Red Hat build of MicroShift 4.16

安装

安装和配置 MicroShift 集群

Last Updated: 2024-07-02

Red Hat build of MicroShift 4.16 安装

安装和配置 MicroShift 集群

法律通告

摘要

本文档提供有关安装 MicroShift 以及一些配置流程的详细信息。

目录

第 1 章 从 RPM 软件包安装

您可以在带有受支持的 Red Hat Enterprise Linux (RHEL)版本的机器上从 RPM 软件包安装 MicroShift。

1.1. 安装 MICROSHIFT 的系统要求

在安装 MicroShift 之前必须满足以下条件：

- 兼容 RHEL 或 RHEL for Edge 版本。
- AArch64 或 x86_64 系统架构。
- 2 个 CPU 内核.
- 用于 MicroShift 或 3 GB RAM 的 2 GB RAM，供 RHEL 用于基于网络的 HTTP 或 FTP 安装。
- 10 GB 存储。
- 在您的红帽帐户上有一个活跃的 MicroShift 订阅。如果您没有相关订阅，请联络您的销售代表以获得更多信息。
- 您有一个逻辑卷管理器 (LVM) 卷组 (VG)，它具有足够容量的工作负载持久性卷 (PV)。

1.2. 兼容性表

计划将受支持的 RHEL for Edge 版本与您使用的 MicroShift 版本配对，如以下兼容性表所述：

Red Hat Device Edge 发行版本兼容性列表

Red Hat Enterprise Linux (RHEL)和 MicroShift 可以一起工作，作为设备边缘计算的单一解决方案。您可以单独更新每个组件，但产品版本必须兼容。例如，MicroShift 从 4.14 更新至 4.16 需要 RHEL 更新。如下表所示，Red Hat Device Edge 的支持的配置为每个 Red Hat Device Edge 使用验证的版本：

RHEL for Edge 版本	MicroShift 版本	MicroShift 发行版本状态	支持的 MicroShift 版本→MicroShift 版本更新
9.4	4.16	正式发布	4.16.0→4.16.z, 4.14→4.16 和 4.15→4.16
9.2, 9.3	4.15	正式发布	4.15.0→4.15.z, 4.14→4.15 和 4.15→4.16
9.2, 9.3	4.14	正式发布	4.14.0→4.14.z, 4.14→4.15 和 4.14→4.16
9.2	4.13	技术预览	None
8.7	4.12	开发者预览	None

1.3. 从 RPM 软件包安装 MICROSHIFT 前

在为内存配置和 FIPS 模式安装 MicroShift 前，建议准备主机机器。

1.3.1. 配置卷组

MicroShift 使用逻辑卷管理器存储(LVMS)容器存储(CSI)插件，为持久性卷(PV)提供存储。LVMS 依赖于 Linux 逻辑卷管理器(LVM)来动态管理 PV 的后备逻辑卷(LV)。因此，您的计算机必须具有带有未使用空间的 LVM 卷组(VG)，其中 LVMS 可以为您的工作负载的 PV 创建 LV。

要配置卷组(VG)，允许 LVMS 为您的工作负载的 PV 创建 LV，请在安装 RHEL 时降低 root 卷的 **Desired Size**。降低根卷的大小可让磁盘上的未分配空间用于 LVMS 在运行时创建的额外 LV。

1.3.2. 为 FIPS 模式准备

如果您的用例需要以 FIPS 模式运行 MicroShift 容器，您必须安装启用了 FIPS 的 RHEL。在将 worker 机器配置为以 FIPS 模式运行后，您的 MicroShift 容器会自动配置为也以 FIPS 模式运行。

重要

因为 FIPS 必须在集群首次启动的操作系统前启用，所以您不能在部署集群后启用 FIPS。

其他资源

- 在 MicroShift 中使用 FIPS 模式

1.4. 准备从 RPM 软件包安装 MICROSHIFT

将您的 RHEL 机器配置为具有足够容量的逻辑卷管理器(LVM)卷组(VG)，用于工作负载的持久性卷(PV)。

先决条件

- 满足安装 MicroShift 的系统要求。
- 有对机器的 root 用户访问权限。
- 您已配置了带有工作负载 PV 所需的容量的 LVM VG。

流程

1. 在 **Storage Configuration** 子部分的 **Installation Destination** 下的图形安装程序中，选择 **Custom → Done** 打开用于配置分区和卷的对话框。此时会显示 Manual Partitioning 窗口。
2. 在 **New Red Hat Enterprise Linux 9.x Installation**下，选择 **Click here to create them automatically**。
3. 选择根分区 **/**，减少 **Desired Capacity**，以便 VG 有足够的容量供您 PV，然后点 **Update Settings**。
4. 完成安装。

注意

有关分区配置的更多信息，请参阅 配置手动分区的附加信息部分中的链接。

5. 以 root 用户身份，运行以下命令来验证系统上可用的 VG 容量：

```
$ sudo vgs
```

输出示例：

```
VG   #PV #LV #SN Attr   VSize    VFree
rhel   1   2   0 wz--n- <127.00g 54.94g
```

其他资源

- 从 Red Hat Hybrid Cloud 控制台下载 pull secret。
- 配置 MicroShift.
- 有关分区配置的更多信息，请参阅 配置手动分区。
- 有关调整现有 LV 大小以释放 VG 中的容量的更多信息，请参阅 管理 LVM 卷组。
- 有关创建 VG 和 PV 的更多信息，请阅读 逻辑卷管理概述。

1.5. 从 RPM 软件包安装红帽 MICROSHIFT 构建

使用以下步骤从 RPM 软件包安装 MicroShift 的红帽构建。

先决条件

- 满足安装红帽构建的 MicroShift 的系统要求。
- 您已完成了准备从 RPM 软件包安装红帽构建的 MicroShift 的步骤。

流程

1. 作为 root 用户，运行以下命令来启用 MicroShift 存储库的红帽构建：

```
$ sudo subscription-manager repos \
    --enable rhocp-4.16-for-rhel-9-$(uname -m)-rpms \
    --enable fast-datapath-for-rhel-9-$(uname -m)-rpms
```

2. 运行以下命令安装 MicroShift 的红帽构建：

```
$ sudo dnf install -y microshift
```

3. 将安装 pull secret 从 Red Hat Hybrid Cloud Console 下载到临时文件夹，例如 **$HOME/openshift-pull-secret**。此 pull secret 允许您通过提供红帽构建的 MicroShift 使用的容器镜像的容器 registry 进行身份验证。

4. 要将 pull secret 复制到 RHEL 机器的 **/etc/crio** 文件夹，请运行以下命令：

```
$ sudo cp $HOME/openshift-pull-secret /etc/crio/openshift-pull-secret
```

5. 运行以下命令使 **/etc/crio/openshift-pull-secret** 文件的所有者成为 root 用户：

```
$ sudo chown root:root /etc/crio/openshift-pull-secret
```

6. 运行以下命令，使 **/etc/crio/openshift-pull-secret** 文件可由 root 用户读取和写入：

```
$ sudo chmod 600 /etc/crio/openshift-pull-secret
```

7. 如果您的 RHEL 机器启用了防火墙，您必须配置几个必需的防火墙规则。对于 **firewalld**，运行以下命令：

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=10.42.0.0/16
```

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=169.254.169.1
```

```
$ sudo firewall-cmd --reload
```

如果您为红帽构建的 MicroShift 准备的卷组(VG)使用默认名称 **rhel**，则不需要进一步配置。如果您使用其他名称，或者要更改更多配置设置，请参阅 配置红帽构建的 MicroShift 部分。

1.6. 安装可选软件包

安装 MicroShift 时，可以添加可选 RPM 软件包。可选 RPM 示例包括用于扩展您的网络、添加和管理操作器以及管理应用程序的用户。使用以下步骤添加您需要的软件包。

1.6.1. 从 RPM 软件包安装 Operator Lifecycle Manager (OLM)

安装 MicroShift 时，Operator Lifecycle Manager (OLM)软件包不会被默认安装。您可以使用 RPM 软件包在 MicroShift 实例上安装 OLM。

流程

1. 运行以下命令来安装 OLM 软件包：

```
$ sudo dnf install microshift-olm
```

2. 要将软件包中的清单应用到活跃集群，请运行以下命令：

```
$ sudo systemctl restart microshift
```

其他资源

- 在 MicroShift 中使用 Operator Lifecycle Manager

1.6.2. 从 RPM 软件包安装 GitOps Argo CD 清单

您可以使用 MicroShift 的 OpenShift GitOps 的轻量级版本来帮助管理应用程序。使用 RPM 软件包安装所需的 Argo CD 清单。此 RPM 软件包包含运行核心 Argo CD 所需的清单。

重要

此过程会安装基本的 GitOps 功能。MicroShift 不提供 Argo CD CLI。

先决条件

- 已安装 MicroShift 版本 4.14 或更高版本
- 建议额外 RAM 存储为 250MB

流程

1. 运行以下命令，使用订阅管理器启用 GitOps 存储库：

```
$ sudo subscription-manager repos --enable=gitops-1.12-for-rhel-9-$(uname -m)-rpms
```

2. 运行以下命令来安装 GitOps 软件包：

```
$ sudo dnf install -y microshift-gitops
```

3. 要部署 Argo CD pod，请运行以下命令重启 MicroShift 服务：

```
$ sudo systemctl restart microshift
```

验证

- 您可以运行以下命令来验证 pod 是否在正确运行：

```
$ oc get pods -n openshift-gitops
```

输出示例

```
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          4m11s
argocd-redis-56844446bc-dzmhf       1/1     Running   0          4m12s
argocd-repo-server-57b4f896cf-7qk8l 1/1     Running   0          4m12s
```

