Red Hat Enterprise Linux 9

使用 RHEL 的镜像模式构建、部署和管理操作系统

在 Red Hat Enterprise Linux 9 上使用 RHEL 可启动的容器镜像

Last Updated: 2024-06-27

Red Hat Enterprise Linux 9 使用 RHEL 的镜像模式构建、部署和管理操作系统

在 Red Hat Enterprise Linux 9 上使用 RHEL 可启动的容器镜像

法律通告

摘要

RHEL 可启动的容器镜像使您能够构建、部署和管理操作系统，就像它是任何其他容器一样。您可以集中在单个容器原生工作流上，来管理从应用程序到底层操作系统的所有内容。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

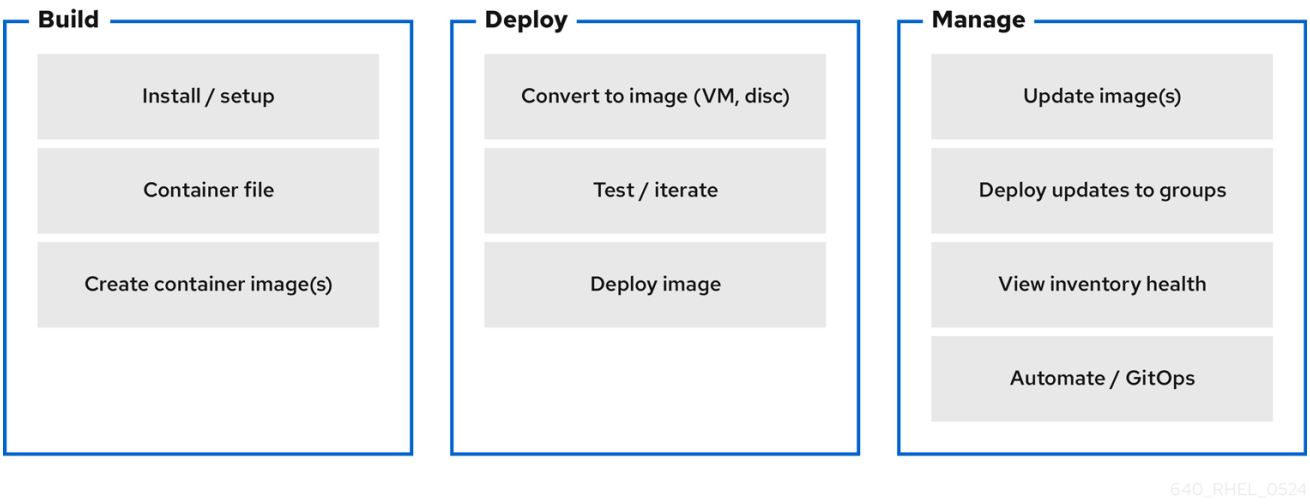
第 1 章 RHEL 的镜像模式简介

使用 RHEL 的镜像模式构建、测试，并使用与应用程序容器一样的工具和技术来部署操作系统。可使用 **registry.redhat.io/rhel9/rhel-bootc** 可引导容器镜像提供 RHEL 的镜像模式。RHEL 可引导的容器镜像与现有的应用程序通用基础镜像(UBI)不同，它们包含引导必需的其它组件，这些组件传统上被排除在外，如内核、initrd、引导装载程序、固件等。

重要

红帽将 **rhel9/rhel-bootc** 容器镜像作为技术预览提供。技术预览功能提供对未来产品创新的早期访问，使客户能够在开发过程中测试功能并提供反馈。但是，这些功能不被完全支持。技术预览功能的文档可能不完整，或者仅包含基本安装和配置信息。如需有关 技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

图 1.1. 使用 RHEL 的镜像模式构建、部署和管理操作系统



红帽为以下计算机架构提供可引导的容器镜像：

- AMD 和 Intel 64 位体系架构 (x86-64-v2)
- 64 位 ARM 架构(ARMv8.0-A)

RHEL 的镜像模式的好处贯穿于系统的整个生命周期。以下列表包含一些最重要的优点：

容器镜像比其他镜像格式更易于理解和使用，且可快速构建

Containerfiles,也称为 Dockerfile,提供了一种简单的方法来定义镜像的内容和构建指令。与其他镜像创建工具相比，构建和迭代容器镜像通常要快得多。

整合流程、基础架构和发布工件

因为您将应用程序作为容器分发，所以您可以使用同样的基础设施和流程来管理底层操作系统。

不可变的更新

正如容器化应用程序以不可变的方式更新一样，使用 RHEL 的镜像模式，操作系统也是如此。您可以以使用 **rpm-ostree** 系统的同样的方式引导到更新，并在需要时回滚。

警告

不支持使用 **rpm-ostree** 进行更改或安装内容。

跨混合云环境的可移植性

您可以在物理的、虚拟化的、云和边缘环境中使用可引导的容器镜像。

虽然容器提供构建、传输和运行镜像的基础，但重要的是要理解，在使用安装机制部署这些可引导容器镜像后，或将它们转换为磁盘镜像后，系统不会作为容器运行。

支持的镜像类型 如下：

- 容器镜像格式：OCI
- 磁盘镜像格式：
  - ISO
  - QEMU copy-on-write (QCOW2), Raw
  - Amazon 机器镜像 (AMI)
  - 虚拟机镜像(VMI)
  - 虚拟机磁盘(VMDK)

容器通过提供以下可能性来简化 RHEL 系统的生命周期：

构建容器镜像

您可以通过修改 Containerfile 来在构建时配置操作系统。RHEL 的镜像模式可使用 **registry.redhat.io/rhel9/rhel-bootc** 容器镜像提供。您可以使用 Podman、OpenShift Container Platform 或其他标准容器构建工具来管理容器和容器镜像。您可以使用 CI/CD 管道自动化构建过程。

版本控制、镜像和测试容器镜像

您可以使用任何容器工具，如 Podman 或 OpenShift Container Platform 来版本化、镜像化、反省和签署派生的可引导容器镜像。

将容器镜像部署到目标环境中

有几个如何部署镜像的选项：

- **Anaconda**: 是 RHEL 使用的安装程序。您可以使用 Anaconda 和 Kickstart 将所有镜像类型部署到目标环境中，以自动化安装过程。
- **bootc-image-builder** ：是一种容器化工具，它将容器镜像转换为不同类型的磁盘镜像，并可选择将它们上传到镜像注册中心或对象存储中。
- **bootc**: 是一个负责从容器注册中心获取容器镜像，并将其安装到系统、更新操作系统，或从现有的基于 ostree 的系统进行切换的工具。RHEL bootable 容器镜像默认包含 **bootc** 工具，并适用于所有镜像类型。但是，请记住不支持 **rpm-ostree**，且不得用于进行更改。

更新您的操作系统

系统支持部署后带回滚的就地事务更新。默认情况下，自动更新是开启的。systemd 服务单元和

systemd 定时器单元文件检查容器注册中心中的更新，并将其应用到系统。由于更新是事务性的，所以需要重新启动。对于需要更复杂的或计划部署的环境，请禁用自动更新，并使用 **bootc** 工具更新您的操作系统。

RHEL 有两种部署模式。两者都在部署期间提供同样的稳定性、可靠性和性能。

1. **软件包模式** ：操作系统使用 RPM 软件包，并使用 **dnf** 软件包管理器进行更新。root 文件系统是可变的。

2. **镜像模式** ：一种构建、部署和管理 RHEL 的容器原生方法。同样的 RPM 软件包作为基础镜像提供，更新作为容器镜像部署。root 文件系统默认是不可变的，但 **/etc** 和 **/var** 除外，大多数内容来自容器镜像。

您可以使用两种部署模式，以与任何其他容器化应用程序相同的方式来构建、测试、共享、部署和管理操作系统。

1.1. 先决条件

- 您有一个订阅的 RHEL 9 系统。如需更多信息，请参阅 RHEL 系统注册文档。
- 您有一个容器注册中心。您可以在本地创建注册中心，或者在 Quay.io 服务上创建一个免费帐户。要创建 Quay.io 帐户，请参阅 Red Hat Quay.io 页面。
- 您有一个具有生产或开发人员订阅的红帽帐户。Red Hat Enterprise Linux Overview 页面上提供了免费的开发人员订阅。
- 您已认证到 registry.redhat.io 。如需更多信息，请参阅 红帽容器注册中心身份验证 文章。

1.2. 其他资源

- 介绍 Podman Desktop 中的 RHEL 的镜像模式和可引导容器 快速入门指南
- Red Hat Enterprise Linux 的镜像模式的快速入门：AI 推理 快速入门指南
- Podman AI Lab 入门 博客文章
- 自定义 Anaconda 产品文档
- 执行高级 RHEL 9 安装 产品文档(Kickstart)
- 创建自定义 RHEL 系统镜像 产品文档
- 准备、安装和管理 RHEL for Edge 镜像 产品文档

