



# Red Hat Enterprise Linux 9

## 准备、安装和管理 RHEL for Edge 镜像

使用 Red Hat Enterprise Linux 9 创建、部署和管理 Edge 系统



# Red Hat Enterprise Linux 9 准备、安装和管理 RHEL for Edge 镜像

---

使用 Red Hat Enterprise Linux 9 创建、部署和管理 Edge 系统

## 法律通告

Copyright © 2024 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js<sup>®</sup> 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<sup>®</sup> 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

使用镜像构建器工具制作为 Edge 优化的自定义 RHEL (rpm-ostree) 镜像。然后，您可以在 Edge 服务器上远程安装并安全地管理和扩展镜像部署。

# 目录

对红帽文档提供反馈 .....	5
<b>第 1 章 RHEL FOR EDGE 镜像 .....</b>	<b>6</b>
1.1. RHEL FOR EDGE 支持的构架	6
1.2. 如何编写和部署 RHEL FOR EDGE 镜像	7
1.3. 非基于网络的部署	8
1.4. 基于网络的部署	9
1.5. RHEL RPM 镜像和 RHEL FOR EDGE 镜像之间的区别	11
<b>第 2 章 设置 RHEL 镜像构建器 .....</b>	<b>13</b>
2.1. 镜像构建器系统要求	13
2.2. 安装 RHEL 镜像构建器	13
<b>第 3 章 配置 RHEL 镜像构建器存储库 .....</b>	<b>15</b>
3.1. 在 RHEL 镜像构建器中添加自定义第三方存储库	15
3.2. 将带有特定发行版的第三方存储库添加到 RHEL 镜像构建器中	15
3.3. 使用 GPG 检查存储库元数据	16
3.4. RHEL 镜像构建器官方存储库覆盖	18
3.5. 覆盖系统存储库	18
3.6. 覆盖需要订阅的系统存储库	19
3.7. 配置和使用 SATELLITE CV 作为内容源	20
3.8. 使用 SATELLITE CV 作为软件仓库在 RHEL 镜像构建器中构建镜像	22
<b>第 4 章 在 RHEL WEB 控制台中使用镜像构建器制作 RHEL FOR EDGE 镜像 .....</b>	<b>23</b>
4.1. 在 RHEL WEB 控制台中访问 RHEL 镜像构建器	23
4.2. 在 WEB 控制台中使用镜像构建器为 RHEL FOR EDGE 镜像创建蓝图	24
4.3. 在 WEB 控制台中使用镜像构建器创建 RHEL FOR EDGE COMMIT 镜像	25
4.4. 在 RHEL WEB 控制台中使用 RHEL 镜像构建器创建 RHEL FOR EDGE 容器镜像	26
4.5. 在 RHEL WEB 控制台中使用镜像构建器创建 RHEL FOR EDGE INSTALLER 镜像	28
4.6. 下载 RHEL FOR EDGE 镜像	29
4.7. 其他资源	29
<b>第 5 章 使用镜像构建器命令行制作 RHEL FOR EDGE 镜像 .....</b>	<b>30</b>
5.1. 基于网络的部署 workflow	30
5.2. 非基于网络的部署 workflow	35
5.3. 支持的镜像自定义	40
5.4. RHEL 镜像构建器安装的软件包	50
<b>第 6 章 构建简化的安装程序镜像以置备 RHEL FOR EDGE 镜像 .....</b>	<b>54</b>
6.1. 简化的安装程序镜像构建和部署	54
6.2. 使用 RHEL 镜像构建器 CLI 为简化的镜像创建蓝图	55
6.3. 使用镜像构建器 CLI 创建 RHEL FOR EDGE SIMPLIFIED INSTALLER 镜像	56
6.4. 使用镜像构建器命令行界面下载简化的 RHEL FOR EDGE 镜像	57
6.5. 使用镜像构建器 GUI 为简化的镜像 RHEL 创建蓝图	58
6.6. 使用镜像构建器 GUI 创建 RHEL FOR EDGE SIMPLIFIED INSTALLER 镜像	61
6.7. 使用镜像构建器 GUI 下载简化的 RHEL FOR EDGE 镜像	61
6.8. 设置 UEFI HTTP 引导服务器	62
6.9. 在虚拟机中部署简化的 ISO 镜像	63
6.10. 从 USB 闪存驱动器部署简化的 ISO 镜像	64
6.11. 在 FIPS 模式下创建并引导 RHEL FOR EDGE 镜像	64
<b>第 7 章 构建并提供一个最小原始镜像 .....</b>	<b>67</b>
7.1. 最小原始镜像构建和部署	67

7.2. 使用 RHEL 镜像构建器 CLI 为 MINIMAL RAW 镜像创建蓝图	67
7.3. 使用 RHEL 镜像构建器 CLI 创建一个 MINIMAL RAW 镜像	68
7.4. 下载并解压缩 MINIMAL RAW 镜像	69
7.5. 从 USB 闪存驱动器部署 MINIMAL RAW 镜像	69
<b>第 8 章 使用 RHEL FOR EDGE SIMPLIFIED INSTALLER 镜像的 IGNITION 工具</b>	<b>71</b>
8.1. 创建一个 IGNITION 配置文件	71
8.2. 在 GUI 中创建支持 IGNITION 的蓝图	73
8.3. 使用 CLI 创建支持 IGNITION 的蓝图	73
<b>第 9 章 为 RHEL FOR EDGE 创建 VMDK 镜像</b>	<b>76</b>
9.1. 使用 IGNITION 配置创建蓝图	76
9.2. 为 RHEL FOR EDGE 创建一个 VMDK 镜像	77
9.3. 上传 VMDK 镜像并在 VSPHERE 中创建 RHEL 虚拟机	78
<b>第 10 章 创建 RHEL FOR EDGE AMI 镜像</b>	<b>80</b>
10.1. 为 EDGE AMI 镜像创建蓝图	80
10.2. 创建一个 RHEL FOR EDGE AMI 镜像	81
10.3. 将 RHEL FOR EDGE AMI 镜像上传到 AWS	82
<b>第 11 章 自动提供和加入带有 FIDO 的 RHEL FOR EDGE 设备</b>	<b>85</b>
11.1. FIDO 设备加入(FIDO)过程	85
11.2. 自动置备和注册 RHEL FOR EDGE 设备	88
11.3. 生成密钥和证书	89
11.4. 安装并运行制造服务器	90
11.5. 安装、配置和运行 RENDEZVOUS 服务器	92
11.6. 安装、配置和运行所有者服务器	93
11.7. 使用 FIDO 身份验证自动加入 RHEL FOR EDGE 设备	95
<b>第 12 章 使用 FIDO 加入一个带有数据库后端的 RHEL FOR EDGE 设备</b>	<b>97</b>
12.1. 加入带有 FIDO 数据库的设备	97
<b>第 13 章 在基于网络的环境中部署 RHEL FOR EDGE 镜像</b>	<b>100</b>
13.1. 提取 RHEL FOR EDGE 镜像提交	100
13.2. 设置 WEB 服务器以安装 RHEL FOR EDGE 镜像	101
13.3. 使用 KICKSTART 对边缘设备执行有人值守的安装	103
13.4. 使用 KICKSTART 对边缘设备执行无人值守安装	105
<b>第 14 章 在非基于网络的环境中部署 RHEL FOR EDGE 镜像</b>	<b>108</b>
14.1. 为非网络部署创建 RHEL FOR EDGE 容器镜像	108
14.2. 为非网络部署创建 RHEL FOR EDGE 安装程序镜像	109
14.3. 为非网络部署安装 RHEL FOR EDGE 镜像	110
<b>第 15 章 管理 RHEL FOR EDGE 镜像</b>	<b>112</b>
15.1. 使用镜像构建器编辑 RHEL FOR EDGE 镜像蓝图	112
15.2. 更新 RHEL FOR EDGE 镜像	114
15.3. 升级 RHEL FOR EDGE 系统	123
15.4. 部署 RHEL FOR EDGE 自动镜像更新	126
15.5. RHEL FOR EDGE 镜像的回滚	128
<b>第 16 章 创建并管理 OSTREE 镜像更新</b>	<b>134</b>
16.1. OSTREE 的基本概念	134
16.2. 创建 OSTREE 存储库	135
16.3. 管理集中式 OSTREE 镜像	135
<b>附录 A. 术语和命令</b>	<b>139</b>

---

A.1. OSTREE 和 RPM-OSTREE 术语	139
A.2. OSTREE 命令	139
A.3. RPM-OSTREE 命令	140
A.4. FDO 自动加入术语	141
A.5. FDO 自动加入技术	142





---

## 对红帽文档提供反馈

我们感谢您对我们文档的反馈。帮助我们如何进行改进。

### 通过 Jira 提交反馈（需要帐户）

1. 登录到 [Jira](#) 网站。
2. 点顶部导航栏中的 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您的建议以改进。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

## 第 1 章 RHEL FOR EDGE 镜像

RHEL for Edge 镜像是一个 **rpm-ostree** 镜像，其中包含在 Edge 服务器中远程安装 RHEL 的系统软件包。

系统软件包包括：

- **Base OS** 软件包
- podman 作为容器引擎
- 其他 RPM 软件包管理器(RPM)内容

与 RHEL 镜像不同，RHEL for Edge 是一个不可变的操作系统，即它包含一个具有以下特征的只读根目录：

- 软件包与根目录隔离
- 软件包安装创建层，使其易于回滚到以前的版本
- 对断开连接环境的有效更新
- 支持多个操作系统分支和存储库
- 有一个混合 **rpm-ostree** 软件包系统

您可以在裸机、设备和 Edge 服务器上部署 RHEL for Edge 镜像。

您可以使用 RHEL 镜像构建器工具制作自定义的 RHEL for Edge 镜像。您还可以通过访问 Red Hat Hybrid Cloud Console 平台中的 [边缘管理](#) 应用程序，并配置自动化管理来创建 RHEL for Edge 镜像。

边缘管理应用程序简化了您可以提供和注册镜像的方法。要了解更多信息有关边缘管理的信息，请参阅 [创建 RHEL for Edge 镜像及配置自动化管理](#) 文档。



### 警告

[边缘管理](#) 应用程序不支持使用 **RHEL 镜像构建器** 内部版本工件创建的 RHEL for Edge 自定义镜像。请参阅 [边缘管理可支持性](#)。

在 RHEL for Edge 镜像中，您可以实现：

#### Atomic upgrades

State of each update is known / no changes are seen until reboot

#### Custom health checks and intelligent rollbacks

Resiliency in case of failed upgrades

#### Container-focused workflow

Separate core OS updates

#### Optimized OTA payloads

Transfer only delta updates

125\_RHEL\_1020

### 1.1. RHEL FOR EDGE 支持的构架

目前，您可以在 AMD 和 Intel 64 位系统中部署 RHEL for Edge 镜像。



### 注意

RHEL for Edge 在某些设备上支持 ARM 系统。有关支持的设备的更多信息，请参阅 [红帽认证的硬件](#)。

## 1.2. 如何编写和部署 RHEL FOR EDGE 镜像

制作和部署 RHEL for Edge 镜像分为两个阶段：

1. 使用 RHEL 镜像构建器工具制作 RHEL **rpm-ostree** 镜像。您可以通过 **composer-cli** 工具中的命令行界面访问 RHEL 镜像构建器，或者在 RHEL web 控制台使用图形用户界面。
2. 使用 RHEL 安装程序部署镜像。

在制作 RHEL for Edge 镜像时，您可以选择以下任何一种镜像类型。编写不同的 RHEL for Edge 镜像可能需要或可能不需要网络访问。请参阅表：

表 1.1. RHEL for Edge 镜像类型

镜像类型	描述	适用于基于网络的部署	适用于基于非网络的部署	使用方法
RHEL for Edge Commit (.tar)	<b>edge-commit</b> 镜像无法直接启动，即使它包含完整的操作系统。要引导 <b>edge-commit</b> 镜像类型，您必须部署它。	是	否	向系统交付原子和安全更新。
RHEL for Edge Container (.tar)	<b>edge-container</b> 创建一个 <b>OSTree</b> 提交，并将其嵌入到带有 web 服务器的 OCI 容器中。当 <b>edge-commit</b> 镜像启动时，Web 服务器会将提交充当 <b>OSTree</b> 存储库。	否	是	提供要从安装程序镜像获取的 <b>OSTree</b> 提交。
RHEL for Edge Installer (.iso)	<b>edge-installer</b> 镜像类型从正在运行的容器拉取提交，并使用 Kickstart 文件创建一个可安装的引导 ISO，其被配置为使用嵌入的 <b>OSTree</b> 提交。	否	是	将镜像写入到一个闪存介质中，并在需要 ISO 镜像的断开连接的环境中使用。

镜像类型	描述	适用于基于网络的部署	适用于基于非网络的部署	使用方法
RHEL for Edge Raw Image(.raw.xz)	<b>edge-raw-image</b> 压缩的原始镜像由一个文件组成，该文件中包含一个现有的部署了 OSTree 提交的分區布局。	是	是	通过在硬盘上闪存 RHEL Raw 镜像或在虚拟机上引导，来用于裸机平台。
RHEL for Edge Simplified Installer (.iso)	<b>edge-simplified-installer</b> 镜像使用 Ignition 工具在引导过程的早期阶段将用户配置注入到镜像中。	是	是	用于无人值守安装到设备。引导后，它将 RHEL for Edge 镜像提供给设备。
RHEL for Edge AMI (.ami)	<b>edge-ami</b> 镜像使用 Ignition 工具在引导过程的早期阶段将用户配置注入到镜像中。您可以将 .ami 镜像上传到 AWS，并在 AWS 中引导 EC2 实例。	是	是	在 AWS 云中启动 EC2 实例。
RHEL for Edge VMDK (.vmdk)	<b>edge-vsphere</b> 镜像使用 Ignition 工具在引导过程的早期阶段，将用户配置注入到镜像中。您可以在 vSphere 上载入镜像，并在 VM vSphere 中引导镜像	是	是	在 vSphere 虚拟机中引导 .vmdk 镜像

镜像类型在内容上有所不同，因此适合不同类型的部署环境。

## 其他资源

- [执行标准 RHEL 9 安装](#)。

## 1.3. 非基于网络的部署

使用 RHEL 镜像构建器创建灵活的 RHEL **rpm-ostree** 镜像以满足您的要求，然后使用 Anaconda 在您的环境中部署它们。

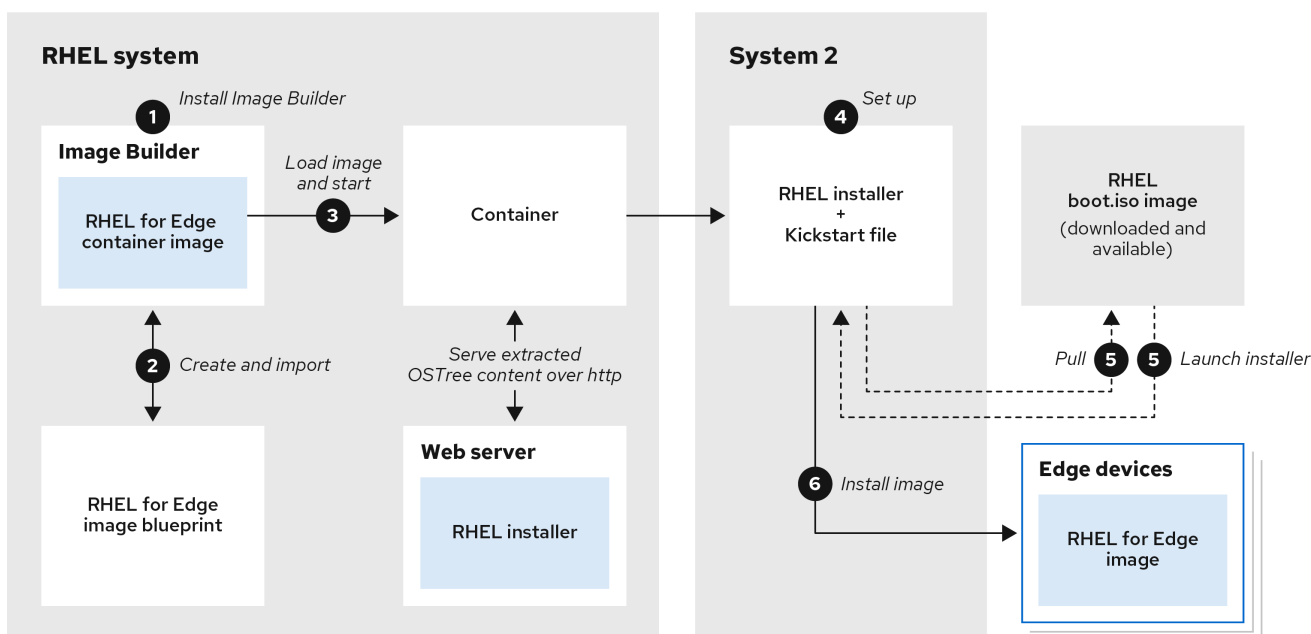
您可以通过 **composer-cli** 工具中的命令行界面访问 RHEL 镜像构建器，或者在 RHEL web 控制台使用图形用户界面。

在非网络部署中制作和部署 RHEL for Edge 镜像涉及以下高级别的步骤：

1. 安装并注册 RHEL 系统
2. 安装 RHEL 镜像构建器
3. 使用 RHEL 镜像构建器，为 RHEL for Edge 容器镜像创建带有自定义的蓝图
4. 在 RHEL 镜像构建器中导入 RHEL for Edge 蓝图
5. 创建嵌入在 OCI 容器中的 RHEL for Edge 镜像，其中包含 webserver 可将提交部署为 OSTree 存储库
6. 下载 RHEL for Edge 容器镜像文件
7. 使用 RHEL for Edge 容器提交部署容器提供存储库
8. 使用 RHEL 镜像构建器，为 RHEL for Edge Installer 镜像创建另一个蓝图
9. 创建一个 RHEL for Edge 安装程序镜像，以便从嵌入的、使用 RHEL for Edge 容器镜像的正在运行的容器中拉取提交
10. 下载 RHEL for Edge 安装程序镜像
11. 运行安装

下图显示了 RHEL for Edge 镜像非网络部署工作流：

图 1.1. 在非网络环境中部署 RHEL for Edge



164\_RHEL\_0621

## 1.4. 基于网络的部署

使用 RHEL 镜像构建器创建灵活的 RHEL **rpm-ostree** 镜像以满足您的要求，然后使用 Anaconda 在您的环境中部署它们。RHEL 镜像构建器自动识别部署设置的详情，并将镜像输出生成为一个 **.tar** 文件的 **edge-commit**。

您可以通过 **composer-cli** 工具中的命令行界面访问 RHEL 镜像构建器，或者在 RHEL web 控制台使用图形用户界面。

您可以通过执行以下高级别步骤编写和部署 RHEL for Edge 镜像：

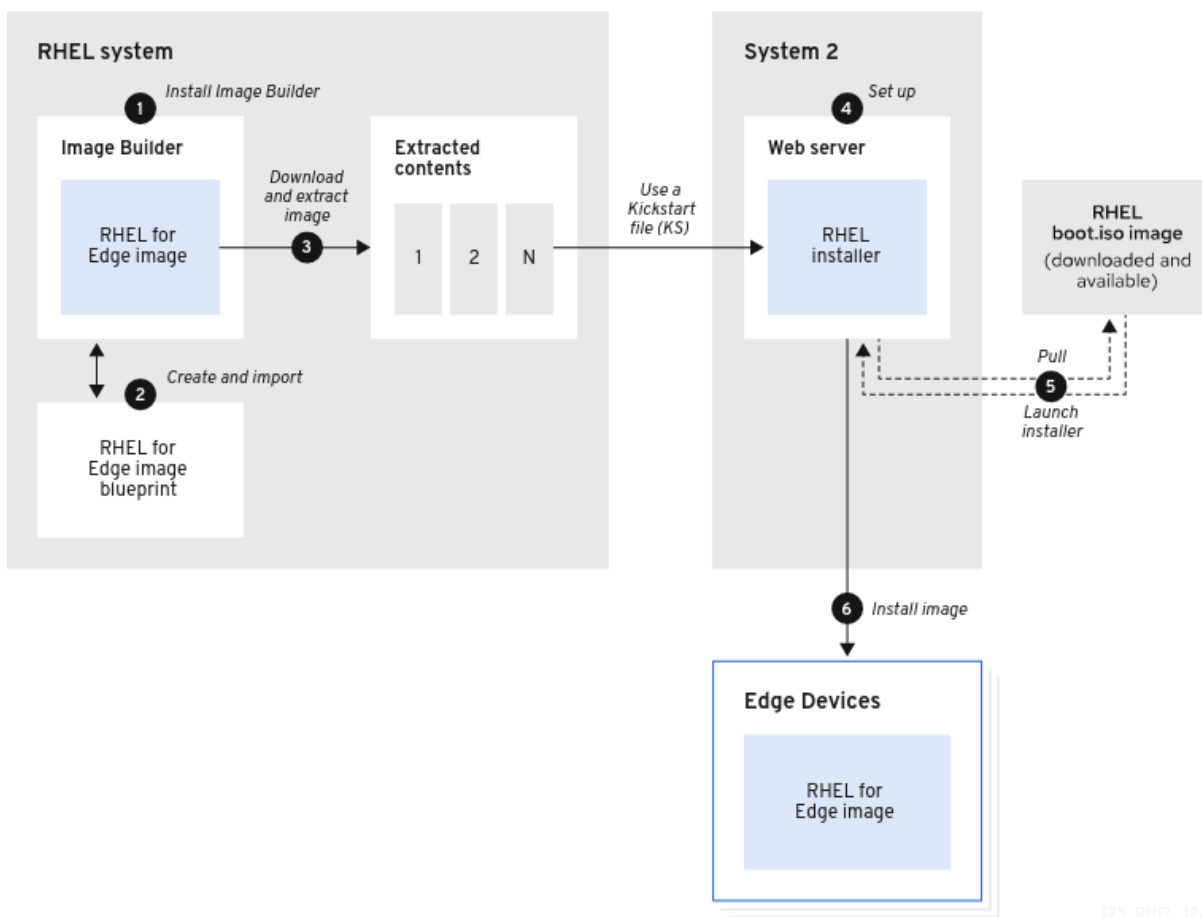
#### 对于有人值守的安装

1. 安装并注册 RHEL 系统
2. 安装 RHEL 镜像构建器
3. 使用 RHEL 镜像构建器，为 RHEL for Edge 镜像创建蓝图
4. 在 RHEL 镜像构建器中导入 RHEL for Edge 蓝图
5. 创建 RHEL for Edge Commit (**.tar**) 镜像
6. 下载 RHEL for Edge 镜像文件
7. 在安装了 RHEL 镜像构建器的同一系统上，安装您要为 RHEL for Edge Commit 内容提供服务的 web 服务器。具体说明请参阅 [设置和配置 NGINX](#)
8. 将 RHEL for Edge Commit (**.tar**) 内容提取到正在运行的 web 服务器
9. 创建一个从正在运行的 web 服务器拉取 OSTree 内容的 Kickstart 文件。有关如何修改 Kickstart 来拉取 OSTree 内容的详情，请参阅 [提取 RHEL for Edge 镜像提交](#)
10. 在边缘设备上引导 RHEL 安装程序 ISO，并为它提供 Kickstart。

对于无人值守安装，您可以自定义 RHEL 安装 ISO，并将 Kickstart 文件嵌入到其中。

下图显示了 RHEL for Edge 网络镜像部署工作流：

图 1.2. 在网络基础环境中部署 RHEL for Edge



## 1.5. RHEL RPM 镜像和 RHEL FOR EDGE 镜像之间的区别

您可以以基于传统软件包的 RPM 格式创建 RHEL 系统镜像，并作为 RHEL for Edge (**rpm-ostree**) 镜像。

您可以使用基于软件包的传统 RPM 在传统数据中心上部署 RHEL。但是，您可以使用 RHEL for Edge 镜像在传统数据中心以外的服务器上部署 RHEL。这些服务器包括处理大量数据的系统，它们与生成数据的源（边缘服务器）最接近。

RHEL for Edge (**rpm-ostree**) 镜像不是软件包管理器。它们只支持完整的可引导文件系统树，而不是单个文件。这些镜像没有有关单个文件的信息，如这些文件是如何生成，或与其原始文件相关的任何东西。

**rpm-ostree** 镜像需要单独的机制（软件包管理器）来在 `/var` 目录中安装其他应用程序。因此，**rpm-ostree** 镜像会使操作系统保持不变，同时维护 `/var` 和 `/etc` 目录的状态。原子更新启用回滚和更新的后台暂存。

请参阅下表以了解 RHEL for Edge 镜像与基于软件包的 RHEL RPM 镜像有何不同。

表 1.2. RHEL RPM 镜像和 RHEL for Edge 镜像之间的区别

主要属性	RHEL RPM 镜像	RHEL for Edge 镜像
<b>OS assemble</b>	您可以在本地编译软件包以组成镜像。	软件包被组合在一个 OSTree 中，您可以在系统上安装它。

<b>OS 更新</b>	您可以使用 <b>dnf update</b> 来应用已启用的存储库中的可用更新。	如果在 OSTree 远程的 <b>/etc/ostree/remotes.d/</b> 中提供了任何新提交，您可以使用 <b>rpm-ostree upgrade</b> 来暂存更新。该更新会在系统重启时生效。
<b>软件仓库</b>	软件包包含 DNF 存储库	软件包包含 OSTree 远程存储库
<b>用户访问权限</b>	读取写入	只读( <b>usr</b> )
<b>数据持久性</b>	您可以将镜像挂载到任何非 <b>tmpfs</b> 挂载点上	<b>/etc</b> 和 <b>/var</b> 已启用读/写，并包含持久性数据。



## 第 2 章 设置 RHEL 镜像构建器

使用 RHEL 镜像构建器创建自定义的 RHEL for Edge 镜像。在 RHEL 系统上安装 RHEL 镜像构建器后，RHEL 镜像构建器在 RHEL web 控制台中作为应用程序提供。您还可以通过 **composer-cli** 工具中的命令行界面访问 RHEL 镜像构建器。



### 注意

建议您在虚拟机上安装 RHEL 镜像构建器。

### 2.1. 镜像构建器系统要求

RHEL 镜像构建器运行的环境（如虚拟机）必须满足下表中列出的要求。



### 注意

不支持在容器内运行 RHEL 镜像构建器。

表 2.1. 镜像构建器系统要求

参数	最低要求值
系统类型	专用虚拟机
处理器	2 个内核
内存	4 GiB
磁盘空间	20 GiB
访问权限	管理员级别(root)
网络	连接至互联网



### 注意

20 GiB 磁盘空间要求足以在主机上安装和运行 RHEL 镜像构建器。要构建和部署镜像构建，您必须分配额外的专用磁盘空间。

### 2.2. 安装 RHEL 镜像构建器

要在专用虚拟机上安装 RHEL 镜像构建器，请按照以下步骤执行：

#### 先决条件

- 虚拟机已创建并处于开机状态。
- 已安装 RHEL，并且已订阅了 RHSM 或红帽卫星。
- 您已启用了 **BaseOS** 和 **AppStream** 存储库，以便能安装 RHEL 镜像构建器软件包。

## 步骤

1. 在虚拟机上安装以下软件包：

- osbuild-composer
- composer-cli
- cockpit-composer
- bash-completion
- firewalld

```
# dnf install osbuild-composer composer-cli cockpit-composer bash-completion firewalld
```

RHEL 镜像构建器安装为 RHEL web 控制台中的一个应用程序。

2. 重启虚拟机
3. 将系统防火墙配置为允许访问 Web 控制台：

```
# firewall-cmd --add-service=cockpit && firewall-cmd --add-service=cockpit --permanent
```

4. 启用 RHEL 镜像构建器。

```
# systemctl enable osbuild-composer.socket cockpit.socket --now
```

osbuild-composer 和 cockpit 服务在第一次访问时自动启动。

5. 载入 shell 配置脚本，以便在不重启的情况下立即启动 **composer-cli** 命令的自动完成功能：

```
$ source /etc/bash_completion.d/composer-cli
```

## 其他资源

- [管理存储库](#)

## 第 3 章 配置 RHEL 镜像构建器存储库

要使用 RHEL 镜像构建器，您必须确保配置了存储库。您可以在 RHEL 镜像构建器中使用以下类型的存储库：

### 官方存储库覆盖

如果您要从 Red Hat Content Delivery Network (CDN) 官方存储库（如网络中的自定义镜像）以外的其他位置下载基础系统 RPM。使用官方存储库覆盖会禁用默认存储库，您的自定义镜像必须包含所有必需的软件包。

### 自定义第三方存储库

使用这些存储库，以包括官方 RHEL 存储库中没有的软件包。

## 3.1. 在 RHEL 镜像构建器中添加自定义第三方存储库

您可以将自定义的第三方源添加到存储库中，并使用 **composer-cli** 管理这些存储库。

### 先决条件

- 您有自定义的第三方存储库的 URL。

### 步骤

1. 创建一个存储库源文件，如 **/root/repo.toml**。例如：

```
id = "k8s"
name = "Kubernetes"
type = "yum-baseurl"
url = "https://server.example.com/repos/company_internal_packages/"
check_gpg = false
check_ssl = false
system = false
```

**type** 字段接受以下有效值 **yum-baseurl**、**yum-mirrorlist** 和 **yum-metalink**。

2. 以 TOML 格式保存文件。
3. 将新的第三方源添加到 RHEL 镜像构建器中：

```
$ composer-cli sources add <file-name>.toml
```

### 验证

1. 检查新源是否已成功添加：

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

2. 检查新源内容：

```
$ composer-cli sources info <source_id>
```

## 3.2. 将带有特定发行版的第三方存储库添加到 RHEL 镜像构建器中

您可以使用可选字段 **distro** 在自定义第三方源文件中指定发行版列表。在镜像构建期间解析依赖项时，存储库文件会使用分发字符串列表。

指定 **rhel-9** 的任何请求都使用此源。例如，如果您列出软件包并指定 **rhel-9**，它将包含此源。但是，列出主机发行版的软件包不包括此源。

### 先决条件

- 您有自定义的第三方存储库的 URL。
- 您有要指定的发行版的列表。

### 步骤

1. 创建一个存储库源文件，如 `/root/repo.toml`。例如，要指定发行版：

```
check_gpg = true
check_ssl = true
distros = ["rhel-9"]
id = "rh9-local"
name = "packages for RHEL"
system = false
type = "yum-baseurl"
url = "https://local/repos/rhel9/projectrepo/"
```

2. 以 TOML 格式保存文件。
3. 将新的第三方源添加到 RHEL 镜像构建器中：

```
$ composer-cli sources add <file-name>.toml
```

### 验证

1. 检查新源是否已成功添加：

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

2. 检查新源内容：

```
$ composer-cli sources info <source_id>
```

## 3.3. 使用 GPG 检查存储库元数据

要检测和避免损坏的软件包，您可以使用 DNF 软件包管理器检查 RPM 软件包上的 GNU Privacy Guard (GPG) 签名，以及检查存储库元数据是否已使用 GPG 密钥进行了签名。

您可以使用密钥 URL 设置 **gpgkeys** 字段，输入您要通过 **https** 进行检查的 **gpgkey**。或者，为了提高安全性，您还可以将整个密钥嵌入到 **gpgkeys** 字段中，来直接导入它，而不是直接从 URL 获取密钥。

### 先决条件

- 您要用作存储库的目录存在，且包含软件包。

## 步骤

1. 访问您要创建存储库的文件夹：

```
$ cd repo/
```

2. 运行 **createrepo\_c**，来从 RPM 软件包创建存储库：

```
$ createrepo_c .
```

3. 访问 repodata 的目录：

```
$ cd repodata/
```

4. 为 **repomd.xml** 文件签名：

```
$ gpg -u <_gpg-key-email_> --yes --detach-sign --armor /srv/repo/example/repomd.xml
```

5. 要在存储库中启用 GPG 签名检查：

- a. 在存储库源中设置 **check\_repogpg = true**。
- b. 输入您要进行检查的 **gpgkey**。如果您的密钥可以通过 **https** 使用，请使用密钥的密钥 URL 设置 **gpgkeys** 字段。您可以根据需要添加任意数量的 URL 密钥。  
以下是一个示例：

```
check_gpg = true
check_ssl = true
id = "signed local packages"
name = "repository_name"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
check_repogpg = true
gpgkeys=["https://local/keys/repokey.pub"]
```

作为替代方案，直接在 **gpgkeys** 字段中添加 GPG 密钥，例如：

```
check_gpg = true
check_ssl = true
check_repogpg
id = "custom-local"
name = "signed local packages"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
gpgkeys=["https://remote/keys/other-repokey.pub",
"-----BEGIN PGP PUBLIC KEY BLOCK-----
...
-----END PGP PUBLIC KEY BLOCK-----"]
```

- 如果测试没有找到签名，GPG 工具会显示一个类似如下的错误：

```
$ GPG verification is enabled, but GPG signature is not available.
This may be an error or the repository does not support GPG verification:
Status code: 404 for http://repo-server/rhel/repodata/repomd.xml.asc (IP:
```

```
192.168.1.3)
```

- 如果签名无效，GPG 工具会显示一个类似如下的错误：

```
repomd.xml GPG signature verification error: Bad GPG signature
```

## 验证

- 手动测试存储库的签名：

```
$ gpg --verify /srv/repo/example/repomd.xml.asc
```

## 3.4. RHEL 镜像构建器官方存储库覆盖

RHEL 镜像构建器 **osbuild-composer** 后端不继承位于 `/etc/yum.repos.d/` 目录中的系统存储库。相反，它拥有自己的一组在 `/usr/share/osbuild-composer/repositories` 目录中定义的官方存储库。这包括红帽官方存储库，其中包含基础系统 RPM，用于向新版本安装附加软件或更新已安装的程序。如果要覆盖官方存储库，您必须在 `/etc/osbuild-composer/repositories/` 中定义覆盖。此目录用于用户定义的覆盖，这里的文件优先于 `/usr/share/osbuild-composer/repositories/` 目录中的文件。

配置文件不是 `/etc/yum.repos.d/` 中常见的 DNF 存储库格式。相反，它们是 JSON 文件。

## 3.5. 覆盖系统存储库

您可以在 `/etc/osbuild-composer/repositories` 目录中为 RHEL 镜像构建器配置自己的存储库覆盖。

### 先决条件

- 您有一个可从主机系统访问的自定义存储库。

### 步骤

1. 创建 `/etc/osbuild-composer/repositories/` 目录来存储存储库覆盖：

```
$ sudo mkdir -p /etc/osbuild-composer/repositories
```

2. 使用与 RHEL 版本相对应的名称，创建一个 JSON 文件。或者，您还可以从 `/usr/share/osbuild-composer/` 中复制用于分发的文件，并修改其内容。