其他资源

- 使用 GitOps 控制器自动执行应用程序管理

1.6.3. 安装多个网络插件

使用这个流程安装 MicroShift Multus CNI 插件以及新的 MicroShift 安装。MicroShift Multus Container Network Interface (CNI)插件不会被默认安装。如果要将额外网络附加到 pod 以进行高性能网络配置，请安装 **microshift-multus** RPM 软件包。

重要

不支持卸载 MicroShift Multus CNI。

流程

- 运行以下命令来安装 Multus RPM 软件包：

```
$ sudo dnf install microshift-multus
```

提示

如果在完成 MicroShift 安装时创建自定义资源(CR)，您可以避免重启服务以应用它们。

后续步骤

1. 继续您的新 MicroShift 安装，包括任何附加组件。
2. 创建 MicroShift Multus CNI 插件所需的自定义资源(CR)。
3. 根据需要配置其他网络 CNI。
4. 安装完您要包含的所有 RPM 后，启动 MicroShift 服务。MicroShift Multus CNI 插件会自动部署。

其他资源

- 关于使用多个网络

1.7. 启动和停止 MICROSHIFT

安装所需的所有 RPM 软件包后，了解如何启动和停止 MicroShift 服务。

1.7.1. 启动 MicroShift 服务

使用以下步骤启动 MicroShift 服务。

先决条件

- 您已从 RPM 软件包安装了 MicroShift。

流程

1. 作为 root 用户，输入以下命令启动 MicroShift 服务：

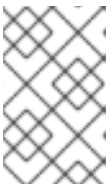
```
$ sudo systemctl start microshift
```

2. 可选：要将 RHEL 机器配置为在机器启动时启动 MicroShift，请输入以下命令：

```
$ sudo systemctl enable microshift
```

3. 可选：要在机器启动时自动启动 MicroShift，请输入以下命令：

```
$ sudo systemctl disable microshift
```



注意

MicroShift 服务首次启动时，它会下载并初始化 MicroShift 的容器镜像。因此，microShift 可能需要几分钟才能首次部署该服务。后续的开始 MicroShift 服务会缩短启动时间。

1.7.2. 停止 MicroShift 服务

使用以下步骤停止 MicroShift 服务。

先决条件

- MicroShift 服务正在运行。

流程

1. 输入以下命令停止 MicroShift 服务：

```
$ sudo systemctl stop microshift
```

2. MicroShift 上部署的工作负载可能会继续运行，即使 MicroShift 服务已停止。输入以下命令显示正在运行的工作负载：

```
$ sudo crictl ps -a
```

3. 输入以下命令停止部署的工作负载：

```
$ sudo systemctl stop kubepods.slice
```

1.7.3. 如何访问 MicroShift 集群

使用本节中的步骤，使用 OpenShift CLI (**oc**)访问 MicroShift 集群。

- 您可以从运行 MicroShift 服务的同一机器或从远程位置访问集群。
- 您可以使用此访问权限来观察和管理工作负载。
- 使用以下步骤时，选择包含您要连接到的主机名或 IP 地址的 **kubeconfig** 文件，并将其放在相关目录中。

其他资源

- 安装 OpenShift CLI 工具.

1.7.4. 本地访问 MicroShift 集群

使用以下步骤使用 **kubeconfig** 文件在本地访问 MicroShift 集群。

先决条件

- 已安装 **oc** 二进制文件。

流程

1. 可选：如果您的 RHEL 机器没有 **~/.kube/** 文件夹，请运行以下命令：

```
$ mkdir -p ~/.kube/
```

2. 运行以下命令，将生成的本地访问 **kubeconfig** 文件复制到 **~/.kube/** 目录中：

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. 运行以下命令更新 ~/**.kube/config** 文件的权限：

```
$ chmod go-r ~/.kube/config
```

验证

- 输入以下命令验证 MicroShift 是否正在运行：

```
$ oc get all -A
```

1.7.5. 打开防火墙以远程访问 MicroShift 集群

使用以下步骤打开防火墙，以便远程用户可以访问 MicroShift 集群。必须在 workstation 用户可以访问集群前完成此步骤。

对于此过程，**user@microshift** 是 MicroShift 主机上的用户，负责设置该机器，使其可以被单独的工作站上的远程用户访问。

先决条件

- 已安装 **oc** 二进制文件。
- 您的帐户具有集群管理特权。

流程

- 在 MicroShift 主机上以 **user@microshift** 的身份，运行以下命令来打开 Kubernetes API 服务器的防火墙端口 (**6443/tcp**)：

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

验证

- 以 **user@microshift** 的身份，输入以下命令验证 MicroShift 是否正在运行：

```
[user@microshift]$ oc get all -A
```

1.7.6. 远程访问 MicroShift 集群

使用以下步骤使用 **kubeconfig** 文件从远程位置访问 MicroShift 集群。

user@workstation 登录用于远程访问主机计算机。该流程中的 **<user>** 值是 **user@workstation** 登录到 MicroShift 主机所使用的用户名。

先决条件

- 已安装 **oc** 二进制文件。
- **user@microshift** 已从本地主机打开防火墙。

流程

1. 以 **user@workstation** 的身份，运行以下命令，创建 **~/.kube/** 文件夹：

```
[user@workstation]$ mkdir -p ~/.kube/
```

2. 以 **user@workstation** 的身份，运行以下命令来为您的 MicroShift 主机的主机名设置变量：

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. 以 **user@workstation** 的身份，通过运行以下命令复制生成的 **kubeconfig** 文件，其中包含您要从运行 MicroShift 的 RHEL 机器连接到您的本地机器的主机名或 IP 地址：

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```

注意

要为此步骤生成 **kubeconfig** 文件，请参阅为远程访问生成额外的 kubeconfig 文件。

4. 以 **user@workstation** 的身份，运行以下命令来更新 **~/.kube/config** 文件的权限：

```
$ chmod go-r ~/.kube/config
```

验证

- 以 **user@workstation** 的身份，输入以下命令验证 MicroShift 是否正在运行：

```
[user@workstation]$ oc get all -A
```

第 2 章 在 MICROSHIFT 中使用 FIPS 模式

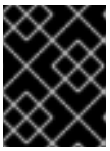
您可以在 Red Hat Enterprise Linux (RHEL) 9 的基于 RPM 的安装中使用 FIPS 模式。

- 要在 MicroShift 容器中启用 FIPS 模式，必须在机器启动前启用 worker 机器内核以 FIPS 模式运行。
- 不支持在 Red Hat Enterprise Linux for Edge (RHEL for Edge)镜像中使用 FIPS。

2.1. 基于 RHEL RPM 的安装的 FIPS 模式

在 Red Hat Enterprise Linux (RHEL)安装中使用 FIPS 需要启用加密模块自我检查。将主机操作系统配置为开始使用 FIPS 模块后，MicroShift 容器会自动启用在 FIPS 模式下运行。

- 当 RHEL 以 FIPS 模式启动时，MicroShift 核心组件使用 RHEL 加密库，这些库在 x86_64 架构上被提交给 NIST 进行 FIPS 140-2/140-3 验证。
- 在计划用作 worker 机器的机器上安装 RHEL 9 时，您必须启用 FIPS 模式。



重要

因为 FIPS 必须在集群首次启动的操作系统前启用，所以您不能在部署集群后启用 FIPS。

- MicroShift 使用 FIPS 兼容的 Golang 编译器。
- CRI-O 容器运行时支持 FIPS。

2.1.1. 限制

- TLS 实现 FIPS 支持没有完成。
- FIPS 实现不提供单个函数来计算哈希函数并验证基于该哈希的键。在以后的 MicroShift 版本中，仍会评估这个限制以改进。

2.1.2. 在 FIPS 模式中安装 RHEL

要使用 FIPS 安装 RHEL，请按照在 RHEL 文档的 FIPS 模式下安装系统 的指导操作。

2.2. 其他资源

- 在 FIPS 模式中安装系统
- 在容器中启用 FIPS 模式
- 联邦信息处理标准 140 和 FIPS 模式

第 3 章 为断开连接的安装镜像容器镜像

在断开连接的网络中部署 MicroShift 时，您可以使用自定义容器 registry。通过从私有 registry 中的镜像容器镜像安装集群，可以在没有直接互联网连接的受限网络中运行集群。

3.1. 将容器镜像镜像(MIRROR)到现有的 REGISTRY 中

使用自定义 air-gapped 容器 registry 或镜像(mirror)需要某些用户环境和工作负载要求。镜像允许传输容器镜像并更新到 air-gapped 环境，可在 MicroShift 实例上安装它们。

要为 MicroShift 容器创建 air-gapped 镜像 registry，您必须完成以下步骤：

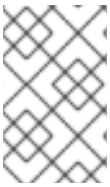
- 获取要镜像的容器镜像列表。
- 配置镜像先决条件。
- 在可访问互联网的主机上下载镜像。
- 将下载的镜像目录复制到 air-gapped 站点。
- 在 air-gapped 站点中将镜像上传到镜像 registry。
- 配置 MicroShift 主机以使用镜像 registry。

其他资源

- 为 Red Hat OpenShift 创建带有 mirror registry 的 mirror registry

3.2. 获取镜像 REGISTRY 容器镜像列表

要使用镜像 registry，您必须知道哪些容器镜像引用被特定的 MicroShift 版本使用。这些引用在 **microshift -release-info RPM 软件包一部分的 release-<arch>.json** 文件中提供。



注意

要在断开连接的环境中镜像 Operator Lifecycle Manager (OLM)，请添加 **microshift-olm** -olm RPM 中包含的 **release-olm-$ARCH.json** 中提供的引用，并按照相同的步骤操作。使用 **oc-mirror** 来镜像 Operator 目录和 Operator。

