OpenShift Container Platform 4.2

容器原生虚拟化

容器原生虚拟化安装、使用与发行注记

Last Updated: 2020-08-21

OpenShift Container Platform 4.2 容器原生虚拟化

容器原生虚拟化安装、使用与发行注记

法律通告

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本文档提供有关如何在 OpenShift Container Platform 4.2 中使用容器原生虚拟化的信息

目录

第 1 章 容器原生虚拟化安装 **4**
1.1. 关于容器原生虚拟化 4
1.2. 为容器原生虚拟化配置集群 4
1.3. 安装容器原生虚拟化 4
1.4. 安装 VIRTCTL 客户端 7
1.5. 升级容器原生虚拟化 7
1.6. 卸载容器原生虚拟化 9

第 2 章 容器原生虚拟化用户指南 **12**
2.1. 创建虚拟机 12
2.2. DATAVOLUME 导入的 TLS 证书 17
2.3. 使用 DATAVOLUME 导入虚拟机镜像 18
2.4. 编辑虚拟机 22
2.5. 删除虚拟机 24
2.6. 控制虚拟机状态 25
2.7. 访问虚拟机控制台 28
2.8. 使用 CLI 工具 32
2.9. 自动执行管理任务 34
2.10. 为虚拟机使用默认 POD 网络 36
2.11. 将虚拟机附加到多个网络 39
2.12. 在虚拟机上安装 QEMU 客户机代理 42
2.13. 在虚拟机上查看 VNIC 的 IP 地址 44
2.14. 为虚拟机配置 PXE 启动 45
2.15. 管理客户机内存 49
2.16. 创建虚拟机模板 51
2.17. 编辑虚拟机模板 54
2.18. 删除虚拟机模板 55
2.19. 将虚拟机磁盘克隆到新 DATAVOLUME 中 56
2.20. 使用 DATAVOLUMETEMPLATE 克隆虚拟机 58
2.21. 使用 VIRTCTL 工具上传本地磁盘镜像 61
2.22. 上传本地磁盘镜像至块存储 DATAVOLUME 63
2.23. 通过添加空白磁盘镜像扩展虚拟存储 66
2.24. 准备 CDI 涂销空间 67
2.25. 使用 DATAVOLUME 将虚拟机镜像导入到块存储 70
2.26. 将虚拟机磁盘克隆到新块存储 DATAVOLUME 中 73
2.27. 虚拟机实时迁移 77
2.28. 实时迁移限制和超时 77
2.29. 迁移虚拟机实例到另一节点 78
2.30. 监控虚拟机实例的实时迁移 79
2.31. 取消虚拟机实例的实时迁移 80
2.32. 配置虚拟机驱除策略 81
2.33. 节点维护模式 81
2.34. 将节点设置为维护模式 82
2.35. 从维护模式恢复节点 83
2.36. 手动刷新 TLS 证书 84
2.37. 在现有 WINDOWS 虚拟机上安装 VIRTIO 驱动程序 85
2.38. 在新 WINDOWS 虚拟机上安装 VIRTIO 驱动程序 87
2.39. 查看虚拟机日志 90
2.40. 使用 OPENSHIFT CONTAINER PLATFORM 仪表板获取集群信息 91
2.41. OPENSHIFT CONTAINER PLATFORM 集群监控、日志记录和遥测技术 92
2.42. 为红帽支持收集容器原生虚拟化数据 94

第 3 章 容器原生虚拟化 2.1 发行注记 **96**
3.1. 容器原生虚拟化 2.1 发行注记 96

第 1 章 容器原生虚拟化安装

1.1. 关于容器原生虚拟化

了解容器原生虚拟化的功能与支持范围。

1.1.1. 容器原生虚拟化的作用

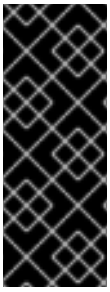
容器原生虚拟化（container-native virtualization）是 OpenShift Container Platform 的一个附加组件，可用于运行和管理虚拟机工作负载以及容器工作负载。

容器原生虚拟化通过 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。

1.1.2. 容器原生虚拟化支持



重要

容器原生虚拟化仅是一项技术预览功能。技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。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，请参阅 https://access.redhat.com/support/offerings/techpreview/。

1.2. 为容器原生虚拟化配置集群

要获取 OpenShift Container Platform 的评估版本，请从 OpenShift Container Platform 主页下载一个试用版。

在默认情况下，容器原生虚拟化可以与 OpenShift Container Platform 协同工作，但推荐进行以下安装配置：

- OpenShift Container Platform 集群安装在裸机上。根据集群中要托管的虚拟机的数量和大小来管理您的计算节点。
- 在集群中配置 Monitoring（监控）。

1.3. 安装容器原生虚拟化

安装容器原生虚拟化以便在 OpenShift Container Platform 集群中添加虚拟化功能。

您可以使用 OpenShift Container Platform 4.2 web 控制台来订阅和部署容器原生虚拟化 Operator。

先决条件

- OpenShift Container Platform 4.2

重要

容器原生虚拟化仅是一项技术预览功能。技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。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，请参阅 https://access.redhat.com/support/offerings/techpreview/。

1.3.1. 准备安装容器原生虚拟化

安装容器原生虚拟化之前，请创建一个名为 **openshift-cnv** 的命名空间。

先决条件

- 用户具有 **cluster-admin** 特权

流程

1. 在 OpenShift Container Platform Web 控制台中，导航至 **Administration → Namespaces** 页面。
2. 点 **Create Namespace**。
3. 在 **Name** 字段中输入 **openshift-cnv**。
4. 点击 **Create**。

1.3.1.1. 订阅 KubeVirt HyperConverged Cluster Operator 目录

安装容器原生虚拟化之前，请先从 OpenShift Container Platform web 控制台订阅 **KubeVirt HyperConverged Cluster Operator** 目录。订阅会授予 **openshift-cnv** 命名空间对容器原生虚拟化 Operator 的访问权限。

先决条件

- 创建名为 **openshift-cnv** 的命名空间。

流程

1. 打开浏览器窗口并登录 OpenShift Container Platform web 控制台。
2. 导航到 **Operators → OperatorHub** 页面。
3. 找到 **KubeVirt HyperConverged Cluster Operator**，并选中。
4. 阅读 Operator 的信息并点击 **Install**。

5. 在 Create Operator Subscription 页面：

   a. 从 Installation Mode 列表中选择 A specific namespace on the cluster，然后选择 **openshift-cnv** 命名空间。

警告

- All namespaces on the cluster (default) 选择该选项会将 Operator 安装至默认 **openshift-operators** 命名空间，以便供集群中的所有命名空间监视和使用。此选项**不**支持与容器原生虚拟化一起使用。您需要在 **openshift-cnv** 命名空间中安装 Operator。

   b. 从可用 Update Channel 选项列表中选择 2.1。

   c. 对于 Approval Strategy，请确保已选择默认值 Automatic。当有新 z-stream 发行版可用时，容器原生虚拟化将自动更新。

6. 单击 Subscribe，以便该 Operator 可供 OpenShift Container Platform 集群上的所选命名空间使用。

1.3.2. 部署容器原生虚拟化

订阅 KubeVirt HyperConverged Cluster Operator 目录后，请创建 KubeVirt HyperConverged Cluster Operator Deployment 自定义资源来部署容器原生虚拟化。

先决条件

- 对 **openshift-cnv** 命名空间中的 KubeVirt HyperConverged Cluster Operator 目录具有有效订阅

流程

1. 导航到 Operators → Installed Operators 页面。
2. 点击 KubeVirt HyperConverged Cluster Operator。
3. 点击 KubeVirt HyperConverged Cluster Operator Deployment 选项卡，然后点击 Create HyperConverged。

   a. 在点 Create HyperConverged后，会显示 YAML 文件。删除 **'false'**的单引号。这是解决 BZ#1767167 里报告的问题的一个临时解决方案。
   最初显示时，YAML 文件类似以下示例：

```
apiVersion: hco.kubevirt.io/v1alpha1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  BareMetalPlatform: 'false' 1
```

1 请确定在进行下一步前此行会读取 **BareMetalPlatform: false**。

4. 点击 **Create** 以启动容器原生虚拟化。
5. 导航到 **Workloads → Pods** 页面，并监控容器原生虚拟化 Pod，直至全部处于 **Running** 状态。在所有 Pod 均处于 **Running** 状态后，您即可访问容器原生虚拟化。

1.4. 安装 VIRTCTL 客户端

virtctl 客户端是用于管理容器原生虚拟化资源的命令行实用程序。

通过启用容器原生虚拟化存储库并安装 **kubevirt-virtctl** 软件包，将客户端安装到您的系统中。

1.4.1. 启用容器原生虚拟化存储库

红帽为 Red Hat Enterprise Linux 8 和 Red Hat Enterprise Linux 7 提供容器原生虚拟化存储库：

- Red Hat Enterprise Linux 8 存储库：**cnv-2.1-for-rhel-8-x86_64-rpms**
- Red Hat Enterprise Linux 7 存储库：**rhel-7-server-cnv-2.1-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 **subscription manager** 为您的系统启用适当的容器原生虚拟化存储库：

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

1.4.2. 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
```

另请参阅：使用 CLI 工具进行容器原生虚拟化。

1.5. 升级容器原生虚拟化

您可在容器原生虚拟化安装过程中启用自动更新。了解预期状况以及如何查看进行中的更新的状态。

重要

容器原生虚拟化仅是一项技术预览功能。技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。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。

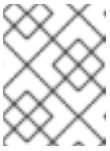
有关红帽技术预览功能支持范围的详情，请参阅 https://access.redhat.com/support/offerings/techpreview/。

1.5.1. 关于升级容器原生虚拟化

如果您在安装容器原生虚拟化时启用了自动更新，您会在更新可用时收到更新。

其他信息

- 在容器原生虚拟化 2.1 中，只有 *z-stream* 更新可用。例如，容器原生虚拟化 2.1.0 → 容器原生虚拟化 2.1.1
- 更新通过 *Marketplace Operator* 传送，它在 OpenShift Container Platform 安装过程中部署。Marketplace Operator 为您的集群提供外部 Operator。
- 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。
- DataVolume 及其关联 PersistentVolumeClaim 会在升级过程中保留。
- 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。

1.5.2. 监控升级状态

监控容器原生虚拟化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

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 安装 OpenShift 命令行界面 (CLI)，通常称为 **oc**。

流程

1. 运行以下命令：

```
$ oc get csv
```

2. 查看输出，检查 **PHASE** 字段。例如：

```
VERSION  REPLACES                                   PHASE
2.1.1    kubevirt-hyperconverged-operator.v2.1.0    Installing
2.1.0                                               Replacing
```

3. 可选：运行以下命令来监控所有容器原生虚拟化组件状况的聚合状态：

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

成功升级后会输出以下内容：

