShift Virtualization 仓库：

- Red Hat Enterprise Linux 8 软件仓库：**cnv-4.8-for-rhel-8-x86_64-rpms**
- Red Hat Enterprise Linux 7 软件仓库：**rhel-7-server-cnv-4.8-rpms**

在 **subscription-manager** 中启用存储库的过程与在两个平台中启用的过程相同。

流程

- 使用以下命令为您的系统启用适当的 OpenShift virtualization 仓库：

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

4.4.3. 安装 virtctl 客户端

从 **kubevirt-virtctl** 软件包安装 **virtctl** 客户端。

流程

- 安装 **kubevirt-virtctl** 软件包：

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

4.4.4. 其他资源

- 使用 CLI 工具用于 OpenShift Virtualization。

4.5. 使用 WEB 控制台卸载 OPENSHIFT VIRTUALIZATION

您可以使用 OpenShift Container Platform Web 控制台卸载 OpenShift Virtualization。

4.5.1. 先决条件

- 已安装 OpenShift Virtualization 4.8。
- 您必须删除所有 虚拟机、虚拟机实例 和 数据卷。

重要

在不删除这些对象的情况下尝试卸载 OpenShift Virtualization 会导致失败。

4.5.2. 删除 OpenShift Virtualization Operator Deployment 自定义资源

要卸载 OpenShift Virtualization，首先需要删除 **OpenShift Virtualization Operator Deployment** 自定义资源。

先决条件

- 创建 **OpenShift Virtualization Operator Deployment** 自定义资源。

流程

1. 在 OpenShift Container Platform web 控制台中，从 **Project** 列表中选择 **openshift-cnv**。
2. 导航到 **Operators → Installed Operators** 页面。
3. 点击 **OpenShift Virtualization**。
4. 点 **OpenShift Virtualization Operator Deployment** 选项卡。
5. 点击包含 **kubevirt-hyperconverged** 自定义资源  行中的 Options 菜单。在展开的菜单中，点击 **Delete HyperConverged Cluster**。
6. 在确认窗口中点击 **Delete**。
7. 进入 **Workloads → Pods** 页面，验证是否只有 Operator pod 正在运行。
8. 在一个终端窗口中，运行以下命令清理剩余的资源：

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

4.5.3. 删除 OpenShift Virtualization 目录订阅

要完成卸载 OpenShift Virtualization，删除 **OpenShift virtualization** 目录订阅。

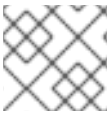
先决条件

- 一个有效的 **OpenShift Virtualization** 目录订阅

流程

流程

1. 导航到 **Operators → OperatorHub** 页面。
2. 搜索 **OpenShift Virtualization** 并选择它。
3. 点击 **Uninstall**。

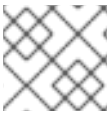


注意

现在可删除 **openshift-cnv** 命名空间。

4.5.4. 使用 web 控制台删除命令空间

您可以使用 OpenShift Container Platform web 控制台删除一个命名空间。



注意

如果您没有删除命名空间的权限，则 **Delete Namespace** 选项不可用。

流程

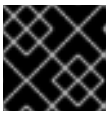
1. 导航至 **Administration → Namespaces**。
2. 在命名空间列表中找到您要删除的命名空间。
3. 在命名空间列表的右侧，从 Options 菜单  中选择 **Delete Namespace**
4. 当 **Delete Namespace** 页打开时，在相关项中输入您要删除的命名空间的名称。
5. 点击 **Delete**。

4.6. 使用 CLI 卸载 OPENSHIFT VIRTUALIZATION

您可以使用 OpenShift Container Platform CLI 卸载 OpenShift Virtualization。

4.6.1. 先决条件

- 已安装 OpenShift Virtualization 4.8。
- 您必须删除所有 虚拟机、虚拟机实例 和 数据卷。



重要

在不删除这些对象的情况下尝试卸载 OpenShift Virtualization 会导致失败。

4.6.2. 删除 OpenShift Virtualization

您可以使用 CLI 删除 OpenShift Virtualization。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 使用具有 **cluster-admin** 权限的账户访问 OpenShift Virtualization 集群。

注意

当使用 CLI 删除 OLM 中的 OpenShift Virtualization Operator 订阅时，集群不会从集群中删除 **ClusterServiceVersion**（CSV）对象。要完全卸载 OpenShift Virtualization，您必须明确删除 CSV。

流程

1. 删除 **HyperConverged** 自定义资源：

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

2. 删除 Operator Lifecycle Manager（OLM）中的 OpenShift Virtualization 订阅：

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

3. 将 OpenShift Virtualization 的集群服务版本（CSV）名称设置为环境变量：

```
$ CSV_NAME=$(oc get csv -n openshift-cnv -o=jsonpath="{.items[0].metadata.name}")
```

4. 通过指定上一步中的 CSV 名称从 OpenShift Virtualization 集群中删除 CSV：

```
$ oc delete csv ${CSV_NAME} -n openshift-cnv
```

当确认消息表示成功删除 CSV 时，则表示 OpenShift Virtualization 被卸载：

输出示例

```
clusterserviceversion.operators.coreos.com "kubevirt-hyperconverged-operator.v4.8.7"
deleted
```

第 5 章 更新 OPENSHIFT VIRTUALIZATION

了解 Operator Lifecycle Manager (OLM) 如何为 OpenShift Virtualization 提供 z-stream 和次要版本更新。

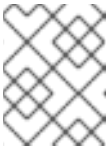
5.1. 关于升级 OPENSHIFT VIRTUALIZATION

5.1.1. OpenShift Virtualization 升级如何工作

- Operator Lifecycle Manager (OLM) 管理 OpenShift Virtualization Operator 的生命周期。Marketplace Operator 在 OpenShift Container Platform 安装过程中部署，使外部 Operator 可供集群使用。
- OLM 为 OpenShift Virtualization 提供 z-stream 和次要版本更新。当您将 OpenShift Container Platform 升级到下一个次版本时，会提供次要版本更新。在没有升级 OpenShift Container Platform 前，您无法将 OpenShift Virtualization 升级到下一个次版本。
- OpenShift Virtualization 订阅使用一个名为 **stable** 的单一更新频道。**stable** 频道确保 OpenShift Virtualization 和 OpenShift Container Platform 版本兼容。
- 如果您的订阅的批准策略被设置为 **Automatic**，则升级过程会在 **stable** 频道中提供新版本的 Operator 时立即启动。强烈建议您使用 **Automatic（自动）** 批准策略来维护可支持的环境。只有在运行对应的 OpenShift Container Platform 版本时，才会支持 OpenShift Virtualization 的每个次要版本。例如，您必须在 OpenShift Container Platform 4.8 上运行 OpenShift Virtualization 4.8。
  - 虽然可以选择 **Manual（手工）** 批准策略，但并不建议这样做，因为它存在集群的支持性和功能风险。使用 **Manual** 批准策略时，您必须手动批准每个待处理的更新。如果 OpenShift Container Platform 和 OpenShift Virtualization 更新不同步，您的集群将无法被支持。
- 更新完成所需时间取决于您的网络连接情况。大部分自动更新可在十五分钟内完成。

5.1.2. OpenShift Virtualization 升级对您的集群有什么影响

- 升级不会中断虚拟机工作负载。
  - 升级过程中不会重启或迁移虚拟机 Pod。如果需要更新 **virt-launcher** Pod，则必须重启或实时迁移该虚拟机。



注意

每个虚拟机均有一个 **virt-launcher** pod，用于运行虚拟机实例。**virt-launcher** pod 运行一个 **libvirt** 实例，用于管理虚拟机进程。

- 升级不会中断网络连接。
- 数据卷及其关联的持久性卷声明会在升级过程中保留。

重要

如果您正在运行无法进行实时迁移的虚拟机，则这些虚拟机可能会阻止 OpenShift Container Platform 集群升级。这包括禁用了 **sriovLiveMigration** 功能门的虚拟机，使用 hostpath 置备程序存储或 SR-IOV 网络接口。

作为临时解决方案，您可以重新配置虚拟机以便在集群升级过程中自动关闭它们。删除 **evictionStrategy: LiveMigrate** 字段，并将 **runStrategy** 字段设置为 **Always**。

5.2. 手动批准待处理的 OPERATOR 更新

如果已安装的 Operator 的订阅被设置为 **Manual**，则当其当前更新频道中发布新更新时，在开始安装前必须手动批准更新。

先决条件

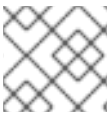
- 之前使用 Operator Lifecycle Manager（OLM）安装的 Operator。

流程

1. 在 OpenShift Container Platform Web 控制台的 **Administrator** 视角中，进入 **Operators → Installed Operators**。
2. 处于待定更新的 Operator 会显示 **Upgrade available** 状态。点您要更新的 Operator 的名称。
3. 点 **Subscription** 标签页。任何需要批准的更新都会在 **Upgrade Status** 旁边显示。例如：它可能会显示 **1 requires approval**。
4. 点 **1 requires approval**，然后点 **Preview Install Plan**。
5. 检查列出可用于更新的资源。在满意后，点 **Approve**。
6. 返回到 **Operators → Installed Operators** 页面，以监控更新的进度。完成后，状态会变为 **Succeeded** 和 **Up to date**。

5.3. 监控升级状态

监控 OpenShift Virtualization 升级状态的最佳方式是查看 ClusterServiceVersion（CSV）**PHASE**。此外您还可在 web 控制台中，或运行此处提供的命令来监控 CSV 状况。



注意

PHASE 和状况值均是基于可用信息的近似值。

先决条件

- 以具有 **cluster-admin** 角色的用户身份登录集群。
- 安装 OpenShift CLI（**oc**）。

流程

1. 运行以下命令：

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

2. 查看输出，检查 **PHASE** 字段。例如：

输出示例

```
VERSION  REPLACES                                   PHASE
4.8.0    kubevirt-hyperconverged-operator.v2.6.5    Installing
4.8.1    kubevirt-hyperconverged-operator.v4.8.0    Replacing
```

3. 可选：运行以下命令来监控所有 OpenShift Virtualization 组件状况的聚合状态：

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

成功升级后会输出以下内容：

输出示例

```
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
```

5.4. 其他资源

- 什么是 Operator?
- Operator Lifecycle Manager 概念和资源
- 集群服务版本（CSV）
- 配置虚拟机驱除策略

第 6 章 为 KUBEVIRT-CONTROLLER 和 VIRT-LAUNCHER 授予额外的安全权限

kubevirt-controller 和 virt-launcher pod 会被授予一些 SELinux 策略和安全上下文约束（除了典型的 pod 拥有者之外）的权限。这些权限可让虚拟机使用 OpenShift Virtualization 功能。

6.1. 为 VIRT-LAUNCHER POD 扩展 SELINUX 策略

virt-launcher Pod 的 **container_t** SELinux 策略会根据以下规则扩展：

- **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))**

这些规则启用以下虚拟化功能：

- 将队列重新标记并把队列附加到其自身的 TUN 插槽，这是支持网络多队列所必需的。多队列可使用网络性能随着 vCPU 数量的增加而扩展网。
- 允许 virt-launcher Pod 将信息写入 sysfs（**/sys**）文件，该文件是启用单根 I/O 虚拟化（SR-IOV）所需要的。
- Read/write **hugetlbfs** 条目，巨页需要它。巨页是通过增加内存页大小来管理大量内存的方法。

6.2. KUBEVIRT-CONTROLLER 服务帐户的其他 OPENSHIFT CONTAINER PLATFORM 安全性上下文约束和 LINUX 功能

Pod 的安全上下文约束（SCC）控制权限。这些权限包括 Pod（容器集合）可以执行的操作以及它们可以访问的资源。您可以使用 SCC 定义 Pod 运行必须满足的一组条件，以便其能被系统接受。

kubevirt-controller 是一个集群控制器，可为集群中的虚拟机创建 virt-launcher Pod。这些 virt-launcher pod 由 **kubevirt-controller** 服务账户授予权限。

6.2.1. kubevirt-controller 服务账户会获得额外的 SCC

kubevirt-controller 服务帐户被授予额外的 SCC 和 Linux 功能，以便能够创建具有适当权限的 virt-launcher Pod。这些扩展权限允许虚拟机利用超出典型 Pod 范围的 OpenShift Virtualization 功能。

kubevirt-controller 服务帐户被授予以下 SCC:

- **scc.AllowHostDirVolumePlugin = true**
  这允许虚拟机使用 hostpath 卷插件。
- **scc.AllowPrivilegedContainer = false**
  可确保 virt-launcher pod 不是作为特权容器运行。
- **scc.AllowedCapabilities = []corev1.Capability{"NET_ADMIN", "NET_RAW", "SYS_NICE"}**
  这可提供以下额外的 Linux 功能 **NET_ADMIN**、**NET_RAW** 和 **SYS_NICE**。

6.2.2. 查看 kubevirt-controller 的 SCC 和 RBAC 定义

您可以使用 **oc** 工具查看 **kubevirt-controller** 的 **SecurityContextConstraints** 定义：

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

您可以使用 **oc** 工具查看 **kubevirt-controller** clusterrole 的 RBAC 定义：

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

6.3. 其他资源

- Red Hat Enterprise Linux Virtualization Tuning and Optimization Guide 对 网络多队列 和 巨页 有更详细的信息。
- **capabilities** man page 包括更多有关 Linux 功能的信息。
- The **sysfs(5)** man page 包括更多 sysfs 的信息。
- 请参阅 OpenShift Container Platform 身份验证指南来了解更多有关安全性上下文约束的信息。

第 7 章 使用 CLI 工具

用于管理集群中资源的两个主要 CLI 工具是：

- OpenShift Virtualization **virtctl** 客户端
- OpenShift Container Platform **oc** 客户端

7.1. 先决条件

- 您必须安装 **virtctl** 客户端。

7.2. VIRTCTL 客户端命令

virtctl 客户端是用于管理 OpenShift Virtualization 资源的命令行实用程序。

要查看 **virtctl** 命令列表，请运行以下命令：

```
$ virtctl help
```

要查看与特定命令一起使用的选项列表，请使用 **-h** 或 **--help** 标记运行该选项。例如：

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

要查看您可以与任何 **virtctl** 命令一起使用的全局命令选项列表，请运行以下命令：

```
$ virtctl options
```

下表包含整个 OpenShift Virtualization 文档中使用的 **virtctl** 命令。

表 7.1. virtctl 客户端命令

命令	描述
virtctl start <vm_name>	启动虚拟机。
virtctl stop <vm_name>	停止虚拟机。
virtctl pause vm|vmi <object_name>	暂停虚拟机或虚拟机实例。机器状态保存在内存中。
virtctl unpause vm|vmi <object_name>	取消暂停虚拟机或虚拟机实例。
virtctl migrate <vm_name>	迁移虚拟机。
virtctl restart <vm_name>	重启虚拟机。
virtctl expose <vm_name>	创建转发虚拟机或虚拟机实例的指定端口的服务，并在节点的指定端口上公开该服务。

命令	描述
virtctl console <vmi_name>	连接至虚拟机实例的串行控制台。
virtctl vnc --kubeconfig=$KUBECONFIG <vmi_name>	打开与虚拟机实例的 VNC（虚拟网络客户端）连接。通过 VNC 访问虚拟机实例的图形控制台，该 VNC 需要本地计算机上的远程查看器。
virtctl vnc --kubeconfig=$KUBECONFIG --proxy-only=true <vmi-name>	显示端口号，并使用任何查看器通过 VNC 连接手动连接到虚拟机实例。
virtctl vnc --kubeconfig=$KUBECONFIG --port=<port-number> <vmi-name>	如果该端口可用，则指定端口号用于在指定端口上运行代理。如果没有指定端口号，代理会在随机端口上运行。
virtctl image-upload dv <datavolume_name> --image-path=</path/to/image> --no-create	将虚拟机镜像上传到已存在的数据卷中。
virtctl image-upload dv <datavolume_name> --size=<datavolume_size> --image-path=</path/to/image>	将虚拟机镜像上传到新数据卷。
virtctl version	显示客户端和服务器版本。
virtctl help	显示 **virtctl** 命令的描述性列表。
virtctl fslist <vmi_name>	返回客户端机器中可用文件系统的完整列表。
virtctl guestosinfo <vmi_name>	返回有关操作系统的客户机代理信息。
virtctl userlist <vmi_name>	返回客户端机器中登录用户的完整列表。

7.3. OPENSHIFT CONTAINER PLATFORM 客户端命令

OpenShift Container Platform **oc** 客户端是一个用于管理 OpenShift Container Platform 资源的命令行实用程序，包括 **VirtualMachine**（**vm**）和 **VirtualMachineInstance**（**vmi**）对象类型。

注意

您可以使用 **-n <namespace>** 指定一个不同的项目。

表 7.2. oc 命令

命令	描述
oc login -u <user_name>	以 **<user_name>** 身份登录 OpenShift Container Platform 集群。
oc get <object_type>	显示当前项目中指定对象类型的对象列表。
oc describe <object_type> <resource_name>	显示当前项目中指定资源的详情。
oc create -f <object_config>	从文件名称或 stdin 在当前项目中创建资源。
oc edit <object_type> <resource_name>	编辑当前项目中的资源。
oc delete <object_type> <resource_name>	删除当前项目中的资源。

有关 **oc** 客户端命令的更全面信息，请参阅 OpenShift Container Platform CLI 工具 文档。

第 8 章 虚拟机

8.1. 创建虚拟机

使用以下其中一个流程来创建虚拟机：

- 快速入门指南
- 运行向导
- 使用虚拟机向导来粘贴预先配置的 YAML 文件
- 使用 CLI
- 使用虚拟机向导来导入 VMware 虚拟机或模板

警告

不要在 **openshift-*** 命名空间中创建虚拟机 。相反，创建一个新命名空间或使用没有 **openshift** 前缀的现有命名空间。

从 web 控制台创建虚拟机时，请选择配置了引导源的虚拟机模板。具有引导源的虚拟机模板标记为 **Available boot source**，或者它们显示自定义标签文本。使用有可用引导源的模板可促进创建虚拟机的过程。

没有引导源的模板被标记为 **Boot source required**。如果完成了向虚拟机中添加引导源的步骤，您可以使用这些模板。

8.1.1. 使用快速入门创建虚拟机

web 控制台为创建虚拟机提供指导指导快速入门。您可以通过在 **Administrator** 视角中选择 Help 菜单来查看 Quick Starts 目录来访问 Quick Starts 目录。当您点快速入门标题并开始使用时，系统会帮助您完成这个过程。

快速入门中的任务以选择红帽模板开始。然后，您可以添加一个引导源并导入操作系统镜像。最后，您可以保存自定义模板，并使用它来创建虚拟机。

先决条件

- 访问您可以下载操作系统镜像的 URL 链接的网站。

流程

1. 在 web 控制台中，从 Help 菜单中选择 **Quick Starts**。
2. 点 Quick Starts 目录里的一个标题。例如：**Creating a Red Hat Linux Enterprise Linux virtual machine**。
3. 按照教程中的说明，完成导入操作系统镜像并创建虚拟机的任务。**Virtual Machines** 标签显示虚拟机。

8.1.2. 运行虚拟机向导来创建虚拟机

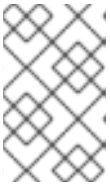
web 控制台带有一个向导，指导您完成选择虚拟机模板和创建虚拟机的过程。红帽虚拟机模板会预先配置操作系统镜像、操作系统的默认设置、flavor（CPU 和内存）以及工作负载类型(server)。当模板配置为使用引导源配置时，会使用自定义标签文本或者默认标签文本 **Available boot source** 进行标记。这些模板可用于创建虚拟机。

您可以从预配置的模板列表中选择模板，查看设置并使用**Create virtual machine from template**创建一个虚拟机。如果您选择自定义虚拟机，向导会帮助您完成 **General**、**Networking**、**Storage**、**Advanced** 和 **Review** 步骤。向导显示的所有必填字段均标有 *。

创建网络接口控制器 (NIC) 和存储磁盘，并将它们附加到虚拟机。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 在 **Virtual Machines** 选项卡或 **Templates** 选项卡中点 **Create** 并选择 **Virtual Machine with Wizard**。
3. 选择一个使用引导源配置的模板。
4. 点 **Next** 进入 **Review and create** 步骤。
5. 如果您不想现在就启动虚拟机，清除 **Start this virtual machine after creation**选择。
6. 点 **Create virtual machine** 并退出向导或继续向导以自定义虚拟机。
7. 点 **Customize virtual machine** 进入 **General** 步骤。
   a. 可选：编辑 **Name** 字段，为虚拟机指定自定义名称。
   b. 可选：在 **Description** 字段中添加描述信息。
8. 点 **Next** 进入 **Networking** 步骤。默认附加 **nic0** NIC。
   a. 可选：点 **Add Network Interface** 来创建额外 NIC。
   b. Optional:您可以通过点 Options 菜单  并选择 **Delete** 来删除任何或所有 NIC。虚拟机无需附加 NIC 也可创建。您可以在创建虚拟机后创建 NIC。
9. 点 **Next** 进入 **Storage** 步骤。
   a. 可选：点击 **Add Disk** 创建额外磁盘。可通过点 Options 菜单  并选择 **Delete** 来删除这些磁盘。
   b. 可选：点击 Options 菜单  来编辑磁盘并保存您的更改。
10. 单击 **Next** 以进入 **Advanced** 步骤，以查看 **Cloud-init** 的详细信息并配置 SSH 访问。



注意

使用 cloud-init 或向导中的自定义脚本，静态注入 SSH 密钥。这使您可以安全、远程管理虚拟机以及管理和传输信息。强烈建议您使用这个步骤来保护您的虚拟机。

11. 点 **Next** 进入 **Review** 步骤并查看虚拟机的设置。
12. 点 **Create Virtual Machine**。
13. 点 **See virtual machine details** 查看此虚拟机的 **Overview**。
虚拟机在 **Virtual Machines** 标签页中列出。

运行 web 控制台向导时，请参考虚拟机向导字段部分。

8.1.2.1. 虚拟机向导字段

名称	参数	描述
名称		名称可包含小写字母 (**a-z**)、数字 (**0-9**) 和连字符 (**-**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格、句点 (**.**) 或特殊字符。
描述		可选的描述字段。
操作系统		模板中为虚拟机选择的操作系统。从模板创建虚拟机时，您无法编辑此字段。
引导源	通过 URL 导入（创建 PVC）	从 HTTP 或 HTTPS 端点提供的镜像导入内容。示例：包含操作系统镜像的网页中的 URL 链接。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有持久性卷声明并克隆它。
	通过 Registry 导入（创建 PVC）	从可通过集群访问的注册表中的可启动操作系统容器置备虚拟机。示例：***kubevirt/cirros-registry-disk-demo***。
	PXE（网络引导 - 添加网络接口）	从网络的服务器引导操作系统。需要一个 PXE 可引导网络附加定义。
持久性卷声明项目		用于克隆 PVC 的项目名称。
持久性卷声明名称		如果您要克隆现有的 PVC，则应用于此虚拟机模板的 PVC 名称。

名称	参数	描述
将它作为光盘引导源挂载		CD-ROM 需要额外的磁盘来安装操作系统。选择添加磁盘的选择框并稍后进行自定义。
Flavor	tiny、small、Medium、Large、Custom	根据与该模板关联的操作系统，在虚拟机模板中预设具有预定义值的 CPU 和内存量。 如果选择了默认模板，您可以使用自定义值覆盖模板中的 **cpus** 和 **memsize** 的值，以创建自定义模板。另外，您可以通过修改 **Workloads → Virtualization** 页面上的 **Details** 选项卡中的 **cpus** 和 **memsize** 值来创建自定义模板。
工作负载类型  **注意** 如果您选择了不正确的 **Workload Type**，则可能会出现性能或资源利用率问题（如缓慢的 UI）。	Desktop	用于桌面的虚拟机配置。适用于小型工作环境。建议与 Web 控制台搭配使用。
	Server	在性能和广泛的服务器工作负载兼容性方面具有最佳平衡。
	高性能	针对高性能负载进行了优化的虚拟机配置。
创建后启动此虚拟机。		默认选择这个复选框并在创建后启动虚拟机。如果您不希望虚拟机在创建时启动，请清除该复选框。

8.1.2.2. 网络字段

名称	描述
名称	网络接口控制器的名称。
model	指明网络接口控制器的型号。支持的值有 **e1000e** 和 **virtio**。
网络	可用网络附加定义的列表。
类型	可用绑定方法列表。对于默认的 pod 网络，**masquerade** 是唯一推荐的绑定方法。对于辅助网络，请使用 **bridge** 绑定方法。非默认网络不支持 **masquerade** 绑定方法。如果您配置了一个 **SR-IOV** 网络设备并在命名空间中定义那个网络，请选择 SR-IOV。

名称	描述
MAC 地址	网络接口控制器的 MAC 地址。如果没有指定 MAC 地址，则会自动分配一个。

8.1.2.3. 存储字段

名称	选择	描述
Source	空白（创建 PVC）	创建一个空磁盘。
	通过 URL 导入（创建 PVC）	通过 URL（HTTP 或 HTTPS 端点）导入内容。
	使用现有的 PVC	使用集群中已可用的 PVC。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有 PVC 并克隆它。
	通过 Registry 导入（创建 PVC）	通过容器 registry 导入内容。
	容器（临时）	从集群可以访问的 registry 中的容器上传内容。容器磁盘应只用于只读文件系统，如 CD-ROM 或临时虚拟机。
名称		磁盘的名称。名称可包含小写字母 (**a-z**)、数字 (**0-9**)、连字符 (**-**) 和句点 (**.**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格或特殊字符。
Size		GiB 中磁盘的大小。
类型		磁盘类型。示例：磁盘或光盘
Interface		磁盘设备的类型。支持的接口包括 **virtIO**、**SATA** 和 **SCSI**。
Storage class		用于创建磁盘的存储类。
Advanced → Volume Mode		定义持久性卷是否使用格式化的文件系统或原始块状态。默认为 **Filesystem**。

名称	选择	描述
Advanced → Access Mode		持久性卷访问模式。支持的访问模式有 **Single User（RWO）**、**Shared Access（RWX）** 和 **Read Only（ROX）**。

高级存储设置
以下高级存储设置可用于 **空白**、**从 URL 导入** 和 **克隆现有的 PVC** 磁盘。所有参数都是可选的。如果没有指定这些参数，系统将使用 **kubevirt-storage-class-defaults** 配置映射中的默认值。

名称	参数	描述
卷模式	Filesystem	在基于文件系统的卷中保存虚拟磁盘。
	Block	直接将虚拟磁盘存储在块卷中。只有底层存储支持时才使用 **Block**。
访问模式	Single User (RWO)	这个卷可以被一个单一的节点以 read/write 的形式挂载。
	Shared Access (RWX)	卷可以被多个节点以读写模式挂载。  **注意** 对于一些功能（如虚拟机在节点间实时迁移）需要这个权限。
	Read Only (ROX)	卷可以被多个节点以只读形式挂载。

8.1.2.4. Cloud-init 字段

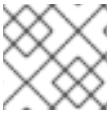
名称	描述
Hostname	为虚拟机设置特定主机名。
授权 SSH 密钥	复制到虚拟机上 *~/.ssh/authorized_keys* 的用户公钥。
自定义脚本	将其他选项替换为您粘贴自定义 cloud-init 脚本的字段。

要配置存储类默认设置，请使用存储配置集。如需更多信息，请参阅自定义存储配置集。

8.1.2.5. 粘贴至预先配置的 YAML 文件中以创建虚拟机

通过写入或粘贴 YAML 配置文件来创建虚拟机。每当您打开 YAML 编辑屏幕，默认会提供一个有效的 **example** 虚拟机配置。

如果您点击 **Create** 时 YAML 配置无效，则错误消息会指示出错的参数。一次仅显示一个错误。



注意

编辑时离开 YAML 屏幕会取消您对配置做出的任何更改。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 点 **Create** 并选择 **Virtual Machine With YAML**。
4. 在可编辑窗口写入或粘贴您的虚拟机配置。
    a. 或者，使用 YAML 屏幕中默认提供的 **example** 虚拟机。
5. 可选：点 **Download** 以下载当前状态下的 YAML 配置文件。
6. 点击 **Create** 以创建虚拟机。

虚拟机在 **Virtual Machines** 标签页中列出。

8.1.3. 使用 CLI 创建虚拟机

您可以从 **virtualMachine** 清单创建虚拟机。

流程

1. 编辑虚拟机的 **VirtualMachine** 清单。例如，以下清单配置 Red Hat Enterprise Linux(RHEL)虚拟机：

例 8.1. RHEL 虚拟机的清单示例

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    app: <vm_name> 1
  name: <vm_name>
spec:
  dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1beta1
    kind: DataVolume
    metadata:
      name: <vm_name>
    spec:
      sourceRef:
        kind: DataSource
        name: rhel9
        namespace: openshift-virtualization-os-images
      storage:
```

```
      resources:
        requests:
          storage: 30Gi
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/domain: <vm_name>
    spec:
      domain:
        cpu:
          cores: 1
          sockets: 2
          threads: 1
        devices:
          disks:
          - disk:
              bus: virtio
            name: rootdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
          interfaces:
          - masquerade: {}
            name: default
          rng: {}
        features:
          smm:
            enabled: true
        firmware:
          bootloader:
            efi: {}
        resources:
          requests:
            memory: 8Gi
      evictionStrategy: LiveMigrate
      networks:
      - name: default
        pod: {}
      volumes:
      - dataVolume:
          name: <vm_name>
        name: rootdisk
      - cloudInitNoCloud:
          userData: |-
            #cloud-config
            user: cloud-user
            password: '<password>' 2
            chpasswd: { expire: False }
        name: cloudinitdisk
```

1 指定虚拟机的名称。

2 指定 cloud-user 的密码。

2. 使用清单文件创建虚拟机：

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

3. 可选：启动虚拟机：

```
$ virtctl start <vm_name>
```

8.1.4. 虚拟机存储卷类型

存储卷类型	描述
ephemeral	将网络卷用作只读后备存储的本地写时复制 (COW) 镜像。后备卷必须为 **PersistentVolumeClaim**。当虚拟机启动并在本地存储所有写入数据时，便会创建临时镜像。当虚拟机停止、重启或删除时，便会丢弃临时镜像。其底层的卷 (PVC) 不会以任何方式发生变化。
persistentVolumeClaim	将可用 PV 附加到虚拟机。附加 PV 可确保虚拟机数据在会话之间保持。 将现有虚拟机导入到 OpenShift Container Platform 中的建议方法是，使用 CDI 将现有虚拟机磁盘导入到 PVC 中，然后将 PVC 附加到虚拟机实例。在 PVC 中使用磁盘需要满足一些要求。
dataVolume	通过导入、克隆或上传操作来管理虚拟机磁盘的准备过程，以此在 **persistentVolumeClaim** 磁盘类型基础上构建数据卷。使用此卷类型的虚拟机可保证在卷就绪前不会启动。 指定 **type: dataVolume** 或 **type: ""**。如果您为 **type** 指定任何其他值，如 **persistentVolumeClaim**，则会显示警告信息，虚拟机也不会启动。
cloudInitNoCloud	附加包含所引用的 cloud-init NoCloud 数据源的磁盘，从而向虚拟机提供用户数据和元数据。虚拟机磁盘内部需要安装 cloud-init。

存储卷类型	描述
containerDisk	引用容器镜像 registry 中存储的镜像，如虚拟机磁盘。镜像从 registry 中拉取，并在虚拟机启动时作为磁盘附加到虚拟机。 **containerDisk** 卷不仅限于一个虚拟机，对于要创建大量无需持久性存储的虚拟机克隆来说也非常有用。 容器镜像 registry 仅支持 RAW 和 QCOW2 格式的磁盘类型。建议使用 QCOW2 格式以减小镜像的大小。  **注意** **containerDisk** 卷是临时的。将在虚拟机停止、重启或删除时丢弃。**containerDisk** 卷对于只读文件系统（如 CD-ROM）或可处理的虚拟机很有用。
emptyDisk	创建额外的稀疏 QCOW2 磁盘，与虚拟机接口的生命周期相关联。当虚拟机中的客户端初始化重启后，数据保留下来，但当虚拟机停止或从 web 控制台重启时，数据将被丢弃。空磁盘用于存储应用程序依赖项和数据，否则这些依赖项和数据会超出临时磁盘有限的临时文件系统。 此外还必须提供磁盘**容量**大小。

8.1.5. 关于虚拟机的 RunStrategies

虚拟机的 **RunStrategy** 会根据一系列条件，决定虚拟机实例（VMI）的行为。**spec.runStrategy** 设置存在于虚拟机配置过程中，作为 **spec.running** 设置的替代方案。**spec.runStrategy** 设置为创建和管理 VMI 提供了更大的灵活性。而 **spec.running** 设置只能有 **true** 或 **false** 响应。但是，这两种设置是相互排斥的。只能同时使用 **spec.running** 或 **spec.runStrategy** 之一。如果两者都存在，则会出现错误。

有四个定义的 RunStrategies。

Always

在创建虚拟机时，始终会存在 VMI。如果因为任何原因造成原始的 VMI 停止运行，则会创建一个新的 VMI，这与 **spec.running: true** 的行为相同。

RerunOnFailure

如果上一个实例因为错误而失败，则会重新创建一个 VMI。如果虚拟机成功停止（例如虚拟机正常关机），则不会重新创建实例。

Manual

start、**stop** 和 **restart** virtctl 客户端命令可以被用来控制 VMI 的状态。

Halted

创建虚拟机时没有 VMI，这与 **spec.running: false** 的行为相同。

start、**stop** 和 **restart** virtctl 命令的不同组合会影响到使用哪个 **RunStrategy**。

下表是虚拟机从不同状态过渡的列表。第一栏显示了 VM 的初始 **RunStrategy**。每个额外的栏都显示一个 virtctl 命令以及在运行该命令后的新的 **RunStrategy**。

初始 RunStrategy	开始	停止	重启
Always	-	Halted	Always
RerunOnFailure	-	Halted	RerunOnFailure
Manual	Manual	Manual	Manual
Halted	Always	-	-

注意

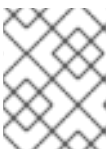
在使用安装程序置备的基础架构安装的 OpenShift Virtualization 集群中，当节点的 MachineHealthCheck 失败且集群不可用时，,带有 **Always** 或 **RerunOnFailure** 的 RunStrategy 的虚拟机会被重新调度到一个新的节点上。

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  RunStrategy: Always (1)
  template:
...
```

1 VMI 的当前 **RunStrategy** 设置。

8.1.6. 其他资源

- KubeVirt v0.41.0 API Reference 中的 **VirtualMachineSpec** 定义为虚拟机规格的参数和等级提供更宽松的上下文。



注意

KubeVirt API Reference 是上游项目参考，可能包含 OpenShift Virtualization 不支持的参数。

- 在将容器磁盘作为 **containerDisk** 卷添加到虚拟机之前，需要先准备容器磁盘。
- 有关部署和启用机器健康检查的详情，请参阅部署机器健康检查。
- 有关安装程序置备的基础架构的详情，请参阅安装程序置备的基础架构概述。
- 如需了解更多与 SR-IOV Network Operator 相关的信息，请参阅配置 SR-IOV Network Operator。

8.2. 编辑虚拟机

您可以使用 web 控制台中的 YAML 编辑器或命令行上的 OpenShift CLI 来更新虚拟机配置。您还可以更新 **Virtual Machine Details** 屏幕中的参数子集。

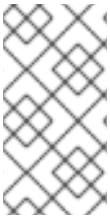
8.2.1. 在 web 控制台中编辑虚拟机

点相关字段旁的铅笔图标来编辑 web 控制台中虚拟机的选择值。可使用 CLI 编辑其他值。

对于预先配置的红帽模板和自定义虚拟机模板，可编辑标签和注解。所有其他值只适用于用户使用红帽模板或 **Create Virtual Machine Template 向导**创建的自定义虚拟机模板。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机。
4. 点 **Details** 标签页。
5. 点击铅笔图标使该字段可编辑。
6. 进行相关的更改并点击 **Save**。



注意

如果虚拟机正在运行，对 **Boot Order** 或 **Flavor** 的更改在重启虚拟机后才会生效。

您可以点击相关字段右侧的 **View Pending Changes** 来查看待处理的更改。页面顶部的 **Pending Changes** 标题显示虚拟机重启时将应用的所有更改列表。

8.2.2. 使用 web 控制台编辑虚拟机 YAML 配置

您可以在 web 控制台中编辑虚拟机的 YAML 配置。某些参数无法修改。如果在有无效配置时点 **Save**，则会出现一个错误消息指示无法更改的参数。

如果在虚拟机运行时编辑 YAML 配置，则更改在重启虚拟机后才会生效。

注意

编辑时离开 YAML 屏幕会取消您对配置做出的任何更改。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 选择虚拟机。
3. 点击 **YAML** 选项卡以显示可编辑的配置。
4. （可选）：您可点击 **Download**，在本地下载当前状态的 YAML 文件。
5. 编辑该文件并点击 **Save**。

确认消息显示修改已成功，其中包含对象的更新版本号。

8.2.3. 使用 CLI 编辑虚拟机 YAML 配置

使用这个步骤，通过 CLI 编辑虚拟机 YAML 配置。

先决条件

- 已使用 YAML 对象配置文件配置了虚拟机。
- 已安装 **oc** CLI。

流程

1. 运行以下命令以更新虚拟机配置：

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

2. 打开对象配置。
3. 编辑 YAML。
4. 如果要编辑正在运行的虚拟机，您需要执行以下任一操作：
   - 重启虚拟机。
   - 运行以下命令使新配置生效：

```
$ oc apply <object_type> <object_ID>
```

8.2.4. 将虚拟磁盘添加到虚拟机

使用这个流程在虚拟机中添加虚拟磁盘。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Disks** 选项卡。
5. 在 **Add Disk** 窗口中，指定 **Source**、**Name**、**Size**、**Type**、**Interface** 和 **Storage Class**。
   a. 可选：在 **Advanced** 列表中，为虚拟磁盘指定 **Volume Mode** 和 **Access Mode**。如果没有指定这些参数，系统将使用 **kubevirt-storage-class-defaults** 配置映射中的默认值。
6. 点 **Add**。

注意

如果虚拟机正在运行，新磁盘处于 **pending restart** 状态，且不会在重启虚拟机前附加。

页面顶部的 **Pending Changes** 标题显示虚拟机重启时将应用的所有更改列表。

要配置存储类默认设置，请使用存储配置集。如需更多信息，请参阅自定义存储配置集。

8.2.4.1. 存储字段

名称	选择	描述
Source	空白（创建 PVC）	创建一个空磁盘。
	通过 URL 导入（创建 PVC）	通过 URL（HTTP 或 HTTPS 端点）导入内容。
	使用现有的 PVC	使用集群中已可用的 PVC。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有 PVC 并克隆它。
	通过 Registry 导入（创建 PVC）	通过容器 registry 导入内容。
	容器（临时）	从集群可以访问的 registry 中的容器上传内容。容器磁盘应只用于只读文件系统，如 CD-ROM 或临时虚拟机。
名称		磁盘的名称。名称可包含小写字母 (**a-z**)、数字 (**0-9**)、连字符 (**-**) 和句点 (**.**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格或特殊字符。
Size		GiB 中磁盘的大小。
类型		磁盘类型。示例：磁盘或光盘
Interface		磁盘设备的类型。支持的接口包括 **virtIO**、**SATA** 和 **SCSI**。
Storage class		用于创建磁盘的存储类。
Advanced → Volume Mode		定义持久性卷是否使用格式化的文件系统或原始块状态。默认为 **Filesystem**。

名称	选择	描述
Advanced → Access Mode		持久性卷访问模式。支持的访问模式有 **Single User（RWO）**、**Shared Access（RWX）** 和 **Read Only（ROX）**。

高级存储设置

以下高级存储设置可用于 **空白**、**从 URL 导入** 和 **克隆现有的 PVC** 磁盘。所有参数都是可选的。如果没有指定这些参数，系统将使用 **kubevirt-storage-class-defaults** 配置映射中的默认值。

名称	参数	描述
卷模式	Filesystem	在基于文件系统的卷中保存虚拟磁盘。
	Block	直接将虚拟磁盘存储在块卷中。只有底层存储支持时才使用 **Block**。
访问模式	Single User (RWO)	这个卷可以被一个单一的节点以 read/write 的形式挂载。
	Shared Access (RWX)	卷可以被多个节点以读写模式挂载。  **注意** 对于一些功能（如虚拟机在节点间实时迁移）需要这个权限。
	Read Only (ROX)	卷可以被多个节点以只读形式挂载。

8.2.5. 将网络接口添加到虚拟机

将网络接口添加到虚拟机.

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Network Interfaces** 选项卡。
5. 点击 **Add Network Interface**。
6. 在 **Add Network Interface** 窗口中，指定网络接口的 **Name**、**Model**、**Network**、**Type** 和 **MAC Address**。

7. 点 **Add**。

注意

如果虚拟机正在运行，新的网络接口处于 **pending restart** 状态，且更改在重启虚拟机后才会生效。

页面顶部的 **Pending Changes** 标题显示虚拟机重启时将应用的所有更改列表。

8.2.5.1. 网络字段

名称	描述
名称	网络接口控制器的名称。
model	指明网络接口控制器的型号。支持的值有 **e1000e** 和 **virtio**。
网络	可用网络附加定义的列表。
类型	可用绑定方法列表。对于默认的 pod 网络，**masquerade** 是唯一推荐的绑定方法。对于辅助网络，请使用 **bridge** 绑定方法。非默认网络不支持 **masquerade** 绑定方法。如果您配置了一个 **SR-IOV** 网络设备并在命名空间中定义那个网络，请选择 SR-IOV。
MAC 地址	网络接口控制器的 MAC 地址。如果没有指定 MAC 地址，则会自动分配一个。

8.2.6. 为虚拟机编辑 CD-ROM

使用以下步骤为虚拟机编辑 CD-ROM。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Disks** 选项卡。
5. 点您要编辑的 CD-ROM 的 Options 菜单  ，然后选择 **Edit**。
6. 在 **Edit CD-ROM** 窗口中，编辑字段： **Source**、**Persistent Volume Claim**、**Name**、**Type** 和 **Interface**。
7. 点 **Save**。

8.3. 编辑引导顺序

您可以使用 Web 控制台或 CLI 更新引导顺序列表的值。

通过 **Virtual Machine Overview** 页面中的 **Boot Order**，您可以：

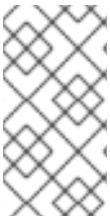
- 选择磁盘或网络接口控制器 (NIC) 并将其添加到引导顺序列表中。
- 编辑引导顺序列表中磁盘或 NIC 的顺序。
- 从引导顺序列表中移除磁盘或者 NIC，然后将其返回到可引导源清单。

8.3.1. 向 web 控制台的引导顺序列表中添加项目

使用 web 控制台将项目添加到引导顺序列表中。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Details** 标签页。
5. 点击位于 **Boot Order** 右侧的铅笔图标。如果 YAML 配置不存在，或者是首次创建引导顺序列表时，会显示以下消息: **No resource selected.虚拟机会根据在 YAML 文件中的顺序从磁盘引导。**
6. 点 **Add Source**，为虚拟机选择一个可引导磁盘或网络接口控制器 (NIC)。
7. 在引导顺序列表中添加附加磁盘或者 NIC。
8. 点 **Save**。



注意

如果虚拟机正在运行，在重启虚拟机后对 **Boot Order** 的更改不会生效。

您可以点 **Boot Order** 字段右侧的 **View Pending Changes** 查看待处理的修改。页面顶部的 **Pending Changes** 标题显示虚拟机重启时将应用的所有更改列表。

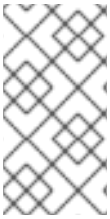
8.3.2. 在 web 控制台中编辑引导顺序列表

在 web 控制台中编辑引导顺序列表。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Details** 标签页。

5. 点击位于 **Boot Order** 右侧的铅笔图标。
6. 选择适当的方法来移动引导顺序列表中的项目：
   - 如果您没有使用屏幕阅读器，请在您想要移动的项目旁的箭头图标上切换，拖动或下移项目，然后将其放到您选择的位置。
   - 如果您使用屏幕阅读器，请按上箭头或者下箭头键移动引导顺序列表中的项目。然后，按 **Tab** 键将项目放到您选择的位置。
7. 点 **Save**。



注意

如果虚拟机正在运行，对引导顺序列表的更改将在重启虚拟机后才会生效。

您可以点 **Boot Order** 字段右侧的 **View Pending Changes** 查看待处理的修改。页面顶部的 **Pending Changes** 标题显示虚拟机重启时将应用的所有更改列表。

8.3.3. 在 YAML 配置文件中编辑引导顺序列表

使用 CLI 编辑 YAML 配置文件中的引导顺序列表。

流程

1. 运行以下命令为虚拟机打开 YAML 配置文件：

```
$ oc edit vm example
```

2. 编辑 YAML 文件并修改与磁盘或网络接口控制器 (NIC) 关联的引导顺序值。例如：

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

(1) 为磁盘指定的引导顺序值。

(2) 为网络接口控制器指定的引导顺序值。

3. 保存 YAML 文件。

4. 点 **reload the content**，使 YAML 文件中更新的引导顺序值应用到 web 控制台的引导顺序列表中。

8.3.4. 从 web 控制台中的引导顺序列表中删除项目

使用 Web 控制台从引导顺序列表中移除项目。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Details** 标签页。
5. 点击位于 **Boot Order** 右侧的铅笔图标。
6. 点击项 ⛔ 旁边的 **Remove** 图标。该项目从引导顺序列表中删除，可用引导源列表的内容被保存。如果您从引导顺序列表中删除所有项目，则会显示以下消息: **No resource selected.虚拟机会根据在 YAML 文件中的顺序从磁盘引导。**

注意

如果虚拟机正在运行，在重启虚拟机后对 **Boot Order** 的更改不会生效。

您可以点 **Boot Order** 字段右侧的 **View Pending Changes** 查看待处理的修改。页面顶部的 **Pending Changes** 标题显示虚拟机重启时将应用的所有更改列表。

8.4. 删除虚拟机

您可从 web 控制台或使用 **oc** 命令行删除虚拟机。

8.4.1. 使用 web 控制台删除虚拟机

删除虚拟机会将其从集群中永久移除。

注意

当您删除虚拟机时，其使用的数据卷会被自动删除。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 点您要删除的虚拟机 ⋮ 的 Options 菜单，然后选择 **Delete Virtual Machine**。
   - 或者，点击虚拟机名称，打开 **Virtual Machine Overview** 屏幕，然后点击 **Actions → Delete Virtual Machine**。

4. 在确认弹出窗口中，点击 **Delete** 永久删除虚拟机。

8.4.2. 使用 CLI 删除虚拟机

您可以使用 **oc** 命令行界面（CLI）删除虚拟机。您可以使用 **oc** 对多个虚拟机执行操作。

注意

当您删除虚拟机时，其使用的数据卷会被自动删除。

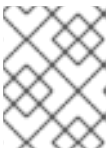
先决条件

- 找到要删除的虚拟机名称。

流程

- 运行以下命令以删除虚拟机：

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



注意

此命令只删除当前项目中存在的对象。如果您要删除其他项目或命名空间中的对象，请使用 **-n <project_name>** 选项。

8.5. 管理虚拟机实例

如果您在 OpenShift Virtualization 环境外创建独立虚拟机实例（VMI），您可以使用 web 控制台或命令行界面（CLI）管理它们。

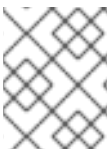
8.5.1. 关于虚拟机实例

虚拟机实例（VMI）代表正在运行的虚拟机(VM)。当某个 VMI 属于某个虚拟机或者其他对象，您可通过 web 控制台中的拥有者或使用 **oc** 命令行界面（CLI）来管理它。

通过自动化或其他 CLI 的方法使用脚本创建并启动独立 VMI。在您的环境中，您可能会在 OpenShift Virtualization 环境之外开发并启动的独立 VMI。您可以使用 CLI 继续管理这些独立的 VMI。您还可以将 Web 控制台用于与独立 VMI 关联的特定任务：

- 列出独立 VMI 及其详情。
- 编辑独立 VMI 的标签和注解。
- 删除独立 VMI。

当删除虚拟机时，相关的 VMI 会被自动删除。您直接删除一个独立的 VMI，因为它不归 VM 或其他对象所有。



注意

在卸载 OpenShift Virtualization 前，使用 CLI 或 Web 控制台列出并查看独立 VMI。然后，删除所有未完成的 VMI。

8.5.2. 使用 CLI 列出所有虚拟机实例

您可以使用 **oc** 命令行界面（CLI）列出集群中的所有虚拟机实例（VMI），包括独立 VMI 和虚拟机拥有的实例。

流程

- 运行以下命令列出所有 VMI：

```
$ oc get vmis
```

8.5.3. 使用 web 控制台列出独立虚拟机实例

使用 web 控制台，您可以列出并查看集群中不属于虚拟机（VM）的独立虚拟机实例（VMI）。

注意

受 VM 或其他对象拥有的 VMI 不会被显示在 web 控制台中。web 控制台仅显示独立 VMI。如果要列出集群中的所有 VMI，则必须使用 CLI。

流程

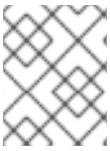
- 从侧边菜单中点 **Workloads → Virtualization**。显示虚拟机和独立 VMI 列表。您可以通过在虚拟机实例名称旁显示的黑色徽标识别独立 VMI。

8.5.4. 使用 web 控制台编辑独立虚拟机实例

您可以使用 web 控制台编辑独立虚拟机实例（VMI）的注解和标签。独立 VMI 的 **Details** 页面中显示的其它项目不可编辑。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。此时会显示虚拟机（VM）和独立的 VMI 列表。
2. 点击独立 VMI 的名称打开 **Virtual Machine Instance Overview** 界面。
3. 点 **Details** 标签页。
4. 点击位于 **Annotations** 右侧的铅笔图标。
5. 进行相关的更改并点击 **Save**。



