Red Hat Enterprise Linux 9

Composing, installing, and managing RHEL for Edge images

Creating, deploying, and managing Edge systems with Red Hat Enterprise Linux 9

Last Updated: 2025-06-13

Red Hat Enterprise Linux 9 Composing, installing, and managing RHEL for Edge images

Creating, deploying, and managing Edge systems with Red Hat Enterprise Linux 9

Legal Notice

Abstract

Use the image builder tool to compose customized RHEL (rpm-ostree) images optimized for Edge. Then, you can remotely install, and securely manage and scale deployments of the images on Edge servers.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. INTRODUCTION TO RHEL FOR EDGE IMAGES

A RHEL for Edge image is an **rpm-ostree** image that includes system packages to remotely install RHEL on Edge servers.

The system packages include:

- **Base OS** package
- Podman as the container engine
- Additional RPM Package Manager (RPM) content

RHEL for Edge is an immutable operating system that contains a **read-only** root directory, and has following characteristics:

- The packages are isolated from the root directory.
- Each version of the operating system is a separate deployment. Therefore, you can roll back the system to a previous deployment when needed.
- The **rpm-ostree** image offers efficient updates over the network.
- RHEL for Edge supports multiple operating system branches and repositories.
- The image contains a hybrid **rpm-ostree** package system.

You can compose customized RHEL for Edge images by using the RHEL image builder tool.

With a RHEL for Edge image, you can achieve the following benefits:

Atomic upgrades

You know the state of each update, and no changes are seen until you reboot your system.

Custom health checks and intelligent rollbacks

You can create custom health checks, and if a health check fails, the operating system rolls back to the previous stable state.

Container-focused workflow

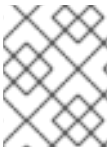
The image updates are staged in the background, minimizing any workload interruptions to the system.

Optimized Over-the-Air updates

You can make sure that your systems are up-to-date, even with intermittent connectivity, thanks to efficient over-the-air (OTA) delta updates.

1.1. RHEL FOR EDGE-SUPPORTED ARCHITECTURE

Currently, you can deploy RHEL for Edge images on AMD and Intel 64-bit systems.



NOTE

RHEL for Edge supports ARM systems on some devices. For more information about the supported devices, see Red Hat certified hardware .

1.2. RHEL FOR EDGE IMAGE TYPES AND THEIR DEPLOYMENTS

Composing and deploying a RHEL for Edge image involves two phases:

1. Composing a RHEL **rpm-ostree** image using the RHEL image builder tool. You can access RHEL image builder on the command line by using the **composer-cli** tool, or use a graphical user interface in the RHEL web console.

2. Deploying the image by using RHEL installer.

The image types vary in terms of their contents, and are therefore suitable for different types of deployment environments. While composing a RHEL for Edge image, you can select any of the following image types:

RHEL for Edge Commit

This image type delivers atomic and safe updates to a system. The **edge-commit**(**.tar**) image contains a full operating system, but it is not directly bootable. To boot the **edge-commit** image type, you must deploy it by using one of the other disk image types.

RHEL for Edge Container

This image type serves the OSTree commits by using an integrated HTTP server. The **edge-container** creates an **OSTree** commit and embeds it into an OCI container with a web server. When the **edge-commit** image starts, the web server serves the commit as an OSTree repository.

RHEL for Edge Installer

The **edge-installer** image type is an Anaconda-based installer image that deploys a RHEL for Edge OSTree commit that is embedded in the installer. The **edge-installer** image type is an Anaconda-based installer image that deploys a RHEL for Edge ostree commit that is embedded in the installer image.

RHEL for Edge Raw Image

Use for bare metal platforms by flashing the RHEL Raw Images on a hard disk or boot the Raw image on a virtual machine. The **edge-raw-image** is a compressed raw images that consist of a file containing a partition layout with an existing deployed OSTree commit in it.

RHEL for Edge Simplified Installer

Use the **edge-simplified-installer** image type for unattended installations, where user configuration is provided via FDO or Ignition. The **edge-simplified-installer** image can use Ignition to inject the user configuration into the images at an early stage of the boot process. Additionally, it is possible to use FDO as a way to inject user configuration during an early stage of the boot process. After booting the Edge Simplified Installer, it provisions the RHEL for Edge image to a device with the injected user configuration.

RHEL for Edge AMI

Use this image to launch an EC2 instance in AWS cloud. The **edge-ami** image uses the Ignition tool to inject the user configuration into the images at an early stage of the boot process. You can upload the **.ami** image to AWS and boot an EC2 instance in AWS.

RHEL for Edge VMDK

Use this image to load the image on vSphere and boot the image in a vSphere VM. The **edge-vsphere** image uses the Ignition tool to inject the user configuration into the images at an early stage of the boot process.

Table 1.1. RHEL for Edge images type

Image type	File type	Suitable for network-based deployments	Suitable for non-network-based deployments
RHEL for Edge Commit	**.tar**	Yes	No
RHEL for Edge Container	**.tar**	No	Yes
RHEL for Edge Installer	**.iso**	No	Yes
RHEL for Edge Raw Image	**.raw.xz**	Yes	Yes
RHEL for Edge Simplified Installer	**.iso**	Yes	Yes
RHEL for Edge AMI	**.ami**	Yes	Yes
RHEL for Edge VMDK	**.vmdk**	Yes	Yes

Additional resources

- Interactively installing RHEL from installation media

1.3. NON-NETWORK-BASED DEPLOYMENTS

Use RHEL image builder to create flexible RHEL **rpm-ostree** images to suit your requirements, and then use Anaconda to deploy them in your environment.

You can access RHEL image builder through a command-line interface in the **composer-cli** tool, or use a graphical user interface in the RHEL web console.

Composing and deploying a RHEL for Edge image in non-network-based deployments involves the following high-level steps:

1. Install and register a RHEL system
2. Install RHEL image builder
3. Using RHEL image builder, create a blueprint with customizations for RHEL for Edge Container image
4. Import the RHEL for Edge blueprint in RHEL image builder
5. Create a RHEL for Edge image embed in an OCI container with a webserver ready to deploy the commit as an OSTree repository
6. Download the RHEL for Edge Container image file
7. Deploy the container serving a repository with the RHEL for Edge Container commit

8. Using RHEL image builder, create another blueprint for RHEL for Edge Installer image
9. Create a RHEL for Edge Installer image configured to pull the commit from the running container embedded with RHEL for Edge Container image
10. Download the RHEL for Edge Installer image
11. Run the installation

1.4. NETWORK-BASED DEPLOYMENTS

Use RHEL image builder to create flexible RHEL **rpm-ostree** images to suit your requirements, and then use Anaconda to deploy them in your environment. RHEL image builder automatically identifies the details of your deployment setup and generates the image output as an **edge-commit** as a **.tar** file.

You can access RHEL image builder through a command-line interface in the **composer-cli** tool, or use a graphical user interface in the RHEL web console.

You can compose and deploy the RHEL for Edge image by performing the following high-level steps:

For an attended installation

1. Install and register a RHEL system
2. Install RHEL image builder
3. Using RHEL image builder, create a blueprint for RHEL for Edge image
4. Import the RHEL for Edge blueprint in RHEL image builder
5. Create a RHEL for Edge Commit (**.tar**) image
6. Download the RHEL for Edge image file
7. On the same system where you have installed RHEL image builder, install a web server that you want to serve the RHEL for Edge Commit content. For instructions, see Setting up and configuring NGINX
8. Extract the RHEL for Edge Commit (**.tar**) content to the running web server
9. Create a Kickstart file that pulls the OSTree content from the running web server. For details on how to modify the Kickstart to pull the OSTree content, see Extracting the RHEL for Edge image commit
10. Boot the RHEL installer ISO on the edge device and provide the Kickstart to it.

For an unattended installation, you can customize the RHEL installation ISO and embed the Kickstart file to it.

1.5. DIFFERENCE BETWEEN RHEL RPM IMAGES AND RHEL FOR EDGE IMAGES

You can create RHEL system images in traditional package-based RPM format and also as RHEL for Edge (**rpm-ostree**) images.

You can use the traditional package-based RPMs to deploy RHEL on traditional data centers. However,

with RHEL for Edge images you can deploy RHEL on servers other than traditional data centers. These servers include systems where processing of large amounts of data is done closest to the source where data is generated, the Edge servers.

The RHEL for Edge (**rpm-ostree**) images are not a package manager. They only support complete bootable file system trees, not individual files. These images do not have information regarding the individual files such as how these files were generated or anything related to their origin.

The **rpm-ostree** images need a separate mechanism, the package manager, to install additional applications in the **/var** directory. With that, the **rpm-ostree** image keeps the operating system unchanged, while maintaining the state of the **/var** and **/etc** directories. The atomic updates enable rollbacks and background staging of updates.

Refer to the following table to know how RHEL for Edge images differ from the package-based RHEL RPM images.

Table 1.2. Difference between RHEL RPM images and RHEL for Edge images

Key attributes	RHEL RPM image	RHEL for Edge image
OS assembly	You can assemble the packages locally to form an image.	The packages are assembled in an OSTree which you can install on a system.
OS updates	You can use **dnf update** to apply the available updates from the enabled repositories.	You can use **rpm-ostree upgrade** to stage an update if any new commit is available in the OSTree remote at **/etc/ostree/remotes.d/**. The update takes effect on system reboot.
Repository	The package contains DNF repositories	The package contains OSTree remote repository
User access permissions	Read write	Read-only (**/usr**)
Data persistence	You can mount the image to any non **tmpfs** mount point	**/etc** & **/var** are read/write enabled and include persisting data.

CHAPTER 2. SETTING UP RHEL IMAGE BUILDER

Use RHEL image builder to create your customized RHEL for Edge images. After you install RHEL image builder on a RHEL system, RHEL image builder is available as an application in RHEL web console. You can also access RHEL image builder on the command line by using the **composer-cli** tool.

NOTE

It is recommended to install RHEL image builder on a virtual machine.

2.1. IMAGE BUILDER SYSTEM REQUIREMENTS

The environment where RHEL image builder runs, for example a virtual machine, must meet the requirements that are listed in the following table.

NOTE

Running RHEL image builder inside a container is not supported.

Table 2.1. Image builder system requirements

Parameter	Minimal Required Value
System type	A dedicated virtual machine
Processor	2 cores
Memory	4 GiB
Disk space	20 GiB
Access privileges	Administrator level (root)
Network	Connectivity to Internet

NOTE

The 20 GiB disk space requirement is enough to install and run RHEL image builder in the host. To build and deploy image builds, you must allocate additional dedicated disk space.

2.2. INSTALLING RHEL IMAGE BUILDER

To install RHEL image builder on a dedicated virtual machine, follow these steps:

Prerequisites

- The virtual machine is created and is powered on.
- You have installed RHEL and you have subscribed to RHSM or Red Hat Satellite.

- You have enabled the **BaseOS** and **AppStream** repositories to be able to install the RHEL image builder packages.

Procedure

1. Install the following packages on the virtual machine.
   - osbuild-composer
   - composer-cli
   - cockpit-composer
   - bash-completion
   - firewalld

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

   RHEL image builder is installed as an application in RHEL web console.
2. Reboot the virtual machine
3. Configure the system firewall to allow access to the web console:

   ```
   # firewall-cmd --add-service=cockpit && firewall-cmd --add-service=cockpit --permanent
   ```
4. Enable RHEL image builder.

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

   The osbuild-composer and cockpit services start automatically on first access.
5. Load the shell configuration script so that the autocomplete feature for the **composer-cli** command starts working immediately without reboot:

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

Additional resources

- Managing repositories

CHAPTER 3. CONFIGURING RHEL IMAGE BUILDER REPOSITORIES

To use RHEL image builder, you must ensure that the repositories are configured. You can use the following types of repositories in RHEL image builder:

Official repository overrides

Use these if you want to download base system RPMs from elsewhere than the Red Hat Content Delivery Network (CDN) official repositories, for example, a custom mirror in your network. Using official repository overrides disables the default repositories, and your custom mirror must contain all the necessary packages.

Custom third-party repositories

Use these to include packages that are not available in the official RHEL repositories.

3.1. ADDING CUSTOM THIRD-PARTY REPOSITORIES TO RHEL IMAGE BUILDER

You can add custom third-party sources to your repositories and manage these repositories by using the **composer-cli**.

Prerequisites

- You have the URL of the custom third-party repository.

Procedure

1. Create a repository source file, such as **/root/repo.toml**. For example:

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

   The **type** field accepts the following valid values: **yum-baseurl**, **yum-mirrorlist**, and **yum-metalink**.

2. Save the file in the TOML format.

3. Add the new third-party source to RHEL image builder:

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

Verification

1. Check if the new source was successfully added:

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

2. Check the new source content:

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

3.2. ADDING THIRD-PARTY REPOSITORIES WITH SPECIFIC DISTRIBUTIONS TO RHEL IMAGE BUILDER

You can specify a list of distributions in the custom third-party source file by using the optional field **distro**. The repository file uses the distribution string list while resolving dependencies during the image building.

Any request that specifies **rhel-9** uses this source. For example, if you list packages and specify **rhel-9**, it includes this source. However, listing packages for the host distribution do not include this source.

Prerequisites

- You have the URL of the custom third-party repository.
- You have the list of distributions that you want to specify.

Procedure

1. Create a repository source file, such as **/root/repo.toml**. For example, to specify the distribution:

```
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. Save the file in the TOML format.
3. Add the new third-party source to RHEL image builder:

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

Verification

1. Check if the new source was successfully added:

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

2. Check the new source content:

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

3.3. CHECKING REPOSITORIES METADATA WITH GPG

To detect and avoid corrupted packages, you can use the DNF package manager to check the GNU Privacy Guard (GPG) signature on RPM packages, and also to check if the repository metadata has been signed with a GPG key.

You can either enter the **gpgkey** that you want to do the check over **https** by setting the **gpgkeys** field with the key URL. Alternatively, to improve security, you can also embed the whole key into the **gpgkeys** field, to import it directly instead of fetching the key from the URL.

Prerequisites

- The directory that you want to use as a repository exists and contains packages.

Procedure

1. Access the folder where you want to create a repository:

```
$ cd repo/
```

2. Run the **createrepo_c** to create a repository from RPM packages:

```
$ createrepo_c .
```

3. Access the directory where the repodata is:

```
$ cd repodata/
```

4. Sign your **repomd.xml** file:

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

5. To enable GPG signature checks in the repository:

   a. Set **check_repogpg = true** in the repository source.

   b. Enter the **gpgkey** that you want to do the check. If your key is available over **https**, set the **gpgkeys** field with the key URL for the key. You can add as many URL keys as you need. The following is an example:

```
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"]
```

   As an alternative, add the GPG key directly in the **gpgkeys** field, for example:

```
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-----"']
```

- If the test does not find the signature, the GPG tool shows an error similar to the following one:

```
$ 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)
```

- If the signature is invalid, the GPG tool shows an error similar to the following one:

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

Verification

- Test the signature of the repository manually:

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

3.4. RHEL IMAGE BUILDER OFFICIAL REPOSITORY OVERRIDES

RHEL image builder **osbuild-composer** back end does not inherit the system repositories located in the **/etc/yum.repos.d/** directory. Instead, it has its own set of official repositories defined in the **/usr/share/osbuild-composer/repositories** directory. This includes the Red Hat official repository, which contains the base system RPMs to install additional software or update already installed programs to newer versions. If you want to override the official repositories, you must define overrides in **/etc/osbuild-composer/repositories/**. This directory is for user defined overrides and the files located there take precedence over those in the **/usr/share/osbuild-composer/repositories/** directory.

The configuration files are not in the usual DNF repository format known from the files in **/etc/yum.repos.d/**. Instead, they are JSON files.

3.5. OVERRIDING A SYSTEM REPOSITORY

You can configure your own repository override for RHEL image builder in the **/etc/osbuild-composer/repositories** directory.

Prerequisites

- You have a custom repository that is accessible from your host system.

Procedure

1. Create the **/etc/osbuild-composer/repositories/** directory to store your repository overrides:

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

2. Create a JSON file, using a name corresponding to your RHEL version. Alternatively, you can copy the file for your distribution from **/usr/share/osbuild-composer/** and modify its content.

For RHEL 9.3, use **/etc/osbuild-composer/repositories/rhel-93.json**.

3. Add the following structure to your JSON file. Specify only one of the following attributes, in the string format:

   - **baseurl** - The base URL of the repository.
   - **metalink** - The URL of a metalink file that contains a list of valid mirror repositories.
   - **mirrorlist** - The URL of a mirrorlist file that contains a list of valid mirror repositories. The remaining fields, such as **gpgkey**, and **metadata_expire**, are optional.
     For example:

```
{
    "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
      }
   ]
}
```

   Alternatively, you can copy the JSON file for your distribution, by replacing **rhel-version.json** with your RHEL version, for example: rhel-9.json.

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

4. Optional: Verify the JSON file:

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

5. Edit the **baseurl** paths in the **rhel-9.json** file and save it. For example:

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

6. Restart the **osbuild-composer.service**:

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

Verification

- Check if the repository points to the correct URLs:

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

  You can see that the repository points to the correct URLs which are copied from the **/etc/yum.repos.d/redhat.repo** file.

Additional resources

- The latest RPMs version available in repository not visible for **osbuild-composer**. (Red Hat Knowledgebase)

3.6. OVERRIDING A SYSTEM REPOSITORY THAT REQUIRES SUBSCRIPTIONS

You can set up the **osbuild-composer** service to use system subscriptions that are defined in the **/etc/yum.repos.d/redhat.repo** file. To use a system subscription in **osbuild-composer**, define a repository override that has the following details:

- The same **baseurl** as the repository defined in **/etc/yum.repos.d/redhat.repo**.
- The value of **"rhsm": true** defined in the JSON object.

NOTE

osbuild-composer does not automatically use repositories defined in **/etc/yum.repos.d/**. You need to manually specify them either as a system repository override or as an additional **source** by using **composer-cli**. The "BaseOS" and "AppStream" repositories usually use system repository overrides, whereas all the other repositories use **composer-cli** sources.

Prerequisites

- Your system has a subscription defined in **/etc/yum.repos.d/redhat.repo**
- You have created a repository override. See Overriding a system repository.

Procedure

1. Get the **baseurl** from the **/etc/yum.repos.d/redhat.repo** file:

```
# 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. Configure the repository override to use the same **baseurl** and set **rhsm** to 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. Restart the **osbuild-composer.service**:

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

Additional resources

- RHEL image builder uses CDN repositories when host is registered to Satellite 6 (Red Hat Knowledgebase)

3.7. CONFIGURING AND USING SATELLITE CV AS A CONTENT SOURCE

You can use Satellite's content views (CV) as repositories to build images with RHEL image builder. For that, on your host registered to Satellite, manually configure the repository references to be able to retrieve from the Satellite repositories, instead of the Red Hat Content Delivery Network (CDN) official repositories.

Prerequisites

- You have installed RHEL image builder. See Installing RHEL image builder .
- You are using RHEL image builder on a host registered to Satellite 6. See linkhttps://docs.redhat.com/en/documentation/red_hat_satellite/6.7/html/provisioning_guide/inc an-image-builder-image-for-provisioning[Using a RHEL image builder image for Provisioning].

Procedure

1. Find the repository URL from your currently configured repositories:

```
$ sudo yum -v repolist "-baseos-rpms" | grep -i repo-baseurl
  Repo-baseurl :
```

The following output is an example:

```
https://satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV/content/dist/rhel9/9/x86_64/baseos/os
```

2. Modify the hard-coded repositories to a Satellite Server.
   a. Create a repository directory with the **0755** permission:

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

   b. Copy the content from **/usr/share/osbuild-composer/repositories/*.json** to the directory that you created:

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

c. Update the Satellite URL and the file contents through the **/content/dist/*** line:

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

d. Verify that the configuration was correctly replaced:

```
$ sudo vi /etc/osbuild-composer/repositories/rhel-9.json
```

3. Restart the services:

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

4. Override the required system repository in Red Hat image builder configuration and use the URL of your Satellite repository as a baseurl. See Overriding a system repository.

Additional resources

- Composer RHEL image builder fails when multiple custom repositories are defined on the Satellite (Red Hat Knowledgebase)

3.8. USING SATELLITE CV AS REPOSITORIES TO BUILD IMAGES IN RHEL IMAGE BUILDER

Configure RHEL image builder to use Satellite's content views (CV) as repositories to build your custom images.

Prerequisites

- You have integrated Satellite with RHEL web console. See Enabling the RHEL web console on Satellite

Procedure

1. In the Satellite web UI, navigate to **Content > Products**, select your **Product** and click the repository you want to use.
2. Search for the secured URL (HTTPS) in the Published field and copy it.
3. Use the URL that you copied as a baseurl for the Red Hat image builder repository. See Adding custom third-party repositories to RHEL image builder.

Next steps

- Build the image. See Creating a system image by using RHEL image builder in the web console interface.

CHAPTER 4. COMPOSING A RHEL FOR EDGE IMAGE USING IMAGE BUILDER IN RHEL WEB CONSOLE

Use RHEL image builder to create a custom RHEL for Edge image (OSTree commit).

To access RHEL image builder and to create your custom RHEL for Edge image, you can either use the RHEL web console interface or the command line.

You can compose RHEL for Edge images by using RHEL image builder in RHEL web console by performing the following high-level steps:

1. Access RHEL image builder in RHEL web console
2. Create a blueprint for RHEL for Edge image.
3. Create a RHEL for Edge image. You can create the following images:
   - RHEL for Edge Commit image.
   - RHEL for Edge Container image.
   - RHEL for Edge Installer image.
4. Download the RHEL for Edge image

4.1. ACCESSING RHEL IMAGE BUILDER IN THE RHEL WEB CONSOLE

To access RHEL image builder in RHEL web console, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You have installed a RHEL system.
- You have administrative rights on the system.
- You have subscribed the RHEL system to Red Hat Subscription Manager (RHSM) or to Red Hat Satellite Server.
- Your system is powered on and accessible over the network.
- You have installed RHEL image builder on the system.

Procedure

1. On your RHEL system, access https://localhost:9090/ in a web browser.
2. For more information about how to remotely access RHEL image builder, see Managing systems using the RHEL 9 web console document.
3. Log in to the web console using an administrative user account.
4. On the web console, in the left hand menu, click **Apps**.
5. Click **Image Builder**.

The RHEL image builder dashboard opens in the right pane. You can now proceed to create a blueprint for the RHEL for Edge images.

4.2. CREATING A BLUEPRINT FOR A RHEL FOR EDGE IMAGE USING IMAGE BUILDER IN THE WEB CONSOLE

To create a blueprint for a RHEL for Edge image by using RHEL image builder in RHEL web console, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- On a RHEL system, you have opened the RHEL image builder dashboard.

Procedure

1. On the RHEL image builder dashboard, click **Create Blueprint**.
   The **Create Blueprint** dialogue box opens.
2. On the **Details** page:
   a. Enter the name of the blueprint and, optionally, its description. Click **Next**.
3. Optional: In the **Packages** page:
   a. On the **Available packages** search, enter the package name and click the **>** button to move it to the Chosen packages field. Search and include as many packages as you want. Click **Next**.

NOTE

These customizations are all optional unless otherwise specified.

4. On the **Kernel** page, enter a kernel name and the command-line arguments.
5. On the **File system** page, select **Use automatic partitioning**. OSTree systems do not support filesystem customization, because OSTree images have their own mount rule, such as read-only. Click **Next**.
6. On the **Services** page, you can enable or disable services:
   a. Enter the service names you want to enable or disable, separating them by a comma, by space, or by pressing the **Enter** key. Click **Next**.
7. On the **Firewall** page, set up your firewall setting:
   a. Enter the **Ports**, and the firewall services you want to enable or disable.
   b. Click the **Add zone** button to manage your firewall rules for each zone independently. Click **Next**.
8. On the **Users** page, add a users by following the steps:
   a. Click **Add user**.
   b. Enter a **Username**, a **password**, and a **SSH key**. You can also mark the user as a privileged user, by clicking the **Server administrator** checkbox. Click **Next**.

9. On the **Groups** page, add groups by completing the following steps:
    a. Click the **Add groups** button:
        i. Enter a **Group name** and a **Group ID**. You can add more groups. Click **Next**.
10. On the **SSH keys** page, add a key:
    a. Click the **Add key** button.
        i. Enter the SSH key.
        ii. Enter a **User**. Click **Next**.
11. On the **Timezone** page, set your timezone settings:
    a. On the **Timezone** field, enter the timezone you want to add to your system image. For example, add the following timezone format: "US/Eastern".
    If you do not set a timezone, the system uses Universal Time, Coordinated (UTC) as default.
    b. Enter the **NTP** servers. Click **Next**.
12. On the **Locale** page, complete the following steps:
    a. On the **Keyboard** search field, enter the package name you want to add to your system image. For example: ["en_US.UTF-8"].
    b. On the **Languages** search field, enter the package name you want to add to your system image. For example: "us". Click **Next**.
13. On the **Others** page, complete the following steps:
    a. On the **Hostname** field, enter the hostname you want to add to your system image. If you do not add a hostname, the operating system determines the hostname.
    b. Mandatory only for the Simplifier Installer image: On the **Installation Devices** field, enter a valid node for your system image. For example: **dev/sda**. Click **Next**.
14. Mandatory only when building FIDO images: On the **FIDO device onboarding** page, complete the following steps:
    a. On the **Manufacturing server URL** field, enter the following information:
        i. On the **DIUN public key insecure** field, enter the insecure public key.
        ii. On the **DIUN public key hash** field, enter the public key hash.
        iii. On the **DIUN public key root certs** field, enter the public key root certs. Click **Next**.
15. On the **OpenSCAP** page, complete the following steps:
    a. On the **Datastream** field, enter the **datastream** remediation instructions you want to add to your system image.
    b. On the **Profile ID** field, enter the **profile_id** security profile you want to add to your system image. Click **Next**.

16. Mandatory only when building Ignition images: On the **Ignition** page, complete the following steps:
    a. On the **Firstboot URL** field, enter the package name you want to add to your system image.
    b. On the **Embedded Data** field, drag or upload your file. Click **Next**.
17. . On the **Review** page, review the details about the blueprint. Click **Create**.

The RHEL image builder view opens, listing existing blueprints.

4.3. CREATING A RHEL FOR EDGE IMAGE

Create a RHEL for Edge image. Choose one of the following image types, according to your needs.

4.3.1. Creating a RHEL for Edge Commit image by using image builder in web console

You can create a **"RHEL for Edge Commit"** image by using RHEL image builder in RHEL web console. The **"RHEL for Edge Commit (.tar)"** image type contains a full operating system, but it is not directly bootable. To boot the Commit image type, you must deploy it in a running container.

Prerequisites

- On a RHEL system, you have accessed the RHEL image builder dashboard.

Procedure

1. On the RHEL image builder dashboard click **Create Image**.
2. On the **Image output** page, perform the following steps:
    a. From the **Select a blueprint** dropdown menu, select the blueprint you want to use.
    b. From the **Image output type** dropdown list, select **"RHEL for Edge Commit (.tar)"**.
    c. Click **Next**.
    d. On the **OSTree settings** page, enter:
        i. **Repository URL**: specify the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/.
        ii. **Parent commit**: specify a previous commit, or leave it empty if you do not have a commit at this time.
        iii. In the **Ref** text box, specify a reference path for where your commit is going to be created. By default, the web console specifies **rhel/9/$ARCH/edge**. The "$ARCH" value is determined by the host machine. Click **Next**.
    e. On the **Review** page, check the customizations and click **Create**.
    RHEL image builder starts to create a RHEL for Edge Commit image for the blueprint that you created.

NOTE

The image creation process takes up to 20 minutes to complete.

Verification

1. To check the RHEL for Edge Commit image creation progress:
    a. Click the **Images** tab.

After the image creation process is complete, you can download the resulting **"RHEL for Edge Commit (.tar)"** image.

Additional resources

- Downloading a RHEL for Edge image

4.3.2. Creating a RHEL for Edge Container image by using RHEL image builder in RHEL web console

You can create RHEL for Edge images by selecting **"RHEL for Edge Container (.tar)"**. The **RHEL for Edge Container (.tar)** image type creates an OSTree commit and embeds it into an OCI container with a web server. When the container is started, the web server serves the commit as an OSTree repository.

Follow the steps in this procedure to create a RHEL for Edge Container image using image builder in RHEL web console.

Prerequisites

- On a RHEL system, you have accessed the RHEL image builder dashboard.
- You have created a blueprint.

Procedure

1. On the RHEL image builder dashboard click **Create Image**.
2. On the **Image output** page, perform the following steps:
3. From the **Select a blueprint** dropdown menu, select the blueprint you want to use.
    a. From the **Image output type** dropdown list, select **"RHEL for Edge Container (.tar)"**.
    b. Click **Next**.
    c. On the **OSTree** page, enter:
        i. **Repository URL**: specify the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. By default, the repository folder for a RHEL for Edge Container image is "/repo".
        To find the correct URL to use, access the running container and check the **nginx.conf** file. To find which URL to use, access the running container and check the **nginx.conf** file. Inside the **nginx.conf** file, find the **root** directory entry to search for the **/repo/** folder information. Note that, if you do not specify a repository URL when creating a RHEL for Edge Container image **(.tar)** by using RHEL image builder, the default **/repo/** entry is created in the **nginx.conf** file.

ii. **Parent commit**: specify a previous commit, or leave it empty if you do not have a commit at this time.

iii. In the **Ref** text box, specify a reference path for where your commit is going to be created. By default, the web console specifies **rhel/9/$ARCH/edge**. The "$ARCH" value is determined by the host machine. Click **Next**.

d. On the **Review** page, check the customizations. Click **Save blueprint**.

4. Click **Create**.
RHEL image builder starts to create a RHEL for Edge Container image for the blueprint that you created.

NOTE

The image creation process takes up to 20 minutes to complete.

Verification

1. To check the RHEL for Edge Container image creation progress:

a. Click the **Images** tab.

After the image creation process is complete, you can download the resulting **"RHEL for Edge Container (.tar)"** image.

Additional resources

- Downloading a RHEL for Edge image

4.3.3. Creating a RHEL for Edge Installer image by using image builder in RHEL web console

You can create RHEL for Edge Installer images by selecting **RHEL for Edge Installer (.iso)**. The **RHEL for Edge Installer (.iso)** image type pulls the OSTree commit repository from the running container served by the **RHEL for Edge Container (.tar)** and creates an installable boot ISO image with a Kickstart file that is configured to use the embedded OSTree commit.

Follow the steps in this procedure to create a RHEL for Edge image using image builder in RHEL web console.

Prerequisites

- On a RHEL system, you have accessed the image builder dashboard.
- You created a blueprint.
- You created a RHEL for Edge Container image and loaded it into a running container. See Creating a RHEL for Edge Container image for non-network-based deployments .

Procedure

1. On the RHEL image builder dashboard click **Create Image**.
2. On the **Image output** page, perform the following steps:

a. From the **Select a blueprint** dropdown menu, select the blueprint you want to use.

b. From the **Image output type** dropdown list, select **RHEL for Edge Installer (.iso)** image.

c. Click **Next**.

d. On the **OSTree settings** page, enter:

i. **Repository URL**: specify the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/.

ii. In the **Ref** text box, specify a reference path for where your commit is going to be created. By default, the web console specifies **rhel/9/$ARCH/edge**. The "$ARCH" value is determined by the host machine. Click **Next**.

e. On the **Review** page, check the customizations. Click **Save blueprint**.

3. Click **Create**.
RHEL image builder starts to create a RHEL for Edge Installer image for the blueprint that you created.

NOTE

The image creation process takes up to 20 minutes to complete.

Verification

1. To check the RHEL for Edge Installer image creation progress:

a. Click the **Images** tab.

After the image creation process is complete, you can download the resulting **RHEL for Edge Installer (.iso)** image and boot the ISO image into a device.

Additional resources

- Downloading a RHEL for Edge image

4.4. DOWNLOADING A RHEL FOR EDGE IMAGE

After you successfully create the RHEL for Edge image by using RHEL image builder, download the image on the local host.