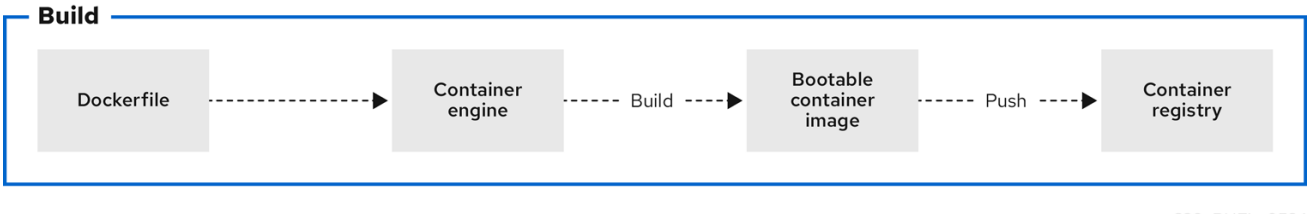
第 2 章 构建和测试 RHEL 可引导容器镜像

以下流程使用 Podman 来构建和测试您的容器镜像。您也可以使用其他工具，如 OpenShift Container Platform。有关使用容器配置 RHEL 系统的更多信息，请参阅 rhel-bootc-examples 存储库。

重要

红帽将 **rhel9/rhel-bootc** 容器镜像作为技术预览提供。技术预览功能提供对未来产品创新的早期访问，使客户能够在开发过程中测试功能并提供反馈。但是，这些功能不被完全支持。技术预览功能的文档可能不完整，或者仅包含基本安装和配置信息。如需有关 技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

图 2.1. 使用 Containerfile 中的指令构建镜像，测试容器，将镜像推送到 registry，并与其他人共享



639_RHEL_0524

通用 **Containerfile** 结构如下：

```
FROM registry.redhat.io/rhel9/rhel-bootc:latest

RUN dnf -y install [software] [dependencies] && dnf clean all

ADD [application]
ADD [configuration files]

RUN [config scripts]
```

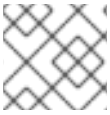
rhel-9-bootc 容器镜像重复利用 OCI 镜像格式。

- **rhel-9-bootc** 容器镜像在安装到系统时会忽略容器配置部分（**Config** ）。
- 当使用容器运行时（如 **podman** 或 **docker** ）运行此镜像时，**rhel-9-bootc** 容器镜像不会忽略容器配置部分（**Config**）。

例如，当 **rhel-9-bootc** 镜像安装到系统时，**Containerfile** 中的以下命令会被忽略：

- **ENTRYPOINT** 和 **CMD** (OCI: **Entrypoint**/**Cmd**)：您可以设置 **CMD /sbin/init**。
- **ENV** (OCI: **Env**): 更改 **systemd** 配置以配置全局系统环境。
- **EXPOSE** (OCI: **exposePorts**)：它独立于系统防火墙和运行时网络功能。
- **USER** (OCI: **User**)：在 RHEL 可启动容器中配置各个服务，来以非特权用户身份运行。

可以在 **Containerfile** 和 **Dockerfile** 中使用的命令一样。



注意

此发行版本不支持构建自定义 **rhel-bootc** 基础镜像。

2.1. 构建容器镜像

使用 **podman build** 命令，使用 **Containerfile** 中的指令构建镜像。

先决条件

- **container-tools** 元数据包已安装。

流程

1. 创建 **Containerfile** ：

```
$ cat Containerfile
FROM registry.redhat.io/rhel9/rhel-bootc:latest
RUN dnf -y install cloud-init && \
    ln -s ../cloud-init.target /usr/lib/systemd/system/default.target.wants && \
    dnf clean all
```

这个 **Containerfile** 示例添加了 **cloud-init** 工具，因此它可以自动获取 SSH 密钥，并运行基础设施中的脚本，也可以从实例元数据中收集配置和 secret。例如，您可以将此容器镜像用于预生成的 AWS 或 KVM 客户机操作系统。

2. 使用当前目录中的 **Containerfile** 构建 ***<image>*** 镜像：

```
$ podman build -t quay.io/<namespace>/<image>:<tag>
```

验证

- 列出所有镜像：

```
$ podman images
REPOSITORY                  TAG     IMAGE ID      CREATED             SIZE
localhost/<image>           latest  b28cd00741b3  About a minute ago  2.1 GB
```

其他资源

- 使用容器注册中心
- 使用 Buildah 从 Containerfile 构建镜像

2.2. 运行容器镜像

使用 **podman run** 命令运行并测试您的容器。

先决条件

- **container-tools** 元数据包已安装。

流程

- 根据 **quay.io/*<namespace>*/*<image>*:*<tag>*** 容器镜像运行名为 **mybootc** 的容器：

```
$ podman run -it --rm --name mybootc quay.io/<namespace>/<image>:<tag> /bin/bash
```

  - **-i** 选项创建一个交互式会话。如果不使用 **-t** 选项，shell 将保持打开状态，但您无法对 shell 输入任何东西。
  - **-t** 选项打开一个终端会话。如果不使用 **-i** 选项，shell 会打开，然后退出。
  - 容器退出后，**--rm** 选项删除 **quay.io/*<namespace>*/*<image>*:*<tag>*** 容器镜像。

验证

- 列出所有正在运行的容器：

```
$ podman ps
CONTAINER ID  IMAGE                                COMMAND         CREATED        STATUS
PORTS   NAMES
7ccd6001166e  quay.io/<namespace>/<image>:<tag>  /sbin/init  6 seconds ago  Up 5
seconds ago          mybootc
```

其他资源

- podman run 命令

2.3. 将容器镜像推送到注册中心

使用 **podman push** 命令将镜像推送到您自己的或第三方注册中心，并与其他人共享。以下流程使用 Red Hat Quay 注册中心。

先决条件

- **container-tools** 元数据包已安装。
- 镜像在本地系统上构建并提供。
- 您已创建了 Red Hat Quay 注册中心。如需更多信息，请参阅 概念验证 - 部署 Red Hat Quay 。

流程

- 将 **quay.io/*<namespace>*/*<image>*:*<tag>*** 容器镜像从本地存储推送到注册中心：

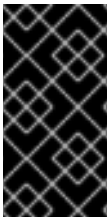
```
$ podman push quay.io/<namespace>/<image>:<tag>
```

其他资源

- 重新分发 UBI 镜像

第 3 章 使用 BOOTC-IMAGE-BUILDER 创建 BOOTC 兼容基本磁盘镜像

作为技术预览提供的 **bootc-image-builder** 是一种容器化工具，来从可引导容器镜像创建磁盘镜像。您可以使用您构建的镜像来在不同的环境（如边缘、服务器和云）中部署磁盘镜像。



重要

红帽将 **bootc-image-builder** 工具作为技术预览提供。技术预览功能提供对未来产品创新的早期访问，使客户能够在开发过程中测试功能并提供反馈。但是，这些功能不被完全支持。技术预览功能的文档可能不完整，或者仅包含基本安装和配置信息。如需有关 技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

3.1. 用于 BOOTC-IMAGE-BUILDER 的 RHEL 的镜像模式介绍

使用 **bootc-image-builder** 工具，您可以将可引导容器镜像转换为磁盘镜像，以用于各种平台和格式。将可引导容器镜像转换为磁盘镜像等同于安装可引导容器。将这些磁盘镜像部署到目标环境后，您可以直接从容器注册中心更新它们。

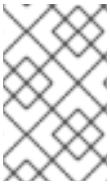
注意

此发行版本不支持使用 **bootc-image-builder** 构建来自私有注册中心的基本磁盘镜像。

bootc-image-builder 工具支持生成以下镜像类型：

- 磁盘镜像格式，如 ISO，适用于断开连接的安装。
- 虚拟磁盘镜像格式，例如：
  - QEMU copy-on-write (QCOW2)
  - Amazon 机器镜像(AMI)/– Raw
  - 虚拟机镜像(VMI)

运行虚拟机或服务器时从容器镜像部署是有好处的，因为您可以获得同样的安装结果。从同一容器镜像构建它们时，这种一致性可跨越多个不同镜像类型和平台。因此，您可以在平台间维护操作系统镜像时最小化工作量。您还可以使用 **bootc** 工具更新从这些磁盘镜像部署的系统，而不必使用 **bootc-image-builder** 重新创建和上传新磁盘镜像。



注意

通用基础容器镜像不包括任何默认密码或 SSH 密钥。另外，使用 **bootc-image-builder** 工具创建的磁盘镜像不包含通用磁盘镜像中提供的工具，如 **cloud-init**。这些磁盘镜像只是转换的容器镜像。

虽然您可以直接部署 **rhel-9-bootc** 镜像，但您也可以创建从这个可引导基础镜像派生的自己的自定义镜像。**bootc-image-builder** 工具使用 **rhel-9-bootc** OCI 容器镜像作为输入。

其他资源

- 使用 cloud-init 的红帽产品

3.2. 安装 BOOTC-IMAGE-BUILDER

bootc-image-builder 旨在用作容器，它不作为 RHEL 中的 RPM 软件包提供。要访问它，请按照以下流程操作。

先决条件

- **container-tools** 元数据软件包已安装。元数据软件包包含所有容器工具，如 Podman、Buildah 和 Skopeo。
- 您已认证到 **registry.redhat.io**。详情请参阅 红帽容器注册中心身份验证。