```
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
```

其他信息

- ClusterServiceVersion (CSV)

1.6. 卸载容器原生虚拟化

您可使用 OpenShift Container Platform web 控制台来卸载容器原生虚拟化。

先决条件

- 容器原生虚拟化 2.1

1.6.1. 删除 KubeVirt HyperConverged 自定义资源

要卸载容器原生虚拟化，首先需要删除 **KubeVirt HyperConverged Cluster Operator Deployment** 自定义资源。

先决条件

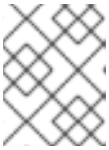
- 具有活跃的 **KubeVirt HyperConverged Cluster Operator Deployment** 自定义资源

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 **KubeVirt HyperConverged Cluster Operator**。
4. 点击 **KubeVirt HyperConverged Cluster Operator Deployment** 选项卡。

5. 点击 Options 菜单  （在包含 **hyperconverged-cluster** 自定义资源的一行中）。在展开的菜单中，点击 **Delete HyperConverged**。
6. 在确认窗口中点击 **Delete**。
7. 导航到 **Workloads → Pods** 页面，验证是否只有 Operator Pod 正在运行。
8. 运行以下命令，打开终端窗口并清理剩余的 KubeVirt 资源：

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



注意

因为某些 KubeVirt 资源当前未正确保留，所以您必须手动将其移除。这些资源将在 BZ1712429 解决后自动移除。

1.6.2. 删除 KubeVirt HyperConverged Cluster Operator 目录订阅

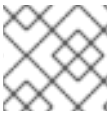
要完成卸载容器原生虚拟化，请卸载 **KubeVirt HyperConverged Cluster Operator** 订阅。

先决条件

- 活跃的 **KubeVirt HyperConverged Cluster Operator** 目录订阅

流程

1. 导航到 **Catalog → OperatorHub** 页面。
2. 找到 **KubeVirt HyperConverged Cluster Operator**，并选中。
3. 点击 **Uninstall**。



注意

现在可删除 **openshift-cnv** 命名空间。

1.6.3. 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**。

第 2 章 容器原生虚拟化用户指南

2.1. 创建虚拟机

使用以下其中一个流程来创建虚拟机：

- 运行虚拟机向导
- 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

2.1.1. 运行虚拟机向导来创建虚拟机

web 控制台具有一个交互式向导，指导您浏览 **Basic Settings**、**Networking** 和 **Storage** 屏幕，以简化虚拟机的创建流程。所有必填字段均标有 *。填写完所有必填字段后，向导才会移至下一屏幕。

可创建 NIC 和存储磁盘，并在创建后将其附加到虚拟机。

Bootable Disk

如果在 **Basic Settings** 屏幕中将 **URL** 或 **Container** 选为 **Provision Source**，则会创建一个 **rootdisk** 磁盘，并将其作为 **Bootable Disk** 附加到虚拟机。您可修改 **rootdisk**，但不可将其移除。

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**。

先决条件

- 在使用向导创建虚拟机时，您的虚拟机存储介质必须支持 Read-Write-Many (RWM) PVC。

流程

1. 从侧边菜单中选择 **Workloads → Virtual Machines**。
2. 点击 **Create Virtual Machine** 并选择 **Create with Wizard**。
3. 填写所有必填 **Basic Settings**。选择一个 **Template** 来自动填写这些字段。
4. 点击 **Next** 进入 **Networking** 屏幕。默认附加 **nic0** NIC。
   a. （可选）点击 **Create NIC** 创建额外 NIC。
   b. （可选）您可通过点击 ⋮ 按钮并选择 **Remove NIC** 来移除任何或所有 NIC。虚拟机无需附加 NIC 也可创建。可在创建虚拟机之后创建 NIC。
5. 点击 **Next** 进入 **Storage** 屏幕。
   a. （可选）点击 **Create Disk** 创建额外磁盘。可通过点击 ⋮ 按钮并选择 **Remove Disk** 来移除这些磁盘。
   b. （可选）点击磁盘修改可用字段。点击 ✓ 按钮保存更新。
   c. （可选）点击 **Attach Disk**，从 **Select Storage** 下拉列表中选择可用磁盘。

6. 点击 **Create Virtual Machine** 。**Results** 屏幕显示虚拟机的 JSON 配置文件。

虚拟机列于 **Workloads → Virtual Machines** 中。

运行 web 控制台向导时，请参考虚拟机向导字段部分。

2.1.1.1. 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。
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。
Provision Source	PXE	从 PXE 菜单置备虚拟机。集群中需要支持 PXE 的 NIC。
	URL	从由 **HTTP** 或 **S3** 端点提供的镜像置备虚拟机。
	Container	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***。
	Cloned Disk	置备源是一个克隆的磁盘。
	Import	从所支持的提供程序导入虚拟机。
Operating System		集群中可用操作系统列表。这是虚拟机的主要操作系统。如果将 **Import** 选为 **Provider Source**，则操作系统将基于待导入 Vmware 虚拟机的操作系统自动填写。
Flavor	small、medium、large、tiny、Custom	预设值，用于决定分配给虚拟机的 CPU 和内存量。
Workload Profile	desktop	用于桌面的虚拟机配置。
	generic	可平衡各种工作负载的性能和兼容性的虚拟机配置。

名称	参数	描述
	high performance	针对高性能负载进行了优化的虚拟机配置。
Start virtual machine on creation		选择此项可在创建时自动启动虚拟机。
Use cloud-init		选择此项可启用 cloud-init 字段。

2.1.1.2. Cloud-init 字段

名称	描述
Hostname	为虚拟机设置具体主机名。
Authenticated SSH Keys	复制到虚拟机上 *~/.ssh/authorized_keys* 的用户公钥。
Use custom script	将其他选项替换为您粘贴自定义 cloud-init 脚本的字段。

2.1.1.3. 网络字段

名称	描述
Create NIC	为虚拟机创建新 NIC。
NIC NAME	NIC 的名称。
MAC ADDRESS	网络接口的 MAC 地址。如果未指定 MAC 地址，将为会话生成一个临时地址。
NETWORK CONFIGURATION	可用 NetworkAttachmentDefinition 对象列表。
BINDING METHOD	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。
PXE NIC	支持 PXE 的网络列表。只有在将 **PXE** 选为 **Provision Source** 时才会显示。

2.1.1.4. 存储字段

名称	描述
Create Disk	为虚拟机创建新磁盘。
Attach Disk	从可用 PVC 列表中选择一个现有磁盘，以附加到虚拟机。
DISK NAME	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 (GB)	磁盘大小（以 GB 为单位）。
STORAGE CLASS	底层 **StorageClass** 的名称。
Bootable Disk	虚拟机将从中启动的可用磁盘列表。如果虚拟机的 Provision Source 为 **URL** 或 **Container**，该字段将锁定为 **rootdisk**。

2.1.2. 粘贴至预先配置的 YAML 文件中以创建虚拟机

通过在 web 控制台的 **Workloads → Virtual Machines** 屏幕中写入或粘贴 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. 从侧边菜单中选择 **Workloads → Virtual Machines**。
2. 点击 **Create Virtual Machine** 并选择 **Create from YAML**。
3. 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。
4. （可选）点击 **Download** 以下载当前状态下的 YAML 配置文件。
5. 点击 **Create** 以创建虚拟机。

虚拟机列于 **Workloads → Virtual Machines** 中。

2.1.3. 使用 CLI 创建虚拟机

流程

VirtualMachine 配置文件的 **spec** 对象会引用虚拟机设置，如内核数、内存量、磁盘类型以及要使用的卷。

1. 通过引用相关 PVC **claimName** 作为卷，将虚拟机磁盘附加到虚拟机。
2. 要利用 OpenShift Container Platform 客户端创建虚拟机，请运行此命令：

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

3. 由于虚拟机创建时处于 **Stopped** 状态，因此需启动虚拟机来运行虚拟机实例。

注意

ReplicaSet 的目的通常是用于保证有指定数量的相同 Pod 可用。容器原生虚拟化当前不支持 ReplicaSet。

表 2.1. 域设置

设置	描述
内核	虚拟机中的内核数。必须大于或等于 1。
内存	按节点分配给虚拟机的 RAM 量。指定一个值，以 **M**（兆字节）或 **Gi**（千兆字节）为单位。
磁盘：名称	所引用卷的名称。必须与卷的名称匹配。

表 2.2. 卷设置

设置	描述
名称	卷的名称，必须是 DNS 标签，且在虚拟机中唯一。
PersistentVolumeClaim	附加到虚拟机的 PVC。PVC 的 **claimName** 必须与虚拟机处于同一个项目中。

2.1.4. 虚拟机存储卷类型

虚拟机存储卷类型以及域和卷设置均已列出。有关虚拟机设置的确定性列表，请参阅 kubevirt API 引用。

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 DataVolume。使用此卷类型的虚拟机可保证在卷就绪前不会启动。
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。
containerDisk	引用容器镜像 registry 中存储的镜像，如虚拟机磁盘。该镜像拉取自 registry，在创建虚拟机时嵌入卷中。**containerDisk** 卷为临时卷，将在虚拟机停止、重启或删除时丢弃。 容器磁盘不限于单个虚拟机，对于创建大量无需持久性存储的虚拟机克隆来说非常有用。 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。
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。

有关虚拟机设置的确定性列表，请参阅 kubevirt API 引用。

2.2. DATAVOLUME 导入的 TLS 证书

2.2.1. 添加用于身份验证 DataVolume 导入的 TLS 证书

registry 或 HTTPS 端点的 TLS 证书必须添加到 ConfigMap 中才可从这些源导入数据。该 ConfigMap 必须存在于目标 DataVolume 的命名空间中。

通过引用 TLS 证书的相对文件路径来创建 ConfigMap。

流程

1. 确定您处于正确的命名空间中。ConfigMap 只有位于相同命名空间中才可被 DataVolume 引用。

```
$ oc get ns
```

2. 创建 ConfigMap：

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

2.2.2. 示例：从 TLS 证书创建的 ConfigMap

以下示例是从 **ca.pem** TLS 证书创建的 ConfigMap。

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

```
-----BEGIN CERTIFICATE-----
... <base64 encoded cert> ...
-----END CERTIFICATE-----
```

2.3. 使用 DATAVOLUME 导入虚拟机镜像

您可将现有虚拟机镜像导入到您的 OpenShift Container Platform 集群中。容器原生虚拟化使用 DataVolume 自动导入数据并创建底层 PersistentVolumeClaim (PVC)。

小心

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。详情请参阅操作系统文档。

先决条件

- 如果端点需要 TLS 证书，该证书必须包含在与 DataVolume 位于相同命名空间的 ConfigMap 中，并在 DataVolume 配置中被引用。
- 您可能需要定义一个 StorageClass 或准备 CDI 涂销空间 才能成功完成此操作。

2.3.1. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	☐ TAR	☐ TAR

✓ 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

+ 存档不支持块模式 DV

2.3.2. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.3.3. 使用 DataVolume 将虚拟机镜像导入到对象中

要从所导入的镜像创建虚拟机，请在创建虚拟机前在 **VirtualMachine** 配置文件中指定镜像位置。

先决条件

- 安装 OpenShift Container Platform 命令行界面 (CLI)，通常称 **oc**。
- 具有 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** 端点，以及访问数据源所需的任何身份验证凭证
- 至少一个可用 PersistentVolume

流程

1. 确定用于托管您要导入的虚拟磁盘镜像的 **HTTP** 文件服务器。您需要一个正确格式的完整 URL：
   - **http://www.example.com/path/to/data**
2. 如果您的数据源需要身份验证凭证，请编辑 **endpoint-secret.yaml** 文件，并在集群中应用更新的配置：

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

(1) 可选：您的密钥或用户名，base64 编码

(2) 可选：您的 secret 或密码，base64 编码

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

3. 编辑虚拟机配置文件，为您要导入的镜像指定数据源。在本例中，导入了一个 Fedora 镜像：

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  creationTimestamp: null
  labels:
    kubevirt.io/vm: vm-fedora-datavolume
  name: vm-fedora-datavolume
spec:
```

```
  dataVolumeTemplates:
  - metadata:
      creationTimestamp: null
      name: fedora-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 2Gi
        storageClassName: local
      source:
        http:
          url:
https://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora
-Cloud-Base-28-1.1.x86_64.qcow2 (1)
          secretRef: "" (2)
          certConfigMap: "" (3)
    status: {}
  running: false
  template:
    metadata:
      creationTimestamp: null
      labels:
        kubevirt.io/vm: vm-fedora-datavolume
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: datavolumedisk1
        machine:
          type: ""
        resources:
          requests:
            memory: 64M
      terminationGracePeriodSeconds: 0
      volumes:
      - dataVolume:
          name: fedora-dv
        name: datavolumedisk1
status: {}
```

(1) 您要导入的镜像的 **HTTP** 源。

(2) **secretRef** 参数是可选的。

(3) 仅在端点需要身份验证时才需要 **certConfigMap**。所引用的 ConfigMap 必须与 DataVolume 位于相同命名空间中。

4. 创建虚拟机：

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

-

注意

oc create 命令创建 DataVolume 和虚拟机。CDI 控制器使用正确注解创建底层 PVC，导入过程随即开始。导入完成后，DataVolume 状态变为 **Succeeded**，虚拟机可以启动。

DataVolume 置备在后台进行，因此无需监控。您可以启动虚拟机，该虚拟机将在导入完成后才会运行。

可选验证步骤

1. 运行 **oc get pods** 并查找导入程序 Pod。该 Pod 会从指定的 URL 下载镜像，并将其存储在置备的 PV 上。
2. 监控 DataVolume 的状态，直至状态显示为 **Succeeded**。