Procedure

To download an image:

1. From the **More Options** menu, click **Download**.
The RHEL image builder tool downloads the file at your default download location.

The downloaded file consists of a **.tar** file with an OSTree repository for RHEL for Edge Commit and RHEL for Edge Container images, or a **.iso** file for RHEL for Edge Installer images, with an OSTree repository. This repository contains the commit and a **json** file which contains information metadata about the repository content.

4.5. ADDITIONAL RESOURCES

- Composing a RHEL for Edge image using image builder command-line .

CHAPTER 5. COMPOSING A RHEL FOR EDGE IMAGE USING IMAGE BUILDER COMMAND-LINE

You can use image builder to create a customized RHEL for Edge image (OSTree commit).

To access image builder and to create your custom RHEL for Edge image, you can either use the RHEL web console interface or the command line.

For Network-based deployments, the workflow to compose RHEL for Edge images by using the CLI, involves the following high-level steps:

1. Create a blueprint for RHEL for Edge image
2. Create a RHEL for Edge Commit image
3. Download the RHEL for Edge Commit image

For Non-Network-based deployments, the workflow to compose RHEL for Edge images by using the CLI, involves the following high-level steps:

1. Create a blueprint for RHEL for Edge image
2. Create a RHEL for Edge image. You can create the following images:
   - RHEL for Edge Commit image.
   - RHEL for Edge Container image.
   - RHEL for Edge Installer image.
3. Download the RHEL for Edge image

To perform the steps, use the **composer-cli** package.

NOTE

To run the **composer-cli** commands as non-root, you must be part of the **weldr** group or you must have administrator access to the system.

5.1. CREATING IMAGES FOR NETWORK-BASED DEPLOYMENTS

This provides steps on how to build **OSTree** commits. These **OSTree** commits contain a full operating system, but are not directly bootable. To boot them, you need to deploy them using a Kickstart file.

5.1.1. Creating a blueprint for the commit image by using image builder CLI

Create a blueprint for RHEL for Edge Commit image by using the CLI.

Prerequisites

- You do not have an existing blueprint. To verify that, list the existing blueprints:

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

Procedure

Procedure

1. Create a plain text file in the TOML format, with the following content:

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

Where,

- *blueprint-name* is the name and blueprint-text-description is the description for your blueprint.
- *0.0.1* is the version number according to the Semantic Versioning scheme.
- *Modules* describe the package name and matching version glob to be installed into the image, for example, the package name = "tmux" and the matching version glob is version = "2.9a".
  Notice that currently there are no differences between packages and modules.
- *Groups* are packages groups to be installed into the image, for example the group package anaconda-tools.
  At this time, if you do not know the modules and groups, leave them empty.

2. Include the required packages and customize the other details in the blueprint to suit your requirements.
   For every package that you want to include in the blueprint, add the following lines to the file:

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

Where,

- package-name is the name of the package, such as httpd, gdb-doc, or coreutils.
- package-version is the version number of the package that you want to use.
  The package-version supports the following DNF version specifications:
- For a specific version, use the exact version number such as 9.0.
- For the latest available version, use the asterisk *.
- For the latest minor version, use formats such as 9.*.

3. Push (import) the blueprint to the RHEL image builder server:

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

4. List the existing blueprints to check whether the created blueprint is successfully pushed and exists.

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

5. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Additional resources

- Supported Image Customizations

5.1.2. Creating a RHEL for Edge Commit image by using image builder CLI

To create a RHEL for Edge Commit image by using RHEL image builder command-line interface, ensure that you have met the following prerequisites and follow the procedure.

Prerequisites

- You have created a blueprint for RHEL for Edge Commit image.

Procedure

1. Create the RHEL for Edge Commit image.

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

Where,

- *blueprint-name* is the RHEL for Edge blueprint name.
- *image-type* is **edge-commit** for **network-based deployment**.
A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also keep the UUID number handy for further tasks.

2. Check the image compose status.

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

The output displays the status in the following format:

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

NOTE

The image creation process takes up to 20 minutes to complete.

To interrupt the image creation process, run:

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

To delete an existing image, run:

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

After the image is ready, you can download it and use the image on your **network deployments**.

Additional resources

- Composing a RHEL for Edge image using RHEL image builder command-line

5.1.3. Creating an image update with a ref commit by using RHEL image builder CLI

If you performed a change in an existing blueprint, for example, you added a new package, and you want to update an existing RHEL for Edge image with this new package, you can use the **--parent** argument to generate an updated **RHEL for Edge Commit (.tar)** image. The **--parent** argument can be either a **ref** that exists in the repository specified by the **URL** argument, or you can use the **Commit ID**, which you can find in the extracted **.tar** image file.

Both the **ref** and **Commit ID** arguments retrieve a parent for the new commit that you are building. RHEL image builder read information from the parent commit user database that affects parts of the new commit that you are building, but preserves UIDs and GIDs for the package-created system users and groups.

Prerequisites

- You have updated an existing blueprint for RHEL for Edge image.
- You have an existing RHEL for Edge image (OSTree commit). See Extracting RHEL for Edge image commit.
- You made the **ref** that you are going to build available at the **OSTree** repository specified by the URL.

Procedure

1. Create the RHEL for Edge commit image:

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

For example:

- To create a new RHEL for Edge commit based on a **parent** and with a new **ref**, run the following command:

```
# 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
```

- To create a new RHEL for Edge commit based on the same **ref**, run the following command:

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

Where:

  - The *--ref* argument specifies the same path value that you used to build an OSTree repository.

- The *--parent* argument specifies the parent commit. It can be ref to be resolved and pulled, for example **rhel/9/x86_64/edge**, or the **Commit ID** that you can find in the extracted **.tar** file.
- *blueprint-name* is the RHEL for Edge blueprint name.
- The **--url** argument specifies the URL to the OSTree repository of the commit to embed in the image, for example, http://10.0.2.2:8080/repo.
- *image-type* is **edge-commit** for **network-based deployment**.

NOTE

- The **--parent** argument can only be used for the **RHEL for Edge Commit (.tar)** image type. Using the **--url** and **--parent** arguments together results in errors with the **RHEL for Edge Container (.tar)** image type.
- If you omit the **parent ref** argument, the system falls back to the **ref** specified by the **--ref** argument.

A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also keep the UUID number handy for further tasks.

2. Check the image compose status.

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

The output displays the status in the following format:

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

NOTE

The image creation process takes a few minutes to complete.

(Optional) To interrupt the image creation process, run:

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

(Optional) To delete an existing image, run:

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

Next steps

After the image creation is complete, to upgrade an existing OSTree deployment, you need:

- Set up a repository. See Deploying a RHEL for Edge image .
- Add this repository as a remote, that is, the http or https endpoint that hosts the OSTree content.

- Pull the new OSTree commit onto their existing running instance. See Deploying RHEL for Edge image updates manually .

Additional resources

- Creating a system image with RHEL image builder on the command line
- Downloading a RHEL for Edge image by using the RHEL image builder command-line interface

5.1.4. Downloading a RHEL for Edge image using the image builder command-line interface

To download a RHEL for Edge image by using RHEL image builder command-line interface, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You have created a RHEL for Edge image.

Procedure

1. Review the RHEL for Edge image status.

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

   The output must display the following:

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

2. Download the image.

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

   RHEL image builder downloads the image as a **tar** file to the current directory.

   The UUID number and the image size is displayed alongside.

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

The image contains a commit and a **json** file with information metadata about the repository content.

Additional resources

- Deploying a RHEL for Edge image in a network-based environment

5.2. CREATING IMAGES FOR NON-NETWORK-BASED DEPLOYMENTS

Build a boot ISO image that installs an OSTree-based system by using the "RHEL for Edge Container" and the "RHEL for Edge Installer" images, and that can be later deployed to a device in disconnected environments.

5.2.1. Creating a RHEL for Edge Container blueprint by using image builder CLI

To create a blueprint for RHEL for Edge Container image, perform the following steps:

Procedure

1. Create a plain text file in the TOML format, with the following content:

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

Where,

- *blueprint-name* is the name and blueprint-text-description is the description for your blueprint.
- *0.0.1* is the version number according to the Semantic Versioning scheme.
- *Modules* describe the package name and matching version glob to be installed into the image, for example, the package name = "tmux" and the matching version glob is version = "2.9a".
  Notice that currently there are no differences between packages and modules.
- *Groups* are packages groups to be installed into the image, for example the group package anaconda-tools.
  At this time, if you do not know the modules and groups, leave them empty.

2. Include the required packages and customize the other details in the blueprint to suit your requirements.
   For every package that you want to include in the blueprint, add the following lines to the file:

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

Where,

- package-name is the name of the package, such as **httpd**, **gdb-doc**, or **coreutils**.
- package-version is the version number of the package that you want to use.
  The package-version supports the following **dnf** version specifications:
- For a specific version, use the exact version number such as 9.0.
- For the latest available version, use the asterisk *.
- For the latest minor version, use formats such as 9.*.

3. Push (import) the blueprint to the RHEL image builder server:

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

4. List the existing blueprints to check whether the created blueprint is successfully pushed and exists.

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

5. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Additional resources

- Supported Image Customizations

5.2.2. Creating a RHEL for Edge Installer blueprint using image builder CLI

You can create a blueprint to build a **RHEL for Edge Installer (.iso)** image, and specify user accounts to automatically create one or more users on the system at installation time.

WARNING

When you create a user in the blueprint with the **customizations.user** customization, the blueprint creates the user under the **/usr/lib/passwd** directory and the password, under the **/usr/etc/shadow** directory. Note that you cannot change the password in further versions of the image in a running system using **OSTree** updates. The users you create with blueprints must be used only to gain access to the created system. After you access the system, you need to create users, for example, using the **useradd** command.

To create a blueprint for RHEL for Edge Installer image, perform the following steps:

Procedure

1. Create a plain text file in the TOML format, with the following content:

```
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"]
```

Where,

- *blueprint-name* is the name and blueprint-text-description is the description for your blueprint.

- *0.0.1* is the version number according to the Semantic Versioning scheme.

2. Push (import) the blueprint to the RHEL image builder server:

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

3. List the existing blueprints to check whether the created blueprint is successfully pushed and exists.

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

4. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Additional resources

- Supported Image Customizations

5.2.3. Creating a RHEL for Edge Container image by using image builder CLI

To create a RHEL for Edge Container image by using RHEL image builder command-line interface, ensure that you have met the following prerequisites and follow the procedure.

Prerequisites

- You have created a blueprint for RHEL for Edge Container image.

Procedure

1. Create the RHEL for Edge Container image.

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

Where,

- **--ref** is the same value that customer used to build OSTree repository
- **--url** is the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. By default, the repository folder for a RHEL for Edge Container image is "/repo". See Setting up a web server to install RHEL for Edge image . To find the correct URL to use, access the running container and check the **nginx.conf** file. To find which URL to use, access the running container and check the **nginx.conf** file. Inside the **nginx.conf** file, find the **root** directory entry to search for the **/repo/** folder information. Note that, if you do not specify a repository URL when creating a RHEL for Edge Container image **(.tar)** by using RHEL image builder, the default **/repo/** entry is created in the **nginx.conf** file.
- *blueprint-name* is the RHEL for Edge blueprint name.
- *image-type* is **edge-container** for **non-network-based deployment**.

A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also keep the UUID number handy for further tasks.

2. Check the image compose status.

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

The output displays the status in the following format:

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

NOTE

The image creation process takes up to 20 minutes to complete.

To interrupt the image creation process, run:

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

To delete an existing image, run:

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

After the image is ready, it can be used for **non-network deployments**. See Creating a RHEL for Edge Container image for non-network-based deployments.

Additional resources

- Composing a RHEL for Edge image using RHEL image builder command-line

5.2.4. Creating a RHEL for Edge Installer image by using the command-line interface for non-network-based deployments

To create a RHEL for Edge Installer image that embeds the **OSTree** commit, use the RHEL image builder command-line interface, and ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You have created a blueprint for RHEL for Edge Installer image.
- You have created a RHEL for Edge Container image and deployed it using a web server.

Procedure

1. Begin to create the RHEL for Edge Installer image.

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

Where,

- *ref* is the same value that customer used to build the OSTree repository
- *URL-OSTree-repository* is the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo. See Creating a RHEL for Edge Container image for non-network-based deployments.
- *blueprint-name* is the RHEL for Edge Installer blueprint name.
- *image-type* is **edge-installer**.
  A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also keep the UUID number handy for further tasks.

2. Check the image compose status.

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

The command output displays the status in the following format:

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

NOTE

The image creation process takes a few minutes to complete.

To interrupt the image creation process, run:

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

To delete an existing image, run:

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

After the image is ready, you can use it for **non-network deployments**. See Installing the RHEL for Edge image for non-network-based deployments.

5.2.5. Downloading a RHEL for Edge Installer image using the image builder CLI

To download a RHEL for Edge Installer image by using RHEL image builder command-line interface, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You have created a RHEL for Edge Installer image.

Procedure

1. Review the RHEL for Edge image status.

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

The output must display the following:

-

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

2. Download the image.

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

RHEL image builder downloads the image as an **.iso** file to the current directory.

The UUID number and the image size is displayed alongside.

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

The resulting image is a bootable ISO image.

Additional resources

- Deploying a RHEL for Edge image in a non-network-based environment

5.3. SUPPORTED IMAGE CUSTOMIZATIONS

You can customize your image by adding customizations to your blueprint, such as:

- Adding an additional RPM package
- Enabling a service
- Customizing a kernel command line parameter.

Between others. You can use several image customizations within blueprints. By using the customizations, you can add packages and groups to the image that are not available in the default packages. To use these options, configure the customizations in the blueprint and import (push) it to RHEL image builder.

Additional resources

- Blueprint import fails after adding filesystem customization "size" (Red Hat Knowledgebase)

5.3.1. Selecting a distribution

You can use the **distro** field to specify the distribution to use when composing your images or solving dependencies in the blueprint. If the **distro** field is left blank, the blueprint automatically uses the host's operating system distribution. If you do not specify a distribution, the blueprint uses the host distribution. When you upgrade the host operating system, blueprints without a specified distribution build images by using the upgraded operating system version.

You can build images for older major versions on a newer system. For example, you can use a RHEL 10 host to create RHEL 9 and RHEL 8 images. However, you cannot build images for newer major versions on an older system.

IMPORTANT

You cannot build an operating system image that differs from the RHEL image builder host. For example, you cannot use a RHEL system to build Fedora or CentOS images.

- Customize the blueprint with the RHEL distribution to always build the specified RHEL image:

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

For example:

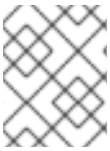
```
name = "tmux"
description = "tmux image with openssh"
version = "1.2.16"
distro = "rhel-9.5"
```

Replace **"*different_minor_version*"** to build a different minor version, for example, if you want to build a RHEL 9.5 image, use **distro** = "rhel-95". On RHEL 9.3 image, you can build minor versions such as RHEL 9.3, RHEL 9.2, and earlier releases.

5.3.2. Selecting a package group

Customize the blueprint with package groups. The **groups** list describes the groups of packages that you want to install into the image. The package groups are defined in the repository metadata. Each group has a descriptive name that is used primarily for display in user interfaces, and an ID that is commonly used in Kickstart files. In this case, you must use the ID to list a group. Groups have three different ways of categorizing their packages: mandatory, default, and optional. Only mandatory and default packages are installed in the blueprints. It is not possible to select optional packages.

The **name** attribute is a required string and must match exactly the package group id in the repositories.



NOTE

Currently, there are no differences between packages and modules in **osbuild-composer**. Both are treated as an RPM package dependency.

- Customize your blueprint with a package:

```
[[groups]]
name = "group_name"
```

Replace ***group_name*** with the name of the group. For example, **anaconda-tools**:

```
[[groups]]
name = "anaconda-tools"
```

5.3.3. Setting the image hostname

The **customizations.hostname** is an optional string that you can use to configure the final image hostname. This customization is optional, and if you do not set it, the blueprint uses the default hostname.

- Customize the blueprint to configure the hostname:

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

5.3.4. Specifying additional users

Add a user to the image, and optionally, set their SSH key. All fields for this section are optional except for the **name**.