先决条件

- 已安装 jq。

流程

1. 使用以下方法之一访问容器镜像引用列表：
   - 如果在 MicroShift 主机上安装该软件包，请运行以下命令来获取文件的位置：

     ```
     $ rpm -ql microshift-release-info
     ```

     输出示例

     ```
     /usr/share/microshift/release/release-x86_64.json
     ```

- 如果没有在 MicroShift 主机上安装软件包，请在不安装 RPM 软件包的情况下下载并解包 RPM 软件包：

```
$ rpm2cpio microshift-release-info*.noarch.rpm | cpio -idmv
```

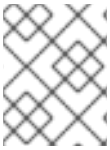
输出示例

```
/usr/share/microshift/release/release-x86_64.json
```

2. 运行以下命令，将容器镜像列表提取到 **microshift-container-refs.txt** 文件中：

```
$ RELEASE_FILE=/usr/share/microshift/release/release-$(uname -m).json
```

```
$ jq -r '.images | .[]' ${RELEASE_FILE} > microshift-container-refs.txt
```



注意

在使用 MicroShift 容器镜像列表创建 **microshift-container-refs.txt** 文件后，您可以在运行镜像步骤前将该文件附加到其他特定于用户的镜像引用中。

3.3. 配置镜像先决条件

您必须创建一个容器镜像 registry 凭证文件，允许从互联网连接的镜像主机镜像到 air-gapped 镜像。按照"配置凭证以允许镜像（mirror"链接中的内容在"Additional resources"部分中提供。这些说明指导您在镜像 registry 主机上创建一个 ~/**.pull-secret-mirror.json** 文件，其中包含用于访问镜像的用户凭证。

3.3.1. 镜像 registry pull secret 条目示例

例如，以下部分添加到使用 **microshift:microshift** 作为用户名和密码的 **microshift_quay:8443** 镜像 registry 的 pull secret 文件中。

pull secret 文件的镜像 registry 部分示例

```
"<microshift_quay:8443>": { 1
   "auth": "<microshift_auth>", 2
   "email": "<microshift_quay@example.com>" 3
},
```

1 将 &**lt;registry_host>:<port** > 值 **microshift_quay:8443** 替换为镜像 registry 服务器的主机名和端口。

2 将 &**lt;microshift_auth** > 值替换为用户密码。

3 将 &**lt;/microshift_quay@example.com** > 值替换为用户电子邮件。

其他资源

- 配置允许对容器镜像进行镜像的凭证

3.4. 下载容器镜像

找到容器列表并完成镜像先决条件后，将容器镜像下载到可访问互联网的主机。

先决条件

- 您已登录到可访问互联网的主机。
- 您已确保 **.pull-secret-mirror.json** 文件和 **microshift-containers** 目录内容在本地可用。

流程

1. 运行以下命令，安装用于复制容器镜像的 **skopeo** 工具：

```
$ sudo dnf install -y skopeo
```

2. 设置指向 pull secret 文件的环境变量：

```
$ PULL_SECRET_FILE=~/.pull-secret-mirror.json
```

3. 设置指向容器镜像列表的环境变量：

```
$ IMAGE_LIST_FILE=~/microshift-container-refs.txt
```

4. 设置指向存储下载数据的目的地目录的环境变量：

```
$ IMAGE_LOCAL_DIR=~/microshift-containers
```

5. 运行以下脚本将容器镜像下载到 **${IMAGE_LOCAL_DIR}** 目录中：

```
while read -r src_img ; do
  # Remove the source registry prefix
  dst_img=$(echo "${src_img}" | cut -d '/' -f 2-)

  # Run the image download command
  echo "Downloading '${src_img}' to '${IMAGE_LOCAL_DIR}'"
  mkdir -p "${IMAGE_LOCAL_DIR}/${dst_img}"
  skopeo copy --all --quiet \
    --preserve-digests \
    --authfile "${PULL_SECRET_FILE}" \
    docker://"${src_img}" dir://"${IMAGE_LOCAL_DIR}/${dst_img}"

done < "${IMAGE_LIST_FILE}"
```

6. 将镜像集传送到目标环境，如 air-gapped 站点。然后，您可以将镜像集上传到镜像 registry。

3.5. 将容器镜像上传到镜像 REGISTRY

要在 air-gapped 站点中使用容器镜像，请使用以下步骤将它们上传到镜像 registry。

先决条件

- 您已登录到一个可访问 **microshift-quay** 的主机。

- **.pull-secret-mirror.json** 文件在本地可用。
- **microshift-containers** 目录内容在本地可用。

流程

1. 运行以下命令，安装用于复制容器镜像的 **skopeo** 工具：

```
$ sudo dnf install -y skopeo
```

2. 设置环境变量指向 pull secret 文件：

```
$ IMAGE_PULL_FILE=~/.pull-secret-mirror.json
```

3. 设置环境变量指向本地容器镜像目录：

```
$ IMAGE_LOCAL_DIR=~/microshift-containers
```

4. 设置环境变量，指向镜像 registry URL 以上传容器镜像：

```
$ TARGET_REGISTRY=<registry_host>:<port> 1
```

1 将 **<registry_host>**:<port> 替换为镜像 registry 服务器的主机名和端口。

5. 运行以下命令，将容器镜像上传到 **${TARGET_REGISTRY}** 镜像 registry：

```
image_tag=mirror-$(date +%y%m%d%H%M%S)
image_cnt=1
  # Uses timestamp and counter as a tag on the target images to avoid
  # their overwrite by the 'latest' automatic tagging

pushd "${IMAGE_LOCAL_DIR}" >/dev/null
while read -r src_manifest ; do
  # Remove the manifest.json file name
  src_img=$(dirname "${src_manifest}")
  # Add the target registry prefix and remove SHA
  dst_img="${TARGET_REGISTRY}/${src_img}"
  dst_img=$(echo "${dst_img}" | awk -F'@' '{print $1}')

  # Run the image upload command
  echo "Uploading '${src_img}' to '${dst_img}'"
  skopeo copy --all --quiet \
    --preserve-digests \
    --authfile "${IMAGE_PULL_FILE}" \
    dir://"${IMAGE_LOCAL_DIR}/${src_img}" docker://"${dst_img}:${image_tag}-
${image_cnt}"
  # Increment the counter
  (( image_cnt += 1 ))

done < <(find . -type f -name manifest.json -printf '%P\n')
popd >/dev/null
```

3.6. 配置主机以进行镜像 REGISTRY 访问

要将 MicroShift 主机配置为使用镜像 registry，您必须创建一个将 Red Hat registry 主机名映射到镜像的配置文件，为 MicroShift 主机授予 registry 的访问权限。

先决条件

- 您的镜像主机可访问互联网。
- 镜像主机可以访问镜像 registry。
- 已将镜像 registry 配置为在受限网络中使用。
- 您下载了 pull secret 并已修改为包含镜像存储库的身份验证。

流程

1. 登录到您的 MicroShift 主机。
2. 通过完成以下步骤，在任何访问镜像 registry 的主机上启用 SSL 证书信任：
   a. 将 **rootCA.pem** 文件从镜像 registry （如 < **registry_path>/quay-rootCA** ）复制到位于 **/etc/pki/ca-trust/source/anchors** 目录的 MicroShift 主机。
   b. 运行以下命令，在系统范围的信任存储配置中启用证书：

```
$ sudo update-ca-trust
```

3. 创建 **/etc/containers/registries.conf.d/999-microshift-mirror.conf** 配置文件，将 Red Hat registry 主机名映射到镜像 registry：

镜像配置文件示例

```
[[registry]]
   prefix = ""
   location = "<registry_host>:<port>" (1)
   mirror-by-digest-only = true
   insecure = false

[[registry]]
   prefix = ""
   location = "quay.io"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false

[[registry]]
   prefix = ""
   location = "registry.redhat.io"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false
```

```
[[registry]]
    prefix = ""
    location = "registry.access.redhat.com"
    mirror-by-digest-only = true
[[registry.mirror]]
    location = "<registry_host>:<port>"
    insecure = false
```

1 将 **<registry_host>:<port** > 替换为镜像 registry 服务器的主机名和端口，例如 < **microshift-quay:8443** >。

4. 运行以下命令来启用 MicroShift 服务：

```
$ sudo systemctl enable microshift
```

5. 运行以下命令来重启主机：

```
$ sudo reboot
```

第 4 章 在 RHEL FOR EDGE 镜像中嵌入

您可以将 MicroShift 嵌入到 Red Hat Enterprise Linux for Edge (RHEL for Edge)镜像中。使用本指南构建包含 MicroShift 的 RHEL 镜像。

4.1. 安装 MICROSHIFT 的系统要求

在安装 MicroShift 之前必须满足以下条件：

- 兼容 RHEL 或 RHEL for Edge 版本。
- AArch64 或 x86_64 系统架构。
- 2 个 CPU 内核.
- 用于 MicroShift 或 3 GB RAM 的 2 GB RAM，供 RHEL 用于基于网络的 HTTP 或 FTP 安装。
- 10 GB 存储。
- 在您的红帽帐户上有一个活跃的 MicroShift 订阅。如果您没有相关订阅，请联络您的销售代表以获得更多信息。
- 您有一个逻辑卷管理器 (LVM) 卷组 (VG)，它具有足够容量的工作负载持久性卷 (PV)。

计划使用与 MicroShift 版本搭配使用的 RHEL 版本，如下表所述。

4.2. 兼容性表

计划将受支持的 RHEL for Edge 版本与您使用的 MicroShift 版本配对，如以下兼容性表所述：

Red Hat Device Edge 发行版本兼容性列表

Red Hat Enterprise Linux (RHEL)和 MicroShift 可以一起工作，作为设备边缘计算的单一解决方案。您可以单独更新每个组件，但产品版本必须兼容。例如，MicroShift 从 4.14 更新至 4.16 需要 RHEL 更新。如下表所示，Red Hat Device Edge 的支持的配置为每个 Red Hat Device Edge 使用验证的版本：