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

1 在虚拟机配置文件中指定的 DataVolume 的数据标签。

3. 要验证置备是否已完成以及 VMI 是否已启动，请尝试访问其串行控制台：

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

2.3.4. 模板：DataVolume 虚拟机配置文件

example-dv-vm.yaml

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: example-vm
  name: example-vm
spec:
  dataVolumeTemplates:
  - metadata:
      name: example-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 1G
      source:
          http:
            url: "" 1
  running: false
  template:
    metadata:
      labels:
```

```
      kubevirt.io/vm: example-vm
  spec:
    domain:
      cpu:
        cores: 1
      devices:
        disks:
        - disk:
            bus: virtio
          name: example-dv-disk
      machine:
        type: q35
      resources:
        requests:
          memory: 1G
    terminationGracePeriodSeconds: 0
    volumes:
    - dataVolume:
        name: example-dv
      name: example-dv-disk
```

1 您要导入的镜像的 **HTTP** 源（如适用）。

2.3.5. 模板：DataVolume 导入配置文件

example-import-dv.yaml

```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: "example-import-dv"
spec:
  source:
      http:
        url: "" 1
        secretRef: "" 2
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: "1G"
```

1 您要导入的镜像的 **HTTP** 源。

2 **secretRef** 参数是可选的。

2.4. 编辑虚拟机

通过完成以下任一任务来编辑虚拟机：

- 使用 web 控制台编辑虚拟机 YAML 配置

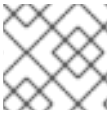
- 使用 CLI 编辑虚拟机 YAML 配置

2.4.1. 使用 web 控制台编辑虚拟机 YAML 配置

使用 web 控制台编辑虚拟机的 YAML 配置。

并非所有参数均可更新。如果您编辑无法更改的值并点击 **Save**，则错误消息会指示参数无法更新。

虚拟机处于 **Running** 状态时可编辑 YAML 配置，但只有在停止并重新启动虚拟机后，更改才会生效。



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. 从侧边菜单中选择 **Workloads → Virtual Machine**。
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。

2.4.2. 使用 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>
```

2.4.3. 将虚拟磁盘添加到虚拟机

为虚拟机添加一个磁盘

流程

1. 在 **Virtual Machines** 选项卡中选择您的虚拟机。
2. 选择 **Disks** 选项卡。
3. 点击 **Add Disks** 打开 **Add Disk** 窗口。
4. 在 **Add Disk** 窗口中，指定 **Source**、**Name**、**Size**、**Interface** 和 **Storage Class**。
5. 使用下拉列表和复选框编辑磁盘配置。
6. 点击 **OK**。

2.4.4. 将网络接口添加到虚拟机

流程

1. 在 **Virtual Machines** 选项卡中选择虚拟机。
2. 选择 **Network Interfaces** 选项卡。
3. 点 **Create Network Interface**。
4. 在 **Create Network Interface** 列表行中，指定网络接口的 **Name**、**Model**、**Network**、**Type** 和 **MAC Address**。
5. 点击行右边的绿色复选框来添加网络接口。
6. 重启虚拟机以启用访问。
7. 编辑下拉列表和复选框来配置网络接口。
8. 点击 **Save Changes**。
9. 点击 **OK**。

新网络接口显示在 **Create Network Interfac** 列表的顶部，直到用户重启。

新网络接口有一个 **Pending VM restart** 链接状态，直到您重启虚拟机。将鼠标悬停在链接状态上方可显示更详细的信息。

当在虚拟机上定义了网络接口卡并连接到网络时，**Link State** 会被默认设置为 **Up**。

2.5. 删除虚拟机

使用以下其中一个流程来删除虚拟机：

- 使用 Web 控制台
- 使用 CLI

2.5.1. 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。

在 **Workloads → Virtual Machines** 列表中使用虚拟机的 ⋮ 按钮删除虚拟机，或使用 **Virtual Machine Details** 屏幕中的 **Actions** 控制键删除虚拟机。

流程

1. 在容器原生虚拟化控制台中，从侧边菜单中点击 **Workloads → Virtual Machines**。
2. 点击待删除虚拟机的 ⋮ 按钮，然后选择 **Delete Virtual Machine**。
   - 或者，点击虚拟机名称，打开 **Virtual Machine Details** 屏幕，然后点击 **Actions → Delete Virtual Machine**。
3. 在确认弹出窗口中，点击 **Delete** 永久删除虚拟机。

2.5.2. 使用 CLI 删除虚拟机及其 DataVolume

当您删除虚拟机时，其使用的 DataVolume 不会被自动删除。

建议删除 DataVolume 以便保持干净的环境并避免可能的混乱。

流程

运行这些命令来删除虚拟机和 PVC。

注意

除非指定 **-n <project_name>** 选项，否则您仅可在当前正在使用的项目中删除对象。

1. 运行以下命令以删除虚拟机：

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

2. 运行以下命令以删除 DataVolume：

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

2.6. 控制虚拟机状态

借助容器原生虚拟化，您既可从 web 控制台也可从命令行界面 (CLI) 来停止、启动和重启虚拟机。

2.6.1. 从 web 控制台控制虚拟机

您还可从 web 控制台来停止、启动和重启虚拟机。

2.6.1.1. 启动虚拟机

您可从 web 控制台启动虚拟机。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 从此屏幕启动虚拟机，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 Details** 屏幕，其中可查看所选虚拟机的综合详情：
   - 点击 Options 菜单  （在虚拟机后面），然后选择 **Start Virtual Machine**。
   - 点击虚拟机名称，打开 **Virtual Machine Details** 屏幕，然后点击 **Actions**，并选择 **Start Virtual Machine**。
3. 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，虚拟机将处于 **Importing** 状态，容器原生虚拟化会从 URL 端点导入容器。根据镜像大小，该过程可能需要几分钟时间。

2.6.1.2. 重启虚拟机

您可从 web 控制台重启正在运行的虚拟机。

重要

不要重启状态为 **Importing** 的虚拟机。重启虚拟机会导致其错误。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 从此屏幕重启虚拟机，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 Details** 屏幕，其中可查看所选虚拟机的综合详情：
   - 点击 Options 菜单  （在虚拟机后面），然后选择 **Restart Virtual Machine**。
   - 点击虚拟机名称，打开 **Virtual Machine Details** 屏幕，然后点击 **Actions**，并选择 **Restart Virtual Machine**。
3. 在确认窗口中，点击 **Restart** 重启虚拟机。

2.6.1.3. 停止虚拟机

您可从 web 控制台停止虚拟机。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 从此屏幕停止虚拟机，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 Details** 屏幕，其中可查看所选虚拟机的综合详情：

- 点击 Options 菜单  （在虚拟机后面），然后选择 **Stop Virtual Machine**。
- 点击虚拟机名称，打开 **Virtual Machine Details** 屏幕，然后点击 **Actions**，并选择 **Stop Virtual Machine**。

3. 在确认窗口中，点击 **Stop** 停止虚拟机。

2.6.2. 控制虚拟机的 CLI 参考

使用以下 **virtctl** 客户端实用程序和 **oc** 命令来更改虚拟机状态，并显示虚拟机列表以及代表虚拟机的虚拟机实例。

注意

运行 **virtctl** 命令可修改虚拟机本身，而非 web 控制台中代表虚拟机的虚拟机实例。

2.6.2.1. 开始

启动虚拟机。

示例：启动当前项目中的虚拟机

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

示例：启动特定项目中的虚拟机

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

2.6.2.2. 重启

重启正在运行的虚拟机。

示例：重启当前项目中的虚拟机

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

示例：重启特定项目中的虚拟机

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

2.6.2.3. 停止

停止正在运行的虚拟机。

示例：停止当前项目中的虚拟机

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

示例：停止特定项目中的虚拟机

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

2.6.2.4. 列表

列出项目中的虚拟机或虚拟机实例。虚拟机实例是指代表虚拟机本身的抽象。

示例：列出当前项目中的虚拟机

```
$ oc get vm
```

示例：列出特定项目中的虚拟机

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

示例：列出当前项目中正在运行的虚拟机实例

```
$ oc get vmi
```

示例：列出特定项目中正在运行的虚拟机实例

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

2.7. 访问虚拟机控制台

容器原生虚拟化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。您可通过 web 控制台并使用 CLI 命令来访问这些控制台。

2.7.1. 虚拟机控制台会话

您可从 web 控制台上 **Virtual Machine Details** 屏幕中的 **Consoles** 选项卡连接至正在运行的虚拟机的 VNC 控制台和串行控制台。

有两个控制台可用：图形 **VNC Console** 和 **Serial Console**。每次导航到 **Consoles** 选项卡时会默认打开 **VNC Console**。您可使用 **VNC Console Serial Console** 列表来切换这两种控制台。

控制台会话将在后台保持活跃，除非断开连接。当 **Disconnect before switching** 复选框活跃且您切换了控制台时，当前控制台会话将断开连接，与选定控制台的新会话将连接至虚拟机。这样可保证一次仅打开一个控制台会话。

VNC Console 的选项

Send Key 按钮列出了要发送至虚拟机的密钥组合。

Serial Console 的选项

使用 **Disconnect** 按钮可手动断开 **Serial Console** 会话与虚拟机的连接。
使用 **Reconnect** 按钮可手动打开 **Serial Console** 会话与虚拟机的连接。

2.7.2. 使用 web 控制台连接至虚拟机

2.7.2.1. 连接至终端

您可使用 web 控制台连接至虚拟机。

流程

1. 确定您处于正确的项目中。如果不是，则点击 **Project** 列表，然后选择适当项目。
2. 点击 **Workloads → Virtual Machines** 以显示该项目中的虚拟机。
3. 选择虚拟机。
4. 进入 **Overview** 选项卡中，点击 **virt-launcher-<vm-name>** Pod。
5. 点击 **Terminal** 选项卡。如果终端空白，则选择终端，然后按任意键启动连接。

2.7.2.2. 连接至串行控制台

从 web 控制台上 **Virtual Machine Details** 屏幕中的 **Consoles** 选项卡连接至正在运行的虚拟机的 **Serial Console**。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 选择虚拟机。
3. 点击 **Consoles**。默认会打开 VNC 控制台。
4. 点击 **VNC Console** 下拉菜单并选择 **Serial Console**。

2.7.2.3. 连接至 VNC 控制台

从 web 控制台上 **Virtual Machine Details** 屏幕中的 **Consoles** 选项卡连接至正在运行的虚拟机的 VNC 控制台。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 选择虚拟机。
3. 点击 **Consoles**。默认会打开 VNC 控制台。

2.7.2.4. 连接至 RDP 控制台

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. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 选择 Windows 虚拟机。
3. 点击 **Consoles** 选项卡。
4. 点击 **Consoles** 列表并选择 **Desktop Viewer**。
5. 在 **Network Interface** 列表中，选择第 2 层 vNIC。
6. 点击 **Launch Remote Desktop** 下载 **console.rdp** 文件。
7. 打开 RDP 客户端并引用 **console.rdp** 文件。例如，使用 **Remmina**：

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

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

2.7.3. 使用 CLI 命令访问虚拟机控制台

2.7.3.1. 通过 SSH 访问虚拟机实例

您在虚拟机上公开 22 号端口后，即可使用 SSH 来访问虚拟机。

使用 **virtctl expose** 命令可将虚拟机实例端口转发至节点端口，并为启用的访问创建服务。以下示例创建 **fedora-vm-ssh** 服务，该服务将 **<fedora-vm>** 虚拟机的 22 号端口转发至节点上的端口：

先决条件

- 您要访问的虚拟机实例必须使用 **masquerade** 绑定方法连接至默认的 Pod 网络。
- 您要访问的虚拟机实例必须正在运行。
- 安装 OpenShift 命令行界面 (CLI)，通常称为 **oc**。

流程

1. 运行以下命令来创建 **fedora-vm-ssh** 服务：

```
$ virtctl expose vm <fedora-vm> --port=20022 --target-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
NAME          TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
fedora-vm-ssh NodePort   127.0.0.1     <none>        20022:32551/TCP  6s
```

在本例中，服务获取了 **32551** 端口。

3. 通过 SSH 登录虚拟机实例。使用节点的 **ipAddress** 以及您在上一步中发现的端口：

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

2.7.3.2. 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>
```

2.7.3.3. 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
```