流程

1. 登录以认证到 **registry.redhat.io** ：

```
$ sudo podman login registry.redhat.io
```

2. 安装 **bootc-image-builder** 工具：

```
$ sudo podman pull registry.redhat.io/rhel9/bootc-image-builder
```

验证

- 列出拉取到本地系统的所有镜像：

```
$ sudo podman images
REPOSITORY                                  TAG         IMAGE ID      CREATED       SIZE
registry.redhat.io/rhel9/bootc-image-builder  latest      b361f3e845ea  24 hours ago  676 MB
```

其他资源

- Red Hat Container Registry 身份验证
- 从 registry 中拉取镜像

3.3. 使用 BOOTC-IMAGE-BUILDER 创建 QCOW2 镜像

为在其上运行命令的架构，将 RHEL 可引导容器镜像(QCOW2)镜像构建到 QEMU 磁盘镜像(QCOW2)镜像中。

RHEL 基础镜像不包括默认用户。可选，您可以使用 **--config** 选项注入用户配置来运行 bootc-image-builder 容器。或者，您可以使用 **cloud-init** 配置基础镜像，以便在第一次引导时注入用户和 SSH 密钥。请参阅 使用 cloud-init 来注入用户和 SSH 密钥 。

先决条件

- 您已在主机机器上安装了 Podman。
- 您已在主机机器上安装了 **virt-install**。
- 您有运行 **bootc-image-builder** 工具的 root 访问权限，并在 **--privileged** 模式下运行容器，以构建镜像。

流程

1. 可选：创建一个 **config.toml** 来配置用户访问，例如：

```
[[blueprint.customizations.user]]
name = "user"
password = "pass"
key = "ssh-rsa AAA ... user@email.com"
groups = ["wheel"]
```

2. 运行 **bootc-image-builder**。另外，如果您想要使用用户访问配置，请将 **config.toml** 作为参数传递。

注意

如果您没有容器存储挂载和 **本地镜像** 选项，则您的镜像必须是公共镜像。

a. 以下是创建公共 QCOW2 镜像的示例：

```
$ sudo podman run \
    --rm \
    -it \
    --privileged \
    --pull=newer \
    --security-opt label=type:unconfined_t \
    -v ./config.toml:/config.toml \
    -v ./output:/output \
    -v /var/lib/containers/storage:/var/lib/containers/storage \
registry.redhat.io/rhel9/bootc-image-builder:latest \
    --type qcow2 \
    --config config.toml \
  quay.io/<namespace>/<image>:<tag>
```

b. 以下是创建私有 QCOW2 镜像的示例：

```
$ sudo podman run \
    --rm \
    -it \
    --privileged \
    --pull=newer \
    --security-opt label=type:unconfined_t \
    -v ./config.toml:/config.toml \
    -v ./output:/output \
    registry.redhat.io/rhel9/bootc-image-builder:latest \
    --type qcow2 \
    --config config.toml \
  quay.io/<namespace>/<image>:<tag>
```

您可以在 output 文件夹中找到 **.qcow2** 镜像。

后续步骤

- 您可以部署镜像。请参阅使用带有 QCOW2 磁盘镜像的 KVM 部署容器镜像。

- 您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

3.4. 使用 BOOTC-IMAGE-BUILDER 创建 AMI 镜像，并将其上传到 AWS

从可引导容器镜像创建一个 Amazon 机器镜像(AMI)，并使用它来启动一个 Amazon Web Service EC2 (Amazon Elastic Compute Cloud)实例。

先决条件

- 您已在主机机器上安装了 Podman。
- AWS 帐户中有一个现有的 **AWS S3** 存储桶。
- 您有运行 **bootc-image-builder** 工具的 root 访问权限，并在 **--privileged** 模式下运行容器，以构建镜像。
- 您已在帐户中配置了 **vmimport** 服务角色，来将 AMI 导入到 AWS 帐户中。

流程

1. 从可引导容器镜像创建磁盘镜像。
   - 在 Containerfile 中配置用户详情。确保为它分配了 sudo 访问权限。
   - 使用 Containerfile 配置中的用户构建一个自定义操作系统镜像。它会创建一个具有无密码 sudo 访问权限的默认用户。
2. 可选：使用 **cloud-init** 配置机器镜像。请参阅 使用 cloud-init 来注入用户和 SSH 密钥。以下是一个示例：

```
FROM registry.redhat.io/rhel9/rhel-bootc:9.4

RUN dnf -y install cloud-init && \
    ln -s ../cloud-init.target /usr/lib/systemd/system/default.target.wants && \
    rm -rf /var/{cache,log} /var/lib/{dnf,rhsm}
```

注意

您还可以使用 **cloud-init**，使用实例元数据添加用户和其他配置。

3. 构建可引导容器镜像。例如，要将镜像部署到 **x86_64** AWS 机器，请使用以下命令：

```
$ podman build -t quay.io/<namespace>/<image>:<tag> .
$ podman push quay.io/<namespace>/<image>:<tag> .
```

4. 使用 **bootc-image-builder** 工具从 bootc 容器镜像创建一个 AMI。

```
$ sudo podman run \
  --rm \
  -it \
  --privileged \
  --pull=newer \
  -v $HOME/.aws:/root/.aws:ro \
  --env AWS_PROFILE=default \
```

```
  registry.redhat.io/rhel9/bootc-image-builder:latest \
  --type ami \
  --aws-ami-name rhel-bootc-x86 \
  --aws-bucket rhel-bootc-bucket \
  --aws-region us-east-1 \
quay.io/<namespace>/<image>:<tag>
```

注意

以下标记必须一起指定。如果没有指定任何标记，AMI 被导出到 output 目录中。

- **--AWS-ami-name** - AWS 中 AMI 镜像的名称
- **--AWS-bucket** - 创建 AMI 时中间存储的目标 S3 存储桶名称
- **--AWS-region** - 用于AWS 上传的目标区域
  bootc-image-builder 工具构建一个 AMI 镜像，并使用 AWS 凭证将其上传到 AWS s3 存储桶，以便在构建后推送并注册一个 AMI 镜像。

后续步骤

- 您可以部署镜像。请参阅使用 AMI 磁盘镜像将容器镜像部署到 AWS。
- 您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

其他资源

- AWS CLI 文档

3.5. 使用 BOOTC-IMAGE-BUILDER 创建原始磁盘镜像

您可以使用 **bootc-image-builder**，将可引导容器镜像转换为带有 MBR 或 GPT 分区表的 Raw 镜像。RHEL 基础镜像不包括默认用户，因此您可以选择使用 **--config** 选项运行 **bootc-image-builder** 容器来注入一个用户配置。或者，您可以使用 **cloud-init** 配置基础镜像，以便在第一次引导时注入用户和 SSH 密钥。请参阅 使用 cloud-init 来注入用户和 SSH 密钥。

先决条件

- 您已在主机机器上安装了 Podman。
- 您有运行 **bootc-image-builder** 工具的 root 访问权限，并在 **--privileged** 模式下运行容器，以构建镜像。
- 您已在容器存储中拉取了目标容器镜像。

流程

1. 可选：创建一个 **config.toml** 来配置用户访问，例如：

```
[[blueprint.customizations.user]]
name = "user"
password = "pass"
key = "ssh-rsa AAA ... user@email.com"
groups = ["wheel"]
```

2. 运行 **bootc-image-builder**。如果要使用用户访问配置，请将 **config.toml** 作为参数传递：

```
$ sudo podman run \
  --rm \
  -it \
  --privileged \
  --pull=newer \
  --security-opt label=type:unconfined_t \
  -v /var/lib/containers/storage:/var/lib/containers/storage \
  -v ./config.toml:/config.toml \
  -v ./output:/output \
  registry.redhat.io/rhel9/bootc-image-builder:latest \
  --local \
  --type raw \
  --config config.toml \
 quay.io/<namespace>/<image>:<tag>
```

您可以在 output 文件夹中找到 **.raw** 镜像。

后续步骤

- 您可以部署镜像。请参阅使用带有 QCOW2 磁盘镜像的 KVM 部署容器镜像。
- 您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

3.6. 使用 BOOTC-IMAGE-BUILDER 创建 ISO 镜像

您可以使用 **bootc-image-builder** 创建一个 ISO，您可以从其执行一个可引导容器的离线部署。

先决条件

- 您已在主机机器上安装了 Podman。
- 您有运行 **bootc-image-builder** 工具的 root 访问权限，并在 **--privileged** 模式下运行容器，以构建镜像。

流程

- 运行 **bootc-image-builder**:

```
$ sudo podman run \
  --rm \
  -it \
  --privileged \
  --pull=newer \
  --security-opt label=type:unconfined_t \
  -v $(pwd)/config.toml:/config.toml \
  -v $(pwd)/output:/output \
  registry.redhat.io/rhel9/bootc-image-builder:latest \
  --type iso \
  --config config.toml \
 quay.io/<namespace>/<image>:<tag>
```