RHEL for Edge 版本	MicroShift 版本	MicroShift 发行版本状态	支持的 MicroShift 版本→MicroShift 版本更新
9.4	4.16	正式发布	4.16.0→4.16.z, 4.14→4.16 和 4.15→4.16
9.2, 9.3	4.15	正式发布	4.15.0→4.15.z, 4.14→4.15 和 4.15→4.16
9.2, 9.3	4.14	正式发布	4.14.0→4.14.z, 4.14→4.15 和 4.14→4.16
9.2	4.13	技术预览	None
8.7	4.12	开发者预览	None

4.3. 准备镜像构建

阅读 制作、安装和管理 RHEL for Edge 镜像 。

要为给定 CPU 架构构建 Red Hat Enterprise Linux for Edge (RHEL for Edge) 9.4 镜像，您需要一个满足 Image Builder 系统要求 的相同 CPU 架构的 RHEL 9.4 构建主机。

按照安装镜像构建器中的说明 安装镜像构建器 和 **composer-cli** 工具。

4.4. 将 MICROSHIFT 存储库添加到镜像构建器

使用以下步骤将 MicroShift 存储库添加到构建主机上的 Image Builder 中。

先决条件

- 您的构建主机满足 Image Builder 系统要求。
- 已安装并设置 Image Builder 和 **composer-cli** 工具。
- 有对构建主机的 root 用户访问权限。

流程

1. 运行以下命令，创建一个镜像构建器配置文件，以添加拉取 MicroShift RPM 所需的 **rhocp-4.16** RPM 存储库源：

```
cat > rhocp-4.16.toml <<EOF
id = "rhocp-4.16"
name = "Red Hat OpenShift Container Platform 4.16 for RHEL 9"
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/rhocp/4.16/os"
check_gpg = true
check_ssl = true
system = false
rhsm = true
EOF
```

2. 运行以下命令，创建镜像构建器配置文件来添加 **fast-datapath** RPM 存储库：

```
cat > fast-datapath.toml <<EOF
id = "fast-datapath"
name = "Fast Datapath for RHEL 9"
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/fast-datapath/os"
check_gpg = true
check_ssl = true
system = false
rhsm = true
EOF
```

3. 运行以下命令，将源添加到镜像构建器中：

```
$ sudo composer-cli sources add rhocp-4.16.toml
```

```
$ sudo composer-cli sources add fast-datapath.toml
```

验证

- 运行以下命令确认源是否已正确添加：

```
$ sudo composer-cli sources list
```

输出示例

```
appstream
baseos
fast-datapath
rhocp-4.16
```

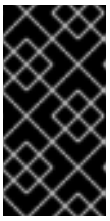
其他资源

- 镜像构建器系统要求
- 安装镜像构建器

4.5. 将 MICROSHIFT 服务添加到蓝图中

将 MicroShift RPM 软件包添加到镜像构建器蓝图中，可以使用嵌入 MicroShift 的 RHEL for Edge 镜像构建。

- 从第 1 步开始，创建自己的最小蓝图文件，这会导致更快地安装 MicroShift。
- 从第 2 步开始，使用生成的蓝图进行安装，其中包括所有 RPM 软件包和容器镜像。这是一个较长的安装过程，但启动更快，因为容器引用在本地访问。



重要

- 将以下流程中的 < *microshift_blueprint.toml* > 替换为您要使用的 TOML 文件的名称。
- 将以下流程中的 & *lt;microshift_blueprint* > 替换为您要用于蓝图的名称。

流程

1. 使用以下示例创建自己的蓝图文件：

自定义镜像构建器蓝图示例

```
cat > <microshift_blueprint.toml> <<EOF (1)
name = "<microshift_blueprint>" (2)

description = ""
version = "0.0.1"
modules = []
groups = []
```

```
[[packages]]
name = "microshift"
version = "*"

[customizations.services]
enabled = ["microshift"]
EOF
```

1. *<microshift_blueprint.toml>* 是 TOML 文件的名称。
2. *<microshift_blueprint>* 是蓝图的名称。

注意

命令中的通配符 * 使用最新的 MicroShift RPM。如果您需要特定版本，请替换您想要版本的通配符。例如，插入 **4.16.0** 以下载 MicroShift 4.16.0 RPM。

2. 可选。使用特定于您的平台架构的 **/usr/share/microshift/blueprint** 目录中安装的蓝图。有关 blueprint 部分的说明，请参见以下示例片断：

生成的镜像构建器蓝图示例片断

```
name = "microshift_blueprint"
description = "MicroShift 4.16.1 on x86_64 platform"
version = "0.0.1"
modules = []
groups = []

[[packages]] 1
name = "microshift"
version = "4.16.1"
...
...

[customizations.services] 2
enabled = ["microshift"]

[customizations.firewall]
ports = ["22:tcp", "80:tcp", "443:tcp", "5353:udp", "6443:tcp", "30000-32767:tcp", "30000-
32767:udp"]
...
...

[[containers]] 3
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:f41e79c17e8b41f1b0a5a32c3e2dd7cd15b8274554d3f1ba12b2598a347475f4"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:dbc65f1fba7d92b36cf7514cd130fe83a9bd211005ddb23a8dc479e0eea645fd"
...
…
EOF
```

1. 使用与 **microshift-release-info** RPM 兼容的相同版本的所有非可选 MicroShift RPM 软件包的引用。
2. 在系统启动时自动启用 MicroShift 并应用默认网络设置的引用。
3. 对离线部署所需的所有非可选 MicroShift 容器镜像的引用。

3. 运行以下命令，将蓝图添加到镜像构建器中：

```
$ sudo composer-cli blueprints push <microshift_blueprint.toml> 1
```

1. 将 <*microshift_blueprint.toml* > 替换为 TOML 文件的名称。

验证

1. 运行以下命令，验证 Image Builder 配置只列出 MicroShift 软件包：

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> | grep microshift 1
```

1. 将 <*microshift_blueprint* > 替换为蓝图的名称。

输出示例

```
blueprint: microshift_blueprint v0.0.1
    microshift-greenboot-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.noarch
    microshift-networking-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.x86_64
    microshift-release-info-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.noarch
    microshift-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.x86_64
    microshift-selinux-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.noarch
```

2. 可选：运行以下命令来验证 Image Builder 配置列出了要安装的所有组件：

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> 1
```

1. 将 <*microshift_blueprint* > 替换为蓝图的名称。

4.6. 在蓝图中添加其他软件包

在 OSTree 蓝图中添加可选 RPM 软件包的引用以启用它们。

先决条件

- 您创建了 Image Builder 蓝图文件。

流程

1. 运行以下命令来编辑 OSTree 蓝图：

```
$ vi <microshift_blueprint.toml> 1
```

1. 将 *<microshift_blueprint.toml>* 替换为用于 MicroShift 服务的蓝图文件的名称。

2. 在蓝图中添加以下示例文本：

```
[[packages]] 1
name = "<microshift-additional-package-name>" 2
version = "*"
```

1. 为每个要添加的其他服务包括一个小节。
2. 将 *<microshift-additional-package-name>* 替换为您要包含的服务的 RPM 名称。例如，**microshift-olm**。

后续步骤

1. 根据需要，将自定义证书颁发机构添加到蓝图中。
2. 在蓝图中添加后，您可以通过构建新的 OSTree 系统并将其部署到客户端，将清单应用到活跃集群：
   - 创建 ISO。
   - 添加蓝图并构建 ISO。
   - 下载 ISO 并做好使用准备。
   - 执行所需的任何配置。

其他资源

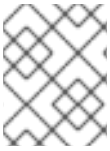
- 蓝图参考
- 使用镜像构建器 CLI 创建 RHEL for Edge 容器蓝图
- 构建 OSTree 镜像
- 安装 Podman

4.7. 添加证书颁发机构捆绑包

MicroShift 在客户端评估服务器证书时使用主机信任捆绑包。您还可以使用自定义的安全证书链来改进端点证书与特定于部署的客户端的兼容性。要做到这一点，您可以将带有 root 和中间证书的证书颁发机构(CA)捆绑包添加到 Red Hat Enterprise Linux for Edge (RHEL for Edge)系统范围信任存储中。

4.7.1. 将证书颁发机构捆绑包添加到 rpm-ostree 镜像

您可以将额外的可信证书颁发机构(CA)包含在 Red Hat Enterprise Linux for Edge (RHEL for Edge) **rpm-ostree** 镜像中，方法是将它们添加到用于创建镜像的蓝图中。在从镜像 registry 中拉取镜像时，使用以下步骤设置操作系统信任的额外 CA。



注意

此流程要求您在蓝图中配置 CA 捆绑包自定义，然后在 kickstart 文件中添加步骤以启用捆绑包。在以下步骤中，**data** 是键，< **value&** gt; 代表 PEM 编码的证书。

先决条件

- 有访问构建主机的 root 用户。
- 您的构建主机满足 Image Builder 系统要求。
- 已安装并设置 Image Builder 和 **composer-cli** 工具。

流程

1. 在蓝图中添加以下自定义值以添加目录。
   a. 在您的构建镜像的主机上为蓝图添加指令，以创建目录，例如： **/etc/pki/ca-trust/source/anchors/** 用于证书捆绑包。

```
[[customizations.directories]]
path = "/etc/pki/ca-trust/source/anchors"
```

   b. 镜像引导后，创建证书捆绑包，如 **/etc/pki/ca-trust/source/anchors/cert1.pem** ：

```
[[customizations.files]]
path = "/etc/pki/ca-trust/source/anchors/cert1.pem"
data = "<value>"
```

2. 要在系统范围的信任存储配置中启用证书捆绑包，请在您要使用的镜像的主机上使用 **update-ca-trust** 命令，例如：

```
$ sudo update-ca-trust
```

注意

update-ca-trust 命令可能包含在用于 MicroShift 主机安装的 kickstart 文件的 **%post** 部分中，以便在第一次引导时启用所有必需的证书信任。在向 kickstart 文件中添加步骤前，您必须在蓝图中配置 CA 捆绑包自定义，以启用捆绑包。

```
%post
# Update certificate trust storage in case new certificates were
# installed at /etc/pki/ca-trust/source/anchors directory
update-ca-trust
%end
```