2.7.3.4. 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**。
- 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. 以具有访问令牌的用户身份通过 **oc** CLI 工具登录容器原生虚拟化集群。

```
$ 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>
```

```
...
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。

2.8. 使用 CLI 工具

用于管理集群中资源的两个主要 CLI 工具是：

- 容器原生虚拟化 **virtctl** 客户端
- OpenShift Container Platform **oc** 客户端

先决条件

- 必须安装 **virtctl** 客户端。

2.8.1. Virtctl 客户端命令

virtctl 客户端是用于管理容器原生虚拟化资源的命令行实用程序。下表包含整个容器原生虚拟化文档中使用的 **virtctl** 命令。

表 2.3. virtctl 客户端命令

命令	描述
virtctl start<vm>	启动虚拟机。
virtctl stop<vm>	停止虚拟机。
virtctl restart<vm>	重启虚拟机。
virtctl expose<vm>	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。
virtctl console <vmi>	连接至虚拟机实例的串行控制台。
virtctl vnc <vmi>	打开虚拟机实例的 VNC 连接。
virtctl image-upload <...>	上传虚拟机镜像至 PersistentVolumeClaim。

2.8.2. OpenShift Container Platform 客户端命令

OpenShift Container Platform **oc** 客户端是用于管理 OpenShift Container Platform 资源的命令行实用程序。下表包含整个容器原生虚拟化文档中使用的 **oc** 命令。

表 2.4. 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>	显示项目中指定对象类型的对象列表。

命令	描述
oc describe <object_type> <resource_name>	显示项目中指定资源的详情。
oc create -f <object_config>	从文件名或从 stdin 在项目中创建资源。
oc edit <object_type> <resource_name>	编辑项目中的资源。
oc delete <object_type> <resource_name>	删除项目中的资源。

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。

2.9. 自动执行管理任务

您可以使用 Red Hat Ansible Automation Platform 自动完成容器原生虚拟化管理任务。通过使用 Ansible Playbook 创建新虚拟机来了解基础知识。

2.9.1. 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 包含对容器原生虚拟化的支持，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容器原生虚拟化管理，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。

2.9.2. 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：

```
(...)
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。

2.9.3. 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。

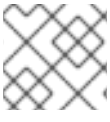
```
---
- name: Ansible Playbook 1
  hosts: localhost
  connection: local
  tasks:
    - name: Create my first VM
      kubevirt_vm:
        namespace: default
        name: vm1
        cpu_cores: 1
        memory: 64Mi
        disks:
          - name: containerdisk
            volume:
              containerDisk:
                image: kubevirt/cirros-container-disk-demo:latest
            disk:
              bus: virtio
```

其他信息

- Playbook 简介
- 验证 Playbook 的工具

2.10. 为虚拟机使用默认 POD 网络

您可以将默认 Pod 网络用于容器原生虚拟化。为此，您必须使用 **masquerade** 绑定方法。这是用于默认 Pod 网络的唯一推荐绑定方法。不要将 **masquerade** 模式用于非默认网络。



注意

对于辅助网络，请使用 **bridge** 绑定方法。

2.10.1. 从命令行配置伪装模式

您可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: red
          masquerade: {} 1
          ports:
            - port: 80 2
  networks:
  - name: red
    pod: {}
```

1 使用伪装模式进行连接

2 允许通过 80 端口传入流量

2. 创建虚拟机：

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

2.10.2. 选择绑定方法

如果从容器原生虚拟化 web 控制台向导创建虚拟机，请从 **Networking** 屏幕选择所需绑定方法。

2.10.2.1. 网络字段

名称	描述
Create NIC	为虚拟机创建新 NIC。
NIC NAME	NIC 的名称。
MAC ADDRESS	网络接口的 MAC 地址。如果未指定 MAC 地址，将为会话生成一个临时地址。
NETWORK CONFIGURATION	可用 NetworkAttachmentDefinition 对象列表。
BINDING METHOD	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。

名称	描述
PXE NIC	支持 PXE 的网络列表。只有在将 **PXE** 选为 **Provision Source** 时才会显示。

2.10.3. 默认网络的虚拟机配置示例

2.10.3.1. 模板：虚拟机配置文件

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: example-vm
  namespace: default
spec:
  running: false
  template:
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
          - masquerade: {}
            name: default
        resources:
          requests:
            memory: 1024M
      networks:
        - name: default
          pod: {}
      volumes:
        - name: containerdisk
          containerDisk:
            image: kubevirt/fedora-cloud-container-disk-demo
        - name: cloudinitdisk
          cloudInitNoCloud:
            userData: |
              #!/bin/bash
              echo "fedora" | passwd fedora --stdin
```

2.10.3.2. 模板：Windows 虚拟机实例配置文件

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
  labels:
```

```
    special: vmi-windows
  name: vmi-windows
spec:
  domain:
    clock:
      timer:
        hpet:
          present: false
        hyperv: {}
        pit:
          tickPolicy: delay
        rtc:
          tickPolicy: catchup
      utc: {}
    cpu:
      cores: 2
    devices:
      disks:
      - disk:
          bus: sata
        name: pvcdisk
      interfaces:
      - masquerade: {}
        model: e1000
        name: default
    features:
      acpi: {}
      apic: {}
      hyperv:
        relaxed: {}
        spinlocks:
          spinlocks: 8191
        vapic: {}
    firmware:
      uuid: 5d307ca9-b3ef-428c-8861-06e72d69f223
    machine:
      type: q35
    resources:
      requests:
        memory: 2Gi
  networks:
  - name: default
    pod: {}
  terminationGracePeriodSeconds: 0
  volumes:
  - name: pvcdisk
    persistentVolumeClaim:
      claimName: disk-windows
```

2.11. 将虚拟机附加到多个网络

容器原生虚拟化提供第 2 层网络功能，支持将虚拟机连接至多个网络。您可使用现有工作负载导入虚拟机，具体取决于对多个接口的访问权限。您还可配置 PXE 网络以便可通过网络启动机器。

首先，网络管理员配置 **cnv-bridge** 类型的 NetworkAttachmentDefinition。然后，用户可将 Pod、虚拟机实例和虚拟机附加到桥接网络。您可以从容器原生虚拟化 web 控制台创建引用桥接网络的 vNIC。

2.11.1. 容器原生虚拟化网络术语表

容器原生虚拟化使用自定义资源和插件提供高级联网功能。

以下是整个容器原生虚拟化文档中使用的术语：

Container Network Interface (CNI)

一个 Cloud Native Computing Foundation 项目，侧重容器网络连接。容器原生虚拟化使用 CNI 插件基于基本 Kubernetes 网络功能进行构建。

Multus

一个"meta"CNI 插件，支持多个 CNI 共存，以便 Pod 或虚拟机可使用其所需的接口。

自定义资源定义 (CRD)

一种 Kubernetes API 资源，用于定义自定义资源，或使用 CRD API 资源定义的对象。

NetworkAttachmentDefinition

一种由 Multus 项目引入的 CRD，用于向一个或多个网络附加 Pod、虚拟机和虚拟机实例。

预启动执行环境 (PXE)

一种接口，让管理员能够通过网络从服务器启动客户端机器。网络启动可用于为客户端远程加载操作系统和其他软件。

2.11.2. 将资源连接至桥接网络

作为网络管理员，您可配置 **cnv-bridge** 类型的 NetworkAttachmentDefinition，为 Pod 和虚拟机提供第 2 层网络。

先决条件

- 容器原生虚拟化 2.0 或更新版本
- 必须在每个节点上配置一个 Linux 网桥，并将其附加到正确的网络接口卡。
- 如果使用 VLAN，则必须在网桥上启用 **vlan_filtering**。
- NIC 必须标记到所有相关 VLAN。
  - 例如：**bridge vlan add dev bond0 vid 1-4095 master**

流程

1. 在任何本地目录中为 NetworkAttachmentDefinition 新建一个新文件。该文件必须具有以下内容，并修改以匹配您的配置：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: a-bridge-network
  annotations:
    k8s.v1.cni.cncf.io/resourceName: bridge.network.kubevirt.io/br0 (1)
spec:
  config: '{
```

```
    "cniVersion": "0.3.1",
    "plugins": [
      {
        "type": "cnv-bridge", (2)
        "bridge": "br0", (3)
        "ipam": {}
      },
      {
        "type": "tuning" (4)
      }
    ]
  }'
```

1. 如果添加此注解到 NetworkAttachmentDefinition，您的虚拟机实例将只在连接 **br0** 网桥的节点上运行。
2. 为该 NetworkAttachmentDefinition 提供网络的 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。
3. 如果该值不是 **br0**，则必须替换网桥的实际名称。
4. 必需。MAC 池管理器可通过此项为连接分配唯一的 MAC 地址。

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

2. 编辑要连接至桥接网络的虚拟机或虚拟机实例的配置文件：

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

1. 您必须替换来自 NetworkAttachmentDefinition 的实际 **name** 值。

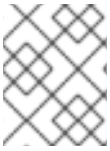
在本例中，NetworkAttachmentDefinition 和 Pod 位于同一个命名空间中。

要指定不同的命名空间，请使用以下语法：

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

3. 向资源应用配置文件：

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



注意

在下一部分中定义 vNIC 时，请确保 **NETWORK** 值是来自您在上一部分中创建的 NetworkAttachmentDefinition 的桥接网络名称。

2.11.3. 为虚拟机创建 NIC

从 web 控制台创建并附加额外 NIC。

流程

1. 在容器原生虚拟化控制台的正确项目中，点击 **Workloads → Virtual Machines**。
2. 选择虚拟机模板。
3. 点击 **Network Interfaces** 以显示已附加到虚拟机的 NIC。
4. 点击 **Create NIC** 在列表中创建新插槽。
5. 填写新 NIC 的 **NAME**、**NETWORK**、**MAC ADDRESS** 和 **BINDING METHOD**。
6. 点击 ✓ 按钮保存并附加 NIC 到虚拟机。

2.11.4. 网络字段

名称	描述
Create NIC	为虚拟机创建新 NIC。
NIC NAME	NIC 的名称。
MAC ADDRESS	网络接口的 MAC 地址。如果未指定 MAC 地址，将为会话生成一个临时地址。
NETWORK CONFIGURATION	可用 NetworkAttachmentDefinition 对象列表。
BINDING METHOD	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。
PXE NIC	支持 PXE 的网络列表。只有在将 **PXE** 选为 **Provision Source** 时才会显示。

在虚拟机上安装可选 QEMU 客户机代理，以便主机可以显示附加网络相关信息。

2.12. 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守护进程。代理将虚拟机上的网络信息（尤其是附加网络的 IP 地址）传递给主机。

先决条件

先决条件

- 通过输入以下命令验证客户机代理是否已安装并正在运行：

```
$ systemctl status qemu-guest-agent
```

2.12.1. 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. 启动 QEMU 客户机代理服务：

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

4. 确保服务持久：

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

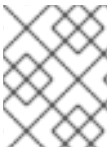
在 web 控制台中创建虚拟机或虚拟机模板时，您还可使用向导的 **cloud-init** 部分中的 **custom script** 字段来安装和启动 QEMU 客户机代理。

2.12.2. 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：

2.12.2.1. 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考您的 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。

2.12.2.2. 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。有关您正在安装的 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。