对于 RHEL 9.3，请使用 `/etc/osbuild-composer/repositories/rhel-93.json`。

3. 将以下结构添加到 JSON 文件中。仅以字符串格式指定以下属性之一：

- **baseurl** - 存储库的基本 URL。
- **metalink** - 包含有效镜像存储库列表的 metalink 文件的 URL。
- **mirrorlist** - 包含有效镜像存储库列表的 mirrorlist 文件的 URL。其余的字段（如 **gpgkey**）和 **metadata\_expire** 是可选的。

例如：

```
{
  "x86_64": [
    {
```

```

        "name": "baseos",
        "baseurl": "http://mirror.example.com/composes/released/RHEL-
9/9.0/BaseOS/x86_64/os/",
        "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (...)",
        "check_gpg": true
    }
]
}

```

或者，您可以通过将 **rhel-version.json** 替换为 RHEL 版本（例如：rhel-9.json）来复制您发行版的 JSON 文件。

```
$ cp /usr/share/osbuild-composer/repositories/rhel-version.json /etc/osbuild-
composer/repositories/
```

4. 可选：验证 JSON 文件：

```
$ json_verify /etc/osbuild-composer/repositories/<file>.json
```

5. 在 **rhel-9.json** 文件中编辑 **baseurl** 路径，并保存。例如：

```
$ /etc/osbuild-composer/repositories/rhel-version.json
```

6. 重启 **osbuild-composer.service**：

```
$ sudo systemctl restart osbuild-composer.service
```

## 验证

- 检查存储库是否指向正确的 URL：

```
$ cat /etc/yum.repos.d/redhat.repo
```

您可以看到仓库指向从 **/etc/yum.repos.d/redhat.repo** 文件中复制的正确 URL。

## 其他资源

- [存储库中提供的最新 RPM 版本对 \*\*osbuild-composer\*\* 不可见。](#)

## 3.6. 覆盖需要订阅的系统存储库

您可以设置 **osbuild-composer** 服务，以使用 **/etc/yum.repos.d/redhat.repo** 文件中定义的系统订阅。要使用 **osbuild-composer** 中的系统订阅，请定义一个具有以下详情的存储库覆盖：

- 与 **/etc/yum.repos.d/redhat.repo** 中定义的存储库相同的 **baseurl**。
- JSON 对象中定义的 **"rhsm": true** 值。



## 注意

**osbuild-composer** 不自动使用 `/etc/yum.repos.d/` 中定义的存储库。您需要手动将它们指定为系统存储库覆盖，或使用 **composer-cli** 将其指定为附加源。  
"BaseOS"和"AppStream"存储库通常使用系统存储库覆盖，而所有其他存储库则使用 **composer-cli** 源。

## 先决条件

- 您的系统有一个在 `/etc/yum.repos.d/redhat.repo` 中定义的订阅
- 您已创建了一个存储库覆盖。请参阅 [覆盖系统存储库](#)。

## 步骤

1. 从 `/etc/yum.repos.d/redhat.repo` 文件中获取 **baseurl** :

```
# cat /etc/yum.repos.d/redhat.repo
[AppStream]
name = AppStream mirror example
baseurl = https://mirror.example.com/RHEL-9/9.0/AppStream/x86_64/os/
enabled = 1
gpgcheck = 0
sslverify = 1
sslcacert = /etc/pki/ca1/ca.crt
sslclientkey = /etc/pki/ca1/client.key
sslclientcert = /etc/pki/ca1/client.crt
metadata_expire = 86400
enabled_metadata = 0
```

2. 配置存储库覆盖以使用相同的 **baseurl**，并将 **rhsm** 设为 `true` :

```
{
  "x86_64": [
    {
      "name": "AppStream mirror example",
      "baseurl": "https://mirror.example.com/RHEL-9/9.0/AppStream/x86_64/os/",
      "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (...)",
      "check_gpg": true,
      "rhsm": true
    }
  ]
}
```

3. 重启 **osbuild-composer.service** :

```
$ sudo systemctl restart osbuild-composer.service
```

## 其他资源

- [当主机注册到 Satellite 6 时，RHEL 镜像构建器使用 CDN 存储库](#)

## 3.7. 配置和使用 SATELLITE CV 作为内容源



您可以使用 Satellite 的内容视图(CV)作为存储库来使用 RHEL 镜像构建器构建镜像。为此，请在注册到 Satellite 的主机上手动配置存储库引用，以便您可以从 Satellite 存储库检索，而不是 Red Hat Content Delivery Network (CDN)官方存储库。

## 先决条件

- 您在注册到 Satellite 6 的主机上使用 RHEL 镜像构建程序。

## 步骤

1. 从您当前配置的软件仓库中找到存储库 URL：

```
$ sudo yum -v repolist rhel-8-for-x86_64-baseos-rpms | grep repo-baseurl
Repo-baseurl :
```

以下输出是一个示例：

```
https://satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV/content/dist/rhel8/8/x86_64/baseos/os
```

2. 将硬编码的存储库修改为 Satellite 服务器。

- a. 创建具有 **0755** 权限的存储库目录：

```
$ sudo mkdir -pvm 0755 /etc/osbuild-composer/repositories
```

- b. 将 **/usr/share/osbuild-composer/repositories suit.json** 中的内容复制到您创建的目录中：

```
$ sudo cp /usr/share/osbuild-composer/repositories/*.json /etc/osbuild-composer/repositories/
```

- c. 通过 **/content/dist** configuration 行更新 Satellite URL 和文件内容：

```
$ sudo sed -i -e
's|cdn.redhat.com|satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV|'
/etc/osbuild-composer/repositories/*.json
```

- d. 验证配置是否已正确替换：

```
$ sudo vi /etc/osbuild-composer/repositories/rhel-8.json
```

3. 重启服务：

```
$ sudo systemctl restart osbuild-worker@1.service osbuild-composer.service
```

4. 覆盖红帽镜像构建器配置中所需的系统存储库，并将 Satellite 存储库的 URL 用作 baseurl。请参阅 [覆盖系统存储库](#)。

## 其他资源

- 当在 Satellite 上定义了多个自定义软件仓库时，RHEL 镜像构建器会失败

## 3.8. 使用 SATELLITE CV 作为软件仓库在 RHEL 镜像构建器中构建镜像

配置 RHEL 镜像构建器，以使用 Satellite 的内容视图(CV)作为存储库来构建您的自定义镜像。

### 先决条件

- 您已将 Satellite 与 RHEL web 控制台集成。请参阅 [在 Satellite 中启用 RHEL web 控制台](#)

### 步骤

1. 在 Satellite Web UI 中，导航到 **Content > Products**，选择您的 **Product** 并点您要使用的存储库。
2. 在 Published 字段中搜索安全 URL (HTTPS)并复制它。
3. 使用您复制的 URL 作为红帽镜像构建器存储库的 baseurl。请参阅 [在 RHEL 镜像构建器中添加自定义第三方存储库](#)。

### 后续步骤

- 构建镜像。请参阅 [在 web 控制台界面中使用 RHEL 镜像构建器 创建系统镜像](#)。

## 第 4 章 在 RHEL WEB 控制台中使用镜像构建器制作 RHEL FOR EDGE 镜像

使用 RHEL 镜像构建器创建一个自定义的 RHEL for Edge 镜像(OSTree 提交)。

要访问 RHEL 镜像构建器并创建自定义的 RHEL for Edge 镜像，您可以使用 RHEL web 控制台界面或命令行界面。

您可以通过执行以下高级别步骤，在 RHEL web 控制台中使用 RHEL 镜像构建器制作 RHEL for Edge 镜像：

1. 在 RHEL web 控制台中访问 RHEL 镜像构建器
2. 为 RHEL for Edge 镜像创建一个蓝图。
3. 创建 RHEL for Edge 镜像。您可以创建以下镜像：
  - RHEL for Edge Commit 镜像。
  - RHEL for Edge 容器镜像。
  - RHEL for Edge 安装程序镜像。
4. 下载 RHEL for Edge 镜像

### 4.1. 在 RHEL WEB 控制台中访问 RHEL 镜像构建器

要在 RHEL web 控制台中访问 RHEL 镜像构建器，请确保您满足以下先决条件，并按照以下流程操作。

#### 先决条件

- 已安装 RHEL 系统。
- 您对系统具有管理权限。
- 您已向 Red Hat Subscription Manager (RHSM)或 Red Hat Satellite Server 订阅了 RHEL 系统。
- 系统开机并通过网络访问。
- 您已在系统上安装了 RHEL 镜像构建器。

#### 步骤

1. 在 RHEL 系统中，在网页浏览器中访问 <https://localhost:9090/>。
2. 有关如何远程访问 RHEL 镜像构建器的更多信息，请参阅 [使用 RHEL 9 web 控制台管理系统](#) 文档。
3. 使用管理用户帐户登录 Web 控制台。
4. 在 Web 控制台中，在左侧菜单中点 **Apps**。
5. 单击 **Image Builder**。  
RHEL 镜像构建器仪表盘在右侧窗格中打开。现在，您可以继续为 RHEL for Edge 镜像创建蓝图。

## 4.2. 在 WEB 控制台中使用镜像构建器为 RHEL FOR EDGE 镜像创建蓝图

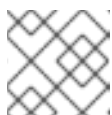
要在 RHEL web 控制台中使用 RHEL 镜像构建器为 RHEL for Edge 镜像创建蓝图，请确保您满足以下先决条件，然后按照流程操作。

### 先决条件

- 在 RHEL 系统上，您已打开了 RHEL 镜像构建器仪表盘。

### 流程

1. 在 RHEL 镜像构建器仪表盘上，点 **Create Blueprint**。  
此时会打开 **Create Blueprint** 对话框。
2. 在 **Details** 页面上：
  - a. 输入蓝图的名称，以及可选的描述。点 **Next**。
3. 可选：在 **Packages** 页面中：
  - a. 在 **Available packages** 搜索中，输入软件包名称并点击 **>** 按钮将其移到所选择的软件包字段。搜索并包含尽可能多的软件包。点 **Next**。



### 注意

除非另有指定，否则这些自定义都是可选的。

4. 在 **Kernel** 页面中，输入内核名称和命令行参数。
5. 在 **File system** 页面中，选择 **Use automatic partitioning**。OSTree 系统不支持文件系统自定义，因为 OSTree 镜像有自己的挂载规则，如只读。点 **Next**。
6. 在 **Services** 页面上，您可以启用或禁用服务：
  - a. 输入您要启用或禁用的服务名称，用逗号、空格分隔它们或按 **Enter** 键。点 **Next**。
7. 在 **Firewall** 页面中，设置防火墙设置：
  - a. 输入 **端口**，以及您要启用或禁用的防火墙服务。
  - b. 点 **Add zone** 按钮，为每个区域单独管理您的防火墙规则。点 **Next**。
8. 在 **Users** 页面中，按照以下步骤添加用户：
  - a. 单击 **Add user**。
  - b. 输入 **Username**、**password**，以及 **SSH key**。您还可以单击 **Server administrator** 复选框，将用户标记为特权用户。点 **Next**。
9. 在 **Groups** 页面中，完成以下步骤来添加组：
  - a. 点 **Add groups** 按钮：
    - i. 输入 **Group name** 和 **Group ID**。您可以添加更多组。点 **Next**。
10. 在 **SSH keys** 页面中，添加一个密钥：

- a. 点 **Add key** 按钮。
  - i. 输入 SSH 密钥。
  - ii. 输入 **User**。点 **Next**。
11. 在 **Timezone** 页面中，设置您的时区设置：
  - a. 在 **Timezone** 字段中输入您要添加到系统镜像的时区。例如，添加以下时区格式："US/Eastern"。  
如果您没有设置时区，系统默认使用 Universal Time, Coordinated (UTC)作为默认值。
  - b. 输入 **NTP** 服务器。点 **Next**。
12. 在 **Locale** 页面中完成以下步骤：
  - a. 在 **Keyboard** 搜索字段中，输入您要添加到系统镜像的软件包名称。例如：["en\_US.UTF-8"]。
  - b. 在 **Languages** 搜索字段中，输入您要添加到系统镜像的软件包名称。例如："us"。点 **Next**。
13. 在 **Others** 页面中，完成以下步骤：
  - a. 在 **Hostname** 字段中输入您要添加到系统镜像的主机名。如果没有添加主机名，操作系统会决定主机名。
  - b. 只对 Simplifier Installer 镜像强制：在 **Installation Devices** 字段中输入您的系统镜像的有效节点。例如：**dev/sda**。点 **Next**。
14. 仅在构建 FIDO 镜像时才强制：在 **FIDO device onboarding** 页面中完成以下步骤：
  - a. 在 **Manufacturing server URL** 字段中输入以下信息：
    - i. 在 **DIUN public key insecure** 字段中，输入不安全的公钥。
    - ii. 在 **DIUN public key hash** 字段中，输入公钥哈希。
    - iii. 在 **DIUN public key root certs** 字段中，输入公钥根证书。点 **Next**。
15. 在 **OpenSCAP** 页面中，完成以下步骤：
  - a. 在 **Datastream** 字段中输入您要添加到系统镜像的 **datastream** 补救指令。
  - b. 在 **Profile ID** 字段中，输入您要添加到系统镜像的 **profile\_id** 安全配置文件。点 **Next**。
16. 仅在构建 Ignition 镜像时才强制：在 **Ignition** 页面中，完成以下步骤：
  - a. 在 **Firstboot URL** 字段中输入您要添加到系统镜像的软件包名称。
  - b. 在 **Embedded Data** 字段中，拖动或上传您的文件。点 **Next**。
17. 在 **Review** 页面中，查看蓝图的详情。点 **Create**。

RHEL 镜像构建器视图打开，列出了现有蓝图。

### 4.3. 在 WEB 控制台使用镜像构建器创建 RHEL FOR EDGE COMMIT 镜像

您可以在 RHEL web 控制台使用 RHEL 镜像构建器创建 "RHEL for Edge Commit" 镜像。"RHEL for Edge Commit (.tar)" 镜像类型包含一个完整的操作系统，但它不能直接启动。要引导 Commit 镜像类型，您必须将它部署在正在运行的容器中。

### 先决条件

- 在 RHEL 系统上，您已访问 RHEL 镜像构建器仪表盘。

### 流程

1. 在 RHEL 镜像构建器仪表盘上，单击 **Create Image**。
2. 在 **Image output** 页面中执行以下步骤：
  - a. 在 **Select a blueprint** 下拉菜单中选择您要使用的蓝图。
  - b. 从 **Image output type** 下拉列表中，选择 "RHEL for Edge Commit (.tar)"。
  - c. 点 **Next**。
  - d. 在 **OSTree settings** 页面中，输入：
    - i. **Repository URL**：指定要嵌入镜像中提交的 OSTree 存储库的 URL。例如：  
`http://10.0.2.2:8080/repo/`。
    - ii. **Parent commit**：指定之前的提交，如果此时还没有提交，则保留为空。
    - iii. 在 **Ref** 文本框中，指定要在哪里创建提交的引用路径。默认情况下，Web 控制台指定 `rhel/9/$ARCH/edge`。"\$ARCH" 值由主机机器决定。点 **Next**。
  - e. 在 **Review** 页面中，检查自定义并点 **Create**。  
RHEL 镜像构建器开始为您创建的蓝图创建一个 RHEL for Edge Commit 镜像。



### 注意

镜像创建过程需要 20 分钟才能完成。

### 验证

1. 检查 RHEL for Edge Commit 镜像创建进度：
  - a. 点 **Images** 选项卡。

镜像创建过程完成后，您可以下载生成的 "RHEL for Edge Commit(.tar)" 镜像。

### 其他资源

- [下载 RHEL for Edge 镜像](#)

## 4.4. 在 RHEL WEB 控制台使用 RHEL 镜像构建器创建 RHEL FOR EDGE 容器镜像

您可以通过选择 "RHEL for Edge Container (.tar)" 创建 RHEL for Edge 镜像。RHEL for Edge **Container(.tar)** 镜像类型会创建一个 OSTree 提交，并将其嵌入到带有 web 服务器的 OCI 容器中。容器启动后，Web 服务器将提交充当 OSTree 存储库。

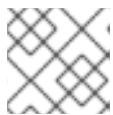
按照此流程中的步骤，再 RHEL web 控制台使用镜像构建器创建 RHEL for Edge 容器镜像。

### 先决条件

- 在 RHEL 系统上，您已访问 RHEL 镜像构建器仪表盘。
- 您已创建了蓝图。

### 流程

1. 在 RHEL 镜像构建器仪表盘上，单击 **Create Image**。
2. 在 **Image output** 页面中执行以下步骤：
  - a. 从 **Image output type** 下拉列表中，选择 **"RHEL for Edge Container (.tar)"**。
  - b. 点 **Next**。
  - c. 在 **OSTree** 页面中，输入：
    - i. **Repository URL**：指定要嵌入镜像中提交的 OSTree 存储库的 URL。例如：  
 http://10.0.2.2:8080/repo/。默认情况下，RHEL for Edge 容器镜像的存储库文件夹为 **"/repo"**。  
 要查找要使用的正确 URL，请访问正在运行的容器并检查 **nginx.conf** 文件。要查找要使用哪个 URL，请访问正在运行的容器并检查 **nginx.conf** 文件。在 **nginx.conf** 文件中，找到 **root** 目录条目，以搜索 **/repo/** 文件夹信息。请注意，如果您在使用 RHEL 镜像构建器创建 RHEL for Edge 容器镜像 **(.tar)** 时没有指定存储库 URL，则会在 **nginx.conf** 文件中创建默认的 **/repo/** 条目。
    - ii. **Parent commit**：指定之前的提交，如果此时还没有提交，则保留为空。
    - iii. 在 **Ref** 文本框中，指定要在哪里创建提交的引用路径。默认情况下，Web 控制台指定 **rhel/9/\$ARCH/edge**。"\$ARCH" 值由主机机器决定。点 **Next**。
  - d. 在 **Review** 页面上，检查自定义。点 **Save blueprint**。
4. 点 **Create**。  
RHEL 镜像构建器开始为您创建的蓝图创建 RHEL for Edge 容器镜像。



#### 注意

镜像创建过程需要 20 分钟才能完成。

### 验证

1. 检查 RHEL for Edge Container 镜像创建进度：
  - a. 点 **Images** 选项卡。

镜像创建过程完成后，您可以下载生成的 **"RHEL for Edge Container(.tar)"** 镜像。

### 其他资源

- [下载 RHEL for Edge 镜像](#)

## 4.5. 在 RHEL WEB 控制台中使用镜像构建器创建 RHEL FOR EDGE INSTALLER 镜像

您可以通过选择 **RHEL for Edge Installer (.iso)** 来创建 RHEL for Edge Installer 镜像。 **RHEL for Edge Installer (.iso)** 镜像类型从 **RHEL for Edge Container (.tar)** 服务的正在运行的容器中拉取 OSTree 提交存储库，并创建一个带有配置为使用嵌入式 OSTree 提交的 Kickstart 文件的可安装的引导 ISO 镜像。

按照此流程中的步骤，在 RHEL web 控制台中使用镜像构建器创建 RHEL for Edge 镜像。

### 先决条件

- 在 RHEL 系统上，您已访问了镜像构建器仪表盘。
- 您创建了蓝图。
- 您已创建了 RHEL for Edge 容器镜像，并将其加载到正在运行的容器中。请参阅[为非基于网络的部署创建 RHEL for Edge 容器镜像](#)。

### 流程

1. 在 RHEL 镜像构建器仪表盘上，单击 **Create Image**。
2. 在 **Image output** 页面中执行以下步骤：
  - a. 在 **Select a blueprint** 下拉菜单中选择您要使用的蓝图。
  - b. 从 **Image output type** 下拉列表中，选择 **RHEL for Edge Installer (.iso)** 镜像。
  - c. 点 **Next**。
  - d. 在 **OSTree settings** 页面中，输入：
    - i. **Repository URL**：指定要嵌入镜像中提交的 OSTree 存储库的 URL。例如：  
`http://10.0.2.2:8080/repo/`
    - ii. 在 **Ref** 文本框中，指定要在哪里创建提交的引用路径。默认情况下，Web 控制台指定 **rhel/9/\$ARCH/edge**。"\$ARCH" 值由主机机器决定。点 **Next**。
  - e. 在 **Review** 页面上，检查自定义。点 **Save blueprint**。
3. 点 **Create**。  
RHEL 镜像构建器开始为您创建的蓝图创建 RHEL for Edge 安装程序镜像。



### 注意

镜像创建过程需要 20 分钟才能完成。

### 验证

1. 检查 RHEL for Edge Installer 镜像创建进度：
  - a. 点 **Images** 选项卡。

镜像创建过程完成后，您可以下载生成的 **RHEL for Edge 安装程序(.iso)** 镜像并将 ISO 镜像引导到设备中。



## 其他资源

- [下载 RHEL for Edge 镜像](#)

## 4.6. 下载 RHEL FOR EDGE 镜像

使用 RHEL 镜像构建器成功创建 RHEL for Edge 镜像后，将镜像下载到本地主机上。

### 流程

下载镜像：

1. 在 **More Options** 菜单中点 **Download**。  
RHEL 镜像构建器工具将文件下载到您的默认下载位置。

下载的文件包含一个 **.tar** 文件，其中包含适用于 RHEL for Edge Commit 和 RHEL for Edge 容器镜像的 OSTree 存储库，或 RHEL for Edge 安装程序镜像（带有 OSTree 存储库）的 **.iso** 文件。此存储库包含提交和 **json** 文件，其中包含有关存储库内容的信息元数据。

## 4.7. 其他资源

- [使用镜像构建器命令行制作 RHEL for Edge 镜像](#)。

## 第 5 章 使用镜像构建器命令行制作 RHEL FOR EDGE 镜像

您可以使用镜像构建器创建自定义的 RHEL for Edge 镜像(OSTree 提交)。

要访问镜像构建器并创建自定义的 RHEL for Edge 镜像，您可以使用 RHEL web 控制台界面或命令行界面。

对于基于网络的部署，使用 CLI 编写 RHEL for Edge 镜像的工作流涉及以下高级别步骤：

1. 为 RHEL for Edge 镜像创建蓝图
2. 创建 RHEL for Edge Commit 镜像
3. 下载 RHEL for Edge Commit 镜像

对于非基于网络的部署，使用 CLI 编写 RHEL for Edge 镜像的工作流涉及以下高级别步骤：

1. 为 RHEL for Edge 镜像创建蓝图
2. 为 RHEL for Edge Installer 镜像创建一个蓝图
3. 创建 RHEL for Edge 容器镜像
4. 为 Edge 安装程序创建 RHEL
5. 下载 RHEL for Edge 镜像

要执行这些步骤，请使用 **composer-cli** 软件包。



### 注意

要以非 root 身份运行 **composer-cli** 命令，您必须是 **weldr** 组的一部分，或者您必须具有系统的管理员访问权限。

## 5.1. 基于网络的部署工作流

这提供了如何构建 **OSTree** 提交的步骤。这些 **OSTree** 提交包含完整的操作系统，但不能直接启动。要引导它们，您需要使用 Kickstart 文件进行部署。

### 5.1.1. 使用镜像构建器命令行界面创建 RHEL for Edge Commit 镜像蓝图

使用 CLI 为 RHEL for Edge Commit 镜像创建一个蓝图。

#### 前提条件

- 您没有现有的蓝图。要验证，列出现有的蓝图：

```
$ sudo composer-cli blueprints list
```

#### 流程

1. 以 TOML 格式创建一个纯文本文件，其内容如下：

```
name = "blueprint-name"
```

```
description = "blueprint-text-description"
version = "0.0.1"
modules = [ ]
groups = [ ]
```

其中,

- *blueprint-name* 是名称, print-text-description 是您的蓝图的描述。
  - 0.0.1 是 Semantic Versioning 方案的版本号。
  - *模块* 描述了要安装到镜像中的软件包名称和匹配版本的 glob, 例如: 软件包名称 = "tmux", 匹配的版本 glob 是 version = "2.9a"。  
请注意, 目前软件包和模块之间没有区别。
  - *组* 是要安装到镜像中的软件包组, 如组软件包 anaconda-tools。  
此时, 如果您不知道模块和组, 请将它们留空。
2. 包含所需的软件包, 并在蓝图中自定义其他详情以满足您的要求。  
对于要包含在蓝图中的每个软件包, 请在文件中添加以下行:

```
[[packages]]
name = "package-name"
version = "package-version"
```

其中,

- package-name 是软件包的名称, 如 httpd、gdb-doc 或 coreutils。
  - package-version 是您要使用的软件包的版本号。  
package-version 支持以下 dnf 版本规格:
  - 对于特定版本, 请使用准确版本号, 如 9.0。
  - 对于最新可用版本, 请使用星号 \*。
  - 对于最新的次版本, 请使用格式 (如 9.\*)。
3. 将蓝图推送 (导入) 到 RHEL 镜像构建器服务器:

```
# composer-cli blueprints push blueprint-name.toml
```

4. 列出现有的蓝图, 以检查创建的蓝图是否已成功推送并存在。

```
# composer-cli blueprints show BLUEPRINT-NAME
```

5. 检查蓝图中列出的组件和版本是否有效:

```
# composer-cli blueprints depsolve blueprint-name
```

## 其他资源

- [支持的镜像自定义](#)

## 5.1.2. 使用镜像构建器命令行界面创建 RHEL for Edge Commit 镜像

要使用 RHEL 镜像构建器命令行界面创建 RHEL for Edge Commit 镜像，请确保您满足以下先决条件，并按照流程操作。

### 先决条件

- 您已为 RHEL for Edge Commit 镜像创建了一个蓝图。

### 流程

1. 创建 RHEL for Edge Commit 镜像。

```
# composer-cli compose start blueprint-name image-type
```

其中,

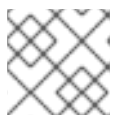
- *blueprint-name* 是 RHEL for Edge 蓝图名称。
- *image-type* 是 **edge-commit** (对于 **network-based deployment**) 。这时将显示一个确认已添加到队列中的 `composer` 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。

2. 检查镜像 `compose` 状态。

```
# composer-cli compose status
```

输出以以下格式显示状态：

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



### 注意

镜像创建过程需要 20 分钟才能完成。

要中断镜像创建过程，请运行：

```
# composer-cli compose cancel <UUID>
```

要删除现有镜像，请运行：

```
# composer-cli compose delete <UUID>
```

镜像就绪后，您可以下载该镜像并使用**网络部署**上的镜像。

### 其他资源

- [使用 RHEL 镜像构建器命令行制作 RHEL for Edge 镜像](#)

## 5.1.3. 使用 RHEL 镜像构建器 CLI 使用 `ref` 提交创建 RHEL for Edge 镜像更新

如果您在现有蓝图中进行了更改，例如，您添加了一个新的软件包，并希望使用这个新软件包更新现有的

RHEL for Edge 镜像，则您可以使用 `--parent` 参数来生成更新的 **RHEL for Edge Commit (.tar)** 镜像。`--parent` 参数可以是 `URL` 参数指定的存储库中存在的一个 **ref**，您也可以使用您在提取的 `.tar` 镜像文件中找到的 **Commit ID**。`ref` 和 **Commit ID** 参数检索您要构建的新提交的父项。RHEL 镜像构建器可以从影响您要构建的新提交的父提交中读取信息。因此，RHEL 镜像构建器读取父提交的用户数据库，并为创建软件包的系统用户和组保留 UID 和 GID。

### 先决条件

- 您已为 RHEL for Edge 镜像更新了现有蓝图。
- 您有一个现有的 RHEL for Edge 镜像(OSTree commit)。请参阅[提取 RHEL for Edge 镜像提交](#)
- 正在构建的 **ref** 位于 URL 指定的 **OSTree** 存储库中。

### 流程

1. 创建 RHEL for Edge commit 镜像：

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --parent parent-OSTree-REF -
-url URL blueprint-name image-type
```

例如：

- 要根据 **parent** 和新的 **ref** 创建新的 RHEL for Edge 提交，请运行以下命令：

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --parent
rhel/9/x86_64/edge --url http://10.0.2.2:8080/repo rhel_update edge-commit
```

- 要根据同一 **ref** 创建新的 RHEL for Edge 提交，请运行以下命令：

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url
http://10.0.2.2:8080/repo rhel_update edge-commit
```

其中：

- `--ref` 参数指定用于构建 OSTree 存储库的相同路径值。
- `--parent` 参数指定父提交。它可以是要解析和拉取的 `ref`（如 `rhel/9/x86_64/edge`），或者您可以在提取的 `.tar` 文件中找到的 **Commit ID**。
- `blueprint-name` 是 RHEL for Edge 蓝图名称。
- `--url` 参数指定要嵌入到镜像中的提交的 OSTree 存储库的 URL，例如 `http://10.0.2.2:8080/repo`。
- `image-type` 是 **edge-commit**（对于 **network-based deployment**）。



### 注意

- `--parent` 参数只能用于 **RHEL for Edge Commit (.tar)** 镜像类型。将 `--url` 和 `--parent` 参数一起使用会导致 **RHEL for Edge Container (.tar)** 镜像类型出错。
- 如果省略 `parent ref` 参数，系统会退回到 `--ref` 参数指定的 `ref`。

这时将显示一个确认已添加到队列中的 composer 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。

## 2. 检查镜像 compose 状态。

```
# composer-cli compose status
```

输出以以下格式显示状态：

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



### 注意

完成镜像创建过程需要几分钟时间。

(可选) 要中断镜像创建过程，请运行：

```
# composer-cli compose cancel <UUID>
```

(可选) 要删除现有镜像，请运行：

```
# composer-cli compose delete <UUID>
```

镜像创建完成后，要升级现有的 OSTree 部署，您需要：

- 设置存储库。请参阅[部署 RHEL for Edge 镜像](#)。
- 将此存储库添加为远程，也就是托管 OSTree 内容的 http 或 https 端点。
- 将新 OSTree 提交拉取到其现有的正在运行的实例。请参阅[手动部署 RHEL for Edge 镜像更新](#)。

## 其他资源

- [在命令行界面中，使用 RHEL 镜像构建器创建系统镜像。](#)
- [使用 RHEL 镜像构建器命令行界面下载 RHEL for Edge 镜像。](#)

### 5.1.4. 使用镜像构建器命令行界面下载 RHEL for Edge 镜像

要使用 RHEL 镜像构建器命令行界面下载 RHEL for Edge 镜像，请确保您满足以下先决条件，然后按照以下流程操作。

#### 先决条件

- 您已创建了 RHEL for Edge 镜像。

#### 流程

##### 1. 查看 RHEL for Edge 镜像状态。

```
# composer-cli compose status
```

输出必须显示以下内容：

```
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
```

2. 下载镜像。

```
# composer-cli compose image <UUID>
```

RHEL 镜像构建器将镜像作为 **tar** 文件下载到当前目录。

UUID 号和镜像大小会同时显示。

```
$ <UUID>-commit.tar: size MB
```

镜像包含提交和 **json** 文件，其中包含有关存储库内容的信息元数据。

### 其他资源

- [在基于网络的环境中部署 RHEL for Edge 镜像](#)

## 5.2. 非基于网络的部署工作流

要构建一个使用 "RHEL for Edge Container" 和 "RHEL for Edge Installer" 镜像安装基于 OSTree 系统的引导 ISO 镜像，并且以后可以在断开连接的环境中部署到设备，请按照以下步骤执行。

### 5.2.1. 使用镜像构建器 CLI 创建 RHEL for Edge 容器蓝图

要为 RHEL for Edge 容器镜像创建蓝图，请执行以下步骤：

#### 流程

1. 以 TOML 格式创建一个纯文本文件，其内容如下：

```
name = "blueprint-name"
description = "blueprint-text-description"
version = "0.0.1"
modules = [ ]
groups = [ ]
```

其中，

- *blueprint-name* 是名称，*print-text-description* 是您的蓝图的描述。
  - *0.0.1* 是 Semantic Versioning 方案的版本号。
  - *模块* 描述了要安装到镜像中的软件包名称和匹配版本的 glob，例如：软件包名称 = "tmux"，匹配的版本 glob 是 *version = "2.9a"*。  
请注意，目前软件包和模块之间没有区别。
  - *组* 是要安装到镜像中的软件包组，如组软件包 *anaconda-tools*。  
此时，如果您不知道模块和组，请将它们留空。
2. 包含所需的软件包，并在蓝图中自定义其他详情以满足您的要求。

对于要包含在蓝图中的每个软件包，请在文件中添加以下行：

```
[[packages]]
name = "package-name"
version = "package-version"
```

其中，

- `package-name` 是软件包的名称，如 `httpd`、`gdb-doc` 或 `coreutils`。
  - `package-version` 是您要使用的软件包的版本号。  
`package-version` 支持以下 `dnf` 版本规范：
  - 对于特定版本，请使用准确版本号，如 9.0。
  - 对于最新可用版本，请使用星号 `*`。
  - 对于最新的次版本，请使用格式（如 9.\*）。
3. 将蓝图推送（导入）到 RHEL 镜像构建器服务器：

```
# composer-cli blueprints push blueprint-name.toml
```

4. 列出现有的蓝图，以检查创建的蓝图是否已成功推送并存在。

```
# composer-cli blueprints show BLUEPRINT-NAME
```

5. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve blueprint-name
```

## 其他资源

- [支持的镜像自定义](#)

### 5.2.2. 使用镜像构建器 CLI 创建 RHEL for Edge Installer 蓝图

您可以创建一个蓝图来构建 **RHEL for Edge Installer(.iso)** 镜像，并指定用户帐户以在安装时自动在系统中创建一个或多个用户。



#### 警告

当您在带有 `customizations.user` 自定义的蓝图中创建用户时，蓝图会在 `/usr/lib/passwd` 目录下创建用户，在 `/usr/etc/shadow` 目录下创建密码。请注意，您无法使用 `OSTree` 更新在运行的系统中更改镜像其他版本中的密码。使用蓝图创建的用户必须仅用于获得对创建的系统的访问权限。访问系统后，您需要创建用户，例如，使用 `useradd` 命令。

要为 RHEL for Edge Installer 镜像创建一个蓝图，请执行以下步骤：



## 流程

1. 以 TOML 格式创建一个纯文本文件，其内容如下：

```
name = "blueprint-installer"
description = "blueprint-for-installer-image"
version = "0.0.1"

[[customizations.user]]
name = "user"
description = "account"
password = "user-password"
key = "user-ssh-key"
home = "path"
groups = ["user-groups"]
```

其中，

- *blueprint-name* 是名称，*print-text-description* 是您的蓝图的描述。
  - *0.0.1* 是 Semantic Versioning 方案的版本号。
2. 将蓝图推送（导入）到 RHEL 镜像构建器服务器：

```
# composer-cli blueprints push blueprint-name.toml
```

3. 列出现有的蓝图，以检查创建的蓝图是否已成功推送并存在。

```
# composer-cli blueprints show blueprint-name
```

4. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve blueprint-name
```

## 其他资源

- [支持的镜像自定义](#)

### 5.2.3. 使用镜像构建器 CLI 创建 RHEL for Edge 容器镜像

要使用 RHEL 镜像构建器命令行界面创建 RHEL for Edge 容器镜像，请确保您满足以下先决条件，并按照流程操作。

#### 先决条件

- 您已为 RHEL for Edge 容器镜像创建了一个蓝图。

#### 流程

1. 创建 RHEL for Edge 容器镜像。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url URL-OSTree-repository
blueprint-name image-type
```

其中,

- `--ref` 与客户用来构建 OSTree 存储库的值相同
- `--url` 是要嵌入到镜像中的提交的 OSTree 存储库的 URL。例如：  
<http://10.0.2.2:8080/repo/>。默认情况下，RHEL for Edge 容器镜像的存储库文件夹为 `/repo`。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。  
 要查找要使用的正确 URL，请访问正在运行的容器并检查 `nginx.conf` 文件。要查找要使用哪个 URL，请访问正在运行的容器并检查 `nginx.conf` 文件。在 `nginx.conf` 文件中，找到 `root` 目录条目，以搜索 `/repo/` 文件夹信息。请注意，如果您在使用 RHEL 镜像构建器创建 RHEL for Edge 容器镜像 (`.tar`) 时没有指定存储库 URL，则会在 `nginx.conf` 文件中创建默认的 `/repo/` 条目。
- `blueprint-name` 是 RHEL for Edge 蓝图名称。
- `image-type` 是用于非基于网络的部署的 `edge-container`。  
 这时将显示一个确认已添加到队列中的 `composer` 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。

## 2. 检查镜像 compose 状态。

```
# composer-cli compose status
```

输出以以下格式显示状态：

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



### 注意

镜像创建过程需要 20 分钟才能完成。

要中断镜像创建过程，请运行：

```
# composer-cli compose cancel <UUID>
```

要删除现有镜像，请运行：

```
# composer-cli compose delete <UUID>
```

镜像就绪后，它可用于非网络部署。请参阅 [为非基于网络的部署创建 RHEL for Edge 容器镜像](#)。

## 其他资源

- [使用 RHEL 镜像构建器命令行制作 RHEL for Edge 镜像](#)

## 5.2.4. 使用命令行界面为非网络部署创建 RHEL for Edge 安装程序镜像

要创建一个嵌入了 OSTree 提交的 RHEL for Edge 安装程序镜像，请使用 RHEL 镜像构建器命令行界面，并确保您已满足以下先决条件，然后按照流程操作。

### 先决条件

- 您已为 RHEL for Edge Installer 镜像创建了蓝图。

- 您已创建了一个 RHEL for Edge 容器镜像，并使用 web 服务器部署了它。

## 流程

1. 开始创建 RHEL for Edge Installer 镜像。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url URL-OSTree-repository
blueprint-name image-type
```

其中,

- *ref* 与客户用来构建 OSTree 存储库的值相同
  - *URL-OSTree-repository* 是要嵌入到镜像中的提交 OSTree 存储库的 URL。例如：  
<http://10.0.2.2:8080/repo>。请参阅[为非基于网络的部署创建 RHEL for Edge 容器镜像](#)。
  - *blueprint-name* 是 RHEL for Edge Installer 蓝图名称。
  - *image-type* 是 **edge-installer**。  
这时将显示一个确认已添加到队列中的 `composer` 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。
2. 检查镜像 compose 状态。

```
# composer-cli compose status
```

命令输出以以下格式显示状态：

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



### 注意

完成镜像创建过程需要几分钟时间。

要中断镜像创建过程，请运行：

```
# composer-cli compose cancel <UUID>
```

要删除现有镜像，请运行：

```
# composer-cli compose delete <UUID>
```

镜像就绪后，您可以将它用于[非网络部署](#)。请参阅[为非基于网络的部署安装 RHEL for Edge 镜像](#)。

## 5.2.5. 使用镜像构建器 CLI 下载 RHEL for Edge Installer 镜像

要使用 RHEL 镜像构建器命令行界面下载 RHEL for Edge Installer 镜像，请确保您满足以下先决条件，然后按照以下流程操作。

### 先决条件

- 您已创建了 RHEL for Edge Installer 镜像。

## 流程

1. 查看 RHEL for Edge 镜像状态。

```
# composer-cli compose status
```

输出必须显示以下内容：

```
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
```

2. 下载镜像。

```
# composer-cli compose image <UUID>
```

RHEL 镜像构建器将镜像作为 **.iso** 文件下载到当前目录。

UUID 号和镜像大小会同时显示。

```
$ <UUID>-boot.iso: size MB
```

生成的镜像是可引导 ISO 镜像。

## 其他资源

- [在非基于网络的环境中部署 RHEL for Edge 镜像。](#)

## 5.3. 支持的镜像自定义

您可以通过在蓝图中添加自定义来自定义镜像，例如：

- 添加一个额外的 RPM 软件包
- 启用服务
- 自定义一个内核命令行参数。

在其他参数之间。您可以在蓝图中使用多个镜像自定义。通过使用自定义配置，您可以将软件包和组添加到默认软件包中不提供的镜像中。要使用这些选项，请在蓝图中配置自定义，并将其导入(push)到 RHEL 镜像构建器中。

## 其他资源

- [在添加文件系统自定义 "size" 后，蓝图导入会失败。](#)

### 5.3.1. 选择一个发行版

您可以使用 **distro** 字段选择在制作镜像时使用的发行版，或者解决蓝图中的依赖问题。如果 **distro** 留空，它将使用主机发行版。如果没有指定发行版，蓝图将使用主机分发。如果您升级主机操作系统，则没有发行版集合的蓝图使用新的操作系统版本构建镜像。您无法构建一个与 RHEL 镜像构建器主机不同的操作系统镜像。

## 流程

- 使用 RHEL 发行版自定义蓝图，以始终构建指定的 RHEL 镜像：

```
name = "blueprint_name"
description = "blueprint_version"
version = "0.1"
distro = "different_minor_version"
```

替换 "**different\_minor\_version**"，来构建一个不同的次版本，例如，如果要构建 RHEL 9.4 镜像，请使用 **distro** = "rhel-94"。在 RHEL 9.3 镜像上，您可以构建次版本，如 RHEL 9.3、RHEL 8.9 和更早的版本。

### 5.3.2. 选择一个软件包组

自定义带有软件包和模块的蓝图。**name** 属性是必需的字符串。**version** 属性是一个可选字符串，如果未提供，则使用存储库中的最新版本。



#### 注意

目前，**osbuild-composer** 中的软件包和模块之间没有区别。两者都被视为 RPM 软件包依赖项。

#### 流程

- 自定义带有软件包的蓝图：

```
[[packages]]
name = "package_group_name"
```

将 "**package\_group\_name**" 替换为组的名称。例如，"tmux"。

```
[[packages]]
name = "tmux"
version = "2.9a"
```

### 5.3.3. 设置镜像主机名

**customizations.hostname** 是一个可选字符串，您可以用来配置最终镜像主机名。此自定义是可选的，如果您未设置它，则蓝图使用默认主机名。

#### 流程

- 自定义蓝图以配置主机名：

```
[customizations]
hostname = "baseimage"
```

### 5.3.4. 指定其他用户

将用户添加到镜像中，并可选择设置其 SSH 密钥。本节的所有字段都是可选的，但 **name** 除外。

#### 流程

- 自定义蓝图来将用户添加到镜像中：

```
[[customizations.user]]
name = "USER-NAME"
description = "USER-DESCRIPTION"
password = "PASSWORD-HASH"
key = "PUBLIC-SSH-KEY"
home = "/home/USER-NAME"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = NUMBER
gid = NUMBER
```

GID 是可选的，且必须在镜像中已存在。（可选）软件包会创建它，或者蓝图使用 `[[customizations.group]]` 条目创建 GID。

将 `PASSWORD-HASH` 替换为实际的 **密码哈希**。要生成 **密码哈希**，请使用如下命令：

```
$ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit()'
```

使用适当的值替换其他占位符。

输入 **name** 值，并省略您不需要的任何行。

为每个要包含的用户重复这个块。

### 5.3.5. 指定附加组

为生成的系统镜像指定一个组。**name** 和 **gid** 属性都是必需的。

#### 流程

- 自定义带有组的蓝图：

```
[[customizations.group]]
name = "GROUP-NAME"
gid = NUMBER
```

为每个组重复此块。

### 5.3.6. 为现有用户设置 SSH 密钥

您可以使用 `customizations.sshkey` 为最终镜像中的现有用户设置 SSH 密钥。**user** 和 **key** 属性是必需的。

#### 流程

- 通过为现有用户设置 SSH 密钥来自定义蓝图：

```
[[customizations.sshkey]]
user = "root"
key = "PUBLIC-SSH-KEY"
```



### 注意

您只能为现有用户配置 **customizations.sshkey** 自定义。要创建用户并设置 SSH 密钥，请参阅 **生成的系统镜像的用户规格** 自定义。

### 5.3.7. 附加一个内核参数

您可以向引导装载程序内核命令行中附加参数。默认情况下，RHEL 镜像构建器将默认内核构建到镜像中。但是，您可以通过在蓝图中配置它来自定义内核。

#### 流程

- 在默认值中附加内核引导选项：

```
[customizations.kernel]
append = "KERNEL-OPTION"
```

- 定义一个要在镜像中使用的内核名称

```
[customizations.kernel]
name = "KERNEL-rt"
```

### 5.3.8. 设置时区和 NTP

您可以自定义蓝图来配置时区和 *网络时间协议* (NTP)。 **timezone** 和 **ntpserver** 属性是可选字符串。如果您没有自定义时区，系统将使用 *Universal Time, Coordinated* (UTC)。如果您没有设置 NTP 服务器，系统将使用默认发行版。

#### 流程

- 自定义带有您想要的 **timezone** 和 **ntpserver** 的蓝图：

```
[customizations.timezone]
timezone = "TIMEZONE"
ntpserver = "NTP_SERVER"
```

例如：

```
[customizations.timezone]
timezone = "US/Eastern"
ntpserver = ["0.north-america.pool.ntp.org", "1.north-america.pool.ntp.org"]
```



### 注意

有些镜像类型，如 Google Cloud，已经建立了 NTP 服务器。您无法覆盖它，因为镜像需要 NTP 服务器来在所选环境中引导。但是，您可以在蓝图中自定义时区。

### 5.3.9. 自定义区域设置

您可以为生成的系统镜像自定义区域设置。 **language** 和 **keyboard** 属性是必需的。您可以添加许多其他语言。您添加的第一个语言是主语言，其他语言是次要语言。

#### 流程

```
'''
```

- 设置区域设置：

```
[customizations.locale]
languages = ["LANGUAGE"]
keyboard = "KEYBOARD"
```

例如：

```
[customizations.locale]
languages = ["en_US.UTF-8"]
keyboard = "us"
```

- 要列出语言支持的值，请运行以下命令：

```
$ localectl list-locales
```

- 要列出键盘支持的值，请运行以下命令：

```
$ localectl list-keymaps
```

### 5.3.10. 自定义防火墙

为生成的系统镜像设置防火墙。默认情况下，防火墙阻止进入的连接，但明确启用其端口的服务除外，如 **sshd**。

如果您不想使用 **[customizations.firewall]** 或 **[customizations.firewall.services]**，可以删除属性，或者将它们设置为空列表 []。如果您只想使用默认的防火墙设置，您可以从蓝图中省略自定义。



#### 注意

Google 和 OpenStack 模板为其环境明确禁用防火墙。您无法通过设置蓝图来覆盖此行为。

#### 流程

- 使用以下设置自定义蓝图，以打开其他端口和服务：

```
[customizations.firewall]
ports = ["PORTS"]
```

其中 **ports** 是包含要打开的端口或一系列端口和协议的可选字符串列表。您可以使用以下格式配置端口：**port:protocol** 格式。您可以使用 **portA-portB:protocol** 格式配置端口范围。例如：

```
[customizations.firewall]
ports = ["22:tcp", "80:tcp", "imap:tcp", "53:tcp", "53:udp", "30000-32767:tcp", "30000-32767:udp"]
```

您可以使用数字端口或 **/etc/services** 中的名称来启用或禁用端口列表。

- 在 **customizations.firewall.service** 部分中指定要启用或禁用哪个防火墙服务：



```
[customizations.firewall.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

- 您可以检查可用的防火墙服务：

```
$ firewall-cmd --get-services
```

例如：

```
[customizations.firewall.services]
enabled = ["ftp", "ntp", "dhcp"]
disabled = ["telnet"]
```



### 注意

**firewall.services** 中列出的服务与 **/etc/services** 文件中提供的 **service-names** 不同。

### 5.3.11. 启用或禁用服务

您可以控制在引导期间要启用哪些服务。有些镜像类型已经启用或禁用了服务，以确保镜像正常工作，您无法覆盖此设置。蓝图中的 **[customizations.services]** 设置不会替代这些服务，但可以将服务添加到已在镜像模板中存在的服务列表中。

#### 流程

- 自定义在引导时要启用哪些服务：

```
[customizations.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

例如：

```
[customizations.services]
enabled = ["sshd", "cockpit.socket", "httpd"]
disabled = ["postfix", "telnetd"]
```

### 5.3.12. 指定一个自定义文件系统配置

您可以在蓝图中指定自定义文件系统配置，因此可以创建具有特定磁盘布局，而不是默认布局配置的镜像。通过使用蓝图中的非默认布局配置，您可以受益于：

- 安全基准合规性
- 防止磁盘不足错误
- 提高的性能
- 与现有设置的一致性



## 注意

OSTree 系统不支持文件系统自定义，因为 OSTree 镜像有自己的挂载规则，如只读。不支持以下镜像类型：

- **image-installer**
- **edge-installer**
- **edge-simplified-installer**

另外，以下镜像类型不支持文件系统自定义，因为这些镜像类型不创建分区的操作系统镜像：

- **edge-commit**
- **edge-container**
- **tar**
- **container**

对于 RHEL 8.10 和 9.4 之前的发行版本，蓝图支持以下 **挂载点** 及其子目录：

- **/** - root 挂载点
- **/var**
- **/home**
- **/opt**
- **/srv**
- **/usr**
- **/app**
- **/data**
- **/tmp**

从 RHEL 9.4 和 8.10 发行版本开始，您可以指定任意自定义挂载点，除了为操作系统保留的特定路径外。

您不能对以下挂载点及其子目录指定任意自定义挂载点：

- **/bin**
- **/boot/efi**
- **/dev**
- **/etc**
- **/lib**
- **/lib64**
- **/lost+found**

- `/proc`
- `/run`
- `/sbin`
- `/sys`
- `/sysroot`
- `/var/lock`
- `/var/run`

您可以在蓝图中为 `/usr` 自定义挂载点自定义文件系统，但不允许对其子目录自定义文件系统。



### 注意

从 RHEL 9.0 发行版开始，才支持使用 CLI 自定义挂载点。在之前的发行版本中，您只能将 `root` 分区指定为挂载点，并将 `size` 参数指定为镜像大小的别名。

如果您在自定义镜像中有多个分区，您可以在 LVM 上创建带有自定义文件系统分区的镜像，并在运行时调整这些分区大小。为此，您可以在蓝图中指定自定义的文件系统配置，因此可以使用所需的磁盘布局创建镜像。默认文件系统布局保持不变 - 如果您使用没有文件系统自定义的普通镜像，`cloud-init` 会调整 `root` 分区的大小。

蓝图自动将文件系统自定义转换为 LVM 分区。

您可以使用自定义文件蓝图自定义来创建新文件或替换现有文件。您指定的文件的父目录必须存在，否则镜像构建会失败。通过在 `[[customizations.directories]]` 自定义中指定它来确保父目录存在。



### 警告

如果您将文件自定义与其他蓝图自定义相结合，这可能会影响其他自定义的功能，或者可能会覆盖当前的文件自定义。

#### 5.3.12.1. 在蓝图中指定自定义文件

使用 `[[customizations.files]]` 蓝图自定义，您可以：

- 创建新文本文件。
- 修改现有文件。警告：这可能会覆盖现有内容。
- 为您要创建的文件设置用户和组所有权。
- 以八进制格式设置模式权限。

您无法创建或替换以下文件：

- `/etc/fstab`

- `/etc/shadow`
- `/etc/passwd`
- `/etc/group`

您可以使用 `[[customizations.files]]` 和 `[[customizations.directories]]` 蓝图自定义在镜像中创建自定义文件和目录。您只能在 `/etc` 目录中使用这些自定义。



### 注意

这些蓝图自定义被所有镜像类型支持，但部署 OSTree 提交的镜像类型除外，如 `edge-raw-image`、`edge-installer`、`edge-simplified-installer`。



### 警告

如果您将 `customizations.directories` 与设置了 `mode`、`user` 或 `group` 的镜像中已存在的目录路径一起使用，则镜像构建无法防止更改现有目录的所有权或权限。

#### 5.3.12.2. 在蓝图中指定自定义目录

使用 `[[customizations.directory]]` 蓝图自定义，您可以：

- 创建新目录。
- 为您要创建的目录设置用户和组所有权。
- 以八进制格式设置目录模式权限。
- 确保根据需要创建父目录。

使用 `[[customizations.files]]` 蓝图自定义，您可以：

- 创建新文本文件。
- 修改现有文件。警告：这可能会覆盖现有内容。
- 为您要创建的文件设置用户和组所有权。
- 以八进制格式设置模式权限。



### 注意

您无法创建或替换以下文件：

- `/etc/fstab`
- `/etc/shadow`
- `/etc/passwd`

- **/etc/group**

以下自定义可用：

- 在蓝图中自定义文件系统配置：

```
[[customizations.filesystem]]
mountpoint = "MOUNTPOINT"
minsize = MINIMUM-PARTITION-SIZE
```

**MINIMUM-PARTITION-SIZE** 值没有默认大小格式。蓝图自定义支持以下值和单位：kB 到 TB 以及 KiB 到 TiB。例如，您可以以字节为单位定义挂载点大小：

```
[[customizations.filesystem]]
mountpoint = "/var"
minsize = 1073741824
```

- 使用单位定义挂载点大小。例如：

```
[[customizations.filesystem]]
mountpoint = "/opt"
minsize = "20 GiB"
```

```
[[customizations.filesystem]]
mountpoint = "/boot"
minsize = "1 GiB"
```

- 通过设置 **minsize** 定义最小分区。例如：

```
[[customizations.filesystem]]
mountpoint = "/var"
minsize = 2147483648
```

- 使用 **[[customizations.directories]]**，在 **/etc** 目录下为镜像创建自定义目录：

```
[[customizations.directories]]
path = "/etc/directory_name"
mode = "octal_access_permission"
user = "user_string_or_integer"
group = "group_string_or_integer"
ensure_parents = boolean
```

蓝图条目如下所述：

- **path** - 必需 - 输入您要创建的目录的路径。它必须是 **/etc** 目录下的绝对路径。
- **mode** - 可选 - 以八进制格式设置目录的访问权限。如果没有指定权限，则默认为 0755。前面的零是可选的。
- **user** - 可选 - 将用户设置为目录的所有者。如果没有指定用户，则默认为 **root**。您可以将用户指定为字符串或整数。
- **group** - 可选 - 将组设置为目录的所有者。如果没有指定组，则默认为 **root**。您可以将组指定为字符串或整数。

- **ensure\_parents** - 可选 - 指定是否要根据需要创建父目录。如果没有指定值，则默认为 **false**。
- 使用 `[[customizations.directories]]`，在 `/etc` 目录下为镜像创建自定义文件：

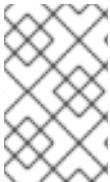
```
[[customizations.files]]
path = "/etc/directory_name"
mode = "octal_access_permission"
user = "user_string_or_integer"
group = "group_string_or_integer"
data = "Hello world!"
```

蓝图条目如下所述：

- **path** - 必需 - 输入您要创建的文件的路径。它必须是 `/etc` 目录下的绝对路径。
- **mode** 可选 - 以八进制格式设置对文件的访问权限。如果没有指定权限，则默认为 `0644`。前面的零是可选的。
- **user** - 可选 - 将用户设置为文件所有者。如果没有指定用户，则默认为 **root**。您可以将用户指定为字符串或整数。
- **group** - 可选 - 将组设置为文件所有者。如果没有指定组，则默认为 **root**。您可以将组指定为字符串或整数。
- **data** - 可选 - 指定纯文本文件的内容。如果没有指定内容，它会创建一个空文件。

## 5.4. RHEL 镜像构建器安装的软件包

当使用 RHEL 镜像构建器创建系统镜像时，系统会安装一组基本软件包组。



### 注意

当您在蓝图中添加其他组件时，请确保添加的组件中的软件包不会与任何其他软件包组件冲突。否则，系统无法解决依赖项并创建自定义镜像失败。您可以通过运行以下命令检查软件包之间没有冲突：

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

表 5.1. 支持创建镜像类型的默认软件包

镜像类型	默认软件包
ami	checkpolicy, chrony, cloud-init, cloud-utils-growpart, @Core, dhcp-client, gdisk, insights-client, kernel, langpacks-en, net-tools, NetworkManager, redhat-release, redhat-release-eula, rng-tools, rsync, selinux-policy-targeted, tar, yum-utils
openstack	@core, langpacks-en

镜像类型	默认软件包
qcow2	@core, chrony, dnf, kernel, dnf, nfs-utils, dnf-utils, cloud-init, python3-jsonschema, qemu-guest-agent, cloud-utils-growpart, dracut-norescue, tar, tcpdump, rsync, dnf-plugin-spacewalk, rhn-client-tools, rhnlib, rhnsd, rhn-setup, NetworkManager, dhcp-client, cockpit-ws, cockpit-system, subscription-manager-cockpit, redhat-release, redhat-release-eula, rng-tools, insights-client
tar	policycoreutils, selinux-policy-targeted
vhd	@core, langpacks-en
vmdk	@core, chrony, cloud-init, firewalld, langpacks-en, open-vm-tools, selinux-policy-targeted
edge-commit	attr, audit, basesystem, bash, bash-completion, chrony, clevis, clevis-dracut, clevis-luks, container-selinux, coreutils, criu, cryptsetup, curl, dnsmasq, dosfstools, dracut-config-generic, dracut-network, e2fsprogs, firewalld, fuse-overlayfs, fwupd, glibc, glibc-minimal-langpack, gnupg2, greenboot, gzip, hostname, ima-evm-utils, iproute, iptables, iputils, keyutils, less, lvm2, NetworkManager, NetworkManager-wifi, NetworkManager-wwan, nss-altfiles, openssh-clients, openssh-server, passwd, pinentry, platform-python, podman, policycoreutils, policycoreutils-python-utils, polkit, procps-ng, redhat-release, rootfiles, rpm, rpm-ostree, rsync, selinux-policy-targeted, setools-console, setup, shadow-utils, shadow-utils, skopeo, slirp4netns, sudo, systemd, tar, tmux, traceroute, usbguard, util-linux, vim-minimal, wpa_supplicant, xz
edge-container	dnf, dosfstools, e2fsprogs, glibc, lorax-templates-generic, lorax-templates-rhel, lvm2, policycoreutils, python36, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz

镜像类型	默认软件包
edge-installer	aajohan-comfortaa-fonts, abattis-cantarell-fonts, alsa-firmware, alsa-tools-firmware, anaconda, anaconda-install-env-deps, anaconda-widgets, audit, bind-utils, bitmapfangsongti-fonts, bzip2, cryptsetup, dbus-x11, dejavu-sans-fonts, dejavu-sans-mono-fonts, device-mapper-persistent-data, dnf, dump, ethtool, fcoe-utils, ftp, gdb-gdbserver, gdisk, gfs2-utils, glibc-all-langpacks, google-noto-sans-cjk-ttc-fonts, gsettings-desktop-schemas, hdparm, hexedit, initscripts, ipmitool, iwl3945-firmware, iwl4965-firmware, iwl6000g2a-firmware, iwl6000g2b-firmware, jomolhari-fonts, kacst-farsi-fonts, kacst-qurn-fonts, kbd, kbd-misc, kdump-anaconda-addon, khmeros-base-fonts, libblockdev-lvm-dbus, libertas-sd8686-firmware, libertas-sd8787-firmware, libertas-usb8388-firmware, libertas-usb8388-olpc-firmware, libibverbs, libreport-plugin-bugzilla, libreport-plugin-reportuploader, libreport-rhel-anaconda-bugzilla, librsvg2, linux-firmware, lklug-fonts, lldpad, lohit-assamese-fonts, lohit-bengali-fonts, lohit-devanagari-fonts, lohit-gujarati-fonts, lohit-gurmukhi-fonts, lohit-kannada-fonts, lohit-odia-fonts, lohit-tamil-fonts, lohit-telugu-fonts, lsof, madan-fonts, metacity, mtr, mt-st, net-tools, nmap-ncat, nm-connection-editor, nss-tools, openssh-server, oscap-anaconda-addon, pciutils, perl-interpreter, pigz, python3-pyatspi, rdma-core, redhat-release-eula, rpm-ostree, rsync, rsyslog, sg3_utils, sil-abyssinica-fonts, sil-padauk-fonts, sil-scheherazade-fonts, smartmontools, smc-meera-fonts, spicevdagent, strace, system-storage-manager, thai-scalable-waree-fonts, tigervnc-server-minimal, tigervnc-server-module, udisks2, udisks2-iscsi, usbutils, vim-minimal, volume_key, wget, xfsdump, xorg-x11-drivers,xorg-x11-fonts-misc,xorg-x11-server-utils,xorg-x11-server-Xorg, xorg-x11-xauth



镜像类型	默认软件包
edge-simplified-installer	attr, basesystem, binutils, bsdtar, clevis-dracut, clevis-luks, cloud-utils-growpart, coreos-installer, coreos-installer-dracut, coreutils, device-mapper-multipath, dnsmasq, dosfstools, dracut-live, e2fsprogs, fcoe-utils, fdo-init, gzip, ima-evm-utils, iproute, iptables, iputils, iscsi-initiator-utils, keyutils, lldpad, lvm2, passwd, policycoreutils, policycoreutils-python-utils, procps-ng, rootfiles, setools-console, sudo, traceroute, util-linux
image-installer	anaconda-dracut, curl, dracut-config-generic, dracut-network, hostname, iwl100-firmware, iwl1000-firmware, iwl105-firmware, iwl135-firmware, iwl2000-firmware, iwl2030-firmware, iwl3160-firmware, iwl5000-firmware, iwl5150-firmware, iwl6000-firmware, iwl6050-firmware, iwl7260-firmware, kernel, less, nfs-utils, openssh-clients, ostree, plymouth, prefixdevname, rng-tools, rpcbind, selinux-policy-targeted, systemd, tar, xfsprogs, xz
edge-raw-image	dnf, dosfstools, e2fsprogs, glibc, lorax-templates-generic, lorax-templates-rhel, lvm2, policycoreutils, python36, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz
gce	@core, langpacks-en, acpid, dhcp-client, dnf-automatic, net-tools, python3, rng-tools, tar, vim

## 其他资源

- [RHEL 镜像构建器描述](#)

## 第 6 章 构建简化的安装程序镜像以置备 RHEL FOR EDGE 镜像

您可以构建 RHEL for Edge Simplified Installer 镜像，该镜像针对以无人值守的形式在设备中安装进行了优化，并将镜像置备到 RHEL for Edge 镜像。

### 6.1. 简化的安装程序镜像构建和部署

使用 **edge-simplified-installer** 镜像类型构建 RHEL for Edge Simplified Installer 镜像。

要构建 RHEL for Edge Simplified Installer 镜像，请提供现有的 **OSTree** 提交。生成的简化镜像包含部署了 OSTree 提交的原始镜像。引导简化的安装程序 ISO 镜像后，它提供边缘系统的 RHEL，您可以在硬盘上使用它，也可以在虚拟机中用作引导镜像。您可以使用您在用于创建简化的安装程序镜像的蓝图中指定的用户名和密码登录到部署的系统。

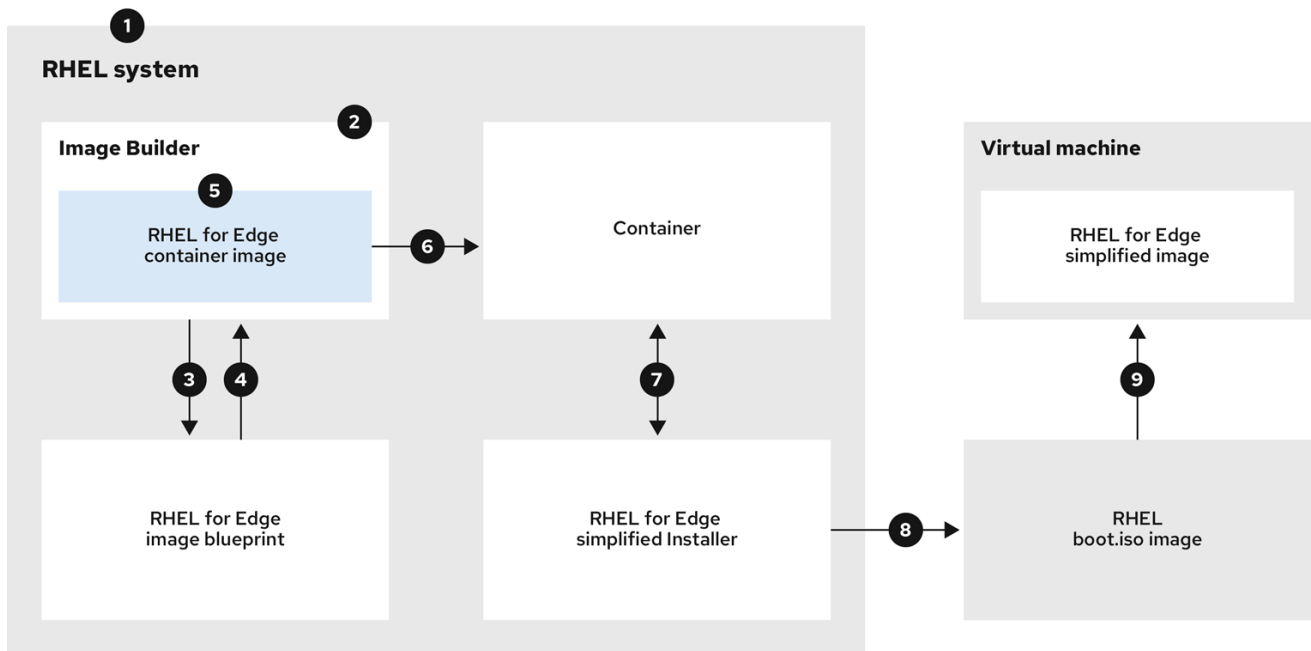
RHEL for Edge Simplified Installer 镜像对设备的无人值守安装进行了优化，并支持基于网络的部署和非基于网络的部署。但是，对于基于网络的部署，它只支持 UEFI HTTP 引导。

制作并部署简化的 RHEL for Edge 镜像涉及以下高级别的步骤：

1. 安装并注册 RHEL 系统
2. 安装 RHEL 镜像构建器
3. 使用 RHEL 镜像构建器，为 RHEL for Edge 容器镜像创建带有自定义的蓝图
4. 在 RHEL 镜像构建器中导入 RHEL for Edge 蓝图
5. 创建嵌入在 OCI 容器中的 RHEL for Edge 镜像，其中已准备好 web 服务器将提交部署为 OSTree 存储库
6. 为 **edge-simplified-installer** 镜像创建一个蓝图
7. 构建一个简化的 RHEL for Edge 镜像
8. 下载 RHEL for Edge 简化镜像
9. 使用 **edge-simplified-installer** `virt-install` 安装原始镜像

下图显示了 RHEL for Edge Simplified 的构建与置备工作流：

图 6.1. 在基于网络的环境中构建并置备 RHEL for Edge



243\_RHEL\_0422

## 6.2. 使用 RHEL 镜像构建器 CLI 为简化的镜像创建蓝图

要为简化的 RHEL for Edge 镜像创建蓝图，您必须使用 **设备文件** 位置进行自定义，以启用对设备的无人值守安装，以及一个执行初始设备凭证交换的 **URL**。您还必须在蓝图中指定用户和用户组。为此，请按照这些步骤：

### 流程

1. 以 Tom's Obvious, Minimal Language (TOML) 格式创建一个纯文本文件，其内容如下：

```
name = "simplified-installer-blueprint"
description = "blueprint for the simplified installer image"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[customizations]
installation_device = "/dev/vda"

[[customizations.user]]
name = "admin"
password = "admin"
groups = ["users", "wheel"]

[customizations.fdo]
manufacturing_server_url = "http://10.0.0.2:8080"
diun_pub_key_insecure = "true"
```



## 注意

蓝图中的 FDO 自定义是可选的，您可以构建 RHEL for Edge Simplified Installer 镜像，且无错误。

- *name* 是蓝图的名称，*description* 是蓝图的描述信息。
  - 0.0.1 是 Semantic Versioning 方案的版本号。
  - *模块* 描述了要安装到镜像中的软件包名称和匹配版本的 glob，例如：软件包名称 = "tmux"，匹配的版本 glob 是 version = "2.9a"。请注意，目前软件包和模块之间没有区别。
  - *组* 是要安装到镜像中的软件包组，如 **anaconda-tools** 组软件包。如果您不知道模块和组，请将其留空。
  - *Installation-device* 是自定义的，可对您的设备进行无人值守安装。
  - *manufacturing\_server\_url* 是执行初始设备凭证交换的 URL。
  - *name* 是要登录到镜像的用户名。
  - *password* 是您选择的密码。
  - *groups* 是任何用户组，如 "widget"。
2. 将蓝图推送（导入）到 RHEL 镜像构建器服务器：

```
# composer-cli blueprints push blueprint-name.toml
```

3. 列出现有的蓝图，以检查创建的蓝图是否已成功推送并存在。

```
# composer-cli blueprints show blueprint-name
```

4. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve blueprint-name
```

## 其他资源

- [使用 RHEL 镜像构建器命令行制作 RHEL for Edge 镜像](#)。

## 6.3. 使用镜像构建器 CLI 创建 RHEL FOR EDGE SIMPLIFIED INSTALLER 镜像

要使用 RHEL 镜像构建器命令行界面创建 RHEL for Edge Simplified 镜像，请确保您满足以下先决条件，然后按照以下流程操作。

### 先决条件

- 为 RHEL for Edge Simplified 的镜像创建了蓝图。
- 您提供提交的 OSTree 存储库来嵌入镜像中。例如：<http://10.0.2.2:8080/repo>。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。

## 流程

1. 创建可引导 ISO 镜像。

```
# composer-cli compose start-ostree \
  blueprint-name \
  edge-simplified-installer \
  --ref rhel/9/x86_64/edge \
  --url URL-OSTree-repository \
```