其他资源

- 创建 RHEL for Edge 镜像
- 使用共享系统证书(RHEL 9)
- 支持的镜像自定义(RHEL 9)

- 创建和管理 OSTree 镜像更新
- 在 OSTree 系统中应用更新

4.8. 创建 RHEL FOR EDGE 镜像

使用以下步骤创建 ISO。RHEL for Edge Installer 镜像从正在运行的容器中拉取提交，并创建一个带有配置为使用嵌入式 **rpm-ostree** 提交的 Kickstart 文件的可安装的引导 ISO。

先决条件

- 您的构建主机满足 Image Builder 系统要求。
- 已安装并设置 Image Builder 和 **composer-cli** 工具。
- 有对构建主机的 root 用户访问权限。
- 已安装 **podman** 工具。

流程

1. 运行以下命令启动 **ostree** 容器镜像构建：

```
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/9/$(uname -m)/edge"
minimal-microshift edge-container | awk '{print $2}')
```

此命令还会返回要监控的构建的标识(ID)。

2. 您可以运行以下命令来定期检查构建的状态：

```
$ sudo composer-cli compose status
```

正在运行的构建的输出示例

```
ID                                   Status    Time                 Blueprint          Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332   RUNNING   Wed Jun 7 12:26:23 2023
minimal-microshift   0.0.1     edge-container
```

已完成构建的输出示例

```
ID                                   Status    Time                 Blueprint          Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332   FINISHED   Wed Jun 7 12:32:37 2023
minimal-microshift   0.0.1     edge-container
```

注意

如果您熟悉如何启动和停止它，您可以使用 **watch** 命令监控构建。

3. 运行以下命令，使用 ID 下载容器镜像，并获取可供使用的镜像：

```
$ sudo composer-cli compose image ${BUILDID}
```

4. 运行以下命令，将下载的容器镜像的所有权改为当前用户：

```
$ sudo chown $(whoami). ${BUILDID}-container.tar
```

5. 运行以下命令，在镜像中为当前用户添加读取权限：

```
$ sudo chmod a+r ${BUILDID}-container.tar
```

6. 通过完成以下步骤，在端口 8085 上引导服务器，供 ISO 构建使用 **ostree** 容器镜像：

   a. 运行以下命令来获取 **IMAGEID** 变量结果：

```
$ IMAGEID=$(cat < "./${BUILDID}-container.tar" | sudo podman load | grep -o -P '(?
<=sha256[@:])[a-z0-9]*')
```

   b. 运行以下命令，使用 **IMAGEID** 变量结果来执行 podman 命令步骤：

```
$ sudo podman run -d --name=minimal-microshift-server -p 8085:8080 ${IMAGEID}
```

   此命令还会返回在 **IMAGEID** 变量中保存的容器 ID，以进行监控。

7. 运行以下命令来生成安装程序蓝图文件：

```
cat > microshift-installer.toml <<EOF
name = "microshift-installer"

description = ""
version = "0.0.0"
modules = []
groups = []
packages = []
EOF
```

4.9. 将蓝图添加到镜像构建器并构建 ISO

1. 运行以下命令，将蓝图添加到镜像构建器中：

```
$ sudo composer-cli blueprints push microshift-installer.toml
```

2. 运行以下命令来启动 **ostree** ISO 构建：

```
$ BUILDID=$(sudo composer-cli compose start-ostree --url http://localhost:8085/repo/ --ref
"rhel/9/$(uname -m)/edge" microshift-installer edge-installer | awk '{print $2}')
```

   此命令还会返回要监控的构建的标识(ID)。

3. 您可以运行以下命令来定期检查构建的状态：

```
$ sudo composer-cli compose status
```

正在运行的构建的输出示例

```
ID                                   Status    Time                  Blueprint           Version   Type
Size
c793c24f-ca2c-4c79-b5b7-ba36f5078e8d   RUNNING   Wed Jun 7 13:22:20 2023
microshift-installer   0.0.0    edge-installer
```

已完成构建的输出示例

```
ID                                   Status    Time                  Blueprint           Version   Type
Size
c793c24f-ca2c-4c79-b5b7-ba36f5078e8d   FINISHED   Wed Jun 7 13:34:49 2023
microshift-installer   0.0.0    edge-installer
```

4.10. 下载 ISO 并为使用做准备

1. 运行以下命令，使用 ID 下载 ISO：

```
$ sudo composer-cli compose image ${BUILDID}
```

2. 运行以下命令，将下载的容器镜像的所有权改为当前用户：

```
$ sudo chown $(whoami). ${BUILDID}-installer.iso
```

3. 运行以下命令，在镜像中为当前用户添加读取权限：

```
$ sudo chmod a+r ${BUILDID}-installer.iso
```

4.11. 为 MICROSHIFT 置备机器

使用 RHEL for Edge 文档中的流程为 RHEL for Edge 镜像置备机器。

要使用 MicroShift，您必须置备系统，使其满足以下要求：

- 您调配的计算机必须满足安装 MicroShift 的系统要求。
- 文件系统必须有一个逻辑卷管理器 (LVM) 卷组 (VG)，且对工作负载的持久性卷 (PV) 有足够的容量。
- Red Hat Hybrid Cloud Console 中的 pull secret 必须显示为 **/etc/crio/openshift-pull-secret**，并具有 root 用户只读读/写权限。
- 防火墙必须使用所需的设置进行配置。

注意

如果您使用 Kickstart，如 RHEL for Edge Installer (ISO)镜像，您可以更新 Kickstart 文件来满足置备要求。

先决条件

1. 您已创建了包含带有红帽构建的 MicroShift 的 RHEL for Edge 提交的 RHEL for Edge 安装程序 (ISO)镜像。

   a. 此要求包括编写 RFE 容器镜像的步骤，创建 RFE 安装程序蓝图、启动 RFE 容器以及制作 RFE 安装程序镜像。

2. 创建一个 Kickstart 文件或使用现有的。在 Kickstart 文件中，您必须包括：

   a. 有关如何创建用户的详细信息。

   b. 如何获取和部署 RHEL for Edge 镜像。

如需更多信息，请参阅"附加资源"。

流程

1. 在 Kickstart 文件的主部分中，更新文件系统的设置，以便它包含一个名为 **rhel** 的 LVM 卷组，至少有 10GB 系统 root。LVMS CSI 驱动程序保留可用空间，用于存储您的工作负载的数据。

   用于配置文件系统的 kickstart 片段示例

```
# Partition disk such that it contains an LVM volume group called `rhel` with a
# 10GB+ system root but leaving free space for the LVMS CSI driver for storing data.
#
# For example, a 20GB disk would be partitioned in the following way:
#
# NAME          MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
# sda             8:0    0  20G  0 disk
# ├─sda1          8:1    0 200M  0 part /boot/efi
# ├─sda1          8:1    0 800M  0 part /boot
# └─sda2          8:2    0  19G  0 part
#   └─rhel-root  253:0    0  10G  0 lvm  /sysroot
#
ostreesetup --nogpg --osname=rhel --remote=edge \
--url=file:///run/install/repo/ostree/repo --ref=rhel/<RHEL VERSION NUMBER>/x86_64/edge
zerombr
clearpart --all --initlabel
part /boot/efi --fstype=efi --size=200
part /boot --fstype=xfs --asprimary --size=800
# Uncomment this line to add a SWAP partition of the recommended size
#part swap --fstype=swap --recommended
part pv.01 --grow
volgroup rhel pv.01
logvol / --vgname=rhel --fstype=xfs --size=10000 --name=root
# To add users, use a line such as the following
user --name=<YOUR_USER_NAME> \
--password=<YOUR_HASHED_PASSWORD> \
--iscrypted --groups=<YOUR_USER_GROUPS>
```

2. 在 Kickstart 文件的 **%post** 部分中，添加您的 pull secret 和强制防火墙规则。

   添加 pull secret 和防火墙规则的 Kickstart 片断示例

```
%post --log=/var/log/anaconda/post-install.log --erroronfail

# Add the pull secret to CRI-O and set root user-only read/write permissions
```

```
cat > /etc/crio/openshift-pull-secret << EOF
YOUR_OPENSHIFT_PULL_SECRET_HERE
EOF
chmod 600 /etc/crio/openshift-pull-secret

# Configure the firewall with the mandatory rules for MicroShift
firewall-offline-cmd --zone=trusted --add-source=10.42.0.0/16
firewall-offline-cmd --zone=trusted --add-source=169.254.169.1

%end
```

3. 运行以下命令安装 **mkksiso** 工具：

```
$ sudo yum install -y lorax
```

4. 运行以下命令，使用您的新 Kickstart 文件更新 ISO 中的 Kickstart 文件：

```
$ sudo mkksiso <your_kickstart>.ks <your_installer>.iso <updated_installer>.iso
```

其他资源

- RHEL for Edge 文档
- 安装 MicroShift 的系统要求
- Red Hat Hybrid Cloud Console pull secret
- 所需的防火墙设置
- 创建 Kickstart 文件
- 如何将 Kickstart 文件嵌入到 ISO 镜像中

4.12. 如何访问 MICROSHIFT 集群

使用本节中的步骤，使用 OpenShift CLI (**oc**)访问 MicroShift 集群。

- 您可以从运行 MicroShift 服务的同一机器或从远程位置访问集群。
- 您可以使用此访问权限来观察和管理工作负载。
- 使用以下步骤时，选择包含您要连接到的主机名或 IP 地址的 **kubeconfig** 文件，并将其放在相关目录中。