您可以在 output 文件夹中找到 **.iso** 镜像。

后续步骤

- 您可以对无人值守安装方法（如 USB 盘或 Install-on-boot）使用 ISO 镜像。可安装的引导 ISO 包含一个配置的 Kickstart 文件。请参阅使用 Anaconda 和 Kickstart 部署容器镜像。

警告

在带有现有操作系统或数据的机器上引导 ISO 可能具有破坏性，因为 Kickstart 被配置为自动重新格式化系统上的第一个磁盘。

- 您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

3.7. 验证和故障排除

如果您在配置 AWS 镜像需求时遇到任何问题，请参阅以下文档

- AWS IAM 帐户管理器
- 使用 AWS CLI 的高级(s3)命令。
- S3 存储桶.
- 地区和区域.
- 在 AWS 上启动自定义 RHEL 镜像。

有关用户、组、SSH 密钥和 secret 的更多详细信息，请参阅

- 在 RHEL 的镜像模式下管理用户、组、SSH 密钥和 secret

第 4 章 部署 RHEL 可引导镜像

您可以使用以下不同的机制来部署 **rhel-bootc** 容器镜像。

- Anaconda
- **bootc-image-builder**
- **bootc install**

提供以下可引导镜像类型：

- 使用 **bootc image-builder** 生成的磁盘镜像，例如：
  - QCOW2 (QEMU copy-on-write、虚拟磁盘)
  - Raw (Mac 格式)
  - AMI (Amazon Cloud)
  - ISO：使用 USB 盘或 Install-on-boot 的无人值守安装方法。

创建可部署的分层的镜像后，有几种方法可将镜像安装到主机上：

- 您可以使用以下机制，使用 RHEL 安装程序和 Kickstart 将分层的镜像安装到裸机系统：
  - 使用 USB 部署
  - PXE
- 您还可以使用 **bootc-image-builder** 将容器镜像转换为可引导镜像，并将其部署到裸机或云环境中。

安装方法仅发生一次。部署镜像后，在更新发布时，将来的任何更新都将直接从容器注册中心应用。

图 4.1. 使用基本构建安装程序 bootc 安装，或使用 Anaconda 和 Kickstart 部署容器镜像，部署可引导的容器镜像

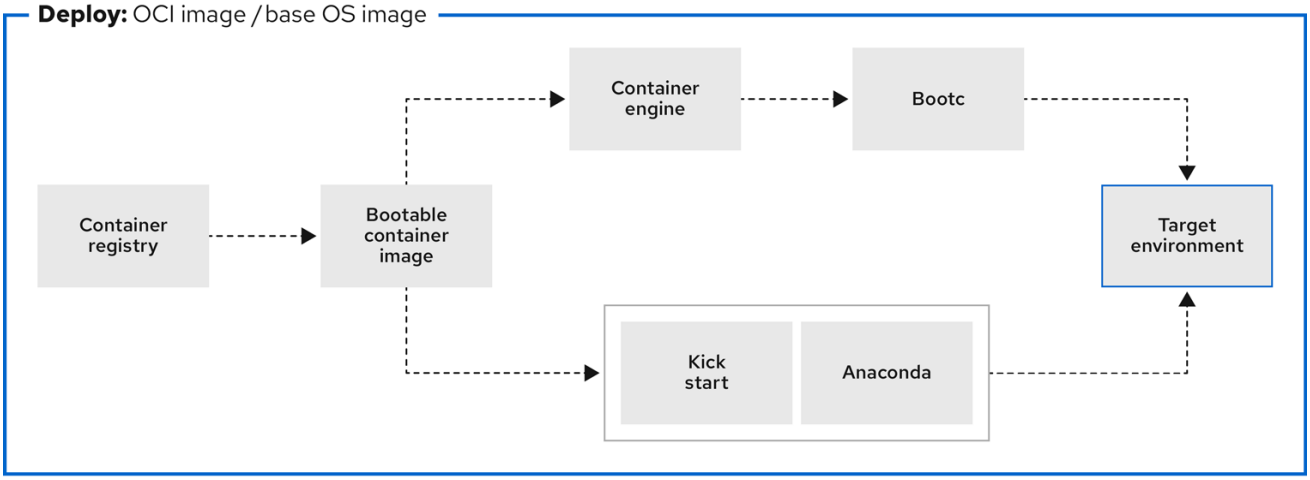
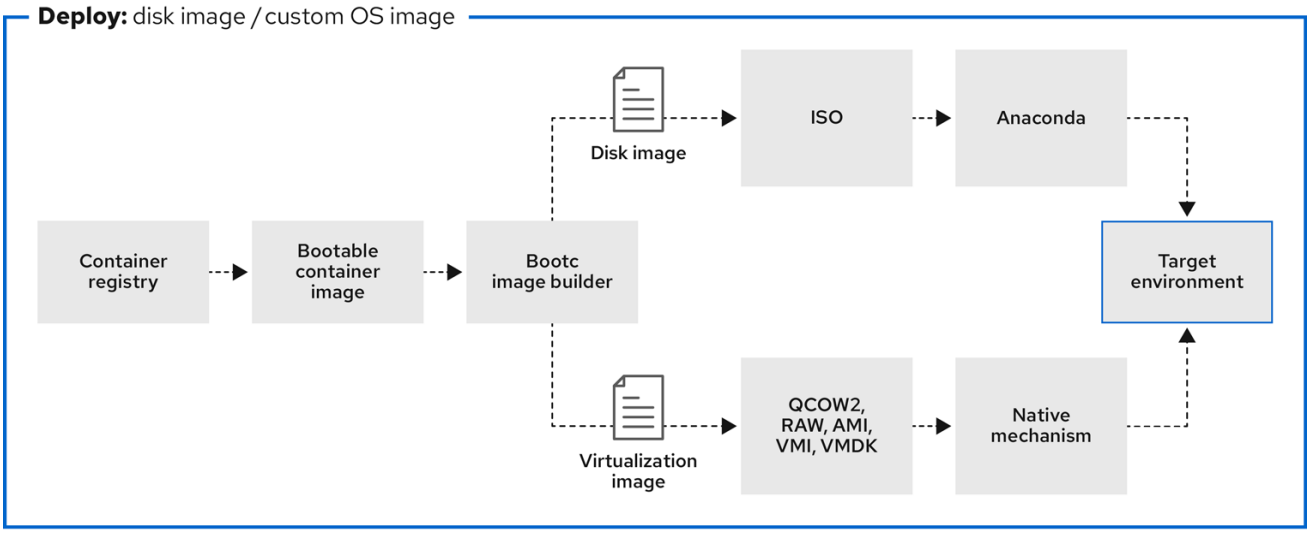


图 4.2. 使用 bootc-image-builder 从可引导容器镜像创建磁盘镜像，并在不同的环境中部署磁盘镜像，如边缘、服务器和云，使用 Anaconda、bootc-image-builder 或 bootc 安装



4.1. 使用带有 QCOW2 磁盘镜像的 KVM 部署容器镜像

使用 **bootc-image-builder** 工具从 RHEL 可引导容器镜像创建 QEMU 磁盘镜像后，您可以使用虚拟化软件引导它。

先决条件

- 您创建了一个容器镜像。请参阅使用 bootc-image-builder 创建 QCOW2 镜像 。
- 您已将容器镜像推送到一个可访问的存储库。

流程

- 使用 **libvirt** 运行您创建的容器镜像。如需了解更多详细信息，请参阅使用命令行界面创建虚拟机。
  - 以下示例使用 **libvirt** ：

```
$ sudo virt-install \
  --name bootc \
  --memory 4096 \
  --vcpus 2 \
  --disk qcow2/disk.qcow2 \
  --import \
  --os-variant rhel9-unknown
```

验证

- 连接到在其中运行容器镜像的虚拟机。如需了解更多详细信息，请参阅 连接到虚拟机。

后续步骤

- 您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

其他资源

- 配置和管理虚拟化

4.2. 使用 AMI 磁盘镜像将容器镜像部署到 AWS

在使用 **bootc-image-builder** 工具从可引导容器镜像创建一个 AMI，并将其上传到 AWS s3 存储桶后，您可以使用 AMI 磁盘镜像将容器镜像部署到 AWS。

先决条件

- 您已从可引导的容器镜像创建一个了 Amazon 机器镜像(AMI)。请参阅使用 bootc-image-builder 创建 AMI 镜像并将其上传到 AWS。
- **cloud-init** 位于您之前创建的 Containerfile 中，以便您可以为您的用例创建分层的镜像。

流程

1. 在浏览器中，访问 Service→EC2 并登录。
2. 在 AWS 控制台仪表盘菜单中，选择正确的地区。镜像必须具有 **Available** 状态，以表示它已被正确上传。
3. 在 AWS 仪表盘上，选择您的镜像并点 **Launch**。
4. 在打开的新窗口中，根据您要启动镜像所需的资源来选择一个实例类型。点击 **Review and Launch**。
5. 查看您的实例详情。如果需要进行任何更改，您可以编辑任何部分。点 **Launch**。
6. 在启动实例之前，选择一个访问它的公钥。您可以使用您已有的密钥对，也可以创建一个新的密钥对。
7. 点 **Launch Instance** 来启动您的实例。您可以检查实例的状态，它显示为 **Initializing**。实例状态变为 **Running** 后，**Connect** 按钮变为可用。
8. 点 **Connect**。此时会出现一个窗口，其中包含有关如何使用 SSH 进行连接的说明。
9. 运行以下命令来设置私钥文件的权限，以便只有您可以读取它。请参阅 连接到您的 Linux 实例。

```
$ chmod 400 <your-instance-name.pem>
```