其中,

- **blueprint-name** 是 RHEL for Edge 蓝图名称。
- **edge-simplified-installer** 是镜像类型。
- **--ref** 指定创建您的提交的位置的引用。
- **--url** 是要嵌入到镜像中的提交的 OSTree 存储库的 URL。例如：  
http://10.0.2.2:8080/repo/。您可以启动一个 RHEL for Edge Container，或设置 web 服务器。请参阅[为非基于网络的部署创建 RHEL for Edge 容器镜像](#)，以及[设置 web 服务器来安装 RHEL for Edge 镜像](#)。  
这时将显示一个确认已添加到队列中的 composer 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。

2. 检查镜像 compose 状态。

```
# composer-cli compose status
```

输出以以下格式显示状态：

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



### 注意

镜像创建过程可能需要最多十分钟才能完成。

要中断镜像创建过程，请运行：

```
# composer-cli compose cancel <UUID>
```

要删除现有镜像，请运行：

```
# composer-cli compose delete <UUID>
```

## 其他资源

- [使用 RHEL 镜像构建器命令行制作 RHEL for Edge 镜像](#)

## 6.4. 使用镜像构建器命令行界面下载简化的 RHEL FOR EDGE 镜像

要使用 RHEL 镜像构建器命令行界面下载 RHEL for Edge 镜像，请确保您满足以下先决条件，然后按照以下流程操作。

## 先决条件

- 您已创建了 RHEL for Edge 镜像。

## 流程

1. 查看 RHEL for Edge 镜像状态。

```
# composer-cli compose status
```

输出必须显示以下内容：

```
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
```

2. 下载镜像。

```
# composer-cli compose image <UUID>
```

RHEL 镜像构建器将镜像作为 **.iso** 文件下载到您运行命令的当前目录路径。

UUID 号和镜像大小会同时显示。

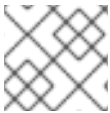
```
$ <UUID>-simplified-installer.iso: size MB
```

因此，您下载了一个 RHEL for Edge Simplified Installer ISO 镜像。您可以直接使用它作为引导 ISO 来安装 RHEL for Edge 系统。

## 6.5. 使用镜像构建器 GUI 为简化的镜像 RHEL 创建蓝图

要创建 RHEL for Edge Simplified Installer 镜像，您必须创建一个蓝图并确保使用以下内容自定义它：

- 设备节点位置，对您的设备进行无人值守安装。
- 执行初始设备凭证交换的 URL。
- 用户或用户组。



### 注意

您还可以添加镜像所需的任何其他自定义。

要在 RHEL 镜像构建器 GUI 中为简化的 RHEL for Edge 镜像创建蓝图，请完成以下步骤：

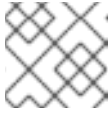
## 先决条件

- 您已在浏览器中从 Web 控制台打开了镜像构建器应用。请参阅 [在 RHEL web 控制台中访问 RHEL 镜像构建器 GUI](#)。

## 流程

1. 点 RHEL 镜像构建器应用程序右上角中的 Create Blueprint。此时会打开一个对话框向导，其中包含蓝图名称和描述字段。

2. 在 **Details** 页面上 :
  - a. 输入蓝图名称, 以及可选的描述。点 **Next**。
3. 可选 : 在 **Packages** 页面中, 完成以下步骤 :
  - a. 在 **Available packages** 搜索中, 输入软件包名称并点击 **>** 按钮将其移到所选的软件包字段。搜索并包含尽可能多的软件包。点 **Next**。



### 注意

除非另有指定, 否则自定义都是可选的。

4. 可选 : 在 **Kernel** 页面中, 输入内核名称和命令行参数。
5. 可选 : 在 **File system** 页面中, 选择 **Use automatic partitioning**。OSTree 系统不支持文件系统自定义, 因为 OSTree 镜像有自己的挂载规则, 如只读。点 **Next**。
6. 可选 : 在 **Services** 页面上, 您可以启用或禁用服务 :
  - a. 输入您要启用或禁用的服务名称, 用逗号、空格分隔它们或按 **Enter** 键。点 **Next**。
7. 可选 : 在 **Firewall** 页面中, 设置防火墙设置 :
  - a. 输入 **端口**, 以及您要启用或禁用的防火墙服务。
  - b. 点 **Add zone** 按钮, 为每个区域单独管理您的防火墙规则。点 **Next**。
8. 在 **Users** 页面中, 按照以下步骤添加用户 :
  - a. 单击 **Add user**。
  - b. 输入 **Username**、**password**, 以及 **SSH key**。您还可以单击 **Server administrator** 复选框, 将用户标记为特权用户。



### 注意

当您在蓝图自定义中指定用户, 然后从该蓝图创建镜像时, 蓝图会在安装过程中在 **/usr/lib/passwd** 目录下创建用户, 在 **/usr/etc/shadow** 下创建密码。您可以使用您为蓝图创建的用户名和密码登录到设备。访问系统后, 您必须创建用户, 例如, 使用 **useradd** 命令。

点 **Next**。

9. 可选 : 在 **Groups** 页面中, 完成以下步骤来添加组 :
  - a. 点 **Add groups** 按钮 :
    - i. 输入 **Group name** 和 **Group ID**。您可以添加更多组。点 **Next**。
10. 可选 : 在 **SSH keys** 页面中, 添加一个密钥 :
  - a. 点 **Add key** 按钮。
    - i. 输入 SSH 密钥。
    - ii. 输入 **User**。点 **Next**。

11. 可选：在 **Timezone** 页面中，设置您的时区设置：
  - a. 在 **Timezone** 字段中输入您要添加到系统镜像的时区。例如，添加以下时区格式："US/Eastern"。  
如果您没有设置时区，系统默认使用 Universal Time, Coordinated (UTC)作为默认值。
  - b. 输入 **NTP** 服务器。点 **Next**。
12. 可选：在 **Locale** 页面中完成以下步骤：
  - a. 在 **Keyboard** 搜索字段中，输入您要添加到系统镜像的软件包名称。例如：["en\_US.UTF-8"]。
  - b. 在 **Languages** 搜索字段中，输入您要添加到系统镜像的软件包名称。例如："us"。点 **Next**。
13. 必选：在 **Others** 页面中，完成以下步骤：
  - a. 在 **Hostname** 字段中输入您要添加到系统镜像的主机名。如果没有添加主机名，操作系统会决定主机名。
  - b. 必选：在 **Installation Devices** 字段中输入您的系统镜像的有效节点，以为您的设备启用无人值守安装。例如：**dev/sda1**。点 **Next**。
14. 可选：在 **FIDO device onboarding** 页面中完成以下步骤：
  - a. 在 **Manufacturing server URL** 字段中，输入 **制造服务器 URL**，以执行初始设备凭据交换，例如："http://10.0.0.2:8080"。蓝图中的 FDO 自定义是可选的，您可以构建 RHEL for Edge Simplified Installer 镜像，且无错误。
  - b. 在 **DIUN public key insecure** 字段中，输入认证公钥哈希来执行初始设备凭证交换。此字段接受 "true" 值，这意味着这是与制造服务器的不安全连接。例如：**manufacturing\_server\_url="http://\${FDO\_SERVER}:8080" diun\_pub\_key\_insecure="true"**。您必须只使用以下三个选项之一："key insecure"、"key hash"和"key root certs"。
  - c. 在 **DIUN public key hash** 字段中，输入您的公钥的哈希版本。例如：**17BD05952222C421D6F1BB1256E0C925310CED4CE1C4FFD6E5CB968F4B73BF73**。您可以根据制造服务器的证书生成密钥来获取密钥哈希。要生成密钥哈希，请运行以下命令：
 

```
# openssl x509 -fingerprint -sha256 -noout -in /etc/fdo/aio/keys/diun_cert.pem | cut -d"=" -f2 | sed 's://g'
```

**/etc/fdo/aio/keys/diun\_cert.pem** 是存储在制造服务器中的证书。
  - d. 在 **DIUN public key root certs** 字段中，输入公钥根证书。此字段接受存储在制造服务器中的认证文件的内容。要获取证书文件的内容，请运行以下命令：
 

```
$ cat /etc/fdo/aio/keys/diun_cert.pem.
```
15. 点 **Next**。
16. 在 **Review** 页面中，查看蓝图的详情。点 **Create**。

RHEL 镜像构建器视图打开，列出了现有蓝图。



## 6.6. 使用镜像构建器 GUI 创建 RHEL FOR EDGE SIMPLIFIED INSTALLER 镜像

要使用 RHEL 镜像构建器 GUI 创建 RHEL for Edge Simplified 镜像，请确保您满足以下先决条件，然后按照以下流程操作。

### 先决条件

- 在浏览器中从 web 控制台打开 RHEL 镜像构建器应用程序。
- 为 RHEL for Edge Simplified 的镜像创建了蓝图。
- 您提供提交的 OSTree 存储库来嵌入到镜像中，例如 <http://10.0.2.2:8080/repo>。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。
- FDO 制造服务器已启动并运行。

### 流程

1. 访问 mage 构建器仪表盘。
2. 在蓝图表中，找到您要为其构建镜像的蓝图。
3. 导航到 **Images** 选项卡，再点 **Create Image**。Create image 向导将打开。
4. 在 **Image 输出** 页面中完成以下步骤：
  - a. 从 **Select a blueprint** 列表中，选择您为 RHEL for Edge Simplified 镜像创建的蓝图。
  - b. 从 **Image output type** 列表中，选择 **RHEL for Edge Simplified Installer (.iso)**。
  - c. 在 **Image Size** 字段中输入镜像大小。Simplified Installer 镜像所需的最小镜像大小为：
5. 点 **Next**。
6. 在 **OSTree settings** 页面中，完成以下步骤：
  - a. 在 **Repository URL** 字段中，输入从中拉取父 OSTree 提交的存储库 URL。
  - b. 在 **Ref** 字段中，输入 **ref** 分支名称路径。如果没有输入 **ref**，则使用 distro 的默认 **ref**。
7. 在 **Review** 页面中，查看镜像自定义并点 **Create**。

镜像构建启动，需要 20 分钟完成。要停止构建，请点击 **Stop build**。

## 6.7. 使用镜像构建器 GUI 下载简化的 RHEL FOR EDGE 镜像

要使用 RHEL 镜像构建器 GUI 下载 RHEL for Edge 镜像，请确保您满足以下先决条件，然后按照以下流程操作。

### 先决条件

- 您已成功创建了一个 RHEL for Edge 镜像。请参阅[链接](#)。

### 流程

1. 访问 RHEL 镜像构建器仪表盘。蓝图列表仪表盘将打开。
2. 在蓝图中，找到您为 RHEL for Edge Simplified Installer 镜像构建的蓝图。
3. 导航到 **Images** 选项卡。
4. 选择其中一个选项：
  - 下载镜像。
  - 下载镜像的日志以检查元素，并验证是否发现了任何问题。



### 注意

您可以使用您直接作为引导 ISO 下载的 RHEL for Edge Simplified Installer ISO 镜像来安装 RHEL for Edge 系统。

## 6.8. 设置 UEFI HTTP 引导服务器

要建立 **UEFI HTTP Boot** 服务器，以便您可以通过连接到这个 UEFI HTTP Boot 服务器，通过网络开始提供 RHEL for Edge 虚拟机，请按照以下步骤执行：

### 先决条件

- 您已创建了 ISO 简化的安装程序镜像。
- 提供 ISO 内容的 http 服务器。

### 流程

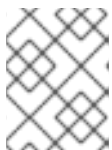
1. 将 ISO 镜像挂载到您选择的目录：

```
# mkdir /mnt/rhel9-install/
# mount -o loop,ro -t iso9660 /path_directory/installer.iso /mnt/rhel9-install/
```

使用 RHEL for Edge 可引导 ISO 镜像的路径替换 **/path\_directory/installer.iso**。

2. 将挂载镜像中的文件复制到 HTTP 服务器 root 中。这个命令创建包含镜像内容的 **/var/www/html/rhel9-install/** 目录。

```
# mkdir /var/www/html/httpboot/
# cp -R /mnt/rhel9-install/* /var/www/html/httpboot/
# chmod -R +r /var/www/html/httpboot/*
```



### 注意

某些复制方法可以跳过 **.treeinfo** 文件（一个有效的安装源需要这个文件）。对于整个目录运行 **cp** 命令，如此过程所示，可正确复制 **.treeinfo**。

3. 更新 **/var/www/html/EFI/BOOT/grub.cfg** 文件：
  - a. **coreos.inst.install\_dev=/dev/sda** with **coreos.inst.install\_dev=/dev/vda**
  - b. 使用 **linuxefi /images/pxeboot/vmlinuz** 替换 **linux /images/pxeboot/vmlinuz**

- c. 使用 `initrdefi /images/pxeboot/initrd.img` 替换 `initrd /images/pxeboot/initrd.img`
- d. 使用 `coreos.inst.image_url=http://{IP-ADDRESS}/disk.img.xz` 替换 `coreos.inst.image_file=/run/media/iso/disk.img.xz`。  
`IP-ADDRESS` 是该计算机的 ip 地址，它将充当 http 引导服务器。

4. 启动 httpd 服务：

```
# systemctl start httpd.service
```

因此，在设置 **UEFI HTTP** 引导服务器后，您可以使用 **UEFI HTTP** 引导安装 RHEL for Edge 设备。

## 6.9. 在虚拟机中部署简化的 ISO 镜像

使用以下安装源创建 RHEL for Edge Simplified 镜像，以此部署您生成的 RHEL for Edge ISO 镜像：

- **UEFI HTTP** 引导
- `virt-install`

本例演示了如何从 ISO 镜像为**基于网络的安装**创建 `virt-install` 安装源。

### 先决条件

- 您已创建了 ISO 镜像。
- 您可以设置网络配置来支持 UEFI HTTP 引导。

### 流程

1. 设置网络配置以支持 **UEFI HTTP** 引导。请参阅[使用 libvirt 设置 UEFI HTTP 引导](#)。
2. 使用 `virt-install` 命令从 UEFI HTTP 引导创建 RHEL for Edge 虚拟机。

```
# virt-install \
  --name edge-install-image \
  --disk path=" ",format=qcow2 \
  --ram 3072 \
  --memory 4096 \
  --vcpus 2 \
  --network network=integration,mac=mac_address \
  --os-type linux \
  --os-variant rhel9 \
  --cdrom "/var/lib/libvirt/images/"ISO_FILENAME" \
  --boot \
  uefi,loader_ro=yes,loader_type=pflash,nvram_template=/usr/share/edk2/ovmf/OVMF_VARS.fd, \
  loader_secure=no \
  --virt-type kvm \
  --graphics none \
  --wait=-1 \
  --noreboot
```

运行命令后，虚拟机安装将启动。

## 验证

- 登录到创建的虚拟机。

## 6.10. 从 USB 闪存驱动器部署简化的 ISO 镜像

使用 **USB 安装** 创建 RHEL for Edge Simplified 镜像，从而部署您生成的 RHEL for Edge ISO 镜像。

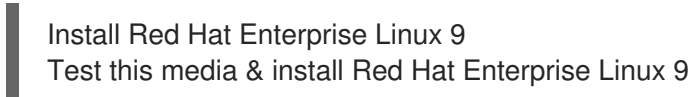
本例演示了如何从 ISO 镜像创建 **USB 安装源**。

### 先决条件

- 您已创建了简化的安装程序镜像，该镜像为 ISO 镜像。
- 您有一个 8 GB USB 闪存。

### 流程

1. 将 ISO 镜像文件复制到 USB 闪存驱动器中。
2. 将 USB 闪存连接到您要引导的计算机的端口。
3. 从 USB 闪存驱动器引导 ISO 镜像。引导菜单显示以下选项：



```

| Install Red Hat Enterprise Linux 9
| Test this media & install Red Hat Enterprise Linux 9

```

4. 选择安装 Red Hat Enterprise Linux 9。这将启动系统安装。

### 其他资源

- [引导安装](#)。

## 6.11. 在 FIPS 模式下创建并引导 RHEL FOR EDGE 镜像

您可以创建并引导启用了 FIPS 的 RHEL for Edge 镜像。您可以在蓝图中指定是否要在生成的镜像中启用 FIPS 模式。默认值为 **false**。

您可以在 FIPS 模式下构建以下镜像类型：

- **edge-installer**
- **edge-simplified-installer**
- **edge-raw-image**
- **edge-ami**
- **edge-vsphere**



## 重要

您只能在镜像置备过程中启用 FIPS 模式。在非 FIPS 镜像构建启动后，您无法切换到 FIPS 模式。如果构建启用了 FIPS 的镜像的主机没有启用 FIPS，则此主机生成的密钥不符合 FIPS，但生成的镜像按预期符合 FIPS。

## 先决条件

- 您创建并下载了一个 RHEL for Edge Container OSTree 提交。
- 已在您的系统上安装了 Podman。请参阅 [如何在 RHEL 中安装 Podman](#)。

## 流程

1. 使用以下内容，创建一个 Tom's Obvious, Minimal Language (TOML) 格式 的纯文本文件：

```
name = "system-fips-mode-enabled"
description = "blueprint with FIPS enabled "
version = "0.0.1"

[ostree]
ref= "example/edge"
url= "http://example.com/repo"

[customizations]
installation_device = "/dev/vda"
fips = true

[[customizations.user]]
name = "admin"
password = "admin"
groups = ["users", "wheel"]

[customizations.fdo]
manufacturing_server_url = "https://fdo.example.com"
diun_pub_key_insecure = true
```

2. 将蓝图导入到 RHEL 镜像构建器服务器中：

```
# composer-cli blueprints push blueprint-name.toml
```

3. 列出现有的蓝图，以检查创建的蓝图是否已成功导入且存在：

```
# composer-cli blueprints show blueprint-name
```

4. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve blueprint-name
```

5. 提供要嵌入到镜像中的提交的 OSTree 存储库，例如 <http://10.0.2.2:8080/repo>。如需更多信息，请参阅 [建立一个 UEFI HTTP 引导服务器](#)。
6. 创建可引导的 ISO 镜像：

```
# composer-cli compose start-ostree \
blueprint-name \
edge-simplified-installer \
--ref rhel/8/x86_64/edge \
--url URL-OSTree-repository \
```

7. 查看 RHEL for Edge 镜像状态：

```
# composer-cli compose status
...
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
...
```

8. 下载镜像：

```
# composer-cli compose image <UUID>
```

RHEL 镜像构建器将镜像作为 **.iso** 文件下载到您运行命令的当前目录路径。UUID 号和镜像大小一同显示：

```
$ <UUID>-simplified-installer.iso: size MB
```

9. 从 UEFI HTTP 引导创建一个 RHEL for Edge 虚拟机，例如：

```
# virt-install \
--name edge-device
--disk path="/var/lib/libvirt/images/edge-device.qcow2",size=5,format=qcow2 \
--memory 4096 \
--vcpus 2 \
--network network=default \
--os-type linux \
--os-variant rhel8.9 \
--cdrom /var/lib/libvirt/images/<UUID>-simplified-installer.iso \
--boot uefi,loader.secure=false \
--virt-type kvm \
--graphics none \
--wait=-1 \
--noreboot
```

运行命令后，虚拟机安装将启动。

## 验证

1. 使用您在蓝图中配置的用户名和密码登录到创建的虚拟机。
2. 检查是否启用了 FIPS 设置：

```
$ fips-mode-setup --check
...
FIPS mode is enabled
...
```

## 第 7 章 构建并提供一个最小原始镜像

**minimal-raw** 镜像是一个预打包的、可引导的、最小 RPM 镜像，以 **xz** 格式压缩。您可以使用 RHEL 镜像构建器构建一个 RHEL for Edge Minimal Raw 镜像类型，并将 Minimal Raw 镜像部署到 **aarch64** 和 **x86** 架构上。

### 7.1. 最小原始镜像构建和部署

使用 **minimal-raw** 镜像类型构建一个 RHEL for Edge Minimal Raw 镜像。要引导镜像，您必须解压缩它，并将其复制到任何可引导设备上，如 SD 卡。您可以使用您在蓝图中指定的用来创建 RHEL for Edge Minimal Raw 镜像的用户名和密码登录到部署的系统。

制作和部署 RHEL for Edge Minimal Raw 镜像涉及以下高级别步骤：

1. 安装并注册 RHEL 系统
2. 安装 RHEL 镜像构建器
3. 使用 RHEL 镜像构建器，为 RHEL for Edge Minimal Raw 镜像创建一个带有自定义的蓝图
4. 在 RHEL 镜像构建器中导入 RHEL for Edge 蓝图
5. 创建一个 RHEL for Edge Minimal Raw 镜像
6. 解压缩 RHEL for Edge Minimal Raw 镜像
7. 部署 RHEL for Edge Minimal Raw 镜像

### 7.2. 使用 RHEL 镜像构建器 CLI 为 MINIMAL RAW 镜像创建蓝图

通过使用 RHEL 镜像构建器，创建一个蓝图，并使用用户名和密码对其进行自定义。这样，您可以创建 Minimal Raw 镜像，并使用您在蓝图中自定义的凭证登录它。

#### 流程

1. 以 Tom's Obvious, Minimal Language (TOML) 格式创建一个纯文本文件，其内容如下：

```
name = "minimal-raw-blueprint"
description = "blueprint for the Minimal Raw image"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[[customizations.user]]
name = "admin"
password = "admin"
groups = ["users", "wheel"]
```

- name 是蓝图的名称，description 是蓝图的描述。
- 0.0.1 是遵循 Semantic Versioning 方案的版本号。

- 模块描述要安装到镜像中的软件包名称和匹配版本 glob，例如：软件包 name = "tmux"，匹配的版本 glob 是 version = "2.9a"。请注意，目前软件包和模块之间没有区别。
  - 组是要安装到镜像中的软件包组，如 anaconda-tools 组软件包。如果您不知道模块和组，请将其留空。
  - name 是登录到镜像的用户名。
  - password 是您选择的密码。
  - groups 是任何用户组，如 "widget"。
2. 将蓝图导入到 RHEL 镜像构建器服务器中：

```
# composer-cli blueprints push <blueprint_name>.toml
```

3. 检查系统上是否有蓝图：

```
# composer-cli blueprints list
```

4. 检查蓝图中组件、版本及其依赖项的有效性：

```
# composer-cli blueprints depsolve <blueprint_name>
```

#### 其他资源

- [使用 RHEL 镜像构建器命令行制作一个 RHEL for Edge 镜像](#)

## 7.3. 使用 RHEL 镜像构建器 CLI 创建一个 MINIMAL RAW 镜像

要使用 RHEL 镜像构建器命令行界面创建一个 RHEL for Edge Minimal Raw 镜像，请确保在按照流程操作前满足以下先决条件。

#### 先决条件

- 您为 RHEL for Edge Minimal Raw 镜像创建了一个蓝图。

#### 流程

1. 创建可引导镜像。

```
# composer-cli compose start <_blueprint_name_> minimal-raw
```

其中：

- **<blueprint\_name>** 是 RHEL for Edge 蓝图名称。
  - **minimal-raw-** 是镜像类型。  
这时将显示一个确认已添加到队列中的 composer 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。
2. 检查镜像 compose 状态。

```
# composer-cli compose status
```



输出以以下格式显示状态：

```
# <UUID> RUNNING date <blueprint_name> blueprint-version minimal-raw
```

### 其他资源

- [使用镜像构建器命令行制作 RHEL for Edge 镜像。](#)

## 7.4. 下载并解压缩 MINIMAL RAW 镜像

使用 RHEL 镜像构建器命令行界面下载 RHEL for Edge Minimal Raw 镜像，然后解压缩镜像以便能够引导它。

### 先决条件

- 您已创建了一个 RHEL for Edge Minimal Raw 镜像。

### 流程

1. 查看 RHEL for Edge Minimal Raw image compose 状态。

```
# composer-cli compose status
```

输出必须显示以下详情：

```
$ <UUID>_ FINISHED date <blueprint_name> <blueprint_version> minimal-raw
```

2. 下载镜像：

```
# composer-cli compose image <_UUID_>
```

镜像构建器将镜像作为 **.raw.xz** 下载到您的工作目录中。UUID 号和镜像大小会同时显示。

```
$ <UUID> minimal_raw.img.xz: size MB
```

3. 解压缩镜像：

```
$ xz -d <UUID>_minimal-raw.img.xz
```

解压缩镜像后，使用可引导的 RHEL for Edge Minimal Raw 镜像来安装一个 RHEL for Edge 系统。

## 7.5. 从 USB 闪存驱动器部署 MINIMAL RAW 镜像

通过创建一个 USB 安装源，来部署您生成的 RHEL for Edge Minimal Raw 镜像。本例演示了如何从 RHEL for Edge Minimal Raw 镜像创建一个 USB 安装源。

### 先决条件

- 您已创建了一个 RHEL for Edge Minimal Raw 镜像。
- 您有一个 8 GB USB 闪存。

- 您有一个运行的 UEFI HTTP 服务器。请参阅 [建立 UEFI HTTP 引导服务器](#)。

## 流程

1. 将 RHEL for Edge Minimal Raw 镜像文件复制到 USB 闪存驱动器中。
2. 将 USB 闪存连接到您要引导的计算机的端口。
3. 从 USB 闪存驱动器引导 RHEL for Edge Minimal Raw 镜像。引导菜单显示以下选项：

```
Install Red Hat Enterprise Linux 9
Test this media & install Red Hat Enterprise Linux 9
```

4. 选择 **Install Red Hat Enterprise Linux 9**。这将启动系统安装。

## 验证

1. 使用您在蓝图中配置的用户名和密码引导到镜像。

- a. 检查发行版本：

```
$ cat /etc/os-release
```

- b. 列出系统中的块设备：

```
$ lsblk
```

## 第 8 章 使用 RHEL FOR EDGE SIMPLIFIED INSTALLER 镜像的 IGNITION 工具

RHEL for Edge 使用 Ignition 工具在引导过程的早期将用户配置注入到镜像中。Ignition 工具注入的用户配置包括：

- 用户配置。
- 写文件，如常规文件和 **systemd** 单元。

第一次引导时，Ignition 从远程 URL 或嵌入在简化的安装程序 ISO 中的文件读取其配置。然后，Ignition 将该配置应用到镜像中。

### 8.1. 创建一个 IGNITION 配置文件

**Butane** 工具是创建 Ignition 配置文件的首选选项。**Butane** 使用一个 **Butane Config YAML** 文件，并以 JSON 格式生成一个 **Ignition Config**。在第一次启动时系统使用 JSON 文件。**Ignition Config** 应用镜像中的配置，如创建用户和 **systemd** 单元安装。

#### 先决条件

- 您已安装了 Butane 工具版本 v0.17.0：

```
$ sudo dnf/yum install -y butane
```

#### 流程

1. 创建一个 **Butane Config** 文件，并将它保存为 **.bu** 格式。对于 RHEL for Edge 镜像，您必须将 **variant** 条目指定为 **r4e**，将 **version** 条目指定为 **1.0.0**。版本 1.0.0 上的 butane **r4e** 变体以 Ignition spec 版本 **3.3.0** 为目标。以下是 Butane Config YAML 文件的一个示例：

```
variant: r4e
version: 1.0.0
ignition:
  config:
    merge:
      - source: http://192.168.122.1:8000/sample.ign
passwd:
  users:
    - name: core
      groups:
        - wheel
      password_hash: password_hash_here
      ssh_authorized_keys:
        - ssh-ed25519 some-ssh-key-here
storage:
  files:
    - path: /etc/NetworkManager/system-connections/enp1s0.nmconnection
      contents:
        inline: |
          [connection]
          id=enp1s0
          type=ethernet
          interface-name=enp1s0
```

```

    [ipv4]
    address1=192.168.122.42/24,192.168.122.1
    dns=8.8.8.8;
    dns-search=
    may-fail=false
    method=manual
    mode: 0600
  - path: /usr/local/bin/startup.sh
    contents:
      inline: |
        #!/bin/bash
        echo "Hello, World!"
      mode: 0755
systemd:
  units:
  - name: hello.service
    contents: |
      [Unit]
      Description=A hello world
      [Install]
      WantedBy=multi-user.target
    enabled: true
  - name: fdo-client-linuxapp.service
    dropins:
    - name: log_trace.conf
      contents: |
        [Service]
        Environment=LOG_LEVEL=trace

```

- 运行以下命令以使用 **Butane Config YAML** 文件，并生成一个 JSON 格式的 Ignition 配置：

```

$ ./path/butane example.bu
{"ignition":{"config":{"merge":[{"source":"http://192.168.122.1:8000/sample.ign"}]},"timeouts":{"httpTotal":30,"version":"3.3.0"},"passwd":{"users":[{"groups":["wheel"],"name":"core","passwordHash":"password_hash_here","sshAuthorizedKeys":["ssh-ed25519 some-ssh-key-here"]}],},"storage":{"files":[{"path":"/etc/NetworkManager/system-connections/enp1s0.nmconnection","contents":{"compression":"gzip","source":"data:;base64,H4sIAAAAAAAC/0yKUcrCMBAG3/csf/ObUKQie5LShyX5SgPNNiSr0NuLgiDzNMPM8VBFtHzoQjkxtPp+ITsrGLahKYyyGtoqEYNKwfeZc32OC0lKDb179rfg/HVyPgQ3hv8w/v0WT0k7T+7D/S1Dh7S4MRU5h1XyzqvsHVRg25G4iD5kp1cAAAD//6Cvq2ihAAAA"},"mode":384},{"path":"/usr/local/bin/startup.sh","contents":{"source":"data:;base64,IyEvYmluL2Jhc2gkZWNObyAiSGVsbG8sIFdvcmxkISIK"},"mode":493]}},{"systemd":{"units":[{"contents":"[Unit]\nDescription=A hello world\n[Install]\nWantedBy=multi-user.target","enabled":true,"name":"hello.service"},{"dropins":{"contents":"[Service]\nEnvironment=LOG_LEVEL=trace\n","name":"log_trace.conf"}],"name":"fdo-client-linuxapp.service"}}}}

```

运行 **Butane Config YAML** 文件以检查并生成 **Ignition Config JSON** 文件后，在使用不支持的字段时（如分区）时可能会收到警告。您可以修复这些字段并重新运行检查。

现在，您已有一个 Ignition JSON 配置文件，可用于自定义蓝图。

## 其他资源

- [RHEL for Edge 规格 v1.0.0](#) .

## 8.2. 在 GUI 中创建支持 IGNITION 的蓝图

在构建简化的安装程序镜像时，您可以通过在蓝图的 Ignition 页面中输入以下详情来自定义您的蓝图：

- **Firstboot URL** - 您必须输入一个指向第一次引导过程中获取的 Ignition 配置的 URL。它可用于原始镜像和简化的安装程序镜像。
- **Embedded Data** - 您必须提供 **base64** 编码的 **Ignition Configuration** 文件。它只能用于简化的安装程序镜像。

要为简化的 RHEL for Edge 镜像自定义蓝图，并支持使用 Ignition 蓝图自定义的 Ignition 配置，请按照以下步骤执行：

### 先决条件

- 您已在浏览器中从 Web 控制台打开了镜像构建器应用。请参阅 [在 RHEL web 控制台中访问镜像构建器 GUI](#)。
- 要完全支持嵌入的部分，**coreos-installer-dracut** 必须能够根据 OSBuild 的文件的存在来定义 - **ignition-url|ignition-file**。

### 流程

#### 1. 点击右上角的 **Create Blueprint**。

此时会打开一个对话框向导，其中包含蓝图名称和描述字段。

- 在 **Details** 页面上：
  - 输入蓝图的名称，以及可选的描述。点 **Next**。
- 在 **Ignition** 页面中，完成以下步骤：
  - 在 **Firstboot URL** 字段中，输入指向要在第一次引导期间获取的 Ignition 配置的 URL。
  - 在 **Embedded Data** 字段中，拖放或上传 **base64** 编码的 **Ignition Configuration** 文件。点 **Next**。
- 检查镜像详情并点 **Create**。

镜像构建器仪表盘视图打开，列出现有的蓝图。

### 下一步

- 您可以使用您创建的蓝图来构建简化的安装程序镜像。请参阅 [使用镜像构建器创建 RHEL for Edge Simplified Installer 镜像](#)。

## 8.3. 使用 CLI 创建支持 IGNITION 的蓝图

在构建简化的安装程序镜像时，您可以通过向其添加 **customizations.ignition** 部分来自定义您的蓝图。因此，您可以创建简化的安装程序镜像或可用于裸机平台的原始镜像。蓝图中的 **customizations.ignition** 自定义可让配置文件用于 **edge-simplified-installer** ISO 和 **edge-raw-image** 镜像。

- 对于 **edge-simplified-installer** ISO 镜像，您可以自定义蓝图来嵌入将在 ISO 镜像中包含的 Ignition 配置文件。例如：

```
[customizations.ignition.embedded]
config = "eyJ --- BASE64 STRING TRIMMED --- 19fQo="
```

您必须提供 **base64** 编码的 Ignition 配置文件。

- 对于 **edge-simplified-installer** ISO 镜像和 **edge-raw-image**，您可以通过定义一个用来在第一次引导时获取 Ignition 配置的 URL 来自定义蓝图。例如：

```
[customizations.ignition.firstboot]
url = "http://your_server/ignition_configuration.ig"
```

您必须输入一个指向在第一次引导过程中要获取的 Ignition 配置的 URL。

要为具有支持 Ignition 配置的简化的 RHEL for Edge 镜像自定义蓝图，请按照以下步骤执行：

### 先决条件

- 如果使用 **[customizations.ignition.embedded]** 自定义，您必须创建一个 Ignition 配置文件。
- 如果使用 **[customizations.ignition.firstboot]** 自定义，您必须已创建了一个其 URL 指向将在第一次引导过程中获取的 Ignition 配置的容器。
- 蓝图自定义 **[customizations.ignition.embedded]** 部分使 **coreos-installer-dracut** 能够根据 **osbuild** 文件的存在来定义 **-ignition-url|-ignition-file**。

### 流程

1. 以 Tom's Obvious, Minimal Language (TOML) 格式创建一个纯文本文件，其内容如下：

```
name = "simplified-installer-blueprint"
description = "Blueprint with Ignition for the simplified installer image"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[customizations.ignition.embedded]
config = "eyJ --- BASE64 STRING TRIMMED --- 19fQo="
```

其中：

- **name** 是蓝图的名称，**description** 是蓝图的描述。
- **version** 是根据语义版本控制方案的版本号。
- **modules** 和 **packages** 描述了要安装到镜像中的软件包名称和匹配版本 glob。例如，软件包 **name = "tmux"**，匹配版本 glob 是 **version = "3.3a"**。请注意，目前软件包和模块之间没有区别。
- **groups** 是要安装到镜像中的软件包组。例如 **groups = "anaconda-tools"** 组软件包。如果您不知道模块和组，请将其留空。



### 警告

如果要使用 Ignition 创建用户，则无法同时使用 FDO 自定义创建用户。您可以使用 Ignition 创建用户，并使用 FDO 复制配置文件。但是，如果您要创建用户，请使用 Ignition 或 FDO 创建他们，但不能同时创建它们。

2. 将蓝图推送（导入）到镜像构建器服务器：