Procedure

- Customize the blueprint to add a user to the image:

```
[[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
```

```
[[customizations.user]]
name = "admin"
description = "Administrator account"
password = "$6$CHO2$3rN8eviE2t50lmVyBYihTgVRHcaecmeCk31L..."
key = "PUBLIC SSH KEY"
home = "/srv/widget/"
shell = "/usr/bin/bash"
groups = ["widget", "users", "wheel"]
uid = 1200
gid = 1200
expiredate = 12345
```

The GID is optional and must already exist in the image. Optionally, a package creates it, or the blueprint creates the GID by using the **[[customizations.group]]** entry.

Replace *PASSWORD-HASH* with the actual **password hash**. To generate the **password hash**, use a command such as:

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

Replace the other placeholders with suitable values.

Enter the **name** value and omit any lines you do not need.

Repeat this block for every user to include.

5.3.5. Specifying additional groups

Specify a group for the resulting system image. Both the **name** and the **gid** attributes are mandatory.

- Customize the blueprint with a group:

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

Repeat this block for every group to include. For example:

```
[[customizations.group]]
name = "widget"
gid = 1130
```

5.3.6. Setting SSH key for existing users

You can use **customizations.sshkey** to set an SSH key for the existing users in the final image. Both **user** and **key** attributes are mandatory.

- Customize the blueprint by setting an SSH key for existing users:

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

For example:

```
[[customizations.sshkey]]
user = "root"
key = "SSH key for root"
```

NOTE

You can only configure the **customizations.sshkey** customization for existing users. To create a user and set an SSH key, see the Specifying additional users customization.

5.3.7. Appending a kernel argument

You can append arguments to the boot loader kernel command line. By default, RHEL image builder builds a default kernel into the image. However, you can customize the kernel by configuring it in the blueprint.

- Append a kernel boot parameter option to the defaults:

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

For example:

```
[customizations.kernel]
name = "kernel-debug"
append = "nosmt=force"
```

5.3.8. Setting time zone and NTP

You can customize your blueprint to configure the time zone and the *Network Time Protocol* (NTP). Both **timezone** and **ntpservers** attributes are optional strings. If you do not customize the time zone, the system uses *Universal Time, Coordinated* (UTC). If you do not set NTP servers, the system uses the default distribution.

- Customize the blueprint with the **timezone** and the **ntpservers** you want:

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

For example:

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

NOTE

Some image types, such as Google Cloud, already have NTP servers set up. You cannot override it because the image requires the NTP servers to boot in the selected environment. However, you can customize the time zone in the blueprint.

5.3.9. Customizing the locale settings

You can customize the locale settings for your resulting system image. Both **language** and the **keyboard** attributes are mandatory. You can add many other languages. The first language you add is the primary language and the other languages are secondary.

Procedure

- Set the locale settings:

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

For example:

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

- To list the values supported by the languages, run the following command:

```
$ localectl list-locales
```

- To list the values supported by the keyboard, run the following command:

```
$ localectl list-keymaps
```

5.3.10. Customizing firewall

Set the firewall for the resulting system image. By default, the firewall blocks incoming connections, except for services that enable their ports explicitly, such as **sshd**.

If you do not want to use the **[customizations.firewall]** or the **[customizations.firewall.services]**, either remove the attributes, or set them to an empty list []. If you only want to use the default firewall setup, you can omit the customization from the blueprint.

NOTE

The Google and OpenStack templates explicitly disable the firewall for their environment. You cannot override this behavior by setting the blueprint.

Procedure

- Customize the blueprint with the following settings to open other ports and services:

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

Where **ports** is an optional list of strings that contain ports or a range of ports and protocols to open. You can configure the ports by using the following format: **port:protocol** format. You can configure the port ranges by using the **portA-portB:protocol** format. For example:

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

You can use numeric ports, or their names from the **/etc/services** to enable or disable port lists.

- Specify which firewall services to enable or disable in the **customizations.firewall.service** section:

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

- You can check the available firewall services:

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

For example:

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

NOTE

The services listed in **firewall.services** are different from the **service-names** available in the **/etc/services** file.

5.3.11. Enabling or disabling services

You can control which services to enable during the boot time. Some image types already have services enabled or disabled to ensure that the image works correctly and you cannot override this setup. The **[customizations.services]** settings in the blueprint do not replace these services, but add the services to the list of services already present in the image templates.

- Customize which services to enable during the boot time:

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

  For example:

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

5.3.12. Specifying a custom file system configuration

You can specify a custom file system configuration in your blueprints and therefore create images with a specific disk layout, instead of the default layout configuration. By using the non-default layout configuration in your blueprints, you can benefit from:

- Security benchmark compliance
- Protection against out-of-disk errors
- Improved performance
- Consistency with existing setups

NOTE

The OSTree systems do not support the file system customizations, because OSTree images have their own mount rule, such as read-only. The following image types are not supported:

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

Additionally, the following image types do not support file system customizations, because these image types do not create partitioned operating system images:

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

However, the following image types have support for file system customization:

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

With some additional exceptions for OSTree systems, you can choose arbitrary directory names at the /**root** level of the file system, for example: ` /local`,` /mypartition`, /**$PARTITION**. In logical volumes, these changes are made on top of the LVM partitioning system. The following directories are supported: /**var**,` /var/log`, and /**var**/**lib**/**containers** on a separate logical volume. The following are exceptions at root level:

- "/home": {Deny: true},
- "/mnt": {Deny: true},
- "/opt": {Deny: true},
- "/ostree": {Deny: true},
- "/root": {Deny: true},
- "/srv": {Deny: true},
- "/var/home": {Deny: true},
- "/var/mnt": {Deny: true},
- "/var/opt": {Deny: true},
- "/var/roothome": {Deny: true},
- "/var/srv": {Deny: true},
- "/var/usrlocal": {Deny: true},

For release distributions before RHEL 8.10 and 9.5, the blueprint supports the following **mountpoints** and their sub-directories:

- / - the root mount point
- /**var**
- /**home**
- /**opt**
- /**srv**
- /**usr**
- /**app**
- /**data**

- **/tmp**

From the RHEL 9.5 and 8.10 release distributions onward, you can specify arbitrary custom mountpoints, except for specific paths that are reserved for the operating system.

You cannot specify arbitrary custom mountpoints on the following mountpoints and their sub-directories:

- **/bin**
- **/boot/efi**
- **/dev**
- **/etc**
- **/lib**
- **/lib64**
- **/lost+found**
- **/proc**
- **/run**
- **/sbin**
- **/sys**
- **/sysroot**
- **/var/lock**
- **/var/run**

You can customize the file system in the blueprint for the **/usr** custom mountpoint, but its subdirectory is not allowed.

NOTE

Customizing mount points is only supported from RHEL 9.0 distributions onward, by using the CLI. In earlier distributions, you can only specify the **root** partition as a mount point and specify the **size** argument as an alias for the image size.

If you have more than one partition in the customized image, you can create images with a customized file system partition on LVM and resize those partitions at runtime. To do this, you can specify a customized file system configuration in your blueprint and therefore create images with the required disk layout. The default file system layout remains unchanged - if you use plain images without file system customization, and **cloud-init** resizes the root partition.

The blueprint automatically converts the file system customization to an LVM partition.

You can use the custom file blueprint customization to create new files or to replace existing files. The parent directory of the file you specify must exist, otherwise, the image build fails. Ensure that the parent directory exists by specifying it in the **[[customizations.directories]]** customization.

WARNING

If you combine the files customizations with other blueprint customizations, it might affect the functioning of the other customizations, or it might override the current files customizations.

5.3.12.1. Specifying customized files in the blueprint

With the **[[customizations.files]]** blueprint customization you can:

- Create new text files.
- Modifying existing files. WARNING: this can override the existing content.
- Set user and group ownership for the file you are creating.
- Set the mode permission in the octal format.

You cannot create or replace the following files:

- **/etc/fstab**
- **/etc/shadow**
- **/etc/passwd**
- **/etc/group**

You can create customized files and directories in your image, by using the **[[customizations.files]]** and the **[[customizations.directories]]** blueprint customizations. You can use these customizations only in the **/etc** directory.

NOTE

These blueprint customizations are supported by all image types, except the image types that deploy OSTree commits, such as **edge-raw-image**, **edge-installer**, and **edge-simplified-installer**.

WARNING

If you use the **customizations.directories** with a directory path which already exists in the image with **mode**, **user** or **group** already set, the image build fails to prevent changing the ownership or permissions of the existing directory.

5.3.12.2. Specifying customized directories in the blueprint

With the **[[customizations.directories]]** blueprint customization you can:

- Create new directories.
- Set user and group ownership for the directory you are creating.
- Set the directory mode permission in the octal format.
- Ensure that parent directories are created as needed.

With the **[[customizations.files]]** blueprint customization you can:

- Create new text files.
- Modifying existing files. WARNING: this can override the existing content.
- Set user and group ownership for the file you are creating.
- Set the mode permission in the octal format.

NOTE

You cannot create or replace the following files:

- **/etc/fstab**
- **/etc/shadow**
- **/etc/passwd**
- **/etc/group**

The following customizations are available:

- Customize the file system configuration in your blueprint:

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

The **MINIMUM-PARTITION-SIZE** value has no default size format. The blueprint customization supports the following values and units: kB to TB and KiB to TiB. For example, you can define the mount point size in bytes:

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

- Define the mount point size by using units. For example:

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

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

-

- Define the minimum partition by setting **minsize**. For example:

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

- Create customized directories under the /**etc** directory for your image by using **[[customizations.directories]]**:

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

The blueprint entries are described as following:

  - **path** - Mandatory - enter the path to the directory that you want to create. It must be an absolute path under the /**etc** directory.
  - **mode** - Optional - set the access permission on the directory, in the octal format. If you do not specify a permission, it defaults to 0755. The leading zero is optional.
  - **user** - Optional - set a user as the owner of the directory. If you do not specify a user, it defaults to **root**. You can specify the user as a string or as an integer.
  - **group** - Optional - set a group as the owner of the directory. If you do not specify a group, it defaults to **root**. You can specify the group as a string or as an integer.
  - **ensure_parents** - Optional - Specify whether you want to create parent directories as needed. If you do not specify a value, it defaults to **false**.

- Create customized file under the /**etc** directory for your image by using **[[customizations.directories]]**:

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

The blueprint entries are described as following:

  - **path** - Mandatory - enter the path to the file that you want to create. It must be an absolute path under the /**etc** directory.
  - **mode** Optional - set the access permission on the file, in the octal format. If you do not specify a permission, it defaults to 0644. The leading zero is optional.
  - **user** - Optional - set a user as the owner of the file. If you do not specify a user, it defaults to **root**. You can specify the user as a string or as an integer.

- **group** - Optional - set a group as the owner of the file. If you do not specify a group, it defaults to **root**. You can specify the group as a string or as an integer.
- **data** - Optional - Specify the content of a plain text file. If you do not specify a content, it creates an empty file.

5.4. PACKAGES INSTALLED BY RHEL IMAGE BUILDER

When you create a system image by using RHEL image builder, the system installs a set of base package groups.

NOTE

When you add additional components to your blueprint, ensure that the packages in the components you added do not conflict with any other package components. Otherwise, the system fails to solve dependencies and creating your customized image fails. You can check if there is no conflict between the packages by running the command:

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

Table 5.1. Default packages to support image type creation

Image type	Default Packages
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**
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**

Image type	Default Packages
edge-commit	**redhat-release**, **glibc**, **glibc-minimal-langpack**, **nss-altfiles**, **dracut-config-generic**, **dracut-network**, **basesystem**, **bash**, **platform-python**, **shadow-utils**, **chrony**, **setup**, **shadow-utils**, **sudo**, **systemd**, **coreutils**, **util-linux**, **curl**, **vim-minimal**, **rpm**, **rpm-ostree**, **polkit**, **lvm2**, **cryptsetup**, **pinentry**, **e2fsprogs**, **dosfstools**, **keyutils**, **gnupg2**, **attr**, **xz**, **gzip**, **firewalld**, **iptables**, **NetworkManager**, **NetworkManager-wifi**, **NetworkManager-wwan**, **wpa_supplicant**, **traceroute**, **hostname**, **iproute**, **iputils**, **openssh-clients**, **procps-ng**, **rootfiles**, **openssh-server**, **passwd**, **policycoreutils**, **policycoreutils-python-utils**, **selinux-policy-targeted**, **setools-console**, **less**, **tar**, **rsync**, **usbguard**, **bash-completion**, **tmux**, **ima-evm-utils**, **audit**, **podman**, **containernetworking-plugins**, **container-selinux**, **skopeo**, **criu**, **slirp4netns**, **fuse-overlayfs**, **clevis**, **clevis-dracut**, **clevis-luks**, **greenboot**, **greenboot-default-health-checks**, **fdo-client**, **fdo-owner-cli**, **sos**,
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**

Image type	Default Packages
edge-installer	**aajohan-comfortaa-fonts, abattis-cantarell-fonts, alsa-firmware, alsa-tools-firmware, anaconda, anaconda-install-env-deps, anaconda-widgets, audit, bind-utils, bitmap-fangsongti-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, spice-vdagent, 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**

Image type	Default Packages
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 type	Default Packages
image-installer	**aajohan-comfortaa-fonts**, **abattis-cantarell-fonts**, **alsa-firmware**, **alsa-tools-firmware**, **anaconda**, **anaconda-dracut**, **anaconda-install-env-deps**, **anaconda-widgets**, **audit**, **bind-utils**, **bitmap-fangsongti-fonts**, **bzip2**, **cryptsetup**, **curl**, **dbus-x11**, **dejavu-sans-fonts**, **dejavu-sans-mono-fonts**, **device-mapper-persistent-data**, **dmidecode**, **dnf**, **dracut-config-generic**, **dracut-network**, **efibootmgr**, **ethtool**, **fcoe-utils**, **ftp**, **gdb-gdbserver**, **gdisk**, **glibc-all-langpacks**, **gnome-kiosk**, **google-noto-sans-cjk-ttc-fonts**, **grub2-tools**, **grub2-tools-extra**, **grub2-tools-minimal**, **grubby**, **gsettings-desktop-schemas**, **hdparm**, **hexedit**, **hostname**, **initscripts**, **ipmitool**, **iwl1000-firmware**, **iwl100-firmware**, **iwl105-firmware**, **iwl135-firmware**, **iwl2000-firmware**, **iwl2030-firmware**, **iwl3160-firmware**, **iwl5000-firmware**, **iwl5150-firmware**, **iwl6000g2a-firmware**, **iwl6000g2b-firmware**, **iwl6050-firmware**, **iwl7260-firmware**, **jomolhari-fonts**, **kacst-farsi-fonts**, **kacst-qurn-fonts**, **kbd**, **kbd-misc**, **kdump-anaconda-addon**, **kernel**, **khmeros-base-fonts**, **less**, **libblockdev-lvm-dbus**, **libibverbs**, **libreport-plugin-bugzilla**, **libreport-plugin-reportuploader**, **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**, **mtr**, **mt-st**, **net-tools**, **nfs-utils**, **nmap-ncat**, **nm-connection-editor**, **nss-tools**, **openssh-clients**, **openssh-server**, **oscap-anaconda-addon**, **ostree**, **pciutils**, **perl-interpreter**, **pigz**, **plymouth**, **prefixdevname**, **python3-pyatspi**, **rdma-core**, **redhat-release-eula**, **rng-tools**, **rpcbind**, **rpm-ostree**, **rsync**, **rsyslog**, **selinux-policy-targeted**, **sg3_utils**, **sil-abyssinica-fonts**, **sil-padauk-fonts**, **sil-scheherazade-fonts**, **smartmontools**, **smc-meera-fonts**, **spice-vdagent**, **strace**, **systemd**, **tar**, **thai-scalable-waree-fonts**, **tigervnc-server-minimal**, **tigervnc-server-module**, **udisks2**,**udisks2-iscsi**,**usbutils**,**vim-minimal**, **volume_key**, **wget**, **xfsdump**, **xfsprogs**, **xorg-x11-drivers**, **xorg-x11-fonts-misc**, **xorg-x11-server-utils**, **xorg-x11-server-Xorg**, **xorg-x11-xauth**, **xz**,

Image type	Default Packages
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**

Additional resources

- RHEL image builder description

CHAPTER 6. BUILDING SIMPLIFIED INSTALLER IMAGES TO PROVISION A RHEL FOR EDGE IMAGE

You can build a RHEL for Edge Simplified Installer image, which is optimized for unattended installation to a device, and provision the image to a RHEL for Edge image.