10. 使用其公共 DNS 连接到您的实例：

```
$ ssh -i <your-instance-name.pem>ec2-user@<your-instance-IP-address>
```

注意

除非停止它，否则您的实例继续运行。

验证

启动镜像后，您可以：

- 尝试在浏览器中连接到 http://<your_instance_ip_address>。

- 检查在使用 SSH 连接到您的实例的过程中是否能够执行任何操作。

后续步骤

- 部署镜像后，您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

其他资源

- 将镜像推送到 AWS Cloud AMI
- Amazon 机器镜像(AMI)

4.3. 使用 ANACONDA 和 KICKSTART 部署容器镜像

使用 **bootc-image-builder** 将可引导容器镜像转换为 ISO 镜像后，您可以使用 Anaconda 和 Kickstart 部署 ISO 镜像来安装容器镜像。可安装的引导 ISO 已经包含配置的 **ostreecontainer** Kickstart 文件，您可用于置备自定义容器镜像。

警告

不支持使用 **rpm-ostree** 进行更改或安装内容。

先决条件

- 您已从红帽下载了用于您架构的 9.4 Boot ISO。请参阅 下载 RH 引导镜像。

流程

1. 创建一个 **ostreecontainer** Kickstart 文件。例如：

```
# Basic setup
text
network --bootproto=dhcp --device=link --activate
# Basic partitioning
clearpart --all --initlabel --disklabel=gpt
reqpart --add-boot
part / --grow --fstype xfs

# Reference the container image to install - The kickstart
# has no %packages section. A container image is being installed.
ostreecontainer --url registry.redhat.io/rhel9/bootc-image-builder:latest

firewall --disabled
services --enabled=sshd

# Only inject a SSH key for root
rootpw --iscrypted locked
sshkey --username root "<your key here>"
reboot
```

–

2. 使用 9.4 Boot ISO 安装介质引导系统。

   a. 将具有以下内容的 Kickstart 文件附加到内核参数中：

   ```
   inst.ks=http://<path_to_your_kickstart>
   ```

3. 按 **CTRL+X** 引导系统。

后续步骤

- 部署容器镜像后，您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

其他资源

- ostreecontainer 文档
- 当使用本地 rpm-ostree 修改解决方案时 bootc 升级会失败

4.4. 部署一个自定义 ISO 容器镜像

使用 **bootc-image-builder** 将可引导容器镜像转换为 ISO 镜像。这会创建一个与可下载的 RHEL ISO 类似的系统，除非您的容器镜像内容被嵌入到 ISO 磁盘镜像中。您不需要在安装过程中访问网络。然后，您将通过 **bootc-image-builder** 创建的 ISO 磁盘镜像安装到裸机系统。

先决条件

- 您已创建了一个自定义容器镜像。

流程

1. 使用 **bootc-image-builder** 创建一个自定义安装程序 ISO 磁盘镜像。请参阅使用 bootc-image-builder 创建 ISO 镜像。
2. 将 ISO 磁盘镜像复制到 USB 闪存中。
3. 使用 USB 盘中的内容在断开连接的环境中执行裸机安装。

后续步骤

- 部署容器镜像后，您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

4.5. 通过 PXE 引导部署一个 ISO 可引导容器

您可以使用网络安装，通过 PXE 引导部署 RHEL ISO 镜像，以运行 ISO 可引导容器镜像。

先决条件

- 您已从红帽下载了用于您架构的 9.4 Boot ISO。请参阅 下载 RH 引导镜像。
- 您已为 PXE 引导配置了服务器。选择以下选项之一：

  - 对于 HTTP 客户端，请参阅为 HTTP 和 PXE 引导配置 DHCPv4 服务器 。
  - 对于基于 UEFI 的客户端，请参阅为基于 UEFI 的客户端配置 TFTP 服务器。
  - 对于基于 BIOS 的客户端，请参阅为基于 BIOS 的客户端配置 TFTP 服务器。
- 您有一个客户端，也称为您要安装 ISO 镜像的系统。

流程

1. 将 RHEL 安装 ISO 镜像导出到 HTTP 服务器中。PXE 引导服务器现在可以提供 PXE 客户端。
2. 引导客户端并开始安装。
3. 当提示指定引导源时选择 PXE 引导。如果没有显示引导选项，请按键盘上的 Enter 键，或等待引导窗口打开。
4. 在 Red Hat Enterprise Linux 引导窗口中，选择您想要的引导选项，然后按 Enter 键。
5. 启动网络安装。

后续步骤

- 您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

其他资源

- 准备使用 PXE 从网络安装
- 使用 PXE 从网络引导安装

4.6. 使用 BOOTC-IMAGE-BUILDER 构建、配置和启动磁盘镜像

您可以使用 Containerfile 将配置注入到自定义镜像中。

流程

1. 创建一个磁盘镜像。以下示例演示了如何将用户添加到磁盘镜像中。

```
[[blueprint.customizations.user]]
name = "user"
password = "pass"
key = "ssh-rsa AAA ... user@email.com"
groups = ["wheel"]
```

   - **Name** - 用户名。必需
   - **password** - 未加密的密码。不是必需的
   - **Key** - SSH 公钥内容。不是必需的
   - **groups** - 要将用户添加到其中的组的数组。不是必需的
2. 运行 **bootc-image-builder** ，并传递以下参数：

```
$ sudo podman run \
    --rm \
    -it \
    --privileged \
    --pull=newer \
    --security-opt label=type:unconfined_t \
    -v $(pwd)/config.toml:/config.toml \
    -v $(pwd)/output:/output \
    registry.redhat.io/rhel9/bootc-image-builder:latest \
    --type qcow2 \
    --config config.toml \
    quay.io/<namespace>/<image>:<tag>
```

3. 启动虚拟机，例如使用 **virt-install** ：

```
$ sudo virt-install \
  --name bootc \
  --memory 4096 \
  --vcpus 2 \
  --disk qcow2/disk.qcow2 \
  --import \
  --os-variant rhel9
```

验证

- 使用 SSH 访问系统：

```
# ssh -i /<path_to_private_ssh-key> <user1>@<ip-address>
```

后续步骤

- 部署容器镜像后，您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

4.7. 使用 BOOTC 部署一个容器镜像

有了 **bootc** ，您就有了一个容器，它是真相的来源。它包含一个基本的构建安装程序，它作为 **bootc install to-disk** 或 **bootc install to-filesystem** 提供。通过使用 **bootc** 安装方法，您不需要执行任何其他步骤来部署容器镜像，因为容器镜像包含一个基本的安装程序。

使用 RHEL 的镜像模式，您可以安装未配置的镜像，例如，没有默认密码或 SSH 密钥的镜像。

使用 RHEL ISO 镜像对设备执行裸机安装。

先决条件

- 您已从红帽下载了用于您架构的 9.4 Boot ISO。请参阅 下载 RH 引导镜像。
- 您已创建了一个配置文件。

流程

- 将配置注入正在运行的 ISO 镜像，例如：

```
$ podman run --rm --privileged --pid=host -v /var/lib/containers:/var/lib/containers --
security-opt label=type:unconfined_t <image> bootc install to-disk <path-to-disk>
```

后续步骤

- 部署容器镜像后，您可以对镜像进行更新，并将更改推送到注册中心。请参阅管理 RHEL 可引导镜像。

4.8. 使用 TO-FILESYSTEM 的高级安装

bootc install 包含两个子命令：**bootc install to-disk** 和 **bootc install to-filesystem**。

- **bootc-install-to-filesystem** 执行到目标文件系统的安装。
- **bootc install to-disk** 子命令包含一组固执己见的较低级别的工具，您可以单独调用它们。命令由以下工具组成：
  - **mkfs.$fs /dev/disk**
  - **mount /dev/disk /mnt**
  - **bootc install to-filesystem --karg=root=UUID=*<uuid of /mnt>* --imgref $self /mnt**

4.8.1. 使用 bootc install to-existing-root

bootc install to-existing-root 是 **install to-filesystem** 的一种变体。您可以使用它来将现有系统转换为目标容器镜像。

警告

这个转换删除了 **/boot** 和 **/boot/efi** 分区，并可删除现有 Linux 安装。转换过程重复使用文件系统，即使已保留了用户数据，系统也不再以软件包模式引导。

先决条件

- 您必须具有 root 权限才能完成这个流程。
- 您必须匹配主机环境和目标容器版本，例如，如果您的主机是 RHEL 9 主机，则您必须有一个 RHEL 9 容器。使用 **btrfs** 在 Fedora 主机上安装 RHEL 容器，因为 RHEL 内核不支持该文件系统。