注意

要编辑独立 VMI 的标签，请点击 **Actions** 并选择 **Edit Labels**。进行相关的更改并点击 **Save**。

8.5.5. 使用 CLI 删除独立虚拟机实例

您可以使用 **oc** CLI 删除独立虚拟机实例。

先决条件

- 找出要删除的 VMI 的名称。

流程

- 运行以下命令来创建 VMI：

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

8.5.6. 使用 web 控制台删除独立虚拟机实例

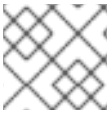
从 web 控制台删除独立虚拟机实例（VMI）。

流程

1. 在 OpenShift Container Platform web 控制台中，从侧面菜单中点 **Workloads → Virtualization**。
2. 点击待删除的独立虚拟机实例（VMI）的 ⋮ 按钮，然后选择 **Delete Virtual Machine Instance**。
   - 或者，点击独立 VMI 的名称。**Virtual Machine Instance Overview** 页会显示。
3. 选择 **Actions → Delete Virtual Machine Instance**。
4. 在弹出的确认窗口中点击 **Delete** 永久删除独立的 VMI。

8.6. 控制虚拟机状态

您可从 web 控制台来停止、启动和重启虚拟机。



注意

要从命令行界面（CLI）控制虚拟机，请使用 **virtctl** 客户端。

8.6.1. 启动虚拟机

您可从 web 控制台启动虚拟机。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 找到包含要启动的虚拟机的行。
4. 导航到适合您的用例的菜单：
   - 要保留此页面（您可以在其中对多个虚拟机执行操作）：
     a. 点击  位于行右边的 Options 菜单。
   - 在启动虚拟机前，要查看有关所选虚拟机的综合信息：

a. 点击虚拟机名称访问 **Virtual Machine Overview** 页面。

b. 点击 **Actions**。

5. 选择 **Start Virtual Machine**。

6. 在确认窗口中，点击 **Start** 启动虚拟机。

注意

首次启动从 **URL** 源置备的虚拟机时，当 OpenShift Virtualization 从 URL 端点导入容器时，虚拟机将处于 **Importing** 状态。根据镜像大小，该过程可能需要几分钟时间。

8.6.2. 重启虚拟机

您可从 web 控制台重启正在运行的虚拟机。

重要

为了避免错误，不要重启状态为 **Importing** 的虚拟机。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 找到包含要启动的虚拟机的行。
4. 导航到适合您的用例的菜单：
   - 要保留此页面（您可以在其中对多个虚拟机执行操作）：
     a. 点击 位于行右边的 Options 菜单。
   - 要在重启前查看有关所选虚拟机的综合信息：
     a. 点击虚拟机名称访问 **Virtual Machine Overview** 页面。
     b. 点击 **Actions**。
5. 选择 **Restart Virtual Machine**。
6. 在确认窗口中，点击 **Restart** 重启虚拟机。

8.6.3. 停止虚拟机

您可从 web 控制台停止虚拟机。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。

3. 找到包含您要停止的虚拟机的行。
4. 导航到适合您的用例的菜单：
   - 要保留此页面（您可以在其中对多个虚拟机执行操作）：
     a. 点击  位于行右边的 Options 菜单。
   - 在停止之前，查看所选虚拟机的综合信息：
     a. 点击虚拟机名称访问 **Virtual Machine Overview** 页面。
     b. 点击 **Actions**。
5. 选择 **Stop Virtual Machine**。
6. 在确认窗口中，点击 **Stop** 停止虚拟机。

8.6.4. 取消暂停虚拟机

您可从 web 控制台取消暂停一个正暂停的虚拟机。

先决条件

- 至少一个虚拟机的状态是 **Paused**。

注意

您可以使用 **virtctl** 客户端暂停虚拟机。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 找到包含您要取消暂停的虚拟机的行。
4. 导航到适合您的用例的菜单：
   - 要保留此页面（您可以在其中对多个虚拟机执行操作）：
     a. 在 **Status** 栏中点 **Paused**。
   - 要在取消暂停之前查看所选虚拟机的综合信息：
     a. 点击虚拟机名称访问 **Virtual Machine Overview** 页面。
     b. 点击位于 **Status** 右侧的铅笔图标
5. 在确认窗口中，点击 **Unpause** 来取消暂停虚拟机。

8.7. 访问虚拟机控制台

OpenShift Virtualization 提供不同的虚拟机控制台，您可使用这些控制台来完成不同的产品任务。您可以使用 CLI 命令通过 OpenShift Container Platform Web 控制台和访问这些控制台。

8.7.1. 在 OpenShift Container Platform web 控制台中访问虚拟机控制台

您可以使用 OpenShift Container Platform web 控制台中的串口控制台或 VNC 控制台连接至虚拟机。

您可以使用 OpenShift Container Platform Web 控制台中的 desktop viewer 控制台（使用 RDP（远程桌面协议）连接到 Windows 虚拟机。

8.7.1.1. 连接至串行控制台

从 web 控制台上 **Virtual Machine Overview** 屏幕中的 **Console** 选项卡连接至正在运行的虚拟机的串行控制台。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择一个虚拟机以打开 **Virtual Machine Overview** 页。
4. 点击 **Console**。默认会打开 VNC 控制台。
5. 选择 **Disconnect before switching**，以确保一次只打开一个控制台会话。否则，VNC 控制台会话会在后台保持活跃。
6. 点击 **VNC Console** 下拉菜单并选择 **Serial Console**。
7. 点 **Disconnect** 结束控制台会话。
8. 可选：点 **Open Console in New Window** 在一个单独的窗口中打开串口控制台。

8.7.1.2. 连接至 VNC 控制台

通过 web 控制台中的 **Virtual Machine Overview** 界面中的 **Console** 标签页连接到运行虚拟机的 VNC 控制台。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择一个虚拟机以打开 **Virtual Machine Overview** 页。
4. 点击 **Console** 选项卡。默认会打开 VNC 控制台。
5. 可选：点 **Open Console in New Window** 在一个单独的窗口中打开 VNC 控制台。
6. 可选：点 **Send Key** 将密钥组合发送到虚拟机。
7. 点控制台窗口外，然后点 **Disconnect** 结束会话。

8.7.1.3. 通过 RDP 连接至 Windows 虚拟机

桌面查看器控制台利用远程桌面协议 (RDP)，为连接至 Windows 虚拟机提供更好的控制台体验。

要使用 RDP 连接至 Windows 虚拟机，请从 web 控制台上 **Virtual Machine Details** 屏幕中的 **Consoles** 选项卡下载虚拟机的 **console.rdp** 文件，并将其提供给您首选的 RDP 客户端。

先决条件

- 正在运行的 Windows 虚拟机装有 QEMU 客户机代理。VirtIO 驱动程序中包含 **qemu-guest-agent**。
- 第 2 层 vNIC 附加到虚拟机。
- 与 Windows 虚拟机处于相同网络的机器上装有 RDP 客户端。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择 Windows 虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点击 **Console** 选项卡。
5. 在 **Console** 列表中，选择 **Desktop Viewer**。
6. 在 **Network Interface** 列表中，选择第 2 层 vNIC。
7. 点击 **Launch Remote Desktop** 下载 **console.rdp** 文件。
8. 打开 RDP 客户端并引用 **console.rdp** 文件。例如，使用 **Remmina**：

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

9. 输入 **Administrator** 用户名和密码以连接至 Windows 虚拟机。

8.7.1.4. 从 Web 控制台复制 SSH 命令

复制命令，以便从 web 控制台中的 **Actions** 列表中通过 SSH 访问正在运行的虚拟机 (VM)。

流程

1. 在 OpenShift Container Platform 控制台中，从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择一个虚拟机以打开 **Virtual Machine Overview** 页。
4. 从 **Actions** 列表中，选择 **Copy SSH Command**。现在，您可以将此命令粘贴到 OpenShift CLI (**oc**)中。

8.7.2. 使用 CLI 命令访问虚拟机控制台

8.7.2.1. 通过 SSH 访问虚拟机实例

在虚拟机上公开 22 端口后，就可以使用 SSH 访问虚拟机（VM）。

virtctl expose 命令可将虚拟机实例（VMI）端口转发到节点端口，并创建一个服务以启用对它的访问。以下示例创建 **fedora-vm-ssh** 服务，它将集群节点的特定端口流量转发到 **<fedora-vm>** 虚拟机的端口 22。

先决条件

- 您必须与 VMI 在同一个项目中。
- 您要访问的 VMI 必须使用 **masquerade** 绑定方法连接到默认 pod 网络。
- 您要访问的 VMI 必须正在运行。
- 安装 OpenShift CLI（**oc**）。

流程

1. 运行以下命令来创建 **fedora-vm-ssh** 服务：

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

1 **<fedora-vm>** 是您在其上运行 **fedora-vm-ssh** 服务的虚拟机的名称。

2. 检查服务，找出服务获取的端口：

```
$ oc get svc
```

输出示例

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

+ 在本例中，服务获取了 **32551** 端口。

1. 通过 SSH 登录 VMI。使用任何集群节点的 **ipAddress**，以及在上一步中找到的端口：

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

8.7.2.2. 使用 YAML 配置通过 SSH 访问虚拟机

您可以启用与虚拟机的 SSH 连接，而无需运行 **virtctl expose** 命令。当配置并应用虚拟机的 YAML 文件和服务的 YAML 文件时，服务会将 SSH 流量转发到虚拟机。

以下示例显示了虚拟机的 YAML 文件和服务 YAML 文件的配置。

先决条件

- 安装 OpenShift CLI (**oc**) 。

- 使用 **oc create namespace** 命令并指定命名空间的名称，为虚拟机的 YAML 文件创建一个命名空间。

流程

1. 在虚拟机的 YAML 文件中，添加标签和一个值来公开 SSH 连接的服务。为接口启用伪装（**masquerade**）功能：

   VirtualMachine 定义示例

```
...
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  namespace: ssh-ns (1)
  name: vm-ssh
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-ssh
        special: vm-ssh (2)
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: containerdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
          interfaces:
          - masquerade: {} (3)
            name: testmasquerade (4)
          rng: {}
        machine:
          type: ""
        resources:
          requests:
            memory: 1024M
      networks:
      - name: testmasquerade
        pod: {}
      volumes:
      - name: containerdisk
        containerDisk:
          image: kubevirt/fedora-cloud-container-disk-demo
      - name: cloudinitdisk
        cloudInitNoCloud:
          userData: |
            #!/bin/bash
            echo "fedora" | passwd fedora --stdin
...
```

1. **oc create namespace** 命令创建的命名空间的名称。
2. 服务用于标识为 SSH 流量连接启用的虚拟机实例的标签。标签可以是任何 **key:value** 对，作为**标签（label）**添加到此 YAML 文件，也可以作为服务 YAML 文件中的 **selector** 添加。
3. 接口类型是 **masquerade**。
4. 此接口的名称为 **testmasquerade**。

2. 创建虚拟机：

```
$ oc create -f <path_for_the_VM_YAML_file>
```

3. 启动虚拟机：

```
$ virtctl start vm-ssh
```

4. 在服务的 YAML 文件中，指定服务名称、端口号和目标端口。

Service 定义示例

```
...
apiVersion: v1
kind: Service
metadata:
  name: svc-ssh (1)
  namespace: ssh-ns (2)
spec:
  ports:
  - targetPort: 22 (3)
    protocol: TCP
    port: 27017
  selector:
    special: vm-ssh (4)
  type: NodePort
...
```

1. SSH 服务的名称。
2. **oc create namespace** 命令创建的命名空间的名称。
3. SSH 连接的目标端口号。
4. 选择器名称和值必须与虚拟机的 YAML 文件中指定的标签匹配。

5. 创建服务：

```
$ oc create -f <path_for_the_service_YAML_file>
```

6. 验证虚拟机是否正在运行：

```
$ oc get vmi
```

输出示例

```
NAME    AGE     PHASE       IP              NODENAME
vm-ssh 6s       Running     10.244.196.152  node01
```

7. 检查服务，找出服务获取的端口：

```
$ oc get svc
```

输出示例

```
NAME          TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
svc-ssh     NodePort       10.106.236.208 <none>        27017:30093/TCP   22s
```

在本例中，服务获取了端口号 30093。

8. 运行以下命令来获取节点的 IP 地址：

```
$ oc get node <node_name> -o wide
```

输出示例

```
NAME    STATUS   ROLES   AGE    VERSION          INTERNAL-IP      EXTERNAL-IP
node01  Ready    worker  6d22h  v1.20.0+5f82cdb  192.168.55.101   <none>
```

9. 通过 SSH 登录虚拟机运行的节点的 IP 地址和端口号。使用 **oc get svc** 命令显示的端口号，以及 **oc get node** 命令显示的节点 IP 地址。以下示例显示了带有用户名、节点 IP 地址和端口号的 **ssh** 命令：

```
$ ssh fedora@192.168.55.101 -p 30093
```

8.7.2.3. 访问虚拟机实例的串行控制台

virtctl console 命令可打开特定虚拟机实例的串行控制台。

先决条件

- 必须安装 **virt-viewer** 软件包。
- 您要访问的虚拟机实例必须正在运行。

流程

- 使用 **virtctl** 连接至串行控制台：

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

8.7.2.4. 使用 VNC 访问虚拟机实例的图形控制台

virtctl 客户端实用程序可使用 **remote-viewer** 功能打开正在运行的虚拟机实例的图形控制台。该功能包含在 **virt-viewer** 软件包中。

先决条件

- 必须安装 **virt-viewer** 软件包。
- 您要访问的虚拟机实例必须正在运行。

注意

如果要通过 SSH 在远程机器上使用 **virtctl**，您必须将 X 会话转发至您的机器。

流程

1. 使用 **virtctl** 实用程序连接至图形界面：

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

2. 如果命令失败，请尝试使用 **-v** 标志来收集故障排除信息：

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

8.7.2.5. 通过 RDP 控制台连接至 Windows 虚拟机

远程桌面协议 (RDP) 为连接至 Windows 虚拟机提供更好的控制台体验。

要通过 RDP 连接至 Windows 虚拟机，请为 RDP 客户端指定附加的 L2 vNIC 的 IP 地址。

先决条件

- 正在运行的 Windows 虚拟机装有 QEMU 客户机代理。VirtIO 驱动程序中包含 **qemu-guest-agent**。
- 附加到虚拟机的第 2 层 vNIC。
- 与 Windows 虚拟机处于相同网络的机器上装有 RDP 客户端。

流程

1. 以具有访问令牌的用户身份通过 **oc** CLI 工具登录 OpenShift Virtualization 集群。

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

2. 使用 **oc describe vmi** 显示正在运行的 Windows 虚拟机的配置。

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

输出示例

```
...
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. 找出并复制第 2 层网络接口的 IP 地址。在以上示例中是 **192.0.2.0**，如果您首选 IPv6，则为 **2001:db8::**。

4. 打开 RDP 客户端，并使用上一步中复制的 IP 地址进行连接。

5. 输入 **Administrator** 用户名和密码以连接至 Windows 虚拟机。

8.8. 解决故障节点来触发虚拟机故障切换

如果节点失败，并且没有在集群中部署 机器健康检查，则带有 **RunStrategy: Always** 配置的虚拟机（VM）不会被自动重新定位到健康的节点上。要触发虚拟机故障切换，您必须手动删除 **Node** 对象。

注意

如果使用安装程序置备的基础架构安装集群，并且正确地配置了机器健康检查：

- 故障节点会被自动回收。
- **RunStrategy** 被设置为 **Always** 或 **RerunOnFailure** 的虚拟机会自动被调度到健康的节点上。

8.8.1. 先决条件

- 运行虚拟机的节点具有 **NotReady** 条件。
- 在故障节点中运行的虚拟机的 **RunStrategy** 设置为 **Always**。
- 已安装 OpenShift CLI（**oc**）。

8.8.2. 从裸机集群中删除节点

当您使用 CLI 删除节点时，节点对象会从 Kubernetes 中删除，但该节点上存在的 pod 不会被删除。任何未由复制控制器支持的裸机 pod 都无法从 OpenShift Container Platform 访问。由复制控制器支持的 Pod 会重新调度到其他可用的节点。您必须删除本地清单 pod。

流程

通过完成以下步骤，从裸机上运行的 OpenShift Container Platform 集群中删除节点：

1. 将节点标记为不可调度：

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

2. 排空节点上的所有 pod：

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

如果节点离线或者无响应，此步骤可能会失败。即使节点没有响应，它仍然在运行写入共享存储的工作负载。为了避免数据崩溃，请在进行操作前关闭物理硬件。

3. 从集群中删除节点：

```
$ oc delete node <node_name>
```

虽然节点对象现已从集群中删除，但它仍然可在重启后或 kubelet 服务重启后重新加入集群。要永久删除该节点及其所有数据，您必须弃用该节点。

4. 如果您关闭了物理硬件，请重新打开它以便节点可以重新加入集群。

8.8.3. 验证虚拟机故障切换

在不健康节点上终止所有资源后，会为每个重新定位的虚拟机在健康的节点上自动创建新虚拟机实例（VMI）。要确认已创建了 VMI，使用 **oc** CLI 查看所有 VMI。

8.8.3.1. 使用 CLI 列出所有虚拟机实例

您可以使用 **oc** 命令行界面（CLI）列出集群中的所有虚拟机实例（VMI），包括独立 VMI 和虚拟机拥有的实例。

流程

- 运行以下命令列出所有 VMI：

```
$ oc get vmis
```

8.9. 在虚拟机上安装 QEMU 客户机代理

QEMU 客户机代理是在虚拟机上运行的一个守护进程，它会将有关虚拟机、用户、文件系统和从属网络的信息传递给主机。

8.9.1. 在 Linux 虚拟机上安装 QEMU 客户机代理

qemu-guest-agent 广泛可用，默认在红帽虚拟机中可用。安装代理并启动服务

流程

1. 通过其中一个控制台或通过 SSH 访问虚拟机命令行。

2. 在虚拟机上安装 QEMU 客户机代理：

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

3. 确保服务持久并启动它：

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

在 web 控制台中创建虚拟机或虚拟机模板时，您还可使用向导的 **cloud-init** 部分中的 **custom script** 字段来安装和启动 QEMU 客户机代理。

8.9.2. 在 Windows 虚拟机上安装 QEMU 客户机代理

对于 Windows 虚拟机，QEMU 客户机代理包含在 VirtIO 驱动程序中，该驱动程序可使用以下流程之一进行安装：

8.9.2.1. 在现有 Windows 虚拟机上安装 VirtIO 驱动程序

从附加的 SATA CD 驱动器将 VirtIO 驱动程序安装到现有 Windows 虚拟机。

注意

该流程使用通用方法为 Windows 添加驱动。具体流程可能会因 Windows 版本而稍有差异。有关具体安装步骤，请参阅您的 Windows 版本安装文档。

流程

1. 启动虚拟机并连接至图形控制台。
2. 登录 Windows 用户会话。
3. 打开 **Device Manager** 并展开 **Other devices** 以列出所有 **Unknown device**。
   a. 打开 **Device Properties** 以识别未知设备。右击设备并选择 **Properties**。
   b. 单击 **Details** 选项卡，并在 **Property** 列表中选择 **Hardware Ids**。
   c. 将 **Hardware Ids** 的 **Value** 与受支持的 VirtIO 驱动程序相比较。
4. 右击设备并选择 **Update Driver Software**。
5. 点击 **Browse my computer for driver software**并浏览所附加的 VirtIO 驱动程序所在 SATA CD 驱动器。驱动程序将按照其驱动程序类型、操作系统和 CPU 架构分层排列。
6. 点击 **Next** 以安装驱动程序。
7. 对所有必要 VirtIO 驱动程序重复这一过程。
8. 安装完驱动程序后，点击 **Close** 关闭窗口。
9. 重启虚拟机以完成驱动程序安装。

8.9.2.2. 在 Windows 安装过程中安装 VirtIO 驱动程序

在 Windows 安装过程中，从附加的 SATA CD 驱动程序安装 VirtIO 驱动程序。

注意

该流程使用通用方法安装 Windows，且安装方法可能因 Windows 版本而异。有关您要安装的 Windows 版本，请参阅相关文档。

流程

1. 启动虚拟机并连接至图形控制台。
2. 开始 Windows 安装过程。
3. 选择 **Advanced** 安装。
4. 加载驱动程序前无法识别存储目的地。点击 **Load driver**。
5. 驱动程序将附加为 SATA CD 驱动器。点击 **OK** 并浏览 CD 驱动器以加载存储驱动程序。驱动程序将按照其驱动程序类型、操作系统和 CPU 架构分层排列。
6. 对所有所需驱动程序重复前面两步。
7. 完成 Windows 安装。

8.10. 查看虚拟机的 QEMU 客户机代理信息

当 QEMU 客户机代理在虚拟机上运行时，您可以使用 web 控制台查看有关虚拟机、用户、文件系统和从属网络的信息。

8.10.1. 先决条件

- 在虚拟机上安装 QEMU 客户机代理。

8.10.2. 关于 web 控制台中的 QEMU 客户机代理信息

安装 QEMU 客户机代理后，**Virtual Machine Overview** 标签页中的 **Details** 面板和 **Details** 标签中会显示主机名、操作系统、时区和登录用户的信息。

Virtual Machine Overview 显示有关在虚拟机上安装的客户端操作系统的信息。**Details** 标签显示有登录用户信息的表。**Disks** 标签页显示含有文件系统信息的表格。

注意

如果没有安装 QEMU 客户机代理，**Virtual Machine Overview** 选项卡和 **Details** 选项卡会显示在创建虚拟机时指定的操作系统信息。

8.10.3. 在 web 控制台中查看 QEMU 客户机代理信息

您可以使用 web 控制台查看由 QEMU 客户机代理传递给主机的虚拟机信息。

流程

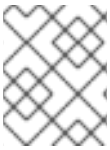
1. 从侧边菜单中选择 **Workloads → Virtual Machines**。
2. 点 **Virtual Machines** 标签页。

3. 选择虚拟机名称以打开 **Virtual Machine Overview** 屏幕，并查看 **Details**。
4. 点 **Logged in users** 查看显示用户信息的 **Details** 选项卡。
5. 点 **Disks** 标签查看文件系统的信息。

8.11. 在虚拟机中管理配置映射、SECRET 和服务帐户

您可以使用 secret、配置映射和服务帐户将配置数据传递给虚拟机。例如，您可以：

- 通过向虚拟机添加 secret 来授予虚拟机对需要凭证的服务的访问权限。
- 在配置映射中存储非机密配置数据，以便 pod 或另一个对象可以使用这些数据。
- 允许组件通过将服务帐户与该组件关联来访问 API 服务器。



注意

OpenShift Virtualization 将 secret、配置映射和服务帐户作为虚拟机磁盘公开，以便可以在平台间使用这些 secret、ConfigMap 和服务帐户而无需额外的开销。

8.11.1. 将 secret、配置映射或服务帐户添加到虚拟机

使用 OpenShift Container Platform Web 控制台向虚拟机添加 secret、配置映射或服务帐户。

先决条件

- 要添加的 secret、配置映射或服务帐户必须与目标虚拟机位于同一命名空间中。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Environment** 标签页。
5. 点 **Select a resource**，再从列表中选择一个 secret、配置映射或服务帐户。为所选资源自动生成带有六个字符的序列号。
6. 点 **Save**。
7. 可选。点 **Add Config Map, Secret or Service Account** 添加另外一个对象。

注意

a. 您可以点击 **Reload** 将表单重置为最后一个保存的状态。

b. **Environment** 资源作为磁盘添加到虚拟机中。您可在挂载任何其他磁盘时挂载 secret、配置映射或服务帐户。

c. 如果虚拟机正在运行，则更改在重启虚拟机之后才会生效。新添加的资源在页面顶部的 **Pending Changes** 中的 **Environment** 和 **Disks** 都会标记为改变待处理。

验证

1. 在 **Virtual Machine Overview** 页面中点击 **Disks** 选项卡。
2. 检查以确保 secret、配置映射或服务帐户包括在磁盘列表中。
3. 可选。选择适当的方法来应用您的更改：
   a. 如果虚拟机正在运行，点 **Actions → Restart Virtual Machine** 重启虚拟机。
   b. 如果停止虚拟机，点 **Actions → Start Virtual Machine** 启动虚拟机。

现在，您可以在挂载任何其他磁盘时挂载 secret、配置映射或服务帐户。

8.11.2. 从虚拟机中删除 secret、配置映射或服务帐户

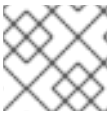
使用 OpenShift Container Platform Web 控制台从虚拟机中删除 secret、配置映射或服务帐户。

先决条件

- 您必须至少有一个 secret、配置映射或服务帐户附加到虚拟机。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Environment** 标签页。
5. 在列表中找到您要删除的项目，然后点击项目右侧的 **Remove** ⛔。
6. 点击 **Save**。



注意

您可以点击 **Reload** 将表单重置为最后一个保存的状态。

验证

1. 在 **Virtual Machine Overview** 页面中点击 **Disks** 选项卡。
2. 检查以确保删除的 secret、配置映射或服务帐户不再包含在磁盘列表中。

8.11.3. 其他资源

- 为 pod 提供敏感数据
- 了解并创建服务帐户
- 了解配置映射

8.12. 在现有 WINDOWS 虚拟机上安装 VIRTIO 驱动程序

8.12.1. 了解 VirtIO 驱动程序

VirtIO 驱动程序是 Microsoft Windows 虚拟机在 OpenShift Virtualization 中运行时所需的半虚拟化设备驱动程序。受支持的驱动程序可在 红帽生态系统目录的 **container-native-virtualization/virtio-win** 容器磁盘中找到。

必须将 **container-native-virtualization/virtio-win** 容器磁盘作为 SATA CD 驱动器附加到虚拟机，以启用驱动程序安装。您可在虚拟机安装 Windows 期间安装 VirtIO 驱动程序，或将其附加到现有 Windows 安装。

安装完驱动程序后，可从虚拟机中移除 **container-native-virtualization/virtio-win** 容器磁盘。

另请参阅： 在新 Windows 虚拟机上安装 Virtio 驱动程序 。

8.12.2. Microsoft Windows 虚拟机支持的 VirtIO 驱动程序

表 8.1. 支持的驱动程序

驱动程序名称	硬件 ID	描述
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	块驱动程序。有时会在 **Other devices** 组中显示为 **SCSI Controller**。
viorng	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	熵源（entropy）驱动程序。有时会在 **Other devices** 组中显示为 **PCI Device**。
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	网络驱动程序。有时会在 **Other devices** 组中显示为 **Ethernet Controller**。仅在配置了 VirtIO NIC 时可用。

8.12.3. 将 VirtIO 驱动程序容器磁盘添加到虚拟机中

针对 Microsoft Windows 的 OpenShift Virtualization VirtIO 驱动程序作为一个容器磁盘提供，可在 Red Hat Ecosystem Catalog 中找到。要为 Windows 虚拟机安装这些驱动程序，请在虚拟机配置文件中将 **container-native-virtualization/virtio-win** 容器磁盘作为 SATA CD 驱动器附加到虚拟机。

先决条件

- 从 Red Hat Ecosystem Catalog 下载 **container-native-virtualization/virtio-win** 容器磁盘。这一步并非强制要求，因为如果集群中不存在容器磁盘，将从 Red Hat registry 中下载，但通过此步下载可节省安装时间。

流程

1. 将 **container-native-virtualization/virtio-win** 容器磁盘作为 **cdrom** 磁盘添加到 Windows 虚拟机配置文件中。如果集群中还没有容器磁盘，将从 registry 中下载。

```
spec:
  domain:
    devices:
```

```
    disks:
      - name: virtiocontainerdisk
        bootOrder: 2 (1)
        cdrom:
          bus: sata
volumes:
  - containerDisk:
      image: container-native-virtualization/virtio-win
    name: virtiocontainerdisk
```

(1) OpenShift Virtualization 按照 **VirtualMachine** 配置文件中定义的顺序启动虚拟机磁盘。您可将虚拟机的其他磁盘定义到 **container-native-virtualization/virtio-win** 容器磁盘前面，也可使用 **bootOrder** 可选参数来确保虚拟机从正确磁盘启动。如果为一个磁盘指定 **bootOrder**，则必须为配置中的所有磁盘指定。

2. 虚拟机启动后，磁盘随即可用：

   - 如果要将容器磁盘添加到正在运行的虚拟机，请在 CLI 中执行 **oc apply -f <vm.yaml>**，或重启虚拟机，以使更改生效。
   - 如果虚拟机还未运行，则使用 **virtctl start <vm>**。

虚拟机启动后，可从附加的 SATA CD 驱动器安装 VirtIO 驱动程序。

8.12.4. 在现有 Windows 虚拟机上安装 VirtIO 驱动程序

从附加的 SATA CD 驱动器将 VirtIO 驱动程序安装到现有 Windows 虚拟机。

注意

该流程使用通用方法为 Windows 添加驱动。具体流程可能会因 Windows 版本而稍有差异。有关具体安装步骤，请参阅您的 Windows 版本安装文档。

流程

1. 启动虚拟机并连接至图形控制台。
2. 登录 Windows 用户会话。
3. 打开 **Device Manager** 并展开 **Other devices** 以列出所有 **Unknown device**。
   a. 打开 **Device Properties** 以识别未知设备。右击设备并选择 **Properties**。
   b. 单击 **Details** 选项卡，并在 **Property** 列表中选择 **Hardware Ids**。
   c. 将 **Hardware Ids** 的 **Value** 与受支持的 VirtIO 驱动程序相比较。
4. 右击设备并选择 **Update Driver Software**。
5. 点击 **Browse my computer for driver software**并浏览所附加的 VirtIO 驱动程序所在 SATA CD 驱动器。驱动程序将按照其驱动程序类型、操作系统和 CPU 架构分层排列。
6. 点击 **Next** 以安装驱动程序。
7. 对所有必要 VirtIO 驱动程序重复这一过程。

8. 安装完驱动程序后，点击 **Close** 关闭窗口。

9. 重启虚拟机以完成驱动程序安装。

8.12.5. 从虚拟机移除 VirtIO 容器磁盘

在向虚拟机安装完所有所需 VirtIO 驱动程序后，**container-native-virtualization/virtio-win** 容器磁盘便不再需要附加到虚拟机。从虚拟机配置文件中移除 **container-native-virtualization/virtio-win** 容器磁盘。

流程

1. 编辑配置文件并移除 **disk** 和 **volume**。

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

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

2. 重启虚拟机以使更改生效。

8.13. 在新 WINDOWS 虚拟机上安装 VIRTIO 驱动程序

8.13.1. 先决条件

- Windows 安装介质可访问虚拟机，比如将 ISO 导入到数据卷 并将其附加到虚拟机。

8.13.2. 了解 VirtIO 驱动程序

VirtIO 驱动程序是 Microsoft Windows 虚拟机在 OpenShift Virtualization 中运行时所需的半虚拟化设备驱动程序。受支持的驱动程序可在 红帽生态系统目录的 **container-native-virtualization/virtio-win** 容器磁盘中找到。

必须将 **container-native-virtualization/virtio-win** 容器磁盘作为 SATA CD 驱动器附加到虚拟机，以启用驱动程序安装。您可在虚拟机安装 Windows 期间安装 VirtIO 驱动程序，或将其附加到现有 Windows 安装。

安装完驱动程序后，可从虚拟机中移除 **container-native-virtualization/virtio-win** 容器磁盘。

另请参阅： 在现有 Windows 虚拟机上安装 VirtIO 驱动程序。

8.13.3. Microsoft Windows 虚拟机支持的 VirtIO 驱动程序

表 8.2. 支持的驱动程序

驱动程序名称	硬件 ID	描述
viostor	VEN_1AF4&DEV_1001 VEN_1AF4&DEV_1042	块驱动程序。有时会在 **Other devices** 组中显示为 **SCSI Controller**。
viorng	VEN_1AF4&DEV_1005 VEN_1AF4&DEV_1044	熵源（entropy）驱动程序。有时会在 **Other devices** 组中显示为 **PCI Device**。
NetKVM	VEN_1AF4&DEV_1000 VEN_1AF4&DEV_1041	网络驱动程序。有时会在 **Other devices** 组中显示为 **Ethernet Controller**。仅在配置了 VirtIO NIC 时可用。

8.13.4. 将 VirtIO 驱动程序容器磁盘添加到虚拟机中

针对 Microsoft Windows 的 OpenShift Virtualization VirtIO 驱动程序作为一个容器磁盘提供，可在 Red Hat Ecosystem Catalog 中找到。要为 Windows 虚拟机安装这些驱动程序，请在虚拟机配置文件中将 **container-native-virtualization/virtio-win** 容器磁盘作为 SATA CD 驱动器附加到虚拟机。

先决条件

- 从 Red Hat Ecosystem Catalog 下载 **container-native-virtualization/virtio-win** 容器磁盘。这一步并非强制要求，因为如果集群中不存在容器磁盘，将从 Red Hat registry 中下载，但通过此步下载可节省安装时间。

流程

1. 将 **container-native-virtualization/virtio-win** 容器磁盘作为 **cdrom** 磁盘添加到 Windows 虚拟机配置文件中。如果集群中还没有容器磁盘，将从 registry 中下载。

```
spec:
  domain:
    devices:
      disks:
        - name: virtiocontainerdisk
          bootOrder: 2 (1)
          cdrom:
            bus: sata
volumes:
  - containerDisk:
      image: container-native-virtualization/virtio-win
    name: virtiocontainerdisk
```

(1) OpenShift Virtualization 按照 **VirtualMachine** 配置文件中定义的顺序启动虚拟机磁盘。您可将虚拟机的其他磁盘定义到 **container-native-virtualization/virtio-win** 容器磁盘前面，也可使用 **bootOrder** 可选参数来确保虚拟机从正确磁盘启动。如果为一个磁盘指定 **bootOrder**，则必须为配置中的所有磁盘指定。

2. 虚拟机启动后，磁盘随即可用：

- 如果要将容器磁盘添加到正在运行的虚拟机，请在 CLI 中执行 **oc apply -f <vm.yaml>**，或重启虚拟机，以使更改生效。
- 如果虚拟机还未运行，则使用 **virtctl start <vm>**。

虚拟机启动后，可从附加的 SATA CD 驱动器安装 VirtIO 驱动程序。

8.13.5. 在 Windows 安装过程中安装 VirtIO 驱动程序

在 Windows 安装过程中，从附加的 SATA CD 驱动程序安装 VirtIO 驱动程序。

注意

该流程使用通用方法安装 Windows，且安装方法可能因 Windows 版本而异。有关您要安装的 Windows 版本，请参阅相关文档。

流程

1. 启动虚拟机并连接至图形控制台。
2. 开始 Windows 安装过程。
3. 选择 **Advanced** 安装。
4. 加载驱动程序前无法识别存储目的地。点击 **Load driver**。
5. 驱动程序将附加为 SATA CD 驱动器。点击 **OK** 并浏览 CD 驱动器以加载存储驱动程序。驱动程序将按照其驱动程序类型、操作系统和 CPU 架构分层排列。
6. 对所有所需驱动程序重复前面两步。
7. 完成 Windows 安装。

8.13.6. 从虚拟机移除 VirtIO 容器磁盘

在向虚拟机安装完所有所需 VirtIO 驱动程序后，**container-native-virtualization/virtio-win** 容器磁盘便不再需要附加到虚拟机。从虚拟机配置文件中移除 **container-native-virtualization/virtio-win** 容器磁盘。

流程

1. 编辑配置文件并移除 **disk** 和 **volume**。

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

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

```
- containerDisk:
    image: container-native-virtualization/virtio-win
  name: virtiocontainerdisk
```

2. 重启虚拟机以使更改生效。

8.14. 高级虚拟机管理

8.14.1. 为虚拟机使用资源配额

为虚拟机创建和管理资源配额。

8.14.1.1. 为虚拟机设置资源配额限制

只有使用请求自动用于虚拟机 (VM) 的资源配额。如果您的资源配额使用限制，则必须为虚拟机手动设置资源限值。资源限值必须至少大于资源请求的 100 MiB。

流程

1. 通过编辑 **VirtualMachine** 清单来为虚拟机设置限值。例如：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: with-limits
spec:
  running: false
  template:
    spec:
      domain:
# ...
        resources:
          requests:
            memory: 128Mi
          limits:
            memory: 256Mi 1
```

1 这个配置被支持，因为 **limits.memory** 值至少比 **requests.memory** 的值大 **100Mi**。

2. 保存 **VirtualMachine** 清单。

8.14.1.2. 其他资源

- 项目的资源配额
- 跨越多个项目的资源配额

8.14.2. 为虚拟机指定节点

您可以使用节点放置规则将虚拟机放置到特定的节点上。

8.14.2.1. 关于虚拟机的节点放置

要确保虚拟机在适当的节点上运行，您可以配置节点放置规则。如果出现以下情况，您可能需要进行此操作：

- 您有多台虚拟机。为确保容错，您希望它们在不同节点上运行。
- 您有两个 chatty 虚拟机。为了避免冗余节点间路由，您希望虚拟机在同一节点上运行。
- 您的虚拟机需要所有可用节点上不存在的特定硬件功能。
- 您有一个 pod 可以向节点添加功能，并想将虚拟机放置到该节点上，以便它可以使用这些功能。

注意

虚拟机放置依赖于工作负载的现有节点放置规则。如果组件级别上的特定节点排除工作负载，则虚拟机无法放置在这些节点上。

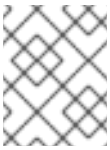
您可以在 **VirtualMachine** 清单的 **spec** 字段中使用以下规则类型：

nodeSelector

允许将虚拟机调度到使用此字段中指定的键值对标记的节点上。节点必须具有与所有列出的对完全匹配的标签。

关联性

这可让您使用更具表达力的语法来设置与虚拟机匹配的规则。例如，您可以指定规则是首选项，而非硬要求，因此在规则不满足时仍然可以调度虚拟机。虚拟机放置支持 Pod 关联性、pod 反关联性和节点关联性。Pod 关联性适用于虚拟机，因为 **VirtualMachine** 工作负载类型基于 **Pod** 对象。



注意

关联性规则仅在调度期间应用。如果不再满足限制，OpenShift Container Platform 不会重新调度正在运行的工作负载。

容限（tolerations）

允许将虚拟机调度到具有匹配污点的节点。如果污点应用到某个节点，则该节点只接受容许该污点的虚拟机。

8.14.2.2. 节点放置示例

以下示例 YAML 文件片断使用 **nodePlacement**、**affinity** 和 **tolerations** 字段为虚拟机自定义节点放置。

8.14.2.2.1. 示例：使用 nodeSelector 放置虚拟机节点

在本例中，虚拟机需要一个包含 **example-key-1 = example-value-1** 和 **example-key-2 = example-value-2** 标签的元数据的节点。

警告

如果没有节点适合此描述，则不会调度虚拟机。

VM 清单示例

```
metadata:
  name: example-vm-node-selector
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  template:
    spec:
      nodeSelector:
        example-key-1: example-value-1
        example-key-2: example-value-2
...
```

8.14.2.2.2. 示例：使用 pod 关联性和 pod 反关联性的虚拟机节点放置

在本例中，虚拟机必须调度到具有标签 **example-key-1 = example-value-1** 的正在运行的 pod 的节点上。如果没有在任何节点上运行这样的 pod，则不会调度虚拟机。

如果可能，虚拟机不会调度到具有标签 **example-key-2 = example-value-2** 的 pod 的节点上。但是，如果所有候选节点都有具有此标签的 pod，调度程序会忽略此约束。

VM 清单示例

```
metadata:
  name: example-vm-pod-affinity
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution: (1)
      - labelSelector:
          matchExpressions:
          - key: example-key-1
            operator: In
            values:
            - example-value-1
        topologyKey: kubernetes.io/hostname
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution: (2)
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: example-key-2
              operator: In
              values:
              - example-value-2
          topologyKey: kubernetes.io/hostname
...
```

(1) 如果您使用 **requiredDuringSchedulingIgnoredDuringExecution** 规则类型，如果没有满足约束，则不会调度虚拟机。

2 如果您使用 **preferredDuringSchedulingIgnoredDuringExecution** 规则类型，只要满足所有必要的限制，仍会调度虚拟机（如果未满足约束）。

8.14.2.2.3. 示例：使用节点关联性进行虚拟机节点放置

在本例中，虚拟机必须调度到具有标签 **example.io/example-key = example-value-1** 或标签 **example.io/example-key = example-value-2** 的节点上。如果节点上只有一个标签，则会满足约束。如果没有标签，则不会调度虚拟机。

若有可能，调度程序会避免具有标签 **example-node-label-key = example-node-label-value** 的节点。但是，如果所有候选节点都具有此标签，调度程序会忽略此限制。

VM 清单示例

```
metadata:
  name: example-vm-node-affinity
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution: 1
        nodeSelectorTerms:
        - matchExpressions:
          - key: example.io/example-key
            operator: In
            values:
            - example-value-1
            - example-value-2
      preferredDuringSchedulingIgnoredDuringExecution: 2
      - weight: 1
        preference:
          matchExpressions:
          - key: example-node-label-key
            operator: In
            values:
            - example-node-label-value
...
```

1 如果您使用 **requiredDuringSchedulingIgnoredDuringExecution** 规则类型，如果没有满足约束，则不会调度虚拟机。

2 如果您使用 **preferredDuringSchedulingIgnoredDuringExecution** 规则类型，只要满足所有必要的限制，仍会调度虚拟机（如果未满足约束）。

8.14.2.2.4. 示例：带有容限的虚拟机节点放置

在本例中，为虚拟机保留的节点已使用 **key=virtualization:NoSchedule** 污点标记。由于此虚拟机具有匹配的**容限**，它可以调度到污点节点上。

注意

容许污点的虚拟机不需要调度到具有该污点的节点。

VM 清单示例

```
metadata:
  name: example-vm-tolerations
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "virtualization"
    effect: "NoSchedule"
...
```

8.14.2.3. 其他资源

- 为虚拟化组件指定节点
- 使用节点选择器将 pod 放置到特定节点
- 使用节点关联性规则控制节点上的 pod 放置
- 使用节点污点控制 pod 放置

8.14.3. 配置证书轮转

配置证书轮转参数以替换现有证书。

8.14.3.1. 配置证书轮转

您可以在 web 控制台中的 OpenShift Virtualization 安装过程中，或者在安装 **HyperConverged** 自定义资源（CR）后完成此操作。

流程

1. 运行以下命令打开 **HyperConverged** CR：

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. 按照以下示例所示，编辑 **spec.certConfig** 字段。要避免系统过载，请确保所有值都大于或等于 10 分钟。将所有值显示为符合 golang **ParseDuration** 格式的字符串。

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
 name: kubevirt-hyperconverged
 namespace: openshift-cnv
spec:
  certConfig:
    ca:
      duration: 48h0m0s
      renewBefore: 24h0m0s (1)
```

```
    server:
      duration: 24h0m0s 2
      renewBefore: 12h0m0s 3
```

1. **ca.renewBefore** 的值必须小于或等于 **ca.duration** 的值。
2. **server.duration** 的值必须小于或等于 **ca.duration** 的值。
3. **server.renewBefore** 的值必须小于或等于 **server.duration** 的值。

3. 将 YAML 文件应用到集群。

8.14.3.2. 证书轮转参数故障排除

删除一个或多个 **certConfig** 值会导致它们恢复到默认值，除非默认值与以下条件之一冲突：

- **ca.renewBefore** 的值必须小于或等于 **ca.duration** 的值。
- **server.duration** 的值必须小于或等于 **ca.duration** 的值。
- **server.renewBefore** 的值必须小于或等于 **server.duration** 的值。

如果默认值与这些条件冲突，您将收到错误。

如果您删除了以下示例中的 **server.duration** 值，则默认值 **24h0m0s** 大于 **ca.duration** 的值，并与指定条件冲突。

示例

```
certConfig:
  ca:
    duration: 4h0m0s
    renewBefore: 1h0m0s
  server:
    duration: 4h0m0s
    renewBefore: 4h0m0s
```

这会生成以下出错信息：

```
error: hyperconvergeds.hco.kubevirt.io "kubevirt-hyperconverged" could not be patched: admission webhook "validate-hco.kubevirt.io" denied the request: spec.certConfig: ca.duration is smaller than server.duration
```

错误消息仅提及第一个冲突。在继续操作前，查看所有 certConfig 值。

8.14.4. 自动执行管理任务

您可以使用 Red Hat Ansible Automation Platform 自动完成 OpenShift Virtualization 管理任务。通过使用 Ansible Playbook 创建新虚拟机来了解基础知识。

8.14.4.1. 关于 Red Hat Ansible Automation

Ansible 是用于配置系统、部署软件和执行滚动更新的自动化工具。Ansible 包含对 OpenShift Virtualization 的支持，Ansible 模块可用于自动执行集群管理任务，如模板、持久性卷声明和虚拟机操作。

Ansible 提供了一种方式来自动执行 OpenShift Virtualization 管理，您也可以使用 **oc** CLI 工具或 API 来完成此项操作。Ansible 独具特色，因其可用于将 KubeVirt 模块与其他 Ansible 模块集成。

8.14.4.2. 自动创建虚拟机

使用 Red Hat Ansible Automation Platform，您可使用 **kubevirt_vm** Ansible Playbook 在 OpenShift Container Platform 集群中创建虚拟机。

先决条件

- Red Hat Ansible Engine 版本 2.8 或更新版本

流程

1. 编辑 Ansible Playbook YAML 文件，以便其包含 **kubevirt_vm** 任务：

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

注意

该片断仅包含 playbook 的 **kubevirt_vm** 部分。

2. 编辑这些值以反应您要创建的虚拟机，包括 **namespace**、**cpu_cores** 数、**memory** 以及 **disks**。例如：

```
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. 如果希望虚拟机创建后立即启动，请向 YAML 文件添加 **state: running**。例如：

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

1 将该值改为 **state: absent** 会删除已存在的虚拟机。

4. 运行 **ansible-playbook** 命令，将 playbook 文件名用作唯一参数：

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

5. 查看输出以确定该 play 是否成功：

   输出示例

```
(...)
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. 如果您未在 playbook 文件中包含 **state: running**，而您希望立即启动虚拟机，请编辑文件使其包含 **state: running** 并再次运行 playbook：

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

要验证是否已创建虚拟机，请尝试访问虚拟机控制台。

8.14.4.3. 示例：用于创建虚拟机的 Ansible Playbook

您可使用 **kubevirt_vm** Ansible Playbook 自动创建虚拟机。

以下 YAML 文件是一个 **kubevirt_vm** playbook 示例。如果运行 playbook，其中会包含必须替换为您自己的信息的样本值。

```
---
- 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
```

其他信息

- Playbook 简介
- 验证 Playbook 的工具

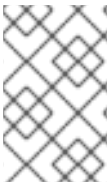
8.14.5. 将 EFI 模式用于虚拟机

您可以使用可扩展固件界面（EFI）模式引导虚拟机（VM）。

8.14.5.1. 关于虚拟机的 EFI 模式

像传统的 BIOS 一样，可扩展固件界面（EFI）在计算机启动时初始化硬件组件和操作系统镜像文件。EFI 支持比 BIOS 更先进的功能和自定义选项，从而可以更快地启动时间。

它将初始化和启动的所有信息保存在带有 **.efi** 扩展的文件中，该扩展被保存在名为 EFI 系统分区（ESP）的特殊分区中。ESP 还包含安装在计算机上的操作系统的引导装载程序程序。



注意

在使用 EFI 模式时，OpenShift Virtualization 只支持使用安全引导机制的虚拟机（VM）。如果没有启用安全引导机制，则虚拟机会重复崩溃。但是，虚拟机可能不支持安全引导。在引导虚拟机前，请通过检查虚拟机设置来验证它是否支持安全引导。

8.14.5.2. 在 EFI 模式中引导虚拟机

您可以通过编辑 VM 或 VMI 清单，将虚拟机配置为在 EFI 模式中引导。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

1. 创建定义虚拟机对象或虚拟机实例（VMI）对象的 YAML 文件。使用示例 YAML 文件的固件小节：

使用安全引导活跃在 EFI 模式中引导

```
apiversion: kubevirt.io/v1
kind: VirtualMachineInstance
metadata:
  labels:
    special: vmi-secureboot
  name: vmi-secureboot
spec:
  domain:
    devices:
      disks:
```

```
      - disk:
          bus: virtio
        name: containerdisk
    features:
      acpi: {}
      smm:
        enabled: true (1)
    firmware:
      bootloader:
        efi:
          secureBoot: true (2)
...
```

1. OpenShift Virtualization 需要为 EFI 模式的安全引导启用系统管理模式（**SMM**）。
2. 在使用 EFI 模式时，OpenShift Virtualization 只支持使用安全引导机制的虚拟机（VM）。如果没有启用安全引导机制，则虚拟机会重复崩溃。但是，虚拟机可能不支持安全引导。在引导虚拟机前，请通过检查虚拟机设置来验证它是否支持安全引导。

2. 运行以下命令，将清单应用到集群：

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

8.14.6. 为虚拟机配置 PXE 启动

OpenShift Virtualization 中提供 PXE 启动或网络启动。网络启动支持计算机启动和加载操作系统或其他程序，无需本地连接的存储设备。例如，在部署新主机时，您可使用 PXE 启动从 PXE 服务器中选择所需操作系统镜像。

8.14.6.1. 先决条件

- Linux 网桥 必须已连接。
- PXE 服务器必须作为网桥连接至相同 VLAN。

8.14.6.2. 使用指定的 MAC 地址的 PXE 引导

作为管理员，您可首先为您的 PXE 网络创建 **NetworkAttachmentDefinition** 对象，以此通过网络引导客户端。然后在启动虚拟机实例前，在您的虚拟机实例配置文件中引用网络附加定义。如果 PXE 服务器需要，您还可在虚拟机实例配置文件中指定 MAC 地址。

先决条件

- 必须已连接 Linux 网桥。
- PXE 服务器必须作为网桥连接至相同 VLAN。

流程

1. 在集群上配置 PXE 网络：
   a. 为 PXE 网络 **pxe-net-conf** 创建网络附加定义文件：

```
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 (1)
      },
      {
        "type": "cnv-tuning" (2)
      }
    ]
  }'
```

1. 可选： VLAN 标签。
2. **cnv-tuning** 插件为自定义 MAC 地址提供支持。

注意

虚拟机实例将通过所请求的 VLAN 的访问端口附加到网桥 **br1**。

2. 使用您在上一步中创建的文件创建网络附加定义：

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

3. 编辑虚拟机实例配置文件以包括接口和网络的详情。
   a. 如果 PXE 服务器需要，请指定网络和 MAC 地址。如果未指定 MAC 地址，则会自动分配一个值。
   请确保 **bootOrder** 设置为 **1**，以便该接口先启动。在本例中，该接口连接到了名为 **<pxe-net>** 的网络中：

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

注意

启动顺序对于接口和磁盘全局通用。

   b. 为磁盘分配一个启动设备号，以确保置备操作系统后能够正确启动。

将磁盘 **bootOrder** 值设置为 **2**：

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

c. 指定网络连接到之前创建的网络附加定义。在这种情况下，**<pxe-net>** 连接到名为 **<pxe-net-conf>** 的网络附加定义：

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

4. 创建虚拟机实例：

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

输**出示例**

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

1. 等待虚拟机实例运行：

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

2. 使用 VNC 查看虚拟机实例：

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

3. 查看启动屏幕，验证 PXE 启动是否成功。

4. 登录虚拟机实例：

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

5. 验证虚拟机上的接口和 MAC 地址，并验证连接到网桥的接口是否具有指定的 MAC 地址。在本例中，我们使用了 **eth1** 进行 PXE 启动，无需 IP 地址。另一接口 **eth0** 从 OpenShift Container Platform 获取 IP 地址。