4.12.1. 本地访问 MicroShift 集群

使用以下步骤使用 **kubeconfig** 文件在本地访问 MicroShift 集群。

先决条件

- 已安装 **oc** 二进制文件。

流程

1. 可选：如果您的 RHEL 机器没有 ~/**.kube**/ 文件夹，请运行以下命令：

```
$ mkdir -p ~/.kube/
```

2. 运行以下命令，将生成的本地访问 **kubeconfig** 文件复制到 ~/**.kube**/ 目录中：

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. 运行以下命令更新 ~/**.kube**/**config** 文件的权限：

```
$ chmod go-r ~/.kube/config
```

验证

- 输入以下命令验证 MicroShift 是否正在运行：

```
$ oc get all -A
```

4.12.2. 打开防火墙以远程访问 MicroShift 集群

使用以下步骤打开防火墙，以便远程用户可以访问 MicroShift 集群。必须在 workstation 用户可以访问集群前完成此步骤。

对于此过程，**user@microshift** 是 MicroShift 主机上的用户，负责设置该机器，使其可以被单独的工作站上的远程用户访问。

先决条件

- 已安装 **oc** 二进制文件。
- 您的帐户具有集群管理特权。

流程

- 在 MicroShift 主机上以 **user@microshift** 的身份，运行以下命令来打开 Kubernetes API 服务器的防火墙端口 (**6443/tcp**)：

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

验证

- 以 **user@microshift** 的身份，输入以下命令验证 MicroShift 是否正在运行：

```
[user@microshift]$ oc get all -A
```

4.12.3. 远程访问 MicroShift 集群

使用以下步骤使用 **kubeconfig** 文件从远程位置访问 MicroShift 集群。

user@workstation 登录用于远程访问主机计算机。该流程中的 **<user>** 值是 **user@workstation** 登录到 MicroShift 主机所使用的用户名。

先决条件

- 已安装 **oc** 二进制文件。
- **user@microshift** 已从本地主机打开防火墙。

流程

1. 以 **user@workstation** 的身份，运行以下命令，创建 **~/.kube/** 文件夹：

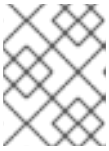
```
[user@workstation]$ mkdir -p ~/.kube/
```

2. 以 **user@workstation** 的身份，运行以下命令来为您的 MicroShift 主机的主机名设置变量：

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. 以 **user@workstation** 的身份，通过运行以下命令复制生成的 **kubeconfig** 文件，其中包含您要从运行 MicroShift 的 RHEL 机器连接到您的本地机器的主机名或 IP 地址：

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```



注意

要为此步骤生成 **kubeconfig** 文件，请参阅为远程访问生成额外的 kubeconfig 文件。

4. 以 **user@workstation** 的身份，运行以下命令来更新 **~/.kube/config** 文件的权限：

```
$ chmod go-r ~/.kube/config
```

验证

- 以 **user@workstation** 的身份，输入以下命令验证 MicroShift 是否正在运行：

```
[user@workstation]$ oc get all -A
```

其他资源

- 为远程访问生成额外的 kubeconfig 文件

第 5 章 在 RHEL FOR EDGE 镜像中嵌入以离线使用

在 **rpm-ostree** 提交中嵌入 MicroShift 容器意味着您可以在 air-gapped、断开连接或离线环境中运行集群。您可以将 MicroShift 容器的红帽构建嵌入到 Red Hat Enterprise Linux for Edge (RHEL for Edge)镜像中，以便容器引擎不需要从容器 registry 通过网络拉取镜像。工作负载可以在没有网络连接的情况下立即启动。

5.1. 安装 MICROSHIFT 的系统要求

在安装 MicroShift 之前必须满足以下条件：

- 兼容 RHEL 或 RHEL for Edge 版本。
- AArch64 或 x86_64 系统架构。
- 2 个 CPU 内核.
- 用于 MicroShift 或 3 GB RAM 的 2 GB RAM，供 RHEL 用于基于网络的 HTTP 或 FTP 安装。
- 10 GB 存储。
- 在您的红帽帐户上有一个活跃的 MicroShift 订阅。如果您没有相关订阅，请联络您的销售代表以获得更多信息。
- 您有一个逻辑卷管理器 (LVM) 卷组 (VG)，它具有足够容量的工作负载持久性卷 (PV)。

5.2. 兼容性表

计划将受支持的 RHEL for Edge 版本与您使用的 MicroShift 版本配对，如以下兼容性表所述：

Red Hat Device Edge 发行版本兼容性列表

Red Hat Enterprise Linux (RHEL)和 MicroShift 可以一起工作，作为设备边缘计算的单一解决方案。您可以单独更新每个组件，但产品版本必须兼容。例如，MicroShift 从 4.14 更新至 4.16 需要 RHEL 更新。如下表所示，Red Hat Device Edge 的支持的配置为每个 Red Hat Device Edge 使用验证的版本：

RHEL for Edge 版本	MicroShift 版本	MicroShift 发行版本状态	支持的 MicroShift 版本→MicroShift 版本更新
9.4	4.16	正式发布	4.16.0→4.16.z, 4.14→4.16 和 4.15→4.16
9.2, 9.3	4.15	正式发布	4.15.0→4.15.z, 4.14→4.15 和 4.15→4.16
9.2, 9.3	4.14	正式发布	4.14.0→4.14.z, 4.14→4.15 和 4.14→4.16
9.2	4.13	技术预览	None
8.7	4.12	开发者预览	None

5.3. 为离线部署嵌入 MICROSHIFT 容器

您可以使用 Image Builder 使用嵌入式 MicroShift 容器镜像创建 **rpm-ostree** 系统镜像。要嵌入容器镜像，您必须对镜像构建器蓝图添加镜像引用。

先决条件

- 有对构建主机的 root 用户访问权限。
- 您的构建主机满足 Image Builder 系统要求。
- 已安装并设置 Image Builder 和 **composer-cli** 工具。
- 您已创建了 RHEL for Edge 镜像蓝图。
- 已安装 jq。

流程

1. 获取您要部署的 MicroShift 版本使用的容器镜像引用列表。您可以按照以下步骤 2 安装 **microshift-release-info** RPM 软件包，或者按照以下步骤 3 下载并解包 RPM。
2. 安装 **microshift-release-info** RPM 软件包：
   a. 运行以下命令安装 **microshift-release-info** RPM 软件包：

   ```
   $ sudo dnf install -y microshift-release-info-<release_version>
   ```

   使用整个版本号，将 < **release_version** > 替换为您要部署的发行版本的数字值，如 **4.16.0**。

   b. 运行以下命令，列出 **/usr/share/microshift/release** 目录的内容，以验证发行信息文件是否存在：

   ```
   $ ls /usr/share/microshift/release
   ```

   输出示例

   ```
   release-x86_64.json
   release-aarch64.json
   ```

   如果安装了 **microshift-release-info** RPM，您可以继续第 4 步。
3. 如果您没有完成第 2 步，请下载并解包 **microshift-release-info** RPM，而不安装它：
   a. 运行以下命令来下载 RPM 软件包：

   ```
   $ sudo dnf download microshift-release-info-<release_version>
   ```

   使用整个版本号，将 < **release_version** > 替换为您要部署的发行版本的数字值，如 **4.16.0**。

   rpm 示例

   ```
   microshift-release-info-4.16.0.*.el9.noarch.rpm 1
   ```

1 * 代表日期和提交 ID。您的输出应同时包含 example - **202311101230.p0.g7dc6a00.assembly.4.16.0**。

b. 运行以下命令，在不安装 RPM 软件包的情况下解包 RPM 软件包：

```
$ rpm2cpio <my_microshift_release_info> | cpio -idmv 1
./usr/share/microshift/release/release-aarch64.json
./usr/share/microshift/release/release-x86_64.json
```

1 将 **<my_microshift_release_info** > 替换为上一步中的 RPM 软件包的名称。

4. 运行以下命令，定义包含容器引用信息的 JSON 文件的位置：

```
$ RELEASE_FILE=</path/to/your/release-$(uname -m).json>
```

将 **</path/to/your/release-$(uname -m).json** > 替换为 JSON 文件的完整路径。确保使用您的架构所需的文件。

5. 运行以下命令，定义 TOML 文件的位置，其中包含构建镜像的说明：

```
$ BLUEPRINT_FILE=</path/to/your/blueprint.toml>
```

将 **</path/to/your/blueprint.toml** > 替换为 JSON 文件的完整路径。

6. 运行以下命令，然后在蓝图 TOML 文件中嵌入容器镜像引用：

```
$  jq -r '.images | .[] | ("[[containers]]\nsource = \"" + . + "\"\n")' "${RELEASE_FILE}" >>
"${BLUEPRINT_FILE}"
```

生成的 &lt ;my_blueprint.toml> 片段示例，显示容器引用

```
[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:82cfef91557f9a70cff5a90accba45841a37524e9b93f98a97b20f6b2b69e5db"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:82cfef91557f9a70cff5a90accba45841a37524e9b93f98a97b20f6b2b69e5db"
```

7. 您可以使用以下示例手动将任何容器镜像添加到镜像构建器蓝图中：

手动将容器镜像嵌入到镜像构建器的部分示例

```
[[containers]]
source = "<my_image_pullspec_with_tag_or_digest>"
```

将 **<my_image_pullspec_with_tag_or_digest** > 替换为您要部署的 MicroShift 版本使用的容器镜像的确切引用。

5.4. 更新 OSBUILDER WORKER 配置以准备镜像构建

更新蓝图后，您必须更新 osbuilder worker 配置，以准备使用嵌入式 MicroShift 容器构建镜像。

先决条件

- 有对构建主机的 root 用户访问权限。
- 您的构建主机满足 Image Builder 系统要求。
- 已安装并设置 Image Builder 和 **composer-cli** 工具。