流程

- 运行以下命令将现有系统转换为目标容器镜像。使用 **-v /:/target** 选项传递目标 **rootfs**。

```
# podman run --rm --privileged -v /dev:/dev -v /var/lib/containers:/var/lib/containers -v
/:/target \
        --pid=host --security-opt label=type:unconfined_t \
        <image> \
        bootc install to-existing-root
```

-

这个命令删除 **/boot** 中的数据，但现有操作系统中的所有其他内容不会被自动删除。这很有用，因为新镜像可以自动从以前的主机系统导入数据。因此，在 **/sysroot** 中后续重新启动后，容器镜像、数据库、用户主目录数据、**/etc** 中的配置文件都可用。

您还可以通过添加 **--root-ssh-authorized-keys /target/root/.ssh/authorized_keys** 来使用 **--root-ssh-authorized-keys** 标志继承 root 用户 SSH 密钥。例如：

```
# podman run --rm --privileged -v /dev:/dev -v /var/lib/containers:/var/lib/containers -v
/:/target \
        --pid=host --security-opt label=type:unconfined_t \
        <image> \
        bootc install to-existing-root --root-ssh-authorized-keys
/target/root/.ssh/authorized_keys
```

第 5 章 管理 RHEL 可引导镜像

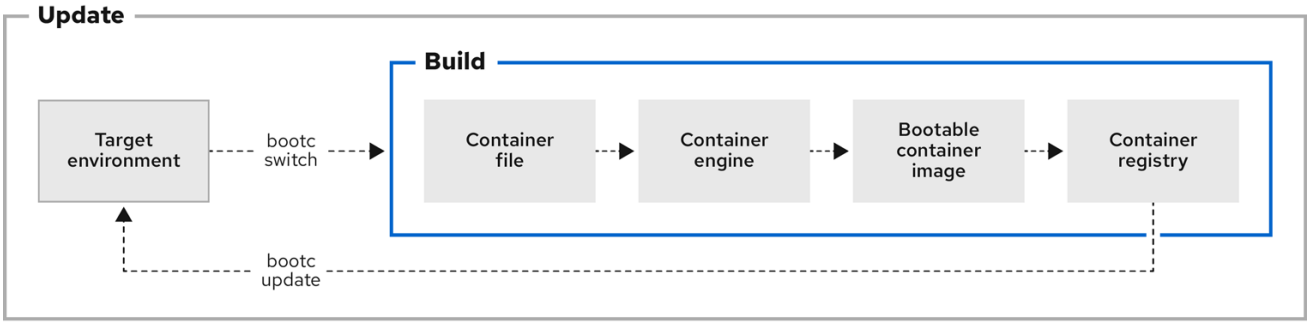
安装和部署 RHEL 可引导镜像后，您可以对容器镜像执行管理操作，如更改或更新系统。系统支持部署后带回滚的就地事务更新。

这种管理（也称为第 2 天管理基线）包括从容器注册中心以事务方式获取新操作系统更新，并将系统引导到这些更新中，同时在出现故障时支持手动或自动回滚。

您还可以依赖自动更新，其默认是开启的。**systemd 服务单元** 和 **systemd 定时器单元** 文件检查容器 registry 是否有更新，并将其应用到系统。您可以触发具有不同事件的更新过程，如更新应用程序。有自动化工具监视这些更新，然后触发 CI/CD 管道。需要重新启动，因为更新是事务性的。对于需要更复杂的或调度的推出部署的环境，您必须禁用自动更新，并使用 **bootc** 实用程序更新您的操作系统。

如需了解更多详细信息，请参阅 第 2 天操作支持。

图 5.1. 手动更新安装的操作系统，根据需要更改容器镜像引用或回滚更改



5.1. 切换容器镜像引用

您可以使用 **bootc switch** 命令更改用于升级的容器镜像引用。例如，您可以从 stage 切换到 production 标签。**bootc switch** 命令执行与 **bootc upgrade** 命令同样的操作，并额外更改容器镜像引用。

要手动切换现有的 **基于 ostree 的** 容器镜像引用，请使用 **bootc switch** 命令。

警告

不支持使用 **rpm-ostree** 进行更改或安装内容。

先决条件

- 使用 **bootc** 引导的系统。

流程

- 运行以下命令：

```
$ bootc switch [--apply] quay.io/<namespace>/<image>:<tag>
```

另外，当您要自动执行操作时，您可以使用 **--apply** 选项，如系统更改后的重启。

注意

bootc switch 命令的作用与 **bootc 升级** 相同。唯一的区别是容器镜像引用已更改。这允许在 **/etc** 和 **/var** 中保留现有状态，例如主机 SSH 密钥和主目录。

其他资源

- **bootc-switch** 手册页

5.2. 在 BOOTC 镜像 INITRAMFS 中添加模块

rhel9/rhel-bootc 镜像使用 **dracut** 基础架构在镜像构建期间构建初始 RAM 磁盘(**initrd**)。**initrd** 构建并包含在容器内的 **/usr/lib/modules/$kver/initramfs.img** 位置。

您可以使用置入配置文件来覆盖 dracut 配置，并将其放在 **/usr/lib/ dracut /dracut.conf.d/ *<50-custom-added-modules.conf>*** 中，因此使用您要添加的模块重新创建 **initrd**。

先决条件

- 使用 bootc 引导的系统。

流程

- 重新创建 **initrd** 作为容器构建的一部分：

```
FROM <baseimage>
COPY <50-custom-added-modules>.conf /usr/lib/dracut/dracut.conf.d
RUN set -x; kver=$(cd /usr/lib/modules && echo *); dracut -vf
/usr/lib/modules/$kver/initramfs.img $kver
```

注意

默认情况下，命令会尝试拉取正在运行的内核版本，这会导致错误。明确传递给 **目标的** kernel 版本以避免出错。

5.3. 修改和重新生成 INITRD

默认容器镜像在 **/usr/lib/modules/$kver/initramfs.img** 中包含预生成的初始 RAM 磁盘(initrd)。要重新生成 **initrd**，例如要添加 dracut 模块，请按照以下步骤执行：

流程

1. 编写 drop-in 配置文件。例如：

```
dracutmodules = "module"
```

2. 将置入配置文件放在 **dracut** 通常使用的位置： **/usr**。例如：

```
/usr/lib/dracut/dracut.conf.d/50-custom-added-modules.conf
```

3. 作为容器构建的一部分，重新生成 **initrd**。您必须将内核版本明确传递给 **dracut**，因为它会尝试拉取运行的内核版本，这会导致错误。以下是一个示例：

```
FROM <baseimage>
COPY 50-custom-added-modules.conf /usr/lib/dracut/dracut.conf.d
RUN set -x; kver=$(cd /usr/lib/modules && echo *); dracut -vf
/usr/lib/modules/$kver/initramfs.img $kver
```

5.4. 从安装的操作系统执行手动更新

安装 RHEL 的镜像模式是一次性任务。您可以通过将更改推送到容器注册中心来执行任何其他管理任务，如更改或更新系统。

当使用 RHEL 的镜像模式时，您可以选择为您的系统执行手动更新。如果您有执行更新的自动化方法，如使用 Ansible，手动更新也很有用。由于默认启用了自动更新，要执行手动更新，您必须关闭自动更新。您可以选择以下选项之一来完成此操作：

- 运行 **bootc upgrade** 命令
- 修改 **systemd** 计时器文件

5.5. 关闭自动更新

要执行手动更新，您必须关闭自动更新。您可以选择以下流程中的选项之一来实现此目的。

流程

- 禁用容器构建的计时器。
  - 通过运行 **bootc upgrade** 命令：

```
$ systemctl mask bootc-fetch-apply-updates.timer
```

  - 通过修改 **systemd** 计时器文件。使用 **systemd** "drop-ins" 来覆盖计时器。在以下示例中，更新计划为每周一次。
    1. 使用以下内容创建一个 **updates.conf** 文件：

```
[Timer]
# Clear previous timers
OnBootSec= OnBootSec=1w OnUnitInactiveSec=1w
```

    2. 将容器添加到您创建的文件中：

```
$ mkdir -p /usr/lib/systemd/system/bootc-fetch-apply-updates.timer.d
$ cp updates.conf /usr/lib/systemd/system/bootc-fetch-apply-updates.timer.d
```

5.6. 手动更新安装的操作系统

要手动从注册中心获取更新，并将系统引导至新更新，请使用 **bootc upgrade**。该命令将事务性就地更新从已安装的操作系统获取到容器镜像注册中心。命令查询注册中心，并为下一次启动将更新的容器镜像排队。它暂存对基础镜像的更改，而不是默认更改正在运行的系统。

流程

- 运行以下命令:

```
$ bootc upgrade [--apply]
```

apply 参数是可选的，当要自动执行操作时，您可以使用它，比如系统更改后的重启。

注意

bootc upgrade 和 **bootc update** 命令是别名。

其他资源

- **bootc-upgrade** 手册页

5.7. 从更新的操作系统执行回滚

您可以使用 **bootc rollback** 命令回滚以前的引导条目来还原更改。此命令通过将 **rollback** 下的部署排队等待下一个启动，来更改引导装载程序条目顺序。然后，当前部署变为回滚。所有已暂存的更改，如未应用的排队的升级，都被丢弃。