```
$ ip addr
```

输**出示例**

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

8.14.6.3. 模板：用于 PXE 引导的虚拟机实例配置文件

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstance
metadata:
  creationTimestamp: null
  labels:
    special: vmi-pxe-boot
  name: vmi-pxe-boot
spec:
  domain:
    devices:
      disks:
      - disk:
          bus: virtio
        name: containerdisk
        bootOrder: 2
      - disk:
          bus: virtio
        name: cloudinitdisk
      interfaces:
      - masquerade: {}
        name: default
      - bridge: {}
        name: pxe-net
        macAddress: de:00:00:00:00:de
        bootOrder: 1
    machine:
      type: ""
    resources:
      requests:
        memory: 1024M
  networks:
  - name: default
    pod: {}
  - multus:
      networkName: pxe-net-conf
    name: pxe-net
  terminationGracePeriodSeconds: 0
  volumes:
  - name: containerdisk
    containerDisk:
      image: kubevirt/fedora-cloud-container-disk-demo
  - cloudInitNoCloud:
      userData: |
        #!/bin/bash
        echo "fedora" | passwd fedora --stdin
    name: cloudinitdisk
status: {}
```

8.14.7. 管理客户机内存

如果要调整客户机内存设置以适应特定用例，可通过编辑客户机的 YAML 配置文件来实现。OpenShift Virtualization 可用于配置客户机内存过量使用，以及禁用客户机内存开销核算。

警告

以下流程会增加虚拟机进程因内存压力而被终止的机率。只有当您了解风险时方可继续。

8.14.7.1. 配置客户机内存过量使用

如果您的虚拟工作负载需要的内存超过可用内存，您可利用内存过量使用来为您的虚拟机实例（VMI）分配全部或大部分主机内存。启用"内存过量使用"意味着您可以最大程度利用通常为主机保留的资源。

例如，如果主机具有 32 Gb RAM，则可利用内存过量功能，运行 8 个每个分配了 4GB RAM 的虚拟机（VM）。该分配方案假设所有虚拟机不会同时使用分配给它们的所有内存。

重要

内存过量使用增加虚拟机进程因内存压力（OOM 终止）被终止的可能性。

虚拟机被 OOM 终止的可能性取决于您的具体配置、节点内存、可用 swap 空间、虚拟机内存消耗、使用内核相同的页面合并（KSM）以及其它因素。

流程

1. 要明确告知虚拟机实例它的可用内存超过集群请求内存，请编辑虚拟机配置文件，并将 **spec.domain.memory.guest** 设置为超过 **spec.domain.resources.requests.memory** 的值。这一过程即为"内存过量使用"。
   在本例中，对集群的请求内存为 **1024M**，但虚拟机实例被告知它有 **2048M** 可用。只要相关的节点上有足够的可用内存，虚拟机实例最多可消耗 2048M 内存。

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

注意

如果节点面临内存压力，则适用于 pod 的驱除规则也适用于虚拟机实例。

2. 创建虚拟机：

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

8.14.7.2. 禁用客户机内存开销核算

除了您所请求的内存量之外，每个虚拟机实例还会额外请求少量内存。这部分额外内存将用于打包每个 **VirtualMachineInstance** 进程的基础结构。

虽然通常不建议这么设置，但可以通过禁用客户机内存开销核算来提高节点上的虚拟机实例密度。

重要

禁用客户机内存开销核算会增加虚拟机进程因内存压力（OOM 被终止）被终止的可能性。

虚拟机被 OOM 终止的可能性取决于您的具体配置、节点内存、可用 swap 空间、虚拟机内存消耗、使用内核相同的页面合并（KSM）以及其它因素。

流程

1. 要禁用客户机内存开销核算，请编辑 YAML 配置文件并将 **overcommitGuestOverhead** 值设置为 **true**。默认禁用此参数。

```
kind: VirtualMachine
spec:
  template:
    domain:
    resources:
      overcommitGuestOverhead: true
      requests:
        memory: 1024M
```

注意

如果启用 **overcommitGuestOverhead**，则会在内存限值中添加客户机开销（如果存在）。

2. 创建虚拟机：

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

8.14.8. 在虚拟机中使用巨页

您可以使用巨页作为集群中虚拟机的后备内存。

8.14.8.1. 先决条件

- 节点必须配置预先分配的巨页。

8.14.8.2. 巨页的作用

内存在块（称为页）中进行管理。在大多数系统中，页的大小为 4Ki。1Mi 内存相当于 256 个页，1Gi 内存相当于 256,000 个页。CPU 有内置的内存管理单元，可在硬件中管理这些页的列表。Translation Lookaside Buffer (TLB) 是虚拟页到物理页映射的小型硬件缓存。如果在硬件指令中包括的虚拟地址可以

在 TLB 中找到，则其映射信息可以被快速获得。如果没有包括在 TLN 中，则称为 TLB miss。系统将会使用基于软件的，速度较慢的地址转换机制，从而出现性能降低的问题。因为 TLB 的大小是固定的，因此降低 TLB miss 的唯一方法是增加页的大小。

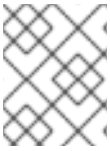
巨页指一个大于 4Ki 的内存页。在 x86_64 构架中，有两个常见的巨页大小: 2Mi 和 1Gi。在其它构架上的大小会有所不同。要使用巨页，必须写相应的代码以便应用程序了解它们。Transparent Huge Pages（THP）试图在应用程序不需要了解的情况下自动管理巨页，但这个技术有一定的限制。特别是，它的页大小会被限为 2Mi。当有较高的内存使用率时，THP 可能会导致节点性能下降，或出现大量内存碎片（因为 THP 的碎片处理）导致内存页被锁定。因此，有些应用程序可能更适用于（或推荐）使用预先分配的巨页，而不是 THP。

在 OpenShift Virtualization 中，可将虚拟机配置为消耗预先分配的巨页。

8.14.8.3. 为虚拟机配置巨页

您可以在虚拟机配置中包括 **memory.hugepages.pageSize** 和 **resources.requests.memory** 参数来配置虚拟机来使用预分配的巨页。

内存请求必须按页大小分离。例如，您不能对大小为 **1Gi** 的页请求 **500Mi** 内存。



注意

主机的内存布局和客户端操作系统不相关。虚拟机清单中请求的巨页适用于 QEMU。客户端中的巨页只能根据虚拟机实例的可用内存量来配置。

如果您编辑了正在运行的虚拟机，则必须重启虚拟机才能使更改生效。

先决条件

- 节点必须配置预先分配的巨页。

流程

1. 在虚拟机配置中，把 **resources.requests.memory** 和 **memory.hugepages.pageSize** 参数添加到 **spec.domain**。以下配置片段适用于请求总计 **4Gi** 内存的虚拟机，页面大小为 **1Gi**:

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

(1) 为虚拟机请求的总内存量。这个值必须可以被按页大小整除。

(2) 每个巨页的大小。x86_64 架构的有效值为 **1Gi** 和 **2Mi**。页面大小必须小于请求的内存。

2. 应用虚拟机配置：

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

8.14.9. 为虚拟机启用专用资源

要提高性能，您可以将节点的资源（如 CPU）专用于特定的一个虚拟机。

8.14.9.1. 关于专用资源

当为您的虚拟机启用专用资源时，您的工作负载将会在不会被其他进程使用的 CPU 上调度。通过使用专用资源，您可以提高虚拟机性能以及延迟预测的准确性。

8.14.9.2. 先决条件

- 节点上必须配置 CPU Manager。在调度虚拟机工作负载前，请确认节点具有 **cpumanager = true** 标签。
- 虚拟机必须关机。

8.14.9.3. 为虚拟机启用专用资源

您可以在 **Details** 选项卡中为虚拟机启用专用资源。使用红帽模板或向导创建的虚拟机可使用专用资源启用。

流程

1. 从侧边菜单中选择 **Workloads → Virtual Machines**。
2. 选择一个虚拟机以打开 **Virtual Machine** 选项卡。
3. 点 **Details** 标签页。
4. 点 **Dedicated Resources** 项右面的铅笔标签打开 **Dedicated Resources** 窗口。
5. 选择 **Schedule this workload with dedicated resources (guaranteed policy)**。
6. 点 **Save**。

8.14.10. 调度虚拟机

在确保虚拟机的 CPU 模型和策略属性与节点支持的 CPU 模型和策略属性兼容的情况下，可在节点上调度虚拟机（VM）。

8.14.10.1. 了解策略属性

您可以指定策略属性和在虚拟机调度到节点上时匹配的 CPU 功能来调度虚拟机（VM）。为虚拟机指定的策略属性决定了如何在节点上调度该虚拟机。

策略属性	描述
force	VM 被强制调度到某个节点上。即使主机 CPU 不支持虚拟机的 CPU，也是如此。

策略属性	描述
require	在虚拟机没有使用特定 CPU 模型和功能规格配置时，应用于虚拟机的默认策略。如果节点没有配置为支持使用此默认策略属性或其他策略属性的 CPU 节点发现，则虚拟机不会调度到该节点上。主机 CPU 必须支持虚拟机的 CPU，或者虚拟机监控程序必须可以模拟支持的 CPU 模型。
optional	如果主机物理机器 CPU 支持该虚拟机，则虚拟机会被添加到节点。
disable	无法通过 CPU 节点发现调度虚拟机。
forbid	即使主机 CPU 支持该功能，且启用了 CPU 节点发现，也不会调度虚拟机。

8.14.10.2. 设置策略属性和 CPU 功能

您可以为每个虚拟机（VM）设置策略属性和 CPU 功能，以确保根据策略和功能在节点上调度该功能。验证您设置的 CPU 功能以确保主机 CPU 支持或者虚拟机监控程序模拟该功能。

流程

- 编辑虚拟机配置文件的 **domain** spec。以下示例设置了虚拟机实例（VMI）的 CPU 功能和 **require** 策略：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: myvmi
spec:
  domain:
    cpu:
      features:
      - name: apic (1)
        policy: require (2)
```

1. VM 或 VMI 的 CPU 功能名称。
2. VM 或 VMI 的策略属性。

8.14.10.3. 使用支持的 CPU 型号调度虚拟机

您可以为虚拟机（VM）或虚拟机实例（VMI）配置 CPU 型号，将其调度到支持其 CPU 模型的节点。

流程

- 编辑虚拟机配置文件的 **domain** spec。以下示例显示了为 VMI 定义的特定 CPU 模型：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstance
metadata:
```

```
  name: myvmi
spec:
  domain:
    cpu:
      model: Conroe 1
```

1 VMI 的 CPU 模型。

8.14.10.4. 使用主机模型调度虚拟机

当将虚拟机（VM）的 CPU 模型设置为 **host-model** 时，虚拟机会继承调度节点的 CPU 模型。

流程

- 编辑虚拟机配置文件的 **domain** spec。以下示例显示了为虚拟机实例（VMI）指定了 **host-model**：

```
apiVersion: kubevirt/v1alpha3
kind: VirtualMachineInstance
metadata:
  name: myvmi
spec:
  domain:
    cpu:
      model: host-model 1
```

1 继承调度节点的 CPU 模型的 VM 或 VMI。

8.14.11. 配置 PCI 透传

借助 Peripheral Component Interconnect（PCI）透传功能，您可以从虚拟机访问和管理硬件设备。配置 PCI 透传后，PCI 设备的功能就如同它们实际上附加到客户机操作系统上一样。

集群管理员可以使用 **oc** CLI 来公开和管理集群中允许在集群中使用的主机设备。

8.14.11.1. 关于为 PCI 透传准备主机设备

要使用 CLI 为 PCI 透传准备主机设备，请创建一个 **MachineConfig** 对象并添加内核参数，以启用输入输出内存管理单元（IOMMU）。将 PCI 设备绑定到虚拟功能 I/O（VFIO）驱动程序，然后通过编辑 **HyperConverged** 自定义资源（CR）的 **allowedHostDevices** 字段在集群中公开它。首次安装 OpenShift Virtualization Operator 时，**allowedHostDevices** 列表为空。

要使用 CLI 从集群中删除 PCI 主机设备，可从 **HyperConverged** CR 中删除 PCI 设备信息。

8.14.11.1.1. 添加内核参数以启用 IOMMU 驱动程序

要在内核中启用 IOMMU（Input-Output Memory Management Unit）驱动程序，请创建 **MachineConfig** 对象并添加内核参数。

先决条件

- 正常运行的 OpenShift 容器平台集群的管理特权。

- Intel 或 AMD CPU 硬件。
- 启用用于直接 I/O 扩展的 Intel 虚拟化技术或 BIOS 中的 AMD IOMMU（基本输入/输出系统）。

流程

1. 创建用于标识内核参数的 **MachineConfig** 对象。以下示例显示了 Intel CPU 的内核参数。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker 1
  name: 100-worker-iommu 2
spec:
  config:
    ignition:
      version: 3.2.0
  kernelArguments:
      - intel_iommu=on 3
...
```

1. 仅将新内核参数应用到 worker 节点。
2. **name** 表示此内核参数（100）在机器配置及其目的中的排名。如果您有 AMD CPU，请将内核参数指定为 **amd_iommu=on**。
3. 将内核参数标识为 Intel CPU 的 **intel_iommu**。

2. 创建新的 **MachineConfig** 对象：

```
$ oc create -f 100-worker-kernel-arg-iommu.yaml
```

验证

- 验证是否添加了新的 **MachineConfig** 对象。

```
$ oc get MachineConfig
```

8.14.11.1.2. 将 PCI 设备绑定到 VFIO 驱动程序

要将 PCI 设备绑定到 VFIO（虚拟功能 I/O）驱动程序，请从每个设备获取 **vendor-ID** 和 **device-ID** 的值，并创建值的列表。将这个列表添加到 **MachineConfig** 对象。**MachineConfig** Operator 在带有 PCI 设备的节点上生成 **/etc/modprobe.d/vfio.conf**，并将 PCI 设备绑定到 VFIO 驱动程序。

先决条件

- 您添加了内核参数来为 CPU 启用 IOMMU。

流程

1. 运行 **lspci** 命令，以获取 PCI 设备的 **vendor-ID** 和 **device-ID**。

```
$ lspci -nnv | grep -i nvidia
```

输**出示例**

```
02:01.0 3D controller [0302]: NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB]
[10de:1eb8] (rev a1)
```

2. 创建 Butane 配置文件 **100-worker-vfiopci.bu**，将 PCI 设备绑定到 VFIO 驱动程序。

注意

有关 Butane 的信息，请参阅"使用 Butane 创建机器配置"。

示例

```
variant: openshift
version: 4.8.0
metadata:
  name: 100-worker-vfiopci
  labels:
    machineconfiguration.openshift.io/role: worker 1
storage:
  files:
  - path: /etc/modprobe.d/vfio.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        options vfio-pci ids=10de:1eb8 2
  - path: /etc/modules-load.d/vfio-pci.conf 3
    mode: 0644
    overwrite: true
    contents:
      inline: vfio-pci
```

1 仅将新内核参数应用到 worker 节点。

2 指定之前确定的 **vendor-ID** 值（**10de**）和 **device-ID** 值（**1eb8**）来将单个设备绑定到 VFIO 驱动程序。您可以使用其供应商和设备信息添加多个设备列表。

3 在 worker 节点上载入 vfio-pci 内核模块的文件。

3. 使用 Butane 生成 **MachineConfig** 对象文件 **100-worker-vfiopci.yaml**，包含要发送到 worker 节点的配置：

```
$ butane 100-worker-vfiopci.bu -o 100-worker-vfiopci.yaml
```

4. 将 **MachineConfig** 对象应用到 worker 节点：

```
$ oc apply -f 100-worker-vfiopci.yaml
```

5. 验证 **MachineConfig** 对象是否已添加。

```
$ oc get MachineConfig
```

输**出示例**

```
NAME                             GENERATEDBYCONTROLLER                      IGNITIONVERSION
AGE
00-master                        d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0             25h
00-worker                        d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0             25h
01-master-container-runtime      d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
01-master-kubelet                d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
01-worker-container-runtime      d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
01-worker-kubelet                d3da910bfa9f4b599af4ed7f5ac270d55950a3a1   3.2.0
25h
100-worker-iommu                                                            3.2.0             30s
100-worker-vfiopci-configuration                                            3.2.0             30s
```

验证

- 验证是否已加载 VFIO 驱动程序。

```
$ lspci -nnk -d 10de:
```

输出确认使用了 VFIO 驱动程序。

输**出示例**

```
04:00.0 3D controller [0302]: NVIDIA Corporation GP102GL [Tesla P40] [10de:1eb8] (rev a1)
        Subsystem: NVIDIA Corporation Device [10de:1eb8]
        Kernel driver in use: vfio-pci
        Kernel modules: nouveau
```

8.14.11.1.3. 使用 CLI 在集群中公开 PCI 主机设备

要在集群中公开 PCI 主机设备，将 PCI 设备的详细信息添加到 **HyperConverged** 自定义资源（CR）的 **spec.permittedHostDevices.pciHostDevices** 数组中。

流程

1. 运行以下命令，在默认编辑器中编辑 **HyperConverged** CR：

```
$ oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

2. 将 PCI 设备信息添加到 **spec.percommitHostDevices.pciHostDevices** 数组。例如：

配置文件示例

```
apiVersion: hco.kubevirt.io/v1
kind: HyperConverged
```

```
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  permittedHostDevices: 1
    pciHostDevices: 2
    - pciDeviceSelector: "10DE:1DB6" 3
      resourceName: "nvidia.com/GV100GL_Tesla_V100" 4
    - pciDeviceSelector: "10DE:1EB8"
      resourceName: "nvidia.com/TU104GL_Tesla_T4"
    - pciDeviceSelector: "8086:6F54"
      resourceName: "intel.com/qat"
      externalResourceProvider: true 5
...
```

1. 允许在集群中使用的主机设备。
2. 节点上可用的 PCI 设备列表。
3. 标识 PCI 设备所需的 **vendor-ID** 和 **device-ID**。
4. PCI 主机设备的名称。
5. 可选：将此字段设置为 **true** 表示资源由外部设备插件提供。OpenShift Virtualization 允许在集群中使用这个设备，但会把分配和监控留给外部设备插件。

注意

上例代码片段显示有两个 PCI 主机设备，名为 **nvidia.com/GV100GL_Tesla_V100** 和 **nvidia.com/TU104GL_Tesla_Tesla_T4**。它们被添加到 **HyperConverged** CR 中的允许主机设备列表中。这些设备已经过测试和验证以用于 OpenShift Virtualization。

3. 保存更改并退出编辑器。

验证

- 运行以下命令，验证 PCI 主机设备是否已添加到节点。示例输出显示，每个设备都与 **nvidia.com/GV100GL_Tesla_V100**、**nvidia.com/TU104GL_Tesla_T4** 和 **intel.com/qat** 资源名称关联。

```
$ oc describe node <node_name>
```

输出示例

```
Capacity:
  cpu:                64
  devices.kubevirt.io/kvm:    110
  devices.kubevirt.io/tun:    110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:          915128Mi
  hugepages-1Gi:              0
```

```
  hugepages-2Mi:                0
  memory:                       131395264Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                1
  pods:                         250
Allocatable:
  cpu:                          63500m
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            863623130526
  hugepages-1Gi:                0
  hugepages-2Mi:                0
  memory:                       130244288Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                1
  pods:                         250
```

8.14.11.1.4. 使用 CLI 从集群中删除 PCI 主机设备

要从集群中删除 PCI 主机设备，请从 **HyperConverged** 自定义资源（CR）中删除该设备的信息。

流程

1. 运行以下命令，在默认编辑器中编辑 **HyperConverged** CR：

```
$ oc edit hyperconverged kubevirt-hyperconverged -n openshift-cnv
```

2. 通过删除相应设备的 **pciDeviceSelector**、**resourceName** 和 **externalResourceProvider**（如果适用）字段来从 **spec.permittedHostDevices.pciHostDevices** 阵列中删除 PCI 设备信息。在本例中，**intel.com/qat** 资源已被删除。

   配置文件示例

```
apiVersion: hco.kubevirt.io/v1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  permittedHostDevices:
    pciHostDevices:
    - pciDeviceSelector: "10DE:1DB6"
      resourceName: "nvidia.com/GV100GL_Tesla_V100"
    - pciDeviceSelector: "10DE:1EB8"
      resourceName: "nvidia.com/TU104GL_Tesla_T4"
...
```

3. 保存更改并退出编辑器。

验证

- 运行以下命令，验证 PCI 主机设备已从节点移除。示例输出显示，与 **intel.com/qat** 资源名称关联的设备为零。

```
$ oc describe node <node_name>
```

输出示例

```
Capacity:
  cpu:                          64
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            915128Mi
  hugepages-1Gi:                0
  hugepages-2Mi:                0
  memory:                       131395264Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                0
  pods:                         250
Allocatable:
  cpu:                          63500m
  devices.kubevirt.io/kvm:      110
  devices.kubevirt.io/tun:      110
  devices.kubevirt.io/vhost-net:  110
  ephemeral-storage:            863623130526
  hugepages-1Gi:                0
  hugepages-2Mi:                0
  memory:                       130244288Ki
  nvidia.com/GV100GL_Tesla_V100   1
  nvidia.com/TU104GL_Tesla_T4     1
  intel.com/qat:                0
  pods:                         250
```

8.14.11.2. 为 PCI 透传配置虚拟机

将 PCI 设备添加到集群中后，您可以将它们分配到虚拟机。PCI 设备现在可用。就像它们被物理地连接到虚拟机一样。

8.14.11.2.1. 为虚拟机分配 PCI 设备

当集群中有 PCI 设备时，您可以将其分配到虚拟机并启用 PCI 透传。

流程

- 将 PCI 设备分配到虚拟机作为主机设备。

示例

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
spec:
  domain:
    devices:
```

```
hostDevices:
- deviceName: nvidia.com/TU104GL_Tesla_T4 1
  name: hostdevices1
```

1 集群中作为主机设备允许的 PCI 设备的名称。虚拟机可以访问此主机设备。

验证

- 使用以下命令，验证主机设备可从虚拟机使用。

```
$ lspci -nnk | grep NVIDIA
```

输出示例

```
$ 02:01.0 3D controller [0302]: NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB]
[10de:1eb8] (rev a1)
```

8.14.11.3. 其他资源

- 在 BIOS 中启用 Intel VT-X 和 AMD-V 虚拟化硬件扩展
- 管理文件权限
- 安装后机器配置任务

8.14.12. 配置 watchdog

通过为 watchdog 设备配置虚拟机（VM）、安装 watchdog 并启动 watchdog 服务来公开 watchdog。

8.14.12.1. 先决条件

- 虚拟机必须具有对 **i6300esb** watchdog 设备的内核支持。Red Hat Enterprise Linux (RHEL) 镜像支持 **i6300esb**。

8.14.12.2. 定义 watchdog 设备

定义在操作系统（OS）没有响应时 watchdog 如何进行处理。

表 8.3. 可能的操作

poweroff	虚拟机（VM）立即关闭。如果 **spec.running** 设为 **true**，或者 **spec.runStrategy** 没有设置为 **manual**，则 VM 会重启。
reset	虚拟机将进行重启，客户机操作系统无法响应。由于客户机操作系统重启所需一定的时间完成，可能会导致存活度探测超时，因此不建议使用这个选项。如果集群级别的保护发现存活度探测失败并强制重新调度存活度探测，则此超时可以延长虚拟机重启所需的时间。
shutdown	虚拟机通过停止所有服务来正常关闭电源。

流程

1. 创建包含以下内容的 YAML 文件：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm2-rhel84-watchdog
  name: <vm-name>
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm2-rhel84-watchdog
    spec:
      domain:
        devices:
          watchdog:
            name: <watchdog>
            i6300esb:
              action: "poweroff" (1)
...
```

(1) 指定 **watchdog** 操作（**poweroff**、**reset** 或 **shutdown**）。

上面的示例使用 poweroff 操作配置 RHEL8 虚拟机上的 **i6300esb** watchdog 设备，并将设备公开为 **/dev/watchdog**。

现在，watchdog 二进制文件可以使用这个设备。

2. 运行以下命令，将 YAML 文件应用到集群：

```
$ oc apply -f <file_name>.yaml
```

重要

此流程仅用于测试 watchdog 功能，且不得在生产环境中运行。

1. 运行以下命令来验证虚拟机是否已连接到 watchdog 设备：

```
$ lspci | grep watchdog -i
```

2. 运行以下命令之一以确认 watchdog 处于活跃状态：

- 触发内核 panic：

```
# echo c > /proc/sysrq-trigger
```

- 终止 watchdog 服务：

```
# pkill -9 watchdog
```

8.14.12.3. 安装 watchdog 设备

在虚拟机上安装 **watchdog** 软件包，再启动 watchdog 服务。

流程

1. 作为 root 用户，安装 **watchdog** 软件包和依赖项：

```
# yum install watchdog
```

2. 在 **/etc/watchdog.conf** 文件中取消注释以下行，并保存更改：

```
#watchdog-device = /dev/watchdog
```

3. 在引导时启用 watchdog 服务：

```
# systemctl enable --now watchdog.service
```

8.14.12.4. 其他资源

- 使用健康检查来监控应用程序的健康状态

8.15. 导入虚拟机

8.15.1. 数据卷导入的 TLS 证书

8.15.1.1. 添加用于身份验证数据卷导入的 TLS 证书

registry 或 HTTPS 端点的 TLS 证书必须添加到配置映射中，才能从这些源导入数据。此配置映射必须存在于目标数据卷的命名空间中。

通过引用 TLS 证书的相对文件路径来创建配置映射。

流程

1. 确定您处于正确的命名空间中。配置映射只能被数据卷引用（如果位于同一命名空间中）。

```
$ oc get ns
```

2. 创建配置映射：

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

8.15.1.2. 示例：从 TLS 证书创建的配置映射

以下示例是从 **ca.pem** TLS 证书创建的配置映射。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: tls-certs
```

```
data:
  ca.pem: |
    -----BEGIN CERTIFICATE-----
    ... <base64 encoded cert> ...
    -----END CERTIFICATE-----
```

8.15.2. 使用数据卷导入虚拟机镜像

使用 Containerized Data Importer（CDI）通过使用数据卷将虚拟机镜像导入到持久性卷声明（PVC）中。您可以将数据卷附加到虚拟机以获取持久性存储。

虚拟机镜像可以托管在 HTTP 或 HTTPS 端点上，也可以内嵌在容器磁盘中，并存储在容器镜像仓库中。

重要

当您将磁盘镜像导入到 PVC 中时，磁盘镜像扩展为使用 PVC 中请求的全部存储容量。要使用该空间，可能需要扩展虚拟机中的磁盘分区和文件系统。

调整大小的流程因虚拟机上安装的操作系统而异。详情请查看操作系统文档。

8.15.2.1. 先决条件

- 如果端点需要 TLS 证书，该证书必须 包含在与数据卷相同的命名空间中的配置映射 中，并在数据卷配置中引用。
- 导入容器磁盘：
  - 您可能需要从虚拟机镜像准备容器磁盘，并在导入前将其存储在容器镜像仓库中。
  - 如果容器镜像仓库没有 TLS，您必须将 registry 添加到 **HyperConverged** 自定义资源的 **insecureRegistries** 字段中，然后才能从中导入容器磁盘。
- 您可能需要定义存储类或准备 CDI 涂销空间才能成功完成此操作。

8.15.2.2. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

☐ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

注意

CDI 现在使用 OpenShift Container Platform 集群范围的代理配置。

8.15.2.3. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.15.2.4. 使用数据卷将虚拟机镜像导入到存储中

您可以使用数据卷将虚拟机镜像导入到存储中。

虚拟机镜像可以托管在 HTTP 或 HTTPS 端点上，或者镜像可以构建到容器磁盘中，并存储在容器镜像仓库中。

您可以在 **VirtualMachine** 配置文件中为镜像指定数据源。创建虚拟机时，包含虚拟机镜像的数据卷将导入到存储中。

先决条件

- 要导入虚拟机镜像，必须有以下内容：
  - RAW、ISO 或 QCOW2 格式的虚拟机磁盘镜像，可选择使用 **xz** 或 **gz** 进行压缩。
  - 托管镜像的 HTTP 或 HTTPS 端点，以及访问数据源所需的任何身份验证凭证。
- 要导入容器磁盘，您必须将虚拟机镜像构建到容器磁盘中，并存储在容器镜像仓库中，以及访问数据源所需的任何身份验证凭证。
- 如果虚拟机必须与未由系统 CA 捆绑包签名的证书的服务器通信，则必须在与数据卷相同的命名空间中创建一个配置映射。

流程

1. 如果您的数据源需要身份验证，请创建一个 **Secret** 清单，指定数据源凭证，并将其保存为 **endpoint-secret.yaml** ：

```
apiVersion: v1
kind: Secret
metadata:
  name: endpoint-secret 1
  labels:
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: "" 2
  secretKey:   "" 3
```

1. 指定 **Secret** 的名称。
2. 指定以 Base64 编码的密钥 ID 或用户名。
3. 指定以 Base64 编码的 secret 密钥或密码。

2. 应用 **Secret** 清单：

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

3. 编辑 **VirtualMachine** 清单，为要导入的虚拟机镜像指定数据源，并将其保存为 **vm-fedora-datavolume.yaml** ：

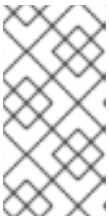
```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  creationTimestamp: null
  labels:
    kubevirt.io/vm: vm-fedora-datavolume
  name: vm-fedora-datavolume (1)
spec:
  dataVolumeTemplates:
  - metadata:
      creationTimestamp: null
      name: fedora-dv (2)
    spec:
      storage:
        resources:
          requests:
            storage: 10Gi
        storageClassName: local
      source:
        http: (3)
          url: "https://mirror.arizona.edu/fedora/linux/releases/35/Cloud/x86_64/images/Fedora-Cloud-Base-35-1.2.x86_64.qcow2" (4)
          secretRef: endpoint-secret (5)
          certConfigMap: "" (6)
    status: {}
  running: true
  template:
    metadata:
      creationTimestamp: null
      labels:
        kubevirt.io/vm: vm-fedora-datavolume
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: datavolumedisk1
        machine:
          type: ""
        resources:
```

```
        requests:
          memory: 1.5Gi
      terminationGracePeriodSeconds: 60
      volumes:
      - dataVolume:
          name: fedora-dv
        name: datavolumedisk1
status: {}
```

1. 指定虚拟机的名称。
2. 指定数据卷的名称。
3. 为 HTTP 或 HTTPS 端点指定 **http**。为从 **registry** 导入的容器磁盘镜像指定 registry。
4. 要导入的虚拟机镜像的源。本例引用了 HTTPS 端点上的虚拟机镜像。容器镜像仓库端点示例为 **url: "docker://kubevirt/fedora-cloud-container-disk-demo:latest"**。
5. 如果您为数据源创建 **Secret**，则需要此项。
6. 可选：指定一个 CA 证书配置映射。

4. 创建虚拟机：

```
$ oc create -f vm-fedora-datavolume.yaml
```



注意

oc create 命令创建数据卷和虚拟机。CDI 控制器创建一个带有正确注解和导入过程的底层 PVC。导入完成后，数据卷状态变为 **Succeeded**。您可以启动虚拟机。

数据卷置备在后台进行，因此无需监控进程。

验证

1. importer pod 从指定的 URL 下载虚拟机镜像或容器磁盘，并将其存储在置备的 PV 上。运行以下命令，查看 importer pod 的状态：

```
$ oc get pods
```

2. 运行以下命令监控数据卷，直到其状态为 **Succeeded** ：

```
$ oc describe dv fedora-dv 1
```

1. 指定您在 **VirtualMachine** 清单中定义的数据卷名称。

3. 通过访问其串行控制台来验证置备是否已完成，并且虚拟机是否已启动：

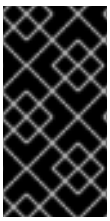
```
$ virtctl console vm-fedora-datavolume
```

8.15.2.5. 其他资源

- 配置预分配模式以提高数据卷操作的写入性能。

8.15.3. 使用数据卷将虚拟机镜像导入到块存储中

您可将现有虚拟机镜像导入到您的 OpenShift Container Platform 集群中。OpenShift Virtualization 使用数据卷自动导入数据并创建底层持久性卷声明（PVC）。



重要

当您将磁盘镜像导入到 PVC 中时，磁盘镜像扩展为使用 PVC 中请求的全部存储容量。要使用该空间，可能需要扩展虚拟机中的磁盘分区和文件系统。

调整大小的流程因虚拟机上安装的操作系统而异。详情请查看操作系统文档。

8.15.3.1. 先决条件

- 如果您根据 CDI 支持的操作列表要求涂销空间，您必须首先定义一个 StorageClass 或准备 CDI 涂销空间才能成功完成此操作。

8.15.3.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.15.3.3. 关于块持久性卷

块持久性卷（PV）是一个受原始块设备支持的 PV。这些卷没有文件系统，可以通过降低开销来为虚拟机提供性能优势。

原始块卷可以通过在 PV 和持久性卷声明（PVC）规格中指定 **volumeMode: Block** 来置备。

8.15.3.4. 创建本地块持久性卷

通过填充文件并将其挂载为循环设备，在节点上创建本地块持久性卷（PV）。然后，您可以在 PV 清单中将该循环设备作为 **Block（块）**卷引用，并将其用作虚拟机镜像的块设备。

流程

1. 以 **root** 身份登录节点，在其上创建本地 PV。本流程以 **node01** 为例。
2. 创建一个文件并用空字符填充，以便可将其用作块设备。以下示例创建 **loop10** 文件，大小为 2Gb（20,100 Mb 块）：

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

3. 将 **loop10** 文件挂载为 loop 设备。

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

1. 挂载 loop 设备的文件路径。
2. 上一步中创建的文件，挂载为 loop 设备。

4. 创建引用所挂载 loop 设备的 **PersistentVolume** 清单。

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <local-block-pv10>
  annotations:
spec:
  local:
    path: </dev/loop10> 1
  capacity:
    storage: <2Gi>
  volumeMode: Block 2
  storageClassName: local 3
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - <node01> 4
```

1. 节点上的 loop 设备路径。
2. 将其指定为块 PV。
3. 可选：为 PV 设置存储类。如果省略此项，将使用默认集群。
4. 挂载块设备的节点。

5. 创建块 PV。

```
# oc create -f <local-block-pv10.yaml> 1
```

1. 上一步中创建的持久性卷的文件名。

8.15.3.5. 使用数据卷将虚拟机镜像导入到块存储中

您可以使用数据卷将虚拟机镜像导入到块存储中。在创建虚拟机前，您要在 **VirtualMachine** 清单中引用数据卷。

先决条件

- RAW、ISO 或 QCOW2 格式的虚拟机磁盘镜像，可选择使用 **xz** 或 **gz** 进行压缩。
- 托管镜像的 HTTP 或 HTTPS 端点，以及访问数据源所需的任何身份验证凭证。

流程

1. 如果您的数据源需要身份验证，请创建一个 **Secret** 清单，指定数据源凭证，并将其保存为 **endpoint-secret.yaml** ：

```
apiVersion: v1
kind: Secret
metadata:
  name: endpoint-secret 1
  labels:
    app: containerized-data-importer
type: Opaque
data:
  accessKeyId: "" 2
  secretKey:   "" 3
```

1. 指定 **Secret** 的名称。
2. 指定以 Base64 编码的密钥 ID 或用户名。
3. 指定以 Base64 编码的 secret 密钥或密码。

2. 应用 **Secret** 清单：

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

3. 创建 **DataVolume** 清单，为虚拟机镜像指定数据源，并为 **storage.volumeMode** 指定 **Block**。

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: import-pv-datavolume 1
spec:
  storageClassName: local 2
    source:
      http:
        url: "https://mirror.arizona.edu/fedora/linux/releases/35/Cloud/x86_64/images/Fedora-Cloud-Base-35-1.2.x86_64.qcow2" 3
        secretRef: endpoint-secret 4
  storage:
    volumeMode: Block 5
    resources:
      requests:
        storage: 10Gi
```

1. 指定数据卷的名称。
2. 可选：设置存储类，或忽略此项，接受集群默认值。
3. 指定要导入的镜像的 HTTP 或 HTTPS URL。
4. 如果您为数据源创建 **Secret**，则需要此项。
5. 对于已知的存储置备程序，会自动检测到卷模式和访问模式。否则，指定 **Block**。

4. 创建数据卷来导入虚拟机镜像：

```
$ oc create -f import-pv-datavolume.yaml
```

在创建虚拟机前，您可以在 **VirtualMachine** 清单中引用此数据卷。

8.15.3.6. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

☐ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

注意

CDI 现在使用 OpenShift Container Platform 集群范围的代理配置。

8.15.3.7. 其他资源

- 配置预分配模式以提高数据卷操作的写入性能。

8.15.4. 导入单个 Red Hat Virtualization 虚拟机

您可以使用 VM Import 向导或 CLI 将单个 Red Hat Virtualization（RHV）虚拟机导入到 OpenShift Virtualization。

重要

导入 RHV VM 是一个已弃用的功能。弃用的功能仍然包含在 OpenShift Virtualization 中，并且仍然被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

有关 OpenShift Virtualization 中已弃用或删除的主要功能的最新列表，请参阅 OpenShift Virtualization 发行注记中 *已弃用和删除的功能* 部分。

这个功能将被虚拟化的 Migration Toolkit 替代。

8.15.4.1. OpenShift Virtualization 存储功能列表

下表描述了支持导入虚拟机的 OpenShift Virtualization 存储类型。

表 8.4. OpenShift Virtualization 存储功能列表

	RHV 虚拟机导入
OpenShift Container Storage: RBD 块模式卷	是
OpenShift Virtualization hostpath 置备程序	不是
其他多节点可写入存储	是 [1]
其他单节点可写入存储	是 [2]

1. PVC 必须请求 ReadWriteMany 访问模式。
2. PVC 必须请求 ReadWriteOnce 访问模式。

8.15.4.2. 导入虚拟机的先决条件

将 Red Hat Virtualization 虚拟机导入到 OpenShift Virtualization 需要以下先决条件：

- 具有 admin 用户权限。
- 存储：
  - OpenShift Virtualization 本地和共享的持久性存储类必须支持虚拟机导入。
  - 如果您使用 Ceph RBD 块模式卷，且可用的存储空间对于虚拟磁盘太小，则导入过程会在完成 75% 的时候停止 20 分钟以上，且迁移也不会成功。web 控制台中没有显示错误消息。BZ#1910019
- 网络：
  - RHV 和 OpenShift Virtualization 网络必须具有相同的名称或被相互映射。
  - RHV VM 网络接口必须是 **e1000**、**rtl8139** 或 **virtio**。
- VM 磁盘：
  - 磁盘接口必须是 **sata**、**virtio_scsi** 或者 **virtio**。
  - 不可将该磁盘配置为直接 LUN。
  - 磁盘状态不得是 **illegal** 或 **locked**。
  - 存储类型必须是 **image**。
  - 必须禁用 SCSI 保留功能。
  - **ScsiGenericIO** 必需被禁用。
- VM 配置：

- 如果 VM 使用 GPU 资源，则必须配置提供 GPU 的节点。
- 不能为 vGPU 资源配置 VM。
- 虚拟机不能具有处于 **illegal** 状态的磁盘的快照。
- 虚拟机一定不能使用 OpenShift Container Platform 创建，并随后添加到 RHV。
- 虚拟机不能为 USB 设备配置。
- watchdog 不能是 **diag288**。

8.15.4.3. 使用 VM 导入向导（Import wizard）导入虚拟机

您可以使用 VM 导入向导导入单个虚拟机。

流程

1. 在 web 控制台中，点 **Workloads → Virtual Machines**。
2. 点击 **Create Virtual Machine**并选择 **Import with Wizard**。
3. 从 **Provider** 列表中选择 **Red Hat Virtualization (RHV)**。
4. 选择 **Connect to New Instance**，或一个保存的 RHV 实例。
   - 如果选择 **Connect to New Instance**，请填写以下字段：
     - **API URL**：例如 **https://<RHV_Manager_FQDN>/ovirt-engine/api**
     - **CA certificate**：点 **Browse** 上传 RHV Manager CA 证书或将 CA 证书粘贴到字段。您可以运行以下命令来查看 CA 证书：

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

       CA 证书是输出中的第二个证书。
     - **Username**：RHV Manager 的用户名，例如 **ocpadmin@internal**
     - **Password**: RHV Manager 密码
   - 如果您选择了一个保存的 RHV 实例，向导将使用保存的凭证连接到 RHV 实例。
5. 点击 **Check and Save**，然后等待连接完成。

注意

连接详情存储在 secret 中。如果您提供的供应商带有不正确的 URL、用户名或密码，点 **Workloads → Secrets** 并删除供应商 secret。

6. 选择一个集群和一个虚拟机。
7. 点 **Next**。
8. 在 **Review** 屏幕中，查看您的设置。

9. 可选：您可以选择 **Start virtual machine on creation**。
10. 点 **Edit** 以更新以下设置：
    - **General → Name**：虚拟机名称限制为 63 个字符。
    - **General → Description**: 虚拟机的描述信息（可选）。
        - **Storage Class**：选择 **NFS** 或 **ocs-storagecluster-ceph-rbd**。
          如果选择 **ocs-storagecluster-ceph-rbd**，您必须将磁盘的 **Volume Mode** 设置为 **Block**。
        - **advanced → Volume Mode**: 选择 **Block**。
    - **advanced → Volume Mode**: 选择 **Block**。
    - **networking → Network**：您可以从可用网络附加定义对象列表中选择网络。
11. 如果您编辑了导入设置，点 **Import** 或 **Review and Import**。
    此时会显示 **Successfully created virtual machine** 消息以及为虚拟机创建的资源列表。虚拟机会出现在 **Workloads → Virtual Machines** 中。

虚拟机向导字段

名称	参数	描述
名称		名称可包含小写字母 (**a-z**)、数字 (**0-9**) 和连字符 (**-**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格、句点 (**.**) 或特殊字符。
描述		可选的描述字段。
操作系统		模板中为虚拟机选择的操作系统。从模板创建虚拟机时，您无法编辑此字段。
引导源	通过 URL 导入（创建 PVC）	从 HTTP 或 HTTPS 端点提供的镜像导入内容。示例：包含操作系统镜像的网页中的 URL 链接。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有持久性卷声明并克隆它。
	通过 Registry 导入（创建 PVC）	从可通过集群访问的注册表中的可启动操作系统容器置备虚拟机。示例：***kubevirt/cirros-registry-disk-demo***。
	PXE（网络引导 - 添加网络接口）	从网络的服务器引导操作系统。需要一个 PXE 可引导网络附加定义。

名称	参数	描述
持久性卷声明项目		用于克隆 PVC 的项目名称。
持久性卷声明名称		如果您要克隆现有的 PVC，则应用于此虚拟机模板的 PVC 名称。
将它作为光盘引导源挂载		CD-ROM 需要额外的磁盘来安装操作系统。选择添加磁盘的选择框并稍后进行自定义。
Flavor	tiny、small、Medium、Large、Custom	根据与该模板关联的操作系统，在虚拟机模板中预设具有预定义值的 CPU 和内存量。 如果选择了默认模板，您可以使用自定义值覆盖模板中的 **cpus** 和 **memsize** 的值，以创建自定义模板。另外，您可以通过修改 **Workloads → Virtualization** 页面上的 **Details** 选项卡中的 **cpus** 和 **memsize** 值来创建自定义模板。
工作负载类型 **注意** 如果您选择了不正确的 **Workload Type**，则可能会出现性能或资源利用率问题（如缓慢的 UI）。	Desktop	用于桌面的虚拟机配置。适用于小型工作环境。建议与 Web 控制台搭配使用。
	Server	在性能和广泛的服务器工作负载兼容性方面具有最佳平衡。
	高性能	针对高性能负载进行了优化的虚拟机配置。
创建后启动此虚拟机。		默认选择这个复选框并在创建后启动虚拟机。如果您不希望虚拟机在创建时启动，请清除该复选框。

网络字段

名称	描述
名称	网络接口控制器的名称。
model	指明网络接口控制器的型号。支持的值有 **e1000e** 和 **virtio**。
网络	可用网络附加定义的列表。

名称	描述
类型	可用绑定方法列表。对于默认的 pod 网络，**masquerade** 是唯一推荐的绑定方法。对于辅助网络，请使用 **bridge** 绑定方法。非默认网络不支持 **masquerade** 绑定方法。如果您配置了一个 **SR-IOV** 网络设备并在命名空间中定义那个网络，请选择 SR-IOV。
MAC 地址	网络接口控制器的 MAC 地址。如果没有指定 MAC 地址，则会自动分配一个。

存储字段

名称	选择	描述
Source	空白（创建 PVC）	创建一个空磁盘。
	通过 URL 导入（创建 PVC）	通过 URL（HTTP 或 HTTPS 端点）导入内容。
	使用现有的 PVC	使用集群中已可用的 PVC。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有 PVC 并克隆它。
	通过 Registry 导入（创建 PVC）	通过容器 registry 导入内容。
	容器（临时）	从集群可以访问的 registry 中的容器上传内容。容器磁盘应只用于只读文件系统，如 CD-ROM 或临时虚拟机。
名称		磁盘的名称。名称可包含小写字母 (**a-z**)、数字 (**0-9**)、连字符 (**-**) 和句点 (**.**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格或特殊字符。
Size		GiB 中磁盘的大小。
类型		磁盘类型。示例：磁盘或光盘
Interface		磁盘设备的类型。支持的接口包括 **virtIO**、**SATA** 和 **SCSI**。
Storage class		用于创建磁盘的存储类。

名称	选择	描述
Advanced → Volume Mode		定义持久性卷是否使用格式化的文件系统或原始块状态。默认为 **Filesystem**。

高级存储设置

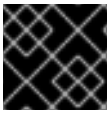
名称	参数	描述
卷模式	Filesystem	在基于文件系统的卷中保存虚拟磁盘。
	Block	直接将虚拟磁盘存储在块卷中。只有底层存储支持时才使用 **Block**。
访问模式 [1]	Single User (RWO)	这个卷可以被一个单一的节点以 read/write 的形式挂载。
	Shared Access (RWX)	卷可以被多个节点以读写模式挂载。
	Read Only (ROX)	卷可以被多个节点以只读形式挂载。

1. 您可以使用命令行界面更改访问模式。

8.15.4.4. 使用 CLI 导入虚拟机

您可以通过 CLI 创建 **Secret** 和 **VirtualMachineImport** 自定义资源（CR）来导入虚拟机。**Secret** CR 存储 RHV Manager 凭证和 CA 证书。**VirtualMachineImport** CR 定义 VM 导入过程的参数。

可选：您可以创建一个与 **VirtualMachineImport** CR 分开的 **ResourceMapping** CR。**ResourceMapping** CR 提供了更大的灵活性，例如导入额外的 RHV VM。



重要

默认目标存储类必须是 NFS。Cinder 不支持 RHV VM 导入。

流程

1. 运行以下命令来创建 **Secret** CR:

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: rhv-credentials
  namespace: default (1)
type: Opaque
stringData:
  ovirt: |
    apiUrl: <api_endpoint> (2)
    username: ocpadmin@internal
```

```
    password: 3
    caCert: |
      -----BEGIN CERTIFICATE-----
      4
      -----END CERTIFICATE-----
EOF
```

1. 可选。您可以在所有 CR 中指定不同的命名空间。
2. 指定 RHV Manager 的 API 端点，如 \"**https://www.example.com:8443/ovirt-engine/api**"。
3. 指定 **ocpadmin@internal** 的密码。
4. 指定 RHV Manager CA 证书。您可以运行以下命令来获取 CA 证书：

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

2. 可选：运行以下命令，创建 **ResourceMapping** CR，以将资源映射与 **VirtualMachineImport** CR 分开：

```
$ 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
```

1. 指定 RHV 逻辑网络和 vNIC 配置集。
2. 指定 OpenShift Virtualization 网络。
3. 如果在 **ResourceMapping** 和 **VirtualMachineImport** CR 中都指定了存储映射，则 **VirtualMachineImport** CR 将具有优先权。
4. 指定 RHV 存储域。
5. 指定 **nfs** 或 **ocs-storagecluster-ceph-rbd**。

6 如果指定了 **ocs-storagecluster-ceph-rbd** 存储类，则必须将 **Block** 指定为卷模式。

3. 运行以下命令来创建 **VirtualMachineImport** CR:

```
$ cat <<EOF | oc create -f -
apiVersion: v2v.kubevirt.io/v1beta1
kind: VirtualMachineImport
metadata:
  name: vm-import
  namespace: default
spec:
  providerCredentialsSecret:
    name: rhv-credentials
    namespace: default
# resourceMapping: (1)
#   name: resourcemapping-example
#   namespace: default
  targetVmName: vm_example (2)
  startVm: true
  source:
    ovirt:
      vm:
        id: <source_vm_id> (3)
        name: <source_vm_name> (4)
      cluster:
        name: <source_cluster_name> (5)
      mappings: (6)
        networkMappings:
          - source:
              name: <source_logical_network>/<vnic_profile> (7)
            target:
              name: <target_network> (8)
            type: pod
        storageMappings: (9)
          - source:
              name: <source_storage_domain> (10)
            target:
              name: <target_storage_class> (11)
            accessMode: <volume_access_mode> (12)
        diskMappings:
          - source:
              id: <source_vm_disk_id> (13)
            target:
              name: <target_storage_class> (14)
EOF
```

1 如果创建一个 **ResourceMapping** CR，取消 **resourceMapping** 部分的注释。

2 指定目标虚拟机名称。

3 指定源虚拟机 ID，例如 **80554327-0569-496b-bdeb-fcbbf52b827b**。您可以通过在 Manager 机器的网页浏览器中输入 **https://www.example.com/ovirt-engine/api/vms/** 来列出所有虚拟机来获取虚拟机 ID。找到您要导入的虚拟机及其对应的虚拟机 ID。您不需要指定