2.13. 在虚拟机上查看 VNIC 的 IP 地址

QEMU 客户机代理在虚拟机上运行，并会将附加 vNIC 的 IP 地址传递给主机，以便您从 web 控制台和 **oc** 客户端查看 IP 地址。

先决条件

1. 通过输入以下命令验证客户机代理是否已安装并正在运行：

```
$ systemctl status qemu-guest-agent
```

2. 如果客户机代理没有安装和运行，请在虚拟机上安装并运行客户机代理。

2.13.1. 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>

...
Interfaces:
  Interface Name:  eth0
  Ip Address:      10.244.0.37/24
  Ip Addresses:
    10.244.0.37/24
    fe80::858:aff:fef4:25/64
  Mac:             0a:58:0a:f4:00:25
  Name:            default
  Interface Name:  v2
  Ip Address:      1.1.1.7/24
  Ip Addresses:
    1.1.1.7/24
    fe80::f4d9:70ff:fe13:9089/64
  Mac:             f6:d9:70:13:90:89
  Interface Name:  v1
  Ip Address:      1.1.1.1/24
  Ip Addresses:
    1.1.1.1/24
    1.1.1.2/24
    1.1.1.4/24
    2001:de7:0:f101::1/64
    2001:db8:0:f101::1/64
    fe80::1420:84ff:fe10:17aa/64
  Mac:             16:20:84:10:17:aa
```

2.13.2. 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. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 点击虚拟机名称以打开 **Virtual Machine Overview** 屏幕。

每个附加 vNIC 的信息会显示在 **IP ADDRESSES** 下。

2.14. 为虚拟机配置 PXE 启动

容器原生虚拟化中提供 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。

先决条件

- 一个 Linux 网桥必须被连接。
- PXE 服务器必须作为网桥连接至相同 VLAN。

2.14.1. 容器原生虚拟化网络术语表

容器原生虚拟化使用自定义资源和插件提供高级联网功能。

以下是整个容器原生虚拟化文档中使用的术语：

Container Network Interface (CNI)

一个 Cloud Native Computing Foundation 项目，侧重容器网络连接。容器原生虚拟化使用 CNI 插件基于基本 Kubernetes 网络功能进行构建。

Multus

一个"meta"CNI 插件，支持多个 CNI 共存，以便 Pod 或虚拟机可使用其所需的接口。

自定义资源定义 (CRD)

一种 Kubernetes API 资源，用于定义自定义资源，或使用 CRD API 资源定义的对象。

NetworkAttachmentDefinition

一种由 Multus 项目引入的 CRD，用于向一个或多个网络附加 Pod、虚拟机和虚拟机实例。

预启动执行环境 (PXE)

一种接口，让管理员能够通过网络从服务器启动客户端机器。网络启动可用于为客户端远程加载操作系统和其他软件。

2.14.2. 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 NetworkAttachmentDefinition。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 NetworkAttachmentDefinition 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",
        "ipam": {}
      },
      {
        "type": "cnv-tuning" (1)
      }
    ]
  }'
```

(1) **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**。

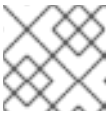
2. 使用您在上一步中创建的文件来创建 NetworkAttachmentDefinition 对象：

```
$ 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。但请注意，自动分配的 MAC 地址不具有持久性。
   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. 指定该网络连接到之前创建的 NetworkAttachmentDefinition。在此场景中，**<pxe-net>** 连接到名为 **<pxe-net-conf>** 的 NetworkAttachmentDefinition：

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

5. 等待虚拟机实例运行：

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

6. 使用 VNC 查看虚拟机实例：

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

7. 查看启动屏幕，验证 PXE 启动是否成功。

8. 登录虚拟机实例：

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

9. 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
...
3. eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen
1000
   link/ether de:00:00:00:00:de brd ff:ff:ff:ff:ff:ff
```

2.14.3. 模板：用于 PXE 启动的虚拟机实例配置文件

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

```
      bootOrder: 2
    - disk:
        bus: virtio
      name: cloudinitdisk
    interfaces:
    - masquerade: {}
      name: default
    - bridge: {}
      name: pxe-net
      macAddress: de:00:00:00:00:de
      bootOrder: 1
  machine:
    type: ""
  resources:
    requests:
      memory: 1024M
networks:
- name: default
  pod: {}
- multus:
    networkName: pxe-net-conf
  name: pxe-net
terminationGracePeriodSeconds: 0
volumes:
- name: containerdisk
  containerDisk:
    image: kubevirt/fedora-cloud-container-disk-demo
- cloudInitNoCloud:
    userData: |
      #!/bin/bash
      echo "fedora" | passwd fedora --stdin
  name: cloudinitdisk
status: {}
```

2.15. 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。容器原生虚拟化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。

这两个程序均有一定程度的风险。只有当您是经验丰富的管理员时方可继续。

2.15.1. 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。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。

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

如果节点处于内存压力下，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
```

2.15.2. 禁用客户机内存开销核算

警告

该程序仅对特定用例有用，且仅限由高级用户操作。

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。

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
```

2.16. 创建虚拟机模板

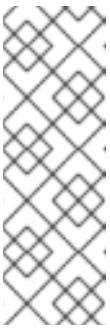
使用虚拟机模板可轻松创建具有相似配置的多个虚拟机。创建完模板后，在创建虚拟机时即可引用该模板。

2.16.1. 利用 web 控制台中的互动向导创建虚拟机模板

web 控制台具有一个交互式向导，指导您完成 **Basic Settings**、**Networking** 和 **Storage** 屏幕操作，以简化虚拟机模板的创建流程。所有必填字段均标有 *。在所有必填字段中提供值后，向导才会移至下一屏幕。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machine Templates**。
2. 点击 **Create Template** 并选择 **Create with Wizard**。
3. 填写所有必填 **Basic Settings**。
4. 点击 **Next** 进入 **Networking** 屏幕。默认会附加名为 **nic0** 的 NIC。
   a. 可选：点击 **Create NIC** 来创建额外 NIC。
   b. 可选：点击 Options 菜单  并选择 **Remove NIC** 即可移除任何或全部 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。
5. 点击 **Next** 进入 **Storage** 屏幕。
   a. 可选：点击 **Create Disk** 创建额外磁盘。
   b. 可选：点击磁盘可修改可用字段。点击 ✓ 按钮保存更改。
   c. 可选：点击 **Attach Disk** 从 **Select Storage** 列表中选择可用磁盘。



注意

如果在 **Basic Settings** 屏幕中将 **URL** 或 **Container** 选为 **Provision Source**，则会创建一个 **rootdisk** 磁盘，并将其作为 **Bootable Disk** 附加到虚拟机。您可修改 **rootdisk**，但不可将其移除。

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**。

6. 点击 **Create Virtual Machine Template** ⁊ **Results** 屏幕显示虚拟机模板的 JSON 配置文件。该模板列于 **Workloads → Virtual Machine Templates** 中。

2.16.2. 虚拟机模板交互式向导字段

下表描述了 **Create Virtual Machine Template**交互式向导中 **Basic Settings**、**Networking** 和 **Storage** 窗格的字段。

2.16.2.1. 虚拟机模板向导字段

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。
Provision Source	PXE	从 PXE 菜单置备虚拟机。集群中需要支持 PXE 的 NIC。
	URL	从由 **HTTP** 或 **S3** 端点提供的镜像置备虚拟机。
	Container	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***。
	Cloned Disk	置备源是一个克隆的磁盘。
	Import	从所支持的提供程序导入虚拟机。
Operating System		集群中可用操作系统列表。这是虚拟机的主要操作系统。如果将 **Import** 选为 **Provider Source,** 则操作系统将基于待导入 Vmware 虚拟机的操作系统自动填写。
Flavor	small、medium、large、tiny、Custom	预设值，用于决定分配给虚拟机的 CPU 和内存量。
Workload Profile	desktop	用于桌面的虚拟机配置。
	generic	可平衡各种工作负载的性能和兼容性的虚拟机配置。

名称	参数	描述
	high performance	针对高性能负载进行了优化的虚拟机配置。
Use cloud-init		选择此项可启用 cloud-init 字段。

2.16.2.2. Cloud-init 字段

名称	描述
Hostname	为虚拟机设置具体主机名。
Authenticated SSH Keys	复制到虚拟机上 *~/.ssh/authorized_keys* 的用户公钥。
Use custom script	将其他选项替换为您粘贴自定义 cloud-init 脚本的字段。

2.16.2.3. 网络字段

名称	描述
Create NIC	为虚拟机创建新 NIC。
NIC NAME	NIC 的名称。
MAC ADDRESS	网络接口的 MAC 地址。如果未指定 MAC 地址，将为会话生成一个临时地址。
NETWORK CONFIGURATION	可用 NetworkAttachmentDefinition 对象列表。
BINDING METHOD	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。
PXE NIC	支持 PXE 的网络列表。只有在将 **PXE** 选为 **Provision Source** 时才会显示。

2.16.2.4. 存储字段

名称	描述
Create Disk	为虚拟机创建新磁盘。

名称	描述
Attach Disk	从可用 PVC 列表中选择一个现有磁盘，以附加到虚拟机。
DISK NAME	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 (GB)	磁盘大小（以 GB 为单位）。
STORAGE CLASS	底层 **StorageClass** 的名称。
Bootable Disk	虚拟机将从中启动的可用磁盘列表。如果虚拟机的 **Provision Source** 为 **URL** 或 **Container**，该字段将锁定为 **rootdisk**。

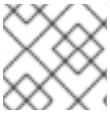
2.17. 编辑虚拟机模板

您可在 web 控制台中编辑虚拟机模板。

2.17.1. 在 web 控制台中编辑虚拟机模板

您可从 web 控制台编辑虚拟机模板的 YAML 配置。

并非所有参数均可修改。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. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machine Templates**。
2. 选择一个模板。
3. 点击 **YAML** 选项卡以显示可编辑的配置。
4. 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。

2.17.2. 将虚拟磁盘添加到虚拟机模板

为虚拟机添加一个磁盘

流程

1. 在 Virtual Machine Templates 选项卡中选择您的虚拟机模板。
2. 选择 Disks 选项卡。
3. 点击 Add Disks 打开 Add Disk 窗口。
4. 在 Add Disk 窗口中，指定 Source、Name、Size、Interface 和 Storage Class。
5. 使用下拉列表和复选框编辑磁盘配置。
6. 点击 OK。

2.17.3. 将网络接口添加到虚拟机模板

流程

1. 在 Virtual Machine Templates 选项卡中选择虚拟机模板。
2. 选择 Network Interfaces 选项卡。
3. 点 Create Network Interface。
4. 在 Create Network Interface 列表行中，指定网络接口的 Name、Model、Network、Type 和 MAC Address。
5. 点击行右边的绿色复选框来添加网络接口。
6. 重启虚拟机以启用访问。
7. 编辑下拉列表和复选框来配置网络接口。
8. 点击 Save Changes。
9. 点击 OK。

新网络接口显示在 Create Network Interfac 列表的顶部，直到用户重启。

新网络接口有一个 **Pending VM restart** 链接状态，直到您重启虚拟机。将鼠标悬停在链接状态上方可显示更详细的信息。

当在虚拟机上定义了网络接口卡并连接到网络时，Link State 会被默认设置为 Up。

2.18. 删除虚拟机模板

您可删除 web 控制台中的虚拟机模板。

2.18.1. 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。

流程

1. 在容器原生虚拟化控制台中，点击 Workloads → Virtual Machine Templates。

2. 您可从本窗格删除虚拟机，这有助于对一个窗格中的多个模板执行操作，也可从 **Virtual Machine Template Details** 窗格，其中可查看所选模板的综合详情：

   - 点击 Options 菜单  （要删除的模板），然后选择 **Delete Template**。
   - 点击模板名称打开 **Virtual Machine Template Details**窗格，然后点击 **Actions → Delete Template**。

3. 在确认弹出窗口中点击 **Delete** 永久删除模板。

2.19. 将虚拟机磁盘克隆到新 DATAVOLUME 中

您可通过引用 DataVolume 配置文件中的源 PVC 来将虚拟机磁盘的 PersistentVolumeClaim (PVC) 克隆到新 DataVolume 中。

先决条件

- 您可能需要定义一个 StorageClass 或准备 CDI 涂销空间 才能成功完成此操作。CDI 支持的操作列表显示需要涂销空间的状况。

2.19.1. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.19.2. 将虚拟机磁盘的 PersistentVolumeClaim 克隆到新 DataVolume 中

您可将现有虚拟机磁盘的 PersistentVolumeClaim (PVC) 克隆到新 DataVolume 中。之后该新 DataVolume 可用于新虚拟机。

注意

当独立于虚拟机创建 DataVolume 时，DataVolume 的生命周期与虚拟机保持独立。如果删除了虚拟机，DataVolume 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 命令行界面 (CLI)，通常称为 **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 DataVolume 对象创建 YAML 文件，用于指定新 DataVolume 的名称、源 PVC 的名称和命名空间，以及新 DataVolume 的大小。
   例如：

```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
```

```
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
    pvc:
      namespace: "<source-namespace>" 2
      name: "<my-favorite-vm-disk>" 3
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 4
```

1. 新 DataVolume 的名称。
2. 源 PVC 所在的命名空间。
3. 源 PVC 的名称。
4. 新 DataVolume 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 DataVolume 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 DataVolume 的虚拟机。

2.19.3. 模板：DataVolume 克隆配置文件

example-clone-dv.yaml

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

2.19.4. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	☐ TAR	☐ TAR

✓ 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

+ 存档不支持块模式 DV

2.20. 使用 DATAVOLUMETEMPLATE 克隆虚拟机

您可通过克隆现有虚拟机的 PersistentVolumeClaim (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 DataVolume。