```
# composer-cli blueprints push blueprint-name.toml
```

3. 列出现有的蓝图，以检查创建的蓝图是否已成功推送并存在。

```
# composer-cli blueprints show blueprint-name
```

4. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve blueprint-name
```

### 下一步

- 您可以使用您创建的蓝图来构建简化的安装程序镜像。请参阅 [使用镜像构建器创建 RHEL for Edge Simplified Installer 镜像](#)。

### 其他资源

- [RHEL for Edge 规格 v1.0.0](#)。

## 第 9 章 为 RHEL FOR EDGE 创建 VMDK 镜像

您可以使用 RHEL 镜像构建器为 RHEL for Edge 创建一个 **.vmdk** 镜像。您可以创建具有 Ignition 支持的 **edge-vsphere** 镜像类型，以便在引导过程的早期阶段将用户配置注入到镜像中。然后，您可以在 vSphere 上载入镜像，并在 vSphere 虚拟机中引导镜像。镜像与 ESXi 7.0 U2、ESXi 8.0 及之后的版本兼容。vSphere VM 与版本 19 和 20 兼容。

### 9.1. 使用 IGNITION 配置创建蓝图

为 **.vmdk** 镜像创建一个蓝图，并使用 **customizations.ignition** 部分对其进行自定义。通过这种方式，您可以创建您的镜像，在引导时，操作系统会将用户配置注入到镜像中。

#### 先决条件

- 您已创建了一个 Ignition 配置文件。例如：

```
{
  "ignition":{
    "version":"3.3.0"
  },
  "passwd":{
    "users":[
      {
        "groups":[
          "wheel"
        ],
        "name":"core",
        "passwordHash":"$6$jfuNnO9t1Bv7N"
      }
    ]
  }
}
```

#### 流程

1. 使用以下内容，创建一个 Tom 的 Obvious, Minimal Language (TOML) 格式的蓝图：

```
name = "vmdk-image"
description = "Blueprint with Ignition for the vmdk image"
version = "0.0.1"
packages = ["open-vm-tools"]
modules = []
groups = []
distro = ""

[[customizations.user]]
name = "admin"
password = "admin"
groups = ["wheel"]

[customizations.ignition.firstboot]
url = http://<IP_address>:8080/config.ig
```

其中：



- **name** 是蓝图的名称，**description** 是蓝图的描述。
- **version** 是根据语义版本控制方案的版本号。
- **modules** 和 **packages** 描述了要安装到镜像中的软件包名称和匹配版本 glob。例如，软件包 **name = "open-vm-tools"**。请注意，目前软件包和模块之间没有区别。
- **groups** 是要安装到镜像中的软件包组。例如 **groups = "anaconda-tools"** 组软件包。如果您不知道模块和组，请将其留空。
- **customizations.user** 创建一个用户名和密码来登录到虚拟机。
- **customizations.ignition.firstboot** 包含提供 Ignition 配置文件的 URL。



### 注意

默认情况下，**open-vm-tools** 软件包不包含在 **edge-vsphere** 镜像中。如果需要这个软件包，则必须将其包含在蓝图自定义中。

2. 将蓝图导入到镜像构建器服务器中：

```
# composer-cli blueprints push <blueprint-name>.toml
```

3. 列出现有的蓝图，以检查创建的蓝图是否已成功推送并存在：

```
# composer-cli blueprints show <blueprint-name>
```

4. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve <blueprint-name>
```

### 后续步骤

- 使用您创建的蓝图来构建 **.vmdk** 镜像。

## 9.2. 为 RHEL FOR EDGE 创建一个 VMDK 镜像

要创建 RHEL for Edge **.vmdk** 镜像，请在 RHEL 镜像构建器命令行界面中使用 'edge-vsphere' 镜像类型。

### 先决条件

- 您已为 **.vmdk** 镜像创建了一个蓝图。
- 您提供了一个提交的 OSTree 存储库，来将其嵌入到镜像中。例如：<http://10.0.2.2:8080/repo>。如需了解更多详细信息，请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。

### 流程

1. 启动 **.vmdk** 镜像的 compose：

```
# composer-cli compose start start-ostree <blueprint-name> edge-vsphere --<url>
```

-- *<url>* 是存储库的 URL，例如：**http://10.88.0.1:8080/repo**。

这时将显示一个确认已添加到队列中的 composer 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，为将来的任务保留 UUID 号。

2. 检查镜像的 compose 状态：

```
# composer-cli compose status
```

输出以以下格式显示状态：

```
$ <UUID> RUNNING date <blueprint-name> <blueprint-version> edge-vsphere
```

3. compose 过程完成后，下载生成的镜像文件：

```
# composer-cli compose image <UUID>
```

## 后续步骤

- 将 **.vmdk** 镜像上传到 vSphere。

## 9.3. 上传 VMDK 镜像并在 VSPHERE 中创建 RHEL 虚拟机

使用 **govc import.vmdk** CLI 工具将 **.vmdk** 镜像上传到 VMware vSphere，并在虚拟机中引导镜像。

### 先决条件

- 您使用 RHEL 镜像构建器创建了一个 **.vmdk** 镜像，并将其下载到主机系统。
- 您已安装 **govc import.vmdk** CLI 工具。
- 您已配置了 **govc import.vmdk** CLI 工具客户端。
  - 您必须在环境中设置以下值：

```
GOVC_URL
GOVC_DATACENTER
GOVC_FOLDER
GOVC_DATASTORE
GOVC_RESOURCE_POOL
GOVC_NETWORK
```

### 流程

1. 导航到您下载 **.vmdk** 镜像的目录。
2. 通过执行以下步骤在 vSphere 上启动镜像：
  - a. 将 **.vmdk** 镜像导入到 vSphere：

```
$ govc import.vmdk ./composer-api.vmdk foldername
```

- b. 在 vSphere 中创建虚拟机而不开机：

-

```
govc vm.create \  
-net="VM Network" -net.adapter=vmxnet3 \  
-disk.controller=pvscsi -on=false \  
-m=4096 -c=2 -g=rhel9_64Guest \  
-firmware=efi vm_name govc vm.disk.attach \  
-disk="foldername/composer-api.vmdk" govc vm.power -on \  
-vm vm_name -link=false \  
vm_name
```

- c. 打开虚拟机：

```
govc vm.power -on vmname
```

- d. 检索虚拟机 IP 地址：

```
HOST=$(govc vm.ip vmname)
```

- e. 使用您在蓝图中指定的用户名和密码，使用 SSH 登录到虚拟机：

```
$ ssh admin@HOST
```

## 第 10 章 创建 RHEL FOR EDGE AMI 镜像

您可以使用 RHEL 镜像构建器创建一个 RHEL for Edge **edge-ami** 自定义镜像。RHEL for Edge **edge-ami** 有 Ignition 支持，可在引导过程的早期阶段将用户配置注入到镜像中。然后，您可以将镜像上传到 AWS 云，并在 AWS 中启动 EC2 实例。您可以在 AMD 或 Intel 64 位构架上使用 AMI 镜像类型。

### 10.1. 为 EDGE AMI 镜像创建蓝图

为 **edge-ami** 镜像创建一个蓝图，并使用 **customizations.ignition** 部分对其进行自定义。因此，您可以创建镜像，并在引导镜像时注入用户配置。

#### 先决条件

- 您已创建了一个 Ignition 配置文件。例如：

```
{
  "ignition":{
    "version":"3.3.0"
  },
  "passwd":{
    "users":[
      {
        "groups":[
          "wheel"
        ],
        "name":"core",
        "passwordHash":"$6$jfuNnO9t1Bv7N"
      }
    ]
  }
}
```

如需了解更多详细信息，请参阅 [创建 Ignition 配置文件](#)。

#### 流程

1. 使用以下内容，创建一个 Tom 的 Obvious, Minimal Language (TOML) 格式的蓝图：

```
name = "ami-edge-image"
description = "Blueprint for Edge AMI image"
version = "0.0.1"
packages = ["cloud-init"]
modules = []
groups = []
distro = ""

[[customizations.user]]
name = "admin"
password = "admin"
groups = ["wheel"]

[customizations.ignition.firstboot]
url = http://<IP_address>:8080/config.ig
```

其中：

- **name** 是蓝图的名称，**description** 是蓝图的描述。
- **version** 是根据语义版本控制方案的版本号。
- **modules** 和 **packages** 描述了要安装到镜像中的软件包名称和匹配版本 glob。例如，软件包 **name = "open-vm-tools"**。请注意，目前软件包和模块之间没有区别。
- **groups** 是要安装到镜像中的软件包组。例如 **groups = "wheel"**。如果您不知道模块和组，请将其留空。
- **customizations.user** 创建一个用户名和密码来登录到虚拟机。
- **customizations.ignition.firstboot** 包含提供 Ignition 配置文件的 URL。



### 注意

默认情况下，**open-vm-tools** 软件包不包含在 **edge-vsphere** 镜像中。如果需要这个软件包，则必须将其包含在蓝图自定义中。

2. 将蓝图导入到镜像构建器服务器中：

```
# composer-cli blueprints push <blueprint-name>.toml
```

3. 列出现有的蓝图，以检查创建的蓝图是否已成功推送并存在：

```
# composer-cli blueprints show <blueprint-name>
```

4. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve <blueprint-name>
```

### 后续步骤

- 使用您创建的蓝图来构建 **edge-ami** 镜像。

## 10.2. 创建一个 RHEL FOR EDGE AMI 镜像

在 RHEL 镜像构建器命令行界面中创建一个 RHEL for Edge **edge-ami** 镜像。

### 先决条件

- 您已为 **edge-ami** 镜像创建了一个蓝图。
- 您提供了一个提交的 OSTree 存储库，来将其嵌入到镜像中。例如：<http://10.0.2.2:8080/repo>。如需了解更多详细信息，请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。

### 流程

1. 启动 **edge-ami** 镜像的 compose：

```
# composer-cli compose start start-ostree <blueprint-name> edge-ami --<url>
```

-- <url> 是存储库的 URL，例如：<http://10.88.0.1:8080/repo>。

这时将显示一个确认已添加到队列中的 composer 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，为将来的任务保留 UUID 号。

2. 检查镜像的 compose 状态：

```
# composer-cli compose status
```

输出以以下格式显示状态：

```
$ <UUID> RUNNING date <blueprint-name> <blueprint-version> edge-ami
```

3. compose 过程完成后，下载生成的镜像文件：

```
# composer-cli compose image <UUID>
```

## 后续步骤

- 将 **edge-ami** 镜像上传到 AWS

## 10.3. 将 RHEL EDGE AMI 镜像上传到 AWS

使用 CLI 将 **edge-ami** 镜像上传到 Amazon AWS Cloud 服务提供商。

### 先决条件

- 您已在 [AWS IAM](#) 账号管理器中配置了一个 **Access Key ID**。您已准备好一个可写的 S3 存储桶。您已为 AWS 存储桶创建了 [所需的角色](#)。您已安装了 **aws-cli** 工具。

### 流程

1. 配置 **aws-cli** 工具：

```
$ aws configure
```

- a. 配置您的配置文件。运行命令并输入您的 Access key ID 凭证、Secret access key、Default region name 和 default output 名称：

```
$ aws configure --profile
```

2. 列出现有存储桶：

```
$ aws s3 ls
```

3. 将您的镜像上传到 S3:

```
$ aws s3 cp <path_to_image/image> s3://<your_bucket_name>
```

4. 列出 S3 存储桶中的镜像：

```
$ aws s3 ls s3://<your_bucket_name>
```

5. 创建一个 **container-simple.json** 文件。将 "URL" 内容替换为 S3 存储桶。例如：**s3://rhel-edge-ami-us-west-2/2ba3c125-cc58-4cc0-861a-4cc78e892df6-image.raw**。

```
{
  "Description": "RHEL for Edge image",
  "Format": "edge-ami",
  "Url": "s3://rhel-edge-ami-us-west-2/UUID-image.raw"
}
```

6. 将 **edge.ami** 镜像作为 EC2 快照导入到 S3 存储桶。



### 注意

EC2 镜像必须位于与您创建的 S3 存储桶一样的区域。

```
$ aws ec2 import-snapshot --description "RHEL edge" \
--disk-container file://container-simple.json --region us-west-2
```

以下 **.json** 是命令输出的一个示例：

```
{
  "Description": "RHEL for Edge image",
  "Format": "edge-ami",
  "Url": "s3://rhel-edge-ami-us-west-2/UUID-image.raw"
}
```

7. 记录 json 中的 "ImportTaskId" 值。使用它来检查导入状态。在本例中，"ImportTaskId" 是 **import-snap-0f3055c4b7a454c85**。
8. 使用上一步中输出 json 文件中的 "ImportTaskId" 值来检查快照的导入状态：

```
$ aws ec2 describe-import-snapshot-tasks \
--import-task-ids import-snap-0f3055c4b7a454c85
{
  "ImportSnapshotTasks": [
    {
      "Description": "RHEL edge",
      "ImportTaskId": "import-snap-0f3055c4b7a454c85",
      "SnapshotTaskDetail": {
        "Description": "RHEL edge",
        "DiskImageSize": 10737418240.0,
        "Format": "RAW",
        "SnapshotId": "snap-001b267e752039eab",
        "Status": "completed",
        "Url": "s3://rhel-edge-ami-us-west-2/2ba3c125-cc58-4cc0-861a-4cc78e892df6-
image.raw",
        "UserBucket": {
          "S3Bucket": "rhel-edge-ami-us-west-2",
          "S3Key": "2ba3c125-cc58-4cc0-861a-4cc78e892df6-image.raw"
        }
      }
    },
    "Tags": []
  ]
}
```

```
    }  
  ]  
}
```

运行这个命令，直到 "Status" 标记为 "completed"。之后，您可以访问 EC2，来从快照创建 AMI 镜像，并启动它。

## 验证

要确认镜像上传成功：

1. 访问菜单中的 EC2，并在 AWS 控制台中选择正确的区域。镜像必须具有 available 状态，以表示它已被成功上传。
2. 在仪表盘上，选择镜像并点 Launch。  
在启动新实例时，您必须选择 UEFI 作为引导模式，并为 EC2 镜像选择至少 4GB RAM。
3. 您可以使用通过 Ignition 配置创建的用户名和密码登录到 AWS 上的 **edge-ami**。

## 其他资源

- [导入虚拟机所需的服务角色](#)



## 第 11 章 自动提供和加入带有 FDO 的 RHEL FOR EDGE 设备

您可以构建 RHEL for Edge Simplified Installer 镜像，并将其置备为 RHEL for Edge 镜像。FIDO 设备加入(FDO)过程会自动置备和加入边缘设备，并与网络连接的其他设备和系统交换数据。



### 重要

红帽提供了 **FDO** 流程作为技术预览功能，应在安全网络上运行。技术预览功能不受红帽产品服务等级协议 (SLA) 支持，且功能可能并不完整。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。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。

### 11.1. FIDO 设备加入(FDO)过程

FIDO 设备加入(FDO)是这样的过程：

- 置备并载入设备。
- 自动配置此设备的凭证。FDO 进程是一个自动加入机制，由安装的新设备触发。
- 允许此设备在网络上安全连接和交互。

借助 FIDO 设备加入(FDO)，您可以通过向 IoT 架构中添加新设备来实现安全的设备加入。这包括需要信任的，以及与正在运行的系统的其余部分集成的指定的设备配置。FDO 进程是一个自动加入机制，由安装的新设备触发。

FDO 协议执行以下任务：

- 解决信任和所有权链，以及大规模安全加入设备所需的自动化。
- 在制造阶段执行设备初始化，并为其实用用途执行后期设备绑定。这意味着，首先将设备绑定到管理系统的实际绑定会在设备第一次引导时进行。
- 支持自动安全设备加入，即在边缘位置不需要任何专业人员的零接触安装和加入。设备加入后，管理平台可以连接到它，并应用补丁、更新和回滚。

有了 FDO，您可以从以下方面获益：

- FDO 是向管理平台注册设备的一种安全、简单的方法。FDO 不是将 Kickstart 配置嵌入到镜像中，而是在设备首次引导过程中将设备凭据直接应用到 ISO 镜像。
- FDO 解决了之后绑定到设备的问题，使任何敏感数据可以通过安全的 FDO 通道共享。
- FDO 口令在注册并将配置和其他 secret 传递给系统之前以加密方式识别系统身份和所有权。这可以让非技术用户加电系统。

要构建 RHEL for Edge 简化的安装程序镜像并自动加入它，请提供现有的 OSTree 提交。生成的简化镜像包含部署了 OSTree 提交的原始镜像。引导简化的安装程序 ISO 镜像后，它提供边缘系统的 RHEL，您可以在硬盘上使用它，也可以在虚拟机中用作引导镜像。

RHEL for Edge Simplified Installer 镜像对设备的无人值守安装进行了优化，并支持基于网络的部署和非基于网络的部署。但是，对于基于网络的部署，它只支持 UEFI HTTP 引导。

FDO 协议基于以下服务器：

#### 制造服务器

1. 生成设备凭据。
2. 创建一个所有权凭证，用于在稍后的过程中设置设备的所有权。
3. 将设备绑定到特定的管理平台。

### 所有者管理系统

1. 从制造服务器接收所有权凭证，并成为相关设备的所有者。
2. 在过程的后期，它会在设备身份验证后在设备和所有者加入服务器之间创建一个安全通道。
3. 使用安全通道来发送所需信息，如将自动化载入到设备的文件和脚本。

### service-info API 服务器

根据客户端上可用的 Service-info API 服务器的配置和模块，它会在目标客户端设备上执行加入的最后步骤，如复制 SSH 密钥和文件、执行命令、创建用户、加密磁盘等

### Rendezvous 服务器

1. 从 Owner 管理系统获取所有权凭证，并创建一个设备 UUID 到 Owner 服务器 IP 的映射。然后，Rendezvous 服务器将设备 UUID 与目标平台匹配，并告知设备有关这个设备必须使用的 Owner 加入 服务器端点。
2. 第一次引导过程中，Rendezvous 服务器将是设备的联系点，它会将设备定向到所有者，因此设备和所有者可以建立一个安全通道。

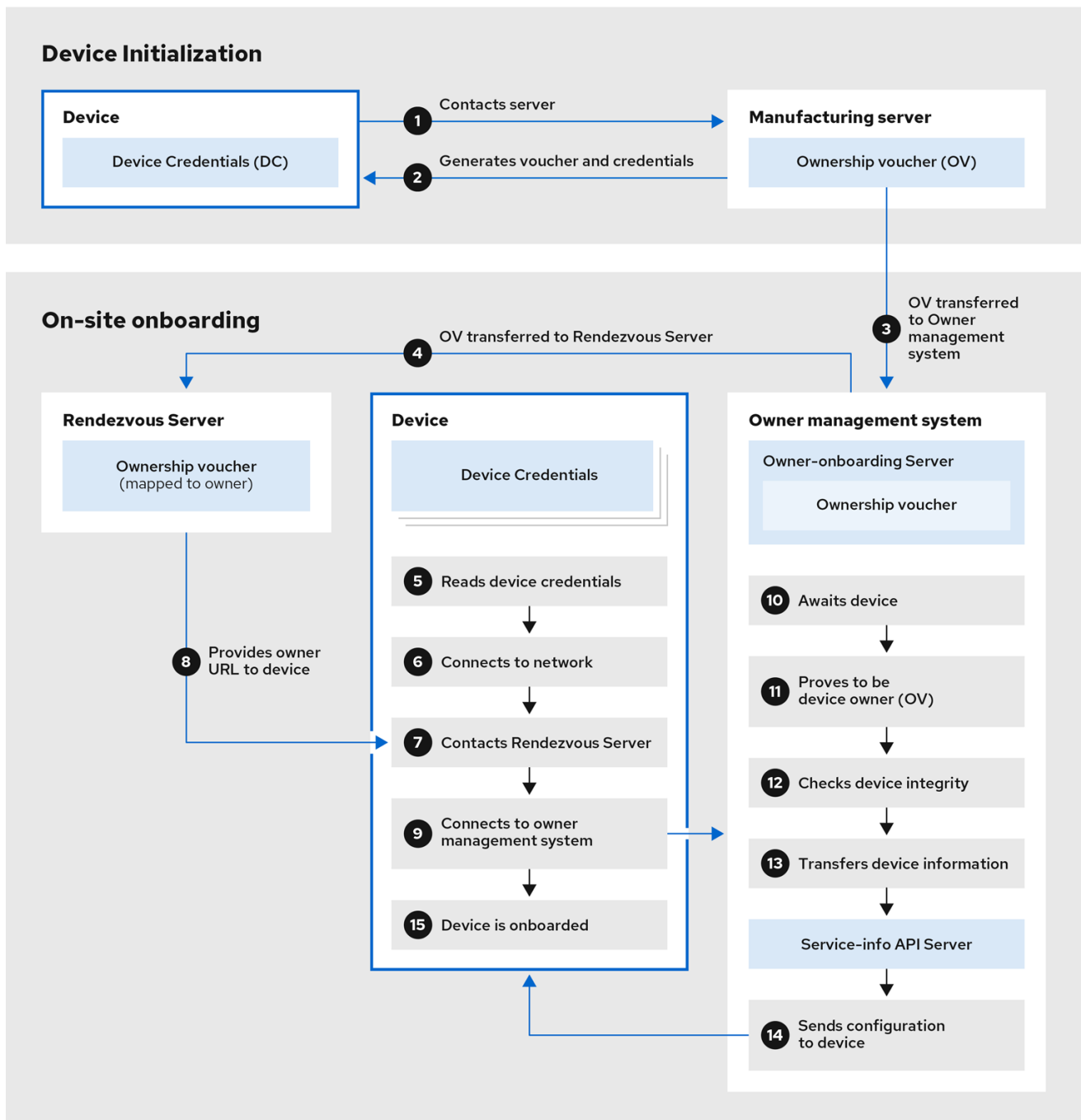
### 设备客户端

这安装在设备上。设备客户端执行以下操作：

1. 将对要执行载入自动化的多个服务器启动查询。
2. 使用 TCP/IP 协议与服务器进行通信。

下图代表 FIDO 设备加入工作流：

图 11.1. 在非网络环境中部署 RHEL for Edge



356\_RHEL\_0823

在 **设备初始化** 中，设备联系制造服务器以获取 FDO 凭证，一组要安装在带有 Rendezvous 服务器端点 (URL) 的操作系统上的证书和密钥。它还会获得所有权凭据，在需要更改所有者分配时单独维护。

1. 设备联系制造服务器
2. 制造服务器为该设备生成一个所有权凭证和设备凭证。
3. 所有权凭证被传到所有者加入服务器。

在 **现场加入** 时，设备会从其设备凭证获取 Rendezvous 服务器端点 (URL)，并联系 Rendezvous 服务器端点以开始加入过程，这将其重定向到所有者管理系统，后者是由所有者加入服务器和 Service Info API 服务器组成的。

4. 所有者加入服务器将所有权凭据传给 Rendezvous 服务器，这将创建一个所有权凭证到所有者的映射。
5. 设备客户端读取设备凭据。
6. 设备客户端连接到网络。
7. 连接到网络后，设备客户端会联系 Rendezvous 服务器。
8. Rendezvous 服务器将所有者端点 URL 发送给设备客户端，并注册该设备。
9. 设备客户端连接到 Rendezvous 服务器共享的所有者加入服务器。
10. 设备通过使用设备密钥签名语句来证明它是正确的设备。
11. 所有者加入服务器通过使用所有者凭证的最后密钥签署声明来证明自己是正确的。
12. 所有者加入服务器将设备的信息传给 Service Info API 服务器。
13. Service info API 服务器发送设备的配置。
14. 设备已加入。

## 11.2. 自动置备和注册 RHEL FOR EDGE 设备

要构建 RHEL for Edge 简化的安装程序镜像并自动加入它，请提供现有的 OSTree 提交。生成的简化镜像包含部署了 OSTree 提交的原始镜像。引导简化的安装程序 ISO 镜像后，它提供边缘系统的 RHEL，您可以在硬盘上使用它，也可以在虚拟机中用作引导镜像。

RHEL for Edge Simplified Installer 镜像对设备的无人值守安装进行了优化，并支持基于网络的部署和非基于网络的部署。但是，对于基于网络的部署，它只支持 UEFI HTTP 引导。

自动置备和加入 RHEL for Edge 设备涉及以下高级别步骤：

1. 安装并注册 RHEL 系统
2. 安装 RHEL 镜像构建器
3. 通过使用 RHEL 镜像构建器，为 **rhel-edge-container** 镜像类型创建一个带有 RHEL 自定义的蓝图。

```
name = "rhel-edge-container"
description = "Minimal RHEL for Edge Container blueprint"
version = "0.0.1"
```

4. 在 RHEL 镜像构建器中导入 RHEL for Edge 容器蓝图
5. 创建 RHEL for Edge 容器镜像
6. 使用 RHEL for Edge 容器镜像提供 OSTree 提交，其稍后在构建 RHEL for Edge Simplified Installer 镜像类型时使用
7. 使用存储设备路径的自定义和 FDO 自定义为 **edge-simplified-installer** 镜像类型创建一个蓝图

```
name = "rhel-edge-simplified-installer-with-fdo"
description = "Minimal RHEL for Edge Simplified Installer with FDO blueprint"
```

```

version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[customizations]
installation_device = "/dev/vda"

[customizations.fdo]
manufacturing_server_url = "http://10.0.0.2:8080"
diun_pub_key_insecure = "true"

```

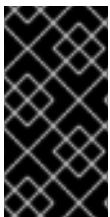
8. 构建简化的安装程序 RHEL for Edge 镜像
9. 下载 RHEL for Edge 简化的安装程序镜像
10. 此时，FDO 服务器基础架构应该已启动并运行，并配置了由 **service-info API** 服务器处理的特定的加入详情，这是所有者基础架构的一部分。
11. 将简化的安装程序 ISO 镜像安装到设备。FDO 客户端在简化的安装程序 ISO 上运行，并且 UEFI 目录结构使镜像可启动。
12. 网络配置可让设备连接到制造服务器，以执行初始设备凭据交换。
13. 系统到达端点后，会为该设备创建设备凭据。
14. 该设备使用设备凭证访问 Rendezvous 服务器，它会根据 Rendezvous 服务器具有的凭证来检查加密凭证，然后 Rendezvous 服务器将设备重定向到所有者服务器。
15. 设备联系所有者服务器。它们建立一个相互信任，并根据 Service-info API 服务器的配置进行加入的最终步骤。例如，它会在设备中安装 SSH 密钥，传输文件，创建用户，运行命令，加密文件系统等等。

## 其他资源

- [FDO 自动加入技术](#)

## 11.3. 生成密钥和证书

要运行 FIDO Device Onboarding (FDO) 基础架构，您需要生成密钥和证书。FDO 生成这些密钥和证书来配置生产服务器。在安装服务时，FDO 会自动生成证书和 **.yaml** 配置文件，重新创建它们是可选的。安装并启动服务后，它会使用默认设置运行。



### 重要

红帽提供了 **fdo-admin-tool** 工具作为技术预览功能，应在安全网络上运行。技术预览功能不受红帽产品服务等级协议 (SLA) 支持，且功能可能并不完整。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。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。

## 先决条件

- 您已安装了 **fdo-admin-cli** RPM 软件包

## 流程

1. 在 `/etc/fdo` 目录中生成密钥和证书：

```
$ for i in "diun" "manufacturer" "device-ca" "owner"; do fdo-admin-tool generate-key-and-cert
$i; done
$ ls keys
device_ca_cert.pem device_ca_key.der diun_cert.pem diun_key.der manufacturer_cert.pem
manufacturer_key.der owner_cert.pem owner_key.der
```

2. 检查在 `/etc/fdo/keys` 目录中创建的密钥和证书：

```
$ tree keys
```

您可以看到以下输出：

```
- device_ca_cert.pem
- device_ca_key.der
- diun_cert.pem
- diun_key.dre
- manufacturer_cert.pem
- manufacturer_key.der
- owner_cert.pem
- owner_key.pem
```

## 其他资源

- 请参阅 `fdo-admin-tool generate-key-and-cert --help` 手册页

## 11.4. 安装并运行制造服务器

**fdo-manufacturing-server** RPM 软件包使您能够运行 FDO 协议的制造服务器组件。它还存储其他组件，如所有者凭证、制造商密钥以及有关制造会话的信息。在设备安装过程中，制造服务器为特定设备生成设备凭证，包括 GUID、rendezvous 信息和其他元数据。在过程的后期，设备使用这个 rendezvous 信息联系 Rendezvous 服务器。



### 重要

红帽提供了 **fdo-manufacturing-server** 工具作为技术预览功能，应在安全网络上运行，因为红帽产品服务级别协议(SLA)不支持技术预览功能，且可能无法正常工作。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。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。

要安装 **manufacturing 服务器** RPM 软件包，请完成以下步骤：

## 流程

1. 安装 **fdo-admin-cli** 软件包：

```
# dnf install -y fdo-admin-cli
```

2. 检查是否安装了 **fdo-manufacturing-server** RPM 软件包：

```
$ rpm -qa | grep fdo-manufacturing-server --refresh
```

- 检查是否已正确安装了这些文件：

```
$ ls /usr/share/doc/fdo
```

您可以看到以下输出：

```
Output:
manufacturing-server.yml
owner-onboarding-server.yml
rendezvous-info.yml
rendezvous-server.yml
serviceinfo-api-server.yml
```

- 可选：检查每个文件的内容，例如：

```
$ cat /usr/share/doc/fdo/manufacturing-server.yml
```

- 配置制造服务器。您必须提供以下信息：

- 制造服务器 URL
- Rendezvous 服务器的 IP 地址或 DNS 名称
- 您生成的密钥和证书的路径。请参阅 [生成密钥和证书](#)。

您可以在 `/usr/share/doc/fdo/manufacturing-server.yml` 目录中找到制造服务器配置文件的示例。以下是创建并保存在 `/etc/fdo` 目录中的一个 **manufacturing server.yml** 示例。它包含您创建的目录、证书、密钥的路径，会合服务器 IP 地址和默认端口。

```
session_store_driver:
  Directory:
    path: /etc/fdo/stores/manufacturing_sessions/
ownership_voucher_store_driver:
  Directory:
    path: /etc/fdo/stores/owner_vouchers
public_key_store_driver:
  Directory:
    path: /etc/fdo/stores/manufacturer_keys
bind: "0.0.0.0:8080"
protocols:
  plain_di: false
diun:
  mfg_string_type: SerialNumber
  key_type: SECP384R1
  allowed_key_storage_types:
    - Tpm
    - FileSystem
  key_path: /etc/fdo/keys/diun_key.der
  cert_path: /etc/fdo/keys/diun_cert.pem
rendezvous_info:
  - deviceport: 8082
  ip_address: 192.168.122.99
  ownerport: 8082
```

```

protocol: http
manufacturing:
  manufacturer_cert_path: /etc/fdo/keys/manufacturer_cert.pem
  device_cert_ca_private_key: /etc/fdo/keys/device_ca_key.der
  device_cert_ca_chain: /etc/fdo/keys/device_ca_cert.pem
  owner_cert_path: /etc/fdo/keys/owner_cert.pem
  manufacturer_private_key: /etc/fdo/keys/manufacturer_key.der

```

## 6. 启动制造服务器。

- a. 检查 systemd 单元文件是否在服务器中：

```

# systemctl list-unit-files | grep fdo | grep manufacturing
fdo-manufacturing-server.service disabled disabled

```

- b. 启用并启动制造服务器。

```

# systemctl enable --now fdo-manufacturing-server.service

```

- c. 在防火墙中打开默认端口：

```

# firewall-cmd --add-port=8080/tcp --permanent
# systemctl restart firewalld

```

- d. 确保该服务正在侦听端口 8080:

```

# ss -ltn

```

7. 使用简化的安装程序将 RHEL for Edge 安装到您的系统上。请参阅 [构建简化的安装程序镜像以提供 RHEL for Edge 镜像](#)。

## 其他资源

- [manufacturing-server.yml 示例](#)
- [FDO 自动加入术语](#)

## 11.5. 安装、配置和运行 RENDEZVOUS 服务器

安装 **fdo-rendezvous-server** RPM 软件包，以使系统在第一个设备引导时能够接收制造服务器生成的凭证。然后 Rendezvous 服务器将设备 UUID 与目标平台或云进行匹配，并告知该设备必须使用哪个所有者服务器端点的设备。

### 先决条件

- 您创建了一个 **manufacturer\_cert.pem** 证书。请参阅 [生成密钥和证书](#)。
- 您可以将 **manufacturer\_cert.pem** 证书复制到 Rendezvous 服务器中的 **/etc/fdo/keys** 目录中。

### 流程

1. 安装 **fdo-rendezvous-server** RPM 软件包：



```
# dnf install -y fdo-rendezvous-server
```

2. 创建 **rendezvous-server.yml** 配置文件，包括制造商证书的路径。您可以在 **/usr/share/doc/fdo/rendezvous-server.yml** 中找到示例。以下示例显示了保存在 **/etc/fdo/rendezvous-server.yml** 中的配置文件。

```
storage_driver:
  Directory:
    path: /etc/fdo/stores/rendezvous_registered
session_store_driver:
  Directory:
    path: /etc/fdo/stores/rendezvous_sessions
trusted_manufacturer_keys_path: /etc/fdo/keys/manufacturer_cert.pem
max_wait_seconds: ~
bind: "0.0.0.0:8082"
```

3. 检查 Rendezvous 服务器服务状态：

```
# systemctl list-unit-files | grep fdo | grep rende
fdo-rendezvous-server.service disabled disabled
```

- a. 如果停止并禁用该服务，请启用并启动它：

```
# systemctl enable --now fdo-rendezvous-server.service
```

4. 检查服务器是否在侦听默认配置的端口 8082：

```
# ss -ltn
```

5. 如果在这个服务器上配置了防火墙，请打开端口：

```
# firewall-cmd --add-port=8082/tcp --permanent
# systemctl restart firewalld
```

## 11.6. 安装、配置和运行所有者服务器

安装 **fdo-owner-cli** 和 **fdo-owner-onboarding-server** RPM 软件包，以使系统能够在第一个设备引导时接收制造服务器生成的凭证。然后 Rendezvous 服务器将设备 UUID 与目标平台或云进行匹配，并告知该设备必须使用哪个所有者服务器端点的设备。

### 先决条件

- 部署服务器的设备有一个受信任的平台模块(TPM)设备来加密磁盘。如果没有，在引导 RHEL for Edge 设备时会出现一个错误。
- 您使用密钥和证书创建了 **device\_ca\_cert.pem**、**owner\_key.der** 和 **owner\_cert.pem**，并将它们复制到 **/etc/fdo/keys** 目录中。

### 流程

1. 在这个服务器中安装所需的 RPM：

```
# dnf install -y fdo-owner-cli fdo-owner-onboarding-server
```