虚拟机名称或集群名称。

4 如果指定源虚拟机名称，还必须同时指定源集群。不要指定源虚拟机 ID。

5 如果指定源集群，还必须指定源虚拟机名称。不要指定源虚拟机 ID。

6 如果创建一个 **ResourceMapping** CR，注释掉 **mappings** 部分。

7 指定源虚拟机的逻辑网络和 vNIC 配置集。

8 指定 OpenShift Virtualization 网络。

9 如果在 **ResourceMapping** 和 **VirtualMachineImport** CR 中都指定了存储映射，则 **VirtualMachineImport** CR 将具有优先权。

10 指定源存储域。

11 指定目标存储类。

12 指定 **ReadWriteOnce**、**ReadWriteMany** 或 **ReadOnlyMany**。如果没有指定访问模式，{virt} 会根据 RHV VM 的 **Host → Migration 模式** 或虚拟磁盘访问模式决定正确的卷访问模式：

- 如果 RHV VM 迁移模式是 **Allow manual and automatic migration**，则默认访问模式为 **ReadWriteMany**。
- 如果 RHV 虚拟磁盘访问模式是 **ReadOnly**，则默认访问模式为 **ReadOnlyMany**。
- 对于所有其他设置，默认的访问模式为 **ReadWriteOnce**。

13 指定源虚拟机磁盘 ID，例如 **8181ecc1-5db8-4193-9c92-3ddab3be7b05**。您可以通过在 Manager 机器的网页浏览器中输入 **https://www.example.com/ovirt-engine/api/vms/vm23** 并查看虚拟机详情来获取磁盘 ID。

14 指定目标存储类。

4. 按照虚拟机导入的过程，以验证导入是否成功：

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

显示成功导入的输出结果类似如下：

输出示例

```
...
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"
    message: Copying virtual machine disks
    reason: CopyingDisks
    status: "True"
    type: Processing
  dataVolumes:
  - name: fedora32-b870c429-11e0-4630-b3df-21da551a48c0
  targetVmName: fedora32
```

8.15.4.4.1. 创建用于导入虚拟机的配置映射

如果要覆盖默认的 **vm-import-controller** 映射或添加额外的映射，您可以创建一个配置映射来将 Red Hat Virtualization（RHV）虚拟机操作系统映射到 OpenShift Virtualization 模板。

默认 **vm-import-controller** 配置映射包含以下 RHV 操作系统，及其对应的通用 OpenShift Virtualization 模板。

表 8.5. 操作系统和模板映射

RHV 虚拟机操作系统	OpenShift Virtualization 模板
rhel_6_10_plus_ppc64	**rhel6.10**
rhel_6_ppc64	**rhel6.10**
rhel_6	**rhel6.10**
rhel_6x64	**rhel6.10**
rhel_6_9_plus_ppc64	**rhel6.9**
rhel_7_ppc64	**rhel7.7**
rhel_7_s390x	**rhel7.7**
rhel_7x64	**rhel7.7**
rhel_8x64	**rhel8.1**
sles_11_ppc64	**opensuse15.0**
sles_11	**opensuse15.0**
sles_12_s390x	**opensuse15.0**
ubuntu_12_04	**ubuntu18.04**

RHV 虚拟机操作系统	OpenShift Virtualization 模板
ubuntu_12_10	ubuntu18.04
ubuntu_13_04	ubuntu18.04
ubuntu_13_10	ubuntu18.04
ubuntu_14_04_ppc64	ubuntu18.04
ubuntu_14_04	ubuntu18.04
ubuntu_16_04_s390x	ubuntu18.04
windows_10	win10
windows_10x64	win10
windows_2003	win10
windows_2003x64	win10
windows_2008R2x64	win2k8
windows_2008	win2k8
windows_2008x64	win2k8
windows_2012R2x64	win2k12r2
windows_2012x64	win2k12r2
windows_2016x64	win2k16
windows_2019x64	win2k19
windows_7	win10
windows_7x64	win10
windows_8	win10
windows_8x64	win10
windows_xp	win10

流程

1. 在网页浏览器中，进入 **http://<RHV_Manager_FQDN>/ovirt-engine/api/vms/<VM_ID>** 找到 RHV 虚拟机操作系统的 REST API 名称。操作系统名称会出现在 XML 输出的 **<os>** 部分，如下例所示：

```
...
<os>
...
<type>rhel_8x64</type>
</os>
```

2. 查看可用 OpenShift Virtualization 模板列表：

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

输**出示例**

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

3. 如果与 RHV 虚拟机操作系统匹配的 OpenShift Virtualization 模板没有出现在可用的模板列表中，使用 OpenShift Virtualization Web 控制台创建一个模板。

4. 创建配置映射将 RHV VM 操作系统映射到 OpenShift Virtualization 模板：

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: os-configmap
  namespace: default (1)
data:
  guestos2common: |
    "Red Hat Enterprise Linux Server": "rhel"
    "CentOS Linux": "centos"
    "Fedora": "fedora"
    "Ubuntu": "ubuntu"
    "openSUSE": "opensuse"
  osinfo2common: |
    "<rhv-operating-system>": "<vm-template>" (2)
EOF
```

(1) 可选：您可以更改 **namespace** 参数的值。

(2) 指定 RHV 操作系统的 REST API 名称及其对应的虚拟机模板，如下例所示。

配置映射示例

-

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: os-configmap
  namespace: default
data:
  osinfo2common: |
    "other_linux": "fedora31"
EOF
```

5. 验证是否已创建自定义配置映射：

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

6. 对 **vm-import-controller-config** 配置映射进行补丁以应用新的配置映射：

```
$ oc patch configmap vm-import-controller-config -n openshift-cnv --patch '{
    "data": {
        "osConfigMap.name": "os-configmap",
        "osConfigMap.namespace": "default" (1)
    }
}'
```

(1) 如果在配置映射中更改了命名空间，请更新它。

7. 验证模板是否出现在 OpenShift Virtualization web 控制台中：
   a. 从侧边菜单中点 **Workloads → Virtualization**。
   b. 点 **Virtual Machine Templates** 标签页，在列表中找到模板。

8.15.4.5. 导入虚拟机的故障排除

8.15.4.5.1. 日志

您可以检查 VM Import Controller pod 日志中的错误。

流程

1. 运行以下命令，查看 VM Import Controller pod 的名称：

```
$ oc get pods -n <namespace> | grep import (1)
```

(1) 指定导入虚拟机的命名空间。

输出示例

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

2. 运行以下命令查看 VM Import Controller pod 日志：

```
-
```

```
$ oc logs <vm-import-controller-f66f7d-zqkz7> -f -n <namespace> 1
```

1 指定 VM Import Controller pod 名称和命名空间。

8.15.4.5.2. é"™è¯¯ä¿¡æ¯¯

可能会出现以下出错信息：

- 如果 OpenShift Virtualization 存储 PV 不合适，VM Import Controller pod 日志中会显示以下错误消息，进度条会在 10% 的位置停止：

  ```
  Failed to bind volumes: provisioning failed for PVC
  ```

  您必须使用兼容的存储类。不支持 Cinder 存储类。

8.15.4.5.3. 已知问题

- 如果您使用 Ceph RBD 块模式卷，且可用的存储空间对于虚拟磁盘太小，则导入过程会在完成 75% 的时候停止 20 分钟以上，且迁移也不会成功。web 控制台中没有显示错误消息。BZ#1910019

8.15.5. 导入一个单一的 VMware 虚拟机或模板

您可以使用 VM I导入向导将 VMware vSphere 6.5、6.7 或 7.0 VM 或 VM 模板导入到 OpenShift Virtualization 中。如果您导入一个虚拟机模板，OpenShift Virtualization 会根据模板创建一个虚拟机。

重要

导入 VMware VM 是一个已弃用的功能。弃用的功能仍然包含在 OpenShift Virtualization 中，并且仍然被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

有关 OpenShift Virtualization 中已弃用或删除的主要功能的最新列表，请参阅 OpenShift Virtualization 发行注记中 *已弃用和删除的功能*部分。

这个功能将被虚拟化的 Migration Toolkit 替代。

8.15.5.1. OpenShift Virtualization 存储功能列表

下表描述了支持导入虚拟机的 OpenShift Virtualization 存储类型。

表 8.6. OpenShift Virtualization 存储功能列表

	VMware VM 导入
OpenShift Container Storage: RBD 块模式卷	是
OpenShift Virtualization hostpath 置备程序	是
其他多节点可写入存储	是 [1]

	VMware VM 导入
其他单节点可写入存储	是 [2]

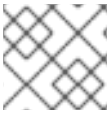
1. PVC 必须请求 ReadWriteMany 访问模式。
2. PVC 必须请求 ReadWriteOnce 访问模式。

8.15.5.2. 准备 VDDK 镜像

导入过程使用 VMware Virtual Disk Development Kit（VDDK）来复制 VMware 虚拟磁盘。

您可以下载 VDDK SDK，创建 VDDK 镜像，将镜像上传到镜像 registry，并将其添加到 **HyperConverged** 自定义资源（CR）的 **spec.vddkInitImage** 字段中。

您可以配置内部 OpenShift Container Platform 镜像 registry 或 VDDK 镜像的安全外部镜像 registry。该 registry 需要可以被 OpenShift Virtualization 环境访问。



注意

在公共仓库中存储 VDDK 镜像可能会违反 VMware 许可证的条款。

8.15.5.2.1. 配置内部镜像 registry

您可以通过更新 Image Registry Operator 配置在裸机上配置内部 OpenShift Container Platform 镜像 registry。

您可以通过一个路由公开 registry，直接从 OpenShift Container Platform 集群内部或外部访问 registry。

更改镜像 registry 的管理状态
要启动镜像 registry，您必须将 Image Registry Operator 配置的 **managementState** 从 **Removed** 改为 **Managed**。

流程

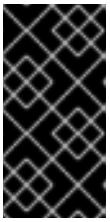
- 将 **managementState** Image Registry Operator 配置从 **Removed** 改为 **Managed**。例如：

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

为裸机和其他手动安装配置 registry 存储
作为集群管理员，在安装后需要配置 registry 来使用存储。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您有一个使用手动置备的 Red Hat Enterprise Linux CoreOS(RHCOS)节点（如裸机）的集群。
- 已为集群置备了持久性存储，如 Red Hat OpenShift Container Storage。



重要

当您只有一个副本时，OpenShift Container Platform 支持对镜像 registry 存储的 **ReadWriteOnce** 访问。**ReadWriteOnce** 访问还要求 registry 使用 **Recreate** rollout 策略。要部署支持高可用性的镜像 registry，需要两个或多个副本，**ReadWriteMany** 访问。

- 必须具有 100Gi 容量。

流程

1. 要将 registry 配置为使用存储，修改 **configs.imageregistry/cluster** 资源中的 **spec.storage.pvc**。

注意

使用共享存储时，请查看您的安全设置以防止外部访问。

2. 验证您没有 registry pod:

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

输出示例

```
No resourses found in openshift-image-registry namespace
```

注意

如果您的输出中有一个 registry pod，则不需要继续这个过程。

3. 检查 registry 配置：

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

输出示例

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

将 **claim** 字段留空以允许自动创建 **image-registry-storage** PVC。

4. 检查 **clusteroperator** 状态：

```
$ oc get clusteroperator image-registry
```

输出示例

```
NAME           VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                               True        False         False      6h50m
```

5. 确保 registry 设置为 managed，以启用镜像的构建和推送。

   - 运行：

     ```
     $ oc edit configs.imageregistry/cluster
     ```

     然后，更改行

     ```
     managementState: Removed
     ```

     to

     ```
     managementState: Managed
     ```

直接从集群访问registry
您可以从集群内部访问registry。

流程

通过使用内部路由从集群访问registry：

1. 通过获取节点的名称来访问节点：

   ```
   $ oc get nodes
   ```

   ```
   $ oc debug nodes/<node_name>
   ```

2. 要在节点上启用对 **oc** 和 **podman** 等工具的访问，请将您的根目录改为 **/host** ：

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

3. 使用您的访问令牌登录到容器镜像registry：

   ```
   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
   ```

   您应该看到一条确认登录的消息，例如：

   ```
   Login Succeeded!
   ```

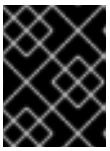
   

   注意

   用户名可以是任何值，令牌包含了所有必要的信息。如果用户名包含冒号，则会导致登录失败。

   因为 Image Registry Operator 创建了路由，所以它将与 **default-route-openshift-image-registry.<cluster_name>** 类似。

4. 针对您的registry执行**podman pull**和**podman push**操作：



重要

您可以抓取任意镜像，但是如果已添加了**system:registry**角色，则只能将镜像推送到您自己的registry中。

在以下示例中，使用：

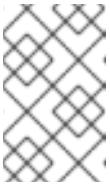
组件	值
<registry_ip>	**172.30.124.220**
<port>	**5000**
<project>	**openshift**
<image>	**image**
<tag>	忽略 (默认为 **latest**)

a. 抓取任意镜像：

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

b. 使用 **<registry_ip>:<port>/<project>/<image>** 格式标记（tag）新镜像。项目名称必须出现在这个 pull 规范中，以供OpenShift Container Platform 把这个镜像正确放置在 registry 中，并在以后正确访问 registry 中的这个镜像：

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



注意

您必须具有指定项目的**system:image-builder**角色，该角色允许用户写或推送镜像。否则，下一步中的**podman push**将失败。为了进行测试，您可以创建一个新项目来推送镜像。

c. 将新标记的镜像推送到 registry：

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

手动公开受保护的registry

通过使用路由可以开放从外部访问OpenShift Container Platform registry的通道，用户不再需要从集群内部登录到OpenShift Container Platform registry。这样，您可以使用路由地址从集群外部登录 registry，并使用路由主机标记并推送到现有项目。

先决条件

- 以下的先决条件会被自动执行：
  - 部署 Registry Operator。

- 部署 Ingress Operator。

流程

您可以使用**configs.imageregistry.operator.openshift.io**资源中的**DefaultRoute**参数或使用自定义路由来公开路由。

使用**DefaultRoute**公开registry：

1. 将**DefaultRoute**设置为**True** ：

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

2. 使用 **podman** 登录：

```
$ 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 (1)
```

(1) 如果集群的默认路由证书不受信任，则需要**--tls-verify=false** 。您可以将一个自定义的可信证书设置为 Ingress Operator 的默认证书。

使用自定义路由公开registry：

1. 使用路由的 TLS 密钥创建一个 secret：

```
$ oc create secret tls public-route-tls \
    -n openshift-image-registry \
    --cert=</path/to/tls.crt> \
    --key=</path/to/tls.key>
```

此步骤是可选的。如果不创建一个secret，则路由将使用Ingress Operator的默认TLS配置。

2. 在 Registry Operator 中：

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

注意

如果为registry的路由提供了一个自定义的 TLS 配置，则仅需设置**secretName** 。

8.15.5.2.2. 配置外部镜像 registry

如果将外部镜像 registry 用于 VDDK 镜像，您可以将外部镜像 registry 的证书颁发机构添加到 OpenShift Container Platform 集群。

另外，您可以从 Docker 凭证中创建一个 pull secret，并将其添加到您的服务帐户中。

在集群中添加证书颁发机构
您可以按照以下流程将证书颁发机构 (CA) 添加到集群，以便在推送和拉取镜像时使用。

先决条件

- 您必须具有集群管理员特权。
- 您必须有权访问 registry 的公共证书，通常是位于 **/etc/docker/certs.d/** 目录中的 **hostname/ca.crt** 文件。

流程

1. 在 **openshift-config** 命名空间中创建一个 **ConfigMap**，其中包含使用自签名证书的 registry 的可信证书。对于每个 CA 文件，确保 **ConfigMap** 中的键是 **hostname[..port]** 格式的容器镜像仓库的主机名：

```
$ oc create configmap registry-cas -n openshift-config \
--from-file=myregistry.corp.com..5000=/etc/docker/certs.d/myregistry.corp.com:5000/ca.crt \
--from-file=otherregistry.com=/etc/docker/certs.d/otherregistry.com/ca.crt
```

2. 更新集群镜像配置：

```
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"registry-cas"}}}' --type=merge
```

允许 Pod 引用其他安全 registry 中的镜像
Docker 客户端的 **.dockercfg $HOME/.docker/config.json** 文件是一个 Docker 凭证文件，如果您之前已登录安全或不安全的 registry，则该文件会保存您的身份验证信息。

要拉取（pull）并非来自 OpenShift Container Platform 内部 registry 的安全容器镜像，您必须从 Docker 凭证创建一个 pull secret，并将其添加到您的服务帐户。

流程

- 如果您已有该安全 registry 的 **.dockercfg** 文件，则可运行以下命令从该文件中创建一个 secret：

```
$ oc create secret generic <pull_secret_name> \
    --from-file=.dockercfg=<path/to/.dockercfg> \
    --type=kubernetes.io/dockercfg
```

- 或者，如果您已有 **$HOME/.docker/config.json** 文件，则可运行以下命令：

```
$ oc create secret generic <pull_secret_name> \
    --from-file=.dockerconfigjson=<path/to/.docker/config.json> \
    --type=kubernetes.io/dockerconfigjson
```

- 如果您还没有安全 registry 的 Docker 凭证文件，则可运行以下命令创建一个 secret：

```
$ oc create secret docker-registry <pull_secret_name> \
    --docker-server=<registry_server> \
    --docker-username=<user_name> \
    --docker-password=<password> \
    --docker-email=<email>
```

- 要使用 secret 为 Pod 拉取镜像，您必须将 secret 添加到您的服务帐户中。本例中服务帐户的名称应与 Pod 使用的服务帐户的名称匹配。默认服务帐户是 **default**：

```
$ oc secrets link default <pull_secret_name> --for=pull
```

8.15.5.2.3. 创建并使用 VDDK 镜像

您可以下载 VMware Virtual Disk Development Kit（VDDK），构建 VDDK 镜像，并将 VDDK 镜像推送到您的镜像 registry。然后，您将 VDDK 镜像添加到 **HyperConverged** 自定义资源（CR）的 **spec.vddkInitImage** 字段中。

先决条件

- 您必须有权访问 OpenShift Container Platform 内部镜像 registry 或安全的外部 registry。

流程

1. 创建并导航到临时目录：

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

2. 在一个浏览器中，进入 VMware code 并点 **SDKs**。
3. 在 **Compute Virtualization** 下，点 **Virtual Disk Development Kit(VDDK)**。
4. 选择与 VMware vSphere 版本对应的 VDDK 版本，例如 vSphere 7.0 的 VDDK 7.0，点 **Download**，然后在临时目录中保存 VDDK 归档。
5. 提取 VDDK 归档：

```
$ tar -xzf VMware-vix-disklib-<version>.x86_64.tar.gz
```

6. 创建 **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. 构建镜像：

```
$ podman build . -t <registry_route_or_server_path>/vddk:<tag> 1
```

1. 指定您的镜像 registry:
   - 对于内部 OpenShift Container Platform registry，请使用内部 registry 路由，如 **image-registry.openshift-image-registry.svc:5000/openshift/vddk:<tag>**。
   - 对于外部 registry，指定服务器名称、路径和标签。例如 **server.example.com:5000/vddk:<tag>**。

8. 将镜像推送至 registry：

```
$ podman push <registry_route_or_server_path>/vddk:<tag>
```

9. 确保镜像可以被 OpenShift Virtualization 环境访问。

10. 编辑 **openshift-cnv** 项目中的 **HyperConverged** CR：

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

11. 将 **vddkInitImage** 参数添加到 **spec** 小节中：

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  vddkInitImage: <registry_route_or_server_path>/vddk:<tag>
```

8.15.5.3. 使用 VM 导入向导（Import wizard）导入虚拟机

您可以使用 VM 导入向导导入单个虚拟机。

您还可以导入虚拟机模板。如果您导入一个虚拟机模板，OpenShift Virtualization 会根据模板创建一个虚拟机。

先决条件

- 具有 admin 用户权限。
- VMware Virtual Disk Development Kit（VDDK）镜像必须位于 OpenShift Virtualization 环境可访问的镜像 registry 中。
- VDDK 镜像必须添加到 **HyperConverged** 自定义资源（CR）的 **spec.vddkInitImage** 字段中。
- 虚拟机必须关机。
- 虚拟磁盘必须连接到 IDE 或者 SCSI 控制器。如果虚拟磁盘连接到一个 SATA 控制器，您可以将其改为 IDE 控制器，然后迁移虚拟机。
- OpenShift Virtualization 本地和共享的持久性存储类必须支持虚拟机导入。
- OpenShift Virtualization 存储必须足够大来保存虚拟磁盘。

警告

如果使用 Ceph RBD 块模式卷，则存储必须有足够的空间来存储虚拟磁盘。如果可用存储的大小无法满足磁盘要求，导入过程会失败，且用于复制虚拟磁盘的 PV 也不会被释放。因为没有足够的资源来删除对象，您将无法导入另一个虚拟机或清除存储。要解决这种情况，您必须在存储后端中添加更多对象存储设备。

- OpenShift Virtualization 出口网络策略必须允许以下流量：

目的地	协议	端口
VMware ESXi 主机	TCP	443
VMware ESXi 主机	TCP	902
VMware vCenter	TCP	5840

流程

1. 在 web 控制台中，点 **Workloads → Virtual Machines**。
2. 点击 **Create Virtual Machine**并选择 **Import with Wizard**。
3. 从 **Provider** 列表中选择 **VMware**。
4. 选择 **Connect to New Instance** 或一个保存的 vCenter 示例。
   - 如果您选择 **Connect to New Instance**，输入 **vCenter hostname**、**Username** 和 **Password**。
   - 如果您选择了一个保存的 vCenter 实例，向导将使用保存的凭证连接到 vCenter 实例。
5. 点击 **Check and Save**，然后等待连接完成。

注意

连接详情存储在 secret 中。如果您添加的供应商带有不正确的主机名、用户名或密码，点 **Workloads → Secrets** 并删除供应商 secret。

6. 选择一个虚拟机或一个模板。
7. 点 **Next**。
8. 在 **Review** 屏幕中，查看您的设置。
9. 点 **Edit** 以更新以下设置：
   - **General**:

- 描述
- 操作系统
- Flavor
- 内存
- CPU
- Workload Profile

- Networking:
  - 名称
  - Model
  - 网络
  - 类型
  - MAC 地址

- Storage：点击 VM 磁盘  的 Options 菜单，然后选择 Edit 来更新以下字段：
  - 名称
  - Source：例如 Import Disk。
  - Size
  - Interface
  - Storage Class:选择 NFS 或 ocs-storagecluster-ceph-rbd(ceph-rbd)。
    如果选择 ocs-storagecluster-ceph-rbd，您必须将磁盘的 Volume Mode 设置为 Block。

    其他存储类可能会正常工作，但不被正式支持。
  - advanced → Volume Mode: 选择 Block。
  - Advanced → Access Mode
- Advanced → Cloud-init:
  - Form: 输入 Hostname 和 Authenticated SSH Keys.
  - Custom script 在文本字段中输入 cloud-init 脚本。
- Advanced → Virtual Hardware：您可以将虚拟 CD-ROM 附加到导入的虚拟机。

10. 如果您编辑了导入设置，点 Import 或 Review and Import。
    此时会显示 Successfully created virtual machine 消息以及为虚拟机创建的资源列表。虚拟机会出现在 Workloads → Virtual Machines 中。

虚拟机向导字段

名称	参数	描述
名称		名称可包含小写字母 (**a-z**)、数字 (**0-9**) 和连字符 (**-**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格、句点 (**.**) 或特殊字符。
描述		可选的描述字段。
操作系统		模板中为虚拟机选择的操作系统。从模板创建虚拟机时，您无法编辑此字段。
引导源	通过 URL 导入（创建 PVC）	从 HTTP 或 HTTPS 端点提供的镜像导入内容。示例：包含操作系统镜像的网页中的 URL 链接。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有持久性卷声明并克隆它。
	通过 Registry 导入（创建 PVC）	从可通过集群访问的注册表中的可启动操作系统容器置备虚拟机。示例：***kubevirt/cirros-registry-disk-demo***。
	PXE（网络引导 - 添加网络接口）	从网络的服务器引导操作系统。需要一个 PXE 可引导网络附加定义。
持久性卷声明项目		用于克隆 PVC 的项目名称。
持久性卷声明名称		如果您要克隆现有的 PVC，则应用于此虚拟机模板的 PVC 名称。
将它作为光盘引导源挂载		CD-ROM 需要额外的磁盘来安装操作系统。选择添加磁盘的选择框并稍后进行自定义。
Flavor	tiny、small、Medium、Large、Custom	根据与该模板关联的操作系统，在虚拟机模板中预设具有预定义值的 CPU 和内存量。 如果选择了默认模板，您可以使用自定义值覆盖模板中的 **cpus** 和 **memsize** 的值，以创建自定义模板。另外，您可以通过修改 **Workloads → Virtualization** 页面上的 **Details** 选项卡中的 **cpus** 和 **memsize** 值来创建自定义模板。

名称	参数	描述
工作负载类型 **注意** 如果您选择了不正确的 **Workload Type**，则可能会出现性能或资源利用率问题（如缓慢的 UI）。	Desktop	用于桌面的虚拟机配置。适用于小型工作环境。建议与 Web 控制台搭配使用。
	Server	在性能和广泛的服务器工作负载兼容性方面具有最佳平衡。
	高性能	针对高性能负载进行了优化的虚拟机配置。
创建后启动此虚拟机。		默认选择这个复选框并在创建后启动虚拟机。如果您不希望虚拟机在创建时启动，请清除该复选框。

Cloud-init 字段

名称	描述
Hostname	为虚拟机设置特定主机名。
授权 SSH 密钥	复制到虚拟机上 *~/.ssh/authorized_keys* 的用户公钥。
自定义脚本	将其他选项替换为您粘贴自定义 cloud-init 脚本的字段。

网络字段

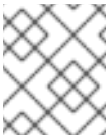
名称	描述
名称	网络接口控制器的名称。
model	指明网络接口控制器的型号。支持的值有 **e1000e** 和 **virtio**。
网络	可用网络附加定义的列表。
类型	可用绑定方法列表。对于默认的 pod 网络，**masquerade** 是唯一推荐的绑定方法。对于辅助网络，请使用 **bridge** 绑定方法。非默认网络不支持 **masquerade** 绑定方法。如果您配置了一个 **SR-IOV** 网络设备并在命名空间中定义那个网络，请选择 SR-IOV。
MAC 地址	网络接口控制器的 MAC 地址。如果没有指定 MAC 地址，则会自动分配一个。

存储字段

名称	选择	描述
Source	空白（创建 PVC）	创建一个空磁盘。
	通过 URL 导入（创建 PVC）	通过 URL（HTTP 或 HTTPS 端点）导入内容。
	使用现有的 PVC	使用集群中已可用的 PVC。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有 PVC 并克隆它。
	通过 Registry 导入（创建 PVC）	通过容器 registry 导入内容。
	容器（临时）	从集群可以访问的 registry 中的容器上传内容。容器磁盘应只用于只读文件系统，如 CD-ROM 或临时虚拟机。
名称		磁盘的名称。名称可包含小写字母 (**a-z**)、数字 (**0-9**)、连字符 (**-**) 和句点 (**.**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格或特殊字符。
Size		GiB 中磁盘的大小。
类型		磁盘类型。示例：磁盘或光盘
Interface		磁盘设备的类型。支持的接口包括 **virtIO**、**SATA** 和 **SCSI**。
Storage class		用于创建磁盘的存储类。
Advanced → Volume Mode		定义持久性卷是否使用格式化的文件系统或原始块状态。默认为 **Filesystem**。
Advanced → Access Mode		持久性卷访问模式。支持的访问模式有 **Single User（RWO）**、**Shared Access（RWX）** 和 **Read Only（ROX）**。

高级存储设置

以下高级存储设置可用于 **空白**、**从 URL 导入** 和 **克隆现有的 PVC** 磁盘。所有参数都是可选的。如果没有指定这些参数，系统将使用 **kubevirt-storage-class-defaults** 配置映射中的默认值。

名称	参数	描述
卷模式	Filesystem	在基于文件系统的卷中保存虚拟磁盘。
	Block	直接将虚拟磁盘存储在块卷中。只有底层存储支持时才使用 **Block**。
访问模式	Single User (RWO)	这个卷可以被一个单一的节点以 read/write 的形式挂载。
	Shared Access (RWX)	卷可以被多个节点以读写模式挂载。  **注意** 对于一些功能（如虚拟机在节点间实时迁移）需要这个权限。
	Read Only (ROX)	卷可以被多个节点以只读形式挂载。

8.15.5.3.1. 更新导入虚拟机的 NIC 名称

您必须更新从 VMware 导入的虚拟机的 NIC 名称，以符合 OpenShift Virtualization 命名约定。

流程

1. 登录虚拟机。
2. 进入 **/etc/sysconfig/network-scripts** 目录。
3. 重新命名网络配置文件：

```
$ mv vmnic0 ifcfg-eth0 (1)
```

(1) 第一个网络配置文件的名称为 **ifcfg-eth0**。额外网络配置文件按顺序编号，例如：**ifcfg-eth1**、**ifcfg-eth2**。

4. 更新网络配置文件中的 **NAME** 和 **DEVICE** 参数：

```
NAME=eth0
DEVICE=eth0
```

5. 重启网络：

```
$ systemctl restart network
```

8.15.5.4. 导入虚拟机的故障排除

8.15.5.4.1. 日志

您可以检查 V2V Conversion pod 日志中的错误。

流程

1. 运行以下命令来查看 V2V Conversion pod 名称：

```
$ oc get pods -n <namespace> | grep v2v 1
```

1 指定导入虚拟机的命名空间。

输出示例

```
kubevirt-v2v-conversion-f66f7d-zqkz7          1/1     Running     0          4h49m
```

2. 运行以下命令来查看 V2V Conversion pod 日志：

```
$ oc logs <kubevirt-v2v-conversion-f66f7d-zqkz7> -f -n <namespace> 1
```

1 指定 VM Conversion pod 名称和命名空间。

8.15.5.4.2. é″™è¯¯ä¿¡æ¯¯

此时会出现以下出错信息：

- 如果导入前 VMware VM 没有关闭，导入的虚拟机会显示错误消息，在 OpenShift Container Platform 控制台中显示**Readiness probe failed**，V2V Conversion pod 日志会显示以下错误消息：

```
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',)"
```

- 如果非 admin 用户尝试导入虚拟机，则 OpenShift Container Platform 控制台中会显示以下错误消息：

```
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"
```

只有 admin 用户可以导入虚拟机。

8.16. 克隆虚拟机

8.16.1. 启用用户权限跨命名空间克隆数据卷

命名空间的隔离性质意味着用户默认无法在命名空间之间克隆资源。

要让用户将虚拟机克隆到另一个命名空间，具有 **cluster-admin** 角色的用户必须创建新的集群角色。将此集群角色绑定到用户，以便其将虚拟机克隆到目标命名空间。

8.16.1.1. 先决条件

- 只有具有 **cluster-admin** 角色的用户才能创建集群角色。

8.16.1.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.16.1.3. 创建用于克隆数据卷的 RBAC 资源

创建一个新的集群角色，为 **datavolumes** 资源的所有操作启用权限。

流程

1. 创建 **ClusterRole** 清单：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <datavolume-cloner> 1
rules:
- apiGroups: ["cdi.kubevirt.io"]
  resources: ["datavolumes/source"]
  verbs: ["*"]
```

1 集群角色的唯一名称。

2. 在集群中创建集群角色：

```
$ oc create -f <datavolume-cloner.yaml> 1
```

1 上一步中创建的 **ClusterRole** 清单的文件名。

3. 创建应用于源和目标命名空间的 **RoleBinding** 清单，并引用上一步中创建的集群角色。

```
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 角色绑定的唯一名称。

2 源数据卷的命名空间。

3 数据卷克隆到的命名空间。

4 上一步中创建的集群角色的名称。

4. 在集群中创建角色绑定：

```
$ oc create -f <datavolume-cloner.yaml> 1
```

1 上一步中创建的 **RoleBinding** 清单的文件名。

8.16.2. 将虚拟机磁盘克隆到新数据卷中

您可以通过引用数据卷配置文件中的源 PVC 来将虚拟机磁盘的持久性卷声明（PVC）克隆到新数据卷中。

警告

支持在不同卷模式间克隆操作，比如从带有 **volumeMode: Block** 的持久性卷（PV）克隆到带有 **volumeMode: Filesystem** 的 PV。

但是，只有在不同的卷模式中存在**contentType: kubevirt** 时才可以克隆它们。

提示

当您全局启用预分配或单个数据卷时，Containerized Data Importer（CDI）会在克隆过程中预分配磁盘空间。预分配可提高写入性能。如需更多信息，请参阅对数据卷使用预分配。

8.16.2.1. 先决条件

- 用户需要额外的权限才能将虚拟机磁盘的 PVC 克隆到另一个命名空间中。

8.16.2.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.16.2.3. 将虚拟机磁盘的持久性卷声明克隆到新数据卷中

您可以将现有虚拟机磁盘的持久性卷声明（PVC）克隆到新数据卷中。新的数据卷可用于新虚拟机。

注意

当独立于虚拟机创建数据卷时，数据卷的生命周期与虚拟机是独立的。如果删除了虚拟机，数据卷及其相关 PVC 都不会被删除。

先决条件

- 确定要使用的现有虚拟机磁盘的 PVC。克隆之前，必须关闭与 PVC 关联的虚拟机。
- 安装 OpenShift CLI（**oc**）。

流程

1. 检查您要克隆的虚拟机磁盘，以识别关联 PVC 的名称和命名空间。
2. 为数据卷创建一个 YAML 文件，用于指定新数据卷的名称、源 PVC 的名称和命名空间，以及新数据卷的大小。
   例如：

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 4
```

1. 新数据卷的名称。
2. 源 PVC 所在的命名空间。
3. 源 PVC 的名称。
4. 新数据卷的大小。您必须分配足够空间，否则克隆操作会失败。其大小不得低于源 PVC。

3. 通过创建数据卷开始克隆 PVC:

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

注意

在 PVC 就绪前，DataVolume 会阻止虚拟机启动，以便您可以在 PVC 克隆期间创建引用新数据卷的虚拟机。

8.16.2.4. 模板：数据卷克隆配置文件

example-clone-dv.yaml

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
```

```
metadata:
  name: "example-clone-dv"
spec:
  source:
     pvc:
       name: source-pvc
       namespace: example-ns
  pvc:
    accessModes:
     - ReadWriteOnce
    resources:
      requests:
        storage: "1G"
```

8.16.2.5. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

☐ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.16.3. 使用数据卷模板克隆虚拟机

您可以通过克隆现有虚拟机的持久性卷声明（PVC）来创建新虚拟机。在虚拟机配置文件中包括 **dataVolumeTemplate**，即可从原始 PVC 创建新数据卷。

警告

支持在不同卷模式间克隆操作，比如从带有 **volumeMode: Block** 的持久性卷（PV）克隆到带有 **volumeMode: Filesystem** 的 PV。

但是，只有在不同的卷模式中存在**contentType: kubevirt** 时才可以克隆它们。

提示

当您全局启用预分配或单个数据卷时，Containerized Data Importer（CDI）会在克隆过程中预分配磁盘空间。预分配可提高写入性能。如需更多信息，请参阅对数据卷使用预分配。

8.16.3.1. 先决条件

- 用户需要额外的权限才能将虚拟机磁盘的 PVC 克隆到另一个命名空间中。

8.16.3.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.16.3.3. 使用数据卷模板从克隆的持久性卷声明创建新虚拟机

您可以创建一个虚拟机，将现有虚拟机的持久性卷声明（PVC）克隆到数据卷中。在虚拟机清单中引用 **dataVolumeTemplate**，并将**源** PVC 克隆到数据卷中，然后自动用于创建虚拟机。

注意

当作为虚拟机的数据卷模板创建数据卷时，数据卷的生命周期依赖于虚拟机。如果删除了虚拟机，数据卷和相关 PVC 也会被删除。

先决条件

- 确定要使用的现有虚拟机磁盘的 PVC。克隆之前，必须关闭与 PVC 关联的虚拟机。
- 安装 OpenShift CLI（**oc**）。

流程

1. 检查您要克隆的虚拟机，以识别关联 PVC 的名称和命名空间。
2. 为 **VirtualMachine** 对象创建 YAML 文件。以下虚拟机示例克隆 **my-favorite-vm-disk**，该磁盘位于 **source-name** 命名空间中。从**my-favorite-vm-disk** 创建的名为 **favorite-clone** 的 **2Gi** 数据卷 。
   例如：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: vm-dv-clone
  name: vm-dv-clone (1)
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: vm-dv-clone
    spec:
      domain:
```

```
        devices:
          disks:
          - disk:
              bus: virtio
            name: root-disk
        resources:
          requests:
            memory: 64M
      volumes:
      - dataVolume:
          name: favorite-clone
        name: root-disk
  dataVolumeTemplates:
  - metadata:
      name: favorite-clone
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 2Gi
      source:
        pvc:
          namespace: "source-namespace"
          name: "my-favorite-vm-disk"
```

1 要创建的虚拟机。

3. 使用 PVC 克隆的数据卷创建虚拟机：

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

8.16.3.4. 模板：数据卷虚拟机配置文件

example-dv-vm.yaml

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: example-vm
  name: example-vm
spec:
  dataVolumeTemplates:
  - metadata:
      name: example-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 1G
```

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

1 您要导入的镜像的 **HTTP** 源（如适用）。

8.16.3.5. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* □ GZ □ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* □ GZ □ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

□ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.16.4. 将虚拟机磁盘克隆到新块存储数据卷中

您可以通过引用数据卷配置文件中的源 PVC 来将虚拟机磁盘的持久性卷声明（PVC）克隆到新的块数据卷中。

警告

支持在不同卷模式间克隆操作，比如从带有 **volumeMode: Block** 的持久性卷（PV）克隆到带有 **volumeMode: Filesystem** 的 PV。

但是，只有在不同的卷模式中存在**contentType: kubevirt** 时才可以克隆它们。

提示

当您全局启用预分配或单个数据卷时，Containerized Data Importer（CDI）会在克隆过程中预分配磁盘空间。预分配可提高写入性能。如需更多信息，请参阅对数据卷使用预分配。

8.16.4.1. 先决条件

- 用户需要额外的权限才能将虚拟机磁盘的 PVC 克隆到另一个命名空间中。

8.16.4.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.16.4.3. 关于块持久性卷

块持久性卷（PV）是一个受原始块设备支持的 PV。这些卷没有文件系统，可以通过降低开销来为虚拟机提供性能优势。

原始块卷可以通过在 PV 和持久性卷声明（PVC）规格中指定 **volumeMode: Block** 来置备。

8.16.4.4. 创建本地块持久性卷

通过填充文件并将其挂载为循环设备，在节点上创建本地块持久性卷（PV）。然后，您可以在 PV 清单中将该循环设备作为 **Block（块）**卷引用，并将其用作虚拟机镜像的块设备。

流程

1. 以 **root** 身份登录节点，在其上创建本地 PV。本流程以 **node01** 为例。
2. 创建一个文件并用空字符填充，以便可将其用作块设备。以下示例创建 **loop10** 文件，大小为 2Gb（20,100 Mb 块）：

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

3. 将 **loop10** 文件挂载为 loop 设备。

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

1. 挂载 loop 设备的文件路径。
2. 上一步中创建的文件，挂载为 loop 设备。

4. 创建引用所挂载 loop 设备的 **PersistentVolume** 清单。

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <local-block-pv10>
  annotations:
spec:
  local:
    path: </dev/loop10> 1
  capacity:
    storage: <2Gi>
  volumeMode: Block 2
  storageClassName: local 3
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - <node01> 4
```

1. 节点上的 loop 设备路径。
2. 将其指定为块 PV。
3. 可选：为 PV 设置存储类。如果省略此项，将使用默认集群。
4. 挂载块设备的节点。

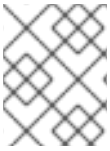
5. 创建块 PV。

```
# oc create -f <local-block-pv10.yaml> 1
```

1. 上一步中创建的持久性卷的文件名。

8.16.4.5. 将虚拟机磁盘的持久性卷声明克隆到新数据卷中

您可以将现有虚拟机磁盘的持久性卷声明（PVC）克隆到新数据卷中。新的数据卷可用于新虚拟机。



注意

当独立于虚拟机创建数据卷时，数据卷的生命周期与虚拟机是独立的。如果删除了虚拟机，数据卷及其相关 PVC 都不会被删除。

先决条件

- 确定要使用的现有虚拟机磁盘的 PVC。克隆之前，必须关闭与 PVC 关联的虚拟机。
- 安装 OpenShift CLI（**oc**）。
- 至少一个可用块持久性卷（PV）大小不低于源 PVC。

流程

1. 检查您要克隆的虚拟机磁盘，以识别关联 PVC 的名称和命名空间。
2. 为数据卷创建一个 YAML 文件，用于指定新数据卷的名称、源 PVC 的名称和命名空间、**volumeMode: Block**，以便使用可用块 PV，以及新数据卷的大小。
   例如：

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 4
    volumeMode: Block 5
```

1. 新数据卷的名称。
2. 源 PVC 所在的命名空间。
3. 源 PVC 的名称。
4. 新数据卷的大小。您必须分配足够空间，否则克隆操作会失败。其大小不得低于源 PVC。
5. 指定目标为一个块 PV

3. 通过创建数据卷开始克隆 PVC:

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

注意

在 PVC 就绪前，DataVolume 会阻止虚拟机启动，以便您可以在 PVC 克隆期间创建引用新数据卷的虚拟机。

8.16.4.6. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

☐ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.17. 虚拟机网络

8.17.1. 为默认 pod 网络配置虚拟机

您可以通过将其网络接口配置为使用 **masquerade** 绑定模式，将虚拟机连接到默认的内部 pod 网络

8.17.1.1. 从命令行配置伪装模式

您可以使用伪装模式将虚拟机的外发流量隐藏在 pod IP 地址后。伪装模式使用网络地址转换 (NAT) 来通过 Linux 网桥将虚拟机连接至 pod 网络后端。

启用伪装模式，并通过编辑虚拟机配置文件让流量进入虚拟机。

先决条件

- 虚拟机必须配置为使用 DHCP 来获取 IPv4 地址。以下示例配置为使用 DHCP。

流程

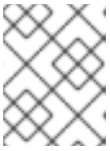
1. 编辑虚拟机配置文件的 **interfaces** 规格：

```
kind: VirtualMachine
spec:
  domain:
```

```
  devices:
    interfaces:
      - name: default
        masquerade: {} 1
        ports: 2
          - port: 80
  networks:
  - name: default
    pod: {}
```

1 使用伪装模式进行连接。

2 可选：列出您要从虚拟机公开的端口，每个都由 **port** 字段指定。**port** 值必须是 0 到 65536 之间的数字。如果没有使用 **port** 数组，则有效范围内的所有端口都开放给传入流量。在本例中，端口 **80** 上允许传入的流量。



注意

端口 49152 和 49153 保留供 libvirt 平台使用，这些端口的所有其他传入流量将被丢弃。

2. 创建虚拟机：

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

8.17.1.2. 使用双栈（IPv4 和 IPv6）配置伪装模式.

您可以使用 cloud-init 将新虚拟机配置为在默认 pod 网络上同时使用 IPv6 和 IPv4。

在虚拟机配置中，必须将 IPv6 网络地址静态设置为 **fd10:0:2::2/120**，默认网关为 **fd10:0:2::1**。virt-launcher Pod 使用它们将 IPv6 流量路由到虚拟机，而不在外部使用。

当虚拟机运行时，虚拟机的传入和传出流量将路由到 IPv4 地址和 virt-launcher Pod 的唯一 IPv6 地址。virt-launcher pod 随后将 IPv4 流量路由到虚拟机的 DHCP 地址，并将 IPv6 流量路由到虚拟机的静态设置 IPv6 地址。

先决条件

- OpenShift Container Platform 集群必须使用为 dual-stack 配置的 OVN-Kubernetes Container Network Interface（CNI）网络供应商。

流程

1. 在新的虚拟机配置中，包含具有 **masquerade** 的接口，并使用 cloud-init 配置 IPv6 地址和默认网关。

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: example-vm-ipv6
...
        interfaces:
          - name: default
```

```
        masquerade: {} (1)
        ports:
          - port: 80 (2)
networks:
- name: default
  pod: {}
volumes:
- cloudInitNoCloud:
    networkData: |
      version: 2
      ethernets:
        eth0:
          dhcp4: true
          addresses: [ fd10:0:2::2/120 ] (3)
          gateway6: fd10:0:2::1 (4)
```

1. 使用伪装模式进行连接。
2. 允许虚拟机上端口 80 上的传入流量。
3. 您必须使用 IPv6 地址 **fd10:0:2::2/120**。
4. 您必须使用网关 **fd10:0:2::1**。

2. 在命名空间中创建虚拟机：

```
$ oc create -f example-vm-ipv6.yaml
```

验证

- 要验证 IPv6 是否已配置，启动虚拟机并查看虚拟机实例的接口状态，以确保它具有 IPv6 地址：

```
$ oc get vmi <vmi-name> -o jsonpath="{.status.interfaces[*].ipAddresses}"
```

8.17.2. 创建服务以公开虚拟机

您可以使用 **Service** 对象在集群内或集群外部公开虚拟机。

8.17.2.1. 关于服务

一个 Kubernetes *服务*是一个抽象的方式，用于公开在一组 pod 上运行的应用程序作为网络服务。服务允许您的应用程序接收流量。通过在 **Service** 对象中指定 **spec.type**，可以使用不同的方式公开服务：

ClusterIP

在集群中的内部 IP 地址上公开服务。**ClusterIP** 是默认服务类型。

NodePort

在集群中每个所选节点的同一端口上公开该服务。**NodePort** 使服务可从集群外部访问。

LoadBalancer

在当前云中创建外部负载均衡器（如果支持），并为该服务分配固定的外部 IP 地址。

8.17.2.1.1. 双栈支持

如果为集群启用了 IPv4 和 IPv6 双栈网络，您可以通过定义 **Service** 对象中的 **spec.ipFamilyPolicy** 和 **spec.ipFamilies** 字段来创建使用 IPv4、IPv6 或两者的服务。

spec.ipFamilyPolicy 字段可以设置为以下值之一：

SingleStack
: control plane 根据配置的第一个服务集群 IP 范围为该服务分配集群 IP 地址。

PreferDualStack
: control plane 为配置了双栈的集群中的服务分配 IPv4 和 IPv6 集群 IP 地址。

RequireDualStack
: 对于没有启用双栈网络的集群，这个选项会失败。对于配置了双栈的集群，其行为与将值设置为 **PreferDualStack** 时相同。control plane 从 IPv4 和 IPv6 地址范围分配集群 IP 地址。

您可以通过将 **spec.ipFamilies** 字段设置为以下数组值之一来定义用于单堆栈的 IP 系列，或者定义双栈 IP 系列的顺序：

- **[IPv4]**
- **[IPv6]**
- **[IPv4, IPv6]**
- **[IPv6, IPv4]**

8.17.2.2. 将虚拟机作为服务公开

创建 **ClusterIP**、**NodePort** 或 **LoadBalancer** 服务，以便从集群内部或外部连接到正在运行的虚拟机 (VM)。

流程

1. 编辑 **VirtualMachine** 清单，为创建服务添加标签：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-ephemeral
  namespace: example-namespace
spec:
  running: false
  template:
    metadata:
      labels:
        special: key (1)
# ...
```

(1) 在 **spec.template.metadata.labels** 部分添加标签 **special: key**。

注意

虚拟机上的标签会传递到 pod。**special: key** 标签必须与 **Service** 清单的 **spec.selector** 属性中的标签匹配。

2. 保存 **VirtualMachine** 清单文件以应用更改。

3. 创建 **Service** 清单以公开虚拟机：

```
apiVersion: v1
kind: Service
metadata:
  name: vmservice 1
  namespace: example-namespace 2
spec:
  externalTrafficPolicy: Cluster 3
  ports:
  - nodePort: 30000 4
    port: 27017
    protocol: TCP
    targetPort: 22 5
  selector:
    special: key 6
  type: NodePort 7
```

1. **Service** 对象的名称。
2. **Service** 对象所在的命名空间。这必须与 **VirtualMachine** 清单的 **metadata.namespace** 字段匹配。
3. 可选：指定节点如何分发在外部 IP 地址上接收的服务流量。这只适用于 **NodePort** 和 **LoadBalancer** 服务类型。默认值为 **Cluster**，它将流量平均路由到所有集群端点。
4. 可选：设置后，**nodePort** 值必须在所有服务间是唯一的。如果没有指定，则会动态分配以上 **30000** 范围中的值。
5. 可选：由服务公开的虚拟机端口。如果虚拟机清单中定义了端口列表，则必须引用打开端口。如果没有指定 **targetPort**，它将采用与 **port** 相同的值。
6. 对您在 **VirtualMachine** 清单的 **spec.template.metadata.labels** 小节中添加的标签的引用。
7. 服务的类型。可能的值有 **ClusterIP**、**NodePort** 和 **LoadBalancer**。

4. 保存 **Service** 清单文件。

5. 运行以下命令来创建服务：

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

6. 启动虚拟机。如果虚拟机已在运行，重启它。

验证

1. 查询 **Service** 对象以验证它是否可用：

```
$ oc get service -n example-namespace
```

ClusterIP 服务的输出示例

```
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)     AGE
vmservice   ClusterIP   172.30.3.149   <none>        27017/TCP   2m
```

NodePort 服务的输出示例

```
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)           AGE
vmservice   NodePort    172.30.232.73   <none>        27017:30000/TCP   5m
```

LoadBalancer 服务的输出示例

```
NAME        TYPE           CLUSTER-IP     EXTERNAL-IP                   PORT(S)           AGE
vmservice   LoadBalancer   172.30.27.5    172.29.10.235,172.29.10.235   27017:31829/TCP   5s
```

2. 选择连接到虚拟机的适当方法：

- 对于 **ClusterIP** 服务，使用服务 IP 地址和服务端口从集群内部连接到虚拟机。例如：

```
$ ssh fedora@172.30.3.149 -p 27017
```

- 对于 **NodePort** 服务，通过指定节点 IP 地址和集群网络之外的节点端口来连接到虚拟机。例如：