先决条件

- 您可能需要定义一个 StorageClass 或准备 CDI 涂销空间 才能成功完成此操作。CDI 支持的操作列表显示需要涂销空间的状况。

2.20.1. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.20.2. 使用 DataVolumeTemplate 从克隆的 PersistentVolumeClaim 创建新虚拟机

您可创建一个虚拟机，它将现有虚拟机的 PersistentVolumeClaim (PVC) 克隆到 DataVolume 中。通过在虚拟机 **spec** 中引用 **dataVolumeTemplate**，**源** PVC 便会克隆到 DataVolume 中，然后自动用于创建虚拟机。

注意

当 DataVolume 作为虚拟机 DataVolumeTemplate 的一部分创建时，DataVolume 的生命周期依赖于虚拟机。如果删除了虚拟机，DataVolume 及其相关 PVC 也会一并删除。

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 命令行界面 (CLI)，通常称为 **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。名为 **favorite-clone** 的 **2Gi** DataVolume 从 **my-favorite-vm-disk** 创建而成。
   例如：

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

```
        storage: 2Gi
    source:
      pvc:
        namespace: "source-namespace"
        name: "my-favorite-vm-disk"
```

1 要创建的虚拟机。

3. 使用 PVC 克隆的 DataVolume 创建虚拟机：

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

2.20.3. 模板：DataVolume 虚拟机配置文件

example-dv-vm.yaml

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: example-vm
  name: example-vm
spec:
  dataVolumeTemplates:
  - metadata:
      name: example-dv
    spec:
      pvc:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 1G
      source:
          http:
            url: "" 1
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/vm: example-vm
    spec:
      domain:
        cpu:
          cores: 1
        devices:
          disks:
          - disk:
              bus: virtio
            name: example-dv-disk
        machine:
          type: q35
        resources:
          requests:
```

```
      memory: 1G
terminationGracePeriodSeconds: 0
volumes:
- dataVolume:
    name: example-dv
  name: example-dv-disk
```

1 您要导入的镜像的 **HTTP** 源（如适用）。

2.20.4. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	□ TAR	□ TAR

✓ 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

+ 存档不支持块模式 DV

2.21. 使用 VIRTCTL 工具上传本地磁盘镜像

您可使用 **virtctl** 命令行实用程序上传本地存储的磁盘镜像。

先决条件

- 安装 **kubevirt-virtctl** 软件包
- 您可能需要定义一个 StorageClass 或准备 CDI 涂销空间 才能成功完成此操作。

2.21.1. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	☐ TAR	☐ TAR

✓ 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

+ 存档不支持块模式 DV

2.21.2. 上传本地磁盘镜像至新 PersistentVolumeClaim

您可使用 **virtctl** CLI 实用程序将虚拟机磁盘镜像从客户端机器上传至集群。上传磁盘镜像可创建一个 PersistentVolumeClaim (PVC)，您可将其与虚拟机关联。

先决条件

- 具有 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。
- **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：
   - 要上传的虚拟机磁盘镜像的文件位置
   - 生成 PVC 的名称和大小
2. 运行 **virtctl image-upload** 命令以上传您的虚拟机镜像。您必须指定 PVC 名称、PVC 大小以及文件位置。例如：

```
$ virtctl image-upload --pvc-name=<upload-fedora-pvc> --pvc-size=<2Gi> --image-path=
</images/fedora.qcow2>
```

小心

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. 要验证 PVC 是否已创建，请查看所有 PVC 对象：

```
$ oc get pvc
```

2.22. 上传本地磁盘镜像至块存储 DATAVOLUME

您可使用 **virtctl** 命令行实用程序上传本地磁盘镜像至块 DataVolume 中。

在此工作流中，您会创建一个本地块设备用作 PersistentVolume，将此块卷与 **upload** DataVolume 关联，并使用 **virtctl** 将本地磁盘镜像上传至 DataVolume 中。

先决条件

- 如果根据 CDI 支持的操作列表规定需要涂销空间，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。

2.22.1. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.22.2. 关于块 PersistentVolume

块 PersistentVolume (PV) 是一个受原始块设备支持的 PV。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虚拟机来说，使用它可以获得性能优势。

原始块卷可以通过在 PV 和 PersistentVolumeClaim (PVC) 规格中指定 **volumeMode: Block** 来置备。

2.22.3. 创建本地块 PersistentVolume

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 PersistentVolume (PV)。然后您可以在 PV 配置中将该循环设备引用为 **Block** 卷，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** 配置。

```
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 设置一个 StorageClass。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 PersistentVolume 的文件名。

2.22.4. 创建上传 DataVolume

使用 **upload** 数据源创建 DataVolume，用于上传本地磁盘镜像。

流程

1. 创建 DataVolume 配置，用于指定 **spec: source: upload{}** ：

```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: <upload-datavolume> 1
spec:
  source:
    upload: {}
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi> 2
```

1 DataVolume 的名称

2 DataVolume 的大小

2. 创建 DataVolume：

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

2.22.5. 上传本地磁盘镜像至新 DataVolume

您可使用 **virtctl** CLI 实用程序将虚拟机磁盘镜像从客户端机器上传至集群中的 DataVolume (DV)。上传镜像后，您可将其添加到虚拟机中。

先决条件

- 具有 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。
- **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。
- 具有一个大小不低于您要上传的磁盘的备用 DataVolume。

流程

1. 确定以下各项：
   - 要上传的虚拟机磁盘镜像的文件位置
   - DataVolume 的名称
2. 运行 **virtctl image-upload** 命令以上传您的磁盘镜像。您必须指定 DV 名称以及文件位置。例如：

```
$ virtctl image-upload --dv-name=<upload-datavolume> --image-path=
</images/fedora.qcow2> 1 2
```

1 正在创建的 DataVolume 的名称。

2 正在上传的虚拟机磁盘镜像的文件路径。

小心

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. 要验证 DV 是否已创建，请查看所有 DV 对象：

```
$ oc get dvs
```

2.22.6. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	☐ TAR	☐ TAR

✓ 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

+ 存档不支持块模式 DV

2.23. 通过添加空白磁盘镜像扩展虚拟存储

您可向容器原生虚拟化添加空白磁盘镜像来提高存储容量或创建新数据分区。

2.23.1. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.23.2. 使用 DataVolume 创建空白磁盘镜像

您可通过自定义和部署 DataVolume 配置文件在 PersistentVolumeClaim 中创建新空白磁盘镜像。

先决条件

- 至少一个可用 PersistentVolume
- 安装 OpenShift 命令行界面 (CLI)，通常称为 **oc**

流程

1. 编辑 DataVolume 配置文件：

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

2. 运行以下命令，创建空白磁盘镜像：

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

2.23.3. 模板：适用于空白磁盘镜像的 DataVolume 配置文件

blank-image-datavolume.yaml

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

2.24. 准备 CDI 涂销空间

2.24.1. 关于 DataVolume

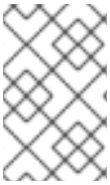
DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.24.2. 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 DataVolume (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。

该 CDIConfig 对象可用于通过设置 CDIConfig 对象的 **spec:** 部分中的 **scratchSpaceStorageClass** 来定义使用哪个 StorageClass 绑定涂销空间 PVC。

如果定义的 StorageClass 与集群中的 StorageClass 不匹配，则会使用为集群定义的默认 StorageClass。如果没有在集群中定义默认 StorageClass，则会使用置备原始 DV 或 PVC 时使用的 StorageClass。



注意

CDI 需要通过 **file** 卷模式来请求涂销空间，与支持原始 DataVolume 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 则会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。

2.24.3. 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。
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。

2.24.4. 在 CDI 配置中定义 StorageClass

在 CDI 配置中定义 StorageClass，为 CDI 操作动态置备涂销空间。

流程

- 使用 **oc** 客户端来编辑 **cdiconfig/config** 并添加或编辑 **spec: scratchSpaceStorageClass**，以便与集群中的 StorageClass 匹配。

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

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

2.24.5. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	□ TAR	□ TAR

✓ 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

+ 存档不支持块模式 DV

其他资源

- 有关 StorageClass 以及如何在集群中对其进行定义的更多信息，请参阅动态置备小节。

2.25. 使用 DATAVOLUME 将虚拟机镜像导入到块存储

您可将现有虚拟机镜像导入到您的 OpenShift Container Platform 集群中。容器原生虚拟化使用 DataVolume 自动导入数据并创建底层 PersistentVolumeClaim (PVC)。

小心

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。详情请参阅操作系统文档。

先决条件

- 如果根据 CDI 支持的操作列表规定需要涂销空间，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。

2.25.1. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.25.2. 关于块 PersistentVolume

块 PersistentVolume (PV) 是一个受原始块设备支持的 PV。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虚拟机来说，使用它可以获得性能优势。

原始块卷可以通过在 PV 和 PersistentVolumeClaim (PVC) 规格中指定 **volumeMode: Block** 来置备。

2.25.3. 创建本地块 PersistentVolume

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 PersistentVolume (PV)。然后您可以在 PV 配置中将该循环设备引用为 **Block** 卷，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** 配置。

```
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 设置一个 StorageClass。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 PersistentVolume 的文件名。

2.25.4. 使用 DataVolume 导入虚拟机镜像至块 PersistentVolume

您可将现有虚拟机镜像导入到您的 OpenShift Container Platform 集群中。容器原生虚拟化使用 DataVolume 自动导入数据并创建底层 PersistentVolumeClaim (PVC)。然后您可以在虚拟机配置中引用 DataVolume。

先决条件

- 具有 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 **s3** 端点，以及访问数据源所需的任何身份验证凭证
- 至少有一个可用块 PV。

流程

1. 如果您的数据源需要身份验证凭证，请编辑 **endpoint-secret.yaml** 文件，并在集群中应用更新的配置。

   a. 利用您首选的文本编辑器来编辑 **endpoint-secret.yaml** 文件：

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

   (1) 可选：您的密钥或用户名，base64 编码

   (2) 可选：您的 secret 或密码，base64 编码

   b. 更新 secret：

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

2. 创建 **DataVolume** 配置，用于指定要导入的镜像的数据源和 **volumeMode: Block**，以便使用可用块 PV。

```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: <import-pv-datavolume> (1)
spec:
  storageClassName: local (2)
  source:
     http:
       url:
<http://download.fedoraproject.org/pub/fedora/linux/releases/28/Cloud/x86_64/images/Fedora
-Cloud-Base-28-1.1.x86_64.qcow2> (3)
       secretRef: <endpoint-secret> (4)
  pvc:
    volumeMode: Block (5)
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: <2Gi>
```

(1) DataVolume 的名称。

(2) 可选：设置存储类，或忽略此项，接受集群默认值。

(3) 要导入的镜像的 **HTTP** 源。

4 仅在数据源需要身份验证时才需要。

5 导入到块 PV 时需要。

3. 创建 DataVolume 以导入虚拟机镜像。

```
$ oc create -f <import-pv-datavolume.yaml>1
```

1 上一步中创建的文件名 DataVolume。

2.25.5. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	☐ TAR	☐ TAR

✓ 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

+ 存档不支持块模式 DV

2.26. 将虚拟机磁盘克隆到新块存储 DATAVOLUME 中

您可通过引用 DataVolume 配置文件中的源 PVC 来将虚拟机磁盘的 PersistentVolumeClaim (PVC) 克隆到新块 DataVolume 中。

先决条件

- 如果根据 CDI 支持的操作列表规定需要涂销空间，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。

2.26.1. 关于 DataVolume

DataVolume 对象是 Containerized Data Importer (CDI) 项目提供的自定义资源。DataVolume 编配与底层 PersistentVolumeClaim (PVC) 关联的导入、克隆和上传操作。DataVolume 与 KubeVirt 集成，它们可在 PVC 准备好前阻止虚拟机启动。

2.26.2. 关于块 PersistentVolume

块 PersistentVolume (PV) 是一个受原始块设备支持的 PV。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虚拟机来说，使用它可以获得性能优势。

原始块卷可以通过在 PV 和 PersistentVolumeClaim (PVC) 规格中指定 **volumeMode: Block** 来置备。

2.26.3. 创建本地块 PersistentVolume

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 PersistentVolume (PV)。然后您可以在 PV 配置中将该循环设备引用为 **Block** 卷，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** 配置。