6.1. SIMPLIFIED INSTALLER IMAGE BUILD AND DEPLOYMENT

The RHEL for Edge Simplified Installer image is optimized for unattended installation to a device and supports both network-based deployment and non-network-based deployments. However, for network-based deployment, it supports only UEFI HTTP boot.

Build a RHEL for Edge Simplified Installer image by using the **edge-simplified-installer** image type.

To build a RHEL for Edge Simplified Installer image, provide an existing **OSTree** commit. The resulting RHEL for Edge Simplified Installer contains a raw image that has a deployed OSTree commit.

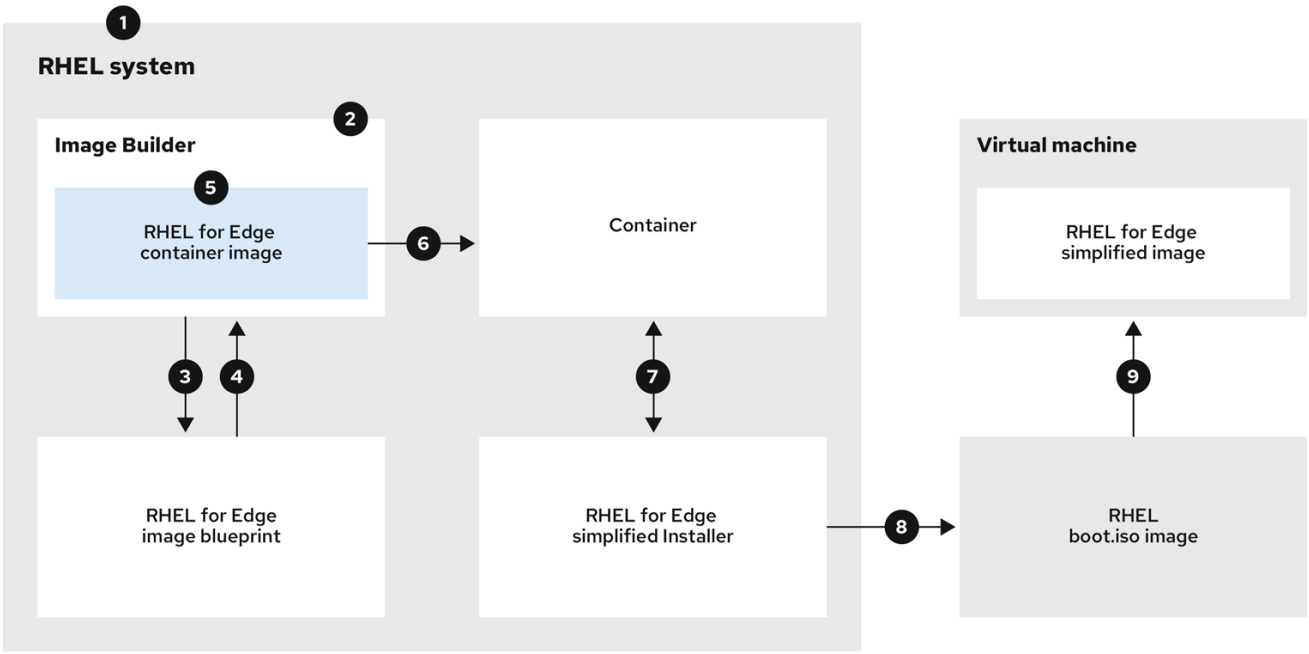
After you boot the Simplified installer ISO image, it provisions a RHEL for Edge system that you can use on a hard disk or as a boot image in a virtual machine. You can log in to the deployed system with the user name and password that you specified in the blueprint that you used to create the Simplified Installer image.

Composing and deploying a simplified RHEL for Edge image involves the following high-level steps:

1. Install and register a RHEL system
2. Install RHEL image builder
3. Using RHEL image builder, create a blueprint with customizations for RHEL for Edge Container image
4. Import the RHEL for Edge blueprint in RHEL image builder
5. Create a RHEL for Edge image embed in an OCI container with a web server ready to deploy the commit as an OSTree repository
6. Create a blueprint for the **edge-simplified-installer** image
7. Build a simplified RHEL for Edge image
8. Download the RHEL for Edge simplified image
9. Install the raw image with the **edge-simplified-installer** virt-install

The following diagram represents the RHEL for Edge Simplified building and provisioning workflow:

Figure 6.1. Building and provisioning RHEL for Edge in network-based environment



6.2. CREATING RHEL FOR EDGE SIMPLIFIED INSTALLER IMAGES BY USING THE CLI

6.2.1. Setting up an UEFI HTTP Boot server

Set up an **UEFI HTTP Boot** server so that you can start to provision a RHEL for Edge Virtual Machine over the network by connecting to this UEFI HTTP Boot server.

Prerequisites

- You have created the ISO simplified installer image.
- An http server that serves the ISO content.

Procedure

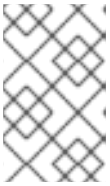
1. Mount the ISO image to the directory of your choice:

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

   Replace **/path_directory/installer.iso** with the path to the RHEL for Edge bootable ISO image.

2. Copy the files from the mounted image to the HTTP server root. This command creates the **/var/www/html/rhel9-install/** directory with the contents of the image.

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



NOTE

Some copying methods can skip the **.treeinfo** file which is required for a valid installation source. Running the **cp** command for whole directories as shown in this procedure will copy **.treeinfo** correctly.

3. Update the **/var/www/html/EFI/BOOT/grub.cfg** file, by replacing:

   a. **coreos.inst.install_dev=/dev/sda** with **coreos.inst.install_dev=/dev/vda**

   b. **linux /images/pxeboot/vmlinuz** with **linuxefi /images/pxeboot/vmlinuz**

   c. **initrd /images/pxeboot/initrd.img** with **initrdefi /images/pxeboot/initrd.img**

   d. **coreos.inst.image_file=/run/media/iso/disk.img.xz** with **coreos.inst.image_url=http://{IP-ADDRESS}/disk.img.xz**
   The *IP-ADDRESS* is the IP address of this machine, which will serve as a http boot server.

4. Start the httpd service:

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

   As a result, after you set up an **UEFI HTTP Boot** server, you can install your RHEL for Edge devices by using **UEFI HTTP** boot.

6.2.2. Creating a blueprint for a Simplified image using RHEL image builder CLI

To create a blueprint for a simplified RHEL for Edge image, you must add the following customizations to the blueprint:

- Customize the blueprint with the **installation_device** customization.
- Add a **device file** location to the blueprint to enable an unattended installation to the device.
- Add a **URL** to perform the initial device credential exchange.
- Customize the blueprint with the **customizations.user** and add with **users** and **user groups** to it.

Procedure

1. Create a plain text file in the Tom's Obvious, Minimal Language (TOML) format, with the following content:

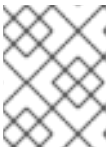
```
name = "simplified-installer-blueprint"
description = "blueprint for the simplified installer image"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[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"
```



NOTE

The FDO customization in the blueprints is optional, and you can build your RHEL for Edge Simplified Installer image with no errors.

- *name* is the name and *description* is the description for your blueprint.
- *0.0.1* is the version number according to the Semantic Versioning scheme.
- *Modules* describe the package name and matching version glob to be installed into the image, for example, the package name = "tmux" and the matching version glob is version = "2.9a". Notice that currently there are no differences between packages and modules.
- *Groups* are packages groups to be installed into the image, for example the **anaconda-tools** group package. If you do not know the modules and groups, leave them empty.
- *installation-device* is the customization to enable an unattended installation to your device.
- *manufacturing_server_url* is the URL to perform the initial device credential exchange.
- *name* is the user name to login to the image.
- *password* is a password of your choice.
- *groups* are any user groups, such as "widget".

2. Push (import) the blueprint to the RHEL image builder server:

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

3. Check whether the created blueprint is successfully pushed and exists.

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

4. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Additional resources

- Composing a RHEL for Edge image using RHEL image builder command-line

6.2.3. Creating a RHEL for Edge Simplified Installer image by using image builder CLI

Create a RHEL for Edge Simplified image by using RHEL image builder command-line interface.

Prerequisites

- You created a blueprint for the RHEL for Edge Simplified image.
- You served an OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo. See ref:setting-up-a-web-server-to-install-rhel-for-edge-image_installing-rpm-ostree-images[Setting up a web server to install RHEL for Edge image].

Procedure

1. Create the bootable ISO image.

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

Where,

- **blueprint-name** is the RHEL for Edge blueprint name.
- **edge-simplified-installer** is the image type .
- **--ref** is the reference for where your commit is going to be created.
- **--url** is the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. You can either start a RHEL for Edge Container or set up a web server. See Creating a RHEL for Edge Container image for non-network-based deployments and Setting up a web server to install RHEL for Edge image .
  A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also keep the UUID number handy for further tasks.

2. Check the image compose status.

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

The output displays the status in the following format:

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

NOTE

The image creation processes can take up to ten minutes to complete.

To interrupt the image creation process, run:

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

To delete an existing image, run:

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

Additional resources

- Composing a RHEL for Edge image using RHEL image builder command-line

6.3. DOWNLOADING A SIMPLIFIED RHEL FOR EDGE IMAGE USING THE IMAGE BUILDER COMMAND-LINE INTERFACE

To download a RHEL for Edge image by using RHEL image builder command-line interface, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You have created a RHEL for Edge image.

Procedure

1. Review the RHEL for Edge image status.

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

   The output must display the following:

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

2. Download the image.

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

   RHEL image builder downloads the image as an **.iso** file at the current directory path where you run the command.

   The UUID number and the image size is displayed alongside.

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

As a result, you downloaded a RHEL for Edge Simplified Installer ISO image. You can use it directly as a boot ISO to install a RHEL for Edge system.

6.4. CREATING RHEL FOR EDGE SIMPLIFIED INSTALLER IMAGES BY USING THE GUI

6.4.1. Creating a blueprint for a Simplified image RHEL using image builder GUI

To create a RHEL for Edge Simplified Installer image, you must create a blueprint and ensure that you customize it with:

- A device node location to enable an unattended installation to your device.
- A URL to perform the initial device credential exchange.

- A user or user group.

NOTE

You can also add any other customizations that your image requires.

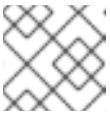
To create a blueprint for a simplified RHEL for Edge image in the RHEL image builder GUI, complete the following steps:

Prerequisites

- You have opened the image builder app from the web console in a browser. See Accessing the RHEL image builder GUI in the RHEL web console.

Procedure

1. Click Create Blueprint in the upper-right corner of the RHEL image builder app.
   A dialog wizard with fields for the blueprint name and description opens.
2. On the **Details** page:
   a. Enter the name of the blueprint and, optionally, its description. Click **Next**.
3. Optional: On the Packages page, complete the following steps:
   a. In the **Available packages** search, enter the package name and click the **>** button to move it to the Chosen packages field. Search and include as many packages as you want. Click **Next**.



NOTE

The customizations are all optional unless otherwise specified.

4. Optional: On the **Kernel** page, enter a kernel name and the command-line arguments.
5. Optional: On the **File system** page, select **Use automatic partitioning**.The filesystem customization is not supported for OSTree systems, because OSTree images have their own mount rule, such as read-only. Click **Next**.
6. Optional: On the **Services** page, you can enable or disable services:
   a. Enter the service names you want to enable or disable, separating them by a comma, by space, or by pressing the **Enter** key. Click **Next**.
7. Optional: On the **Firewall** page, set up your firewall setting:
   a. Enter the **Ports**, and the firewall services you want to enable or disable.
   b. Click the **Add zone** button to manage your firewall rules for each zone independently. Click **Next**.
8. On the **Users** page, add a users by following the steps:
   a. Click **Add user**.

b. Enter a **Username**, a **password**, and a **SSH key**. You can also mark the user as a privileged user, by clicking the **Server administrator** checkbox.

NOTE

When you specify the user in the blueprint customization and then create an image from that blueprint, the blueprint creates the user under the **/usr/lib/passwd** directory and the password under the **/usr/etc/shadow** during installation time. You can log in to the device with the username and password you created for the blueprint. After you access the system, you must create users, for example, using the **useradd** command.

Click **Next**.

9. Optional: On the **Groups** page, add groups by completing the following steps:

   a. Click the **Add groups** button:

      i. Enter a **Group name** and a **Group ID**. You can add more groups. Click **Next**.

10. Optional: On the **SSH keys** page, add a key:

    a. Click the **Add key** button.

       i. Enter the SSH key.

       ii. Enter a **User**. Click **Next**.

11. Optional: On the **Timezone** page, set your timezone settings:

    a. On the **Timezone** field, enter the timezone you want to add to your system image. For example, add the following timezone format: "US/Eastern".
    If you do not set a timezone, the system uses Universal Time, Coordinated (UTC) as default.

    b. Enter the **NTP** servers. Click **Next**.

12. Optional: On the **Locale** page, complete the following steps:

    a. On the **Keyboard** search field, enter the package name you want to add to your system image. For example: ["en_US.UTF-8"].

    b. On the **Languages** search field, enter the package name you want to add to your system image. For example: "us". Click **Next**.

13. Mandatory: On the **Others** page, complete the following steps:

    a. In the **Hostname** field, enter the hostname you want to add to your system image. If you do not add a hostname, the operating system determines the hostname.

    b. Mandatory: In the **Installation Devices** field, enter a valid node for your system image to enable an unattended installation to your device. For example: **dev/sda1**. Click **Next**.

14. Optional: On the **FIDO device onboarding** page, complete the following steps:

    a. On the **Manufacturing server URL** field, enter the **manufacturing server URL** to perform the initial device credential exchange, for example: "http://10.0.0.2:8080". The FDO customization in the blueprints is optional, and you can build your RHEL for Edge Simplified

Installer image with no errors.

b. On the **DIUN public key insecure** field, enter the certification public key hash to perform the initial device credential exchange. This field accepts "true" as value, which means this is an insecure connection to the manufacturing server. For example: **manufacturing_server_url="http://${FDO_SERVER}:8080" diun_pub_key_insecure="true"**. You must use only one of these three options: "key insecure", "key hash" and "key root certs".

c. On the **DIUN public key hash** field, enter the hashed version of your public key. For example: **17BD05952222C421D6F1BB1256E0C925310CED4CE1C4FFD6E5CB968F4B73BF73**. You can get the key hash by generating it based on the certificate of the manufacturing server. To generate the key hash, run the command:

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

The **/etc/fdo/aio/keys/diun_cert.pem** is the certificate that is stored in the manufacturing server.

d. On the **DIUN public key root certs** field, enter the public key root certs. This field accepts the content of the certification file that is stored in the manufacturing server. To get the content of certificate file, run the command:

```
$ cat /etc/fdo/aio/keys/diun_cert.pem.
```

15. Click **Next**.

16. On the **Review** page, review the details about the blueprint. Click **Create**.

The RHEL image builder view opens, listing existing blueprints.

6.4.2. Creating a RHEL for Edge Simplified Installer image using image builder GUI

To create a RHEL for Edge Simplified image by using RHEL image builder GUI, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You opened the RHEL image builder app from the web console in a browser.
- You created a blueprint for the RHEL for Edge Simplified image.
- You served an OSTree repository of the commit to embed in the image, for example, **http://10.0.2.2:8080/repo**. See Setting up a web server to install RHEL for Edge image .
- The FDO manufacturing server is up and running.

Procedure

1. Access mage builder dashboard.
2. On the blueprint table, find the blueprint you want to build an image for.

3. Navigate to the **Images** tab and click **Create Image**. The **Create image** wizard opens.
4. On the **Image output** page, complete the following steps:
    a. From the **Select a blueprint** list, select the blueprint you created for the RHEL for Edge Simplified image.
    b. From the **Image output type** list, select **RHEL for Edge Simplified Installer (.iso)**.
    c. In the **Image Size** field, enter the image size. Minimum image size required for Simplified Installer image is:
5. Click **Next**.
6. In the **OSTree settings** page, complete the following steps:
    a. In the **Repository URL** field, enter the repository URL to where the parent OSTree commit will be pulled from.
    b. In the **Ref** field, enter the **ref** branch name path. If you do not enter a **ref**, the default **ref** for the distro is used.
7. On the **Review** page, review the image customization and click **Create**.