```
$ ssh fedora@$NODE_IP -p 30000
```

- 对于 **LoadBalancer** 服务，使用 **vinagre** 客户端使用公共 IP 地址和端口连接到虚拟机。外部端口是动态分配的。

8.17.2.3. 其他资源

- 使用 NodePort 配置集群入口流量
- 使用负载均衡器配置集群入口流量

8.17.3. 将虚拟机附加到 Linux 网桥网络

默认情况下，OpenShift Virtualization 安装了一个内部 pod 网络。

您必须创建一个 Linux 网桥网络附加定义(NAD)以连接到额外网络。

将虚拟机附加到额外网络：

1. 创建 Linux 网桥节点网络配置策略。
2. 创建 Linux 网桥网络附加定义。
3. 配置虚拟机，使虚拟机能够识别网络附加定义。

有关调度、接口类型和其他节点网络活动的更多信息，请参阅节点网络部分。

8.17.3.1. 通过网络附加定义连接到网络

8.17.3.1.1. 创建 Linux 网桥节点网络配置策略

使用 **NodeNetworkConfigurationPolicy** 清单 YAML 文件来创建 Linux 网桥。

流程

- 创建 **NodeNetworkConfigurationPolicy** 清单。本例包含示例值，您必须替换为您自己的信息。

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-policy (1)
spec:
  desiredState:
    interfaces:
      - name: br1 (2)
        description: Linux bridge with eth1 as a port (3)
        type: linux-bridge (4)
        state: up (5)
        ipv4:
          enabled: false (6)
        bridge:
          options:
            stp:
              enabled: false (7)
          port:
            - name: eth1 (8)
```

1. 策略的名称。
2. 接口的名称。
3. 可选：接口人类可读的接口描述。
4. 接口的类型。这个示例会创建一个桥接。
5. 创建后接口的请求状态。
6. 在这个示例中禁用 IPv4。
7. 在这个示例中禁用 STP。
8. 网桥附加到的节点 NIC。

8.17.3.2. 创建 Linux 网桥网络附加定义

8.17.3.2.1. 先决条件

- 必须在每个节点上配置并附加 Linux 网桥。如需更多信息，请参阅节点网络中的内容。

警告

不支持在虚拟机的网络附加定义中配置 IP 地址管理(IPAM)。

8.17.3.2.2. 在 web 控制台中创建 Linux 网桥网络附加定义

网络附加定义是一个自定义资源，可向 OpenShift Virtualization 集群中的特定命名空间公开第 2 层设备。

网络管理员可创建网络附加定义，以向 Pod 和虚拟机提供现有的第 2 层网络。

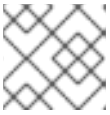
流程

1. 在 Web 控制台中，点击 **Networking → Network Attachment Definitions**。
2. 点 **Create Network Attachment Definition**。

注意

网络附加定义必须与 pod 或虚拟机位于同一个命名空间中。

3. 输入唯一 **Name** 和可选 **Description**。
4. 点击 **Network Type** 列表并选择 **CNV Linux bridge**。
5. 在 **Bridge Name** 字段输入网桥名称。
6. 可选：如果资源配置了 VLAN ID，请在 **VLAN Tag Number** 字段中输入 ID 号。
7. 可选： 选择 **MAC Spoof Check** 来启用 MAC spoof 过滤。此功能只允许单个 MAC 地址退出 pod，从而可以防止使用 MAC 欺骗进行的安全攻击。
8. 点 **Create**。

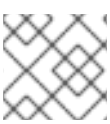


注意

Linux 网桥网络附加定义是将虚拟机连接至 VLAN 的最有效方法。

8.17.3.2.3. 在 CLI 中创建 Linux 网桥网络附加定义

作为网络管理员，您可以配置 **cnv-bridge** 类型的网络附加定义，为 Pod 和虚拟机提供第 2 层网络。



注意

网络附加定义必须与 pod 或虚拟机位于同一个命名空间中。

流程

1. 在与虚拟机相同的命名空间中创建网络附加定义。

2. 在网络附加定义中添加虚拟机，如下例所示：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: <bridge-network> (1)
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/<bridge-interface> (2)
spec:
  config: '{
    "cniVersion": "0.3.1",
    "name": "<bridge-network>", (3)
    "type": "cnv-bridge", (4)
    "bridge": "<bridge-interface>", (5)
    "macspoofchk": true, (6)
    "vlan": 1 (7)
  }'
```

(1) **NetworkAttachmentDefinition** 对象的名称。

(2) 可选：为节点选择注解键值对，其中 **bridge-interface** 是在某些节点上配置的桥接名称。如果在网络附加定义中添加此注解，您的虚拟机实例将仅在连接 **bridge-interface** 网桥的节点中运行。

(3) 配置的名称。建议您将配置名称与网络附加定义的 **name** 值匹配。

(4) 为这个网络附加定义的 Container Network Interface（CNI）插件的实际名称。不要更改此字段，除非要使用不同的 CNI。

(5) 节点上配置的 Linux 网桥名称。

(6) 可选：启用 MAC 欺骗检查。当设置为 **true** 时，您无法更改 pod 或客户机接口的 MAC 地址。此属性仅允许单个 MAC 地址退出容器集，从而防止 MAC 欺骗攻击。

(7) 可选： VLAN 标签。节点网络配置策略不需要额外的 VLAN 配置。

注意

Linux 网桥网络附加定义是将虚拟机连接至 VLAN 的最有效方法。

3. 创建网络附加定义：

```
$ oc create -f <network-attachment-definition.yaml> (1)
```

(1) 其中 **<network-attachment-definition.yaml>** 是网络附加定义清单的文件名。

验证

- 运行以下命令验证网络附加定义是否已创建：

```
$ oc get network-attachment-definition <bridge-network>
```

8.17.3.3. 为 Linux 网桥网络配置虚拟机

8.17.3.3.1. 在 web 控制台中为虚拟机创建 NIC

从 web 控制台创建并附加额外 NIC。

流程

1. 在 OpenShift Virtualization 控制台的正确项目中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点击 **Network Interfaces** 以显示已附加到虚拟机的 NIC。
5. 点 **Add Network Interface** 在列表中创建新插槽。
6. 使用 **Network** 下拉列表为额外网络选择网络附加定义。
7. 填写新 NIC 的 **Name**、**Model**、**Type** 和 **MAC Address**。
8. 点 **Add** 保存并附加 NIC 到虚拟机。

8.17.3.3.2. 网络字段

名称	描述
名称	网络接口控制器的名称。
model	指明网络接口控制器的型号。支持的值有 **e1000e** 和 **virtio**。
网络	可用网络附加定义的列表。
类型	可用绑定方法列表。对于默认的 pod 网络，**masquerade** 是唯一推荐的绑定方法。对于辅助网络，请使用 **bridge** 绑定方法。非默认网络不支持 **masquerade** 绑定方法。如果您配置了一个 **SR-IOV** 网络设备并在命名空间中定义那个网络，请选择 SR-IOV。
MAC 地址	网络接口控制器的 MAC 地址。如果没有指定 MAC 地址，则会自动分配一个。

8.17.3.3.3. 在 CLI 中将虚拟机附加到额外网络

通过添加桥接接口并在虚拟机配置中指定网络附加定义，将虚拟机附加到额外网络。

此流程使用 YAML 文件来演示编辑配置，并将更新的文件应用到集群。您还可以使用 **oc edit <object> <name>** 命令编辑现有虚拟机。

先决条件

- 在编辑配置前关闭虚拟机。如果编辑正在运行的虚拟机，您必须重启虚拟机才能使更改生效。

流程

1. 创建或编辑您要连接到桥接网络的虚拟机的配置。
2. 将网桥接口添加到 **spec.template.spec.domain.devices.interfaces** 列表中，把网络附加定义添加到 **spec.template.spec.networks** 列表中。这个示例添加了名为 **bridge-net** 的桥接接口，它连接到 **a-bridge-network** 网络附加定义：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
   name: <example-vm>
spec:
 template:
   spec:
     domain:
       devices:
         interfaces:
           - masquerade: {}
             name: <default>
           - bridge: {}
             name: <bridge-net> 1
...
     networks:
       - name: <default>
         pod: {}
       - name: <bridge-net> 2
         multus:
           networkName: <network-namespace>/<a-bridge-network> 3
...
```

1. 网桥接口的名称。
2. 网络的名称。这个值必须与对应的 **spec.template.spec.domain.devices.interfaces** 条目的 **name** 值匹配。
3. 网络附加定义的名称，并以其存在的命名空间为前缀。命名空间必须是 **default** 命名空间或创建虚拟机的同一命名空间。本例中使用 **multus**。Multus 是一个云网络接口(CNI)插件，它支持多个 CNI 存在，以便 pod 或虚拟机可以使用它所需的接口。

3. 应用配置：

```
$ oc apply -f <example-vm.yaml>
```

4. 可选：如果编辑了正在运行的虚拟机，您必须重启它才能使更改生效。

8.17.4. 为虚拟机配置 IP 地址

您可以为虚拟机配置动态或静态置备的 IP 地址。

先决条件

- 虚拟机必须连接到一个外部网络。
- 您必须在额外网络中有一个 DHCP 服务器，才能为虚拟机配置动态 IP。

8.17.4.1. 使用 cloud-init 为新虚拟机配置 IP 地址

在创建虚拟机时，您可以使用 cloud-init 来配置 IP 地址。IP 地址可以动态部署，也可以静态置备。

流程

- 创建虚拟机配置并在虚拟机配置的 **spec.volumes.cloudInitNoCloud.networkData** 字段中包含 cloud-init 网络详情：

  a. 要配置动态 IP，请指定接口名称和 **dhcp4** 布尔值：

```
kind: VirtualMachine
spec:
...
  volumes:
  - cloudInitNoCloud:
      networkData: |
        version: 2
        ethernets:
          eth1: 1
            dhcp4: true 2
```

  1 接口名称。

  2 使用 DHCP 来置备 IPv4 地址。

  b. 要配置静态 IP，请指定接口名称和 IP 地址：

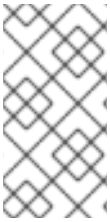
```
kind: VirtualMachine
spec:
...
  volumes:
  - cloudInitNoCloud:
      networkData: |
        version: 2
        ethernets:
          eth1: 1
            addresses:
            - 10.10.10.14/24 2
```

  1 接口名称。

  2 虚拟机的静态 IP 地址。

8.17.5. 为虚拟机配置 SR-IOV 网络设备

您可以为集群中的虚拟机配置单一根 I/O 虚拟化（SR-IOV）设备。此过程类似于为 OpenShift Container Platform 配置 SR-IOV 设备，当并不完全相同。



注意

只有在 HyperConverged Cluster 自定义资源（CR）中启用了 **sriovLiveMigration** 功能门时，附加到 SR-IOV 网络接口的虚拟机才支持实时迁移。当 **spec.featureGate.sriovLiveMigration 字** 段设置为 **true** 时，**virt-launcher** pod 使用 **SYS_RESOURCE** 功能运行。这可能会降低其安全性。

8.17.5.1. 先决条件

- 您必须已安装了 SR-IOV Operator。
- 您必须已配置了 SR-IOV Operator。

8.17.5.2. 自动发现 SR-IOV 网络设备

SR-IOV Network Operator 将搜索集群以获取 worker 节点上的 SR-IOV 功能网络设备。Operator 会为每个提供兼容 SR-IOV 网络设备的 worker 节点创建并更新一个 SriovNetworkNodeState 自定义资源 (CR)。

为 CR 分配了与 worker 节点相同的名称。**status.interfaces** 列表提供有关节点上网络设备的信息。

重要

不要修改 **SriovNetworkNodeState** 对象。Operator 会自动创建和管理这些资源。

8.17.5.2.1. SriovNetworkNodeState 对象示例

以下 YAML 是由 SR-IOV Network Operator 创建的 **SriovNetworkNodeState** 对象的示例：

一个 SriovNetworkNodeState 对象

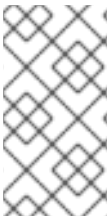
```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodeState
metadata:
  name: node-25 (1)
  namespace: openshift-sriov-network-operator
  ownerReferences:
  - apiVersion: sriovnetwork.openshift.io/v1
    blockOwnerDeletion: true
    controller: true
    kind: SriovNetworkNodePolicy
    name: default
spec:
  dpConfigVersion: "39824"
status:
  interfaces: (2)
  - 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: 158b
    driver: i40e
    mtu: 1500
    name: ens817f0
    pciAddress: 0000:81:00.0
    totalvfs: 64
    vendor: "8086"
  - deviceID: 158b
    driver: i40e
    mtu: 1500
    name: ens817f1
    pciAddress: 0000:81:00.1
    totalvfs: 64
    vendor: "8086"
  - deviceID: 158b
    driver: i40e
    mtu: 1500
    name: ens803f0
    pciAddress: 0000:86:00.0
    totalvfs: 64
    vendor: "8086"
  syncStatus: Succeeded
```

1. **name** 字段的值与 worker 节点的名称相同。
2. **interfaces** 小节包括 Operator 在 worker 节点上发现的所有 SR-IOV 设备列表。

8.17.5.3. 配置 SR-IOV 网络设备

SR-IOV Network Operator 把 **SriovNetworkNodePolicy.sriovnetwork.openshift.io** CRD 添加到 OpenShift Container Platform。您可以通过创建一个 SriovNetworkNodePolicy 自定义资源 (CR) 来配置 SR-IOV 网络设备。



注意

在应用由 **SriovNetworkNodePolicy** 对象中指定的配置时，SR-IOV Operator 可能会排空节点，并在某些情况下会重启节点。

它可能需要几分钟时间来应用配置更改。

先决条件

- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- 已安装 SR-IOV Network Operator。
- 集群中有足够的可用节点，用于处理从排空节点中驱除的工作负载。
- 您还没有为 SR-IOV 网络设备配置选择任何 control plane 节点。

流程

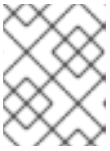
1. 创建一个 **SriovNetworkNodePolicy** 对象，然后在 **<name>-sriov-node-network.yaml** 文件中保存 YAML。使用配置的实际名称替换 **<name>**。

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
  name: <name> 1
  namespace: openshift-sriov-network-operator 2
spec:
  resourceName: <sriov_resource_name> 3
  nodeSelector:
    feature.node.kubernetes.io/network-sriov.capable: "true" 4
  priority: <priority> 5
  mtu: <mtu> 6
  numVfs: <num> 7
  nicSelector: 8
    vendor: "<vendor_code>" 9
    deviceID: "<device_id>" 10
    pfNames: ["<pf_name>", ...] 11
    rootDevices: ["<pci_bus_id>", "..."] 12
  deviceType: vfio-pci 13
  isRdma: false 14
```

1. 为 CR 对象指定一个名称。
2. 指定 SR-IOV Operator 安装到的命名空间。
3. 指定 SR-IOV 设备插件的资源名称。您可以为一个资源名称创建多个 **SriovNetworkNodePolicy** 对象。
4. 指定节点选择器来选择要配置哪些节点。只有所选节点上的 SR-IOV 网络设备才会被配置。SR-IOV Container Network Interface（CNI）插件和设备插件仅在所选节点上部署。
5. 可选：指定一个 **0** 到 **99** 之间的整数。较小的数值具有较高的优先权，优先级 **10** 高于优先级 **99**。默认值为 **99**。
6. 可选：为虚拟功能（VF）的最大传输单位 (MTU) 指定一个值。最大 MTU 值可能因不同的 NIC 型号而有所不同。
7. 为 SR-IOV 物理网络设备指定要创建的虚拟功能 (VF) 的数量。对于 Intel 网络接口控制器（NIC），VF 的数量不能超过该设备支持的 VF 总数。对于 Mellanox NIC，VF 的数量不能超过 **128**。
8. **nicSelector** 映射为 Operator 选择要配置的以太网设备。您不需要为所有参数指定值。建议您以足够的准确度来识别以太网适配器，以便尽量减小意外选择其他以太网设备的可能性。如果指定了**rootDevices**，则必须同时为 **vendor**、 **deviceID** 或 **pfNames** 指定一个值。如

果同时指定了 **pfNames** 和 **rootDevices**，请确保它们指向同一个设备。

9. 可选：指定 SR-IOV 网络设备的厂商十六进制代码。允许的值只能是 **8086** 或 **15b3**。
10. 可选：指定 SR-IOV 网络设备的设备十六进制代码。允许的值只能是 **158b**、**1015**、**1017**。
11. 可选：参数接受包括以太网设备的一个或多个物理功能 (PF) 的数组。
12. 参数接受一个包括一个或多个 PCI 总线地址，用于以太网设备的物理功能的数组。使用以下格式提供地址: **0000:02:00.1**。
13. OpenShift Virtualization 中的虚拟功能需要 **vfio-pci** 驱动程序类型。
14. 可选：指定是否启用远程直接访问（RDMA）模式。对于 Mellanox 卡，请将 **isRdma** 设置为 **false**。默认值为 **false**。



注意

如果将 **RDMA** 标记设定为 **true**，您可以继续使用启用了 RDMA 的 VF 作为普通网络设备。设备可在其中的一个模式中使用。

2. 可选：将 SR-IOV 功能的集群节点标记为 **SriovNetworkNodePolicy.Spec.NodeSelector**（如果它们还没有标记）。有关标记节点的更多信息，请参阅"了解如何更新节点上的标签"。
3. 创建 **SriovNetworkNodePolicy** 对象：

```
$ oc create -f <name>-sriov-node-network.yaml
```

其中 **<name>** 指定这个配置的名称。

在应用配置更新后，**sriov-network-operator** 命名空间中的所有 Pod 都会变为 **Running** 状态。

4. 要验证是否已配置了 SR-IOV 网络设备，请输入以下命令。将 **<node_name>** 替换为带有您刚才配置的 SR-IOV 网络设备的节点名称。

```
$ oc get sriovnetworknodestates -n openshift-sriov-network-operator <node_name> -o
jsonpath='{.status.syncStatus}'
```

8.17.5.4. 后续步骤

- 为虚拟机配置一个 SR-IOV 网络附件

8.17.6. 定义 SR-IOV 网络

您可以为虚拟机的单根 I/O 虚拟化（SR-IOV）设备创建一个网络附加。

定义网络后，您可以将虚拟机附加到 SR-IOV 网络。

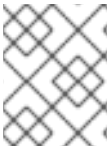
8.17.6.1. 先决条件

- 您必须已 为虚拟机配置了 SR-IOV 设备。

8.17.6.2. 配置 SR-IOV 额外网络

您可以通过创建一个 **SriovNetwork** 对象来配置使用 SR-IOV 硬件的额外网络。创建 **SriovNetwork** 对象时，SR-IOV Operator 会自动创建一个 **NetworkAttachmentDefinition** 对象。

然后，用户可以通过在虚拟机配置中指定网络将虚拟机附加到 SR-IOV 网络中。



注意

如果一个 **SriovNetwork** 对象已被附加到状态为 **running** 的 Pod 或虚拟机上，则不能修改或删除它。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 特权的用户身份登录。

流程

1. 创建以下 **SriovNetwork** 对象，然后在 **<name>-sriov-network.yaml** 文件中保存 YAML。用这个额外网络的名称替换 **<name>** 。

```
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. 将 **<name>** 替换为对象的名称。SR-IOV Network Operator 创建一个名称相同的 **NetworkAttachmentDefinition** 对象。
2. 指定 SR-IOV Network Operator 安装到的命名空间。
3. 将 **<sriov_resource_name>** 替换为来自为这个额外网络定义 SR-IOV 硬件的 **SriovNetworkNodePolicy** 对象的 **.spec.resourceName** 参数的值。
4. 将 **<target_namespace>** 替换为 SriovNetwork 的目标命名空间。只有目标命名空间中的 pod 或虚拟机可以附加到 SriovNetwork。
5. 可选：使用额外网络的虚拟 LAN (VLAN) ID 替换 **<vlan>**。它需要是一个从 **0** 到 **4095** 范围内的一个整数值。默认值为 **0**。
6. 可选：将 **<spoof_check>** 替换为 VF 的 spoof 检查模式。允许的值是字符串 **"on"** 和 **"off"**。

重要

指定的值必须由引号包括，否则 SR-IOV Network Operator 将拒绝 CR。

7 可选：将 **<link_state>** 替换为 Virtual Function (VF) 的链接状态。允许的值是 **enable**、**disable** 和 **auto**。

8 可选：将 **<max_tx_rate>** 替换为 VF 的最大传输率（以 Mbps 为单位）。

9 可选：将 **<min_tx_rate>** 替换为 VF 的最小传输率（以 Mbps 为单位）。这个值应该总是小于或等于最大传输率。

注意

Intel NIC 不支持 **minTxRate** 参数。如需更多信息，请参阅 BZ#1772847。

10 可选：将 **<vlan_qos>** 替换为 VF 的 IEEE 802.1p 优先级级别。默认值为 **0**。

11 可选：将 **<trust_vf>**替换为 VF 的信任模式。允许的值是字符串 **"on"** 和 **"off"**。

重要

指定的值必须由引号包括，否则 SR-IOV Network Operator 将拒绝 CR。

12 可选：将 **<capabilities>** 替换为为这个网络配置的功能。

2. 运行以下命令来创建对象。用这个额外网络的名称替换 **<name>** 。

```
$ oc create -f <name>-sriov-network.yaml
```

3. 可选： 要确认与您在上一步中创建的 **SriovNetwork** 对象关联的 **NetworkAttachmentDefinition** 对象是否存在，请输入以下命令。将 **<namespace>** 替换为您在 **SriovNetwork** 对象中指定的命名空间。

```
$ oc get net-attach-def -n <namespace>
```

8.17.6.3. 后续步骤

- 为 SR-IOV 网络附加一个虚拟机。

8.17.7. 把一个虚拟机附加到一个 SR-IOV 网络

您可以附加虚拟机以使用单根 I/O 虚拟化（SR-IOV）网络作为二级网络。

8.17.7.1. 先决条件

- 您必须已为虚拟机配置了 SR-IOV 设备。
- 您必须已定义了一个 SR-IOV 网络。

8.17.7.2. 把一个虚拟机附加到一个 SR-IOV 网络

您可以通过在虚拟机配置中包含网络详情将虚拟机附加到 SR-IOV 网络中。

流程

1. 在虚拟机配置的 **spec.domain.devices.interfaces** 和 **spec.networks** 中包括 SR-IOV 网络详情：

```
kind: VirtualMachine
...
spec:
  domain:
    devices:
      interfaces:
      - name: <default> 1
        masquerade: {} 2
      - name: <nic1> 3
        sriov: {}
  networks:
  - name: <default> 4
    pod: {}
  - name: <nic1> 5
    multus:
        networkName: <sriov-network> 6
...
```

1. 连接到 pod 网络的接口的唯一名称。
2. **masquerade** 绑定到默认 pod 网络。
3. SR-IOV 接口的唯一名称。
4. pod 网络接口的名称。这必须与之前定义的 **interfaces.name** 相同。
5. SR-IOV 接口的名称。这必须与之前定义的 **interfaces.name** 相同。
6. SR-IOV 网络附加定义的名称。

2. 应用虚拟机配置：

```
$ oc apply -f <vm-sriov.yaml> 1
```

1. 虚拟机 YAML 文件的名称。

8.17.8. 在虚拟机上查看 NIC 的 IP 地址

您可以使用 Web 控制台或 **oc** 客户端查看网络接口控制器 (NIC) 的 IP 地址。 QEMU 客户机代理显示有关虚拟机辅助网络的附加信息。

8.17.8.1. 在 CLI 中查看虚拟机接口的 IP 地址

oc describe vmi <vmi_name> 命令中包含网络接口配置。

您还可通过在虚拟机上运行 **ip addr** 或通过运行 **oc get vmi <vmi_name> -o yaml** 来查看 IP 地址信息。

流程

- 使用 **oc describe** 命令来显示虚拟机接口配置：

```
$ oc describe vmi <vmi_name>
```

输**出示例**

```
...
Interfaces:
  Interface Name:  eth0
  Ip Address:      10.244.0.37/24
  Ip Addresses:
    10.244.0.37/24
    fe80::858:aff:fef4:25/64
  Mac:             0a:58:0a:f4:00:25
  Name:            default
  Interface Name:  v2
  Ip Address:      1.1.1.7/24
  Ip Addresses:
    1.1.1.7/24
    fe80::f4d9:70ff:fe13:9089/64
  Mac:             f6:d9:70:13:90:89
  Interface Name:  v1
  Ip Address:      1.1.1.1/24
  Ip Addresses:
    1.1.1.1/24
    1.1.1.2/24
    1.1.1.4/24
    2001:de7:0:f101::1/64
    2001:db8:0:f101::1/64
    fe80::1420:84ff:fe10:17aa/64
  Mac:             16:20:84:10:17:aa
```

8.17.8.2. 在 web 控制台中查看虚拟机接口的 IP 地址

IP 信息显示在虚拟机的 **Virtual Machine Overview** 屏幕中。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机名称以打开 **Virtual Machine Overview** 屏幕。

每个附加 NIC 的信息会显示在 **IP Address** 下。

8.17.9. 为虚拟机使用 MAC 地址池

KubeMacPool 组件为命名空间中的虚拟机 NIC 提供 MAC 地址池服务。

8.17.9.1. 关于 KubeMacPool

KubeMacPool 为每个命名空间提供一个 MAC 地址池，并从池中为虚拟机 NIC 分配 MAC 地址。这样可确保为 NIC 分配一 个唯一的 MAC 地址，该地址与另一个虚拟机的 MAC 地址不会有冲突。

从该虚拟机创建的虚拟机实例会在重启后保留分配的 MAC 地址。

注意

KubeMacPool 不处理独立于虚拟机创建的虚拟机实例。

安装 OpenShift Virtualization 时， KubeMacPool 会被默认启用。您可以通过将 **mutatevirtualmachines.kubemacpool.io=ignore** 标签添加到命名空间来在命名空间中禁用一个 MAC 地址池。通过删除标签为命名空间重新启用 KubeMacPool。

8.17.9.2. 在 CLI 中为命名空间禁用 MAC 地址池

通过将 **mutatevirtualmachines.kubemacpool.io=ignore** 标签添加到命名空间来禁用命名空间中虚拟机的 MAC 地址池。

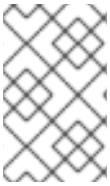
流程

- 将 **mutatevirtualmachines.kubemacpool.io=ignore** 标签添加到命名空间。以下示例为 **<namespace1>** 和 **<namespace2>** 这两个命名空间禁用 KubeMacPool：

```
$ oc label namespace <namespace1> <namespace2>
mutatevirtualmachines.kubemacpool.io=ignore
```

8.17.9.3. 在 CLI 中为命名空间重新启用 MAC 地址池

如果您为命名空间禁用 KubeMacPool 并希望重新启用它，请从命名空间中删除 **mutatevirtualmachines.kubemacpool.io=ignore** 标签。



注意

早期版本的 OpenShift Virtualization 使用标签 **mutatevirtualmachines.kubemacpool.io=allocate** 为命名空间启用 KubeMacPool。这仍然被支持，但是冗余的，因为 KubeMacPool 现在被默认启用。

流程

- 从命名空间中删除 KubeMacPool 标签。以下示例为 **<namespace1>** 和 **<namespace2>** 这两个命名空间重新启用 KubeMacPool：

```
$ oc label namespace <namespace1> <namespace2>
mutatevirtualmachines.kubemacpool.io-
```

8.18. 虚拟机磁盘

8.18.1. 存储特性

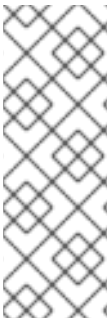
使用下表来决定 OpenShift Virtualization 中的本地和共享持久性存储的功能可用性。

8.18.1.1. OpenShift Virtualization 存储功能列表

表 8.7. OpenShift Virtualization 存储功能列表

	虚拟机实时迁移	主机辅助虚拟机磁盘克隆	存储辅助虚拟机磁盘克隆	虚拟机快照
OpenShift Container Storage: RBD 块模式卷	是	是	是	是
OpenShift Virtualization hostpath 置备程序	不是	是	否	否
其他多节点可写入存储	是 [1]	是	是 [2]	是 [2]
其他单节点可写入存储	否	是	是 [2]	是 [2]

1. PVC 必须请求 ReadWriteMany 访问模式。
2. 存储供应商必须支持 Kubernetes 和 CSI 快照 API



注意

您无法实时迁移使用以下配置的虚拟机：

- 具有 ReadWriteOnce（RWO）访问模式的存储类
- 透传功能（如 GPU 或 SR-IOV 网络接口）带有 **sriovLiveMigration** 功能被禁用。

对于这些虚拟机，不要将 **evictionStrategy** 字段设置为 **LiveMigrate**。

8.18.2. 为虚拟机配置本地存储

您可以使用 hostpath 置备程序功能为您的虚拟机配置本地存储。

8.18.2.1. 关于 hostpath 置备程序

hostpath 置备程序是设计用于 OpenShift Virtualization 的本地存储置备程序。如果要为虚拟机配置本地存储，您必须首先启用 hostpath 置备程序。

安装 OpenShift Virtualization Operator 时，会自动安装 hostpath 置备程序 Operator。要使用它，您必须：

- 配置 SELinux：
  - 如果使用 Red Hat Enterprise Linux CoreOS(RHCOS)8 worker，您必须在每个节点上创建一个 **MachineConfig** 对象。
  - 否则，将 SELinux 标签 **container_file_t** 应用到每个节点上的由持久性卷（PV）支持的目录中。
- 创建 **HostPathProvisioner** 自定义资源。

- 为 hostpath 置备程序创建 **StorageClass** 对象。

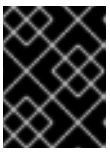
在创建自定义资源时，hostpath 置备程序 Operator 将部署置备程序作为每个节点上的 *DaemonSet*。在自定义资源文件中，您将指定 hostpath 置备程序创建的持久性卷的后端目录。

8.18.2.2. 在 Red Hat Enterprise Linux CoreOS(RHCOS)8 上为 hostpath 置备程序配置 SELinux

在创建 **HostPathProvisioner** 自定义资源前，您必须配置 SELinux。要在 Red Hat Enterprise Linux CoreOS(RHCOS)8 worker 上配置 SELinux，您必须在每个节点上创建一个 **MachineConfig** 对象。

先决条件

- 在每个节点上为 hostpath 置备程序创建的持久性卷（PV）创建后端目录。



重要

后备目录不得位于文件系统的根目录中，因为 / 分区在 RHCOS 中是只读的。例如，您可以使用 **/var/<directory_name>** 而不是 **/<directory_name>**。

警告

如果您选择了与您的操作系统共享空间的目录，则可能会耗尽该分区中的空间，且节点可能会无法正常工作。创建单独的分区，并将 hostpath 置备程序指向单独的分区，以避免干扰您的操作系统。

流程

1. 创建 **MachineConfig** 文件。例如：

```
$ touch machineconfig.yaml
```

2. 编辑该文件，确保包含希望 hostpath 置备程序在其中创建 PV 的目录。例如：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 50-set-selinux-for-hostpath-provisioner
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - contents: |
            [Unit]
            Description=Set SELinux chcon for hostpath provisioner
            Before=kubelet.service
```

```
    [Service]
    ExecStart=/usr/bin/chcon -Rt container_file_t <backing_directory_path> 1

    [Install]
    WantedBy=multi-user.target
  enabled: true
  name: hostpath-provisioner.service
```

1 指定希望置备程序在其中创建 PV 的后备目录。该目录不能位于文件系统的根目录（/）中。

3. 创建 **MachineConfig** 对象：

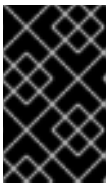
```
$ oc create -f machineconfig.yaml -n <namespace>
```

8.18.2.3. 使用 hostpath 置备程序启用本地存储

要部署 hostpath 置备程序并使虚拟机能够使用本地存储，请首先创建一个 **HostPathProvisioner** 自定义资源。

先决条件

- 在每个节点上为 hostpath 置备程序创建的持久性卷（PV）创建后端目录。



重要

后备目录不得位于文件系统的根目录中，因为 / 分区在 Red Hat Enterprise Linux CoreOS(RHCOS)中是只读的。例如，您可以使用 **/var/<directory_name>** 而不是 **/<directory_name>**。

警告

如果您选择了与您的操作系统共享空间的目录，则可能会耗尽该分区中的空间，且节点可能会无法正常工作。创建单独的分区，并将 hostpath 置备程序指向单独的分区，以避免干扰您的操作系统。

- 将 SELinux 上下文 **container_file_t** 应用到每个节点上的 PV 后备目录。例如：

```
$ sudo chcon -t container_file_t -R <backing_directory_path>
```

注意

如果使用 Red Hat Enterprise Linux CoreOS(RHCOS)8 worker，则必须使用 **MachineConfig** 清单配置 SELinux。

流程

1. 创建 **HostPathProvisioner** 自定义资源文件。例如：

```
$ touch hostpathprovisioner_cr.yaml
```

2. 编辑该文件，确保 **spec.pathConfig.path** 值是您希望 hostpath 置备程序在其中创建 PV 的目录。例如：

```
apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
kind: HostPathProvisioner
metadata:
  name: hostpath-provisioner
spec:
  imagePullPolicy: IfNotPresent
  pathConfig:
    path: "<backing_directory_path>" 1
    useNamingPrefix: false 2
  workload: 3
```

1. 指定希望置备程序在其中创建 PV 的后备目录。该目录不能位于文件系统的根目录（/）中。
2. 如果要使用绑定到所创建的 PV 的持久性卷声明（PVC）名称作为目录名的前缀，请将该值更改为 **true**。
3. 可选：您可以使用 **spec.workload** 字段为 hostpath 置备程序配置节点放置规则。

注意

如果您没有创建后备目录，则置备程序会尝试为您创建该目录。如果您没有应用 **container_file_t** SELinux 上下文，这会导致 **Permission denied**。

3. 在 **openshift-cnv** 命名空间中创建自定义资源：

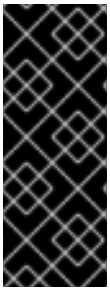
```
$ oc create -f hostpathprovisioner_cr.yaml -n openshift-cnv
```

其他资源

- 为虚拟化组件指定节点

8.18.2.4. 创建存储类

当您创建存储类时，您将设置参数，它们会影响属于该存储类的持久性卷(PV)的动态置备。您不能在创建 **StorageClass** 对象后更新其参数。



重要

在 OpenShift Container Platform Container Storage 中使用 OpenShift Virtualization 时，指定创建虚拟机磁盘时 RBD 块模式持久性卷声明(PVC)。使用虚拟机磁盘时，RBD 块模式卷更高效，并且比 Ceph FS 或 RBD 文件系统模式 PVC 提供更好的性能。

要指定 RBD 块模式 PVC，请使用 'ocs-storagecluster-ceph-rbd' 存储类和 **VolumeMode: Block**。

流程

1. 创建用于定义存储类的 YAML 文件。例如：

```
$ touch storageclass.yaml
```

2. 编辑该文件。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: hostpath-provisioner (1)
provisioner: kubevirt.io/hostpath-provisioner
reclaimPolicy: Delete (2)
volumeBindingMode: WaitForFirstConsumer (3)
```

(1) 您可以通过更改此值来选择性地重命名存储类。

(2) 两个可能的 **reclaimPolicy** 值为 **Delete** 和 **Retain**。如果您没有指定值，则存储类默认为 **Delete**。

(3) **volumeBindingMode** 值决定何时发生动态置备和卷绑定。指定 **WaitForFirstConsumer**，将 PV 的绑定和置备延迟到创建使用 PVC 的 Pod 后。这样可确保 PV 满足 pod 的调度要求。

注意

虚拟机使用基于本地 PV 的数据卷。本地 PV 与特定节点绑定。虽然磁盘镜像准备供虚拟机消耗，但可能不会将虚拟机调度到之前固定本地存储 PV 的节点。

要解决这个问题，使用 Kubernetes pod 调度程序将 PVC 绑定到正确的节点上的 PV。通过使用 **volumeBindingMode** 设置为 **WaitForFirstConsumer** 的 **StorageClass**，PV 的绑定和置备会延迟到 **Pod** 创建前。

3. 创建 **StorageClass** 对象：

```
$ oc create -f storageclass.yaml
```

其他资源

- 存储类

8.18.3. 使用配置集创建数据卷

当您创建数据卷时，Containerized Data Importer(CDI)会创建一个持久性卷声明(PVC)，并使用您的数据填充 PVC。您可以将数据卷创建为独立资源，也可以使用虚拟机规格中的 **dataVolumeTemplate** 资源。您可以使用 PVC API 或存储 API 创建数据卷。

重要

在 OpenShift Container Platform Container Storage 中使用 OpenShift Virtualization 时，指定创建虚拟机磁盘时 RBD 块模式持久性卷声明(PVC)。使用虚拟机磁盘时，RBD 块模式卷更高效，并且比 Ceph FS 或 RBD 文件系统模式 PVC 提供更好的性能。

要指定 RBD 块模式 PVC，请使用 'ocs-storagecluster-ceph-rbd' 存储类和 **VolumeMode: Block**。

提示

在可能的情况下，使用存储 API 来优化空间分配并最大限度地提高性能。

*存储配置集*是 CDI 管理的自定义资源。它根据关联的存储类提供推荐的存储设置。为每个存储类分配一个存储配置文件。

存储配置集可让您快速创建数据卷，同时减少编码并最大程度减少潜在的错误。

对于可识别的存储类型，CDI 提供优化 PVC 创建的值。但是，如果您自定义存储配置集，您可以为存储类配置自动设置。

8.18.3.1. 使用存储 API 创建数据卷

当您使用存储 API 创建数据卷时，Containerized Data Interface（CDI）会根据所选存储类支持的存储类型优化持久性卷声明（PVC）分配。您只需要指定数据卷名称、命名空间和要分配的存储量。

例如：

- 使用 Ceph RBD 时，**accessModes** 会自动设置为 **ReadWriteMany**，这将启用实时迁移。**volumeMode** 设置为 **Block** 以最大化性能。
- 当使用 **volumeMode: Filesystem** 时，如果需要满足文件系统开销,CDI 将自动请求更多空间。

在以下 YAML 中，使用存储 API 请求具有 2G 可用空间的数据卷。用户不需要知道 **volumeMode** 就可正确估算所需的持久性卷声明（PVC）大小。CDI 自动选择 **accessModes** 和 **volumeMode** 属性的最佳组合。这些最佳值基于存储类型或您在存储配置文件中定义的默认值。如果要提供自定义值，它们会覆盖系统计算的值。

DataVolume 定义示例

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <datavolume> 1
spec:
  source:
    pvc: 2
      namespace: "<source_namespace>" 3
      name: "<my_vm_disk>" 4
  storage: 5
    resources:
      requests:
        storage: 2Gi 6
    storageClassName: <storage_class> 7
```

1. 新数据卷的名称。
2. 指明导入的来源是一个现有的持久性卷声明（PVC）。
3. 源 PVC 所在的命名空间。
4. 源 PVC 的名称。
5. 表示使用存储 API 的分配。
6. 指定您为 PVC 请求的可用空间量。
7. 可选：存储类的名称。如果没有指定存储类，则会使用系统默认存储类。

8.18.3.2. 使用 PVC API 创建数据卷

当使用 PVC API 创建数据卷时，Containerized Data Interface（CDI）会根据您为以下字段指定的内容创建数据卷：

- **accessModes**（**ReadWriteOnce**、**ReadWriteMany** 或 **ReadOnlyMany**）
- **volumeMode**（**Filesystem** 或 **Block**）
- **capacity** of **storage**（例如，**5Gi**）

在以下 YAML 中，使用 PVC API 分配存储容量为 2GB 的数据卷。您可以指定 **ReadWriteMany** 访问模式来启用实时迁移。因为您知道系统可以支持的值，所以可以指定 **Block** 存储而不是默认的 **Filesystem**。

DataVolume 定义示例

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <datavolume> 1
spec:
  source:
    pvc: 2
      namespace: "<source_namespace>" 3
      name: "<my_vm_disk>" 4
  pvc: 5
    accessModes: 6
      - ReadWriteMany
    resources:
      requests:
        storage: 2Gi 7
    volumeMode: Block 8
    storageClassName: <storage_class> 9
```

1. 新数据卷的名称。
2. 在 **source** 部分中，**pvc** 表示导入的来源是一个现有的持久性卷声明（PVC）。
3. 源 PVC 所在的命名空间。

4 源 PVC 的名称。

5 表示使用 PVC API 的分配。

6 使用 PVC API 时需要 **accessModes**。

7 指定为数据卷请求的空间量。

8 指定目标为一个块 PVC。

9 另外，还可指定存储类。如果没有指定存储类，则会使用系统默认存储类。

重要

当您使用 PVC API 明确分配数据卷且没有使用 **volumeMode: Block** 时，请考虑文件系统开销。

文件系统开销是文件系统维护其元数据所需的空间量。文件系统元数据所需的空间量取决于文件系统。无法考虑存储容量请求中的文件系统开销会导致底层持久性卷声明（PVC）不足以容纳您的虚拟机磁盘。

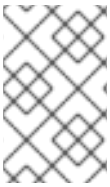
如果您使用存储 API，CDI 将包含在文件系统开销中，并请求更大的持久性卷声明（PVC）以确保您的分配请求成功。

8.18.3.3. 自定义存储配置集

您可以通过编辑置备程序存储类的 **StorageProfile** 对象来指定默认参数。这些默认参数只有在 **DataVolume** 对象中没有配置持久性卷声明 (PVC) 时才适用。

先决条件

- 确保存储类及其供应商支持您计划的配置。在存储配置集中指定不兼容的配置会导致卷置备失败。



注意

存储配置文件中的空 **status** 部分表示存储置备程序不被 Containerized Data Interface（CDI）识别。如果您有一个存储置备程序无法被 CDI 识别，则需要自定义存储配置集。在这种情况下，管理员在存储配置集中设置适当的值以确保分配成功。

警告

如果您创建数据卷并省略 YAML 属性，且存储配置集中没有定义这些属性，则不会分配请求的存储，也不会创建底层持久性卷声明（PVC）。

流程

1. 编辑存储配置文件。在本例中，CDI 不支持置备程序：

```
$ oc edit -n openshift-cnv storageprofile <storage_class>
```

存储配置集示例

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: StorageProfile
metadata:
  name: <some_unknown_provisioner_class>
#   ...
spec: {}
status:
  provisioner: <some_unknown_provisioner>
  storageClass: <some_unknown_provisioner_class>
```

2. 在存储配置集中提供所需的属性值：

存储配置集示例

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: StorageProfile
metadata:
  name: <some_unknown_provisioner_class>
#   ...
spec:
  claimPropertySets:
  - accessModes:
    - ReadWriteOnce (1)
    volumeMode:
      Filesystem (2)
status:
  provisioner: <some_unknown_provisioner>
  storageClass: <some_unknown_provisioner_class>
```

(1) 您选择的 **accessModes**。

(2) 您选择的 **volumeMode**。

保存更改后，所选值将显示在存储配置集的 **status** 项中。

8.18.3.4. 其他资源

- 创建存储类
- 覆盖默认文件系统开销值
- 使用 smart-cloning（智能克隆）克隆数据卷

8.18.4. 为文件系统开销保留 PVC 空间

默认情况下，Containerized Data Importer（CDI）会在使用 **Filesystem** 卷模式的持久性卷声明（PVC）中为文件系统的开销数据预留空间。您可以在全局范围内设置 CDI 保留的百分比，并用于特定存储类。

8.18.4.1. 文件系统的开销对虚拟机磁盘空间的影响

当您向使用 **Filesystem** 卷模式的持久性卷声明（PVC）添加虚拟机磁盘时，必须确保 PVC 中有足够的空间用于：

- 虚拟机磁盘。
- 用于文件系统开销（如元数据）的 Containerized Data Importer（CDI）保留空间。

默认情况下，CDI 为开销保留 5.5% 的 PVC 空间，从而减少了虚拟机磁盘的可用空间。

您可以编辑 **CDI** 对象来配置使用不同的开销值。您可以在全局范围内更改值，也可以为特定存储类指定值。

8.18.4.2. 覆盖默认文件系统开销值

通过编辑 **CDI** 对象的 **spec.config.filesystemOverhead** 属性来更改 Containerized Data Importer（CDI）为文件系统开销保留的持久性卷声明（PVC）空间大小。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

1. 运行以下命令打开 **CDI** 对象进行编辑：

```
$ oc edit cdi
```

2. 编辑 **spec.config.filesystemOverhead** 字段，使用您选择的值填充它们：

```
...
spec:
  config:
    filesystemOverhead:
      global: "<new_global_value>" 1
      storageClass:
        <storage_class_name>: "<new_value_for_this_storage_class>" 2
```

1. CDI 在集群中使用的文件系统开销百分比。例如，**global: "0.07"** 为文件系统开销保留 PVC 的 7%。
2. 指定存储类的文件系统开销百分比。例如，**mystorageclass: "0.04"** 将 **mystorageclass** 存储类中 PVC 的默认开销值改为 4%。

3. 保存并退出编辑器以更新 **CDI** 对象。

验证

- 运行以下命令来查看 **CDI** 状态并验证您的更改：

```
$ oc get cdi -o yaml
```

8.18.5. 配置 CDI 以使用具有计算资源配额的命名空间

您可以使用 Containerized Data Importer（CDI）将虚拟机磁盘导入、上传并克隆到命名空间中，这可能受 CPU 和内存资源限制。

8.18.5.1. 关于命名空间中的 CPU 和内存配额

*资源配额*由 **ResourceQuota** 对象定义，对一个命名空间实施限制，该命名空间限制可被该命名空间中资源消耗的计算资源总量。

HyperConverged 自定义资源（CR）定义了 Containerized Data Importer（CDI）的用户配置。CPU 和内存请求和限制值设置为默认值 **0**。这样可确保由 CDI 创建的无需计算资源要求的 Pod 具有默认值，并允许在使用配额限制的命名空间中运行。

8.18.5.2. 覆盖 CPU 和内存默认值

通过将 **spec.resourceRequirements.storageWorkloads** 小节添加到 **HyperConverged** 自定义资源（CR），为您的用例修改 CPU 和内存请求和限值的默认设置。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

1. 运行以下命令来编辑 **HyperConverged** CR：

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. 将 **spec.resourceRequirements.storageWorkloads** 小节添加到 CR，根据您的用例设置值。例如：

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  resourceRequirements:
    storageWorkloads:
      limits:
        cpu: "500m"
        memory: "2Gi"
      requests:
        cpu: "250m"
        memory: "1Gi"
```

3. 保存并退出编辑器以更新 **HyperConverged** CR。

8.18.5.3. 其他资源

- 项目的资源配额

8.18.6. 管理数据卷注解

数据卷（DV）注解允许您管理 pod 行为。您可以将一个或多个注解添加到数据卷，然后将其传播到创建的导入程序 pod。

8.18.6.1. 示例：数据卷注解

本例演示了如何配置数据卷（DV）注解来控制 importer pod 使用的网络。**v1.multus-cni.io/default-network: bridge-network** 注解会导致 pod 使用名为 **bridge-network** 的 multus 网络作为其默认网络。如果您希望 importer pod 使用集群中的默认网络和从属 multus 网络，请使用 **k8s.v1.cni.cncf.io/networks: <network_name>** 注解。

Multus 网络注解示例

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: dv-ann
  annotations:
    v1.multus-cni.io/default-network: bridge-network (1)
spec:
  source:
    http:
      url: "example.exampleurl.com"
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 1Gi
```

(1) Multus 网络注解

8.18.7. 对数据卷使用预分配

Containerized Data Importer 可以预先分配磁盘空间，以便在创建数据卷时提高写入性能。

您可以为特定数据卷启用预分配。

8.18.7.1. 关于预分配

Containerized Data Importer（CDI）可以使用 QEMU 预先分配数据卷模式来提高写入性能。您可以使用预分配模式导入和上传操作，并在创建空白数据卷时使用。

如果启用了预分配，CDI 根据底层文件系统和设备类型使用更好的预分配方法：

fallocate

如果文件系统支持它，CDI 通过使用 **posix_fallocate** 功能（它分配块并将其标记为未初始化），来使用操作系统本身的（**fallocate** 调用来预分配空间。

full

如果无法使用 **fallocate** 模式，则会使用 **full** 模式通过将数据写入底层存储来为镜像分配空间。根据存储位置，所有空分配的空间都可能会为零。

8.18.7.2. 为数据卷启用预分配

您可以通过在数据卷清单中包含 **spec.preallocation** 字段来为特定数据卷启用预分配。您可以在 web 控制台中或使用 OpenShift CLI (**oc**) 启用预分配模式。

所有 CDI 源类型都支持 Preallocation 模式。

流程

- 指定数据卷清单中的 **spec.preallocation** 字段：

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: preallocated-datavolume
spec:
  source: 1
    ...
  pvc:
    ...
  preallocation: true 2
```

1. 所有 CDI 源类型支持预分配，但在克隆操作中会忽略预分配。
2. **preallocation** 字段是一个布尔值，默认值为 false。

8.18.8. 使用 Web 控制台上传本地磁盘镜像

您可以使用 web 控制台上传本地存储的磁盘镜像文件。

8.18.8.1. 先决条件

- 您必须有 IMG、ISO 或 QCOW2 格式的虚拟机镜像文件。
- 如果您根据 CDI 支持的操作列表要求涂销空间，您必须首先定义一个 StorageClass 或准备 CDI 涂销空间才能成功完成此操作。

8.18.8.2. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

□ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.18.8.3. 使用 Web 控制台上传镜像文件

使用 Web 控制台将镜像文件上传到新持久性卷声明（PVC）。之后，您可以使用此 PVC 将镜像附加到新虚拟机。

先决条件

- 您必须有以下之一：
  - 原始虚拟机镜像文件，可以是 ISO 或 IMG 格式。
  - 虚拟机镜像文件（QCOW2 格式）。
- 要获得最佳结果，先根据以下方法压缩您的镜像文件：
  - 使用 **xz** 或 **gzip** 压缩原始映像文件。

注意

使用压缩的原始镜像文件的上传效果最佳。

  - 使用为您的客户端推荐的方法压缩 QCOW2 镜像文件：
    - 如果使用 Linux 客户端，使用 virt-sparsify 工具对 QCOW2 文件进行 *sparsify*。
    - 如果您使用 Windows 客户端。使用 **xz** 或者 **gzip** 压缩 QCOW2 文件。

流程

1. 在 web 控制台的侧边菜单中点击 **Storage → Persistent Volume Claims**。
2. 点 **Create Persistent Volume Claim** 下拉列表展开它。
3. 点 **With Data Upload Form** 打开 **Upload Data to Persistent Volume Claim** 页面。
4. 点 **Browse** 打开文件管理器并选择要上传的镜像，或者将文件拖到 **Drag a file here or browse to upload** 项中。
5. 可选：将此镜像设置为特定操作系统的默认镜像。
   a. 选择 **Attach this data to a virtual machine operating system** 复选框。
   b. 从列表中选择一个操作系统。
6. **Persistent Volume Claim Name** 字段自动填充唯一名称，且无法编辑。记录分配给 PVC 的名称，以便以后根据需要指定它。
7. 从 **Storage Class** 列表中选择存储类。
8. 在 **Size** 字段中输入 PVC 的大小值。从下拉列表中选择对应的度量单位。

警告

PVC 大小必须大于未压缩的虚拟磁盘的大小。

9. 选择与您选择的存储类匹配的 **Access Mode**。
10. 点 **Upload**。

8.18.8.4. 其他资源

- 配置预分配模式以提高数据卷操作的写入性能。

8.18.9. 使用 virtctl 工具上传本地磁盘镜像

您可使用 **virtctl** 命令行实用程序将本地存储的磁盘镜像上传到新的或已有的数据卷中。

8.18.9.1. 先决条件

- 安装 **kubevirt-virtctl** 软件包。
- 如果您根据 CDI 支持的操作列表要求涂销空间，您必须首先定义一个 StorageClass 或准备 CDI 涂销空间才能成功完成此操作。

8.18.9.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.18.9.3. 创建上传数据卷

您可以使用 **upload** 数据源手动创建数据源，用于上传本地磁盘镜像。

流程

1. 创建指定 **spec: source: upload{}** 的数据卷配置：

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <upload-datavolume> 1
spec:
  source:
      upload: {}
  pvc:
    accessModes:
      - ReadWriteOnce
```

```
  resources:
    requests:
      storage: <2Gi> 2
```