```
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 设置一个 StorageClass。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 PersistentVolume 的文件名。

2.26.4. 将虚拟机磁盘的 PersistentVolumeClaim 克隆到新 DataVolume 中

您可将现有虚拟机磁盘的 PersistentVolumeClaim (PVC) 克隆到新 DataVolume 中。之后该新 DataVolume 可用于新虚拟机。

注意

当独立于虚拟机创建 DataVolume 时，DataVolume 的生命周期与虚拟机保持独立。如果删除了虚拟机，DataVolume 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 命令行界面 (CLI)，通常称为 **oc**。
- 至少一个可用块 PersistentVolume (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 DataVolume 对象创建 YAML 文件，用于指定新 DataVolume 的名称、源 PVC 的名称和命名空间、**volumeMode: Block**（以使用可用块 PV），以及新 DataVolume 的大小。
   例如：

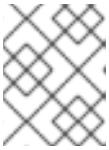
```
apiVersion: cdi.kubevirt.io/v1alpha1
kind: DataVolume
metadata:
  name: <cloner-datavolume> 1
spec:
  source:
```

```
  pvc:
    namespace: "<source-namespace>" 2
    name: "<my-favorite-vm-disk>" 3
pvc:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <2Gi> 4
  volumeMode: Block 5
```

1. 新 DataVolume 的名称。
2. 源 PVC 所在的命名空间。
3. 源 PVC 的名称。
4. 新 DataVolume 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 DataVolume 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 DataVolume 的虚拟机。

2.26.5. 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*
Archive+	✓ TAR	✓ TAR	✓ TAR	☐ TAR	☐ TAR

✓ 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

+ 存档不支持块模式 DV

2.27. 虚拟机实时迁移

2.27.1. 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迁移到集群中另一节点的过程。如果您选择将一个虚拟机实例迁移到另一节点，则该过程为手动过程，如果虚拟机实例具有 **LiveMigrate** 驱除策略且运行它的节点已置于维护中，则该过程为自动过程。

重要

虚拟机必须具有一个采用共享 ReadWriteMany (RWX) 访问模式的 PersistentVolumeClaim (PVC) 才能实时迁移。

其他资源：

- 迁移虚拟机实例到另一节点
- 节点维护模式
- 实时迁移限制

2.28. 实时迁移限制和超时

应用实时迁移限制和超时，以防迁移过程使集群不堪重负。通过编辑 **kubevirt-config** 配置文件来配置这些设置。

2.28.1. 配置实时迁移限制和超时

通过向 **kubevirt-config** 配置文件添加更新的 key:value 字段来为集群配置实时迁移限制和超时，该文件位于 **openshift-cnv** 命名空间中。

流程

- 编辑 **kubevirt-config** 配置文件并添加必要的实时迁移参数。以下示例显示默认值：

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

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

```
parallelMigrationsPerCluster: 5
parallelOutboundMigrationsPerNode: 2
bandwidthPerMigration: 64Mi
completionTimeoutPerGiB: 800
progressTimeout: 150
```

2.28.2. 集群范围内的实时迁移限制和超时

表 2.5. 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。	64Mi
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

2.29. 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。

2.29.1. 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. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 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 Details** 屏幕中进行，其中可查看所选虚拟机的综合详情：

- 点击 Options 菜单  （在虚拟机后面），然后选择 **Migrate Virtual Machine**。
- 点击虚拟机名称，打开 **Virtual Machine Details** 屏幕，然后点击 **Actions → Migrate Virtual Machine**。

3. 点击 **Migrate** 把虚拟机迁移到另一节点。

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

2. 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：

- 监控虚拟机实例的实时迁移
- 取消虚拟机实例的实时迁移

2.30. 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。

2.30.1. 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** 列表中，或显示在正在迁移的虚拟机的 **Virtual Machine Details** 屏幕中。

流程

- 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。

2.30.2. 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
```

```
...
Status:
  Conditions:
    Last Probe Time:       <nil>
    Last Transition Time:  <nil>
    Status:                True
    Type:                  LiveMigratable
  Migration Method:  LiveMigration
  Migration State:
    Completed:                true
    End Timestamp:            2018-12-24T06:19:42Z
    Migration UID:            d78c8962-0743-11e9-a540-fa163e0c69f1
    Source Node:              node2.example.com
    Start Timestamp:          2018-12-24T06:19:35Z
    Target Node:              node1.example.com
    Target Node Address:      10.9.0.18:43891
    Target Node Domain Detected:  true
```

2.31. 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。

2.31.1. 在 web 控制台中取消虚拟机实例的实时迁移

可以使用 Workloads → Virtual Machines 屏幕中每个虚拟机上的 Options 菜单， ，或从 **Virtual Machine Details** 屏幕上的 **Actions** 菜单来取消虚拟机实例的实时迁移。

流程

1. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 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 Details** 屏幕进行，其中可查看所选虚拟机的综合详情：
   - 点击 Options 菜单 （在虚拟机后面），然后选择 **Cancel Virtual Machine Migration**。
   - 点击虚拟机名称，打开 **Virtual Machine Details** 屏幕，然后点击 **Actions → Cancel Virtual Machine Migration**。
3. 点击 **Cancel Migration** 以取消虚拟机实时迁移。

2.31.2. 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
```

2.32. 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。

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

2. 重启虚拟机以使更新生效：

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

2.33. 节点维护模式

2.33.1. 了解节点维护模式

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在该节点上被删除，并会在另一节点上重新创建。

重要

虚拟机必须具有一个采用共享 ReadWriteMany (RWX) 访问模式的 PersistentVolumeClaim (PVC) 才能实时迁移。

其他资源：

- 虚拟机实时迁移
- 配置虚拟机驱除策略

2.34. 将节点设置为维护模式

2.34.1. 了解节点维护模式

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在该节点上被删除，并会在另一节点上重新创建。

重要

虚拟机必须具有一个采用共享 ReadWriteMany (RWX) 访问模式的 PersistentVolumeClaim (PVC) 才能实时迁移。

通过 web 控制台或 CLI 将节点置于维护模式

2.34.2. 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** 屏幕中的 **Actions** 控件将节点设置为维护模式。

流程

1. 在容器原生虚拟化控制台中，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 Options 菜单  （在节点后面），然后选择 **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。

2.34.3. 在 CLI 中将节点设置为维护模式

通过创建 **NodeMaintenance** 自定义资源 (CR) 对象来将节点设置为维护模式，该对象需引用节点名称以及将节点设置为维护模式的原因。

流程

1. 创建节点维护模式的 CR 配置。本例使用名为 **node02-maintenance.yaml** 的 CR：

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

2. 在集群中创建 **NodeMaintenance** 对象：

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

该节点会实时迁移具有 **liveMigration** 驱除策略的虚拟机实例，并污染该节点，使其不可再调度。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。

其他资源：

- 从维护模式恢复节点

2.35. 从维护模式恢复节点

恢复节点会使节点退出维护模式，可再次调度。

通过 web 控制台或 CLI 从维护模式恢复节点。

2.35.1. 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** 控件将节点设置为维护模式。

流程

1. 在容器原生虚拟化控制台中，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 Options 菜单  （在节点后面），然后选择 **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。

2.35.2. 在 CLI 中从维护模式恢复节点

通过删除节点的 **NodeMaintenance** 对象从维护模式恢复节点并使其可再次调度。

流程

1. 查找 **NodeMaintenance** 对象：

```
$ oc get nodemaintenance
```

2. 可选：检查 **NodeMaintenance** 对象以确保其与正确节点关联：

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

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

3. 删除 **NodeMaintenance** 对象：

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

2.36. 手动刷新 TLS 证书

容器原生虚拟化组件的 TLS 证书是在安装时创建的，并在一年内有效。您必须在这些证书过期前进行手动刷新。

2.36.1. 刷新 TLS 证书

要刷新容器原生虚拟化的 TLS 证书，请下载并运行 **rotate-certs** 脚本。该脚本可从 GitHub 上的 **kubevirt/hyperconverged-cluster-operator** 存储库中获得。

重要

刷新证书时，以下操作会受到影响：

- 迁移会取消
- 镜像上传会取消
- VNC 和控制台连接会关闭

先决条件

- 确保您以具有 **cluster-admin** 权限的用户身份登录集群。该脚本使用与集群的活跃会话来刷新 **openshift-cnv** 命名空间中的证书。

流程

1. 从 GitHub 下载 **rotate-certs.sh** 脚本：

```
$ curl -O https://raw.githubusercontent.com/kubevirt/hyperconverged-cluster-operator/master/tools/rotate-certs.sh
```

2. 确保脚本可以执行：

```
$ chmod +x rotate-certs.sh
```

3. 运行脚本：

```
$ ./rotate-certs.sh -n openshift-cnv
```

TLS 证书已刷新，并在一年内有效。

2.37. 在现有 WINDOWS 虚拟机上安装 VIRTIO 驱动程序

2.37.1. 了解 VirtIO 驱动程序

VirtIO 驱动程序是 Microsoft Windows 虚拟机在容器原生虚拟化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 Red Hat Container Catalog 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 。

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

表 2.6. 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。

2.37.3. 将 VirtIO 驱动程序容器磁盘添加到虚拟机中

针对 Microsoft Windows 的容器原生虚拟化 VirtIO 驱动程序作为一个容器磁盘提供，可在 Red Hat Container 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 Container 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) 容器原生虚拟化按照 **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。

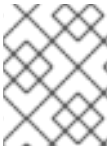
2. 虚拟机启动后，磁盘随即可用：

   - 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。

2.37.4. 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考您的 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。

2.37.5. 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。

2.38. 在新 WINDOWS 虚拟机上安装 VIRTIO 驱动程序

先决条件

- Windows 安装介质可从虚拟机访问，例如将 ISO 导入到数据卷 并将其附加到虚拟机。

2.38.1. 了解 VirtIO 驱动程序

VirtIO 驱动程序是 Microsoft Windows 虚拟机在容器原生虚拟化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 Red Hat Container Catalog 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。

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

表 2.7. 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。

2.38.3. 将 VirtIO 驱动程序容器磁盘添加到虚拟机中

针对 Microsoft Windows 的容器原生虚拟化 VirtIO 驱动程序作为一个容器磁盘提供，可在 Red Hat Container 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 Container 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 容器原生虚拟化按照 **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。

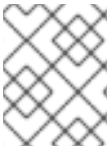
2. 虚拟机启动后，磁盘随即可用：

   - 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。

2.38.4. 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。有关您正在安装的 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。

2.38.5. 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。

2.39. 查看虚拟机日志

2.39.1. 了解虚拟机日志

收集 OpenShift Container Platform Build、Deployment 和 Pod 日志。在容器原生虚拟化中，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。