注意

如果不存在，您可以创建 **/etc/osbuild-worker/osbuild-worker.toml** 目录和配置文件。

流程

1. 通过在 **/etc/osbuild-worker/osbuild-worker.toml** osbuilder worker 配置文件的 **[containers]** 部分中设置 **auth_file_path**，添加用于向 registry 进行身份验证的 pull secret：

```
[containers]
auth_file_path = "/etc/osbuild-worker/pull-secret.json"
```

2. 重启 **osbuild-worker**，通过重启主机来应用配置更改。重启主机可确保当前运行的所有 **osbuild-worker** 服务都已重启。

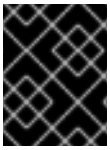
5.5. 构建并使用 RPM-OSTREE 镜像进行离线部署

您可以使用 Image Builder 使用嵌入式 MicroShift 容器镜像创建 **rpm-ostree** 系统镜像。要嵌入容器镜像，您必须对镜像构建器蓝图添加镜像引用。您可以根据需要创建您的用例的提交和 ISO。

将此处列出的先决条件添加到遵循的步骤中包含的先决条件。

5.5.1. 离线部署的额外先决条件

- 您已创建了和更新 RHEL for Edge 镜像蓝图，以便离线使用。以下流程使用通过容器镜像创建的蓝图示例。您必须使用您在 "Embedding MicroShift 容器中创建的更新蓝图进行离线部署"。
- 您已更新了 **/etc/osbuild-worker/osbuild-worker.toml** 配置文件以离线使用。



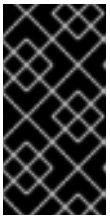
重要

将以下流程中的 **minimal-microshift.toml** 替换为您更新用于离线使用的 TOML 名称，<my_blueprint_name>。

5.5.2. 将 MicroShift 服务添加到蓝图中

将 MicroShift RPM 软件包添加到镜像构建器蓝图中，可以使用嵌入 MicroShift 的 RHEL for Edge 镜像构建。

- 从第 1 步开始，创建自己的最小蓝图文件，这会导致更快地安装 MicroShift。
- 从第 2 步开始，使用生成的蓝图进行安装，其中包括所有 RPM 软件包和容器镜像。这是一个较长的安装过程，但启动更快，因为容器引用在本地访问。



重要

- 将以下流程中的 < *microshift_blueprint.toml* > 替换为您要使用的 TOML 文件的名称。
- 将以下流程中的 &*lt;microshift_blueprint* > 替换为您要用于蓝图的名称。

流程

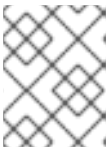
1. 使用以下示例创建自己的蓝图文件：

自定义镜像构建器蓝图示例

```
cat > <microshift_blueprint.toml> <<EOF (1)
name = "<microshift_blueprint>" (2)

description = ""
version = "0.0.1"
modules = []
groups = []

[[packages]]
name = "microshift"
version = "*"

[customizations.services]
enabled = ["microshift"]
EOF
```

(1) *<microshift_blueprint.toml* > 是 TOML 文件的名称。

(2) *<microshift_blueprint* > 是蓝图的名称。



注意

命令中的通配符 * 使用最新的 MicroShift RPM。如果您需要特定版本，请替换您想要版本的通配符。例如，插入 **4.16.0** 以下载 MicroShift 4.16.0 RPM。

2. 可选。使用特定于您的平台架构的 **/usr/share/microshift/blueprint** 目录中安装的蓝图。有关 blueprint 部分的说明，请参见以下示例片断：

生成的镜像构建器蓝图示例片断

```
name = "microshift_blueprint"
description = "MicroShift 4.16.1 on x86_64 platform"
version = "0.0.1"
modules = []
groups = []

[[packages]] (1)
name = "microshift"
version = "4.16.1"
...
```

```
...

[customizations.services] 2
enabled = ["microshift"]

[customizations.firewall]
ports = ["22:tcp", "80:tcp", "443:tcp", "5353:udp", "6443:tcp", "30000-32767:tcp", "30000-
32767:udp"]
...
...

[[containers]] 3
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:f41e79c17e8b41f1b0a5a32c3e2dd7cd15b8274554d3f1ba12b2598a347475f4"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:dbc65f1fba7d92b36cf7514cd130fe83a9bd211005ddb23a8dc479e0eea645fd"
...
…
EOF
```

1. 使用与 **microshift-release-info** RPM 兼容的相同版本的所有非可选 MicroShift RPM 软件包的引用。
2. 在系统启动时自动启用 MicroShift 并应用默认网络设置的引用。
3. 对离线部署所需的所有非可选 MicroShift 容器镜像的引用。

3. 运行以下命令，将蓝图添加到镜像构建器中：

```
$ sudo composer-cli blueprints push <microshift_blueprint.toml> 1
```

1. 将 *<microshift_blueprint.toml >* 替换为 TOML 文件的名称。

验证

1. 运行以下命令，验证 Image Builder 配置只列出 MicroShift 软件包：

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> | grep microshift 1
```

1. 将 *<microshift_blueprint >* 替换为蓝图的名称。

输出示例

```
blueprint: microshift_blueprint v0.0.1
    microshift-greenboot-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.noarch
    microshift-networking-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.x86_64
    microshift-release-info-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.noarch
    microshift-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.x86_64
    microshift-selinux-4.16.1-202305250827.p0.g4105d3b.assembly.4.16.1.el9.noarch
```

2. 可选：运行以下命令来验证 Image Builder 配置列出了要安装的所有组件：

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> 1
```

1 将 *<microshift_blueprint>* 替换为蓝图的名称。

5.5.3. 创建 RHEL for Edge 镜像

使用以下步骤创建 ISO。RHEL for Edge Installer 镜像从正在运行的容器中拉取提交，并创建一个带有配置为使用嵌入式 **rpm-ostree** 提交的 Kickstart 文件的可安装的引导 ISO。

先决条件

- 您的构建主机满足 Image Builder 系统要求。
- 已安装并设置 Image Builder 和 **composer-cli** 工具。
- 有对构建主机的 root 用户访问权限。
- 已安装 **podman** 工具。

流程

1. 运行以下命令启动 **ostree** 容器镜像构建：

```
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/9/$(uname -m)/edge"
minimal-microshift edge-container | awk '{print $2}')
```

此命令还会返回要监控的构建的标识(ID)。

2. 您可以运行以下命令来定期检查构建的状态：

```
$ sudo composer-cli compose status
```

正在运行的构建的输出示例

```
ID                                   Status    Time                 Blueprint          Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332   RUNNING   Wed Jun 7 12:26:23 2023
minimal-microshift   0.0.1     edge-container
```

已完成构建的输出示例

```
ID                                   Status    Time                 Blueprint          Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332   FINISHED   Wed Jun 7 12:32:37 2023
minimal-microshift   0.0.1     edge-container
```

注意

如果您熟悉如何启动和停止它，您可以使用 **watch** 命令监控构建。

3. 运行以下命令，使用 ID 下载容器镜像，并获取可供使用的镜像：

```
$ sudo composer-cli compose image ${BUILDID}
```

4. 运行以下命令，将下载的容器镜像的所有权改为当前用户：

```
$ sudo chown $(whoami). ${BUILDID}-container.tar
```

5. 运行以下命令，在镜像中为当前用户添加读取权限：

```
$ sudo chmod a+r ${BUILDID}-container.tar
```

6. 通过完成以下步骤，在端口 8085 上引导服务器，供 ISO 构建使用 **ostree** 容器镜像：

   a. 运行以下命令来获取 **IMAGEID** 变量结果：

```
$ IMAGEID=$(cat < "./${BUILDID}-container.tar" | sudo podman load | grep -o -P '(?
<=sha256[@:])[a-z0-9]*')
```

   b. 运行以下命令，使用 **IMAGEID** 变量结果来执行 podman 命令步骤：

```
$ sudo podman run -d --name=minimal-microshift-server -p 8085:8080 ${IMAGEID}
```

   此命令还会返回在 **IMAGEID** 变量中保存的容器 ID，以进行监控。

7. 运行以下命令来生成安装程序蓝图文件：

```
cat > microshift-installer.toml <<EOF
name = "microshift-installer"

description = ""
version = "0.0.0"
modules = []
groups = []
packages = []
EOF
```

5.6. 其他资源

- 将容器推送到 registry 中并将其嵌入到镜像中
- 容器 registry 凭证
- 为完全断开连接的主机配置网络设置
- 在 MicroShift 中使用 Operator Lifecycle Manager
- 使用 oc-mirror 插件创建自定义目录

第 6 章 GREENBOOT 健康检查框架

Greenboot 是 **rpm-ostree** 系统上 **systemd** 服务的通用健康检查框架，如 Red Hat Enterprise Linux for Edge (RHEL for Edge)。此框架包含在 MicroShift 安装中，带有 **microshift-greenboot** 和 **greenboot-default-health-checks** RPM 软件包。

Greenboot 健康检查会在不同时间运行来评估系统健康状况，并在软件出现问题的情况下自动回滚到最后一个健康状态，例如：

- **每次系统启动时都运行默认健康检查脚本。**
- **除了默认的健康检查外，您还可以编写、安装和配置应用程序健康检查脚本，以便在系统每次启动时也运行。**
- **Greenboot 可减少在更新过程中锁定边缘设备的风险，并防止在更新失败时中断服务。**
- **当检测到失败时，系统会使用 rpm-ostree 回滚功能引导至最后一个已知的工作配置中。此功能对于直接服务有限或不存在的边缘设备特别有用。**

MicroShift 应用程序健康检查脚本包含在 microshift-greenboot RPM 中。greenboot-default-health-checks RPM 包括健康检查脚本，验证 DNS 和 ostree 服务是否可以访问。您可以为正在运行的工作负载创建自己的健康检查脚本。您可以编写一个来验证应用程序是否已启动，例如：