The image build starts and takes up to 20 minutes to complete. To stop the building, click **Stop build**.

6.4.3. Downloading a simplified RHEL for Edge image using the image builder GUI

To download a RHEL for Edge image by using RHEL image builder GUI, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- You have successfully created a RHEL for Edge image. See link.

Procedure

1. Access the RHEL image builder dashboard. The blueprint list dashboard opens.
2. In the blueprint table, find the blueprint you built your RHEL for Edge Simplified Installer image for.
3. Navigate to the **Images** tab.
4. Choose one of the options:
    - Download the image.
    - Download the logs of the image to inspect the elements and verify if any issue is found.

NOTE

You can use the RHEL for Edge Simplified Installer ISO image that you downloaded directly as a boot ISO to install a RHEL for Edge system.

6.5. PROVISIONING THE RHEL FOR EDGE SIMPLIFIED INSTALLER IMAGE

6.5.1. Deploying the Simplified ISO image in a Virtual Machine

Deploy the RHEL for Edge ISO image you generated by creating a RHEL for Edge Simplified image by using any the following installation sources:

- **UEFI HTTP Boot**
- **virt-install**

This example shows how to create a **virt-install** installation source from your ISO image for a **network-based** installation .

Prerequisites

- You have created an ISO image.
- You set up a network configuration to support UEFI HTTP boot.

Procedure

1. Set up a network configuration to support **UEFI HTTP** boot. See Setting up UEFI HTTP boot with libvirt.
2. Use the **virt-install** command to create a RHEL for Edge Virtual Machine from the UEFI HTTP Boot.

```
# 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
```

After you run the command, the Virtual Machine installation starts.

Verification

- Log in to the created Virtual Machine.

6.5.2. Deploying the Simplified ISO image from a USB flash drive

Deploy the RHEL for Edge ISO image you generated by creating a RHEL for Edge Simplified image by using an **USB installation**.

This example shows how to create a **USB installation** source from your ISO image.

Prerequisites

- You have created a simplified installer image, which is an ISO image.
- You have a 8 GB USB flash drive.

Procedure

1. Copy the ISO image file to a USB flash drive.
2. Connect the USB flash drive to the port of the computer you want to boot.
3. Boot the ISO image from the USB flash drive.The boot menu shows you the following options:

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

4. Choose Install Red Hat Enterprise Linux 9. This starts the system installation.

Additional resources

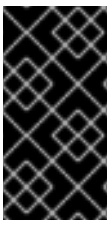
- Booting the installation

6.6. CREATING AND BOOTING A RHEL FOR EDGE IMAGE IN FIPS MODE

You can create and boot a FIPS-enabled RHEL for Edge image. Before you compose the image, you must change the value of the **fips** directive in your blueprint.

You can build the following image types in FIPS mode:

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



IMPORTANT

You can enable FIPS mode only during the image provisioning process. You cannot change to FIPS mode after the non-FIPS image build starts. If the host that builds the FIPS-enabled image is not FIPS-enabled, any keys generated by this host are not FIPS-compliant, but the resulting image is FIPS-compliant.

Prerequisites

- You created and downloaded a RHEL for Edge Container OSTree commit.
- You have installed Podman on your system. See the Red Hat Knowledgebase solution How to install Podman in RHEL.
- You are logged in as the root user or a user who is a member of the **weldr** group.

Procedure

1. Create a plain text file in the Tom's Obvious, Minimal Language (TOML) format with the following content:

```
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. Import the blueprint to the RHEL image builder server:

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

3. List the existing blueprints to check whether the created blueprint is successfully imported and exists:

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

4. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

5. Serve an OSTree repository of the commit to embed in the image, for example, **http://10.0.2.2:8080/repo**. For more information, see Setting up an UEFI HTTP Boot server.

6. Create the bootable ISO image:

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

7. Review the RHEL for Edge image status:

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

8. Download the image:

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

RHEL image builder downloads the image as an **.iso** file to the current directory path. The UUID number and the image size are displayed alongside:

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

9. Create a RHEL for Edge virtual machine from the UEFI HTTP Boot server, for example:

```
# 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
```

After you enter the command, the virtual machine installation starts.

Verification

1. Log in to the created virtual machine with the username and password that you configured in your blueprint.
2. Check if FIPS mode is enabled:

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

CHAPTER 7. BUILDING AND PROVISIONING A MINIMAL RAW IMAGE

The **minimal-raw** image is a pre-packaged, bootable, minimal RPM image, compressed in the **xz** format. The image consists of a file containing a partition layout with an existing deployed OSTree commit in it. You can build a RHEL for Edge Minimal Raw image type by using RHEL image builder and deploy the Minimal Raw image to the **aarch64** and **x86** architectures.

7.1. THE MINIMAL RAW IMAGE BUILD AND DEPLOYMENT

Build a RHEL for Edge Minimal Raw image by using the **minimal-raw** image type. To boot the image, you must decompress it and copy to any bootable device, such as an SD card or a USB flash drive. You can log in to the deployed system with the user name and password that you specified in the blueprint that you used to create the RHEL for Edge Minimal Raw image.

Composing and deploying a RHEL for Edge Minimal Raw image involves the following high-level steps:

1. Install and register a RHEL system
2. Install RHEL image builder
3. Using RHEL image builder, create a blueprint with your customizations for RHEL for Edge Minimal Raw image
4. Import the RHEL for Edge blueprint in RHEL image builder
5. Create a RHEL for Edge Minimal Raw image
6. Download and decompress the RHEL for Edge Minimal Raw image
7. Create a bootable USB drive from the decompressed Raw image
8. Deploy the RHEL for Edge Minimal Raw image

7.2. CREATING THE BLUEPRINT FOR A MINIMAL RAW IMAGE BY USING RHEL IMAGE BUILDER CLI

Create a blueprint, and customize it with a username and a password. You can use the resulting blueprint to create a Minimal Raw image and log in to it by using the credentials that you configured in the blueprint.

Procedure

1. Create a plain text file in the Tom's Obvious, Minimal Language (TOML) format, with the following content:

```
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 is the name and description is the description for your blueprint.
- 0.0.1 is the version number according to the Semantic Versioning scheme.
- Modules describe the package name and matching version glob to be installed into the image, for example, the package name = "tmux" and the matching version glob is version = "2.9a". Currently there are no differences between packages and modules.
- Groups are packages groups to be installed into the image, for example the anaconda-tools group package. If you do not know the modules and groups, leave them empty.
- Under **customizations.user**:
  - **name** is the username to login to the image
  - **password** is a password of your choice
  - **groups** are any user groups, such as "widget"

2. Import the blueprint to the RHEL image builder server:

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

3. Check if the blueprint is available on the system:

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

4. Check the validity of components, versions, and their dependencies in the blueprint:

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

Additional resources

- Composing a RHEL for Edge image by using RHEL image builder command-line

7.3. CREATING A MINIMAL RAW IMAGE BY USING RHEL IMAGE BUILDER CLI

Create a RHEL for Edge Minimal Raw image with the RHEL image builder command-line interface.

Prerequisites

- You created a blueprint for the RHEL for Edge Minimal Raw image.

Procedure

1. Build the image.

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

- ***<blueprint_name>*** is the RHEL for Edge blueprint name
- **minimal-raw** is the image type

2. Check the image compose status.

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

The output displays the status in the following format:

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

Additional resources

- Composing a RHEL for Edge image using image builder command-line

7.4. DOWNLOADING AND DECOMPRESSING THE MINIMAL RAW IMAGE

Download the RHEL for Edge Minimal Raw image by using RHEL image builder command-line interface, and then decompress the image to be able to boot it.

Prerequisites

- You have created a RHEL for Edge Minimal Raw image.

Procedure

1. Review the RHEL for Edge Minimal Raw image compose status.

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

The output must display the following details:

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

2. Download the image:

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

Image builder downloads the image into your working directory. The following output is an example:

```
3f9223c1-6ddb-4915-92fe-9e0869b8e209-raw.img.xz
```

3. Decompress the image:

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

Next

Use the decompressed bootable RHEL for Edge Minimal Raw image to create a bootable installation medium and use it as a boot device. The following documentation describes the procedure of creating an USB bootable device from an ISO image. However, the same steps apply to the RAW images, because the RAW image is equivalent to the ISO image.

See Creating a bootable USB device on Linux for more details.

7.5. DEPLOYING THE MINIMAL RAW IMAGE FROM A USB FLASH DRIVE

After you created a bootable USB installation medium from the customized RHEL for Edge Minimal Raw image, you can continue the installation process by deploying the Minimal Raw image from the USB flash drive and booting your customized image.

Prerequisites

- You have a 8 GB USB flash drive.
- You have created a bootable installation medium from the RHEL for Edge Minimal Raw image to the USB drive.

Procedure

1. Connect the USB flash drive to the computer where you want to boot your customized image.
2. Power on the system.
3. Boot the RHEL for Edge Minimal Raw image from the USB flash drive. The boot menu shows you the following options:

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

4. Choose **Install Red Hat Enterprise Linux 9** This starts the system installation.

Verification

1. Boot into the image by using the username and password you configured in the blueprint.
   a. Check the release:

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

   b. List the block devices in the system:

```
$ lsblk
```

7.6. SERVING A RHEL FOR EDGE CONTAINER IMAGE TO BUILD A RHEL FOR EDGE RAW IMAGE

Create a RHEL for Edge Container image to serve it to a running container.

Prerequisites

- You have created a RHEL for Edge Minimal Raw image and downloaded it.

Procedure

1. Create a blueprint for the **rhel-edge-container** image type, for example:

```
name = "rhel-edge-container-no-users"
description = ""
version = "0.0.1"
```

2. Build a **rhel-edge-container** image:

```
# composer-cli compose start-ostree <rhel-edge-container-no-users> rhel-edge-container
```

3. Check if the image is ready:

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

4. Download the **rhel-edge-container** image as a **.tar** file:

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

5. Import the RHEL for Edge Container into Podman:

```
$ skopeo copy oci-archive:_<UUID>_-container.tar \
containers-storage:localhost/rfe-93-mirror:latest
```

6. Start the container and make it available by using the port 8080:

```
$ podman run -d --rm --name <rfe-93-mirror> -p 8080:8080 localhost/<rfe-93-mirror>
```

7. Create a blueprint for the **edge-raw-image** image type, for example:

```
name = "<edge-raw>"
description = ""
version = "0.0.1"
[[customizations.user]]
name = "admin"
password = "admin"
groups = ["wheel"]
```

8. Build a RHEL for Edge Raw image by serving the RHEL Edge Container to it:

```
# composer-cli compose start-ostree edge-raw edge-raw-image \
--url http://10.88.0.1:8080/repo
```

9. Download the RHEL for Edge Raw image as a **.raw** file:

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

10. Decompress the RHEL for Edge Raw image:

```
# xz --decompress <UUID>>-image.raw.xz
```

CHAPTER 8. USING THE IGNITION TOOL FOR THE RHEL FOR EDGE SIMPLIFIED INSTALLER IMAGES

RHEL for Edge uses the Ignition tool to inject the user configuration into the images at an early stage of the boot process. The user configuration that the Ignition tool injects includes:

- The user configuration.
- Writing files, such as regular files, and **systemd** units.

On the first boot, Ignition reads its configuration either from a remote URL or a file embedded in the simplified installer ISO. Then, Ignition applies that configuration into the image.

8.1. CREATING AN IGNITION CONFIGURATION FILE

The **Butane** tool is the preferred option to create an Ignition configuration file. **Butane** consumes a **Butane Config YAML** file and produces an **Ignition Config** in the JSON format. The JSON file is used by a system on its first boot. The **Ignition Config** applies the configuration in the image, such as user creation, and **systemd** units installation.

Prerequisites

- You have installed the Butane tool version v0.17.0:

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

Procedure

1. Create a **Butane Config** file and save it in the **.bu** format. You must specify the **variant** entry as **r4e**, and the **version** entry as **1.0.0**, for RHEL for Edge images. The butane **r4e** variant on version 1.0.0 targets Ignition spec version **3.3.0**. The following is a Butane Config YAML file example:

```
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
```

2. Run the following command to consume the **Butane Config YAML** file and generate an Ignition Config in the JSON format:

```
$ ./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,H4sIAAAAAAAAC/0yKUcrCMBAG3/csf/ObUKQie5
LShyX5SgPNNiSr0NuLgiDzNMPM8VBFtHzoQjkxtPp+ITsrGLahKYyyGtoqEYNKwfeZc32OC0l
KDb179rfg/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"}]}}
```

After you run the **Butane Config YAML** file to check and generate the **Ignition Config JSON** file, you might get warnings when using unsupported fields, like partitions, for example. You can fix those fields and rerun the check.

You have now an Ignition JSON configuration file that you can use to customize your blueprint.

Additional resources

- RHEL for Edge Specification v1.0.0

8.2. CREATING A BLUEPRINT IN THE GUI WITH SUPPORT TO IGNITION

When building a Simplified Installer image, you can customize your blueprint by entering the following details in the Ignition page of the blueprint:

- **Firstboot URL** - You must enter a URL that points to the Ignition configuration that will be fetched during the first boot. It can be used for both the raw image and simplified installer image.
- **Embedded Data** - You must provide the **base64** encoded **Ignition Configuration** file. It can be used only for the Simplified Installer image.

To customize your blueprint for a simplified RHEL for Edge image with support to Ignition configuration using the Ignition blueprint customization, follow the steps:

Prerequisites

- You have opened the image builder app from web console in a browser. See Accessing the image builder GUI in the RHEL web console.
- To fully support the embedded section, **coreos-installer-dracut** has to be able to define **-ignition-url|-ignition-file** based on the presence of the OSBuild's file.

Procedure

1. Click **Create Blueprint** in the upper-right corner.
   A dialog wizard with fields for the blueprint name and description opens.
   - On the **Details** page:
     - Enter the name of the blueprint and, optionally, its description. Click **Next**.
   - On the **Ignition** page, complete the following steps:
     - On the **Firstboot URL** field, enter the URL that points to the Ignition configuration to be fetched during the first boot.
     - On the **Embedded Data** field, drag or upload the **base64** encoded **Ignition Configuration** file. Click **Next**.
   - Review the image details and click **Create**.

The image builder dashboard view opens, listing the existing blueprints.

Next

- You can use the blueprint you created to build your Simplified Installer image. See Creating a RHEL for Edge Simplified Installer image using image builder CLI.

8.3. CREATING A BLUEPRINT WITH SUPPORT TO IGNITION USING THE CLI

When building a simplified installer image, you can customize your blueprint by adding the **customizations.ignition** section to it. With that, you can create either a simplified installer image or a raw image that you can use for the bare metal platforms. The **customizations.ignition** customization in the blueprint enables the configuration files to be used in **edge-simplified-installer** ISO and **edge-raw-image** images.

- For the **edge-simplified-installer** ISO image, you can customize the blueprint to embed an Ignition configuration file that will be included in the ISO image. For example:

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

  You must provide a **base64** encoded Ignition configuration file.

- For both the **edge-simplified-installer** ISO image and also the **edge-raw-image**, you can customize the blueprint, by defining a URL that will be fetched to obtain the Ignition configuration at the first boot. For example:

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

  You must enter a URL that points to the Ignition configuration that will be fetched during the first boot.

To customize your blueprint for a Simplified RHEL for Edge image with support to Ignition configuration, follow the steps:

Prerequisites

- If using the **[customizations.ignition.embedded]** customization, you must create an Ignition configuration file.
- If using the **[customizations.ignition.firstboot]** customization, you must have created a container whose URL points to the Ignition configuration that will be fetched during the first boot.
- The blueprint customization **[customizations.ignition.embedded]** section enables **coreos-installer-dracut** to define **-ignition-url**|**-ignition-file** based on the presence of the osbuild's file.