2. 准备 **owner-onboarding-server.yml** 配置文件，并将其保存到 **/etc/fdo/** 目录中。包括您已复制的证书的路径，以及有关此文件中发布所有者服务器服务的信息。  
 以下是 **/usr/share/doc/fdo/owner-onboarding-server.yml** 中的一个示例。您可以查找对服务信息 API 的引用，如 URL 或身份验证令牌。

```
---
ownership_voucher_store_driver:
  Directory:
    path: /etc/fdo/stores/owner_vouchers
session_store_driver:
  Directory:
    path: /etc/fdo/stores/owner_onboarding_sessions
trusted_device_keys_path: /etc/fdo/keys/device_ca_cert.pem
owner_private_key_path: /etc/fdo/keys/owner_key.der
owner_public_key_path: /etc/fdo/keys/owner_cert.pem
bind: "0.0.0.0:8081"
service_info_api_url: "http://localhost:8083/device_info"
service_info_api_authentication:
  BearerToken:
    token: Kpt5P/5fIBkaiNSvDYS3cEdBQXJn2Zv9n1D50431/lo=
owner_addresses:
  - transport: http
    addresses:
      - ip_address: 192.168.122.149
```

3. 创建和配置服务信息 API。
  - a. 为加入添加自动信息，如要用户创建、要复制或创建的文件、要执行的命令、要加密的磁盘等。使用 **/usr/share/doc/fdo/serviceinfo-api-server.yml** 中的服务信息 API 配置文件示例作为模板，来在 **/etc/fdo/** 下创建配置文件。

```
---
service_info:
  initial_user:
    username: admin
    sshkeys:
      - "ssh-rsa AAAA...."
  files:
    - path: /root/resolv.conf
      source_path: /etc/resolv.conf
  commands:
    - command: touch
      args:
        - /root/test
      return_stdout: true
      return_stderr: true
  diskencryption_clevis:
    - disk_label: /dev/vda4
      binding:
        pin: tpm2
        config: "{}"
      reencrypt: true
  additional_serviceinfo: ~
```

```
bind: "0.0.0.0:8083"
device_specific_store_driver:
  Directory:
    path: /etc/fdo/stores/serviceinfo_api_devices
service_info_auth_token: Kpt5P/5flBkaiNSvDYS3cEdBQXJn2Zv9n1D50431/lo=
admin_auth_token: zJNoErq7aa0RusJ1w0tkTjdITdMCWYkndzVv7F0V42Q=
```

#### 4. 检查 systemd 单元的状态：

```
# systemctl list-unit-files | grep fdo
fdo-owner-onboarding-server.service    disabled    disabled
fdo-serviceinfo-api-server.service     disabled    disabled
```

##### a. 如果停止并禁用该服务，请启用并启动它：

```
# systemctl enable --now fdo-owner-onboarding-server.service
# systemctl enable --now fdo-serviceinfo-api-server.service
```



#### 注意

每次更改配置文件时，都必须重启 **systemd** 服务。

#### 5. 检查服务器是否在侦听默认配置的端口 8083：

```
# ss -ltn
```

#### 6. 如果在这个服务器上配置了防火墙，请打开端口：

```
# firewall-cmd --add-port=8081/tcp --permanent
# firewall-cmd --add-port=8083/tcp --permanent
# systemctl restart firewalld
```

## 11.7. 使用 FDO 身份验证自动加入 RHEL FOR EDGE 设备

要准备设备以自动加入 RHEL for Edge 设备，并将其为安装过程的一部分提供，请完成以下步骤：

### 先决条件

- 为 RHEL for Edge 构建一个 OSTree 提交，并用它来生成一个 **edge-simplified-installer** 工件。
- 您的设备已组装好。
- 您已安装了 **fdo-manufacturing-server** RPM 软件包。请参阅 [安装制造服务器软件包](#)。

### 流程

1. 通过在设备上引导 RHEL for Edge Simplified installer 镜像来启动安装过程。您可以使用 CD-ROM 或 USB 闪存驱动器安装它，例如：
2. 通过终端验证设备是否已达到制造服务，以执行初始设备凭据交换，并且生成所有权凭证。您可以在由 **manufacturing-sever.yml** 文件中的 **ownership\_voucher\_store\_driver:** 参数配置的存储位置找到所有权凭证。

目录应有一个 **ownership\_voucher** 文件，其名称为 GUID 格式，这表示正确的设备凭据已添加到设备中。

onboarding 服务器使用设备凭据对服务器进行身份验证。然后，它会将配置传递给设备。设备从 onboarding 服务器接收配置后，它会接收 SSH 密钥并在设备上安装操作系统。最后，系统会自动重启，使用存储在 TPM 中的强大密钥对其进行加密。

## 验证

设备自动重启后，您可以使用您作为 FDO 进程的一部分创建的凭证登录到该设备。

- 通过提供您在服务信息 API 中创建的用户名和密码，登录该设备。

## 其他资源

- [从 USB 闪存驱动器部署简化的 ISO 镜像](#)

## 第 12 章 使用 FDO 加入一个带有数据库后端的 RHEL FOR EDGE 设备

您可以使用 FDO 服务器 - **manufacturer-server**, **join-server**, 和 **rendezvous**- 支持从 SQL 后端（如 SQLite 或 PostgreSQL 数据库）而不是文件，来存储和查询所有者凭证。这样，您可以在 FDO 服务器选项选择一个 SQL 数据存储，以及凭证和其他参数，将所有者凭证存储在 SQL 数据库中，从而加入了一个 RHEL for Edge 设备。SQL 文件已打包在 RPM 中。



### 注意

目前，SQL 后端不支持所有 FDO 功能。

### 12.1. 加入带有 FDO 数据库的设备

使用 SQL 数据库来加入 Edge 设备。以下示例使用 **diesel** 工具，但您也可以使用 SQLite 或 PostgreSQL 数据库。



### 注意

您可以在一些服务器中使用与其他服务器中的文件系统存储不同的数据库存储，例如，使用文件系统存储加入制造服务器和用于 Rendezvous 和 Owner 服务器的 Postgres。

#### 先决条件

- 您已使用 FDO 生成配置制造服务器的密钥和证书。请参阅链接 [\[生成密钥和证书\]](#)
- 您已安装并配置了制造服务器。请参阅 [安装和运行制造服务器](#)
- 您已安装并配置了 rendezvous 服务器。请参阅 [安装、配置和运行 Rendezvous 服务器](#)
- 您已安装并配置了所有者服务器。请参阅 [安装、配置和运行所有者服务器](#)
- 您在 `/etc/fdo` 中有服务器配置文件
- 您为 RHEL for Edge 构建了一个 OSTree 提交，并用它来生成一个 **edge-simplified-installer** 工件
- 您的设备已组装
- 您已将 **diesel** 工具或 SQL 数据库安装到您的主机上。
- 您已配置了数据库系统，并有创建表的权限。

#### 流程

1. 安装以下软件包：

```
$ dnf install -y sqlite sqlite-devel libpq libpq-devel
```

2. 访问 `/usr/share/doc/fdo/migrations/*` 目录。它包含您需要在安装完制造、rendezvous 和所有者服务器的 RPM 后，为每个服务器和类型组合创建数据库的 **.sql** 文件。
3. 初始化数据库内容。您可以使用 SQL 数据库（如 SQLite 或 PostgreSQL）或 **diesel** 工具来为您运行 SQL。

- 如果您使用 SQL 数据库，使用例如，用户创建、访问管理来配置数据库服务器。配置完数据库服务器后，您可以运行数据库。
  - 如果您不想在数据库系统中运行 **.sql** 文件，您可以使用 **diesel** 工具为您运行 sql。如果您使用 **diesel** 工具，在配置了数据库后，请使用 **diesel migration run** 命令来创建数据库：
4. 配置 DB 系统后，您可以使用安装在 **/usr/share/doc/fdo/migrations/\*** 处的 **.sql** 文件来为每个服务器类型创建数据库。
- 您必须使用与您要初始化的服务器类型和数据库类型匹配的 **.sql** 文件。例如，当在 PostgreSQL 数据库中初始化 Owner Onboarding Server 时，您必须使用 **/usr/share/doc/fdo/migrations/migrations\_owner\_onboarding\_server\_postgres/up.sql** 文件夹。迁移文件夹中的 **up.sql** 文件创建数据库，**down.sql** 文件销毁数据库。
5. 创建数据库后，更改特定服务器的配置文件，以使其使用数据库。  
每台服务器都有一个存储配置部分。

- 制造商服务器：**ownership\_voucher\_store\_driver**
- 所有者服务器：**ownership\_voucher\_store\_driver**
- Rendezvous 服务器：**storage\_driver**

- a. 对于 **manufacturing-server.yml** 文件，在编辑器中打开它，并更改存储数据库：

```
$ sudo editor manufacturing-server.yml
```

- b. 在 Directory 部分下更改 **ownership\_voucher\_store\_driver** 配置：

```
$ /home/rhel/fido-device-onboard-rs/aio-dir/stores/owner_vouchers
```

- c. 指定以下详情：

- 您正在使用的数据库类型：SQLite 或 PostgreSQL
- 服务器类型：例如，在使用 PostgreSQL 时设置以下配置：

```
ownership_voucher_store_driver:
  Postgres:
    Manufacturer
```

- a. 重复步骤来配置所有者服务器和 Rendezvous 服务器。

6. 运行 FDO 加入服务。如需了解更多详细信息，请参阅自动提供和加入带有 FDO 的 RHEL for Edge 设备运行制造服务器来启动 FDO 加入过程设备初始化：

```
$ sudo LOG-LEVEL=debug SQLITE_MANUFACTURER-DATABASE_URL=./manufacturer-db.sqlite ./usr/libexec/fdo/fdo-manufacturing-server
```

加入过程在两个阶段发生：

- 通常发生在制造站点中的设备初始化阶段。
- 发生在设备的最终目的地的设备加入过程。  
因此，存储在制造服务器的数据库中的所有者凭证必须导出，并传到最终的所有者数据库。

7. 要从 manufacturing-vouchers 数据库文件导出所有权凭证，请将 Owner Voucher 复制到所有者，以继续 FDO 加入协议。

- a. 创建一个文件夹 **export** :

```
$ mkdir export
```

- b. 通过提供命令所需的所有变量，导出制造数据库中存在的所有者凭证。

```
$ fdo-owner-tool export-manufacturer-vouchers DB_TYPE DB_URL PATH [GUID]
```

#### DB\_TYPE

保存所有者凭证的制造 DB 的类型：sqlite、postgres

#### DB\_URL

数据库连接 URL 或数据库文件的路径

#### PATH

将导出所有者凭证的目录的路径

#### GUID

要导出的所有者凭证的 GUID。如果没有提供 GUID，则将导出所有的所有者凭证。

8. OV 必须传送到最终的所有者的数据库。为此，请使用 **fdo-owner-tool** 来导入所有者凭证。改为所有者数据库。通过运行以下命令来导入所有权凭证：

```
$ fdo-owner-tool import-ownership-vouchers DB_TYPE DB_URL SOURCE_PATH
```

#### DB\_TYPE

导入 OV 的所有者 DB 的类型。可能的值：sqlite、postgres

#### DB\_URL

DB 连接 URL 或 DB 文件的路径

#### SOURCE\_PATH

要导入的 OV 的路径，或要导入的所有 OV 所在的目录的路径

命令逐个读取 <SOURCE\_PATH> 中指定的每个 OV 一次，并尝试将它们导入到数据库。如果命令发现错误，它会返回一个输出，其中包含有故障的 OV 的 GUID，并制定了导致错误的信息。无故障 OVs 被导入到数据库中。设备从加入服务器接收配置。然后，设备接收一个到 SSH 密钥，并开始在上安装操作系统。最后，操作系统自动在设备中重启，并使用存储在 TPM 中的强大密钥对设备进行加密。

## 第 13 章 在基于网络的环境中部署 RHEL FOR EDGE 镜像

您可以使用 RHEL 安装程序图形用户界面或 Kickstart 文件部署 RHEL for Edge 镜像。部署 RHEL for Edge 镜像的整体流程取决于部署环境是基于网络的还是非基于网络的。



### 注意

要在裸机上部署镜像，请使用 Kickstart 文件。

### 基于网络的部署

在基于网络的环境中部署 RHEL for Edge 镜像涉及以下高级别步骤：

- a. 提取镜像文件内容。
- b. 设置 Web 服务器
- c. 安装镜像

### 13.1. 提取 RHEL FOR EDGE 镜像提交

下载提交后，提取 `.tar` 文件，并记下 `ref` 名称和提交 ID。

下载的提交文件由一个 `.tar` 文件和 **OSTree** 存储库组成。**OSTree** 存储库有一个提交和一个 **compose.json** 文件。

**compose.json** 文件包含与提交相关的信息（如 "Ref"、引用 ID 和提交 ID）的信息元数据。提交 ID 具有 RPM 软件包。

要提取软件包内容，请执行以下步骤：

#### 先决条件

- 创建一个 Kickstart 文件或使用现有的文件。

#### 流程

1. 提取下载的镜像 `.tar` 文件：

```
# tar xvf <UUID>-commit.tar
```

2. 转到已提取 `.tar` 文件的目录。

它有一个 **compose.json** 文件和一个 OSTree 目录。**compose.json** 文件有提交号，**OSTree** 目录有 RPM 软件包。

3. 打开 **compose.json** 文件，再记下提交 ID 号。当您继续设置 Web 服务器时，您需要这个数字。如果您安装了 **jq** JSON 处理程序，您还可以通过使用 **jq** 工具来检索提交 ID：

```
# jq '["ostree-commit"]' < compose.json
```

4. 列出提交中的 RPM 软件包。

```
# rpm-ostree db list rhel/9/x86_64/edge --repo=repo
```



5. 使用 Kickstart 文件运行 RHEL 安装程序。另外，您可以使用任何现有的文件，或使用 Kickstart Generator 工具创建一个文件。
- 在 Kickstart 文件中，确保包含有关如何置备文件系统、创建用户以及如何获取和部署 RHEL for Edge 镜像的详细信息。RHEL 安装程序在安装过程中使用此信息。

以下是 Kickstart 文件示例：

```
lang en_US.UTF-8
keyboard us
timezone Etc/UTC --isUtc
text
zerombr
clearpart --all --initlabel
autopart
reboot
user --name=core --group=wheel
sshkey --username=core "ssh-rsa AAAA3Nza...."
rootpw --lock
network --bootproto=dhcp

ostreesetup --nogpg --osname=rhel --remote=edge --url=https://mirror.example.com/repo/ --
ref=rhel/9/x86_64/edge
```

基于 OSTree 的安装使用 **ostreesetup** 命令来设置配置。它使用以下标记获取 OSTree 提交：

- **--nogpg** - 禁用 GNU Privacy Guard (GPG) 密钥验证。
- **--osname** - 操作系统安装的管理根。
- **--remote** - 操作系统安装的管理根
- **--url** - 要从中安装的存储库的 URL。
- **--ref** - 安装使用的存储库中的分支名称。
- **--url=http://mirror.example.com/repo/** - 是您提取边缘提交，并通过 **nginx** 提供它的主机系统的地址。您可以使用该地址从客户机计算机访问主机系统。  
例如，如果您在 `/var/www/html` 目录中提取提交镜像，并在主机名为 **www.example.com** 的计算机上通过 **nginx** 提供提交，则 **--url** 参数的值为 **http://www.example.com/repo**。



### 注意

使用 http 协议启动一个服务来提供提交，因为 Apache HTTP 服务器上没有启用 https。

### 其他资源

- [下载 RHEL for Edge 镜像](#)
- [创建 Kickstart 文件](#)

## 13.2. 设置 WEB 服务器以安装 RHEL FOR EDGE 镜像

提取 RHEL for Edge 镜像内容后，设置一个 web 服务器，以使用 HTTP 向 RHEL 安装程序提供镜像提交详情。

下例提供了使用容器建立 Web 服务器的步骤。

## 先决条件

- 已在您的系统上安装了 Podman。请参阅 [如何在 RHEL 中安装 Podman](#)

## 流程

1. 使用以下步骤创建 **nginx** 配置文件：

```
events {  
  
}  
  
http {  
    server{  
        listen 8080;  
        root /usr/share/nginx/html;  
    }  
}  
  
pid /run/nginx.pid;  
daemon off;
```

2. 使用以下说明创建 Dockerfile：

```
FROM registry.access.redhat.com/ubi8/ubi  
RUN dnf -y install nginx && dnf clean all  
COPY kickstart.ks /usr/share/nginx/html/  
COPY repo /usr/share/nginx/html/  
COPY nginx /etc/nginx.conf  
EXPOSE 8080  
CMD ["/usr/sbin/nginx", "-c", "/etc/nginx.conf"]  
ARG commit  
ADD ${commit} /usr/share/nginx/html/
```

其中,

- **Kickstart.ks** 是 RHEL for Edge 镜像的 Kickstart 文件的名称。Kickstart 文件包含指令信息。为了帮助您稍后管理镜像，建议包含用于 greenboot 检查的检查和设置。因此，您可以更新 Kickstart 文件以含以下设置：

```
lang en_US.UTF-8  
keyboard us  
timezone Etc/UTC --isUtc  
text  
zerombr  
clearpart --all --initlabel  
autopart  
reboot  
user --name=core --group=wheel  
sshkey --username=core "ssh-rsa AAAA3Nza...."  
  
ostreesetup --nogpg --osname=rhel --remote=edge  
--url=https://mirror.example.com/repo/
```

```

--ref=rhel/9/x86_64/edge

%post
cat << EOF > /etc/greenboot/check/required.d/check-dns.sh
#!/bin/bash

DNS_SERVER=$(grep nameserver /etc/resolv.conf | cut -f2 -d" ")
COUNT=0

# check DNS server is available
ping -c1 $DNS_SERVER
while [ $? != '0' ] && [ $COUNT -lt 10 ]; do

    COUNT++
    echo "Checking for DNS: Attempt $COUNT ."
    sleep 10
    ping -c 1 $DNS_SERVER
done
EOF
%end

```

任何 HTTP 服务都可以托管 OSTree 存储库，而使用容器的示例只是如何执行此操作的一个选项。Dockerfile 执行以下任务：

- a. 使用最新的通用基础镜像(UBI)
- b. 安装 Web 服务器(nginx)
- c. 向服务器添加 Kickstart 文件
- d. 将 RHEL for Edge 镜像提交添加到服务器

### 3. 构建 Docker 容器

```
# podman build -t name-of-container-image --build-arg commit=uuid-commit.tar .
```

### 4. 运行容器

```
# podman run --rm -d -p port:8080 localhost/name-of-container-image
```

因此，服务器建立好了，可以通过使用 **commit.tar** 存储库和 Kickstart 文件来准备启动 RHEL 安装程序。

## 13.3. 使用 KICKSTART 对边缘设备执行有人值守的安装

对于基于网络的环境中的有人值守安装，您可以使用 RHEL 安装程序 ISO、Kickstart 文件和 Web 服务器将 RHEL for Edge 镜像安装到设备。web 服务器提供 RHEL for Edge Commit 和 Kickstart 文件，来引导 [RHEL Installer ISO](#) 镜像。

### 先决条件

- 您已通过运行 web 服务器使 RHEL for Edge Commit 可用。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。
- 您已创建了一个 **.qcow2** 磁盘镜像，来用作有人值守安装的目标。请参阅 [使用 qemu-img 创建虚拟磁盘镜像](#)。

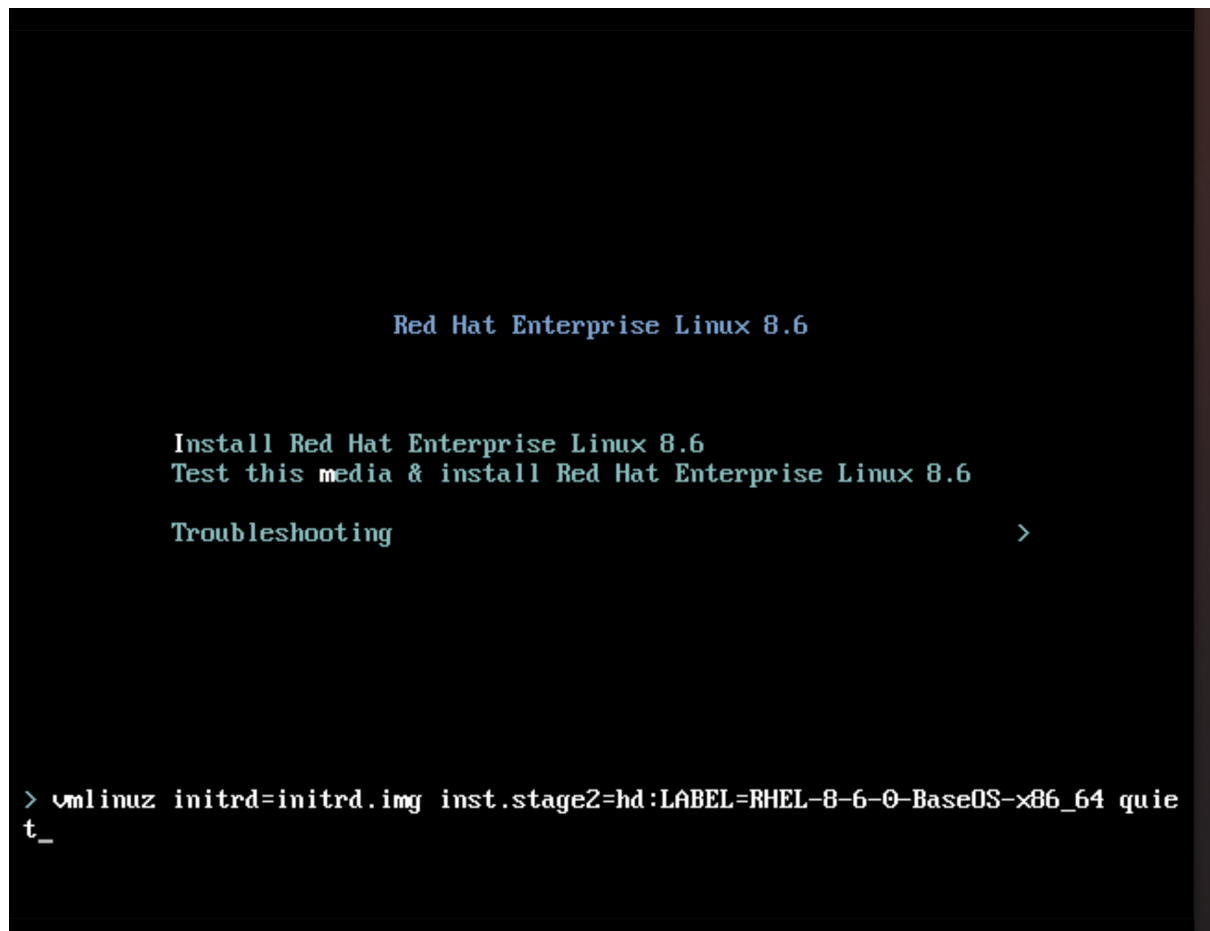
## 流程

1. 使用 **libvirt virt-install** 工具运行 RHEL Anaconda 安装程序，来创建一个具有 RHEL 操作系统的虚拟机(VM)。在有人值守安装中使用 **.qcow2** 磁盘镜像作为目标磁盘：

```
virt-install \
--name rhel-edge-test-1 \
--memory 2048 \
--vcpus 2 \
--disk path=prepared_disk_image.qcow2,format=qcow2,size=8 \
--os-variant rhel9 \
--cdrom /home/username/Downloads/rhel-9-x86_64-boot.iso
```

2. 在安装屏幕中：

图 13.1. Red Hat Enterprise Linux 引导菜单



- a. 按 **e** 键添加额外的内核参数：

```
inst.ks=http://edge_device_ip:port/kickstart.ks
```

kernel 参数指定您要使用 Kickstart 文件而不是 RHEL 安装程序中包含的 RHEL 镜像来安装 RHEL。

- b. 添加内核参数后，按 **Ctrl+X** 使用 Kickstart 文件引导 RHEL 安装。  
RHEL 安装程序启动、从服务器 (HTTP) 端点获取 Kickstart 文件并执行命令，包括从 HTTP 端点安装 RHEL for Edge 镜像提交的命令。安装完成后，RHEL 安装程序会提示您登录详情。

## 验证

1. 在登录屏幕中，输入您的用户帐户凭证并点 **Enter**。
2. 验证 RHEL for Edge 镜像是否已成功安装。

```
$ rpm-ostree status
```

命令输出提供镜像提交 ID，并显示安装成功。

以下是输出示例：

```
State: idle
Deployments:
* ostree://edge:rhel/9/x86_64/edge
  Timestamp: 2020-09-18T20:06:54Z
  Commit: 836e637095554e0b634a0a48ea05c75280519dd6576a392635e6fa7d4d5e96
```

## 其他资源

- [如何将 Kickstart 文件嵌入到 ISO 镜像中。](#)
- [引导安装。](#)

## 13.4. 使用 KICKSTART 对边缘设备执行无人值守安装

对于在基于网络的环境中的无人值守安装，您可以使用 Kickstart 文件和 web 服务器将 RHEL for Edge 镜像安装到 Edge 设备。web 服务器提供 RHEL for Edge Commit 和 Kickstart 文件，这两个工件都用于启动 [RHEL Installer ISO](#) 镜像。

### 先决条件

- 您已在主机系统上安装了 **qemu-img** 工具。
- 您已创建了一个 **.qcow2** 磁盘镜像，来安装您创建的提交。请参阅 [在 CLI 中使用 RHEL 镜像构建器创建系统镜像](#)。
- 您有一个正在运行的 Web 服务器。请参阅 [为非基于网络的部署创建 RHEL for Edge 容器镜像](#)。

### 流程

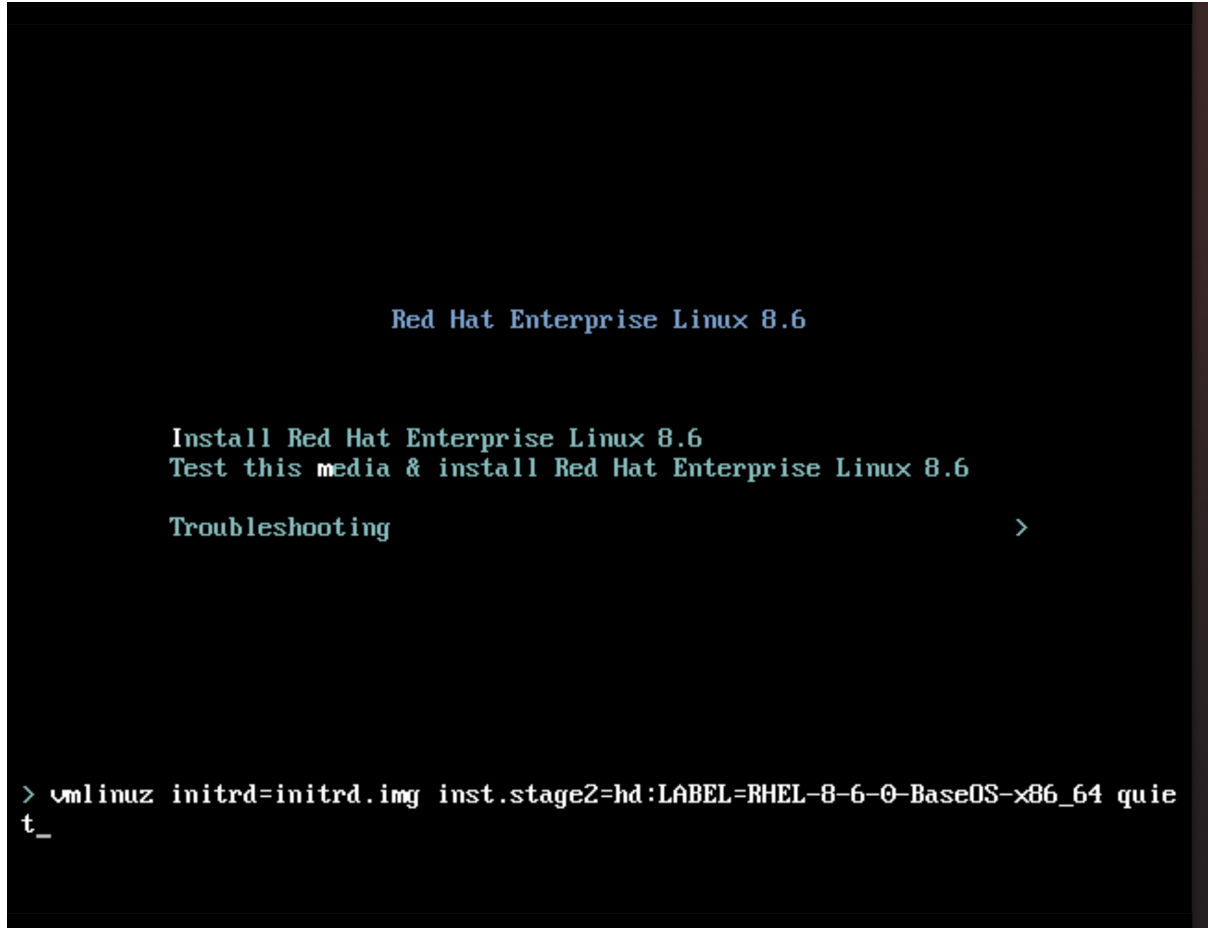
1. 运行 RHEL for Edge 容器镜像以启动 web 服务器。服务器获取 RHEL for Edge 容器镜像中的提交，并变为可用且正在运行。
2. 使用 **libvirt virt-install** 运行 RHEL Anaconda 安装程序，传递自定义的 **.qcow2** 磁盘镜像：

```
virt-install \
--name rhel-edge-test-1 \
--memory 2048 \
```

```
--vcpus 2 \
--disk path=prepared_disk_image.qcow2,format=qcow2,size=8 \
--os-variant rhel9 \
--cdrom /home/username/Downloads/rhel-9-x86_64-boot.iso
```

3. 在安装屏幕中：

图 13.2. Red Hat Enterprise Linux 引导菜单



- a. 按 **TAB** 键，并添加 Kickstart 内核参数：

```
inst.ks=http://web-server_device_ip:port/kickstart.ks
```

kernel 参数指定您要使用 Kickstart 文件而不是 RHEL 安装程序中包含的 RHEL 镜像来安装 RHEL。

- b. 添加内核参数后，按 **Ctrl+X** 使用 Kickstart 文件引导 RHEL 安装。  
RHEL 安装程序启动，从服务器(HTTP)端点获取 Kickstart 文件，并执行命令，包括从 HTTP 端点安装 RHEL for Edge 镜像提交的命令。安装完成后，RHEL 安装程序会提示您登录详情。

## 验证

1. 在登录屏幕中，输入您的用户帐户凭证并点 **Enter**。
2. 验证 RHEL for Edge 镜像是否已成功安装。

```
$ rpm-ostree status
```

命令输出提供镜像提交 ID，并显示安装成功。

以下是一个输出示例：

```
State: idle
Deployments:
* ostree://edge:rhel/9/x86_64/edge
  Timestamp: 2020-09-18T20:06:54Z
  Commit: 836e637095554e0b634a0a48ea05c75280519dd6576a392635e6fa7d4d5e96
```

### 其他资源

- [如何将 Kickstart 文件嵌入到 ISO 镜像中。](#)
- [引导安装。](#)

## 第 14 章 在非基于网络的环境中部署 RHEL FOR EDGE 镜像

RHEL for Edge Container (**.tar**)与 RHEL for Edge Installer (**.iso**)镜像类型相结合将生成一个 ISO 镜像。ISO 镜像可以在镜像部署到设备过程中在断开连接的环境中使用。但是，网络访问可能需要网络访问权限来构建不同的工件。

在非网络环境中部署 RHEL for Edge 镜像涉及以下高级别步骤：

1. 下载 RHEL for Edge 容器。有关如何下载 RHEL for Edge 镜像的信息，请参阅 [下载 RHEL for Edge 镜像](#)。
2. 将 RHEL for Edge 容器镜像加载到 Podman 中
3. 在 Podman 中运行 RHEL for Edge 容器镜像
4. 加载 RHEL for Edge Installer 蓝图
5. 构建 RHEL for Edge 安装程序镜像
6. 准备一个 **.qcow2** 磁盘
7. 引导虚拟机 (VM)
8. 安装镜像

### 14.1. 为非网络部署创建 RHEL FOR EDGE 容器镜像

您可以通过将下载的 RHEL for Edge Container OSTree 提交加载到 Podman 来构建正在运行的容器。为此，请按照以下步骤执行：

#### 先决条件

- 您创建并下载了一个 RHEL for Edge Container OSTree 提交。
- 已在您的系统上安装了 **Podman**。请参阅 [如何在 RHEL 中安装 Podman](#)。

#### 流程

1. 进入您下载了 RHEL for Edge Container OSTree 提交的目录。
2. 将 RHEL for Edge 容器 OSTree 提交加载到 **Podman** 中。

```
$ sudo podman load -i UUID-container.tar
```

命令输出提供了镜像 ID，例如：

```
@8e0d51f061ff1a51d157804362bc875b649b27f2ae1e66566a15e7e6530ce63
```

3. 使用上一步中生成的镜像 ID，标记新的 RHEL for Edge 容器镜像。

```
$ sudo podman tag image-ID localhost/edge-container
```

**podman tag** 命令为本地镜像分配额外名称。

4. 运行名为 **edge-container** 的容器。



```
$ sudo podman run -d --name=edge-container -p 8080:8080 localhost/edge-container
```

`podman run -d --name=edge-container` 命令根据 `localhost/edge-container` 镜像将名称分配给容器。

#### 5. 列出容器：

```
$ sudo podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
2988198c4c4b .../localhost/edge-container /bin/bash 3 seconds ago Up 2 seconds ago
edge-container
```

因此，**Podman** 运行一个容器，它会使用一个带有 RHEL for Edge 容器提交的 OSTree 存储库。

## 14.2. 为非网络部署创建 RHEL FOR EDGE 安装程序镜像

构建正在运行的容器来为带有 **RHEL for Edge Container** 提交的存储库提供服务后，创建一个 **RHEL for Edge Installer (.iso)** 镜像。**RHEL for Edge Installer (.iso)** 拉取 **RHEL for Edge Container (.tar)** 提供的提交。在 Podman 中载入 **RHEL for Edge Container** 提交后，它会以 URL 格式公开 **OSTree**。