6.1. GREENBOOT 如何使用目录运行脚本

从四个 /etc/greenboot 目录运行的健康检查脚本。这些脚本按字母顺序运行。当您为工作负载配置脚本时请注意这一点。

当系统启动时，Greenboot 在 required.d 目录中 运行脚本。根据这些脚本的结果，Greenboot 会继续启动或尝试回滚，如下所示：

1. **系统如预期：当 required.d 目录中的所有脚本都成功运行，Greenboot 会运行 /etc/greenboot/green.d 目录中的任何脚本。**
2. **系统问题：如果 required.d 目录中的任何脚本失败，Greenboot 会运行 red.d 目录中的任何 prerollback 脚本，然后重启系统。**

注意

greenboot 将脚本和健康检查输出重定向到系统日志。登录后，每日信息会提供整个系统健康输出。

6.1.1. Greenboot 目录详情

从任何脚本返回非零退出代码意味着该脚本失败。greenboot 在尝试回滚到之前的版本前，会多次重启系统以重试脚本。

- /etc/greenboot/check/required.d 包含不能失败的健康检查。
  - 如果脚本失败，Greenboot 默认重试三次。您可以通过将 GREENBOOT_MAX_BOOTS 参数设置为所需的重试次数，在 /etc/greenboot/greenboot.conf 文件中配置重试次数。
  - 在所有重试失败后，Greenboot 会在一个可用时自动启动回滚。如果没有回滚，系统日志输出显示需要手动干预。
  - MicroShift 的 40_microshift_running_check.sh 健康检查脚本被安装到这个目录中。
- /etc/greenboot/check/wanted.d 包含允许失败的健康脚本，而不会导致系统被回滚。
  - 如果这些脚本失败，Greenboot 会记录失败，但不会启动回滚。
- /etc/greenboot/green.d 包含在 Greenboot 声明启动成功后运行的脚本。
- /etc/greenboot/red.d 包含在 Greenboot 声明启动后运行的脚本，包括 40_microshift_pre_rollback.sh prerollback 脚本。这个脚本在系统回滚前执行。脚本执行 MicroShift pod 和 OVN-Kubernetes 清理，以避免在系统回滚到以前的版本后潜在的冲突。

6.2. MICROSHIFT 健康检查脚本

40_microshift_running_check.sh 健康检查脚本仅执行核心 MicroShift 服务的验证。在 Greenboot 目录中安装自定义工作负载健康检查脚本，以确保应用程序在系统更新后成功运行。脚本以字母顺序运

行。

MicroShift 健康检查在下表中列出：

表 6.1. MicroShift 的验证状态和结果

验证	Pass	Fail
检查脚本是否使用 root 权限运行	下一步	exit 0
检查 microshift.service 是否已启用	下一步	exit 0
等待 microshift.service 处于活动状态(!failed)	下一步	exit 1
等待 Kubernetes API 健康端点正常工作并接收流量	下一步	exit 1
等待任何 pod 启动	下一步	exit 1
对于每个核心命名空间，等待拉取镜像	下一步	exit 1
对于每个核心命名空间，等待 pod 就绪	下一步	exit 1
对于每个核心命名空间，检查 pod 是否没有重启	exit 0	exit 1

6.2.1. 验证等待周期

默认情况下，每个验证中的等待周期为五分钟。在等待时间后，如果验证没有成功，它将声明失败。每次在验证循环引导后，这个等待周期会递增增加。

- 您可以通过在 /etc/greenboot/greenboot.conf 配置文件中设置 MICROSHIFT_WAIT_TIMEOUT_SEC 环境变量来覆盖基础等待周期。例如，您可以通过将值重置为 180 秒（如 MICROSHIFT_WAIT_TIMEOUT_SEC=180 ）将等待时间更改为三分钟。

6.3. 启用 SYSTEMD 日志服务数据持久性

systemd 日志服务的默认配置将数据存储在易失性 /run/log/journal 目录中。要在系统启动并重启时查看系统日志，您必须启用日志持久性并设置最大日志数据大小的限制。

流程

1. 运行以下命令制作目录：

```
$ sudo mkdir -p /etc/systemd/journald.conf.d
```

2. 运行以下命令来创建配置文件：

```
cat <<EOF | sudo tee /etc/systemd/journald.conf.d/microshift.conf &>/dev/null
[Journal]
Storage=persistent
SystemMaxUse=1G
RuntimeMaxUse=1G
EOF
```

3. 根据您的大小要求编辑配置文件值。

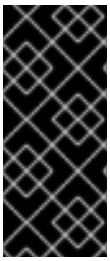
其他资源

- 自动应用清单

6.4. 更新和第三方工作负载

健康检查在更新后特别有用。您可以检查 Greenboot 健康检查的输出，并确定更新是否有效。此健康检查可帮助您确定系统是否正常工作。

健康检查脚本会安装到 /etc/greenboot/check/required.d 目录中，并在每次系统启动时都自动执行。退出具有非零状态的脚本意味着系统启动声明为失败。



重要

在启动第三方工作负载前，等待更新被声明为有效。如果在工作负载启动后执行回滚，可能会丢失数据。在更新完成前，有些第三方工作负载在设备上创建或更新数据。在回滚时，文件系统会在更新前恢复到其状态。

6.5. 检查更新的结果

成功启动后，Greenboot 在 GRUB 中将变量 boot_success= 设置为 1。您可以按照以下流程在系统日志中更新后查看系统健康检查的整体状态。

流程

- 要访问系统健康检查的整体状态，请运行以下命令：

```
$ sudo grub2-editenv - list | grep ^boot_success
```

成功系统启动的输出示例

```
boot_success=1
```

6.6. 访问系统日志中的健康检查输出

您可以按照以下流程手动访问系统日志中的健康检查输出。

流程

- 要访问健康检查的结果，请运行以下命令：

```
$ sudo journalctl -o cat -u greenboot-healthcheck.service
```

失败健康检查的输出示例

```
...
...
Running Required Health Check Scripts...
STARTED
GRUB boot variables:
boot_success=0
boot_indeterminate=0
boot_counter=2
...
...
Waiting 300s for MicroShift service to be active and not failed
```

```
FAILURE
...
...
```

6.7. 在系统日志中访问预滚动健康检查输出

您可以在系统日志中访问健康检查脚本的输出。例如，按照以下流程检查 prerollback 脚本的结果。

流程

- 要访问 prerollback 脚本的结果，请运行以下命令：

```
$ sudo journalctl -o cat -u redboot-task-runner.service
```

一个 prerollback 脚本的输出示例

```
...
...
Running Red Scripts...
STARTED
GRUB boot variables:
boot_success=0
boot_indeterminate=0
boot_counter=0
The ostree status:
* rhel c0baa75d9b585f3dd989a9cf05f647eb7ca27ee0dbd4b94fe8c93ed3a4b9e4a5.0
    Version: 9.1
    origin: <unknown origin type>
  rhel 6869c1347b0e0ba1bbf0be750cdf32da5138a1fcbc5a4c6325ab9eb647b64663.0 (rollback)
    Version: 9.1
    origin refspec: edge:rhel/9/x86_64/edge
System rollback imminent - preparing MicroShift for a clean start
Stopping MicroShift services
Removing MicroShift pods
Killing conmon, pause and OVN processes
Removing OVN configuration
Finished greenboot Failure Scripts Runner.
Cleanup succeeded
Script '40_microshift_pre_rollback.sh' SUCCESS
FINISHED
redboot-task-runner.service: Deactivated successfully.
```

6.8. 使用健康检查脚本检查更新

使用以下步骤在更新后访问系统日志中 Greenboot 健康检查脚本的输出。

流程

- 要访问更新检查的结果，请运行以下命令：

```
$ sudo grub2-editenv - list | grep ^boot_success
```

成功更新的输出示例

```
boot_success=1
```

如果您的命令返回 boot_success=0，则 Greenboot 健康检查仍然在运行，或者更新失败。

6.9. 其他资源

- Greenboot 工作负载健康检查脚本

第 7 章 安装问题的故障排除

要排除失败的 MicroShift 安装，您可以运行 sos 报告。使用 sos report 命令生成详细的报告，其中显示了系统中不同组件和应用程序中的所有已启用插件和数据。

7.1. 从 SOS 报告收集数据

先决条件

- 已安装 sos 软件包。

流程

1. 以 root 用户身份登录到失败的主机。
2. 运行以下命令执行 debug 报告创建步骤：

```
$ microshift-sos-report
```

输出示例

```
sosreport (version 4.5.1)

This command will collect diagnostic and configuration information from
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp/sos.o0sznf_8 and may be provided to a Red Hat support
representative.

Any information provided to Red Hat will be treated in accordance with
the published support policies at:

        Distribution Website : https://www.redhat.com/
        Commercial Support   : https://www.access.redhat.com/

The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.

No changes will be made to system configuration.
```

```
Setting up archive ...
Setting up plugins ...
Running plugins. Please wait ...

 Starting 1/2   microshift      [Running: microshift]
 Starting 2/2   microshift_ovn  [Running: microshift microshift_ovn]
 Finishing plugins              [Running: microshift]

 Finished running plugins

Found 1 total reports to obfuscate, processing up to 4 concurrently

sosreport-microshift-rhel9-2023-03-31-axjbyxw :    Beginning obfuscation...
sosreport-microshift-rhel9-2023-03-31-axjbyxw :    Obfuscation completed

Successfully obfuscated 1 report(s)

Creating compressed archive...

A mapping of obfuscated elements is available at
 /var/tmp/sosreport-microshift-rhel9-2023-03-31-axjbyxw-private_map

Your sosreport has been generated and saved in:
 /var/tmp/sosreport-microshift-rhel9-2023-03-31-axjbyxw-obfuscated.tar.xz

 Size 444.14KiB
 Owner root
 sha256 922e5ff2db25014585b7c6c749d2c44c8492756d619df5e9838ce863f83d4269

Please send this file to your support representative.
```

7.2. 其他资源

- 关于 MicroShift sos 报告
- 为技术支持生成 sos 报告