Procedure

1. Create a plain text file in the Tom's Obvious, Minimal Language (TOML) format, with the following content:

   ```
   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="
   ```

Where:

- The **name** is the name and **description** is the description for your blueprint.
- The **version** is the version number according to the Semantic Versioning scheme.
- The **modules** and **packages** describe the package name and matching version glob to be installed into the image. For example, the package **name = "tmux"** and the matching version glob is **version = "3.3a"**. Notice that currently there are no differences between packages and modules.
- The **groups** are packages groups to be installed into the image. For example **groups = "anaconda-tools"** group package. If you do not know the modules and groups, leave them empty.

WARNING

If you want to create a user with Ignition, you cannot use the FDO customizations to create a user at the same time. You can create users using Ignition and copy configuration files using FDO. But if you are creating users, create them using Ignition or FDO, but not both at the same time.

2. Push (import) the blueprint to the image builder server:

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

3. List the existing blueprints to check whether the created blueprint is successfully pushed and exists.

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

4. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Next

- You can use the blueprint you created to build your Simplified Installer image. See Creating a RHEL for Edge Simplified Installer image using image builder CLI.

Additional resources

- RHEL for Edge Specification v1.0.0

CHAPTER 9. CREATING VMDK IMAGES FOR RHEL FOR EDGE

You can create a **.vmdk** image for RHEL for Edge by using the RHEL image builder. You can create an **edge-vsphere** image type with Ignition support, to inject the user configuration into the image at an early stage of the boot process. Then, you can load the image on vSphere and boot the image in a vSphere VM. The image is compatible with ESXi 7.0 U2, ESXi 8.0 and later. The vSphere VM is compatible with version 19 and 20.

9.1. CREATING A BLUEPRINT WITH THE IGNITION CONFIGURATION

Create a blueprint for the **.vmdk** image and customize it with the **customizations.ignition** section. With that, you can create your image and, at boot time, the operating system will inject the user configuration to the image.

Prerequisites

- You have created an Ignition configuration file. For example:

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

Procedure

1. Create a blueprint in the Tom's Obvious, Minimal Language (TOML) format, with the following content:

```
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
```

Where:

- The **name** is the name and **description** is the description for your blueprint.
- The **version** is the version number according to the Semantic Versioning scheme.
- The **modules** and **packages** describe the package name and matching version glob to be installed into the image. For example, the package **name = "open-vm-tools"**. Notice that currently there are no differences between packages and modules.
- The **groups** are packages groups to be installed into the image. For example **groups = "anaconda-tools"** group package. If you do not know the modules and groups, leave them empty.
- The **customizations.user** creates a username and password to log in to the VM.
- The **customizations.ignition.firstboot** contains the URL where the Ignition configuration file is being served.

NOTE

By default, the **open-vm-tools** package is not included in the **edge-vsphere** image. If you need this package, you must include it in the blueprint customization.

2. Import the blueprint to the image builder server:

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

3. List the existing blueprints to check whether the created blueprint is successfully pushed and exists:

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

4. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Next steps

- Use the blueprint you created to build your **.vmdk** image.

9.2. CREATING A VMDK IMAGE FOR RHEL FOR EDGE

To create a RHEL for Edge **.vmdk** image, use the 'edge-vsphere' image type in the RHEL image builder command-line interface.

Prerequisites

- You created a blueprint for the **.vmdk** image.
- You served an OSTree repository of the commit to embed it in the image. For example, **http://10.0.2.2:8080/repo**. For more details, see Setting up a web server to install RHEL for Edge image.

Procedure

1. Start the compose of a **.vmdk** image:

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

   The *--<url>* is the URL of your repo, for example: **http://10.88.0.1:8080/repo**.

   A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also, keep the UUID number handy for further tasks.

2. Check the image compose status:

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

   The output displays the status in the following format:

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

3. After the compose process finishes, download the resulting image file:

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

Next steps

- Upload the **.vmdk** image to vSphere.

9.3. UPLOADING VMDK IMAGES AND CREATING A RHEL VIRTUAL MACHINE IN VSPHERE

Upload the **.vmdk** image to VMware vSphere by using the **govc import.vmdk** CLI tool and boot the image in a VM.

Prerequisites

- You created an **.vmdk** image by using RHEL image builder and downloaded it to your host system.
- You installed the **govc import.vmdk** CLI tool.
- You configured the **govc import.vmdk** CLI tool client.
  - You must set the following values in the environment:

```
GOVC_URL
GOVC_DATACENTER
```

```
GOVC_FOLDER
GOVC_DATASTORE
GOVC_RESOURCE_POOL
GOVC_NETWORK
```

Procedure

1. Navigate to the directory where you downloaded your **.vmdk** image.
2. Launch the image on vSphere by executing the following steps:
   a. Import the **.vmdk** image in to vSphere:

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

   b. Create the VM in vSphere without powering it on:

```
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. Power-on the VM:

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

   d. Retrieve the VM IP address:

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

   e. Use SSH to log in to the VM, using the username and password you specified in your blueprint:

```
$ ssh admin@HOST
```

CHAPTER 10. CREATING RHEL FOR EDGE AMI IMAGES

You can create a RHEL for Edge **edge-ami** customized image by using RHEL image builder. The RHEL for Edge **edge-ami** has Ignition support to inject the user configuration into the images at an early stage of the boot process. Then, you can upload the image to AWS cloud and launch an EC2 instance in AWS. You can use the AMI image type on AMD or Intel 64-bit architectures.

10.1. CREATING A BLUEPRINT FOR EDGE AMI IMAGES

Create a blueprint for the **edge-ami** image and customize it with the **customizations.ignition** section. With that, you can create your image and when booting the image, inject the user configuration.

Prerequisites

- You have created an Ignition configuration file. For example:

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

  For more details, see Creating an Ignition configuration file .

Procedure

1. Create a blueprint in the Tom's Obvious, Minimal Language (TOML) format, with the following content:

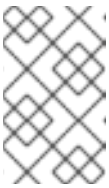
```
name = "ami-edge-image"
description = "Blueprint for Edge AMI image"
version = "0.0.1"
packages = ["cloud-init"]
modules = []
groups = []
distro = ""

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

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

Where:

- The **name** is the name and **description** is the description for your blueprint.
- The **version** is the version number according to the Semantic Versioning scheme.
- The **modules** and **packages** describe the package name and matching version glob to be installed into the image. For example, the package **name = "open-vm-tools"**. Notice that currently there are no differences between packages and modules.
- The **groups** are packages groups to be installed into the image. For example **groups = "wheel"**. If you do not know the modules and groups, leave them empty.
- The **customizations.user** creates a username and password to log in to the VM.
- The **customizations.ignition.firstboot** contains the URL where the Ignition configuration file is being served.



NOTE

By default, the **open-vm-tools** package is not included in the **edge-vsphere** image. If you need this package, you must include it in the blueprint customization.

2. Import the blueprint to the image builder server:

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

3. List the existing blueprints to check whether the created blueprint is successfully pushed and exists:

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

4. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

Next steps

- Use the blueprint you created to build your **edge-ami** image.

10.2. CREATING A RHEL FOR EDGE AMI IMAGE

Create a RHEL for Edge **edge-ami** image in the RHEL image builder command-line interface.

Prerequisites

- You created a blueprint for the **edge-ami** image.

- You served an OSTree repository of the commit to embed it in the image. For example, **http://10.0.2.2:8080/repo**. For more details, see Setting up a web server to install RHEL for Edge image.

Procedure

1. Start the compose of a **edge-ami** image:

```
# composer-cli compose start-ostree <blueprint-name> edge-ami --ref rhel/9/x86_64/edge --url <ostree repo url>
```

   The ***<ostree repo url>*** is the URL of your repo, for example: **http://10.88.0.1:8080/{*<blueprint-name>*}/repo**.

   A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also, keep the UUID number handy for further tasks.

2. Check the image compose status:

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

   The output displays the status in the following format:

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

3. After the compose process finishes, download the resulting image file:

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

Next steps

- Upload the **edge-ami** image to AWS

10.3. UPLOADING A RHEL EDGE AMI IMAGE TO AWS

Upload the **edge-ami** image to Amazon AWS Cloud service provider by using the CLI.

Prerequisites

- You have an **Access Key ID** configured in the AWS IAM account manager.
- You have a writable S3 bucket prepared.
- You have created the required roles for your AWS bucket.
- You have the **aws-cli** tool installed .

Procedure

1. Configure the **aws-cli** tool:

```
$ aws configure
```

a. Configure your profile. Run the command and enter your Access key ID credential, Secret access key, Default region name, and default output name:

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

2. List the existing buckets:

```
$ aws s3 ls
```

3. Upload your image to S3:

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

4. List the image in the S3 bucket:

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

5. Create a **container-simple.json** file. Replace the "URL" content with the S3 bucket. For example: **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. Import the **edge.ami** image to the S3 bucket as an EC2 snapshot.

NOTE

The EC2 image must be in the same region that you have created the S3 bucket.

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

The following .**json** is an example of the command output:

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

7. Take note of "ImportTaskId" value from the **.json** file. Use it to check the import status. In this example, the "ImportTaskId" is **import-snap-0f3055c4b7a454c85**.

8. Check the import status of the snapshot, by using the "ImportTaskId" value from the output **.json** file from the previous step:

```
$ 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": []
        }
    ]
}
```


Run this command until the "Status" is marked as "completed".

9. Register the **edge.ami** image to be able to launch an instance from the image.

```
$ aws ec2 register-image \ --name ami-edge-name-ami-x86" \ --architecture <architecture> \ --tag-specifications 'ResourceType=image,Tags=\\\{Key=Name,Value={{ blueprint_name }}-ami-x86}' \ --root-device-name /dev/sda1 \ --block-device-mappings DeviceName=/dev/sda1,Ebs={SnapshotId={{ snapshot_id }}} DeviceName=/dev/sdh,Ebs={VolumeSize=10} \ --boot-mode uefi-preferred --query="ImageId" --output=text \ --region="{{ aws_region }}" i
```

If you do not specify an architecture, it defaults to the **i386** architecture.

After registering the image, you can access EC2 to create the AMI image from the snapshot, and launch it.

Verification

To confirm that the image upload was successful:

1. Access EC2 in the menu and select the correct region in the AWS console. The image must have the available status, to indicate that it was successfully uploaded.

2. On the dashboard, select your image and click Launch.
When launching the new instance, you must select UEFI as the boot mode, and choose at least 4GB of RAM for the EC2 image.

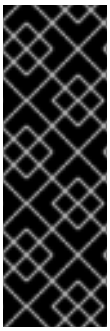
3. You can log in into the **edge-ami** on AWS by using the username and password you created with the Ignition configuration.

Additional resources

- Required service role to import a VM

CHAPTER 11. AUTOMATICALLY PROVISIONING AND ONBOARDING RHEL FOR EDGE DEVICES WITH FDO

You can build a RHEL for Edge Simplified Installer image, and provision it to a RHEL for Edge image. The FIDO Device Onboarding (FDO) process automatically provisions and onboards your Edge devices, and exchanges data with other devices and systems connected on the networks.



IMPORTANT

Red Hat provides the **FDO** process as a Technology Preview feature and should run on secure networks. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

11.1. THE FIDO DEVICE ONBOARDING (FDO) PROCESS

The FIDO Device Onboarding (FDO) is the process that:

- Provisions and onboards a device.
- Automatically configures credentials for this device. The FDO process is an automatic onboarding mechanism that is triggered by the installation of a new device.
- Enables this device to securely connect and interact on the network.

With FIDO Device Onboarding (FDO), you can perform a secure device onboarding by adding new devices into your IoT architecture. This includes the specified device configuration that needs to be trusted and integrated with the rest of the running systems. The FDO process is an automatic onboarding mechanism that is triggered by the installation of a new device.

The FDO protocol performs the following tasks:

- Solves the trust and chain of ownership along with the automation needed to securely onboard a device at scale.
- Performs device initialization at the manufacturing stage and late device binding for its actual use. This means that actual binding of the device to a management system happens on the first boot of the device without requiring manual configuration on the device.
- Supports automated secure devices onboarding, that is, zero touch installation and onboarding that does not need any specialized person at the edge location. After the device is onboarded, the management platform can connect to it and apply patches, updates, and rollbacks.

With FDO, you can benefit from the following:

- FDO is a secure and simple way to enroll a device to a management platform. Instead of embedding a Kickstart configuration to the image, FDO applies the device credentials during the device first boot directly to the ISO image.
- FDO solves the issue of late binding to a device, enabling any sensitive data to be shared over a secure FDO channel.

- FDO cryptographically identifies the system identity and ownership before enrolling and passing the configuration and other secrets to the system. That enables non-technical users to power-on the system.

To build a RHEL for Edge Simplified Installer image and automatically onboard it, provide an existing OSTree commit. The resulting simplified image contains a raw image that has the OSTree commit deployed. After you boot the Simplified installer ISO image, it provisions a RHEL for Edge system that you can use on a hard disk or as a boot image in a virtual machine.

The RHEL for Edge Simplified Installer image is optimized for unattended installation to a device and supports both network-based deployment and non-network-based deployments. However, for network-based deployment, it supports only UEFI HTTP boot.

The FDO protocol is based on the following servers:

Manufacturing server

1. Generates the device credentials.
2. Creates an Ownership voucher that is used to set the ownership of the device, later in the process.
3. Binds the device to a specific management platform.

Owner management system

1. Receives the Ownership voucher from the Manufacturing server and becomes the owner of the associated device.
2. Later in the process, it creates a secure channel between the device and the Owner onboarding server after the device authentication.
3. Uses the secure channel to send the required information, such as files and scripts for the onboarding automation to the device.

Service-info API server

Based on Service-info API server's configuration and modules available on the client, it performs the final steps of onboarding on target client devices, such as copying SSH keys and files, executing commands, creating users, encrypting disks and so on

Rendezvous server

1. Gets the Ownership voucher from the Owner management system and makes a mapping of the device UUID to the Owner server IP. Then, the Rendezvous server matches the device UUID with a target platform and informs the device about which Owner onboarding server endpoint this device must use.
2. During the first boot, the Rendezvous server will be the contact point for the device and it will direct the device to the owner, so that the device and the owner can establish a secure channel.

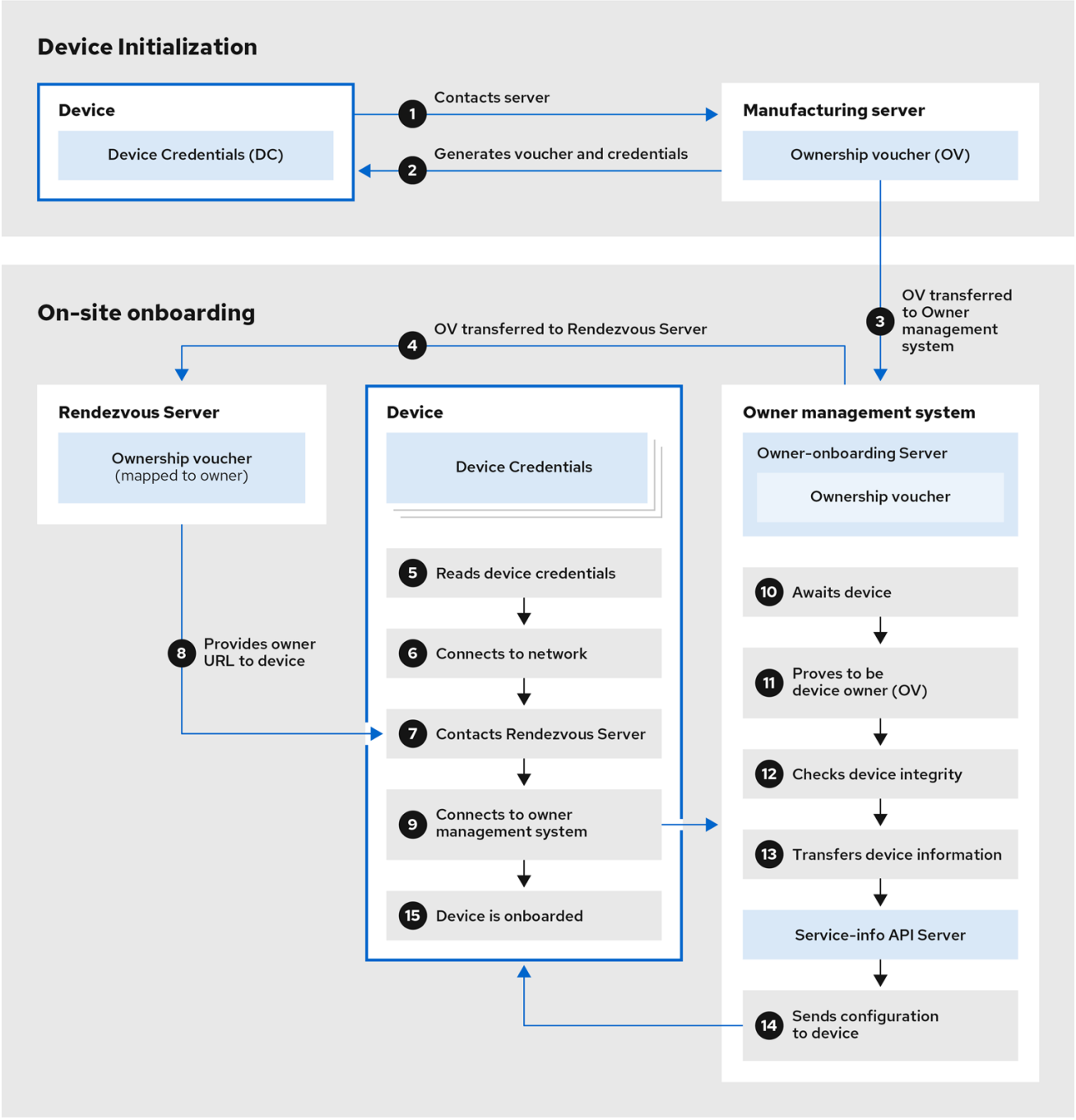
Device client

This is installed on the device. The Device client performs the following actions:

1. Starts the queries to the multiple servers where the onboarding automation will be executed.
2. Uses TCP/IP protocols to communicate with the servers.