1 数据卷的名称。

2 数据卷的大小。请确定这个值大于或等于您要上传的磁盘的大小。

2. 运行以下命令来创建数据卷：

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

8.18.9.4. 上传本地磁盘镜像至数据卷

您可使用 **virtctl** CLI 实用程序将客户端机器中的本地磁盘镜像上传到集群中的数据卷（DV）。您可以使用集群中已存在的 DV，也可以在此过程中创建新的 DV。

注意

上传本地磁盘镜像后，您可将其添加到虚拟机中。

先决条件

- 您必须有以下之一：
  - 原始虚拟机镜像文件，可以是 ISO 或 IMG 格式。
  - 虚拟机镜像文件（QCOW2 格式）。
- 要获得最佳结果，先根据以下方法压缩您的镜像文件：
  - 使用 **xz** 或 **gzip** 压缩原始映像文件。

    注意

    使用压缩的原始镜像文件的上传效果最佳。

  - 使用为您的客户端推荐的方法压缩 QCOW2 镜像文件：
    - 如果使用 Linux 客户端，使用 virt-sparsify 工具对 QCOW2 文件进行 *sparsify*。
    - 如果您使用 Windows 客户端。使用 **xz** 或者 **gzip** 压缩 QCOW2 文件。
- **kubevirt-virtctl** 软件包必须安装在客户端机器上。
- 客户端机器必须配置为信任 OpenShift Container Platform 路由器的证书。

流程

1. 确定以下各项：
   - 要使用的上传数据卷的名称。如果这个数据卷不存在，则会自动创建。
   - 在上传过程中创建数据卷的大小。大小必须大于或等于磁盘镜像的大小。

- 要上传的虚拟机磁盘镜像的文件位置。

2. 运行 **virtctl image-upload** 命令上传磁盘镜像。指定您在上一步中获得的参数。例如：

```
$ virtctl image-upload dv <datavolume_name> \ 1
--size=<datavolume_size> \ 2
--image-path=</path/to/image> \ 3
```

1. 数据卷的名称。
2. 数据卷的大小。例如： **--size=500Mi**, **--size=1G**
3. 虚拟机磁盘镜像的文件路径。

注意

- 如果您不想创建新数据卷，请省略 **--size** 参数，并包含 **--no-create** 标志。
- 将磁盘镜像上传到 PVC 时，PVC 大小必须大于未压缩的虚拟磁盘的大小。
- 若要在使用 HTTPS 时允许不安全的服务器连接，请使用 **--insecure** 参数。注意，在使用 **--insecure** 标志时，**不会**验证上传端点的真实性。

3. 可选。要验证数据卷是否已创建，运行以下命令来查看所有数据卷：

```
$ oc get dvs
```

8.18.9.5. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* □ GZ □ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* □ GZ □ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

□ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.18.9.6. 其他资源

- 配置预分配模式以提高数据卷操作的写入性能。

8.18.10. 上传本地磁盘镜像至块存储数据卷

您可以使用 **virtctl** 命令行实用程序将本地磁盘镜像上传到块数据卷中。

在此工作流中，您会创建一个本地块设备用作 PV，将此块卷与 **upload** 数据卷关联，并使用 **virtctl** 将本地磁盘镜像上传至数据卷中。

8.18.10.1. 先决条件

- 安装 **kubevirt-virtctl** 软件包。
- 如果您根据 CDI 支持的操作列表要求涂销空间，您必须首先定义一个 StorageClass 或准备 CDI 涂销空间才能成功完成此操作。

8.18.10.2. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.18.10.3. 关于块持久性卷

块持久性卷（PV）是一个受原始块设备支持的 PV。这些卷没有文件系统，可以通过降低开销来为虚拟机提供性能优势。

原始块卷可以通过在 PV 和持久性卷声明（PVC）规格中指定 **volumeMode: Block** 来置备。

8.18.10.4. 创建本地块持久性卷

通过填充文件并将其挂载为循环设备，在节点上创建本地块持久性卷（PV）。然后，您可以在 PV 清单中将该循环设备作为 **Block（块）**卷引用，并将其用作虚拟机镜像的块设备。

流程

1. 以 **root** 身份登录节点，在其上创建本地 PV。本流程以 **node01** 为例。
2. 创建一个文件并用空字符填充，以便可将其用作块设备。以下示例创建 **loop10** 文件，大小为 2Gb（20,100 Mb 块）：

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

3. 将 **loop10** 文件挂载为 loop 设备。

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

1. 挂载 loop 设备的文件路径。
2. 上一步中创建的文件，挂载为 loop 设备。

4. 创建引用所挂载 loop 设备的 **PersistentVolume** 清单。

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <local-block-pv10>
  annotations:
spec:
  local:
    path: </dev/loop10> (1)
  capacity:
    storage: <2Gi>
  volumeMode: Block (2)
  storageClassName: local (3)
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - <node01> (4)
```

(1) 节点上的 loop 设备路径。

(2) 将其指定为块 PV。

(3) 可选：为 PV 设置存储类。如果省略此项，将使用默认集群。

(4) 挂载块设备的节点。

5. 创建块 PV。

```
# oc create -f <local-block-pv10.yaml> (1)
```

(1) 上一步中创建的持久性卷的文件名。

8.18.10.5. 创建上传数据卷

您可以使用 **upload** 数据源手动创建数据源，用于上传本地磁盘镜像。

流程

1. 创建指定 **spec: source: upload{}** 的数据卷配置：

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <upload-datavolume> (1)
spec:
  source:
```

```
    upload: {}
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 2
```

1 数据卷的名称。

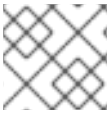
2 数据卷的大小。请确定这个值大于或等于您要上传的磁盘的大小。

2. 运行以下命令来创建数据卷：

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

8.18.10.6. 上传本地磁盘镜像至数据卷

您可使用 **virtctl** CLI 实用程序将客户端机器中的本地磁盘镜像上传到集群中的数据卷（DV）。您可以使用集群中已存在的 DV，也可以在此过程中创建新的 DV。



注意

上传本地磁盘镜像后，您可将其添加到虚拟机中。

先决条件

- 您必须有以下之一：
  - 原始虚拟机镜像文件，可以是 ISO 或 IMG 格式。
  - 虚拟机镜像文件（QCOW2 格式）。
- 要获得最佳结果，先根据以下方法压缩您的镜像文件：
  - 使用 **xz** 或 **gzip** 压缩原始映像文件。

    注意

    使用压缩的原始镜像文件的上传效果最佳。

  - 使用为您的客户端推荐的方法压缩 QCOW2 镜像文件：
    - 如果使用 Linux 客户端，使用 virt-sparsify 工具对 QCOW2 文件进行 *sparsify*。
    - 如果您使用 Windows 客户端。使用 **xz** 或者 **gzip** 压缩 QCOW2 文件。
- **kubevirt-virtctl** 软件包必须安装在客户端机器上。
- 客户端机器必须配置为信任 OpenShift Container Platform 路由器的证书。

流程

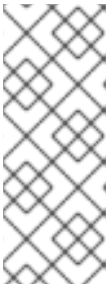
1. 确定以下各项：

- 要使用的上传数据卷的名称。如果这个数据卷不存在，则会自动创建。
- 在上传过程中创建数据卷的大小。大小必须大于或等于磁盘镜像的大小。
- 要上传的虚拟机磁盘镜像的文件位置。

2. 运行 **virtctl image-upload** 命令上传磁盘镜像。指定您在上一步中获得的参数。例如：

```
$ virtctl image-upload dv <datavolume_name> \ 1
--size=<datavolume_size> \ 2
--image-path=</path/to/image> \ 3
```

1. 数据卷的名称。
2. 数据卷的大小。例如： **--size=500Mi**, **--size=1G**
3. 虚拟机磁盘镜像的文件路径。



注意

- 如果您不想创建新数据卷，请省略 **--size** 参数，并包含 **--no-create** 标志。
- 将磁盘镜像上传到 PVC 时，PVC 大小必须大于未压缩的虚拟磁盘的大小。
- 若要在使用 HTTPS 时允许不安全的服务器连接，请使用 **--insecure** 参数。注意，在使用 **--insecure** 标志时，**不会**验证上传端点的真实性。

3. 可选。要验证数据卷是否已创建，运行以下命令来查看所有数据卷：

```
$ oc get dvs
```

8.18.10.7. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

☐ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.18.10.8. 其他资源

- 配置预分配模式以提高数据卷操作的写入性能。

8.18.11. 管理离线虚拟机快照

您可以为关闭（离线）的虚拟机创建、恢复和删除虚拟机快照。OpenShift Virtualization 支持以下离线虚拟机快照：

- OpenShift Container Storage
- 使用支持 Kubernetes 卷快照 API 的 Container Storage Interface（CSI）驱动程序的任何其他存储供应商

8.18.11.1. 关于虚拟机快照

*快照*代表虚拟机（VM）在特定时间点的状态和数据。您可以使用快照将现有虚拟机恢复到以前的状态（由快照代表）进行备份和恢复，或者快速回滚到以前的开发版本。

从关闭（停止状态）的虚拟机创建离线虚拟机快照。快照存储附加到虚拟机的每个 Container Storage Interface（CSI）卷的副本以及虚拟机规格和元数据的副本。创建后无法更改快照。

通过离线虚拟机快照功能，集群管理员和应用程序开发人员可以：

- 创建新快照
- 列出附加到特定虚拟机的所有快照
- 从快照恢复虚拟机
- 删除现有虚拟机快照

8.18.11.1.1. 虚拟机快照控制器和自定义资源定义（CRD）

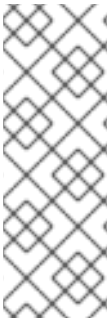
VM 快照功能引入了三个新的 API 对象，定义为 CRD，用于管理快照：

- **VirtualMachineSnapshot**:代表创建快照的用户请求。它包含有关虚拟机当前状态的信息。
- **VirtualMachineSnapshotContent**:代表集群中置备的资源（快照）。它由虚拟机快照控制器创建，其中包含恢复虚拟机所需的所有资源的引用。
- **VirtualMachineRestore**:代表从快照中恢复虚拟机的用户请求。

VM 快照控制器会把一个 **VirtualMachineSnapshotContent** 对象与创建它的 **VirtualMachineSnapshotContent** 对象绑定，并具有一对一的映射。

8.18.11.2. 在 web 控制台中创建离线虚拟机快照

您可以使用 web 控制台创建虚拟机（VM）快照。



注意

VM 快照只包含满足以下要求的磁盘：

- 必须是数据卷或持久性卷声明
- 属于支持容器存储接口（CSI）卷快照的存储类

如果您的虚拟机存储包含不支持快照的磁盘，您可以编辑它们或联系集群管理员。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 如果虚拟机正在运行，请点 **Actions → Stop Virtual Machine** 关闭它。
5. 点 **Snapshots** 标签页，然后点 **Take Snapshot**。
6. 填写 **Snapshot Name** 和可选的 **Description** 字段。
7. 扩展 **Disks included in this Snapshot** 以查看快照中包含的存储卷。
8. 如果您的虚拟机磁盘无法包含在快照中，并且您仍然希望继续，请选择 **I am aware of this warning and wish to proceed** 复选框。
9. 点 **Save**。

8.18.11.3. 通过 CLI 创建离线虚拟机快照

您可以通过创建一个 **VirtualMachineSnapshot** 对象来为离线虚拟机创建虚拟机快照。

先决条件

- 确保持久性卷声明（PVC）位于支持 Container Storage Interface（CSI）卷快照的存储类中。
- 安装 OpenShift CLI（**oc**）。
- 关闭您要为其创建快照的虚拟机。

流程

1. 创建一个 YAML 文件来定义 **VirtualMachineSnapshot** 对象，以指定新 **VirtualMachineSnapshot** 的名称和源虚拟机的名称。
例如：

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineSnapshot
metadata:
  name: my-vmsnapshot 1
spec:
  source:
```

```
  apiGroup: kubevirt.io
  kind: VirtualMachine
  name: my-vm 2
```

1 新的 **VirtualMachineSnapshot** 对象的名称。

2 源虚拟机的名称。

2. 创建 **VirtualMachineSnapshot** 资源。快照控制器会创建一个 **VirtualMachineSnapshotContent** 对象，将其绑定到 **VirtualMachineSnapshot** 并更新 **VirtualMachineSnapshot** 对象的 **status** 和 **readyToUse** 字段。

```
$ oc create -f <my-vmsnapshot>.yaml
```

验证

1. 验证 **VirtualMachineSnapshot** 对象是否已创建并绑定到 **VirtualMachineSnapshotContent**。**readyToUse** 标志必须设为 **true**。

```
$ oc describe vmsnapshot <my-vmsnapshot>
```

输**出示例**

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineSnapshot
metadata:
creationTimestamp: "2020-09-30T14:41:51Z"
finalizers:
- snapshot.kubevirt.io/vmsnapshot-protection
generation: 5
name: mysnap
namespace: default
resourceVersion: "3897"
selfLink:
/apis/snapshot.kubevirt.io/v1alpha1/namespaces/default/virtualmachinesnapshots/my-
vmsnapshot
uid: 28eedf08-5d6a-42c1-969c-2eda58e2a78d
spec:
source:
apiGroup: kubevirt.io
kind: VirtualMachine
name: my-vm
status:
conditions:
  - lastProbeTime: null
  lastTransitionTime: "2020-09-30T14:42:03Z"
  reason: Operation complete
  status: "False" 1
  type: Progressing
  - lastProbeTime: null
  lastTransitionTime: "2020-09-30T14:42:03Z"
  reason: Operation complete
  status: "True" 2
```

```
  type: Ready
creationTime: "2020-09-30T14:42:03Z"
readyToUse: true 3
sourceUID: 355897f3-73a0-4ec4-83d3-3c2df9486f4f
virtualMachineSnapshotContentName: vmsnapshot-content-28eedf08-5d6a-42c1-969c-
2eda58e2a78d 4
```

1. **Progressing** 的 **status** 字段指定快照是否仍然在创建。
2. **Ready** 条件的 **status** 字段指定快照创建过程是否完成。
3. 指定快照是否准备就绪可用被使用。
4. 指定快照被绑定到快照控制器创建的 **VirtualMachineSnapshotContent** 对象。

2. 检查 **VirtualMachineSnapshotContent** 资源的 **spec:volumeBackups** 属性，以验证快照中包含了预期的 PVC。

8.18.11.4. 在 web 控制台中从快照中恢复虚拟机

您可以将虚拟机（VM）恢复到 web 控制台中由快照表示的以前的配置。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 如果虚拟机正在运行，请点 **Actions → Stop Virtual Machine** 关闭它。
5. 点 **Snapshots** 标签页。该页面显示与虚拟机关联的快照列表。
6. 选择以下方法之一恢复虚拟机快照：
   a. 对于您要用作恢复虚拟机的源的快照，点 **Restore**。
   b. 选择快照以打开 **Snapshot Details** 屏幕，然后点 **Actions → Restore Virtual Machine Snapshot**。
7. 在确认弹出窗口中，点 **Restore** 可将虚拟机恢复到快照代表的以前的配置中。

8.18.11.5. 通过 CLI 从快照中恢复虚拟机

您可以使用虚拟机快照将现有虚拟机（VM）恢复到以前的配置。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 关闭您要恢复到之前状态的虚拟机。

流程

1. 创建一个 YAML 文件来定义 **VirtualMachineRestore** 对象，它指定您要恢复的虚拟机的名称以及要用作源的快照名称。
例如：

```
apiVersion: snapshot.kubevirt.io/v1alpha1
kind: VirtualMachineRestore
metadata:
  name: my-vmrestore 1
spec:
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: my-vm 2
  virtualMachineSnapshotName: my-vmsnapshot 3
```

1. 新 **VirtualMachineRestore** 对象的名称。
2. 要恢复的目标虚拟机的名称。
3. 作为源的 **VirtualMachineSnapshot** 对象的名称。

2. 创建 **VirtualMachineRestore** 资源。快照控制器更新了 **VirtualMachineRestore** 对象的 status 字段，并将现有虚拟机配置替换为快照内容。

```
$ oc create -f <my-vmrestore>.yaml
```

验证

- 验证虚拟机是否已恢复到快照代表的以前的状态。**complete** 标志需要被设置为 **true**。

```
$ oc get vmrestore <my-vmrestore>
```

输出示例

```
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/v1
  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 (1)
  conditions:
  - lastProbeTime: null
  lastTransitionTime: "2020-09-30T14:46:28Z"
  reason: Operation complete
  status: "False" (2)
  type: Progressing
  - lastProbeTime: null
  lastTransitionTime: "2020-09-30T14:46:28Z"
  reason: Operation complete
  status: "True" (3)
  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. 指定将虚拟机恢复到快照代表的状态的进程是否已完成。
2. **Progressing** 条件的 **status** 字段指定 VM 是否仍然被恢复。
3. **Ready** 条件的 **status** 字段指定 VM 恢复过程是否完成。

8.18.11.6. 删除 web 控制台中的虚拟机快照

您可以使用 web 控制台删除现有虚拟机快照。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点 **Snapshots** 标签页。该页面显示与虚拟机关联的快照列表。
5. 选择以下方法之一删除虚拟机快照：

a. 点您要删除的虚拟机快照  的 Options 菜单，然后选择 **Delete Virtual Machine Snapshot**。

b. 选择快照以打开 **Snapshot Details** 屏幕，然后点 **Actions → Delete Virtual Machine Snapshot**。

6. 在确认弹出窗口中点 **Delete** 删除快照。

8.18.11.7. 通过 CLI 删除虚拟机快照

您可以通过删除正确的 **VirtualMachineSnapshot** 对象来删除现有虚拟机（VM）快照。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

- 删除 **VirtualMachineSnapshot** 对象。快照控制器会删除 **VirtualMachineSnapshot** 和关联的 **VirtualMachineSnapshotContent** 对象。

```
$ oc delete vmsnapshot <my-vmsnapshot>
```

验证

- 验证快照是否已删除，且不再附加到此虚拟机：

```
$ oc get vmsnapshot
```

8.18.11.8. 其他资源

- CSI 卷快照

8.18.12. 将本地虚拟机磁盘移动到不同的节点中

使用本地卷存储的虚拟机可被移动，以便在特定节点中运行。

因为以下原因，您可能想要将该虚拟机移动到特定的节点上：

- 当前节点对本地存储配置有限制。
- 新节点对那个虚拟机的工作负载进行了更好的优化。

要移动使用本地存储的虚拟机，您必须使用数据卷克隆基础卷。克隆操作完成后，您可以 编辑虚拟机配置，以便使用新数据卷，或 将新数据卷添加到其他虚拟机。

提示

当您全局启用预分配或单个数据卷时，Containerized Data Importer（CDI）会在克隆过程中预分配磁盘空间。预分配可提高写入性能。如需更多信息，请参阅对数据卷使用预分配。

注意

没有 **cluster-admin** 角色的用户需要 额外的用户权限 才能在命名空间间克隆卷。

8.18.12.1. 克隆本地卷到另一个节点

您可以通过克隆底层 PVC，移动虚拟机磁盘使其在特定节点上运行。

要确保虚拟机磁盘克隆到正确的节点，您必须创建新的持久性卷（PV）或在正确的节点上识别它。对 PV 应用一个唯一标签，以便数据卷可以引用它。

注意

目标 PV 的大小不得小于源 PVC。如果目标 PV 小于源 PVC，克隆操作会失败。

先决条件

- 虚拟机不能正在运行。在克隆虚拟机磁盘前关闭虚拟机。

流程

1. 在节点上创建新本地 PV，或使用已有的本地 PV:
   - 创建包含 **nodeAffinity.nodeSelectorTerms** 参数的本地 PV。以下 manifest 在 **node01** 上创建了一个 **10Gi** 的本地 PV。

```
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. PV 的名称。
2. PV 的大小。您必须分配足够空间，否则克隆操作会失败。其大小不得低于源 PVC。
3. 节点上的挂载路径。

4 要创建 PV 的节点的名称。

- 已存在于节点上的一个 PV。您可以通过查看其配置中的 **nodeAffinity** 字段来标识置备 PV 的节点：

```
$ oc get pv <destination-pv> -o yaml
```

以下输出显示 PV 位于 **node01**：

输出示例

```
...
spec:
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname 1
          operator: In
          values:
          - node01 2
...
```

1 **Kubernetes.io/hostname** 使用节点主机名来选择节点。

2 节点的主机名。

2. 为 PV 添加唯一标签：

```
$ oc label pv <destination-pv> node=node01
```

3. 创建引用以下内容的数据卷清单：

- 虚拟机的 PVC 名称和命名空间。
- 应用上一步中的 PV 标签。
- 目标 PV 的大小。

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <clone-datavolume> 1
spec:
  source:
    pvc:
      name: "<source-vm-disk>" 2
      namespace: "<source-namespace>" 3
  pvc:
    accessModes:
      - ReadWriteOnce
    selector:
      matchLabels:
```

```
    node: node01 4
  resources:
    requests:
      storage: <10Gi> 5
```

1. 新数据卷的名称。
2. 源 PVC 的名称。如果您不知道 PVC 名称，您可以在虚拟机配置中找到它：**spec.volumes.persistentVolumeClaim.claimName**。
3. 源 PVC 所在的命名空间。
4. 应用于上一步中 PV 的标签。
5. 目标 PV 的大小。

4. 通过将数据卷清单应用到集群来开始克隆操作：

```
$ oc apply -f <clone-datavolume.yaml>
```

数据卷将虚拟机的 PVC 克隆到特定节点上的 PV 中。

8.18.13. 通过添加空白磁盘镜像扩展虚拟存储

您可向 OpenShift Virtualization 添加空白磁盘镜像来提高存储容量或创建新数据分区。

8.18.13.1. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.18.13.2. 使用数据卷创建空白磁盘镜像

您可以通过自定义和部署数据卷配置文件在持久性卷声明中创建新空白磁盘镜像。

先决条件

- 至少一个可用持久性卷。
- 安装 OpenShift CLI（**oc**）。

流程

1. 编辑数据卷配置文件：

```
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
    resources:
      requests:
        storage: 500Mi
```

2. 运行以下命令，创建空白磁盘镜像：

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

8.18.13.3. 模板：空白磁盘镜像的数据卷配置文件

blank-image-datavolume.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
    resources:
      requests:
        storage: 500Mi
```

8.18.13.4. 其他资源

- 配置预分配模式以提高数据卷操作的写入性能。

8.18.14. 使用 smart-cloning（智能克隆） 克隆数据卷

Smart-cloning（智能克隆）是 OpenShift Container Platform Storage（OCS）的内置功能，旨在提高克隆过程的性能。使用智能克隆进行克隆比主机辅助克隆更快、效率更高。

您不需要执行任何操作来启用智能克隆功能，但需要确保您的存储环境与智能克隆兼容。

使用持久性卷声明（PVC）源创建数据卷时，会自动启动克隆过程。如果您的环境支持智能克隆，则始终会收到数据卷的克隆。但是，只有存储供应商支持智能克隆时，才会获得智能克隆的性能优势。

8.18.14.1. 了解智能克隆

当一个数据卷被智能克隆时，会出现以下情况：

1. 创建源持久性卷声明（PVC）的快照。

2. 从快照创建一个 PVC。

3. 快照被删除。

8.18.14.2. 克隆数据卷

先决条件

要实现智能克隆，需要满足以下条件：

- 您的存储供应商必须支持快照。
- 源和目标 PVC 必须定义为相同的存储类。
- **VolumeSnapshotClass** 对象必须引用定义为源和目标 PVC 的存储类。

流程

启动数据卷克隆：

1. 为 **DataVolume** 对象创建一个 YAML 文件，用于指定新数据卷的名称以及源 PVC 的名称和命名空间。在这个示例中，因为指定了 **storage** API，因此不需要指定 **accessModes** 或 **volumeMode**。将自动为您计算最佳值。

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  storage: 4
    resources:
      requests:
        storage: <2Gi> 5
```

1. 新数据卷的名称。
2. 源 PVC 所在的命名空间。
3. 源 PVC 的名称。
4. 使用 **storage** API 指定分配
5. 新数据卷的大小。

2. 通过创建数据卷开始克隆 PVC:

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

注意

在 PVC 就绪前，DataVolume 会阻止虚拟机启动，以便您可以在 PVC 克隆期间创建引用新数据卷的虚拟机。

8.18.14.3. 其他资源

- 将虚拟机磁盘的持久性卷声明克隆到新数据卷中
- 配置预分配模式以提高数据卷操作的写入性能。

8.18.15. 创建并使用引导源

引导源包含可引导操作系统(OS)以及操作系统的所有配置设置，如驱动程序。

您可以使用引导源创建带有特定配置的虚拟机模板。这些模板可用于创建任意数量的可用虚拟机。

OpenShift Container Platform Web 控制台提供了快速入门导览，可帮助您创建自定义引导源、上传引导源和其他任务。从 **Help** 菜单中选择 **Quick Starts** 以查看快速入门。

8.18.15.1. 关于虚拟机和引导源

虚拟机由虚拟机定义以及由数据卷支持的一个或多个磁盘组成。虚拟机模板允许您使用预定义的虚拟机规格创建虚拟机。

每个虚拟机模板都需要一个引导源，它是一个完全配置的虚拟机磁盘镜像，包括配置的驱动程序。每个虚拟机模板包含一个虚拟机定义，其中包含指向引导源的指针。每个引导源都有一个预定义的名称和命名空间。对于某些操作系统，会自动提供一个引导源。如果没有提供，管理员必须准备自定义引导源。

提供的引导源会自动更新至操作系统的最新版本。对于自动更新的引导源，持久性卷声明(PVC)使用集群的默认存储类创建。如果在配置后选择了不同的默认存储类，您必须删除使用之前的默认存储类配置的集群命名空间中的现有数据卷。

要使用引导源功能，请安装 OpenShift Virtualization 的最新版本。命名空间 **openshift-virtualization-os-images** 启用该功能，并安装 OpenShift Virtualization Operator。安装引导源功能后，您可以创建引导源，将它们附加到模板，并从模板创建虚拟机。

使用通过上传本地文件、克隆现有 PVC、从 registry 或 URL 导入的持久性卷声明（PVC）定义引导源。使用 web 控制台将引导源附加到虚拟机模板。在启动源附加到虚拟机模板后，您可从模板创建任意数量的已完全配置的可随时使用虚拟机。

8.18.15.2. 将 Red Hat Enterprise Linux 镜像导入为引导源

您可以通过指定镜像的 URL 地址，导入 Red Hat Enterprise Linux（RHEL）镜像作为引导源。

先决条件

- 您必须有权访问带有操作系统镜像的 Web 服务器。例如：带有镜像的 Red Hat Enterprise Linux 网页。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 找到您要为其配置引导源的 RHEL 模板并点 **Add source**。
4. 在 **Add boot source to template** 窗口中，从 **Boot source type** 列表中选择 **Import via URL(creates PVC)**。

5. 点击 **RHEL 下载页面** 访问红帽客户门户。下载 Red Hat Enterprise Linux 页面中显示可用安装程序和镜像的列表。

6. 确定您要下载的 Red Hat Enterprise Linux KVM 客户机镜像。右键单击 **Download Now**，再复制镜像的 URL。

7. 在 **Add boot source to template** 窗口中，将 guest 镜像的复制 URL 粘贴到 **Import URL** 字段中，然后单击 **Save and import**。

验证

验证引导源是否已添加到模板中：

1. 点 **Templates** 选项卡。

2. 确认此模板的标题显示绿色勾号。

现在，您可以使用此模板创建 RHEL 虚拟机。

8.18.15.3. 为虚拟机模板添加引导源

对于您要用于创建虚拟机或自定义模板的任何虚拟机模板，可以配置引导源。当使用引导源配置虚拟机模板时，会在 **Templates** 选项卡中被标记为 **Available**。在向模板中添加引导源后，您可以使用该模板创建新虚拟机。

在 web 控制台中选择和添加引导源有四个方法：

- **上传本地文件（创建 PVC）**
- **通过 URL 导入（创建 PVC）**
- **克隆现有的 PVC（创建 PVC）**
- **通过 Registry 导入（创建 PVC）**

先决条件

- 要添加引导源，您必须以具有 **os-images.kubevirt.io:edit** RBAC 角色或管理员的用户身份登录。您不需要特殊权限才能从附加了引导源的模板创建虚拟机。
- 要上传本地文件，操作系统镜像文件必须存在于本地机器中。
- 要通过 URL 导入，您需要访问带操作系统镜像的 web 服务器。例如：带有镜像的 Red Hat Enterprise Linux 网页。
- 要克隆现有的 PVC，需要使用 PVC 访问项目。
- 要通过 registry 导入，需要访问容器 registry。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。

2. 点 **Templates** 选项卡。

3. 找到您要配置引导源的虚拟机模板并点击 **Add source**。

4. 在 **Add boot source to template**窗口中，点 **Select boot source**，选择创建持久性卷声明（PVC）的方法： **Upload local file**、**Import via URL**、**Clone existing PVC** 或 **Import via Registry**。

5. 可选：点 **This is a CD-ROM boot source**来挂载 CD-ROM，并使用它将操作系统安装到空磁盘。OpenShift Virtualization 会自动创建并挂载额外的空磁盘。如果不需要额外磁盘，您可以在创建虚拟机时将其删除。

6. 为 **持久性卷声明大小** 输入一个值，以指定适合未压缩镜像的 PVC 大小，以及任何需要的额外空间。

   a. 可选：输入 **Source 供应商** 名称以将名称与此模板关联。

   b. 可选 ：**高级存储设置**：点 **Storage class** 并选择用于创建磁盘的存储类。通常，这个存储类是创建供所有 PVC 使用的默认存储类。

   注意

   提供的引导源会自动更新至操作系统的最新版本。对于自动更新的引导源，持久性卷声明(PVC)使用集群的默认存储类创建。如果在配置后选择了不同的默认存储类，您必须删除使用之前的默认存储类配置的集群命名空间中的现有数据卷。

   c. 可选：**高级存储设置**：点 **Access 模式**，并为持久性卷选择访问模式：

      - **单用户(RWO)** 将卷由单一节点以读写模式挂载。
      - **共享访问(RWX)** 将卷挂载为许多节点以读写模式。
      - **只读(ROX)** 将卷挂载为多个节点只读。

   d. 可选：**高级存储设置**：如果您需要选择 **Block** 而不是默认的值 **Filesystem**，点 **Volume mode**。OpenShift Virtualization 可以静态置备原始块卷。这些卷没有文件系统。对于可以直接写入磁盘或者实现其自己的存储服务的应用程序来说，使用它可以获得性能优势。

7. 选择保存引导源的适当方法：

   a. 如果您上传一个本地文件，请点击 **Save and upload**。

   b. 如果从 URL 或 registry 中导入内容，点 **Save and import**。

   c. 如果克隆现有的 PVC，点 **Save and clone**。

Templates 选项卡中列出了带有引导源的自定义虚拟机模板，您可以使用此模板创建虚拟机。

8.18.15.4. 从带有附加引导源的模板创建虚拟机

在向模板中添加引导源后，您可以使用该模板创建新虚拟机。

流程

1. 在 OpenShift Container Platform web 控制台中，在侧边单中点 **Workloads > Virtualization**。
2. 在 **Virtual Machines** 选项卡或 **Templates** 选项卡中点 **Create** 并选择 **Virtual Machine with Wizard**。

3. 在 **Select a template** 步骤中，从 Operating System 列表中选择一个操作系统，它具有 OS 和版本名称旁的 **(Source available)** 标签。**(Source available)** 标签表示这个操作系统可以使用引导源。

4. 点 **Review and Confirm**。

5. 检查您的虚拟机设置，并根据需要编辑它们。

6. 点 **Create Virtual Machine** 创建您的虚拟机。此时会显示 **Successfully created virtual machine** 页面。

8.18.15.5. 创建自定义引导源

您可以基于现有磁盘镜像准备自定义磁盘镜像，以用作引导源。

使用这个流程完成以下任务：

- 准备自定义磁盘镜像
- 从自定义磁盘镜像创建引导源
- 将引导源附加到自定义模板

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads** > **Virtualization**。

2. 点 **Templates** 选项卡。

3. 单击您要自定义的模板的 **Source provider** 列中的链接。此时将显示一个窗口，表明模板当前具有定义的源。

4. 在窗口中，单击 **Customize source** 链接。

5. 在阅读有关引导源自定义过程的信息后，单击 **About boot source Custom** 窗口中的 **Continue** 以继续自定义。

6. 在 **Prepare boot source Custom** 页面中的 **Define new template** 部分：

   a. 选择 **New template namespace** 字段，然后选择项目。

   b. 在 **New template name** 字段中输入自定义模板的名称。

   c. 在 **New template provider** 字段中输入模板提供程序名称。

   d. 选择 **New template support** 字段，然后选择适当的值，指示您创建的自定义模板的支持联系人。

   e. 选择 **New template flavor** 字段，然后选择您创建的自定义镜像的适当 CPU 和内存值。

7. 在 **Prepare boot source for custom** 部分中，根据需要自定义 **cloud-init** YAML 脚本来定义登录凭据。否则，脚本会为您生成默认凭据。

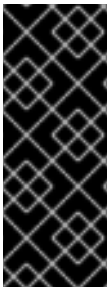
8. 单击 **Start Customization**。自定义过程开始并显示 **Preparing boot source Custom** 页面，然后显示 **Customize boot source** 页面。**Customize boot source** 页面显示正在运行的脚本的输出。脚本完成后，您的自定义镜像将可用。

9. 在 **VNC 控制台** 中，单击 **Guest login credentials** 部分中的 **show password**。您的登录凭据显示。
10. 镜像准备好登录时，通过提供 **guest 登录凭证**部分中显示的用户名和密码来登录 **VNC** 控制台。
11. 验证自定义镜像按预期工作。如果存在，点 **Make this boot source available**。
12. 在 **Finish custom and make template available** 窗口中，选择 **I have sealed the boot source**，以便它用作模板，然后单击 **Apply**。
13. 在 **Finishing boot source Custom** 页面上，等待模板创建过程完成。点 **Navigate to template details** 或 **Navigate to template list** 查看从自定义引导源创建的自定义模板。

8.18.15.6. 其他资源

- 创建虚拟机模板
- 从云镜像创建 Microsoft Windows 引导源
- 在 OpenShift Container Platform 中自定义现有 Microsoft Windows 引导源
- 使用 CLI 将 PVC 设置为 Microsoft Windows 模板的引导源
- 使用自动化脚本创建引导源
- 在 pod 中自动创建引导源

8.18.16. 热插拔虚拟磁盘



重要

热插拔虚拟磁盘只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

希望在不停止虚拟机或虚拟机实例的情况下添加或删除热插拔和热插拔虚拟磁盘。当您需要在不发生停机时间的情况下向正在运行的虚拟机添加存储时，此功能非常有用。

当您*热插拔*虚拟磁盘时，可以在虚拟机运行时将虚拟磁盘附加到虚拟机实例中。

当*热拔*虚拟磁盘时，您可以在虚拟机运行时从虚拟机实例分离虚拟磁盘。

只有数据卷和持久性卷声明（PVC）才能热插和热拔。您不能热插或热拔容器磁盘。

警告

将热插虚拟磁盘附加到虚拟机时，无法在虚拟机上执行实时迁移。如果您试图在附加有热插磁盘的虚拟机上执行实时迁移，实时迁移会失败。

8.18.16.1. 使用 CLI 热插虚拟磁盘

在虚拟机运行时热插要附加到虚拟机实例（VMI）的虚拟磁盘。

先决条件

- 您必须有一个正在运行的虚拟机才能热插虚拟磁盘。
- 您必须至少有一个数据卷或持久性卷声明（PVC）可用于热插。

流程

- 运行以下命令进行虚拟磁盘热插：

```
$ virtctl addvolume <virtual-machine|virtual-machine-instance> --volume-name=
<datavolume|PVC> \
[--persist] [--serial=<label-name>]
```

  - 使用可选 **--persist** 标志，将热插磁盘作为永久挂载的虚拟磁盘添加到虚拟机规格中。停止、重新启动或重新启动虚拟机以永久挂载虚拟磁盘。指定 **--persist** 标志后，您无法再热插或热拔虚拟磁盘。**--persist** 标志适用于虚拟机，而不是虚拟机接口。
  - 可选 **--serial** 标志允许您添加您选择的字母数字字符串标签。这有助于您识别客户机虚拟机中的热插磁盘。如果没有指定这个选项，则标签默认为热插数据卷或 PVC 的名称。

8.18.16.2. 使用 CLI 热拔虚拟磁盘

在虚拟机运行时，热拔您想要从虚拟机实例（VMI）断开的虚拟磁盘。

先决条件

- 您的虚拟机必须正在运行。
- 您必须至少有一个数据卷或持久性卷声明（PVC）可用并为热插。

流程

- 运行以下命令来热拔虚拟磁盘：

```
$ virtctl removevolume <virtual-machine|virtual-machine-instance> --volume-name=
<datavolume|PVC>
```

8.18.17. 将容器磁盘与虚拟机搭配使用

您可以将虚拟机镜像构建到容器磁盘中，并将其存储在容器 registry 中。然后，您可以将容器磁盘导入虚拟机的持久性存储中，或者将其直接附加到虚拟机临时存储。

重要

如果您使用大型容器磁盘，则 I/O 流量可能会增加，影响 worker 节点。这可能导致不可用的节点。您可以通过以下方法解决这个问题：

- 修剪 **DeploymentConfig** 对象
- 配置垃圾回收

8.18.17.1. 关于容器磁盘

容器磁盘是一个虚拟机镜像，它作为容器镜像存储在容器镜像 registry 中。您可以使用容器磁盘将同一磁盘镜像传送到多个虚拟机，并创建大量虚拟机克隆。

容器磁盘可以使用附加到虚拟机的数据卷导入到持久性卷声明（PVC），也可以作为临时 **containerDisk** 卷直接附加到虚拟机。

8.18.17.1.1. 使用数据卷将容器磁盘导入到 PVC 中

通过 Containerized Data Importer（CDI）使用数据卷将容器磁盘导入到 PVC 中。然后，您可以将数据卷附加到虚拟机以获取持久性存储。

8.18.17.1.2. 将容器磁盘作为 containerDisk 卷附加到虚拟机

containerDisk 卷是临时的。将在虚拟机停止、重启或删除时丢弃。当一个带有 **containerDisk** 卷的虚拟机启动时，容器镜像从 registry 中拉取，并托管在托管虚拟机的节点上。

将 **containerDisk** 卷用于只读文件系统，如 CD-ROM 或可处理的虚拟机。

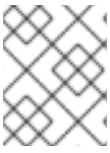
重要

不建议将 **containerDisk** 卷用于读写文件系统，因为数据是临时写入托管节点上的本地存储。这会减慢虚拟机的实时迁移速度，如节点维护，因为数据必须迁移到目标节点。另外，如果节点断电或者意外关闭，则所有数据都会丢失。

8.18.17.2. 为虚拟机准备容器磁盘

您必须使用虚拟机镜像构建容器磁盘，并将其推送到容器 registry，然后才能用于虚拟机。然后，您可以使用数据卷将容器磁盘导入到 PVC 中，并将其附加到虚拟机，或者将容器磁盘作为临时 **containerDisk** 卷直接附加到虚拟机。

容器磁盘中磁盘镜像的大小受托管容器磁盘的 registry 的最大层大小的限制。



注意

对于 Red Hat Quay，您可以通过编辑首次部署 Red Hat Quay 时创建的 YAML 配置文件来更改最大层大小。

先决条件

- 如果还没有安装，安装 **podman**。
- 虚拟机镜像必须是 QCOW2 或 RAW 格式。

流程

1. 创建一个 Dockerfile 以将虚拟机镜像构建到容器镜像中。虚拟机镜像必须属于 QEMU，其 UID 为 **107**，并放置在容器的 **/disk/** 目录中。**/disk/** 目录的权限必须设为 **0440**。
以下示例在第一阶段使用 Red Hat Universal Base Image（UBI）来处理这些配置更改，并使用第二阶段中的最小 **scratch** 镜像存储结果：

```
$ 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 其中，<vm_image> 是 QCOW2 或 RAW 格式的虚拟机镜像。
要使用远程虚拟机镜像，将 **<vm_image>.qcow2** 替换为远程镜像的完整 url。

2. 构建和标记容器：

```
$ podman build -t <registry>/<container_disk_name>:latest .
```

3. 将容器镜像推送到 registry:

```
$ podman push <registry>/<container_disk_name>:latest
```

如果容器镜像敞开没有 TLS，您必须将其添加为一个不安全的容器镜像仓库，然后才能将容器磁盘导入持久性存储。

8.18.17.3. 禁用容器镜像仓库的 TLS，以用作不安全的容器镜像仓库

您可以通过编辑 **HyperConverged** 自定义资源的 **insecureRegistries** 字段来禁用一个或多个容器 registry 的 TLS（传输层安全）。

先决条件

- 以具有 **cluster-admin** 角色的用户身份登录集群。

流程

- 编辑 **HyperConverged** 自定义资源，将不安全 registry 列表添加到 **spec.storageImport.insecureRegistries** 字段中。

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  storageImport:
    insecureRegistries: 1
      - "private-registry-example-1:5000"
      - "private-registry-example-2:5000"
```

1 将此列表中的示例替换为有效的 registry 主机名。

8.18.17.4. 后续步骤

- 将容器磁盘导入虚拟机的持久性存储中。
- 创建一个虚拟机，它使用一个 **containerDisk** 卷作为临时存储。

8.18.18. 准备 CDI 涂销空间

8.18.18.1. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.18.18.2. 了解涂销空间

Containerized Data Importer (CDI) 需要涂销空间（临时存储）来完成一些操作，如导入和上传虚拟机镜像。在此过程中，CDI 会提供一个与支持目标数据卷（DV）的 PVC 大小相等的涂销空间 PVC。该涂销空间 PVC 将在操作完成或中止后删除。

您可以在 **HyperConverged** 自定义资源的 **spec.scratchSpaceStorageClass** 字段中定义绑定涂销空间 PVC 的存储类。

如果定义的存储类与集群中的存储类不匹配，则会使用为集群定义的默认存储类。如果没有在集群中定义默认存储类，则会使用置备原始 DV 或 PVC 的存储类。

注意

CDI 需要通过 **file** 卷模式来请求涂销空间，与支持原始数据卷的 PVC 无关。如果 **block** 卷模式支持原始 PVC，则您必须定义一个能够置备 **file** 卷模式 PVC 的 StorageClass。

手动调配
如果没有存储类，CDI 将使用项目中与镜像的大小要求匹配的任何 PVC。如果没有与这些要求匹配的 PVC，则 CDI 导入 Pod 将保持 **Pending** 状态，直至有适当的 PVC 可用或直至超时功能关闭 Pod。

8.18.18.3. 需要涂销空间的 CDI 操作

类型	原因
registry 导入	CDI 必须下载镜像至涂销空间，并对层进行提取，以查找镜像文件。然后镜像文件传递至 QEMU-IMG 以转换成原始磁盘。
上传镜像	QEMU-IMG 不接受来自 STDIN 的输入。相反，要上传的镜像保存到涂销空间中，然后才可传递至 QEMU-IMG 进行转换。

类型	原因
存档镜像的 HTTP 导入	QEMU-IMG 不知道如何处理 CDI 支持的存档格式。相反，镜像取消存档并保存到涂销空间中，然后再传递至 QEMU-IMG。
经过身份验证的镜像的 HTTP 导入	QEMU-IMG 未充分处理身份验证。相反，镜像保存到涂销空间中并进行身份验证，然后再传递至 QEMU-IMG。
自定义证书的 HTTP 导入	QEMU-IMG 未充分处理 HTTPS 端点的自定义证书。相反，CDI 下载镜像到涂销空间，然后再将文件传递至 QEMU-IMG。

8.18.18.4. 定义存储类

您可以通过将 **spec.scratchSpaceStorageClass** 字段添加到 **HyperConverged** 自定义资源（CR）来定义 Containerized Data Importer（CDI）在分配涂销空间时使用的存储类。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

1. 运行以下命令来编辑 **HyperConverged** CR：

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

2. 将 **spec.scratchSpaceStorageClass** 字段添加到 CR，将值设置为集群中存在的存储类的名称：

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
spec:
  scratchSpaceStorageClass: "<storage_class>" 1
```

1 如果您没有指定存储类，CDI 将使用正在填充的持久性卷声明的存储类。

3. 保存并退出默认编辑器以更新 **HyperConverged** CR。

8.18.18.5. CDI 支持的操作列表

此列表针对端点显示内容类型支持的 CDI 操作，以及哪些操作需要涂销空间（scratch space）。

内容类型	HTTP	HTTPS	HTTP 基本身份验证	Registry	上传
KubeVirt (QCOW2)	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2** ✓ GZ* ✓ XZ*	✓ QCOW2 ✓ GZ* ✓ XZ*	✓ QCOW2* ☐ GZ ☐ XZ	✓ QCOW2* ✓ GZ* ✓ XZ*
KubeVirt (RAW)	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW ✓ GZ ✓ XZ	✓ RAW* ☐ GZ ☐ XZ	✓ RAW* ✓ GZ* ✓ XZ*

✓ 支持的操作

☐ 不支持的操作

* 需要涂销空间

** 如果需要自定义证书颁发机构，则需要涂销空间

8.18.18.6. 其他资源

- 动态置备

8.18.19. 重新使用持久性卷

要重新使用静态置备的持久性卷（PV），,您必须首先重新声明该卷。这涉及删除 PV，以便重新使用存储配置。

8.18.19.1. 关于重新声明静态置备的持久性卷

当重新声明持久性卷（PV）时，您从持久性卷声明（PVC）中卸载 PV 并删除 PV。根据底层存储，您可能需要手动删除共享存储。

然后，您可以重新使用 PV 配置来创建具有不同名称的 PV。

静态置备的 PV 必须具有 **Retain** 的重新声明策略才能重新声明。如果没有，则当 PVC 取消和 PV 的绑定后，PV 将进入失败的状态。

重要

在 OpenShift Container Platform 4 中， **Recycle** 重新声明策略已被弃用。

8.18.19.2. 重新声明静态置备的持久性卷

通过取消绑定持久性卷声明（PVC）并删除 PV 重新声明静态置备的持久性卷（PV）。您可能还需要手动删除共享存储。

重新声明静态置备的 PV 依赖于底层存储。此流程提供一般方法，可能需要根据您的存储进行调整。

流程

1. 确保 PV 的 reclaim 策略被设置为 **Retain**:

a. 检查 PV 上的 reclaim 策略：

```
$ oc get pv <pv_name> -o yaml | grep 'persistentVolumeReclaimPolicy'
```

b. 如果 **persistentVolumeReclaimPolicy** 没有设置为 **Retain**，使用以下命令编辑 reclaim 策略：

```
$ oc patch pv <pv_name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
```

2. 确保没有资源在使用 PV:

```
$ oc describe pvc <pvc_name> | grep 'Mounted By:'
```

在继续操作前，删除所有使用 PVC 的资源。

3. 删除 PVC 以释放 PV:

```
$ oc delete pvc <pvc_name>
```

4. 可选：将 PV 配置导出到 YAML 文件。如果在稍后手动删除共享存储，您可以参考此配置。您还可以使用该文件中的 **spec** 参数作为基础，在重新声明 PV 后创建具有相同存储配置的新 PV:

```
$ oc get pv <pv_name> -o yaml > <file_name>.yaml
```

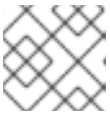
5. 删除 PV：

```
$ oc delete pv <pv_name>
```

6. 可选：根据存储类型，您可能需要删除共享存储文件夹的内容：

```
$ rm -rf <path_to_share_storage>
```

7. 可选：创建一个使用与删除 PV 相同的存储配置的 PV。如果您之前导出了重新声明的 PV 配置，您可以使用该文件的 **spec** 参数作为新 PV 清单的基础：



注意

为了避免可能的冲突，最好为新 PV 对象赋予与您删除的名称不同的名称。

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

其他资源

- 为虚拟机配置本地存储
- OpenShift Container Platform Storage 文档包含更多有关持久性存储的信息。

8.18.20. 删除数据卷

您可以使用 **oc** CLI 手动删除数据卷。

注意

当您删除虚拟机时，其使用的数据卷会被自动删除。

8.18.20.1. 关于数据卷

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层持久性卷声明（PVC）关联的导入、克隆和上传操作。数据卷与 OpenShift Virtualization 集成，它们可在 PVC 准备好前阻止虚拟机启动。