在回滚完成后，系统会重启，更新计时器在 1 到 3 小时内运行，后者自动更新并将您的系统重启到您刚刚从中回滚的镜像。

警告

如果您执行回滚，系统将再次更新，除非您关闭自动更新。请参阅 关闭自动更新。

先决条件

- 您已对系统执行了更新。

流程

- 运行以下命令:

```
$ bootc rollback [-h|--help] [-V|--version]
```

注意

bootc rollback 命令的作用与 **bootc 升级** 相同。唯一的区别是被跟踪的容器镜像。这允许在 **/etc** 和 **/var** 中保留现有状态，例如主机 SSH 密钥和主目录。

验证

- 使用 **systemd journal** 来检查检测的回滚调用的日志消息。

```
$ journalctl -b
```

-

您可以看到类似如下的日志：

```
MESSAGE_ID=26f3b1eb24464d12aa5e7b544a6b5468
```

其他资源

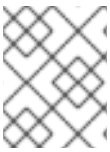
- **bootc-rollback** 手册页

5.8. 向系统组部署更新

您可以通过修改 Containerfile 来更改操作系统的配置。然后，您可以构建容器镜像并将其推送到注册中心。当您下次引导操作系统时，更新将被应用。

您还可以使用 **bootc switch** 命令更改容器镜像源。容器注册中心是数据源。请参阅 切换容器镜像参考。

通常，在向系统组部署更新时，您可以使用中央管理服务来提供要安装在连接到中央服务的每个系统上的客户端。通常，管理服务要求客户端执行一次性注册。以下是如何将更新部署到系统组的一个示例。如果需要，您可以修改它来创建一个持久的 **systemd** 服务。



注意

为了清晰起见，示例中的 Containerfile 没有优化。例如，避免在镜像中创建多个层的一个好的优化是通过调用 RUN 一次。

您可以将客户端安装到 RHEL 镜像的镜像模式中，并在启动时运行它来注册系统。

先决条件

- management-client 使用 **cron** 作业或单独的 **systemd** 服务处理未来的到服务器的连接。

流程

- 创建一个具有以下特征的管理服务：它决定何时升级系统。
  1. 如果已包含在基础镜像中，请禁用 **bootc-fetch-apply-updates.timer**。
  2. 使用 **dnf** 或适用于您客户端的一些其他方法安装客户端。
  3. 将管理服务的凭据注入到镜像中。

5.9. 检查清单健康状况

健康检查是第 2 天的操作之一。您可以手动检查容器镜像的系统健康状况和容器中运行的事件。

您可以通过在命令行上创建容器来设置健康检查。您可以使用 **podman inspect** 或 **podman ps** 命令显示容器的健康检查状态。

您可以使用 **podman events** 命令监控和输出 Podman 中发生的事件。如果适用，每个事件都包括时间戳、类型、状态、名称，以及镜像。

有关健康检查和事件的更多信息，请参阅监控容器 一章。

5.10. 自动化和 GITOPS

您可以使用 CI/CD 管道自动化构建过程，以便更新过程可被事件触发，如更新一个应用程序。您可以使用自动化工具来跟踪这些更新并触发 CI/CD 管道。管道通过使用事务后台操作系统更新来保持系统最新。

第 6 章 在 RHEL 的镜像模式中管理文件系统

目前，RHEL 的镜像模式使用 OSTree 作为后端，并默认为存储启用 **composefs**。**/opt** 和 **/usr/local** 路径是纯文本目录，而不是指向 **/var** 的符号链接。这可让您轻松在派生容器镜像中安装写入 **/opt** 中的第三方内容，例如：

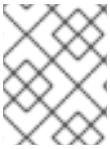
6.1. 使用 /SYSROOT 的物理和虚拟 ROOT

当系统被完全引导时，它与 **chroot** 类似，即操作系统更改当前运行的进程及其子项的明显根目录。物理主机根文件系统挂载到 **/sysroot**。**chroot** 文件系统称为部署 root。

其余的文件系统路径是部署 root 的一部分，用作系统引导的最终目标。系统使用 **ostree=kernel** 参数来查找部署根。

/usr

这个文件系统会将所有操作系统内容保存在 **/usr** 中，目录（如 **/bin** ）作为 **/usr/bin** 的符号链接。



注意

启用 **composefs** 的 **/usr** 与 / 不同。这两个目录都是同一不可变镜像的一部分，因此您不需要使用 bootc 系统执行完整的 **UsrMove**。

/usr/local

基础镜像被配置为使用 **/usr/local** 作为默认目录。

/etc

/etc 目录默认包含 mutable 持久状态，但它支持启用 **etc.transient** 配置选项。当目录处于可变持久性状态时，它会在升级过程中执行三向合并：

- 使用新默认 **/etc** 作为基础
- 将当前和之前的 **/etc** 之间的 diff 应用到新的 **/etc** 目录
- 保留与 **/etc** 中相同部署的默认 **/usr/etc** 不同的本地修改文件。

ostree-finalize-staged.service 在创建新引导装载程序条目前在关闭过程中执行这些任务。

这是因为 Linux 系统的许多组件在 **/etc** 目录中提供默认配置文件。即使默认软件包没有提供，默认情况下，软件只检查 **/etc** 中的配置文件。非基于 bootc 镜像的、没有不同版本的 **/etc** 更新系统只在安装过程中填充，且不会在安装后随时更改。这会导致初始镜像版本影响 **/etc** 系统状态，并可能导致问题来应用更改，例如：**/etc/sudoers.conf**，需要外部干预。有关文件配置的详情，请参阅 构建和测试 RHEL 可引导容器镜像。

/var

默认情况下，**/var** 目录中的内容是持久的。您还可以使 **/var** 或 子目录挂载点持久，无论是网络还是 **tmpfs**。

只有一个 **/var** 目录。如果它不是不同的分区，则 **/var** 目录物理是绑定到 **/ostree/deploy/$stateroot/var**，并在可用的引导装载程序条目部署间共享。

默认情况下，**/var** 中的内容充当卷，即在初始安装过程中复制容器镜像中的内容，并在之后不会更新。

/var 和 **/etc** 目录不同。您可以将 **/etc** 用于相对较小的配置文件，预期的配置文件通常绑定到 **/usr** 中的操作系统二进制文件。**/var** 目录有任意的大型数据，如系统日志、数据库和默认，如果操作系统状态被回滚，则不会回滚。

例如，进行 **dnf downgrade postgresql** 等更新不会影响 **/var/lib/postgres** 中的物理数据库。同样，进行 **bootc 更新** 或 **bootc 回滚** 不会影响这个应用程序数据。

使用 **/var** 也会在应用新操作系统更新之前完全正常工作，即下载并就绪更新，但仅在重启时生效。同样适用于 Docker 卷，因为它将应用程序代码与其数据分离。

如果您希望应用程序具有预先创建的目录结构，例如 **/var/lib/postgresql**，则可以使用这个情况。为此，请使用 **systemd tmpfiles.d**。您还可以以单位使用 **StateDirectory= <*directory&* gt;**。

其他目录

不支持在容器镜像中的 **/run**、**/proc** 或其他 API Filesystems 中提供内容。除此之外，其他顶级目录（如 **/usr** 和 **/opt** ）与容器镜像相关联。

/opt

使用 **composefs** 的 **bootc** 时，**/opt** 目录是只读的，以及其他顶级目录，如 **/usr**。

当软件需要写入 **/opt/exampleapp** 中的其自己的目录时，常见的模式是使用要重定向到的符号链接，例如 **/var** 用于日志文件等操作：

```
RUN rmdir /opt/exampleapp/logs && ln -sr /var/log/exampleapp /opt/exampleapp/logs
```

另外，您可以将 systemd 单元配置为启动服务来动态进行这些挂载。例如：

```
BindPaths=/var/log/exampleapp:/opt/exampleapp/logs
```

启用临时 root

要默认启用完全临时可写 **rootfs**，请在 **prepare-root.conf** 中设置以下选项：

```
[root]
transient = true
```

这可让软件临时写入 **/opt**，对于必须保留的内容，到 **/var** 的符号链接。

6.2. 版本选择和引导

RHEL 的镜像模式默认使用 GRUB，但 **s390x** 架构除外。系统中当前可用的 RHEL 镜像模式的每个版本都有一个菜单条目。

菜单条目引用由 Linux 内核、**initramfs** 和链接到 OSTree 提交的哈希组成的 OSTree 部署，您可以使用 **ostree=kernel** 参数传递。