要在 CLI 中创建 RHEL for Edge Installer 镜像，请按照以下步骤操作：

### 先决条件

- 为 RHEL for Edge 镜像创建了一个蓝图。
- 您创建了一个 RHEL for Edge 容器镜像，并使用 web 服务器部署了它。

### 流程

#### 1. 开始创建 RHEL for Edge Installer 镜像。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url URL-OSTree-repository
blueprint-name image-type
```

其中，

- `ref` 与客户用来构建 OSTree 存储库的值相同
  - `URL-OSTree-repository` 是要嵌入到镜像中的提交 OSTree 存储库的 URL。例如：`http://10.0.2.2:8080/repo/`。请参阅[为非基于网络的部署创建 RHEL for Edge 容器镜像](#)。
  - `blueprint-name` 是 RHEL for Edge Installer 蓝图名称。
  - `image-type` 是 **edge-installer**。  
这时将显示一个确认已添加到队列中的 `composer` 进程。它还显示创建的镜像的通用唯一标识符 (UUID) 号。使用 UUID 号来跟踪构建。另外，记录 UUID 号以易于执行进一步的任务。
- #### 2. 检查镜像 compose 状态。

```
# composer-cli compose status
```

命令输出以以下格式显示状态：

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



### 注意

完成镜像创建过程需要几分钟时间。

要中断镜像创建过程，请运行：

```
# composer-cli compose cancel <UUID>
```

要删除现有镜像，请运行：

```
# composer-cli compose delete <UUID>
```

RHEL 镜像构建器在镜像构建期间拉取由正在运行的容器提供的提交。

镜像构建完成后，您可以下载生成的 ISO 镜像。

3. 下载镜像。请参阅[下载 RHEL for Edge 镜像](#)。  
镜像就绪后，您可以将它用于[非网络部署](#)。请参阅[为非基于网络的部署安装 RHEL for Edge 镜像](#)。

### 其他资源

- [使用命令行界面为非基于网络的部署创建 RHEL for Edge 安装程序镜像](#)。

## 14.3. 为非网络部署安装 RHEL FOR EDGE 镜像

要安装 RHEL for Edge 镜像，请按照以下步骤执行：

### 先决条件

- 您创建了 RHEL for Edge Installer ISO 镜像。
- 已停止运行的容器。
- 安装您创建的提交的磁盘镜像。
- 已安装 **edk2-ovmf** 软件包。
- 已安装 **virt-viewer** 软件包。
- 您可以使用用户帐户自定义您的蓝图。请参阅[在 RHEL web 控制台使用 RHEL 镜像构建器为 RHEL for Edge 镜像创建蓝图](#)。



### 警告

如果您没有在蓝图中定义用户帐户自定义，您将无法登录到 ISO 镜像。

## 流程

1. 创建一个 **qcow** VM 磁盘文件来安装(.iso)镜像。这是虚拟机(VM)的硬盘镜像。例如：

```
$ qemu-img create -f qcow2 diskfile.qcow2 20G
```

2. 使用 **virt-install** 命令，将磁盘用作驱动器，将安装程序 ISO 镜像用作 CD-ROM 来引导虚拟机。例如：

```
$ virt-install \
--boot uefi \
--name VM_NAME \
--memory 2048 \
--vcpus 2 \
--disk path=diskfile.qcow2 \
--cdrom /var/lib/libvirt/images/UUID-installer.iso \
--os-variant rhel9.0
```

这个命令指示 **virt-install** 进行：

- 指示虚拟机使用 UEFI 引导，而不是 BIOS。
- 挂载安装 ISO。
- 使用在第一步中创建的硬盘镜像。  
它为您提供 Anaconda 安装程序。RHEL 安装程序启动，从 ISO 中加载 Kickstart 文件并执行命令，包括安装 RHEL for Edge 镜像提交的命令。安装完成后，RHEL 安装程序会提示输入登录详情。



### 注意

Anaconda 已预先配置为在安装过程中使用容器提交。但是，您需要在分区之间设置系统配置，如磁盘分区、时区等。

3. 使用 **virt-viewer** 连接到 Anaconda GUI 来设置系统配置：

```
$ virt-viewer --connect qemu:///system --wait VM_NAME
```

4. 重启系统以完成安装。
5. 在登录屏幕中，指定您的用户帐户凭证并点 **Enter**。

## 验证步骤

1. 验证 RHEL for Edge 镜像是否已成功安装。

```
$ rpm-ostree status
```

命令输出提供镜像提交 ID，并显示安装成功。

## 第 15 章 管理 RHEL FOR EDGE 镜像

要管理 RHEL for Edge 镜像，您可以执行以下任何管理任务：

- 在 RHEL web 控制台或命令行上使用镜像构建器编辑 RHEL for Edge 镜像蓝图
- 使用镜像构建器命令行构建提交更新
- 更新 RHEL for Edge 镜像
- 在节点上配置 `rpm-ostree` 远程，以更新节点策略
- 手动恢复 RHEL for Edge 镜像，或使用 `greenboot` 自动恢复

### 15.1. 使用镜像构建器编辑 RHEL FOR EDGE 镜像蓝图

您可以将 RHEL for Edge 镜像蓝图编辑为：

- 添加您可能需要的其他组件
- 修改任何现有组件的版本
- 删除任何现有组件

#### 15.1.1. 在 RHEL web 控制台中使用镜像构建器向 RHEL for Edge 蓝图中添加组件

要在 RHEL for Edge 镜像蓝图中添加组件，请确保您满足以下先决条件，然后按照步骤编辑对应的蓝图。

##### 先决条件

- 在 RHEL 系统上，您已访问 RHEL 镜像构建器仪表盘。
- 您已为 RHEL for Edge 镜像创建了蓝图。

##### 流程

1. 在 RHEL 镜像构建器仪表盘上，点击您要编辑的蓝图。  
要搜索特定的蓝图，请在过滤器文本框中输入蓝图名称，然后点 **Enter**。
2. 在蓝图的右上角点 **Edit Packages**。  
这会打开 **Edit blueprints** 向导。
3. 在 **Details** 页面中，更新蓝图名称并点 **Next**。
4. 在 **Packages** 页面中，按照以下步骤操作：
  - a. 在 **Available Packages** 中，在过滤器文本框中输入您要添加的软件包名称，然后单击 **Enter**。  
此时会显示组件名称的列表。
  - b. 点 **>** 将组件添加到蓝图中。
5. 在 **Review** 页面中，点 **Save**。  
蓝图现在使用新软件包进行了更新。

### 15.1.2. 在 web 控制台中使用 RHEL 镜像构建器从蓝图中删除组件

要从使用 RHEL 镜像构建器创建的蓝图中删除一个或多个不需要的组件，请确保您满足以下先决条件，然后按照以下流程操作。

#### 先决条件

- 在 RHEL 系统上，您已访问 RHEL 镜像构建器仪表盘。
- 您已为 RHEL for Edge 镜像创建了蓝图。
- 您已在 RHEL for Edge 蓝图中添加至少一个组件。

#### 流程

1. 在 RHEL 镜像构建器仪表盘上，点击您要编辑的蓝图。  
要搜索特定的蓝图，请在过滤器文本框中输入蓝图名称，然后点 **Enter**。
2. 在蓝图的右上角点 **Edit Packages**。  
这会打开 **Edit blueprints** 向导。
3. 在 **Details** 页面中，更新蓝图名称并点 **Next**。
4. 在 **Packages** 页面中，按照以下步骤操作：
  - a. 从 **Chosen packages** 软件包中，点 < 删除所选的组件。您还可以点击 << 来一次删除所有软件包。
5. 在 **Review** 页面中，点 **Save**。  
现在，蓝图已被更新。

### 15.1.3. 使用命令行界面编辑 RHEL for Edge 镜像蓝图

您可以使用 RHEL 镜像构建器命令行更改 RHEL for Edge 镜像蓝图的规范。为此，请确保您满足以下先决条件，然后按照步骤编辑对应的蓝图。

#### 先决条件

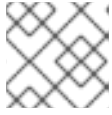
- 您有访问 RHEL 镜像构建器命令行的权限。
- 您已创建了 RHEL for Edge 镜像蓝图。

#### 流程

1. 将蓝图保存（导出）到本地文本文件：

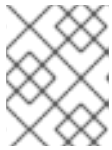
```
# composer-cli blueprints save BLUEPRINT-NAME
```

2. 使用您选择的文本编辑器编辑 **BLUEPRINT-NAME.toml** 文件并进行更改。  
在完成编辑前，验证该文件是否为一个有效的蓝图：
3. 增加版本号。  
确保您使用 Semantic Versioning 方案。

**注意**

如果您不更改版本，则会自动增加版本的补丁组件。

4. 检查内容是否是有效的 TOML 规格。如需更多信息，请参阅 TOML 文档。

**注意**

TOML 文档是一款社区产品，不受红帽支持。您可以在 <https://github.com/toml-lang/toml/issues> 中报告任何问题。

5. 保存文件并关闭编辑器。
6. 将蓝图推送（导入）回 RHEL 镜像构建器服务器：

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

**注意**

将蓝图推送回 RHEL 镜像构建器服务器时，请提供包括 **.toml** 扩展名的文件名。

7. 验证上传到 RHEL 镜像构建器的内容是否与您的编辑匹配：

```
# composer-cli blueprints show BLUEPRINT-NAME
```

8. 检查蓝图中列出的组件和版本是否有效：

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

## 15.2. 更新 RHEL FOR EDGE 镜像

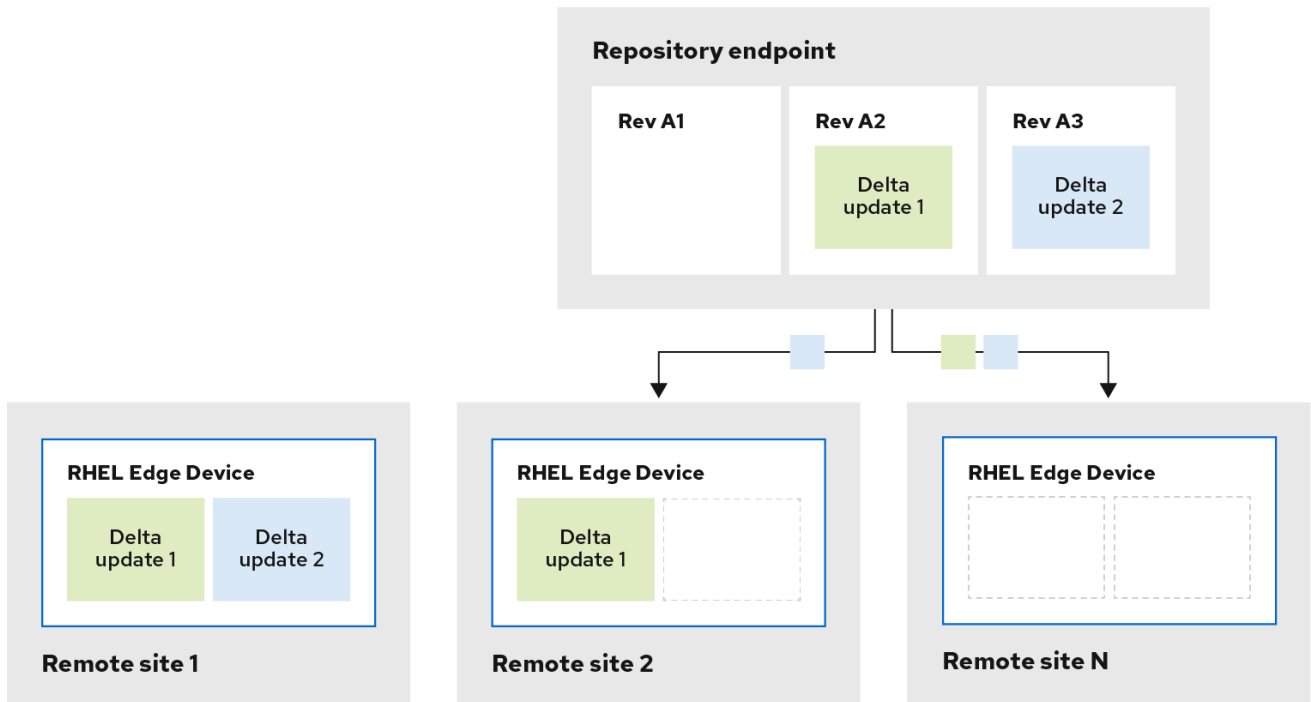
### 15.2.1. RHEL for Edge 镜像更新是如何部署的

使用 RHEL for Edge 镜像，您可以手动部署更新，也可以自动部署部署过程。更新以原子方式应用，其中知道每个更新的状态，更新仅在重启时暂存和应用。由于在重新引导设备之前不会看到任何更改，因此您可以调度重新引导以确保尽可能最高的正常运行时间。

在镜像更新过程中，只有更新的操作系统内容通过网络传输。与传输整个镜像相比，这可以使部署过程更有效率。**/usr** 中的操作系统二进制文件和库是 **只读的**，**/var** 和 **/etc** 目录保持 **读和写** 状态。

移至新部署时，**/etc** 和 **/var** 目录会复制到新部署中，具有 **读和写** 权限。**/usr** 目录作为软链接复制到新部署目录，具有 **只读** 权限。

下图演示了 RHEL for Edge 镜像更新部署过程：



125\_RHEL\_1020

默认情况下，使用与 **chroot** 操作类似的流程引导新系统，即系统启用对文件系统的控制访问，同时控制对底层服务器环境的公开。新的 **/sysroot** 目录主要有以下部分：

- 位于 **/sysroot/ostree/repo** 目录中的存储库数据库。
- 位于 **/sysroot/ostree/deploy/rhel/deploy** 目录中的文件系统修订，其由系统更新中的每个操作创建。
- **/sysroot/ostree/boot** 目录，它链接到在之前点上的部署。请注意，**/ostree** 是到 **/sysroot/ostree** 的软链接。**/sysroot/ostree/boot** 目录中的文件不会重复。如果部署期间文件没有改变，则使用同一文件。这些文件是硬链接到存储在 **/sysroot/ostree/repo/objects** 目录中的另一个文件。

操作系统使用以下方法选择部署：

1. **dracut** 工具解析 **initramfs root** 文件系统中的 **ostree** 内核参数，并将 **/usr** 目录设置为只读绑定挂载。
2. 将 **/sysroot** 中的部署目录绑定到 **/** 目录。
3. 使用 **MS\_MOVE** 挂载标志重新挂载已挂载 **dirs** 的操作系统

如果出现错误，您可以使用 **rpm-ostree** 清楚命令，通过删除旧部署来执行部署回滚。每个客户端机器都包含一个存储在 **/ostree/repo** 中的 **OSTree** 存储库，以及存储在 **/ostree/deploy/\$STATEROOT/\$CHECKSUM** 中的一组部署。

借助 RHEL for Edge 镜像中的部署更新，您可以从跨多个设备的更好的系统一致性、更容易的可重复性，以及系统状态更改前后之间更好的隔离中受益。

### 15.2.2. 构建一个提交更新

您可以在蓝图中进行更改后构建一个提交更新，例如：

- 添加系统所需的附加软件包
- 修改任何现有组件的软件包版本
- 删除任何现有的软件包。

## 先决条件

- 您已更新了运行 RHEL 镜像构建器的系统。
- 您创建了一个蓝图更新。
- 您之前已创建了一个 OSTree 存储库，并通过 HTTP 提供服务。请参阅 [设置一个 web 服务器以安装 RHEL for Edge 镜像](#)。

## 流程

1. 使用以下参数开始新提交镜像的 `compose` : `--url`、`--ref`、`blueprint-name`、`edge-commit`。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url http://localhost:8080/repo
<blueprint-name> edge-commit
```

在开始 `compose` 前，命令指示 `compose` 进程从 OSTree 存储库获取元数据。生成的新 OSTree 提交包含一个作为父镜像的原始 OSTree 提交的引用。

2. `compose` 进程完成后，获取 `.tar` 文件。

```
# composer-cli compose image <UUID>
```

3. 将提交提取到一个临时目录，以便您可以将提交历史记录存储在 OSTree 仓库中。

```
$ tar -xf UUID.tar -C /var/tmp
```

4. 使用 `tar -xf` 命令检查生成的 OSTree 存储库提交。它将 `tar` 文件提取到磁盘，以便您可以检查生成的 OSTree 存储库：

```
$ ostree --repo=/var/tmp/repo log rhel/9/x86_64/edge
commit d523ef801e8b1df69ddb73ce810521b5c44e9127a379a4e3bba5889829546fa
Parent: f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
ContentChecksum:
f0f6703696331b661fa22d97358db48ba5f8b62711d9db83a00a79b3ae0dfe16
Date: 2023-06-04 20:22:28 +0000
Version: 9
```

在输出示例中，存储库中有一个引用父提交的 OSTree 提交。父提交与之前所做的原始 OSTree 提交中的校验和相同。

5. 使用 `ostree pull-local` 命令合并两个提交：

```
$ sudo ostree --repo=/var/srv/httpd/repo pull-local /var/tmp/repo
20 metadata, 22 content objects imported; 0 bytes content written
```

此命令将磁盘位置（如 `/var/tmp`）的任何新元数据和内容复制到 `/var/srv/httpd` 中的目标 OSTree 存储库。



## 验证

### 1. 检查目标 OSTree 存储库：

```
$ ostree --repo=/var/srv/httpd/repo log rhel/9/x86_64/edge
commit d523ef801e8b1df69ddbf73ce810521b5c44e9127a379a4e3bba5889829546fa
Parent: f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
ContentChecksum:
f0f6703696331b661fa22d97358db48ba5f8b62711d9db83a00a79b3ae0dfe16
Date: 2023-06-04 20:22:28 /+0000
Version: 9
(no subject)

commit f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
ContentChecksum:
9054de3fe5f1210e3e52b38955bea0510915f89971e3b1ba121e15559d5f3a63
Date: 2023-06-04 20:01:08 /+0000
Version: 9
(no subject)
```

您可以看到目标 OSTree 存储库现在在存储库中按逻辑顺序包含两个提交。验证成功后，您可以更新 RHEL for Edge 系统。

### 15.2.3. 手动部署 RHEL for Edge 镜像更新

编辑 RHEL for Edge 蓝图后，您可以更新镜像提交。RHEL 镜像构建器为更新的 RHEL for Edge 镜像生成一个新的提交。使用此新提交来部署具有最新软件包版本或附加软件包的镜像。

要部署 RHEL for Edge 镜像更新，请确保您满足先决条件，然后按照以下步骤操作。

#### 先决条件

- 在 RHEL 系统上，您已访问 RHEL 镜像构建器仪表盘。
- 您已创建了 RHEL for Edge 镜像蓝图。
- 您编辑了 RHEL for Edge 镜像蓝图。

#### 流程

1. 在 RHEL 镜像构建器仪表盘上，单击 **Create Image**。
2. 在 **Create Image** 窗口中，执行以下步骤：
  - a. 在 Image 输出页面中：
    - i. 从 **Select a blueprint** 下拉列表中，选择您编辑的蓝图。
    - ii. 从 **Image output type** 下拉列表中，选择 **RHEL for Edge Commit (.tar)**。点 **Next**。
  - b. 在 **OSTree settings** 页面中，输入：
    - i. 在 **Repository URL** 字段中，输入要嵌入到镜像中的提交的 OSTree 存储库的 URL。例如：<http://10.0.2.2:8080/repo/>。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。

- ii. 在 **Parent commit** 字段中，指定之前生成的父提交 ID。请参阅 [提取 RHEL for Edge 镜像提交](#)
- iii. 在 **Ref** 字段中，您可以为提交指定名称或将其留空。默认情况下，web 控制台将 **Ref** 指定为 **rhel/9/arch\_name/edge**。点 **Next**。
- c. 在 **Review** 页面中，检查自定义，并点 **Create image**。RHEL 镜像构建器开始为更新的蓝图创建一个 RHEL for Edge 镜像。完成镜像创建过程需要几分钟时间。  
要查看 RHEL for Edge 镜像创建进度，点 breadcrumbs 中的蓝图名称，然后单击 **Images** 选项卡。

生成的镜像包含您添加的最新软件包（若有），并且具有原始 **提交 ID** 作为父项。

3. 下载生成的 RHEL for Edge Commit (**.tar**) 镜像。
  - a. 在 **Images** 选项卡中，点 **Download** 将 RHEL for Edge Commit (**.tar**) 镜像保存到您的系统上。
4. 提取 OSTree 提交(**.tar**)文件。

```
# tar -xf UUID-commit.tar -C UPGRADE_FOLDER
```

5. 升级 OSTree 存储库：

```
# ostree --repo=/usr/share/nginx/html/repo pull-local UPGRADE_FOLDER
# ostree --repo=/usr/share/nginx/html/repo summary -u
```

6. 在置备的 RHEL 系统上，从原始边缘镜像验证当前状态。

```
$ rpm-ostree status
```

如果没有新的提交 ID，请运行以下命令验证是否有可用的升级：

```
$ rpm-ostree upgrade --check
```

命令输出提供当前活动的 OSTree 提交 ID。

7. 更新 OSTree，使新 OSTree 提交 ID 可用。

```
$ rpm-ostree upgrade
```

ostree 验证存储库是否有更新。如果是，它将获取您重新引导系统的更新和请求，以便您可以激活此新提交更新的部署。

8. 再次检查当前状态：

```
$ rpm-ostree status
```

现在，您可以看到有 2 个提交可用：

- 活跃的父级提交。
- 一个未激活且包含 1 个添加的差异的新提交。

9. 要激活新部署并使新提交处于活动状态，请重启您的系统。

```
# systemctl reboot
```

Anaconda 安装程序将重新引导至新部署。在登录屏幕上，您可以看到可供您引导的新部署。

10. 如果要引导到最新的部署(提交)，**rpm-ostree upgrade** 命令会自动排序引导条目，因此新部署使列表中的第一个。（可选）您可以使用键盘中的箭头键选择 GRUB 菜单条目并按 **Enter**。
11. 提供您的登录用户帐户凭证。
12. 验证 OSTree 状态：

```
$ rpm-ostree status
```

命令输出提供活动的提交 ID。

13. 要查看更改的软件包（如果有），请在父提交和新提交之间运行差异：

```
$ rpm-ostree db diff parent_commit new_commit
```

更新显示您已安装的软件包可用并可供使用。

#### 15.2.4. 使用命令行手动部署 RHEL for Edge 镜像更新

编辑 RHEL for Edge 蓝图后，您可以更新镜像提交。RHEL 镜像构建器为更新的 RHEL for Edge 镜像生成一个新的提交。使用新的提交，通过 CLI 使用最新的软件包版本或使用其他软件包来部署镜像。

要使用 CLI 部署 RHEL for Edge 镜像更新，请确保您满足先决条件，然后按照以下步骤执行。

##### 先决条件

- 已创建 RHEL for Edge 镜像蓝图。
- 已编辑 RHEL for Edge 镜像蓝图。请参阅 [使用命令行界面编辑 RHEL for Edge 镜像蓝图](#)。

##### 流程

1. 使用以下参数创建 RHEL for Edge Commit(.tar)镜像：

```
# composer-cli compose start-ostree --ref ostree_ref --url URL-OSTree-repository - blueprint_name_ image-type
```

其中

- **ref** 是您在为 Edge Container 提交创建 RHEL 时提供的引用。例如：**rhel/9/x86\_64/edge**。
  - **URL-OSTree-repository** 是要嵌入到镜像中的提交的 OSTree 存储库的 URL。例如：<http://10.0.2.2:8080/repo/>。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。
  - **image-type** 为 **edge-commit**。  
RHEL 镜像构建器为更新的蓝图创建一个 RHEL for Edge 镜像。
2. 检查 RHEL 中的 Edge 镜像创建进度：

```
# composer-cli compose status
```



## 注意

镜像创建过程可能需要长达十到 30 分钟才能完成。

生成的镜像包含您添加的最新软件包（若有），并且具有原始 **提交 ID** 作为父项。

3. 下载生成的 RHEL for Edge 镜像。如需更多信息，请参阅 [使用 RHEL 镜像构建器命令行界面下载 RHEL for Edge 镜像](#)。
4. 提取 OSTree 提交。

```
# tar -xf UUID-commit.tar -C upgrade_folder
```

5. 使用 httpd 为 OSTree 提交提供服务。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。
6. 升级 OSTree 存储库：

```
# ostree --repo=/var/www/html/repo pull-local /tmp/ostree-commit/repo
# ostree --repo=/var/www/html/repo summary -u
```

7. 在从原始边缘镜像置备的 RHEL 系统中，验证当前状态：

```
$ rpm-ostree status
```

如果没有新的提交 ID，请运行以下命令验证是否有可用的升级：

```
$ rpm-ostree upgrade --check
```

命令输出提供当前活动的 OSTree 提交 ID。

8. 更新 OSTree 以使新的 OSTree 提交 ID 可用：

```
$ rpm-ostree upgrade
```

ostree 验证存储库是否有更新。如果是，它将获取您重新引导系统的更新和请求，以便您可以激活此新提交更新的部署。

9. 再次检查当前状态：

```
$ rpm-ostree status
```

现在，您应看到有 2 个可用的提交：

- 活跃的父级提交
- 一个未激活且包含 1 个添加的差异的新提交。

10. 要激活新部署并使新提交处于活动状态，请重启您的系统。

```
# systemctl reboot
```

Anaconda 安装程序将重新引导至新部署。在登录屏幕上，您可以看到可供您引导的新部署。

如果您引导进入新的部署，您可以看到新的提交 ID。如果您引导进入旧的部署，您可以看到旧的提交 ID。

11. 如果要引导进入最新的部署，`rpm-ostree upgrade` 命令会自动订购引导条目，以使新部署在列表中第一个。（可选）您可以使用键盘中的箭头键选择 GRUB 菜单条目并按 **Enter**。
12. 使用您的帐户凭据登录。
13. 验证 OSTree 状态：

```
$ rpm-ostree status
```

命令输出提供活动的提交 ID。

14. 要查看更改的软件包（如果有），请在父提交和新提交之间运行差异：

```
$ rpm-ostree db diff parent_commit new_commit
```

更新显示您已安装的软件包可用并可供使用。

### 15.2.5. 为非网络部署手动部署 RHEL for Edge 镜像更新

编辑 RHEL for Edge 蓝图后，您可以使用这些更新更新 RHEL for Edge Commit 镜像。使用 RHEL 镜像构建器产生一个新提交来更新已在虚拟机中部署的 RHEL for Edge 镜像，例如：使用此新提交来部署具有最新软件包版本或附加软件包的镜像。

要部署 RHEL for Edge 镜像更新，请确保您满足先决条件，然后按照以下步骤操作。

#### 先决条件

- 在您的主机上，您已在浏览器中从 web 控制台打开了 RHEL 镜像构建器应用程序。
- 您已置备了一个启动并正在运行的 RHEL for Edge 系统。
- 您有一个通过 HTTP 提供服务的 OSTree 存储库。
- 您已编辑了之前创建的 RHEL for Edge 镜像蓝图。

#### 流程

1. 在系统主机上，在 RHEL 镜像构建器仪表盘上，单击 **Create Image**。
2. 在 **Create Image** 窗口中，执行以下步骤：
  - a. 在 **Image output** 页面中：
    - i. 从 **Select a blueprint** 下拉列表中，选择您编辑的蓝图。
    - ii. 从 **Image output type** 下拉列表中，选择 **RHEL for Edge Container (.tar)**。
    - iii. 点 **Next**。
  - b. 在 **OSTree settings** 页面中，输入：
    - i. 在 **Repository URL** 字段中，输入要嵌入到镜像中的提交的 OSTree 存储库的 URL。例如：<http://10.0.2.2:8080/repo/>。请参阅 [设置 web 服务器以安装 RHEL for Edge 镜像](#)。

- ii. 在 **Parent commit** 字段中，指定之前生成的父提交 ID。请参阅 [提取 RHEL for Edge 镜像提交](#)
  - iii. 在 **Ref** 字段中，您可以为提交指定名称或将其留空。默认情况下，web 控制台将 **Ref** 指定为 **rhel/9/arch\_name/edge**。
  - iv. 点 **Next**。
- c. 在 **Review** 页面中，检查自定义并点 **Create**。  
RHEL 镜像构建器为更新的蓝图创建一个 RHEL for Edge 镜像。
  - d. 点 **Images** 选项卡来查看 RHEL for Edge 镜像创建的进度。

**注意**

完成镜像创建过程需要几分钟时间。

生成的镜像包含您添加的最新软件包（若有），并且具有原始 **提交 ID** 作为父项。

3. 在主机上下载生成的 RHEL for Edge 镜像：
  - a. 在 **Images** 选项卡中，点 **Download** 将 RHEL for Edge Container (**.tar**) 镜像保存到主机系统上。
4. 在从原始边缘镜像置备的 RHEL 系统上，执行以下步骤：
  - a. 将 RHEL for Edge 容器镜像加载到 Podman 中，这一次提供子提交 ID。

```
$ cat ./child-commit_ID-container.tar | sudo podman load
```

- b. 运行 **Podman**。

```
# sudo podman run -p 8080:8080 localhost/edge-test
```

- c. 升级 OSTree 存储库：

```
# ostree --repo=/var/www/html/repo pull-local /tmp/ostree-commit/repo
# ostree --repo=/var/www/html/repo summary -u
```

- d. 在置备的 RHEL 系统上，从原始边缘镜像验证当前状态。

```
$ rpm-ostree status
```

如果没有新的提交 ID，请运行以下命令验证是否有可用的升级：

```
$ rpm-ostree upgrade --check
```

如果有可用的更新，命令输出提供关于 OSTree 存储库中可用更新的信息，如当前活动的 OSTree 提交 ID。否则，它会提示一条信息通知没有可用的更新。

- e. 更新 OSTree，使新 OSTree 提交 ID 可用。

```
$ rpm-ostree upgrade
```

ostree 验证存储库是否有更新。如果是，它将获取您重新引导系统的更新和请求，以便您可以激活此新提交更新的部署。

- f. 检查当前系统状态：

```
$ rpm-ostree status
```

现在，您可以看到有 2 个提交可用：

- 活跃的父级提交。
- 一个未激活且包含 1 个添加的差异的新提交。

- g. 要激活新部署并使新提交处于活动状态，请重启您的系统。

```
# systemctl reboot
```

Anaconda 安装程序将重新引导至新部署。在登录屏幕上，您可以看到可供您引导的新部署。

- h. 要引导到最新的提交，请运行以下命令来自动排序引导条目，以便新部署是列表中的第一个：

```
$ rpm-ostree upgrade
```

(可选) 您可以使用键盘中的箭头键选择 GRUB 菜单条目并按 **Enter**。

5. 提供您的登录用户帐户凭证。

6. 验证 OSTree 状态：

```
$ rpm-ostree status
```

命令输出提供活动的提交 ID。

7. 要查看更改的软件包（如果有），请在父提交和新提交之间运行差异：

```
$ rpm-ostree db diff parent_commit new_commit
```

更新显示您已安装的软件包可用并可供使用。

## 15.3. 升级 RHEL FOR EDGE 系统

### 15.3.1. 将 RHEL 8 系统升级到 RHEL 9

您可以使用 **rpm-ostree rebase** 命令将 RHEL 8 系统升级到 RHEL 9。命令完全支持从 RHEL 8 的最新更新升级到 RHEL 9 的最新更新的 RHEL for Edge 的默认软件包集合。升级会在后台下载并安装 RHEL 9 镜像。升级完成后，您必须重启系统以使用新的 RHEL 9 镜像。



## 警告

升级并不支持所有可能的 **rpm** 软件包版本并包括所有软件包。您必须测试添加的软件包，以确保这些软件包可以按预期工作。

## 先决条件

- 您有一个正在运行的 RHEL for Edge 8 系统
- 您有一个 OSTree 软件仓库服务器 (HTTP)
- 您为您要升级的 RHEL for Edge 9 镜像创建了蓝图

## 流程

1. 在 RHEL 镜像构建器运行的系统上，创建一个 RHEL for Edge 9 镜像：

- a. 启动镜像合成：

```
$ sudo composer-cli compose start blueprint-name edge-commit
```

另外，您可以使用以下命令，使用预先存在的 OSTree 存储库创建新的 RHEL for Edge 9 镜像：

```
$ sudo composer-cli compose start-ostree --ref rhel/8/x86_64/edge --parent parent-OSTree-REF --url URL blueprint-name edge-commit
```

- b. 完成合成后，下载镜像。
- c. 将下载的镜像提取到 `/var/www/html/` 文件夹：

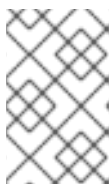
```
$ sudo tar -xf image_file -C /var/www/html
```

- d. 启动 **httpd** 服务：

```
$ systemctl start httpd.service
```

2. 在 RHEL for Edge 设备上，检查当前的远程存储库配置：

```
$ sudo cat /etc/ostree/remotes.d/edge.conf
```



## 注意

根据 Kickstart 文件的配置方式，`/etc/ostree/remotes.d` 存储库可以为空。如果您配置了远程存储库，则您可以看到其配置。对于 **edge-installer**、**raw-image** 和 **simplified-installer** 镜像，默认配置了远程。

3. 检查当前的 URL 存储库：



```
$ sudo ostree remote show-url edge
```

*edge* 是 Ostree 存储库。

- 列出远程引用分支：

```
$ ostree remote refs edge
```

您可以看到以下输出：

```
Error: Remote refs not available; server has no summary file
```

- 添加新存储库：

- 配置 URL 密钥以添加远程存储库。例如：

```
$ sudo ostree remote add \ --no-gpg-verify rhel9 http://192.168.122.1/repo/
```

- 将 URL 键配置为指向升级的 RHEL 9 提交。例如：

```
$ sudo cat /etc/ostree/remotes.d/edge.conf

[remote "edge"]
url=http://192.168.122.1/ostree/repo/
gpg-verify=false
```

- 确认 URL 是否已设置为新的远程存储库：

```
$ sudo cat /etc/ostree/remotes.d/rhel9.conf

[remote "edge"]
url=http://192.168.122.1/repo/
gpg-verify=false
```

- 查看新 URL 存储库：

```
$ sudo ostree remote show-url rhel9 http://192.168.122.1/ostree-rhel9/repo/
```

- 列出当前的远程列表选项：

```
$ sudo ostree remote list

output:
edge
rhel9
```

- 将您的系统升级到 RHEL 版本，为 RHEL 9 版本提供参考路径：

```
$ rpm-ostree rebase rhel9:rhel/9/x86_64/edge
```

- 重启您的系统。

```
$ systemctl reboot
```

8. 输入您的用户名和密码。
9. 检查当前系统状态：

```
$ rpm-ostree status
```

## 验证

1. 检查当前运行的部署的当前状态：

```
$ rpm-ostree status
```

2. 可选：列出内核实时管理的处理器和任务。

```
$ top
```

3. 如果升级不支持您的要求，您可以选择手动回滚到以前的稳定部署 RHEL 8 版本：

```
$ sudo rpm-ostree rollback
```

4. 重启您的系统。输入您的用户名和密码：

```
$ systemctl reboot
```

重新引导后，您的系统成功运行了 RHEL 9。



### 注意

如果升级成功，且您不想使用以前的部署 RHEL 8 版本，您可以删除旧软件仓库：

```
$ sudo ostree remote delete edge
```

## 其他资源

- [rpm-ostree 更新和 rebase 失败，且无法找到内核错误](#)