8.18.20.2. 列出所有数据卷

您可以使用 **oc** CLI 列出集群中的数据卷。

流程

- 运行以下命令列出所有数据卷：

```
$ oc get dvs
```

8.18.20.3. 删除数据卷

您可以使用 **oc** CLI 删除数据卷。

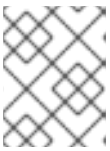
先决条件

- 找出您要删除的数据卷的名称。

流程

- 运行以下命令来删除数据卷：

```
$ oc delete dv <datavolume_name>
```



注意

此命令只删除当前项目中存在的对象。如果您要删除其他项目或命名空间中的对象，请使用 **-n <project_name>** 选项。

第 9 章 虚拟机模板

9.1. 创建虚拟机模板

9.1.1. 关于虚拟机模板

Virtualization 页面中的 **Templates** 选项卡中列出了预先配置的红帽虚拟机模板。这些模板适用于 Red Hat Enterprise Linux、Fedora、微软 Windows 10 和 Microsoft Windows Servers 的不同版本。每个红帽虚拟机模板都预先配置了操作系统镜像、操作系统的默认设置、类型（CPU 和内存）以及工作负载类型(server)。

Templates 标签显示四个类型的虚拟机模板：

- 红**帽支持的**模板完全被红帽模板。
- **用户支持的**模板 红**帽支持的** 模板，但由用户克隆并创建。
- 红**帽提供的**模板，红帽会提供有限的支持。
- **用户提供的**模板是由 红**帽提供的**，但由用户克隆和创建的模板。

注意

在 Templates 选项卡中，您无法编辑或删除红帽支持的或红帽提供的模板。您只能编辑或删除用户创建的自定义虚拟机模板。

使用红帽模板非常方便，因为模板已预先配置。当您选择一个红帽模板来创建自定义模板时，**Create Virtual Machine Template** 向导会在之前没有添加引导源时提示您添加引导源。然后，您可以保存自定义模板或继续自定义它并保存它。

您还可以直接选择 **Create Virtual Machine Template**向导并创建自定义虚拟机模板。向导提示您提供有关操作系统、类型、工作负载类型和其他设置的配置详情。您可以添加引导源，并继续自定义模板并保存它。

9.1.2. 关于虚拟机和引导源

虚拟机由虚拟机定义以及由数据卷支持的一个或多个磁盘组成。虚拟机模板允许您使用预定义的虚拟机规格创建虚拟机。

每个虚拟机模板都需要一个引导源，它是一个完全配置的虚拟机磁盘镜像，包括配置的驱动程序。每个虚拟机模板包含一个虚拟机定义，其中包含指向引导源的指针。每个引导源都有一个预定义的名称和命名空间。对于某些操作系统，会自动提供一个引导源。如果没有提供，管理员必须准备自定义引导源。

提供的引导源会自动更新至操作系统的最新版本。对于自动更新的引导源，持久性卷声明(PVC)使用集群的默认存储类创建。如果在配置后选择了不同的默认存储类，您必须删除使用之前的默认存储类配置的集群命名空间中的现有数据卷。

要使用引导源功能，请安装 OpenShift Virtualization 的最新版本。命名空间 **openshift-virtualization-os-images** 启用该功能，并安装 OpenShift Virtualization Operator。安装引导源功能后，您可以创建引导源，将它们附加到模板，并从模板创建虚拟机。

使用通过上传本地文件、克隆现有 PVC、从 registry 或 URL 导入的持久性卷声明（PVC）定义引导源。使用 web 控制台将引导源附加到虚拟机模板。在启动源附加到虚拟机模板后，您可从模板创建任意数量的已完全配置的可随时使用虚拟机。

9.1.3. 为虚拟机模板添加引导源

对于您要用于创建虚拟机或自定义模板的任何虚拟机模板，可以配置引导源。当使用引导源配置虚拟机模板时，会在 **Templates** 选项卡中被标记为 **Available**。在向模板中添加引导源后，您可以使用该模板创建新虚拟机。

在 web 控制台中选择和添加引导源有四个方法：

- **上传本地文件（创建 PVC）**
- **通过 URL 导入（创建 PVC）**
- **克隆现有的 PVC（创建 PVC）**
- **通过 Registry 导入（创建 PVC）**

先决条件

- 要添加引导源，您必须以具有 **os-images.kubevirt.io:edit** RBAC 角色或管理员的用户身份登录。您不需要特殊权限才能从附加了引导源的模板创建虚拟机。
- 要上传本地文件，操作系统镜像文件必须存在于本地机器中。
- 要通过 URL 导入，您需要访问带操作系统镜像的 web 服务器。例如：带有镜像的 Red Hat Enterprise Linux 网页。
- 要克隆现有的 PVC，需要使用 PVC 访问项目。
- 要通过 registry 导入，需要访问容器 registry。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 找到您要配置引导源的虚拟机模板并点击 **Add source**。
4. 在 **Add boot source to template** 窗口中，点 **Select boot source**，选择创建持久性卷声明（PVC）的方法： **Upload local file**、**Import via URL**、**Clone existing PVC** 或 **Import via Registry**。
5. 可选：点 **This is a CD-ROM boot source** 来挂载 CD-ROM，并使用它将操作系统安装到空磁盘。OpenShift Virtualization 会自动创建并挂载额外的空磁盘。如果不需要额外磁盘，您可以在创建虚拟机时将其删除。
6. 为 **持久性卷声明大小** 输入一个值，以指定适合未压缩镜像的 PVC 大小，以及任何需要的额外空间。
    a. 可选：输入 **Source 供应商** 名称以将名称与此模板关联。
    b. 可选 ：**高级存储设置**：点 **Storage class** 并选择用于创建磁盘的存储类。通常，这个存储类是创建供所有 PVC 使用的默认存储类。

注意

提供的引导源会自动更新至操作系统的最新版本。对于自动更新的引导源，持久性卷声明(PVC)使用集群的默认存储类创建。如果在配置后选择了不同的默认存储类，您必须删除使用之前的默认存储类配置的集群命名空间中的现有数据卷。

c. 可选：**高级存储设置**：点 **Access 模式**，并为持久性卷选择访问模式：

- **单用户(RWO)** 将卷由单一节点以读写模式挂载。
- **共享访问(RWX)** 将卷挂载为许多节点以读写模式。
- **只读(ROX)** 将卷挂载为多个节点只读。

d. 可选：**高级存储设置**：如果您需要选择 **Block** 而不是默认的值 **Filesystem**，点 **Volume mode**。OpenShift Virtualization 可以静态置备原始块卷。这些卷没有文件系统。对于可以直接写入磁盘或者实现其自己的存储服务的应用程序来说，使用它可以获得性能优势。

7. 选择保存引导源的适当方法：

a. 如果您上传一个本地文件，请点击 **Save and upload**。

b. 如果从 URL 或 registry 中导入内容，点 **Save and import**。

c. 如果克隆现有的 PVC，点 **Save and clone**。

Templates 选项卡中列出了带有引导源的自定义虚拟机模板，您可以使用此模板创建虚拟机。

9.1.3.1. 用于添加引导源的虚拟机模板字段

下表描述了在 **模板窗口中添加引导源的字段**。当您点击 **Templates** 选项卡中的虚拟机模板的 **Add Source** 时会显示这个窗口。

名称	参数	描述
引导源类型	上传本地文件（创建 PVC）	从本地设备上传文件。支持的文件类型包括 gz、xz、tar 和 qcow2。
	通过 URL 导入（创建 PVC）	从 HTTP 或 HTTPS 端点提供的镜像导入内容。从镜像下载的网页获取下载链接 URL，并在 **通过 URL 导入（创建 PVC）** 字段中输入该 URL 链接。示例：对于 Red Hat Enterprise Linux 镜像，登录到红帽客户门户网站，访问进行下载页面，并复制 KVM 客户机镜像的下载链接 URL。
	克隆现有的 PVC（创建 PVC）	使用集群中已可用的 PVC 并克隆它。
	通过 Registry 导入（创建 PVC）	指定位于 registry 中并可从集群访问的可引导操作系统容器。示例： kubevirt/cirros-registry-dis-demo。

名称	参数	描述
源供应商		可选字段。添加有关模板源的描述性文本，或者创建模板的用户名称。例如：Red Hat。
高级	Storage class	用于创建磁盘的存储类。
	访问模式	持久性卷访问模式。支持的访问模式有：**单用户（RWO）**、**共享访问（RWX）** 和 **只读（ROX）**。如果选择了 **Single User（RWO）**，则该磁盘可以被单一节点以读写模式挂载。如果选择了 **Shared Access（RWX）**，则该磁盘可以被多个节点以读写模式挂载。**kubevirt-storage-class-defaults** 配置映射为数据卷提供默认的访问模式。默认值会根据集群中每个存储类的最佳选项设置。 + **注意** 对一些功能（如虚拟机在节点间实时迁移）需要共享访问(RWX)。
	卷模式	定义持久性卷是否使用格式化的文件系统或原始块状态。支持的模式是 Block 和 Filesystem。**kubevirt-storage-class-defaults** 配置映射为数据卷的卷模式提供默认卷模式。默认值会根据集群中每个存储类的最佳选项设置。

9.1.4. 将虚拟机模板标记为热门

为更轻松地访问经常使用的虚拟机模板，您可以将这些模板标记为热门。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 识别您要标记为热门的红帽模板。
4. 点击 Options 菜单 并选择 **Favorite 模板**。在显示的模板列表中，这些模板会被移到更高的位置。

9.1.5. 根据供应商过滤虚拟机模板列表

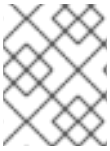
在 **Templates** 选项卡中，您可以使用 **Search by name** 字段通过指定模板名称或标识标识模板来搜索虚拟机模板。您还可以根据供应商过滤模板，并只显示满足过滤条件的模板。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 要过滤模板，点 **Filter**。
4. 从列表中选择适当的复选框来过滤模板： **红帽支持的**、**用户支持的**、**红帽提供的** 和 **用户提供的**。

9.1.6. 使用 web 控制台中的向导创建虚拟机模板

web 控制台具有 **Create Virtual Machine Template**向导，指导您完成 **General**、**Networking**、**Storage**、**Advanced** 和 **Review** 步骤，以简化虚拟机模板的创建过程。所有必填字段均带有一个 *。在为所有所需字段都提供了值之前，**Create Virtual Machine Template**向导会阻止进行到下一步。



注意

向导会指导您创建一个自定义虚拟机模板，在其中可指定操作系统、引导源、类型和其他设置。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 点 **Create** 并选择 **Template with Wizard**。
4. 在 **General** 步骤中填写所有必填字段。
5. 点击 **Next** 进入 **Networking** 步骤。默认会附加名为 **nic0** 的 NIC。
   a. 可选：点 **Add Network Interface** 来创建额外 NIC。
   b. 可选：您可以通过点 Options 菜单  并选择 **Delete** 来删除任何或所有 NIC。从模板创建的虚拟机无需附加 NIC。可在创建虚拟机之后创建 NIC。
6. 点 **Next** 进入 **Storage** 步骤。
7. 点 **Add Disk** 添加磁盘，在 **Add Disk** 页中输入相关信息。

注意

如果选择 **Import via URL (creates PVC)**、**Import via Registry (creates PVC)** 或 **Container (ephemeral)** 作为 **Source**，会创建一个 **rootdisk** 磁盘，并被添加到虚拟机作为 **Bootable Disk**。

如果虚拟机上未附加任何磁盘，则从 **PXE** 源置备的虚拟机无需 **Bootable Disk**。如有一个或多个磁盘附加到虚拟机，您必须将其中一个选为 **Bootable Disk**。

空白磁盘、没有有效引导源的 PVC 磁盘以及 cloudinitdisk 不能用作引导源。

8. 可选：点击 **Advanced** 来配置 cloud-init 和 SSH 访问。

注意

使用 cloud-init 或向导中的自定义脚本，静态注入 SSH 密钥。这使您可以安全、远程管理虚拟机以及管理和传输信息。强烈建议您使用这个步骤来保护您的虚拟机。

9. 点 **Review** 来查看并确认您的设置。
10. 点 **Create Virtual Machine 模板**。
11. 点 **See virtual machine template details** 查看有关虚拟机模板的详情。

模板也会在 **Templates** 选项卡中列出。

9.1.7. 虚拟机模板向导字段

下表描述了 **Create Virtual Machine Template**向导中的 **General**、**Networking**、**Storage** 和 **Advanced** 步骤的字段。

9.1.7.1. 虚拟机模板向导字段

名称	参数	描述
Template		从中创建虚拟机的模板。选择一个模板将自动填写其他字段。
名称		名称可包含小写字母 (**a-z**)、数字 (**0-9**) 和连字符 (**-**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格、句点 (**.**) 或特殊字符。
模板供应商		为集群创建模板的用户名称或用于标识此模板的任何有意义的名称。
模板支持	没有额外的支持	此模板在集群中没有额外的支持。
	由模板供应商支持	此模板由模板供应商支持。
描述		可选的描述字段。
操作系统		为虚拟机选择的操作系统。选择操作系统会自动选择那个操作系统的默认 **Flavor** 和 **Workload 类型**。
引导源	通过 URL 导入（创建 PVC）	从 HTTP 或 HTTPS 端点提供的镜像导入内容。示例：包含操作系统镜像的网页中的 URL 链接。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有持久性卷声明并克隆它。

名称	参数	描述
	通过 Registry 导入（创建 PVC）	从可通过集群访问的注册表中的可启动操作系统容器置备虚拟机。示例：***kubevirt/cirros-registry-disk-demo***。
	PXE（网络引导 - 添加网络接口）	从网络的服务器引导操作系统。需要一个 PXE 可引导网络附加定义。
持久性卷声明项目		用于克隆 PVC 的项目名称。
持久性卷声明名称		如果您要克隆现有的 PVC，则应用于此虚拟机模板的 PVC 名称。
将它作为光盘引导源挂载		CD-ROM 需要额外的磁盘来安装操作系统。选择添加磁盘的选择框并稍后进行自定义。
Flavor	tiny、small、Medium、Large、Custom	根据与该模板关联的操作系统，在虚拟机模板中预设具有预定义值的 CPU 和内存量。 如果选择了默认模板，您可以使用自定义值覆盖模板中的 **cpus** 和 **memsize** 的值，以创建自定义模板。另外，您可以通过修改 **Workloads → Virtualization** 页面上的 **Details** 选项卡中的 **cpus** 和 **memsize** 值来创建自定义模板。
工作负载类型 **注意** 如果您选择了不正确的 **Workload Type**，则可能会出现性能或资源利用率问题（如缓慢的 UI）。	Desktop	用于桌面的虚拟机配置。适用于小型工作环境。建议与 Web 控制台搭配使用。
	Server	在性能和广泛的服务器工作负载兼容性方面具有最佳平衡。
	高性能	针对高性能负载进行了优化的虚拟机配置。

9.1.7.2. 网络字段

名称	描述
名称	网络接口控制器的名称。

名称	描述
model	指明网络接口控制器的型号。支持的值有 **e1000e** 和 **virtio**。
网络	可用网络附加定义的列表。
类型	可用绑定方法列表。对于默认的 pod 网络，**masquerade** 是唯一推荐的绑定方法。对于辅助网络，请使用 **bridge** 绑定方法。非默认网络不支持 **masquerade** 绑定方法。如果您配置了一个 **SR-IOV** 网络设备并在命名空间中定义那个网络，请选择 SR-IOV。
MAC 地址	网络接口控制器的 MAC 地址。如果没有指定 MAC 地址，则会自动分配一个。

9.1.7.3. 存储字段

名称	选择	描述
Source	空白（创建 PVC）	创建一个空磁盘。
	通过 URL 导入（创建 PVC）	通过 URL（HTTP 或 HTTPS 端点）导入内容。
	使用现有的 PVC	使用集群中已可用的 PVC。
	克隆现有的 PVC（创建 PVC）	选择集群中可用的现有 PVC 并克隆它。
	通过 Registry 导入（创建 PVC）	通过容器 registry 导入内容。
	容器（临时）	从集群可以访问的 registry 中的容器上传内容。容器磁盘应只用于只读文件系统，如 CD-ROM 或临时虚拟机。
名称		磁盘的名称。名称可包含小写字母 (**a-z**)、数字 (**0-9**)、连字符 (**-**) 和句点 (**.**)，最多 253 个字符。第一个和最后一个字符必须为字母数字。名称不得包含大写字母、空格或特殊字符。
Size		GiB 中磁盘的大小。

名称	选择	描述
类型		磁盘类型。示例：磁盘或光盘
Interface		磁盘设备的类型。支持的接口包括 virtIO、SATA 和 SCSI。
Storage class		用于创建磁盘的存储类。
Advanced → Volume Mode		定义持久性卷是否使用格式化的文件系统或原始块状态。默认为 Filesystem。
Advanced → Access Mode		持久性卷访问模式。支持的访问模式有 Single User（RWO）、Shared Access（RWX） 和 Read Only（ROX）。

高级存储设置

以下高级存储设置可用于 **空白、从 URL 导入** 和 **克隆现有的 PVC** 磁盘。所有参数都是可选的。如果没有指定这些参数，系统将使用 **kubevirt-storage-class-defaults** 配置映射中的默认值。

名称	参数	描述
卷模式	Filesystem	在基于文件系统的卷中保存虚拟磁盘。
	Block	直接将虚拟磁盘存储在块卷中。只有底层存储支持时才使用 **Block**。
访问模式	Single User (RWO)	这个卷可以被一个单一的节点以 read/write 的形式挂载。
	Shared Access (RWX)	卷可以被多个节点以读写模式挂载。 **注意** 对于一些功能（如虚拟机在节点间实时迁移）需要这个权限。
	Read Only (ROX)	卷可以被多个节点以只读形式挂载。

9.1.7.4. Cloud-init 字段

名称	描述
Hostname	为虚拟机设置特定主机名。

名称	描述
授权 SSH 密钥	复制到虚拟机上 *~/.ssh/authorized_keys* 的用户公钥。
自定义脚本	将其他选项替换为您粘贴自定义 cloud-init 脚本的字段。

9.1.8. 其他资源

- 配置 SR-IOV Network Operator
- 创建并使用引导源

9.2. 编辑虚拟机模板

您可在 web 控制台中更新虚拟机模板，可在 YAML 编辑器中编辑完整的配置，或者在 **Templates** 选项卡中选择自定义模板并修改可编辑的项目。

9.2.1. 在 web 控制台中编辑虚拟机模板

点相关字段旁的铅笔图标来编辑 web 控制台中虚拟机磁盘的选择值。可使用 CLI 编辑其他值。

对于预先配置的红帽模板和自定义虚拟机模板，可编辑标签和注解。所有其他值只适用于用户使用红帽模板或 **Create Virtual Machine Template 向导创建**的自定义虚拟机模板。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 选择虚拟机模板。
4. 点 **VM Template Details**选项卡。
5. 点击铅笔图标使该字段可编辑。
6. 进行相关的更改并点击 **Save**。

编辑虚拟机模板不会影响已经从该模板中创建的虚拟机。

9.2.2. 在 web 控制台中编辑虚拟机模板 YAML 配置

您可从 web 控制台编辑虚拟机模板的 YAML 配置。

某些参数无法修改。如果在进行了无效配置情况下点击 **Save**，则会出现一个错误消息用来指出不可修改的参数。

注意

编辑时离开 YAML 屏幕会取消您对配置做出的任何更改。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 选择模板以打开 **VM Template Details** 屏幕。
4. 点击 **YAML** 选项卡以显示可编辑的配置。
5. 编辑该文件并点击 **Save**。

确认消息显示 YAML 配置已被成功编辑，其中包含对象的更新版本号。

9.2.3. 将虚拟磁盘添加到虚拟机模板

使用这个步骤将虚拟磁盘添加到虚拟机模板。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 选择虚拟机模板以打开 **VM Template Details** 屏幕。
4. 点 **Disks** 选项卡。
5. 在 **Add Disk** 窗口中，指定 **Source**、**Name**、**Size**、**Type**、**Interface** 和 **Storage Class**。
    a. 可选：在 **Advanced** 列表中，为虚拟磁盘指定 **Volume Mode** 和 **Access Mode**。如果没有指定这些参数，系统将使用 **kubevirt-storage-class-defaults** 配置映射中的默认值。
6. 点 **Add**。

9.2.4. 将网络接口添加到虚拟机模板

将网络接口添加到虚拟机模板。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 选择虚拟机模板以打开 **VM Template Details** 屏幕。
4. 点 **Network Interfaces** 选项卡。
5. 点击 **Add Network Interface**。
6. 在 **Add Network Interface** 窗口中，指定网络接口的 **Name**、**Model**、**Network**、**Type** 和 **MAC Address**。
7. 点 **Add**。

9.2.5. 为模板编辑 CD-ROM

使用以下步骤为虚拟机模板编辑 CD-ROM。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。
3. 选择虚拟机模板以打开 **VM Template Details** 屏幕。
4. 点 **Disks** 选项卡。
5. 点您要编辑的 CD-ROM 的 Options 菜单  ，然后选择 **Edit**。
6. 在 **Edit CD-ROM** 窗口中，编辑字段： **Source**、**Persistent Volume Claim**、**Name**、**Type** 和 **Interface**。
7. 点 **Save**。

9.3. 为虚拟机模板启用专用资源

虚拟机可以具有一个节点的资源，比如 CPU，以便提高性能。

9.3.1. 关于专用资源

当为您的虚拟机启用专用资源时，您的工作负载将会在不会被其他进程使用的 CPU 上调度。通过使用专用资源，您可以提高虚拟机性能以及延迟预测的准确性。

9.3.2. 先决条件

- 节点上必须配置 CPU Manager。在调度虚拟机工作负载前，请确认节点具有 **cpumanager = true** 标签。

9.3.3. 为虚拟机模板启用专用资源

您可以在 **Details** 选项卡中为虚拟机模板启用专用资源。使用红帽模板或向导创建的虚拟机可使用专用资源启用。

流程

1. 从侧边菜单中选择 **Workloads → Virtual Machine Templates**。
2. 选择虚拟机模板以打开 **Virtual Machine Template** 选项卡。
3. 点 **Details** 标签页。
4. 点 **Dedicated Resources** 项右面的铅笔标签打开 **Dedicated Resources** 窗口。
5. 选择 **Schedule this workload with dedicated resources (guaranteed policy)**。
6. 点 **Save**。

9.4. 删除虚拟机模板

无法删除红帽虚拟机模板。您可以使用 Web 控制台删除：

- 从红帽模板创建的虚拟机模板
- 使用 **Create Virtual Machine Template**向导创建的自定义虚拟机模板。

9.4.1. 删除 web 控制台中的虚拟机模板

删除虚拟机模板会将其从集群中永久移除。

注意

您可以使用红帽模板或 **Create Virtual Machine Template**向导删除创建的虚拟机模板。预先配置的由红帽提供的虚拟机模板不能被删除。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Templates** 选项卡。选择删除虚拟机模板的适当方法：
   - 点击要删除的模板  的 Options 菜单,然后选择 **Delete Template**。
   - 点模板名称，打开 **Virtual Machine Template Details**屏幕，点 **Actions → Delete Template**。
3. 在确认弹出窗口中点击 **Delete** 永久删除模板。

第 10 章 实时迁移

10.1. 虚拟机实时迁移

10.1.1. 了解实时迁移

实时迁移是在不中断虚拟工作负载或访问的情况下，将正在运行的虚拟机实例（VMI）迁移到集群中另一节点的过程。如果 VMI 使用 **LiveMigrate** 驱除策略，它会在 VMI 运行的节点置于维护模式时自动迁移。您还可以选择要迁移的 VMI 手动启动实时迁移。

虚拟机必须具有一个采用共享 ReadWriteMany (RWX) 访问模式的 PVC 才能实时迁移。

注意

只有在 HyperConverged Cluster 自定义资源（CR）中启用了 **sriovLiveMigration** 功能门时，附加到 SR-IOV 网络接口的虚拟机才支持实时迁移。当 **spec.featureGate.sriovLiveMigration 字** 段设置为 **true** 时，**virt-launcher** pod 使用 **SYS_RESOURCE** 功能运行。这可能会降低其安全性。

10.1.2. 更新实时迁移访问模式

要使实时迁移可以正常工作，必须使用 ReadWriteMany（RWX）访问模式。如果需要，使用此流程更新访问模式。

流程

- 要设置 RWX 访问模式,请运行以下 **oc patch** 命令：

```
$ oc patch -n openshift-cnv \
    cm kubevirt-storage-class-defaults \
    -p '{"data":{"'$<STORAGE_CLASS>'.accessMode":"ReadWriteMany"}}'
```

其他资源：

- 迁移虚拟机实例到另一节点
- 实时迁移限制
- 自定义存储配置集

10.2. 实时迁移限制和超时

应用实时迁移限制和超时，以便迁移过程不会给集群造成负担。通过编辑 **HyperConverged** 自定义资源（CR）来配置这些设置。

10.2.1. 配置实时迁移限制和超时

通过更新位于 **openshift-cnv** 命名空间中的 **HyperConverged** 自定义资源（CR）为集群配置实时迁移限制和超时。

流程

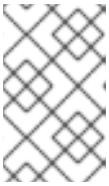
- 编辑 **HyperConverged** CR 并添加必要的实时迁移参数。

```
$ oc edit hco -n openshift-cnv kubevirt-hyperconverged
```

配置文件示例

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  liveMigrationConfig: 1
    bandwidthPerMigration: 64Mi
    completionTimeoutPerGiB: 800
    parallelMigrationsPerCluster: 5
    parallelOutboundMigrationsPerNode: 2
    progressTimeout: 150
```

1 在本例中，**spec.liveMigrationConfig** 数组包含每个字段的默认值。



注意

您可以通过删除该键/值对并保存文件来恢复任何 **spec.liveMigrationConfig** 字段的默认值。例如，删除 **progressTimeout: <value>** 以恢复默认的 **progressTimeout: 150**。

10.2.2. 集群范围内的实时迁移限制和超时

表 10.1. 迁移参数

参数	描述	默认
parallelMigrationsPerCluster	集群中并行运行的迁移数。	5
parallelOutboundMigrationsPerNode	每个节点的最大出站迁移数。	2
bandwidthPerMigration	每次迁移的带宽限制，以 MiB/s 为单位。	0 [1]
completionTimeoutPerGiB	如果迁移未能在此时间内完成则会取消，以每 GiB 内存秒数为单位。例如，如果 6GiB 内存的虚拟机实例未能在 4800 秒内完成，该虚拟机实例将超时。如果 **Migration Method** 是 **BlockMigration**，则迁移磁盘的大小纳入计算中。	800
progressTimeout	如果内存复制未能在此时间内取得进展，则会取消迁移，以秒为单位。	150

1. 默认值 **0** 代表没有限制。

10.3. 迁移虚拟机实例到另一节点

使用 web 控制台或 CLI 手动将虚拟机实例实时迁移到另一节点。

10.3.1. 在 web 控制台中启动虚拟机实例的实时迁移

将正在运行的虚拟机实例迁移到集群中的不同节点。

注意

Migrate Virtual Machine对所有用户可见，但只有管理员用户可以启动虚拟机迁移。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 您可从此屏幕启动迁移，这有助于在一个屏幕中对多个虚拟机执行操作，也可从 **Virtual Machine Overview** 屏幕中进行，其中可查看所选虚拟机的综合详情：
   - 点击虚拟机  末尾的 Options 菜单,然后选择 **Migrate Virtual Machine**。
   - 点击虚拟机名称，打开 **Virtual Machine Overview** 屏幕，然后点击 **Actions → Migrate Virtual Machine**。
4. 点击 **Migrate** 把虚拟机迁移到另一节点。

10.3.2. 在 CLI 中启动虚拟机实例的实时迁移

通过在集群中创建 **VirtualMachineInstanceMigration** 对象并引用虚拟机实例的名称来启动正在运行的虚拟机实例的实时迁移。

流程

1. 为要迁移的虚拟机实例创建 **VirtualMachineInstanceMigration** 配置文件。例如 **VMI-migrate.yaml**：

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstanceMigration
metadata:
  name: migration-job
spec:
  vmiName: vmi-fedora
```

2. 运行以下命令在集群中创建对象：

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

VirtualMachineInstanceMigration 对象触发虚拟机实例的实时迁移。只要虚拟机实例在运行，该对象便始终存在于集群中，除非手动删除。

其他资源：

其他资源：

- 监控虚拟机实例的实时迁移
- 取消虚拟机实例的实时迁移

10.4. 监控虚拟机实例的实时迁移

您可通过 web 控制台或 CLI 监控虚拟机实例的实时迁移进程。

10.4.1. 在 web 控制台中监控虚拟机实例的实时迁移

在迁移期间，虚拟机的状态为 **Migrating**。该状态显示在 **Virtual Machines** 标签页中，或显示在正在迁移的虚拟机的 **Virtual Machine Overview** 屏幕中。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。

10.4.2. 在 CLI 中监控虚拟机实例的实时迁移

虚拟机迁移的状态保存在 **VirtualMachineInstance** 配置的 **Status** 组件中。

流程

- 在正在迁移的虚拟机实例上使用 **oc describe** 命令：

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

输**出示例**

```
...
Status:
  Conditions:
    Last Probe Time:       <nil>
    Last Transition Time:  <nil>
    Status:                True
    Type:                  LiveMigratable
  Migration Method:  LiveMigration
  Migration State:
    Completed:                true
    End Timestamp:            2018-12-24T06:19:42Z
    Migration UID:            d78c8962-0743-11e9-a540-fa163e0c69f1
    Source Node:              node2.example.com
    Start Timestamp:          2018-12-24T06:19:35Z
    Target Node:              node1.example.com
    Target Node Address:      10.9.0.18:43891
    Target Node Domain Detected:  true
```

10.5. 取消虚拟机实例的实时迁移

取消实时迁移，以便虚拟机实例保留在原始节点上。

您可通过 web 控制台或 CLI 取消实时迁移。

10.5.1. 在 web 控制台中取消虚拟机实例的实时迁移

您可以使用 **Virtualization → Virtual Machines** 选项卡中每个虚拟机的 Options 菜单  ，或使用 **Virtual Machine Overview** 屏幕中所有标签页中的 **Actions** 菜单来取消虚拟机实例的实时迁移。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 您可从此屏幕取消迁移，这有助于在一个屏幕中对多个虚拟机执行操作，也可通过 **Virtual Machine Overview** 屏幕进行，其中可查看所选虚拟机的综合详情：
   - 点击虚拟机  末尾的 Options 菜单，然后选择 **Cancel Virtual Machine Migration**。
   - 选择虚拟机名称，打开 **Virtual Machine Overview** 屏幕，然后点击 **Actions → Cancel Virtual Machine Migration**。
4. 点击 **Cancel Migration** 以取消虚拟机实时迁移。

10.5.2. 在 CLI 中取消虚拟机实例的实时迁移

通过删除与迁移关联的 **VirtualMachineInstanceMigration** 对象来取消虚拟机实例的实时迁移。

流程

- 删除触发实时迁移的 **VirtualMachineInstanceMigration** 对象，本例中为 **migration-job**：

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

10.6. 配置虚拟机驱除策略

LiveMigrate 驱除策略可确保当节点置于维护中或排空时，虚拟机实例不会中断。具有驱除策略的虚拟机实例将实时迁移到另一节点。

10.6.1. 使用 LiveMigration 驱除策略配置自定义虚拟机

您只需在自定义虚拟机上配置 **LiveMigration** 驱除策略。通用模板默认已配置该驱除策略。

流程

1. 将 **evictionStrategy: LiveMigrate** 选项添加到虚拟机配置文件的 **spec.template.spec** 部分。本例使用 **oc edit** 来更新 **VirtualMachine** 配置文件中的相关片段：

```
-
```

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

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: custom-vm
spec:
  template:
    spec:
      evictionStrategy: LiveMigrate
...
```

2. 重启虚拟机以使更新生效：

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

第 11 章 节点维护

11.1. 关于节点维护

11.1.1. 了解节点维护模式

节点可以使用 **oc adm** 实用程序或者使用 **NodeMaintenance** 自定义资源（CR）置于维护模式。

将节点置于维护中可将节点标记为不可调度，并排空其中的所有虚拟机和 pod。具有 **LiveMigrate** 驱除策略的虚拟机实例实时迁移到另一节点不会丢失服务。在从通用模板创建的虚拟机中默认配置此驱除策略，而自定义虚拟机则必须手动更配置。

没有驱除策略的虚拟机实例将被关闭。具有 **Running** 或 **RerunOnFailure** 的 **RunStrategy** 的虚拟机会在另一节点上重新创建。带有 **Manual** 的 **RunStrategy** 虚拟机不会被自动重启。

重要

虚拟机必须具有一个采用共享 **ReadWriteMany**（RWX）访问模式的 PVC 才能实时迁移。

当作为 OpenShift Virtualization 的一部分安装时，Node Maintenance Operator 会监视是否有新的或已删除的 **NodeMaintenance** CR。当检测到新的 **NodeMaintenance** CR 时，不会调度新的工作负载，并且该节点从集群的其余部分中分离。所有可被驱除的 pod 都会从节点上驱除。删除 **NodeMaintenance** CR 时，CR 中引用的节点将可用于新工作负载。

注意

使用 **NodeMaintenance** CR 进行节点维护任务可实现与 **oc adm cordon** 和 **oc adm drain** 命令相同的结果，使用标准 OpenShift Container Platform 自定义资源处理。

11.1.2. 维护裸机节点

当您在裸机基础架构上部署 OpenShift Container Platform 时，与在云基础架构上部署相比，还需要考虑其他的注意事项。与集群节点被视为临时的云环境中不同，重新置备裸机节点需要大量时间和精力进行维护任务。

当裸机节点出现故障时，例如，如果发生致命内核错误或发生 NIC 卡硬件故障，在修复或替换问题节点时，故障节点上的工作负载需要重启。节点维护模式允许集群管理员安全关闭节点，将工作负载移到集群的其它部分，并确保工作负载不会中断。详细进度和节点状态详情会在维护过程中提供。

其他资源：

- 关于虚拟机的 RunStrategies
- 虚拟机实时迁移
- 配置虚拟机驱除策略

11.2. 将节点设置为维护模式

通过 Web 控制台、CLI 或者使用 **NodeMaintenance** 自定义资源将节点置于维护模式。

11.2.1. 通过 web 控制台将节点设置为维护模式

使用 **Compute → Nodes** 列表中每个节点上  的 Options 菜单,或使用 **Node Details** 屏幕的 Actions 控件,将节点设置 为 维护模式。

流程

1. 在 OpenShift Virtualization 中，点击 **Compute → Nodes**。
2. 您可从此屏幕将节点设置为维护，这有助于在一个屏幕中对多个虚拟机执行操作，也可通过 **Node Details** 屏幕进行，其中可查看所选节点的综合详情：
   - 点击节点  末尾的 Options 菜单并选择 **Start Maintenance**。
   - 点击节点名称以打开 **Node Details** 屏幕，然后点击 **Actions → Start Maintenance**。
3. 在确认窗口中点击 **Start Maintenance**。

该节点将实时迁移具有 **LiveMigration** 驱除策略的虚拟机实例，且该节点不可再调度。该节点上的所有其他 pod 和虚拟机均被删除，并会在另一节点上重新创建。

11.2.2. 在 CLI 中将节点设置为维护模式

通过将节点标记为不可调度，并使用 **oc adm drain** 命令从节点驱除或删除 pod，将节点设置为维护模式。

流程

1. 将节点标记为不可调度。节点状态变为 **NotReady,SchedulingDisabled**。

```
$ oc adm cordon <node1>
```

2. 排空节点以准备进行维护。节点实时迁移 **LiveMigratable** 条件设置为 **True**，**spec:evictionStrategy** 字段设置为 **LiveMigrate** 的虚拟机实例。该节点上的所有其他 pod 和虚拟机均被删除，并会在另一节点上重新创建。

```
$ oc adm drain <node1> --delete-emptydir-data --ignore-daemonsets=true --force
```

   - **--delete-emptydir-data** 标志会删除节点上使用 **emptyDir** 卷的任何虚拟机实例。这些卷中的数据是临时的，在终止后可以被安全地删除。
   - **--ignore-daemonsets=true** 标志确保守护进程集被忽略，pod 驱除可以成功继续。
   - 需要 **--force** 标志来删除不是由副本集或守护进程设置控制器管理的 pod。

11.2.3. 使用 NodeMaintenance 自定义资源将节点设置为维护模式

您可以使用 **NodeMaintenance** 自定义资源（CR）将节点置于维护模式。应用 **NodeMaintenance** CR 时，所有允许的 pod 都会被驱除并关闭该节点。被驱除的 pod 会被放入到集群中的另一节点中。

先决条件

- 安装 OpenShift Container Platform CLI **oc**。

- 以具有 **cluster-admin** 权限的用户身份登录集群。

流程

1. 创建以下节点维护 CR，并将文件保存为 **nodemaintenance-cr.yaml**：

```
apiVersion: nodemaintenance.kubevirt.io/v1beta1
kind: NodeMaintenance
metadata:
  name: maintenance-example 1
spec:
  nodeName: node-1.example.com 2
  reason: "Node maintenance" 3
```

1. 节点维护 CR 名称
2. 要置于维护模式的节点名称
3. 有关维护原因的纯文本描述

2. 运行以下命令来应用节点维护计划：

```
$ oc apply -f nodemaintenance-cr.yaml
```

3. 运行以下命令，将 **<node-name>** 替换为节点的名称来检查维护任务的进度：

```
$ oc describe node <node-name>
```

输**出示例**

```
Events:
  Type    Reason                  Age                From     Message
  ----    ------                  ----               ----     -------
  Normal  NodeNotSchedulable      61m                kubelet  Node node-1.example.com
status is now: NodeNotSchedulable
```

11.2.3.1. 检查**当前 NodeMaintenance CR 任务的状**态

您可以检查当前 **NodeMaintenance** CR 任务的状态。

先决条件

- 安装 OpenShift Container Platform CLI **oc**。
- 以具有 **cluster-admin** 权限的用户身份登录。

流程

- 运行以下命令，检查当前节点维护任务的状态：

```
$ oc get NodeMaintenance -o yaml
```

输出示例

```
apiVersion: v1
items:
- apiVersion: nodemaintenance.kubevirt.io/v1beta1
  kind: NodeMaintenance
  metadata:
...
  spec:
    nodeName: node-1.example.com
    reason: Node maintenance
  status:
    evictionPods: 3 (1)
    pendingPods:
    - pod-example-workload-0
    - httpd
    - httpd-manual
    phase: Running
    lastError: "Last failure message" (2)
    totalpods: 5
...
```

1. **evictionPods** 是调度用于驱除的 pod 的数量。
2. **lastError** 记录了最新的驱除错误（若有）。

其他资源：

- 从维护模式恢复节点
- 虚拟机实时迁移
- 配置虚拟机驱除策略

11.3. 从维护模式恢复节点

恢复节点会使节点退出维护模式，并使其可再次调度。

通过 web 控制台、CLI 或删除 **NodeMaintenance** 自定义资源从维护模式恢复节点。

11.3.1. 通过 web 控制台从维护模式恢复节点

使用 **Compute → Nodes** 列表中每个节点上  的 Options 菜单,或使用 **Node Details** 屏幕中的 **Actions** 控制,从维护模式恢复节点。

流程

1. 在 OpenShift Virtualization 中，点击 **Compute → Nodes**。
2. 您可从此屏幕恢复节点，这有助于在一个屏幕中对多个虚拟机执行操作，也可从 **Node Details** 屏幕，其中可查看所选节点的综合详情：

- 点击节点  末尾的 Options 菜单并选择 **Stop Maintenance**。
- 点击节点名称以打开 **Node Details** 屏幕，然后点击 **Actions → Stop Maintenance**。

3. 在确认窗口中点击 **Stop Maintenance**。

之后该节点将变为可调度，但维护前在该节点上运行的虚拟机实例不会自动迁移回该节点。

11.3.2. 在 CLI 中从维护模式恢复节点

把节点重新设置为可以调度来恢复处于维护模式的节点。

流程

- 将节点标记为可以调度。然后，您可以恢复在此节点上调度新工作负载。

```
$ oc adm uncordon <node1>
```

11.3.3. 从 NodeMaintenance CR 启动的维护模式恢复节点

您可以通过删除 **NodeMaintenance** CR 来恢复节点。

先决条件

- 安装 OpenShift Container Platform CLI **oc**。
- 以具有 **cluster-admin** 权限的用户身份登录集群。

流程

- 节点维护任务完成后,删除活跃的 **NodeMaintenance** CR：

```
$ oc delete -f nodemaintenance-cr.yaml
```

输**出示例**

```
nodemaintenance.nodemaintenance.kubevirt.io "maintenance-example" deleted
```

11.4. 自动续订 TLS 证书

OpenShift Virtualization 组件的所有 TLS 证书都会被更新并自动轮转。您不需要手动刷新它们。

11.4.1. TLS 证书自动续订计划

TLS 证书会根据以下调度自动删除并替换：

- kubeVirt 证书每天都会被更新。
- 容器化数据导入程序控制器（CDI）证书每 15 天更新一次。
- MAC 池证书会每年续订。

自动 TLS 证书轮转不会破坏任何操作。例如，以下操作可在没有任何中断的情况下继续工作：

- 迁移
- 镜像上传
- VNC 和控制台连接

11.5. 为过时的 CPU 型号管理节点标签

只要节点支持 VM CPU 模型和策略，您可以在节点上调度虚拟机（VM）。

11.5.1. 关于过时 CPU 型号的节点标签

OpenShift Virtualization Operator 使用预定义的过时 CPU 型号列表来确保节点只支持调度的虚拟机的有效 CPU 型号。

默认情况下，从为节点生成的标签列表中删除了以下 CPU 型号：

例 11.1. 过时的 CPU 型号

```
"486"
Conroe
athlon
core2duo
coreduo
kvm32
kvm64
n270
pentium
pentium2
pentium3
pentiumpro
phenom
qemu32
qemu64
```

在 **HyperConverged** CR 中无法看到这个预定义列表。您无法从此列表中 *删除* CPU 型号，但您可以通过编辑 **HyperConverged** CR 的 **spec.obsoleteCPUs.cpuModels** 字段来添加到列表中。

11.5.2. 关于 CPU 功能的节点标签

在迭代过程中，从为节点生成的标签列表中删除最小 CPU 模型中的基本 CPU 功能。

例如：

- 一个环境可能有两个支持的 CPU 型号： **Penryn** 和 **Haswell**。
- 如果将 **Penryn** 指定为 **minCPU** 的 CPU 型号， **Penryn** 的每个基本 CPU 功能都会与 **Haswell** 支持的 CPU 功能列表进行比较。

  例 11.2. Penryn支持的 CPU 功能

  ```
  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
```

例 11.3. Haswell支持的 CPU 功能

```
aes
apic
avx
avx2
bmi1
bmi2
clflush
cmov
cx16
cx8
de
erms
fma
fpu
fsgsbase
fxsr
hle
invpcid
lahf_lm
lm
mca
mce
mmx
```

```
movbe
msr
mtrr
nx
pae
pat
pcid
pclmuldq
pge
pni
popcnt
pse
pse36
rdtscp
rtm
sep
smep
sse
sse2
sse4.1
sse4.2
ssse3
syscall
tsc
tsc-deadline
x2apic
xsave
```

- 如果 **Penryn** 和 **Haswell** 都支持特定的 CPU 功能，则不会为该功能创建一个标签。为仅受 **Haswell** 支持且不受 **Penryn** 支持的 CPU 功能生成标签。

例 11.4. 迭代后为 CPU 功能创建的节点标签

```
aes
avx
avx2
bmi1
bmi2
erms
fma
fsgsbase
hle
invpcid
movbe
pcid
pclmuldq
popcnt
rdtscp
rtm
sse4.2
tsc-deadline
x2apic
xsave
```

11.5.3. 配置过时的 CPU 型号

您可以通过编辑 **HyperConverged** 自定义资源（CR）来配置过时的 CPU 型号列表。

流程

- 编辑 **HyperConverged** 自定义资源，在 **obsoleteCPUs** 阵列中指定过时的 CPU 型号。例如：

```
apiVersion: hco.kubevirt.io/v1beta1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  obsoleteCPUs:
    cpuModels: 1
      - "<obsolete_cpu_1>"
      - "<obsolete_cpu_2>"
    minCPUModel: "<minimum_cpu_model>" 2
```

1. 将 **cpuModels** 数组中的示例值替换为过时的 CPU 型号。您指定的任何值都会添加到预定义的过时 CPU 型号列表中。预定义的列表在 CR 中不可见。
2. 使用您要用于基本 CPU 功能的最低 CPU 型号替换这个值。如果没有指定值，则默认使用 **Penryn**。

11.6. 防止节点协调

使用 **skip-node** 注解来防止 **node-labeller** 协调节点。

11.6.1. 使用 skip-node 注解

如果您希望 **node-labeller** 跳过节点，请使用 **oc** CLI 注解该节点。

先决条件

- 已安装 OpenShift CLI（**oc**）。

流程

- 运行以下命令来注解您要跳过的节点：

```
$ oc annotate node <node_name> node-labeller.kubevirt.io/skip-node=true 1
```

1. 将 **<node_name>** 替换为要跳过的相关节点的名称。

在节点注解被删除或设置为 false 后，协调会在下一个周期中恢复。

11.6.2. 其他资源

- 为过时的 CPU 型号管理节点标签

第 12 章 节点网络

12.1. 观察节点网络状态

节点网络状态是集群中所有节点的网络配置。

12.1.1. 关于 nmstate

OpenShift Virtualization 使用 **nmstate** 来报告并配置节点网络的状态。这样就可以通过将单个配置清单应用到集群来修改网络策略配置，例如在所有节点上创建 Linux 桥接。

节点网络由以下对象监控和更新：

NodeNetworkState

报告该节点上的网络状态。

NodeNetworkConfigurationPolicy

描述节点上请求的网络配置。您可以通过将 **NodeNetworkConfigurationPolicy**清单应用到集群来更新节点网络配置，包括添加和删除网络接口 。

NodeNetworkConfigurationEnactment

报告每个节点上采用的网络策略。

OpenShift Virtualization 支持使用以下 nmstate 接口类型：

- Linux Bridge
- VLAN
- bond
- Ethernet

注意

如果您的 OpenShift Container Platform 集群使用 OVN-Kubernetes 作为默认 Container Network Interface（CNI）供应商，则无法将 Linux 网桥或绑定附加到主机的默认接口，因为 OVN-Kubernetes 的主机网络拓扑发生了变化。作为临时解决方案，您可以使用连接到主机的二级网络接口，或切换到 OpenShift SDN 默认 CNI 供应商。

12.1.2. 查看节点的网络状态

一个 **NodeNetworkState** 对象存在于集群中的每个节点上。此对象定期更新，并捕获该节点的网络状态。

流程

1. 列出集群中的所有 **NodeNetworkState** 对象：

```
$ oc get nns
```

2. 检查 **NodeNetworkState** 对象以查看该节点上的网络。为了清楚，这个示例中的输出已被重新编辑：

```
$ oc get nns node01 -o yaml
```

输出示例

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkState
metadata:
  name: node01 (1)
status:
  currentState: (2)
    dns-resolver:
...
    interfaces:
...
    route-rules:
...
    routes:
...
  lastSuccessfulUpdateTime: "2020-01-31T12:14:00Z" (3)
```

1. **NodeNetworkState** 对象的名称从节点获取。
2. **currentState** 包含节点的完整网络配置，包括 DNS 、接口和路由。
3. 最新成功更新的时间戳。只要节点可以被访问，这个时间戳就会定期更新，它可以用来指示报告的新旧程度。

12.2. 更新节点网络配置

您可以通过将 **NodeNetworkConfigurationPolicy** 清单应用到集群来更新节点网络的配置，如为节点添加或删除接口。

12.2.1. 关于 nmstate

OpenShift Virtualization 使用 **nmstate** 来报告并配置节点网络的状态。这样就可以通过将单个配置清单应用到集群来修改网络策略配置，例如在所有节点上创建 Linux 桥接。

节点网络由以下对象监控和更新：

NodeNetworkState

报告该节点上的网络状态。

NodeNetworkConfigurationPolicy

描述节点上请求的网络配置。您可以通过将 **NodeNetworkConfigurationPolicy**清单应用到集群来更新节点网络配置，包括添加和删除网络接口 。

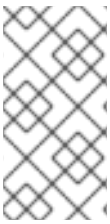
NodeNetworkConfigurationEnactment

报告每个节点上采用的网络策略。

OpenShift Virtualization 支持使用以下 nmstate 接口类型：

- Linux Bridge

- VLAN
- bond
- Ethernet



注意

如果您的 OpenShift Container Platform 集群使用 OVN-Kubernetes 作为默认 Container Network Interface（CNI）供应商，则无法将 Linux 网桥或绑定附加到主机的默认接口，因为 OVN-Kubernetes 的主机网络拓扑发生了变化。作为临时解决方案，您可以使用连接到主机的二级网络接口，或切换到 OpenShift SDN 默认 CNI 供应商。

12.2.2. 在节点上创建接口

通过将一个 **NodeNetworkConfigurationPolicy** 清单应用到集群来在集群的节点上创建一个接口。清单详细列出了请求的接口配置。

默认情况下，清单会应用到集群中的所有节点。要将接口只添加到特定的节点，在节点选择器上添加 **spec: nodeSelector** 参数和适当的 **<key>:<value>** 。

流程

1. 创建 **NodeNetworkConfigurationPolicy** 清单。以下示例在所有 worker 节点上配置了一个 Linux 桥接：

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: <br1-eth1-policy> 1
spec:
  nodeSelector: 2
    node-role.kubernetes.io/worker: "" 3
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with eth1 as a port 4
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: eth1
```

1. 策略的名称。
2. 可选： 如果没有包括 **nodeSelector** 参数，策略会应用到集群中的所有节点。
3. 本例使用 **node-role.kubernetes.io/worker： ""** 节点选择器来选择集群中的所有 worker 节点。

4 可选：接口人类可读的描述。

2. 创建节点网络策略：

```
$ oc apply -f <br1-eth1-policy.yaml> 1
```

1 节点网络配置策略清单的文件名。

其他资源

- 不同接口的策略配置示例
- 在相同策略中创建多个接口的示例
- 策略中不同 IP 管理方法示例

12.2.3. 确认节点上的节点网络策略更新

NodeNetworkConfigurationPolicy 清单描述了您为集群中的节点请求的网络配置。节点网络策略包括您请求的网络配置以及整个集群中的策略执行状态。

当您应用节点网络策略时，会为集群中的每个节点创建一个 **NodeNetworkConfigurationEnactment** 对象。节点网络配置是一个只读对象，代表在该节点上执行策略的状态。如果策略在节点上应用失败，则该节点会包括 traceback 用于故障排除。

流程

1. 要确认策略已应用到集群，请列出策略及其状态：