2.39.2. 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>
```

2.39.3. 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. 在容器原生虚拟化控制台中，点击 **Workloads → Virtual Machines**。
2. 点击虚拟机以打开 **Virtual Machine Details** 面板。
3. 进入 **Overview** 选项卡，点击 **POD** 部分的 **virt-launcher-<vm-name>** Pod。
4. 点击 **Logs**。

2.40. 使用 OPENSHIFT CONTAINER PLATFORM 仪表板获取集群信息

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*。

2.40.1. 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
  - 持久性存储卷声明
  - 虚拟机（如果安装了容器原生虚拟化，则可用）
  - 集群中的裸机主机，根据其状态列出（仅在 **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。如果安装了容器原生虚拟化，还会诊断容器原生虚拟化的整体健康状况。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

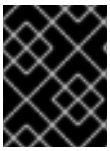
- **Cluster Utilization** 显示在指定时间段内各种资源的能力，以帮助管理员了解高资源消耗的范围和频率。
- **Events** 列出了与集群中最近活动相关的消息，如创建 Pod，或者虚拟机迁移到另一台主机。
- **Top Consumers** 可帮助管理员了解集群资源是如何被消耗的。点一个资源可以进入一个包括详细信息的页面，它列出了对指定集群资源（CPU、内存或者存储）消耗最多的 Pod 和节点。

2.41. 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。

2.41.1. 关于 OpenShift Container Platform 集群监控

OpenShift Container Platform 包括一个预配置、预安装且自助更新的监控堆栈，它基于 Prometheus 开源项目及其更广的生态系统。它提供对集群组件的监控，并且包含一组警报（在发生任何问题时立即通知集群管理员）以及一组 Grafana 仪表板。集群监控堆栈只支持监控 OpenShift Container Platform 集群。



重要

为确保与将来的 OpenShift Container Platform 更新兼容，只支持配置特定的监控堆栈选项。

2.41.2. 集群日志记录组件

集群日志记录组件基于 Elasticsearch、Fluentd 或 Rsyslog 以及 Kibana（EFK）。收集器 Fluentd 部署到 OpenShift Container Platform 集群中的每个节点。它收集所有节点和容器日志，并将它们写入 Elasticsearch (ES)。Kibana 是一个集中式 Web UI，用户和管理员可以在其中使用汇总的数据创建丰富的视觉化和仪表板。

目前有 5 种不同类型的集群日志记录组件：

- logStore（存储） - 存储日志的位置。当前的实现是 Elasticsearch。
- collection（收集） - 此组件从节点收集日志，将日志格式化并存储到 logStore 中。当前的实现是 Fluentd。
- 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。
- curation（策展） - 此组件按日志时间进行筛检。当前的实现是 Curator。
- event routing - 用来把 OpenShift Container Platform 的事件转发到集群日志记录的组件。当前的实现是 Event Router。

有关集群日志记录的更多信息，请参阅 OpenShift Container Platform 集群日志 文档。

2.41.3. 关于 Telemetry

Telemetry 会向红帽发送一组精选的集群监控指标子集。这些指标会持续发送并描述：

- OpenShift Container Platform 集群的大小
- OpenShift Container Platform 组件的健康和状态
- 正在进行的任何升级的健康和状态
- 有关 OpenShift Container Platform 组件和功能的有限使用情况信息
- 有关集群监控组件所报告的警报的摘要信息

红帽将使用这一持续数据流实时监控集群的健康，必要时将对影响客户的问题做出反应。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。所有这些信息都不会与第三方共享。

2.41.3.1. Telemetry 收集的信息

Telemetry 收集的主要信息包括：

- 每个集群可用的更新数
- 用于更新的频道和镜像仓库
- 更新期间发生的错误数
- 正在运行的更新的进度信息
- 每个集群的机器数
- 机器的 CPU 内核数和 RAM 大小
- etcd 集群中的成员数，以及当前存储在 etcd 集群中的对象数
- 每种机器类型（infra 或 master）使用的 CPU 内核数和 RAM 大小
- 每个集群使用的 CPU 内核数和 RAM 大小
- 每个集群的 OpenShift Container Platform 框架组件的使用情况
- OpenShift Container Platform 集群的版本
- 集群上安装的任何 OpenShift Container Platform 框架组件（如 Cluster Version Operator、Cluster Monitoring、Image Registry、Elasticsearch for Logging）的健康、情况和状态。
- 安装期间生成的随机的唯一标识符
- OpenShift Container Platform 部署平台的名称，如 Amazon Web Services

Telemetry 不会收集任何身份识别的信息，如用户名、密码、用户资源的名称或地址。

2.41.4. 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。

2.42. 为红帽支持收集容器原生虚拟化数据

在提交问题单时同时提供您的集群信息，可以帮助红帽支持为您进行排除故障。

您可使用 **must-gather** 工具来收集有关 OpenShift Container Platform 集群的诊断信息，包括虚拟机和有关容器原生虚拟化的其他数据。

为了获得快速支持，请提供 OpenShift Container Platform 和容器原生虚拟化的诊断信息。

重要

容器原生虚拟化仅是一项技术预览功能。技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。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，请参阅 https://access.redhat.com/support/offerings/techpreview/。

2.42.1. 关于 must-gather 工具

oc adm must-gather CLI 命令可收集最有助于解决问题的集群信息，如：

- 资源定义
- 审计日志
- 服务日志

您在运行该命令时，可通过包含 **--image** 参数来指定一个或多个镜像。指定镜像后，该工具便会收集有关相应功能或产品的信息。

您在运行 **oc adm must-gather** 时，集群上会创建一个新 Pod。在该 Pod 上收集数据，并保存至以 **must-gather.local** 开头的一个新目录中。此目录在当前工作目录中创建。

2.42.2. 关于收集容器原生虚拟化数据

您可使用 **oc adm must-gather** CLI 命令来收集有关集群的信息，包括与容器原生虚拟化关联的功能和对象：

- Hyperconverged Cluster Operator 命名空间（及子对象）
- 属于任何容器原生虚拟化资源的所有命名空间（及其子对象）
- 所有容器原生虚拟化的自定义资源定义 (CRD)
- 包含虚拟机的所有命名空间
- 所有虚拟机定义

要使用 **must-gather** 来收集容器原生虚拟化数据，您必须指定容器原生虚拟化镜像：**--image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8**。

2.42.3. 收集有关特定功能的数据

您可通过将 **oc adm must-gather** CLI 命令与 **--image** 或 **--image-stream** 参数结合使用来收集有关特定功能的调试信息。**must-gather** 工具支持多个镜像，这样您便可通过运行单个命令收集多个功能的数据。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift Container Platform CLI (**oc**)。

流程

1. 导航至您要存储 **must-gather** 数据的目录。
2. 使用一个或多个 **--image** 或 **--image-stream** 参数运行 **oc adm must-gather** 命令。例如，使用以下命令可收集默认集群数据和容器原生虚拟化特定信息：

```
$ oc adm must-gather \
 --image-stream=openshift/must-gather \ 1
 --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8 2
```

1. 默认 OpenShift Container Platform must-gather 镜像
2. 容器原生虚拟化 must-gather 镜像

3. 从刚刚在您的工作目录中创建的 **must-gather** 目录创建一个压缩文件。例如，在使用 Linux 操作系统的计算机上运行以下命令：

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

1. 务必将 **must-gather-local.5421342344627712289/** 替换为实际目录名称。

4. 在红帽客户门户中为您的问题单附上压缩文件。

第 3 章 容器原生虚拟化 2.1 发行注记

3.1. 容器原生虚拟化 2.1 发行注记

3.1.1. 关于容器原生虚拟化

3.1.1.1. 容器原生虚拟化的作用

容器原生虚拟化（container-native 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。

容器原生虚拟化通过 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。

3.1.1.2. 容器原生虚拟化支持

重要

容器原生虚拟化仅是一项技术预览功能。技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。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，请参阅 https://access.redhat.com/support/offerings/techpreview/。

3.1.2. 新增和改变的功能

3.1.2.1. Web 控制台的改进

- OpenShift Container Platform 仪表板收集有关集群的高级别信息。在 OpenShift Container Platform Web 控制台中，点击 **Home → Dashboards → Overview** 访问仪表板。请注意，Web 控制台项目概述中不再列出虚拟机。虚拟机现在列在 **Cluster Inventory** 仪表板卡中。

3.1.2.2. 其他改进

- 安装容器原生虚拟化后，MVM 池管理器会自动启动。如果您在不指定 MAC 地址的情况下定义了第二个 NIC，则 MAC 池管理器会为 NIC 分配一个唯一的 MAC 地址。

注意

如果您使用特定 MAC 地址定义了二级 NIC，则 MAC 地址可能会与集群中的另一个 NIC 冲突。

3.1.3. 已解决的问题

- 以前，如果您使用 web 控制台创建与现有虚拟机名称相同的虚拟机模板，则操作会失败。这会导致出现 **Name is already used by another virtual machine**信息。此问题已在容器原生虚拟化 2.1 中解决。(BZ#1717802)
- 以前，如果您通过以 **bridge** 模式连接的 Pod 网络创建虚拟机并使用 **cloud-init** 磁盘，则虚拟机重启后会断开网络连接。此问题已在容器原生虚拟化 2.1 中解决。(BZ#1708680)

3.1.4. 已知问题

- 虚拟机的 **masquerade** 绑定方法不能用于包含 RHEL 7 计算节点的集群。(BZ#1741626)
- 当在容器原生虚拟化安装过程中创建 **KubeVirt HyperConverged Cluster Operator Deployment** 自定义资源时，会显示一个带有不正确值的 YAML 文件。该文件类似于以下示例：

```
apiVersion: hco.kubevirt.io/v1alpha1
kind: HyperConverged
metadata:
  name: kubevirt-hyperconverged
  namespace: openshift-cnv
spec:
  BareMetalPlatform: 'false' 1
```

1 **'false'** 的单引号不正确。在点 **Create** 前，需要编辑这个文件来使行读取 **BareMetalPlatform: false**。如果没有删除引号，则部署将无法成功。(BZ#1767167)

- 当通过 web 控制台中的 **Disks** 选项卡向虚拟机添加磁盘时，添加的磁盘总是具有 **Filesystem** 的 volumeMode，而不考虑 **kubevirt-storage-class-default** ConfigMap 中设置的 volumeMode。(BZ#1753688)
- 迁移后，会为虚拟机分配一个新的 IP 地址。但是，**oc get vmi** 和 **oc describe vmi** 命令仍会生成包含过时 IP 地址的输出。(BZ#1686208)
  - 作为临时解决方案，请运行以下命令来查看正确的 IP 地址：

```
$ oc get pod -o wide
```

- 对于没有管理员特权的用户，虚拟机向导不会加载。此问题是由于缺少允许用户加载网络附加定义的权限造成的。(BZ#1743985)
  - 作为临时解决方案，请为用户提供加载网络附加定义的权限。
    1. 使用以下示例将 **ClusterRole** 和 **ClusterRoleBinding** 对象定义到 YAML 配置文件：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: cni-resources
rules:
```

```
- apiGroups: ["k8s.cni.cncf.io"]
 resources: ["*"]
 verbs: ["*"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: <role-binding-name>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cni-resources
subjects:
- kind: User
  name: <user to grant the role to>
  namespace: <namespace of the user>
```

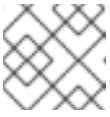
2. 作为 **cluster-admin** 用户，运行以下命令来创建您定义的 **ClusterRole** 和 **ClusterRoleBinding** 对象：

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

- 当导航到 **Virtual Machines Console** 选项卡时，有时不会显示任何内容。作为临时解决方案，请使用串行控制台。(BZ#1753606)
- 当您尝试从浏览器中列出容器原生虚拟化 Operator 的所有实例时，您会收到 404（未找到页面）错误。(BZ#1757526)
  - 作为临时解决方案，请运行以下命令：

```
$ oc get pods -n openshift-cnv | grep operator
```

- 移除容器原生虚拟化时，仍会不当保留某些资源。您必须手动删除这些资源以便重新安装容器原生虚拟化。(BZ#1712429)、BZ#1757705)
  - 作为临时解决方案，请遵循此流程：从容器原生虚拟化 2.1 卸载中残留的资源
- 如果虚拟机使用有保证的 CPU，则可不调度，因为节点上不会自动设置 **cpumanager=true** 标签。为解决这一问题，请从 **kubevirt-config** ConfigMap 中移除 **CPUManager** 条目。然后，为节点手动添加 **cpumanager=true** 标签，然后在您的集群上运行有保证 CPU 的虚拟机。(BZ#1718944)
- 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，请在 **kubevirt-config** ConfigMap 中设置默认 CPU 型号，如下例所示：



注意

您必须在启动支持实时迁移的虚拟机前进行此更改。

1. 运行以下命令，打开 **kubevirt-config** ConfigMap 以进行编辑：

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

2. 编辑 ConfigMap：

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

- 由于 Operator Lifecycle Manager (OLM) 的中断，容器原生虚拟化升级过程偶尔会失败。造成此问题的原因是使用声明性 API 来跟踪容器原生虚拟化 Operator 状态具有局限性。在安装过程中启用自动更新降低了遇到此问题的风险。(BZ#1759612)

- 容器原生虚拟化无法可靠识别由运行 **oc adm drain** 或 **kubectl drain** 触发的节点排空。不要在部署容器原生虚拟化的任何集群节点上运行这些命令。节点上如有虚拟机正在运行，则不会排空。当前解决方案为将节点设置为维护模式。(BZ#1707427)

- 当运行 **virtctl image-upload** 以 **qcow2** 格式上传大型虚拟机磁盘镜像时，在传输数据后可能会报告一个文件终止 (EOF) 错误，即使上传正常或完成。(BZ#1754920)
  运行以下命令，检查指定 PVC 中上传的状态：

```
$ oc describe pvc <pvc-name> | grep cdi.kubevirt.io/storage.pod.phase
```