在启动过程中，OSTree 读取内核参数，以确定将哪些部署用作根文件系统。每次更新或更改系统（如软件包安装、添加内核参数）会创建一个新的部署。

这会启用在更新导致问题时回滚到以前的部署。

第 7 章 附录：在 RHEL 的镜像模式下管理用户、组、SSH 密钥和 SECRET

了解更多有关 RHEL 的镜像模式下用户、组、SSH 密钥和 secret 管理的信息。

7.1. 用户和组配置

RHEL 的镜像模式是通用操作系统更新和配置机制。您不能使用它来配置用户或组。唯一的例外是 **bootc install** 命令有 **--root-ssh-authorized-keys** 选项。

通用基础镜像的用户和组配置

通常，分发基础镜像没有任何配置。因为安全风险，不要在通用镜像中使用公开可用的私钥加密密码和 SSH 密钥。

通过 systemd 凭证注入 SSH 密钥

您可以使用 **systemd** 在某些环境中注入 root 密码或 SSH **authorized_keys** 文件。例如，使用系统管理 BIOS (SMBIOS)注入 SSH 密钥系统固件。您可以在本地虚拟化环境中配置这个，如 **qemu**。

使用 cloud-init注入用户和 SSH 密钥

许多基础架构即服务(IaaS)和虚拟化系统使用通常由软件（如 **cloud-init** 或 **ignition** ）处理的元数据服务器。请参阅 AWS 实例元数据。您使用的基础镜像可能包括 **cloud-init** 或 Ignition，或者您可以将其安装在您自己的派生镜像中。在此模型中，SSH 配置在可引导镜像之外管理。

使用容器或单元自定义逻辑添加用户和凭证

cloud-init 等系统没有特权。您可以以您要启动容器镜像的方式注入您要管理凭证的任何逻辑，例如使用 **systemd** 单元。要管理凭据，您可以使用自定义网络托管的源，如 FreeIPA。

在容器构建中静态添加用户和凭证

在面向软件包的系统中，您可以使用以下命令，使用派生的构建来注入用户和凭证：

```
RUN useradd someuser
```

您可以在 **useradd** 的默认 **shadow-utils** 实现中发现问题：用户和组 ID 是动态分配的，这可能导致偏移。

用户和组主目录，以及 /var 目录

对于使用持久性 **/home** → **/var/home** 配置的系统，在初始安装后，对容器镜像的 **/var** 所做的任何更改都不会应用到后续更新中。
例如，如果您将 **/var/home/someuser/.ssh/authorized_keys** 注入到一个容器构建中，则现有的系统不会获得更新的 **authorized_keys** 文件。

对 systemd 单元使用 DynamicUser=yes

对于系统用户，在可能的时候使用 **systemd DynamicUser=yes** 选项。
这比在软件包安装时分配用户或组的模式要好得多，因为它避免了潜在的 UID 或 GID 偏移。

使用 systemd-sysusers

例如，在派生的构建中使用 **systemd**-sysusers。如需更多信息，请参阅 **systemd** -sysusers 文档。

```
COPY mycustom-user.conf /usr/lib/sysusers.d
```

sysusers 工具在引导过程中根据需要更改传统的 **/etc/passwd** 文件。如果 **/etc** 是持久的，则这可以避免 **UID** 或 **GID** 偏移。这意味着 **UID** 或 **GID** 分配取决于特定机器随时间升级的方式。

使用 systemd JSON 用户记录

请参阅 JSON 用户记录 **systemd** 文档。与 **sysusers** 不同，这些用户的规范状态在 **/usr** 中。如果后续镜像删除了用户记录，则它也会从系统中消失。

使用 nss-altfiles

使用 **nss-altfiles**，您可以删除 **systemd** JSON 用户记录。它将系统用户分成 **/usr/lib/passwd** 和 **/usr/lib/group**，与 OSTree 项目处理合并 **/etc** 的方式保持一致，因为它与 **/etc/passwd** 相关。目前，如果 **/etc/passwd** 文件在本地系统上以任何方式被修改，则不会在容器镜像中应用对 **/etc/passwd** 的后续更改。
rpm-ostree 构建的基础镜像默认启用了 **nns-altfiles**。

另外，基础镜像有一个由 NSS 文件管理和预分配的系统用户，以避免 UID 或 GID 偏移。

在派生的容器构建中，您还可以将用户附加到 **/usr/lib/passwd** 中，例如：使用 **sysusers.d** 或 **DynamicUser=yes**。

用户的机器本地状态

文件系统布局取决于基础镜像。
默认情况下，用户数据存储在 **/etc**、**/etc/passwd**、**/etc/shadow** 、**groups** 以及 **/home**中，具体取决于基础镜像。但是，通用基础镜像必须是机器本地持久状态。在此模型中，**/home** 是到 **/var/home/*user*** 的符号链接。

在系统置备时注入用户和 SSH 密钥

对于 **/etc** 和 **/var** 被默认配置为持久的基础镜像，您可以使用 Anaconda 或 Kickstart 等安装程序注入用户。
通常，通用安装程序是为一次性 bootstrap 而设计的。然后，配置成为一个可变的机器本地状态，您可以使用一些其他机制在第 2 天操作中进行更改。

您可以使用 Anaconda 安装程序设置初始密码。但是，更改此初始密码需要不同的系统内工具，如 **passwd**。

这些流在 **bootc 兼容** 系统中等效地工作，以支持用户直接安装通用基础镜像，而无需更改为不同的系统内工具。

临时主目录

许多操作系统部署最小化持久、可变和可执行状态。这可能会损坏用户主目录。
/home 目录可以被设置为 **tmpfs**，以确保用户数据在重启后被清除。当与临时 **/etc** 目录结合时，此方法工作得非常好。

要将用户的主目录设置为，例如，注入 SSH **authorized_keys** 或其他文件，请使用 **systemd tmpfiles.d** 片断：

```
f~ /home/user/.ssh/authorized_keys 600 user user - <base64 encoded data>
```

SSH 嵌入在镜像中，如：**/usr/lib/tmpfiles.d/<username-keys.conf**。另一个示例是嵌入在镜像中的服务，其可以从网络获取密钥并编写它们。这是 **cloud-init** 使用的模式。

UID 和 GID 偏移

/etc/passwd 和类似的文件是名称和数字标识符之间的映射。当映射是动态的，并与"无状态"容器镜像构建混合时，可能它可能会导致问题。每个容器镜像构建可能会因 RPM 安装顺序或其他原因而导致 UID 改变。如果该用户保持持久状态，则这可能是一个问题。要处理这种情况，请将其转换为使用 **sysusers.d** 或使用 **DynamicUser=yes**。

7.2. 在 RHEL 的镜像模式下注入 SECRET

RHEL 的镜像模式没有用于 secret 的固执己见的机制。在某些情况下，您可以在系统中注入容器 pull secret，例如：

- 要使 **bootc** 从需要身份验证的注册中心获取更新，您必须在文件中包含一个 pull secret。在以下示例中，**creds** secret 包含注册中心 pull secret。

```
FROM registry.redhat.io/rhel9/bootc-image-builder:latest
COPY containers-auth.conf /usr/lib/tmpfiles.d/link-podman-credentials.conf
RUN --mount=type=secret,id=creds,required=true cp /run/secrets/creds /usr/lib/container-auth.json && \
    chmod 0600 /usr/lib/container-auth.json && \
    ln -sr /usr/lib/container-auth.json /etc/ostree/auth.json
```

  要构建它，请运行 **podman build --secret id=creds,src=$HOME/.docker/config.json**。使单个 pull secret 用于 **bootc** 和 Podman，方法是使用到容器镜像中嵌入的通用持久文件的两个位置的符号链接，如 **/usr/lib/container-auth.json**。

- 要让 Podman 获取容器镜像，请在 **/etc/containers/auth.json** 种包含一个 pull secret。使用这个配置，两个堆栈共享 **/usr/lib/container-auth.json** 文件。

  通过将它们嵌入到容器构建中来注入 secret

  如果注册中心服务器被适当保护，则您可以在容器镜像中包含 secret。在某些情况下，仅将 bootstrap secret 嵌入到容器镜像中是一种可行的模式，特别是在与让机器认证到集群的机制一起使用时。在此模式中，作为主机系统的一部分或容器镜像运行的置备工具使用 bootstrap secret 注入或更新其他 secret，如 SSH 密钥、证书等。

  使用云元数据注入 secret

  大多数生产基础设施即服务(IaaS)系统支持元数据服务器或等同的服务器，其可以安全地保护主机 secret，特别是 bootstrap secret。容器镜像可以包含 **cloud-init** 或 **ignition** 等工具来获取这些 secret。

  通过在磁盘镜像中嵌入 secret 来注入它们

  您只能在磁盘镜像中嵌入 **bootstrap secret**。例如，当您从一个输入容器镜像（如 AMI 或 OpenStack）生成一个云磁盘镜像时，磁盘镜像可以包含是有效的机器本地状态的 secret。轮转它们需要额外的管理工具或刷新磁盘镜像。

  使用裸机安装程序注入 secret

  安装程序工具通常支持通过 secret 注入配置。

  通过 systemd 凭证注入 secret

  systemd 项目有一个凭证概念，来确保获取凭证数据，并将其传递给系统和服务，这适用于某些部署方法。详情请查看 systemd 凭证 文档。

其他资源

- 请参阅 可引导容器的示例。