```
$ oc get nncp
```

2. 可选：如果策略配置成功的时间比预期的要长，您可以检查特定策略请求的状态和状态条件：

```
$ oc get nncp <policy> -o yaml
```

3. 可选：如果策略在所有节点上配置成功的时间比预期的要长，您可以列出集群中的 Enactments 的状态：

```
$ oc get nnce
```

4. 可选：要查看特定的 Enactment 的配置，包括对失败配置进行任何错误报告：

```
$ oc get nnce <node>.<policy> -o yaml
```

12.2.4. 从节点中删除接口

您可以通过编辑 **NodeNetworkConfigurationPolicy** 对象从集群中的一个或多个节点中删除接口，并将接口的**状**态设置为 **absent**。

从节点中删除接口不会自动将节点网络配置恢复到以前的状态。如果要恢复之前的状态，则需要在策略中定义节点网络配置。

如果删除了网桥或绑定接口，以前附加到该网桥或绑定接口的任何节点 NIC 都会处于 **down** 状态并变得不可访问。为了避免连接丢失，在相同策略中配置节点 NIC，使其具有 **up** 状态，以及使用 DHCP 或一个静态 IP 地址。

注意

删除添加接口的节点网络策略不会更改节点上的策略配置。虽然 **NodeNetworkConfigurationPolicy** 是集群中的一个对象，但它只代表请求的配置。同样，删除接口不会删除策略。

流程

1. 更新用来创建接口的 **NodeNetworkConfigurationPolicy** 清单。以下示例删除了 Linux 网桥，并使用 DHCP 配置 **eth1** NIC 以避免断开连接：

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: <br1-eth1-policy> 1
spec:
  nodeSelector: 2
    node-role.kubernetes.io/worker: "" 3
  desiredState:
    interfaces:
    - name: br1
      type: linux-bridge
      state: absent 4
    - name: eth1 5
      type: ethernet 6
      state: up 7
      ipv4:
        dhcp: true 8
        enabled: true 9
```

1. 策略的名称。
2. 可选： 如果没有包括 **nodeSelector** 参数，策略会应用到集群中的所有节点。
3. 本例使用 **node-role.kubernetes.io/worker： ""** 节点选择器来选择集群中的所有 worker 节点。
4. 将状态改为 **absent** 会删除接口。
5. 要从网桥接口中取消附加的接口名称。
6. 接口的类型。这个示例创建了以太网网络接口。
7. 接口的请求状态。
8. 可选：如果您不使用 **dhcp**，可以设置静态 IP，或让接口没有 IP 地址。
9. 在这个示例中启用 **ipv4** 。

2. 更新节点上的策略并删除接口：

```
$ oc apply -f <br1-eth1-policy.yaml> 1
```

1 策略清单的文件名。

12.2.5. 不同接口的策略配置示例

12.2.5.1. 示例： Linux bridge interface 节点网络配置策略

通过将一个 **NodeNetworkConfigurationPolicy** 清单应用到集群来在集群的节点上创建一个 Linux 网桥接口。

以下 YAML 文件是 Linux 网桥界面的清单示例。如果运行 playbook，其中会包含必须替换为您自己的信息的样本值。

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-policy 1
spec:
  nodeSelector: 2
    kubernetes.io/hostname: <node01> 3
  desiredState:
    interfaces:
      - name: br1 4
        description: Linux bridge with eth1 as a port 5
        type: linux-bridge 6
        state: up 7
        ipv4:
          dhcp: true 8
          enabled: true 9
        bridge:
          options:
            stp:
              enabled: false 10
          port:
            - name: eth1 11
```

1 策略的名称。

2 可选： 如果没有包括 **nodeSelector** 参数，策略会应用到集群中的所有节点。

3 这个示例使用 **hostname** 节点选择器。

4 接口的名称。

5 可选：接口人类可读的接口描述。

6 接口的类型。这个示例会创建一个桥接。

7 创建后接口的请求状态。

8 可选：如果您不使用 **dhcp**，可以设置静态 IP，或让接口没有 IP 地址。

9 在这个示例中启用 **ipv4** 。

10 在这个示例中禁用 **stp** 。

11 网桥附加到的节点 NIC。

12.2.5.2. 示例：VLAN 接口节点网络配置策略

通过将一个 **NodeNetworkConfigurationPolicy** 清单应用到集群来在集群的节点上创建一个 VLAN 接口。

以下 YAML 文件是 VLAN 接口的清单示例。如果运行 playbook，其中会包含必须替换为您自己的信息的样本值。

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: vlan-eth1-policy 1
spec:
  nodeSelector: 2
    kubernetes.io/hostname: <node01> 3
  desiredState:
    interfaces:
    - name: eth1.102 4
      description: VLAN using eth1 5
      type: vlan 6
      state: up 7
      vlan:
        base-iface: eth1 8
        id: 102 9
```

1 策略的名称。

2 可选： 如果没有包括 **nodeSelector** 参数，策略会应用到集群中的所有节点。

3 这个示例使用 **hostname** 节点选择器。

4 接口的名称。

5 可选：接口人类可读的接口描述。

6 接口的类型。这个示例创建了一个 VLAN。

7 创建后接口的请求状态。

8 附加 VLAN 的节点 NIC。

9 VLAN 标签。

12.2.5.3. 示例：绑定接口节点网络配置策略

通过将一个 **NodeNetworkConfigurationPolicy** 清单应用到集群来在集群的节点上创建一个绑定接口。

注意

OpenShift Virtualization 只支持以下绑定模式：

- mode=1 active-backup
- mode=2 balance-xor
- mode=4 802.3ad
- mode=5 balance-tlb
- mode=6 balance-alb

以下 YAML 文件是绑定接口的清单示例。如果运行 playbook，其中会包含必须替换为您自己的信息的样本值。

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: bond0-eth1-eth2-policy (1)
spec:
  nodeSelector: (2)
    kubernetes.io/hostname: <node01> (3)
  desiredState:
    interfaces:
    - name: bond0 (4)
      description: Bond enslaving eth1 and eth2 (5)
      type: bond (6)
      state: up (7)
      ipv4:
        dhcp: true (8)
        enabled: true (9)
      link-aggregation:
        mode: active-backup (10)
        options:
          miimon: '140' (11)
        slaves: (12)
        - eth1
        - eth2
      mtu: 1450 (13)
```

1. 策略的名称。
2. 可选： 如果没有包括 **nodeSelector** 参数，策略会应用到集群中的所有节点。
3. 这个示例使用 **hostname** 节点选择器。
4. 接口的名称。
5. 可选：接口人类可读的接口描述。
6. 接口的类型。这个示例创建了一个绑定。

7 创建后接口的请求状态。

8 可选：如果您不使用 **dhcp**，可以设置静态 IP，或让接口没有 IP 地址。

9 在这个示例中启用 **ipv4** 。

10 Bond 的驱动模式。这个示例使用 active 备份模式。

11 可选：本例使用 miimon 检查每 140ms 的绑定链接。

12 绑定中的下级节点 NIC。

13 可选：绑定的最大传输单元（MTU）。如果没有指定，其默认值为 **1500**。

12.2.5.4. 示例：以太网接口节点网络配置策略

通过将 **NodeNetworkConfigurationPolicy** 清单应用到集群，在集群的节点上配置以太网接口。

以下 YAML 文件是一个以太接口的清单示例。它包含了示例值，需要使用自己的信息替换。

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: eth1-policy 1
spec:
  nodeSelector: 2
    kubernetes.io/hostname: <node01> 3
  desiredState:
    interfaces:
    - name: eth1 4
      description: Configuring eth1 on node01 5
      type: ethernet 6
      state: up 7
      ipv4:
        dhcp: true 8
        enabled: true 9
```

1 策略的名称。

2 可选： 如果没有包括 **nodeSelector** 参数，策略会应用到集群中的所有节点。

3 这个示例使用 **hostname** 节点选择器。

4 接口的名称。

5 可选：接口人类可读的接口描述。

6 接口的类型。这个示例创建了以太网网络接口。

7 创建后接口的请求状态。

8 可选：如果您不使用 **dhcp**，可以设置静态 IP，或让接口没有 IP 地址。

9 在这个示例中启用 **ipv4** 。

12.2.5.5. 示例：同一节点网络配置策略中的多个接口

您可以在相同的节点网络配置策略中创建多个接口。这些接口可以相互引用，允许您使用单个策略清单来构建和部署网络配置。

以下示例片断在两个 NIC 间创建一个名为 **bond10** 的绑定和一个名为 **br1** 连接到绑定的 Linux 网桥。

```
...
    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
      state: up
      bridge:
        port:
        - name: bond10
...
```

12.2.6. 示例：IP 管理

以下配置片段示例演示了不同的 IP 管理方法。

这些示例使用 **ethernet** 接口类型来简化示例，同时显示 Policy 配置中相关的上下文。这些 IP 管理示例可与其他接口类型一起使用。

12.2.6.1. Static

以下片段在以太网接口中静态配置 IP 地址：

```
...
    interfaces:
    - name: eth1
      description: static IP on eth1
      type: ethernet
      state: up
      ipv4:
        dhcp: false
        address:
        - ip: 192.168.122.250 1
          prefix-length: 24
        enabled: true
...
```

1 使用接口的静态 IP 地址替换这个值。

12.2.6.2. 没有 IP 地址

以下片段确保接口没有 IP 地址：

```
...
    interfaces:
    - name: eth1
      description: No IP on eth1
      type: ethernet
      state: up
      ipv4:
        enabled: false
...
```

12.2.6.3. 动态主机配置

以下片段配置了一个以太网接口，它使用动态 IP 地址、网关地址和 DNS：

```
...
    interfaces:
    - name: eth1
      description: DHCP on eth1
      type: ethernet
      state: up
      ipv4:
        dhcp: true
        enabled: true
...
```

以下片段配置了一个以太网接口，它使用动态 IP 地址，但不使用动态网关地址或 DNS：

```
...
    interfaces:
    - name: eth1
      description: DHCP without gateway or DNS on eth1
      type: ethernet
      state: up
      ipv4:
        dhcp: true
        auto-gateway: false
        auto-dns: false
        enabled: true
...
```

12.2.6.4. DNS

以下片段在主机上设置 DNS 配置。

```
...
    interfaces:
      ...
    dns-resolver:
      config:
        search:
```

```
        - example.com
        - example.org
        server:
        - 8.8.8.8
...
```

12.2.6.5. 静态路由

以下片段在接口 **eth1** 中配置静态路由和静态 IP。

```
...
    interfaces:
    - name: eth1
      description: Static routing on eth1
      type: ethernet
      state: up
      ipv4:
        dhcp: false
        address:
        - ip: 192.0.2.251 1
          prefix-length: 24
        enabled: true
    routes:
      config:
      - destination: 198.51.100.0/24
        metric: 150
        next-hop-address: 192.0.2.1 2
        next-hop-interface: eth1
        table-id: 254
...
```

1 以太网接口的静态 IP 地址。

2 节点流量的下一跳地址。这必须与为以太接口设定的 IP 地址位于同一个子网中。

12.3. 对节点网络配置进行故障排除

如果节点网络配置遇到问题，则策略会自动回滚，且报告失败。这包括如下问题：

- 配置没有在主机上应用。
- 主机丢失了到默认网关的连接。
- 断开了与 API 服务器的连接。

12.3.1. 对不正确的节点网络配置策略配置进行故障排除

您可以通过应用节点网络配置策略，对整个集群中的节点网络配置应用更改。如果应用了不正确的配置，您可以使用以下示例进行故障排除并修正失败的节点网络策略。

在本例中，一个 Linux 网桥策略应用到一个具有三个 control plane 节点（master）和三个计算（worker）节点的示例集群。策略无法应用，因为它会引用了一个不正确的接口。要查找错误，调查可用的 NMState 资源。然后您可以使用正确配置来更新策略。

流程

1. 创建策略并将其应用到集群。以下示例在 **ens01** 接口上创建了一个简单桥接：

```
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ens01-bridge-testfail
spec:
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with the wrong port
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: ens01
```

```
$ oc apply -f ens01-bridge-testfail.yaml
```

输**出示例**

```
nodenetworkconfigurationpolicy.nmstate.io/ens01-bridge-testfail created
```

2. 运行以下命令，验证策略的状态：

```
$ oc get nncp
```

输出显示策略失败：

输**出示例**

```
NAME                  STATUS
ens01-bridge-testfail   FailedToConfigure
```

但是，仅有策略状态并不表示它在所有节点或某个节点子集中是否失败。

3. 列出节点网络配置以查看策略在任意节点上是否成功。如果策略只针对某个节点子集失败，这表示问题在于特定的节点配置。如果策略在所有节点上都失败，这表示问题在于策略。

```
$ oc get nnce
```

输出显示策略在所有节点上都失败：

输**出示例**

```
NAME                                STATUS
control-plane-1.ens01-bridge-testfail      FailedToConfigure
control-plane-2.ens01-bridge-testfail      FailedToConfigure
control-plane-3.ens01-bridge-testfail      FailedToConfigure
compute-1.ens01-bridge-testfail            FailedToConfigure
compute-2.ens01-bridge-testfail            FailedToConfigure
compute-3.ens01-bridge-testfail            FailedToConfigure
```

4. 查看失败的原因之一并查看回溯信息。以下命令使用输出工具 **jsonpath** 来过滤输出结果：

```
$ oc get nnce compute-1.ens01-bridge-testfail -o jsonpath='{.status.conditions[?
(@.type=="Failing")].message}'
```

这个命令会返回一个大的回溯信息，它被编辑为 brevity:

输出示例

```
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:
      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

difference
==========
--- 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: []
  line 651, in _assert_interfaces_equal\n
current_state.interfaces[ifname],\nlibnmstate.error.NmstateVerificationError:
```

NmstateVerificationError 列出了 **desired（期望的）** 策略配置，策略在节点上的**current（当前的）** 配置，并高亮标识了不匹配参数间的**difference（不同）**。在本示例中，此 **port** 包含在 **difference** 部分，这表示策略中的端口配置问题。

5. 要确保正确配置了策略，请求 **NodeNetworkState** 来查看一个或多个节点的网络配置。以下命令返回 **control-plane-1** 节点的网络配置：

```
$ oc get nns control-plane-1 -o yaml
```

输出显示节点上的接口名称为 **ens1**，但失败的策略使用了 **ens01**:

输出示例

```
  - ipv4:
...
    name: ens1
    state: up
    type: ethernet
```

6. 通过编辑现有策略修正错误：

```
$ oc edit nncp ens01-bridge-testfail
```

```
...
        port:
          - name: ens1
```

保存策略以应用更正。

7. 检查策略的状态，以确保它被成功更新：

```
$ oc get nncp
```

输出示例

```
NAME                    STATUS
ens01-bridge-testfail   SuccessfullyConfigured
```

在集群中的所有节点上都成功配置了更新的策略。

第 13 章 日志记录、事件和监控

13.1. 查看虚拟机日志

13.1.1. 了解虚拟机日志

收集 OpenShift Container Platform 构建、部署和 Pod 日志。在 OpenShift Virtualization 中，可在 web 控制台或 CLI 中从虚拟机启动程序 pod 中检索虚拟机日志。

-f 选项会实时跟随日志输出，可用于监控进度和进行错误检查。

如果启动程序 pod 无法启动，则请使用 **--previous** 选项查看最后一次尝试的日志。

警告

所引用镜像的部署配置不当或存在问题均可能引发 **ErrImagePull** 和 **ImagePullBackOff** 错误。

13.1.2. 在 CLI 中查看虚拟机日志

从虚拟机启动程序 Pod 中获取虚拟机日志。

流程

- 使用以下命令：

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

13.1.3. 在 web 控制台中查看虚拟机日志

从关联的虚拟机启动程序 Pod 中获取虚拟机日志。

流程

1. 在 OpenShift Virtualization 控制台中，从侧边菜单中点击 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 进入 **Details** 选项卡，点击 **Pod** 部分的 **virt-launcher-<vm-name>** pod。
5. 点击 **Logs**。

13.2. 查看事件

13.2.1. 了解虚拟机事件

OpenShift Container Platform 事件是命名空间中重要生命周期信息的记录，有助于对资源调度、创建和删除问题进行监控和故障排除。

OpenShift Virtualization 为虚拟机和虚拟机实例添加事件。您可以在 Web 控制台或 CLI 中查看这些事件。

另请参阅：查看 OpenShift Container Platform 集群中的系统事件信息。

13.2.2. 在 web 控制台中查看虚拟机的事件

您可以在 web 控制台的 **Virtual Machine Overview** 面板中查看正在运行的虚拟机的流事件。

▮▮ 按钮可暂停事件流。
▶ 按钮可继续暂停的事件流。

流程

1. 从侧边菜单中点 **Workloads → Virtualization**。
2. 点 **Virtual Machines** 标签页。
3. 选择虚拟机以打开 **Virtual Machine Overview** 屏幕。
4. 点击 **Events** 以查看虚拟机的所有事件。

13.2.3. 在 CLI 中查看命名空间事件

使用 OpenShift Container Platform 客户端来获取命名空间的事件。

流程

- 在命名空间中，使用 **oc get** 命令：

```
$ oc get events
```

13.2.4. 在 CLI 中查看资源事件

资源描述中包含事件，您可以使用 OpenShift Container Platform 客户端获取这些事件。

流程

- 在命名空间中，使用 **oc describe** 命令。以下示例演示了如何为虚拟机、虚拟机实例和虚拟机的 virt-launcher pod 获取事件：

```
$ oc describe vm <vm>
```

```
$ oc describe vmi <vmi>
```

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

13.3. 使用事件和条件诊断数据卷

使用 **oc describe** 命令分析并帮助解决数据卷的问题。

13.3.1. 关于条件和事件

通过检查命令生成的 **Conditions** 和 **Events** 部分的输出结果来诊断数据卷的问题：

```
$ oc describe dv <DataVolume>
```

在 **Conditions** 部分会有三个 **Types**：

- **Bound**
- **Running**
- **Ready**

Events 部分提供以下额外信息：

- 事件类**型**
- 日志**原因**
- 事件**源**
- 包含其他诊断信息的**消息**。

oc describe 的输出并不总是包含 **Events**。

当 **Status**、**Reason** 或 **Message** 改变时会产生一个事件。条件和事件均响应数据卷状态的变化。

例如，在导入操作中错误拼写了 URL，则导入会生成 404 信息。该消息的更改会生成一个带有原因的事件。**Conditions** 部分中的输出也会更新。

13.3.2. 使用条件和事件分析数据卷

通过检查 **describe** 命令生成的 **Conditions** 和 **Events** 部分，您可以确定与 PVC相关的数据卷的状态，以及某个操作是否正在主动运行或完成。您可能还会收到信息，它们提供了有关数据卷状态的特定详情，以及如何处于当前状态。

有多种条件的组合。对每个条件组合的评估都必须其特定的环境下进行。

下面是各种组合的例子。

- **Bound** - 本示例中显示一个成功绑定 PVC。
  请注意, **Type** 是 **Bound**,所以 **Status** 为 **True**。如果 PVC 没有绑定，**Status** 为 **False**。

  当 PVC 被绑定时，会生成一个事件声明 PVC 已被绑定。在本例中, **Reason** 为 **Bound**，**Status** 为 **True**。**Message** 指明了哪个 PVC 拥有数据卷。

  在 **Events** 部分，**Message** 提供了更多详细信息，包括 PVC 被绑定的时间（**Age**）和它的源（**From**），在本例中是 **datavolume-controller**:

  输**出示例**

  ```
  Status:
   Conditions:
  ```

```
  Last Heart Beat Time:  2020-07-15T03:58:24Z
  Last Transition Time:  2020-07-15T03:58:24Z
  Message:               PVC win10-rootdisk Bound
  Reason:                Bound
  Status:                True
  Type:                  Bound

Events:
  Type    Reason    Age    From                Message
  ----    ------    ----   ----                -------
  Normal  Bound     24s    datavolume-controller  PVC example-dv Bound
```

- **Running** - 在本例中，请注意 **Type** 是 **Running**，**Status** 为 **False**。这表示发生事件导致尝试的操作失败，将 Status 从 **True** 改为 **False**。
  然而，请注意 **Reason** 是 **Completed**，**Message** 显示 **Import Complete**。

  在 **Events** 部分，**Reason** 和 **Message** 包含有关失败操作的额外故障排除信息。在这个示例中，**Message** 显示因为 **404** 无法连接，这在 **Events** 部分的第一个 **Warning** 中列出。

  根据这些信息，您认为导入操作正在运行，并为试图访问数据卷的其他操作创建竞争：

  输**出示例**

```
Status:
  Conditions:
    Last Heart Beat Time:  2020-07-15T04:31:39Z
    Last Transition Time:  2020-07-15T04:31:39Z
    Message:               Import Complete
    Reason:                Completed
    Status:                False
    Type:                  Running

  Events:
    Type     Reason          Age              From                Message
    ----     ------          ----             ----                -------
    Warning  Error           12s (x2 over 14s)  datavolume-controller  Unable to connect
    to http data source: expected status code 200, got 404. Status: 404 Not Found
```

- **Ready** - 如果 **Type** 是 **Ready**，**Status** 为**True**，则代表数据卷已就绪，如下例所示。如果数据卷未就绪，则 **Status** 为 **False**:

  输**出示例**

```
Status:
  Conditions:
    Last Heart Beat Time: 2020-07-15T04:31:39Z
    Last Transition Time:  2020-07-15T04:31:39Z
    Status:                True
    Type:                  Ready
```

13.4. 查看有关虚拟机工作负载的信息

您可以使用 OpenShift Container Platform Web 控制台中的 **Virtual Machines** dashboard 来查看有关虚拟机的高级别的信息。

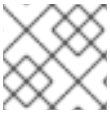
13.4.1. 关于虚拟机仪表板

在 OpenShift Container Platform web 控制台中，进入 **Workloads → Virtualization** 页来访问虚拟机。**Workloads → Virtualization** 页面包含两个标签：

- **虚拟机**
- **虚拟机模板**

以下卡描述了每个虚拟机：

- **Details** 提供了有关虚拟机的识别信息，包括：
  - 名称
  - 命名空间
  - 创建日期
  - 节点名称
  - IP 地址
- **Inventory** 列出虚拟机的资源，包括：
  - 网络接口控制卡 (NIC)
  - 磁盘
- **Status** 包括：
  - 虚拟机的当前状态
  - 标明是否在虚拟机上安装了 QEMU 客户机代理
- **使用率**包括显示以下使用数据的图表 ：
  - CPU
  - 内存
  - Filesystem
  - 网络传输



注意

使用下拉列表选择使用率数据持续的时间。可用选项包括 1 Hour、6 Hours 和 24 Hours。

- **Events** 列出了过去几小时中有关虚拟机活动的信息 。要查看其他事件，请点击 **View all**。

13.5. 监控虚拟机健康状况

虚拟机实例 (VMI) 可能会变得不健康，原因可能源自连接丢失、死锁或外部依赖项相关问题等临时问题。健康检查使用就绪度和存活度探测的组合定期对 VMI 执行诊断。

13.5.1. 关于就绪度和存活度探测

使用就绪度和存活度探测来检测和处理不健康的虚拟机实例 (VMI)。您可以在 VMI 规格中包含一个或多个探测，以确保流量无法访问未准备好的 VMI，并在 VMI 变得不响应时创建新实例。

*就绪度探测*决定 VMI 是否准备好接受服务请求。如果探测失败，则 VMI 会从可用端点列表中移除，直到 VMI 就绪为止。

*存活度探测*决定 VMI 是否响应。如果探测失败，则会删除 VMI 并创建一个新实例来恢复响应性。

您可以通过设置 **VirtualMachineInstance** 对象的 **spec.readinessProbe** 和 **spec.livenessProbe** 字段来配置就绪度和存活度探测。这些字段支持以下测试：

HTTP GET

该探测使用 Web hook 确定 VMI 的健康状况。如果 HTTP 响应代码介于 200 和 399 之间，则测试成功。您可以将 HTTP GET 测试用于在完全初始化时返回 HTTP 状态代码的应用程序。

TCP 套接字

该探测尝试为 VMI 打开一个套接字。只有在探测可以建立连接时，VMI 才被视为健康。对于在初始化完成前不会开始监听的应用程序，可以使用 TCP 套接字测试。

13.5.2. 定义 HTTP 就绪度探测

通过设置虚拟机实例 (VMI) 配置的 **spec.readinessProbe.httpGet** 字段来定义 HTTP 就绪度探测。

流程

1. 在 VMI 配置文件中包括就绪度探测的详细信息。

使用 HTTP GET 测试就绪度探测示例

```
# ...
spec:
  readinessProbe:
    httpGet: 1
      port: 1500 2
      path: /healthz 3
      httpHeaders:
      - name: Custom-Header
        value: Awesome
    initialDelaySeconds: 120 4
    periodSeconds: 20 5
    timeoutSeconds: 10 6
    failureThreshold: 3 7
    successThreshold: 3 8
# ...
```

1. 执行的 HTTP GET 请求以连接 VMI。
2. 探测查询的 VMI 端口。在上例中，探测查询端口 1500。
3. 在 HTTP 服务器上访问的路径。在上例中，如果服务器的 /healthz 路径的处理程序返回成功代码，则 VMI 被视为健康。如果处理程序返回失败代码，则 VMI 将从可用端点列表中移除。

4. VMI 启动就绪度探测前的时间（以秒为单位）。
5. 执行探测之间的延迟（以秒为单位）。默认延迟为 10 秒。这个值必须大于 **timeoutSeconds**。
6. 在不活跃的时间（以秒为单位）超过这个值时探测会超时，且假定 VMI 失败。默认值为 1。这个值必须小于 **periodSeconds**。
7. 探测允许失败的次数。默认值为 3。在进行了指定数量的尝试后，pod 被标记为 **Unready**。
8. 在失败后，在探测报告成功的次数达到这个值时才能被视为成功。默认值为 1。

2. 运行以下命令来创建 VMI：

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

13.5.3. 定义 TCP 就绪度探测

通过设置虚拟机实例 (VMI) 配置的 **spec.readinessProbe.tcpSocket** 字段来定义 TCP 就绪度探测。

流程

1. 在 VMI 配置文件中包括 TCP 就绪探测的详细信息。

使用 TCP 套接字测试的就绪度探测示例

```
...
spec:
  readinessProbe:
    initialDelaySeconds: 120 1
    periodSeconds: 20 2
    tcpSocket: 3
      port: 1500 4
    timeoutSeconds: 10 5
...
```

1. VMI 启动就绪度探测前的时间（以秒为单位）。
2. 执行探测之间的延迟（以秒为单位）。默认延迟为 10 秒。这个值必须大于 **timeoutSeconds**。
3. 要执行的 TCP 操作。
4. 探测查询的 VMI 端口。
5. 在不活跃的时间（以秒为单位）超过这个值时探测会超时，且假定 VMI 失败。默认值为 1。这个值必须小于 **periodSeconds**。

2. 运行以下命令来创建 VMI：

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

13.5.4. 定义 HTTP 存活度探测

通过设置虚拟机实例 (VMI) 配置的 **spec.livenessProbe.httpGet** 字段来定义 HTTP 存活度探测。您可以按照与就绪度探测相同的方式为存活度探测定义 HTTP 和 TCP 测试。此流程使用 HTTP GET 测试配置示例存活度探测。

流程

1. 在 VMI 配置文件中包括 HTTP 存活度探测的详细信息。

使用 HTTP GET 测试的存活度探测示例

```
# ...
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
# ...
```

1. VMI 启动存活度探测前的时间（以秒为单位）。
2. 执行探测之间的延迟（以秒为单位）。默认延迟为 10 秒。这个值必须大于 **timeoutSeconds**。
3. 执行的 HTTP GET 请求以连接 VMI。
4. 探测查询的 VMI 端口。在上例中，探测查询端口 1500。VMI 通过 cloud-init 在端口 1500 上安装并运行最小 HTTP 服务器。
5. 在 HTTP 服务器上访问的路径。在上例中，如果服务器的 **/healthz** 路径的处理程序返回成功代码，则 VMI 被视为健康。如果处理程序返回失败代码，则 VMI 被删除并创建新实例。
6. 在不活跃的时间（以秒为单位）超过这个值时探测会超时，且假定 VMI 失败。默认值为 1。这个值必须小于 **periodSeconds**。

2. 运行以下命令来创建 VMI：

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

13.5.5. 模板：用于定义健康检查的虚拟机实例配置文件

```
apiVersion: kubevirt.io/v1
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:
  - name: containerdisk
    containerDisk:
      image: kubevirt/fedora-cloud-registry-disk-demo
  - cloudInitNoCloud:
      userData: |-
        #cloud-config
        password: fedora
        chpasswd: { expire: False }
        bootcmd:
          - setenforce 0
          - dnf install -y nmap-ncat
          - systemd-run --unit=httpserver nc -klp 1500 -e '/usr/bin/echo -e HTTP/1.1 200 OK\\n\\nHello
World!'
    name: cloudinitdisk
```

13.5.6. 其他资源

- 使用健康检查来监控应用程序的健康状态

13.6. 使用 OPENSHIFT CONTAINER PLATFORM DASHBOARD 获取集群信息

从 OpenShift Container Platform web 控制台点击 **Home > Dashboards > Overview**，访问 OpenShift Container Platform 仪表板，其中包含有关集群的高级信息。

OpenShift Container Platform 仪表板提供各种集群信息，包含在各个仪表板 *卡*。

13.6.1. 关于 OpenShift Container Platform 仪表板页面

OpenShift Container Platform 仪表板由以下各卡组成：

- **Details** 提供有关信息型集群详情的简单概述。
  状态包括 **ok**、**error**、**warning**、**in progress** 和 **unknown**。资源可添加自定义状态名称。
  - 集群 ID
  - 提供者
  - 版本
- **Cluster Inventory** 详细列出资源数目和相关状态。这在通过干预解决问题时非常有用，其中包含以下相关信息：
  - 节点数
  - pod 数量
  - 持久性存储卷声明
  - 虚拟机（如果安装了 OpenShift Virtualization 则可用）
  - 集群中的裸机主机，根据其状态列出（只在 **metal3** 环境中可用）。
- **Cluster Health** 总结了整个集群的当前健康状况，包括相关警报和描述。如果安装了 OpenShift Virtualization，还会诊断 OpenShift virtualization 的整体健康状况。如出现多个子系统，请点击 **See All** 查看每个子系统的状态。
- **Cluster Capacity** 表有助于管理员了解集群何时需要额外资源。此表包含一个内环和一个外环。内环显示当前的消耗，外环显示为资源配置的阈值，其中包括以下信息：
  - CPU 时间
  - 内存分配
  - 所消耗的存储
  - 所消耗的网络资源
- **Cluster Utilization** 显示在指定时间段内各种资源的能力，以帮助管理员了解高资源消耗的范围和频率。
- **Events** 列出了与集群中最近活动相关的消息，如创建 pod 或虚拟机迁移到另一台主机。
- **Top Consumers** 可帮助管理员了解集群资源是如何被消耗的。点一个资源可以进入一个包括详细信息的页面，它列出了对指定集群资源（CPU 、内存或者存储）消耗最多的 Pod 和节点。

13.7. OPENSHIFT CONTAINER PLATFORM 集群监控、日志记录和遥测技术

OpenShift Container Platform 在集群层面提供各种监控资源。

13.7.1. 关于 OpenShift Container Platform 监控

OpenShift Container Platform 包括一个预配置、预安装和自我更新的监控堆栈，用于**监控核心平台组件**。OpenShift Container Platform 提供了与监控相关的现成的最佳实践。其中默认包括一组警报，可立即就集群问题通知集群管理员。OpenShift Container Platform Web 控制台中的默认仪表板包括集群指标的直观表示，以帮助您快速了解集群状态。

安装 OpenShift Container Platform 4.8 后，集群管理员可以选择性地为用户定义的项目启用监控。通过使用此功能，集群管理员、开发人员和其他用户可以指定在其自己的项目中如何监控服务和 Pod。然后，您可以在 OpenShift Container Platform web 控制台中查询指标、查看仪表板，并管理您自己的项目的警报规则和静默。

注意

集群管理员可以授予开发人员和其他用户权限来监控他们自己的项目。通过分配一个预定义的监控角色即可授予权限。

13.7.2. 关于 OpenShift Logging 组件

OpenShift Logging（日志记录）组件包括了一个要部署到 OpenShift Container Platform 集群中每个节点的收集器，用于收集所有节点和容器日志并将其写入日志存储。您可以使用集中 web UI 使用汇总的数据创建丰富的视觉化和仪表板。

OpenShift Logging 的主要组件有：

- collection（收集） - 此组件从集群中收集日志，格式化日志并将其转发到日志存储。当前的实现是 Fluentd。
- log store（日志存储） - 存储日志的位置。默认是 Elasticsearch。您可以使用默认的 Elasticsearch 日志存储，或将日志转发到外部日志存储。默认日志存储经过优化并测试以进行简短存储。
- visualization（可视化） - 此 UI 组件用于查看日志、图形和图表等。当前的实现是 Kibana。

如需有关 OpenShift Logging 的更多信息，请参阅 OpenShift Logging 文档。

13.7.3. 关于 Telemetry

Telemetry 会向红帽发送一组精选的集群监控指标子集。Telemeter 客户端每四分三十秒获取一次指标值，并将数据上传到红帽。本文档中描述了这些指标。

红帽使用这一数据流来实时监控集群，必要时将对影响客户的问题做出反应。它同时还有助于红帽向客户推出 OpenShift Container Platform 升级，以便最大程度降低服务影响，持续改进升级体验。

这类调试信息将提供给红帽支持和工程团队，其访问限制等同于访问通过问题单报告的数据。红帽利用所有连接集群信息来帮助改进 OpenShift Container Platform，提高其易用性。

13.7.3.1. Telemetry 收集的信息

Telemetry 收集以下信息：

- 安装期间生成的唯一随机标识符
- 版本信息，包括 OpenShift Container Platform 集群版本并安装了用于决定更新版本可用性的更新详情
- 更新信息，包括每个集群可用的更新数、用于更新的频道和镜像存储库、更新进度信息以及更新中发生的错误数
- 部署 OpenShift Container Platform 的供应商平台的名称及数据中心位置
- 有关集群、机器类型和机器的大小信息，包括 CPU 内核数和每个机器所使用的 RAM 量

- 集群中正在运行的虚拟机实例的数量
- etcd 成员数和存储在 etcd 集群中的对象数量
- 在集群中安装的 OpenShift Container Platform 框架组件及其状况和状态
- 有关组件、功能和扩展的使用情况信息
- 有关技术预览和不受支持配置的使用详情
- 有关降级软件的信息
- 标记为 **NotReady** 的节点的信息
- 为降级 Operator 列出为 "related objects" 的所有命名空间的事件
- 帮助红帽支持为客户提供有用支持的配置详情，包括云基础架构级别的节点配置、主机名、IP 地址、Kubernetes pod 名称、命名空间和服务
- 有关证书的有效性的信息
- 根据构建策略类型进行应用构建数量

Telemetry 不会收集任何身份识别的信息，如用户名或密码。红帽不会收集个人信息。如果红帽发现个人信息被意外地收到，红帽会删除这些信息。有关红帽隐私实践的更多信息，请参考红帽隐私声明。

13.7.4. CLI 故障排除和调试命令

如需 **oc** 客户端故障排除和调试命令列表，请参阅 OpenShift Container Platform CLI 工具 文档。

13.8. PROMETHEUS 对虚拟资源的查询

OpenShift Virtualization 提供用于监控集群中如何使用基础架构资源的指标。指标涵盖了以下资源：

- vCPU
- Network
- 存储
- 虚拟机内存交换

使用 OpenShift Container Platform 监控仪表板查询虚拟化指标。

13.8.1. 先决条件

- 要使用 vCPU 指标，必须将 **schedstats=enable** 内核参数应用到 **MachineConfig** 对象。此内核参数启用用于调试和性能调优的调度程序统计，并为调度程序添加较小的额外负载。如需有关应用内核参数的更多信息，请参阅 OpenShift Container Platform 机器配置任务 文档。
- 要进行客户机内存交换查询以返回数据，必须在虚拟客户机上启用内存交换。

13.8.2. 查询指标

OpenShift Container Platform 监控仪表板可供您运行 Prometheus Query Language (PromQL) 查询来查看图表中呈现的指标。此功能提供有关集群以及要监控的任何用户定义工作负载的状态信息。

作为**集群管理员**，您可以查询所有 OpenShift Container Platform 核心项目和用户定义的项目的指标。

作为**开发者**，您必须在查询指标时指定项目名称。您必须具有所需权限才能查看所选项目的指标。

13.8.2.1. 以集群管理员身份查询所有项目的指标

作为集群管理员，或具有所有项目的查看权限的用户，您可以在 Metrics UI 中访问所有 OpenShift Container Platform 默认项目和用户定义的项目的指标。

注意

只有集群管理员可以访问 OpenShift Container Platform Monitoring 提供的第三方 UI。

先决条件

- 您可以使用具有 **cluster-admin** 角色或所有项目的查看权限的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 在 OpenShift Container Platform Web 控制台内的 **Administrator** 视角中，选择 **Monitoring → Metrics**。
2. 选择 **Insert Metric at Cursor** 来查看预定义的查询列表。
3. 要创建自定义查询，请将 Prometheus Query Language (PromQL) 查询添加到 **Expression** 字段。
4. 要添加多个查询，选择 **Add Query**。
5. 要删除查询，选择查询旁边的  ，然后选择 **Delete query**。
6. 要禁止运行查询，请选择查询旁边的  并选择 **Disable query**。
7. 选择 **Run Queries** 来运行您已创建的查询。图表中会直观呈现查询的指标。如果查询无效，则 UI 会显示错误消息。

注意

如果查询对大量数据进行运算，这可能会在绘制时序图时造成浏览器超时或过载。要避免这种情况，请选择 **Hide graph** 并且仅使用指标表来校准查询。然后，在找到可行的查询后，启用图表来绘制图形。

8. 可选：页面 URL 现在包含您运行的查询。要在以后再次使用这一组查询，请保存这个 URL。

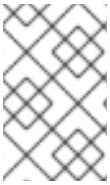
其他资源

- 有关创建 PromQL 查询的更多详情，请参阅 Prometheus query 文档。

13.8.2.2. 以开发者身份查询用户定义的项目的指标

您可以以开发者或具有项目查看权限的用户身份访问用户定义项目的指标。

在 **Developer** 视角中， Metrics UI 包括所选项目的一些预定义 CPU、内存、带宽和网络数据包查询。您还可以对项目的 CPU、内存、带宽、网络数据包和应用程序指标运行自定义 Prometheus Query Language (PromQL) 查询。



注意

开发者只能使用 **Developer** 视角，而不能使用 **Administrator** 视角。作为开发者，您一次只能查询一个项目的指标。开发人员无法访问 OpenShift Container Platform 监控提供的用于核心平台组件的第三方 UI。取而代之，为您的用户定义的项目使用 Metrics UI。

先决条件

- 对于您要查看指标的项目，您可以作为开发者或具有查看权限的用户访问集群。
- 您已为用户定义的项目启用了监控。
- 您已在用户定义的项目中部署了服务。
- 您已为该服务创建了 **ServiceMonitor** 自定义资源定义（CRD），以定义如何监控该服务。

流程

1. 从 OpenShift Container Platform Web 控制台中的 **Developer** 视角，选择 **Monitoring → Metrics**。
2. 在 **Project:** 列表中选择您要查看指标的项目。
3. 从 **Select Query** 列表中选择查询，或者通过选择 **Show PromQL** 运行自定义 PromQL 查询。

注意

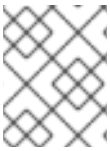
在 **Developer** 视角中，您一次只能运行一个查询。

其他资源

- 有关创建 PromQL 查询的更多详情，请参阅 Prometheus query 文档。

13.8.3. 虚拟化指标

以下指标描述包括示例 Prometheus Query Language（PromQL）查询。这些指标不是 API，可能在不同版本之间有所变化。



注意

以下示例使用 **topk** 查询来指定时间段。如果在那个时间段内删除虚拟机，它们仍然会显示在查询输出中。

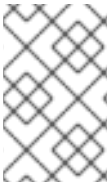
13.8.3.1. vCPU 指标

以下查询可以识别等待输入/输出 (I/O) 的虚拟机：

kubevirt_vmi_vcpu_wait_seconds

返回虚拟机的 vCPU 等待时间（以秒为单位）。

高于"0" 的值表示 vCPU 要运行，但主机调度程序还无法运行它。无法运行代表 I/O 存在问题。



注意

要查询 vCPU 指标，必须首先将 **schedstats=enable** 内核参数应用到 **MachineConfig** 对象。此内核参数启用用于调试和性能调优的调度程序统计，并为调度程序添加较小的额外负载。

vCPU 等待时间查询示例

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_vcpu_wait_seconds[6m]))) > 0 1
```

1 此查询会返回在六分钟内每次都等待 I/O 的 3 台虚拟机。

13.8.3.2. 网络指标

以下查询可以识别正在饱和网络的虚拟机：

kubevirt_vmi_network_receive_bytes_total

返回虚拟机网络中接收的流量总数（以字节为单位）。

kubevirt_vmi_network_transmit_bytes_total

返回虚拟机网络上传输的流量总数（以字节为单位）。

网络流量查询示例

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_network_receive_bytes_total[6m])) + sum by
(name, namespace) (rate(kubevirt_vmi_network_transmit_bytes_total[6m]))) > 0 1
```

1 此查询会返回每给定时间在六分钟内传输最多流量的 3 台虚拟机。

13.8.3.3. 存储指标

13.8.3.3.1. 与存储相关的流量

以下查询可以识别正在写入大量数据的虚拟机：

kubevirt_vmi_storage_read_traffic_bytes_total

返回虚拟机与存储相关的流量的总量（以字节为单位）。

kubevirt_vmi_storage_write_traffic_bytes_total

返回虚拟机与存储相关的流量的存储写入总量（以字节为单位）。

与存储相关的流量查询示例

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_storage_read_traffic_bytes_total[6m])) + sum
by (name, namespace) (rate(kubevirt_vmi_storage_write_traffic_bytes_total[6m]))) > 0 1
```

1 此查询会返回在 6 分钟内每给定时间执行最多存储流量的 3 台虚拟机。

13.8.3.3.2. I/O 性能

以下查询可决定存储设备的 I/O 性能：

kubevirt_vmi_storage_iops_read_total
返回虚拟机每秒执行的写入 I/O 操作量。

kubevirt_vmi_storage_iops_write_total
返回虚拟机每秒执行的读取 I/O 操作量。

I/O 性能查询示例

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_storage_iops_read_total[6m])) + sum by
(name, namespace) (rate(kubevirt_vmi_storage_iops_write_total[6m]))) > 0 1
```

1 此查询返回在六分钟内每给定时间每秒执行最多 I/O 操作数的 3 台虚拟机。

13.8.3.4. 客户机内存交换指标

以下查询可识别启用了交换最多的客户端执行内存交换最多：

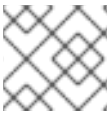
kubevirt_vmi_memory_swap_in_traffic_bytes_total
返回虚拟客户机交换的内存总量（以字节为单位）。

kubevirt_vmi_memory_swap_out_traffic_bytes_total
返回虚拟 guest 正在交换的内存总量（以字节为单位）。

内存交换查询示例

```
topk(3, sum by (name, namespace) (rate(kubevirt_vmi_memory_swap_in_traffic_bytes_total[6m])) +
sum by (name, namespace) (rate(kubevirt_vmi_memory_swap_out_traffic_bytes_total[6m]))) > 0 1
```

1 此查询返回前 3 台虚拟机，其中客户机在六分钟内执行每个给定时间段内每个给定时间最多的内存交换。



注意

内存交换表示虚拟机面临内存压力。增加虚拟机的内存分配可以缓解此问题。

13.8.4. 其他资源

- 监控概述

13.9. 为红帽支持收集数据

当您向红帽支持 提交支持问题单 时，使用以下工具为 OpenShift Container Platform 和 OpenShift Virtualization 提供调试信息会很有帮助：

must-gather 工具

must-gather 工具收集诊断信息，包括资源定义和服务日志。

Prometheus

Prometheus 是一个时间序列数据库和用于指标的规则评估引擎。Prometheus 将警报发送到 Alertmanager 进行处理。

Alertmanager

Alertmanager 服务处理从 Prometheus 接收的警报。Alertmanager 还负责将警报发送到外部通知系统。

13.9.1. 收集有关环境的数据

收集有关环境的数据可最小化分析和确定根本原因所需的时间。

先决条件

- 将 Prometheus 指标数据的保留时间设置为最少 7 天。
- 配置 Alertmanager 以捕获相关的警报并将其发送到专用邮箱，以便可以在集群外部查看和保留它们。
- 记录受影响的节点和虚拟机的确切数量。

流程

1. 使用默认的 **must-gather** 镜像为集群收集 **must-gather** 数据。
2. 如果需要，为 Red Hat OpenShift Container Storage 收集 **must-gather** 数据。
3. 使用 OpenShift Virtualization **must-gather** 镜像收集 OpenShift Virtualization 的 **must-gather** 数据。
4. 收集集群的 Prometheus 指标。

13.9.1.1. 其他资源

- 为 Prometheus 指标数据配置保留时间
- 配置 Alertmanager 将警报通知发送到外部系统
- 为 OpenShift Container Platform 收集 **must-gather** 数据
- 为 OpenShift Virtualization 收集 **must-gather** 数据
- 以集群管理员身份收集所有项目的 Prometheus 指标

13.9.2. 收集虚拟机的数据

收集有关出现故障的虚拟机 (VM) 的数据可最小化分析和确定根本原因所需的时间。

先决条件

- Windows 虚拟机：
  - 记录红帽支持的 Windows 补丁更新详情。

- 安装最新版本的 VirtIO 驱动程序。VirtIO 驱动程序包括 QEMU 客户机代理。
- 如果启用了远程桌面协议 (RDP)，请尝试通过 RDP 连接到虚拟机，以确定是否存在与连接软件相关的问题。

流程

1. 收集故障虚拟机的详细 **must-gather** 数据。
2. 收集在重启前崩溃的虚拟机截图。
3. 记录出现故障的虚拟机通常具有的因素。例如，虚拟机具有相同的主机或网络。

13.9.2.1. 其他资源

- 在 Windows 虚拟机上安装 VirtIO 驱动程序
- 在没有主机访问的 Windows 虚拟机上下载并安装 VirtIO 驱动程序
- 使用 Web 控制台或命令行通过 RDP 连接到 Windows 虚拟机
- 收集有关虚拟机的 **must-gather** 数据

13.9.3. 为 OpenShift Virtualization 使用 must-gather 工具

您可以使用 OpenShift Virtualization 镜像运行 **must-gather** 命令收集有关 OpenShift Virtualization 资源的数据。

默认数据收集包含有关以下资源的信息：

- OpenShift Virtualization Operator 命名空间，包括子对象
- OpenShift Virtualization 自定义资源定义
- 包含虚拟机的命名空间
- 基本虚拟机定义

流程

- 运行以下命令来收集有关 OpenShift Virtualization 的数据：

```
$ oc adm must-gather --image-stream=openshift/must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v{HCOVersion}
```

13.9.3.1. must-gather 工具选项

您可以为以下选项指定脚本和环境变量组合：

- 从命名空间收集详细虚拟机 (VM) 信息
- 收集指定虚拟机的详细信息
- 收集镜像和镜像流信息

- 限制 **must-gather** 工具使用的最大并行进程数

13.9.3.1.1. 参数

环**境**变**量**

您可以为兼容脚本指定环境变量。

NS=<namespace_name>

从您指定的命名空间中收集虚拟机信息，包括 **virt-launcher** pod 详情。为所有命名空间收集 **VirtualMachine** 和 **VirtualMachineInstance** CR 数据。

VM=<vm_name>

收集特定虚拟机的详情。要使用这个选项，还必须使用 **NS** 环境变量指定命名空间。

PROS=<number_of_processes>

修改 **must-gather** 工具使用的最大并行进程数。默认值为 **5**。

重要

使用太多的并行进程可能会导致性能问题。不建议增加并行进程的最大数量。

脚本

每个脚本仅与某些环境变量组合兼容。

gather_vms_details

收集属于 OpenShift Virtualization 资源的虚拟机日志文件、虚拟机定义和命名空间（及其子对象）。如果您在指定命名空间或虚拟机的情况下使用这个参数，**must-gather** 工具会为集群中的所有虚拟机收集这个数据。此脚本与所有环境变量兼容，但是如果使用 **VM** 变量，则必须指定一个命名空间。

gather

使用默认 **must-gather** 脚本，从所有命名空间中收集集群数据，且仅包含基本的虚拟机信息。此脚本只与 **PROS** 变量兼容。

gather_images

收集镜像和镜像流自定义资源信息。此脚本只与 **PROS** 变量兼容。

13.9.3.1.2. 使用和示例

环境变量是可选的。您可以自行运行脚本，也可以使用一个或多个兼容环境变量来运行脚本。

表 13.1. 兼容参数

脚本	兼容环境变量
gather_vms_details	• 对于命名空间：**NS=<namespace_name>** • 对于虚拟机：**VM=<vm_name> NS=<namespace_name>** • **PROS=<number_of_processes>**

脚本	兼容环境变量
gather	• **PROS=<number_of_processes>**
gather_images	• **PROS=<number_of_processes>**

要自定义 **must-gather** 收集的数据，您可以在该命令中附加一个双短划线 (**--**)，后跟一个空格和一个或多个兼容参数。

语**法**

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.8.7 \
  -- <environment_variable_1> <environment_variable_2> <script_name>
```

详细**虚**拟**机信息**

以下命令在 **mynamespace** 命名空间中收集 **my-vm** 虚拟机的详细虚拟机信息：

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.8.7 \
  -- NS=mynamespace VM=my-vm gather_vms_details 1
```

1 如果您使用 **VM** 环境变量，则需要 **NS** 环境变量。

默认**数据收集限制**为**三个并行**进**程**

以下命令使用最多三个并行进程收集默认的 **must-gather** 信息：

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.8.7 \
  -- PROS=3 gather
```

镜**像和**镜**像流信息**

以下命令从集群收集镜像和镜像流信息：

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.8.7 \
  -- gather_images
```

13.9.3.2. **其他**资**源**

- 关于 **must-gather** 工具