## 15.4. 部署 RHEL FOR EDGE 自动镜像更新

在 Edge 设备中安装 RHEL for Edge 镜像后，您可以检查可用的镜像更新（如果有）并可自动应用。

**rpm-ostreed-automatic.service** (systemd 服务)和 **rpm-ostreed-automatic.timer** (systemd 计时器)控制检查和升级的频率。可用的更新（若有）显示为暂存部署。

部署自动镜像更新涉及以下高级别步骤：

- 更新镜像更新策略
- 启用自动下载和暂存更新

### 15.4.1. 更新 RHEL for Edge 镜像更新策略

要更新镜像更新策略，请使用边缘设备上位于 `/etc/rpm-ostreed.conf` 处的 `rpm-ostreed.conf` 文件中的 `AutomaticUpdatePolicy` 和 `IdleExitTimeout` 设置。

`AutomaticUpdatePolicy` 设置控制自动更新策略，并有以下更新检查选项：

- **none**: 禁用自动更新。默认情况下，`AutomaticUpdatePolicy` 设置被设为 **none**。
- **check** : 下载足够的元数据以显示具有 `rpm-ostree` 状态的可用更新。
- **stage** : 下载并解压缩重启时应用的更新。

`IdleExitTimeout` 设置控制守护进程退出前不活跃的时间，并具有以下选项：

- 0: 禁用自动退出。
- 60: 默认情况下，`IdleExitTimeout` 设置被设置为 **60**。

要启用自动更新，请执行以下步骤：

#### 流程

1. 在 `/etc/rpm-ostreed.conf` 文件中更新以下内容：
  - 把 `AutomaticUpdatePolicy` 的值改为 **check**。
  - 要运行更新检查，请为 `IdleExitTimeout` 指定一个以秒为单位的值。

2. 重新加载 `rpm-ostreed` 服务并启用 `systemd` 定时器。

```
# systemctl reload rpm-ostreed
# systemctl enable rpm-ostreed-automatic.timer --now
```

3. 验证 `rpm-ostree` 状态，以确保配置了自动更新策略，并且时间处于活跃状态。

```
# rpm-ostree status
```

命令输出显示以下内容：

```
State: idle; auto updates enabled (check; last run <minutes> ago)
```

此外，输出中也显示有关可用更新的信息。

### 15.4.2. 启用 RHEL for Edge 自动下载和保存更新

在更新了镜像更新策略以检查镜像更新后，如果显示了任何更新详情，则进行更新。如果您决定应用更新，请启用策略来自动下载和暂存更新。然后，下载并暂存可用的镜像更新以进行部署。更新会被应用并在重启 Edge 设备时生效。

要启用自动下载和暂存更新的策略，请执行以下操作：

#### 流程

1. 在 `/etc/rpm-ostreed.conf` 文件中，将 "AutomaticUpdatePolicy" 更新为 **stage**。

2. 重新载入 **rpm-ostreed** 服务。

```
# systemctl enable rpm-ostreed-automatic.timer --now
```

3. 验证 **rpm-ostree** 状态

```
# rpm-ostree status
```

命令输出显示以下内容：

```
State: idle
AutomaticUpdates: stage; rpm-ostreed-automatic.timer: last run <time> ago
```

4. 要启动更新，您可以等待计时器启动更新，也可以手动启动该服务。

```
# systemctl start rpm-ostreed-automatic.service
```

启动更新后，**rpm-ostree** 状态显示如下：

```
# rpm-ostree status
State: busy
AutomaticUpdates: stage; rpm-ostreed-automatic.service: running
Transaction: automatic (stage)
```

更新完成后，部署列表中会暂存新的部署，原始引导的部署将保持不变。您可以决定您是否要使用新部署引导系统，或者可以等待下一次更新。

要查看部署列表，请运行 **rpm-ostree status** 命令。

以下是输出示例：

```
# rpm-ostree status
State: idle
AutomaticUpdates: stage; rpm-ostreed-automatic.timer: last run <time> ago
Deployments:
```

要使用更新的软件包详情查看部署列表，请运行 **rpm-ostree status -v** 命令。

## 15.5. RHEL FOR EDGE 镜像的回滚

因为 RHEL for Edge 对操作系统应用事务更新，所以您可以手动或自动将更新回滚到最后已知的良好状态，这可防止系统在更新过程中失败。您可以使用 **greenboot** 框架自动化验证和回滚过程。

**greenboot** 健康检查框架利用 **rpm-ostree** 来在系统启动时运行自定义健康检查。如果有问题，系统会回滚到最后一个工作状态。当您部署 **rpm-ostree** 更新时，它会运行脚本来检查关键服务在更新后是否仍然可以正常工作。如果系统无法正常工作，例如因为一些失败的软件包，则您可以将系统回滚到系统以前的稳定版本。此过程可确保您的 RHEL for Edge 设备处于可操作状态。

更新镜像后，它会创建一个新镜像部署，同时保留以前的镜像部署。您可以验证更新是否成功。如果更新失败，例如因为软件包失败，您可以将系统回滚到以前的稳定版本。

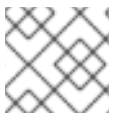
### 15.5.1. 介绍 **greenboot** 检查

**Greenboot** 是基于 **rpm-ostree** 的系统上提供的 **systemd** 的通用健康检查框架。它包含您可以在系统上安装的以下 RPM 软件包：

- **greenboot** - 一个包含以下功能的软件包：
  - 检查提供的脚本
  - 如果检查失败，则重启系统
  - 重启无法解决问题，回滚到以前的部署。
- **greenboot-default-health-checks** - 由 **greenboot** 系统维护者提供的一组可选和选定的健康检查。

**Greenboot** 通过使用运行在系统上的健康检查脚本对系统健康状况进行评估来在 RHEL for Edge 系统中工作，并在某些软件失败时自动回滚到最后的健康状态。这些健康检查脚本位于 `/etc/greenboot/check/required.d` 目录中。**Greenboot** 支持用于健康检查的 shell 脚本。当您需要检查软件问题并在直接可用性有限或不存在的边缘设备上执行系统回滚时，有一个健康检查框架特别有用。当您安装和配置健康检查脚本时，它会在每次系统启动时触发运行健康检查。

您可以创建自己的健康检查脚本，来评估工作负载和应用程序的健康状态。这些额外的健康检查脚本是软件问题检查和自动系统回滚的有用组件。



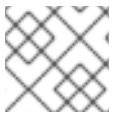
#### 注意

在任何健康检查在不使用 OSTree 的系统上失败时，您不能使用回滚。

### 15.5.2. RHEL for Edge 镜像使用 **greenboot** 进行回滚

对于 RHEL for Edge 镜像，仅将事务更新应用到操作系统。事务更新是原子的，这意味着只有在所有更新都成功并且支持回滚时，才会应用更新。通过事务更新，您可以轻松地将失败的更新回滚到最后已知的良好状态，从而防止更新期间系统失败。

当您需要检查软件问题并在直接服务有限或不存在的边缘设备上执行系统回滚时，执行健康检查特别有用。



#### 注意

在不使用 OSTree 的系统上更新失败时，您无法使用回滚，即使健康检查可以运行。

您可以将智能回滚与 **greenboot** 健康检查框架一起使用，来在系统启动时自动评估系统健康状况。您可以从 **greenboot-default-health-checks** 子软件包中获取预配置的健康检查。这些检查位于 **rpm-ostree** 系统的 `/usr/lib/greenboot/check` 只读目录中。

**Greenboot** 利用 **rpm-ostree**，并在系统启动时运行自定义健康检查。如果出现问题，系统会回滚更改，并保留最后一个工作状态。当您部署 **rpm-ostree** 更新时，它会运行脚本来检查关键服务在更新后是否仍然可以工作。如果系统无法正常工作，更新会回滚到系统的最后一个已知工作版本。此过程可确保您的 RHEL for Edge 设备处于可操作状态。

您可以从 **greenboot-default-health-checks** 的子软件包中获取预配置的健康检查。这些检查位于 **rpm-ostree** 系统的 `/usr/lib/greenboot/check` 只读目录中。您还可以将 shell 脚本配置为以下类型的检查：

#### 例 15.1. **greenboot** 目录结构

etc

```

├─ greenboot
│  └─ check
│     └─ required.d
│     └─ init.py
│     └─ green.d
│     └─ red.d

```

## 必需

包含不能失败的健康检查。将所需的 shell 脚本放在 `/etc/greenboot/check/required.d` 目录中。如果脚本失败，`greenboot` 会默认重试三次。您可以通过将 `GREENBOOT_MAX_BOOTS` 参数设置为您想要的重试次数，来在 `/etc/greenboot/greenboot.conf` 文件中配置重试次数。

当所有重试失败后，`greenboot` 会在有可用回滚时自动启动回滚。如果没有回滚，系统日志输出会显示您需要执行人工干预。

## 想要

包含可能失败的健康检查，而不会导致系统回滚。将所需的 shell 脚本放在 `/etc/greenboot/check/wanted.d` 目录中。`greenboot` 告知脚本失败，系统健康状态保持不变，且不会执行回滚或重启。

您还可以指定在检查后将运行的 shell 脚本：

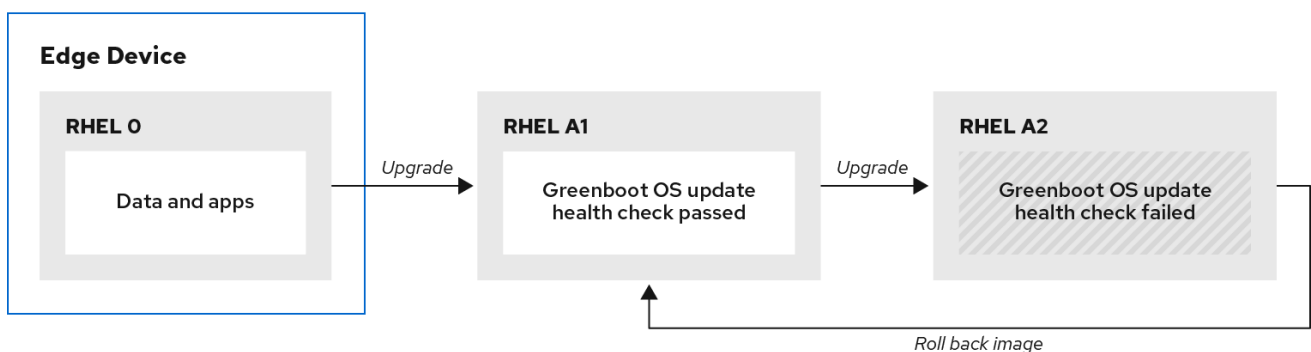
## 绿色

包含成功引导后要运行的脚本。将这些脚本放在 `/etc/greenboot/green.d` 目录中。`greenboot` 告知引导成功。

## 红色

包含在引导失败后要运行的脚本。将这些脚本放在 `/etc/greenboot/red.d` 目录中。系统会尝试引导三次，并在失败时执行脚本。`greenboot` 告知引导失败。

下图演示了 RHEL for Edge 镜像回滚过程。



125\_RHEL\_1020

引导更新的操作系统后，`greenboot` 会运行 `required.d` 和 `wanted.d` 目录中的脚本。如果 `required.d` 目录中的任何脚本失败，则 `greenboot` 会运行 `red.d` 目录中的任何脚本，然后重启系统。

`Greenboot` 在升级的系统上尝试进行 2 次引导。如果在第三次尝试引导时 `required.d` 中的脚本仍然失败，`greenboot` 会最后一次运行 `red.d` 脚本，以确保 `red.d` 目录中的脚本尝试进行正确的操作来修复问题，且没有成功。然后，`greenboot` 将系统从当前的 `rpm-ostree` 部署回滚到以前的稳定部署。

### 15.5.3. Greenboot 健康检查状态

在部署更新的系统时，请等待 greenboot 健康检查完成，然后再进行更改，以确保如果 greenboot 将系统回滚回较早状态时，这些更改不会丢失。如果要进行配置更改或部署应用程序，您必须等待 greenboot 健康检查完成。这样可确保，如果 greenboot 将 **rpm-ostree** 系统回滚回较早状态，您的更改不会丢失。

**greenboot-healthcheck** 服务运行一次，然后退出。您可以使用以下命令检查服务的状态，以了解其是否已完成，并知道结果：

#### **systemctl is-active greenboot-healthcheck.service**

在服务已退出时，此命令报告 **active**。如果服务根本没有运行，它显示 **inactive**。

#### **systemctl show --property=SubState --value greenboot-healthcheck.service**

完成后报告 **exited**，仍在运行时报告 **running**。

#### **systemctl show --property=Result --value greenboot-healthcheck.service**

在检查通过时报告 **success**。

#### **systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service**

报告服务的数字退出码，0 表示成功，非零值表示发生了故障。

#### **cat /run/motd.d/boot-status**

显示一条消息，如 "Boot Status is GREEN - Health Check SUCCESS"。

### 15.5.4. 检查 greenboot 健康检查状态

在对系统进行更改或故障排除期间，检查 greenboot 健康检查的状态。您可以使用以下任一命令来帮助确保 greenboot 脚本已经完成运行。

- 使用以下选项之一检查状态：

- 要查看健康检查状态的报告，请输入：

```
$ systemctl show --property=SubState --value greenboot-healthcheck.service
```

以下输出是可能的：

- **start** 表示 greenboot 检查仍在运行。
- **exited** 表示检查已通过，并且 greenboot 已退出。当系统处于健康状态时，greenboot 运行 **green.d** 目录中的脚本。
- **failed** 表示检查没有通过。当系统处于此状态时，greenboot 运行 **red.d** 目录中的脚本，并可能重启系统。
- 要查看显示服务的数字退出码的报告，其中 **0** 表示成功，非零值表示发生了故障，请使用以下命令：

```
$ systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service
```

- 要查看显示有关引导状态消息的报告，如 **Boot Status** 为 **GREEN - Health Check SUCCESS**，请输入：

```
$ cat /run/motd.d/boot-status
```

### 15.5.5. 手动回滚 RHEL for Edge 镜像

升级操作系统时，会创建一个新的部署，**rpm-ostree** 软件包也会保持以前的部署。如果操作系统更新版本中有问题，您可以使用单个 **rpm-ostree** 命令手动回滚到以前的部署，或者在 GRUB 引导装载程序中选择以前的部署。

要手动回滚到以前的版本，请执行以下步骤。

### 前提条件

1. 您已更新了您的系统，但失败了。

### 流程

1. 可选：检查失败的错误消息：

```
$ journalctl -u greenboot-healthcheck.service.
```

2. 运行 **rollback** 命令：

```
# rpm-ostree rollback
```

命令输出提供有关正在移动的提交 ID 的详细信息，并指示与正在删除的软件包的详细信息相关的已完成事务。

3. 重启系统。

```
# systemctl reboot
```

命令将激活上一个带有稳定内容的提交。应用更改并恢复之前的版本。

## 15.5.6. 使用自动化流程回滚 RHEL for Edge 镜像

**Greenboot** 检查提供了一个集成到引导过程的框架，并在健康检查失败时触发 **rpm-ostree** 回滚。对于健康检查，您可以创建一个自定义脚本来指示健康检查是否通过或失败。根据结果，您可以决定何时触发回滚。以下流程演示了如何创建一个健康检查脚本的示例：

### 流程

1. 创建返回标准退出代码 **0** 的脚本。  
例如，以下脚本确保配置的 DNS 服务器可用：

```
#!/bin/bash

DNS_SERVER=$(grep ^nameserver /etc/resolv.conf | head -n 1 | cut -f2 -d" ")
COUNT=0
# check DNS server is available
ping -c1 $DNS_SERVER
while [ $? != '0' ] && [ $COUNT -lt 10 ]; do
  ((COUNT++))
  echo "Checking for DNS: Attempt $COUNT ."
  sleep 10
  ping -c 1 $DNS_SERVER
done
```

2. 在 **/etc/greenboot/check/required.d/** 中包括健康检查的可执行文件。



```
chmod +x check-dns.sh
```

在下次重启过程中，在系统进入 **boot-complete.target** 单元前，会作为引导过程的一部分来执行脚本。如果健康检查成功，则不执行任何操作。如果健康检查失败，则系统会重启多次，然后再将更新标记为失败，并回滚到之前的更新。

## 验证步骤

要检查默认网关是否可访问，请运行以下健康检查脚本：

1. 创建返回标准退出代码 **0** 的脚本。

```
#!/bin/bash

DEF_GW=$(ip r | awk '/^default/ {print $3}')
SCRIPT=$(basename $0)

count=10
connected=0
ping_timeout=5
interval=5

while [ $count -gt 0 -a $connected -eq 0 ]; do
  echo "$SCRIPT: Pinging default gateway $DEF_GW"
  ping -c 1 -q -W $ping_timeout $DEF_GW > /dev/null 2>&1 && connected=1 || sleep
  $interval
  ((--count))
done

if [ $connected -eq 1 ]; then
  echo "$SCRIPT: Default gateway $DEF_GW is reachable."
  exit 0
else
  echo "$SCRIPT: Failed to ping default gateway $DEF_GW!" 1>&2
  exit 1
fi
```

2. 在 **/etc/greenboot/check/required.d/** 目录中包括健康检查的可执行文件。

```
chmod +x check-gw.sh
```

## 第 16 章 创建并管理 OSTREE 镜像更新

您可以轻松为 RHEL for Edge 系统创建和管理 OSTree 镜像更新，并使其立即用于 RHEL for Edge 设备。使用 OSTree，您可以使用镜像构建器将 RHEL for Edge Commit 或 RHEL for Edge 容器镜像创建为包含 OSTree 提交的 **.tar** 文件。OSTree 更新版本系统作为一个"Git 存储库"，来存储和版本化 OSTree 提交。**rpm-ostree** 镜像和软件包系统随后在客户端设备上组装提交。当您使用 RHEL 镜像构建器创建新镜像来执行更新时，RHEL 镜像构建器会从这些存储库中拉取更新。

### 16.1. OSTREE 的基本概念

OSTree 和 **rpm-ostree** 在镜像更新过程中使用的基本术语。

#### rpm-ostree

在边缘设备上处理 OSTree 提交是如何在设备上组装的技术。它作为镜像和软件包系统之间的混合使用。使用 **rpm-ostree** 技术，您可以对您的系统进行原子升级和回滚。

#### OSTree

ostree 是一种技术，可让您创建提交并下载可引导的文件系统树。您还可以使用它来部署树，并管理引导装载程序配置。

#### Commit

OSTree 提交包含不能直接启动的完整的操作系统。要引导系统，您必须使用 RHEL 可安装镜像部署它。

#### 参考

它也被称为 **ref**。OSTree ref 类似于 Git 分支，它是一个名称。以下引用名称示例是有效的：

- **rhel/9/x86\_64/edge**
- **ref-name**
- **app/org.gnome.Calculator/x86\_64/stable**
- **ref-name-2**

默认情况下，镜像构建器将 **rhel/9/\$ARCH/edge** 指定为路径。"\$ARCH" 值由主机机器决定。

#### 父

**parent** 参数是一个 OSTree 提交，您可以提供它来使用镜像构建器构建一个新提交。您可以使用 **parent** 参数指定一个现有的 **ref**，该 **ref** 为您要构建的新提交检索父提交。您必须将父提交指定为要解析和拉取的 **ref** 值，如 **rhel/9/x86\_64/edge**。您可以将 **--parent** 提交用于 RHEL for Edge Commit (**.tar**)和 RHEL for Edge Container (**.tar**)镜像类型。

#### 远程

承载 OSTree 内容的 http 或 https 端点。这与 yum 存储库的 **baseurl** 类似。

#### 静态 delta

静态 deltas 是在两个 OSTree 提交之间产成的更新集合。这可让系统客户端获取较小数量的文件，这些文件很大。静态 deltas 更新的网络效率更高，因为在更新基于 ostree 的主机时，系统客户端只会从系统中不存在的新 OSTree 提交中获取对象。通常，新的 OSTree 提交包含许多小文件，这需要多个 TCP 连接。

#### 概述

摘要文件是一个简洁的方式，枚举 **refs**、**checksums** 和 OSTree 存储库中可用的静态 deltas。您可以检查 Ostree 仓库中提供的所有 **refs** 和静态 deltas 的状态。但是，在每次将新的 **ref**、**commit** 或 **static-delta** 添加至 OSTree 仓库时，您必须生成摘要文件。

## 16.2. 创建 OSTREE 存储库

您可以使用 **RHEL for Edge Commit (.tar)** 或 **RHEL for Edge Container (.tar)** 镜像类型，使用 RHEL 镜像构建器创建 OSTree 存储库。这些镜像类型包含一个 OSTree 存储库，其包含一个 OSTree 提交。

- 您可以在 web 服务器上提取 **RHEL for Edge Commit (.tar)**，其已准备好提供服务。
- 您必须将 **RHEL for Edge Container (.tar)** 导入到本地容器镜像存储中，或将镜像推送到容器注册中心。启动容器后，它通过集成的 **nginx** web 服务器为提交提供服务。

在带有 Podman 的 RHEL 服务器上使用 **RHEL for Edge Container (.tar)** 来创建一个 OSTree 存储库：

### 前提条件

- 您创建了一个 **RHEL for Edge Container (.tar)** 镜像。

### 流程

1. 从镜像构建器下载容器镜像：

```
$ composer-cli compose image _<UUID>
```

2. 将容器导入到 Podman：

```
$ skopeo copy oci-archive:_<UUID>_container.tar containers-storage:localhost/ostree
```

3. 启动容器并使用端口 **8080** 使其可用：

```
$ podman run -rm -p 8080:8080 ostree
```

### 验证

- 检查容器是否正在运行：

```
$ podman ps -a
```

## 16.3. 管理集中式 OSTREE 镜像

对于生产环境，有一个提供所有提交的集中式 OSTree 镜像具有多个优点，包括：

- 去重和最小化磁盘存储
- 使用静态 delta 更新优化对客户端的更新
- 为部署生命周期指向一个 OSTree 镜像。

要管理集中式 OSTree 镜像，您必须将镜像构建器中的每个提交拉取到可供您的用户使用的集中式存储库中。



### 注意

您还可以使用 **osbuild.infra** Ansible 集合自动管理 OSTree 镜像。请参阅 [osbuild.infra Ansible](https://github.com/osbuild/ansible)。

要创建一个集中式存储库，您可以直接在 web 服务器上运行以下命令：

## 流程

1. 创建一个空蓝图，对其定制为使用 "rhel-92" 作为发行版：

```
name = "minimal-rhel92"
description = "minimal blueprint for ostree commit"
version = "1.0.0"
modules = []
groups = []
distro = "rhel-92"
```

2. 将蓝图推送到服务器：

```
# composer-cli blueprints push minimal-rhel92.toml
```

3. 从您创建的蓝图中构建 RHEL for Edge Commit (**.tar**) 镜像：

```
# composer-cli compose start-ostree minimal-rhel92 edge-commit
```

4. 检索 **.tar criule**，并将其解压缩到磁盘：

```
# composer-cli compose image _<rhel-92-uuid>
$ tar -xf <rhel-92-uuid>.tar -C /usr/share/nginx/html/
```

磁盘上的 **/usr/share/nginx/html/repo** 位置将成为所有 refs 和 commit 的一个 OSTree 存储库。

5. 创建另一个空蓝图，对其定制为使用 "rhel-87" 作为发行版：

```
name = "minimal-rhel87"
description = "minimal blueprint for ostree commit"
version = "1.0.0"
modules = []
groups = []
distro = "rhel-87"
```

6. 推送蓝图，并创建另一个 RHEL for Edge Commit (**.tar**) 镜像：

```
# composer-cli blueprints push minimal-rhel87.toml
# composer-cli compose start-ostree minimal-rhel87 edge-commit
```

7. 检索 **.tar criule**，并将其解压缩到磁盘：

```
# composer-cli compose image <rhel-87-uuid>
$ tar -xf <rhel-87-uuid>.tar
```

8. 将提交拉取到本地存储库。通过使用 **ostree pull-local**，您可以将提交数据从一个本地存储库复制到另一个本地存储库。

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo
```

9. 可选：检查 OSTree 存储库的状态。以下是一个输出示例：

```
$ ostree --repo=/usr/share/nginx/html/repo refs
```

```
rhel/8/x86_64/edge
```

```
rhel/9/x86_64/edge
```

```
$ ostree --repo=/usr/share/nginx/html/repo show rhel/8/x86_64/edge
```

```
commit f7d4d95465fbd875f6358141f39d0c573df6a321627bafde68c73850667e5443
```

```
ContentChecksum:
```

```
41bf2f8b442a770e9bf03e096a46a286f5836e0a0702b7c3516ef4e0acec2dea
```

```
Date: 2023-09-15 16:17:04 +0000
```

```
Version: 8.7
```

```
(no subject)
```

```
$ ostree --repo=/usr/share/nginx/html/repo show rhel/9/x86_64/edge
```

```
commit 89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
```

```
ContentChecksum:
```

```
70235bfb9cae82c53f856183750e809becf0b9b076122b19c40fec92fc6d74c1
```

```
Date: 2023-09-15 15:30:24 +0000
```

```
Version: 9.2
```

```
(no subject)
```

10. 更新 RHEL 9.2 蓝图，以使其包含新软件包并构建新提交，例如：

```
name = "minimal-rhel92"
```

```
description = "minimal blueprint for ostree commit"
```

```
version = "1.1.0"
```

```
modules = []
```

```
groups = []
```

```
distro = "rhel-92"
```

```
[[packages]]
```

```
name = "strace"
```

```
version = ""
```

11. 推送更新的蓝图并创建一个新的 RHEL for Edge Commit (**.tar**) 镜像，将 compose 指向现有的 OSTree 存储库：

```
# composer-cli blueprints push minimal-rhel92.toml
```

```
# composer-cli compose start-ostree minimal-rhel92 edge-commit --url http://localhost/repo --
```

```
ref rhel/9/x86_64/edge
```

12. 检索 **.tar** 文件，并将其解压缩到磁盘上：

```
# rm -rf repo
```

```
# composer-cli compose image <rhel-92-uuid>
```

```
# tar -xf <rhel-92-uuid>.tar
```

13. 将提交拉取到存储库：

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo
```

14. 可选：再次检查 OSTree 存储库状态：

```
$ ostree --repo=/usr/share/nginx/html/repo refs
```

```
rhel/8/x86_64/edge
```

```
rhel/9/x86_64/edge
```

```
$ ostree --repo=/usr/share/nginx/html/repo show rhel/8/x86_64/edge
commit f7d4d95465fbd875f6358141f39d0c573df6a321627bafde68c73850667e5443
ContentChecksum:
41bf2f8b442a770e9bf03e096a46a286f5836e0a0702b7c3516ef4e0acec2dea
Date: 2023-09-15 16:17:04 +0000
Version: 8.7
(no subject)
```

```
$ ostree --repo=/usr/share/nginx/html/repo show rhel/9/x86_64/edge
commit a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
Parent: 89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ContentChecksum:
2335930df6551bf7808e49f8b35c45e3aa2a11a6c84d988623fd3f36df42a1f1
Date: 2023-09-15 18:21:31 +0000
Version: 9.2
(no subject)
```

```
$ ostree --repo=/usr/share/nginx/html/repo log rhel/9/x86_64/edge
commit a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
Parent: 89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ContentChecksum:
2335930df6551bf7808e49f8b35c45e3aa2a11a6c84d988623fd3f36df42a1f1
Date: 2023-09-15 18:21:31 +0000
Version: 9.2
(no subject)
```

```
commit 89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ContentChecksum:
70235bfb9cae82c53f856183750e809becf0b9b076122b19c40fec92fc6d74c1
Date: 2023-09-15 15:30:24 +0000
Version: 9.2
(no subject)
```

```
$ rpm-ostree db diff --repo=/usr/share/nginx/html/repo
89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
ostree diff commit from:
89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ostree diff commit to:
a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
Added:
elfutils-default-yama-scope-0.188-3.el9.noarch
elfutils-libs-0.188-3.el9.x86_64
strace-5.18-2.el9.x86_64
```

## 附录 A. 术语和命令

了解有关 **rpm ostree** 术语和命令的更多信息。

### A.1. OSTREE 和 RPM-OSTREE 术语

以下是一些在上下文与 OSTree 和 **rpm-ostree** 镜像有关的术语。

表 A.1. OSTree 和 rpm-ostree 术语

术语	定义
<b>OSTree</b>	此工具用于管理基于 Linux 的操作系统版本。OSTree 树视图与 Git 类似，它基于相似的概念。
<b>rpm-ostree</b>	托管操作系统更新的混合镜像或系统软件包。
<b>Commit</b>	操作系统的发行版本或镜像版本。RHEL 镜像构建器为 RHEL for Edge 镜像生成一个 OSTree 提交。您可以使用这些镜像在 Edge 服务器上安装或更新 RHEL。
<b>Refs</b>	代表 OSTree 中的一个分支。Refs 始终解析为最新的提交。例如： <b>rhel/9/x86_64/edge</b> 。
<b>修订 (Rev)</b>	特定提交的 SHA-256。
<b>远程</b>	承载 OSTree 内容的 http 或 https 端点。这类似于 dnf 存储库的 baseurl。
<b>static-delta</b>	对 OSTree 镜像的更新始终是 delta 更新。如果是 RHEL for Edge 镜像，则 TCP 开销可能高于预期值，因为更新了文件的数量。为避免 TCP 开销，您可以在特定提交之间生成 static-delta，并在单个连接中发送更新。这种优化有助于连接受限的大型部署。

### A.2. OSTREE 命令

下表提供了几个您可以在安装或管理 OSTree 镜像时使用的 OSTree 命令，。

表 A.2. ostree 命令

ostree pull	<b>ostree pull-local --repo [path] src</b> <b>ostree pull-local &lt;path&gt; &lt;rev&gt; --repo=&lt;repo-path&gt;</b> <b>ostree pull &lt;URL&gt; &lt;rev&gt; --repo=&lt;repo-path&gt;</b>
ostree 概况	<b>ostree summary -u --repo=&lt;repo-path&gt;</b>

查看 refs	<code>ostree refs --repo ~/Code/src/osbuild-iot/build/repo/ --list</code>
查看 repo 中的提交	<code>ostree log --repo=/home/gicmo/Code/src/osbuild-iot/build/repo/ &lt;REV&gt;</code>
检查一个提交	<code>ostree show --repo build/repo &lt;REV&gt;</code>
列出 repo 的远程	<code>ostree remote list --repo &lt;repo-path&gt;</code>
解析一个 REV	<code>ostree rev-parse --repo ~/Code/src/osbuild-iot/build/repo fedora/x86_64/osbuild-demo</code> <code>ostree rev-parse --repo ~/Code/src/osbuild-iot/build/repo b3a008eceeddd0cfd</code>
创建 static-delta	<code>ostree static-delta generate --repo=[path] --from=REV --to=REV</code>
使用 GPG 密钥签署一个现有的 ostree 提交	<code>ostree gpg-sign --repo=&lt;repo-path&gt; --gpg-homedir &lt;gpg_home&gt; COMMIT KEY-ID...</code>

### A.3. RPM-OSTREE 命令

下表提供了几个您可以在安装或管理 OSTree 镜像时使用的 `rpm-ostree` 命令。

表 A.3. rpm-ostree 命令

命令	描述
<code>rpm-ostree --repo=/home/gicmo/Code/src/osbuild-iot/build/repo/ db list &lt;REV&gt;</code>	此命令会列出 <REV> 提交到存储库中的现有软件包。
<code>rpm-ostree rollback</code>	OSTree 管理引导装载程序条目的有序列表，称为 <b>部署</b> 。index 0 处的条目是默认的引导装载程序条目。每个条目都有一个单独的 <code>/etc</code> 目录，但所有条目共享单个 <code>/var</code> 目录。您可以通过按 Tab 键中断启动，来使用启动加载程序在条目之间进行选择。这会回滚到以前的状态，即默认部署更改在非默认位置。
<code>rpm-ostree status</code>	此命令提供有关当前正在使用的部署的信息。按顺序列出所有可能部署的名称和 <b>refspec</b> ，以便列表中的第一个部署是启动时的默认设置。标记为 * 的部署是当前的引导部署，使用 'r' 标记代表最新的升级。
<code>rpm-ostree db list</code>	使用此命令查看提交或提交中的软件包。您必须至少指定一个提交，但多个提交也起作用。



命令	描述
<b>rpm-ostree db diff</b>	使用此命令显示两个 rev（修订）中的树之间的软件包如何不同。如果没有提供 revs，则引导的提交将与待处理提交进行比较。如果只提供单个 rev，则引导的提交将与该 rev 进行比较。
<b>rpm-ostree upgrade</b>	此命令将下载当前树的最新版本并进行部署，将当前树设置为下一次启动的默认树。这不会影响运行的文件系统树。您必须重启才能使任何更改生效。

### 其他资源

- **rpm-ostree** man page。

## A.4. FDO 自动加入术语

了解有关 FDO 术语的更多信息。

表 A.4. FDO 术语

命令	描述
FDO	FIDO Device Onboarding（FIDO 设备加入）。
设备	任何硬件、设备或计算机。
所有者	设备的最终所有者 - 公司或 IT 部门。
制造商	设备制造商。
制造商服务器	为该设备创建设备凭证。
制造商客户端	告知 manufacturing 服务器的位置。
所有权变量(OV)	<p>单独设备的所有权的记录。</p> <p>包含以下信息：</p> <ul style="list-style-type: none"> <li>* Owner (<b>fdo-owner-onboarding-service</b>)</li> <li>* Rendezvous Server - FIDO server (<b>fdo-rendezvous-server</b>)</li> <li>* Device (at least one combination) (<b>fdo-manufacturing-service</b>)</li> </ul>
设备凭据(DC)	存储在制造商设备的密钥凭证和 rendezvous。

命令	描述
Keys	配置 manufacturing 服务器的密钥  * key_path  * cert_path  * key_type  * mfg_string_type: 设备序列号  * allowed_key_storage_types: 文件系统和受信任的平台模块(TPM), 用于保护用来验证您使用的设备的数据。
Rendezvous 服务器	指向设备使用的服务器及之后使用的服务器, 用于查找该设备的所有者是谁

### 其他资源

- [FIDO IoT 规格](#)

## A.5. FDO 自动加入技术

以下是在上下文中用于 FDO 自动加入的技术。

表 A.5. OSTree 和 rpm-ostree 术语

技术	定义
UEFI	统一可扩展固件接口。
RHEL	Red Hat® Enterprise Linux® operating system
<b>rpm-ostree</b>	基于镜像的后台升级。
Greenboot	<b>rpm-ostree</b> 上的 systemd 的 <b>Healthcheck</b> 框架。
osbuild	用于操作系统工件的基于管道的构建系统。
Container	Linux® 容器是与系统其余部分隔离的一个或多个进程的集合。
Coreos-installer	辅助安装 RHEL 镜像, 使用 UEFI 引导系统。
FIDO FDO	调配配置和加入设备的规格协议。