The following diagram represents the FIDO device onboarding workflow:

Figure 11.1. Deploying RHEL for Edge in non-network environment



At the **Device Initialization**, the device contacts the Manufacturing server to get the FDO credentials, a set of certificates and keys to be installed on the operating system with the Rendezvous server endpoint (URL). It also gets the Ownership Voucher, that is maintained separately in case you need to change the owner assignment.

1. The Device contacts the Manufacturing server
2. The Manufacturing server generates an Ownership Voucher and the Device Credentials for the Device.
3. The Ownership Voucher is transferred to the Owner onboarding server.

At the **On-site onboarding**, the Device gets the Rendezvous server endpoint (URL) from its device credentials and contacts Rendezvous server endpoint to start the onboarding process, which will redirect it to the Owner management system, that is formed by the Owner onboarding server and the Service Info API server.

4. The Owner onboarding server transfers the Ownership Voucher to the Rendezvous server, which makes a mapping of the Ownership Voucher to the Owner.
5. The device client reads device credentials.
6. The device client connects to the network.
7. After connecting to the network, the Device client contacts the Rendezvous server.
8. The Rendezvous server sends the owner endpoint URL to the Device Client, and registers the device.
9. The Device client connects to the Owner onboarding server shared by the Rendezvous server.
10. The Device proves that it is the correct device by signing a statement with a device key.
11. The Owner onboarding server proves itself correct by signing a statement with the last key of the Owner Voucher.
12. The Owner onboarding server transfers the information of the Device to the Service Info API server.
13. The Service info API server sends the configuration for the Device.
14. The Device is onboarded.

11.2. AUTOMATICALLY PROVISIONING AND ONBOARDING RHEL FOR EDGE DEVICES

To build a RHEL for Edge Simplified Installer image and automatically onboard it, provide an existing OSTree commit. The resulting simplified image contains a raw image that has the OSTree commit deployed. After you boot the Simplified installer ISO image, it provisions a RHEL for Edge system that you can use on a hard disk or as a boot image in a virtual machine.

The RHEL for Edge Simplified Installer image is optimized for unattended installation to a device and supports both network-based deployment and non-network-based deployments. However, for network-based deployment, it supports only UEFI HTTP boot.

Automatically provisioning and onboarding a RHEL for Edge device involves the following high-level steps:

1. Install and register a RHEL system
2. Install RHEL image builder
3. By using RHEL image builder, create a blueprint with customizations for RHEL for a **rhel-edge-container** image type.

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

4. Import the RHEL for Edge Container blueprint in RHEL image builder
5. Create a RHEL for Edge Container image
6. Use the RHEL for Edge Container image to serve the OSTree commit, which will be later used when building the RHEL for Edge Simplified Installer image type
7. Create a blueprint for and **edge-simplified-installer** image type with customizations for storage device path and FDO customizations

```
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. Build a simplified installer RHEL for Edge image
9. Download the RHEL for Edge simplified installer image
10. At this point, the FDO server infrastructure should be up and running, and the specific onboarding details handled by the **service-info API** server, that is part of the owner's infrastructure, are configured
11. Install the simplified installer ISO image to a device. The FDO client runs on the Simplified Installer ISO and the UEFI directory structure makes the image bootable.
12. The network configuration enables the device to reach out to the manufacturing server to perform the initial device credential exchange.
13. After the system reaches the endpoint, the device credentials are created for the device.
14. The device uses the device credentials to reach the Rendezvous server, where it checks the cryptographic credentials based on the vouchers that the Rendezvous server has, and then the Rendezvous server redirects the device to the Owner server.
15. The device contacts the Owner server. They establish a mutual trust and the final steps of onboarding happen based on the configuration of the Service-info API server. For example, it installs the SSH keys in the device, transfer the files, create the users, run the commands, encrypt the filesystem, and so on.

Additional resources

- FDO automatic onboarding technologies

11.3. GENERATING KEY AND CERTIFICATES

To run the FIDO Device Onboarding (FDO) infrastructure, you need to generate keys and certificates. FDO generates these keys and certificates to configure the manufacturing server. FDO automatically generates the certificates and **.yaml** configuration files when you install the services, and re-creating them is optional. After you install and start the services, it runs with the default settings.

IMPORTANT

Red Hat provides the **fdo-admin-tool** tool as a Technology Preview feature and should run on secure networks. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

Prerequisites

- You installed the **fdo-admin-cli** RPM package

Procedure

1. Generate the keys and certificates in the **/etc/fdo** directory:

```
$ 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. Check the key and certificates that were created in the **/etc/fdo/keys** directory:

```
$ tree keys
```

You can see the following output:

```
- 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
```

Additional resources

- See the **fdo-admin-tool generate-key-and-cert --help** manual page

11.4. INSTALLING AND RUNNING THE MANUFACTURING SERVER

The **fdo-manufacturing-server** RPM package enables you to run the Manufacturing Server component of the FDO protocol. It also stores other components, such as the owner vouchers, the manufacturer keys, and information about the manufacturing sessions. During the device installation, the

Manufacturing server generates the device credentials for the specific device, including its GUID, rendezvous information and other metadata. Later on in the process, the device uses this rendezvous information to contact the Rendezvous server.

IMPORTANT

Red Hat provides the **fdo-manufacturing-server** tool as a Technology Preview feature and should run on secure networks because Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

To install the **manufacturing server** RPM package, complete the following steps:

Procedure

1. Install the **fdo-admin-cli** package:

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

2. Check if the **fdo-manufacturing-server** RPM package is installed:

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

3. Check if the files were correctly installed:

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

You can see the following output:

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

4. Optional: Check the content of each file, for example:

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

5. Configure the Manufacturing server. You must provide the following information:

   - The Manufacturing server URL
   - The IP address or DNS name for the Rendezvous server
   - The path to the keys and certificates you generated. See Generating key and certificates. You can find an example of a Manufacturing server configuration file in the **/usr/share/doc/fdo/manufacturing-server.yml** directory. The following is a **manufacturing**

server.yml example that is created and saved in the **/etc/fdo** directory. It contains paths to the directories, certificates, keys that you created, the Rendezvous server IP address and the default port.

```
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. Start the Manufacturing server.

   a. Check if the systemd unit file are in the server:

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

   b. Enable and start the manufacturing server.

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

   c. Open the default ports in your firewall:

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

   d. Ensure that the service is listening on the port 8080:

```
# ss -ltn
```

7. Install RHEL for Edge onto your system using the simplified installer. See Building simplified installer images to provision a RHEL for Edge image.

Additional resources

- The manufacturing-server.yml example
- FDO automatic onboarding terminology

11.5. INSTALLING, CONFIGURING, AND RUNNING THE RENDEZVOUS SERVER

Install the **fdo-rendezvous-server** RPM package to enable the systems to receive the voucher generated by the Manufacturing server during the first device boot. The Rendezvous server then matches the device UUID with the target platform or cloud and informs the device about which Owner server endpoint the device must use.

Prerequisites

- You created a **manufacturer_cert.pem** certificate. See Generating key and certificates.
- You copied the **manufacturer_cert.pem** certificate to the **/etc/fdo/keys** directory in the Rendezvous server.

Procedure

1. Install the **fdo-rendezvous-server** RPM packages:

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

2. Create the **rendezvous-server.yml** configuration file, including the path to the manufacturer certificate. You can find an example in **/usr/share/doc/fdo/rendezvous-server.yml**. The following example shows a configuration file that is saved in **/etc/fdo/rendezvous-server.yml**.

```
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. Check the Rendezvous server service status:

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

   a. If the service is stopped and disabled, enable and start it:

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

4. Check that the server is listening on the default configured port 8082:

```
# ss -ltn
```

5. Open the port if you have a firewall configured on this server:

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

11.6. INSTALLING, CONFIGURING, AND RUNNING THE OWNER SERVER

Install the **fdo-owner-cli** and **fdo-owner-onboarding-server** RPM package to enable the systems to receive the voucher generated by the Manufacturing server during the first device boot. The Rendezvous server then matches the device UUID with the target platform or cloud and informs the device about which Owner server endpoint the device must use.

Prerequisites

- The device where the server will be deployed has a Trusted Platform Module (TPM) device to encrypt the disk. If not, you will get an error when booting the RHEL for Edge device.
- You created the **device_ca_cert.pem**, **owner_key.der**, and **owner_cert.pem** with keys and certificates and copied them into the **/etc/fdo/keys** directory.

Procedure

1. Install the required RPMs in this server:

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

2. Prepare the **owner-onboarding-server.yml** configuration file and save it to the **/etc/fdo/** directory. Include the path to the certificates you already copied and information about where to publish the Owner server service in this file.
The following is an example available in **/usr/share/doc/fdo/owner-onboarding-server.yml**. You can find references to the Service Info API, such as the URL or the authentication token.

```
---
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/5flBkaiNSvDYS3cEdBQXJn2Zv9n1D50431/lo=
owner_addresses:
  - transport: http
    addresses:
      - ip_address: 192.168.122.149
    port: 8081
report_to_rendezvous_endpoint_enabled: true
```

3. Create and configure the Service Info API.

   a. Add the automated information for onboarding, such as user creation, files to be copied or created, commands to be executed, disk to be encrypted, and so on. Use the Service Info API configuration file example in **/usr/share/doc/fdo/serviceinfo-api-server.yml** as a template to create the configuration file under **/etc/fdo/**.

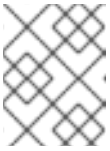
```
---
service_info:
  initial_user:
    username: admin
    sshkeys:
    - "ssh-rsa AAAA...."
  files:
  - path: /root/resolv.conf
    source_path: /etc/resolv.conf
  commands:
  - command: touch
    args:
    - /root/test
    return_stdout: true
    return_stderr: true
  diskencryption_clevis:
  - 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. Check the status of the systemd units:

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

   a. If the service is stopped and disabled, enable and start it:

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



NOTE

You must restart the **systemd** services every time you change the configuration files.

5. Check that the server is listening on the default configured port 8083:

```
# ss -ltn
```

6. Open the port if you have a firewall configured on this server:

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

11.7. AUTOMATICALLY ONBOARDING A RHEL FOR EDGE DEVICE BY USING FDO AUTHENTICATION

To prepare your device to automatically onboard a RHEL for Edge device and provision it as part of the installation process, complete the following steps:

Prerequisites

- You built an OSTree commit for RHEL for Edge and used that to generate an **edge-simplified-installer** artifact.
- Your device is assembled.
- You installed the **fdo-manufacturing-server** RPM package. See Installing the manufacturing server package.

Procedure

1. Start the installation process by booting the RHEL for Edge simplified installer image on your device. You can install it from a CD-ROM or from a USB flash drive, for example.
2. Verify through the terminal that the device has reached the manufacturing service to perform the initial device credential exchange and has produced an ownership voucher.
   You can find the ownership voucher at the storage location configured by the **ownership_voucher_store_driver:** parameter at the **manufacturing-sever.yml** file.

   The directory should have an **ownership_voucher** file with a name in the GUID format which indicates that the correct device credentials were added to the device.

   The onboarding server uses the device credential to authenticate against the onboarding server. It then passes the configuration to the device. After the device receives the configuration from the onboarding server, it receives an SSH key and installs the operating system on the device. Finally, the system automatically reboots, encrypts it with a strong key stored at TPM.

Verification

After the device automatically reboots, you can log in to the device with the credentials that you created as part of the FDO process.

- Log in to the device by providing the username and password you created in the Service Info API.

Additional resources

- Deploying the Simplified ISO image from a USB flash drive

CHAPTER 12. USING FDO TO ONBOARD RHEL FOR EDGE DEVICES WITH A DATABASE BACKEND

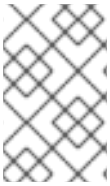
You can use the FDO servers - **manufacturer-server**, **onboarding-server**, and **rendezvous**- to support storing and querying Owner Vouchers from an SQL backend such as the SQLite or the PostgreSQL databases, instead of a file. This way, you can select an SQL datastore in the FDO server options, along with credentials and other parameters, to store the Owner Vouchers in the SQL database and thus onboard a RHEL for Edge device. The SQL files are packaged in the RPMs.

NOTE

The SQL backend does not support all FDO features at this point.

12.1. ONBOARDING DEVICES WITH AN FDO DATABASE

Use an SQL database to onboard your Edge device. The following example uses the **diesel** tool, but you can also use the SQLite or the PostgreSQL databases.



NOTE

You can use different database storages in some servers with filesystem storage in other servers, for example, using the filesystem storage onboarding for the Manufacturing server and Postgres for Rendezvous and Owner servers.

Prerequisites

- You used FDO to generate the key and certificates to configure the manufacturing server. See link to [Generating key and certificates]
- You installed and configured the manufacturing server. See Installing and running the manufacturing server
- You installed and configured the rendezvous server. See Installing, configuring, and running the Rendezvous server
- You installed and configured the owner server. See Installing, configuring, and running the Owner server
- You have your server configuration files in **/etc/fdo**
- You built an OSTree commit for RHEL for Edge and used that to generate an **edge-simplified-installer** artifact
- Your device is assembled
- You installed the **diesel** tool or a SQL database to your host.
- You have configured your database system and have permissions to create tables.

Procedure

1. Install the following packages:

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

2. Access the **/usr/share/doc/fdo/migrations/*** directory. It contains **.sql** files that you need to create the database for each of the server and type combinations, after you have installed the manufacturing, rendezvous, and owner server's RPMs.

3. Initialize the database content. You can use either an SQL database, such as SQLite or PostgreSQL, or the **diesel** tool to run the SQL for you.

   - If you are using an SQL database, configure the database server with, for example, user creation, access management, for example. After you configure the database server, you can run the database.

   - If you do not want to run the **.sql** files in your database system, you can use the **diesel** tool to run the sql for you. If you are using the **diesel** tool, after you have configured the database, use the **diesel migration run** command to create the database:

4. After you configure the DB system, you can use the .sql files installed at **/usr/share/doc/fdo/migrations/*** to create the database for each server type.
   You must use the **.sql** file that matches the server type and database type that you are initializing. For example, when initializing the Owner Onboarding Server in a PostgreSQL database, you must use the **/usr/share/doc/fdo/migrations/migrations_owner_onboarding_server_postgres/up.sql** folder. The **up.sql** file in the migration folder creates the database, and the **down.sql** file destroys it.

5. After you create the database, change the configuration file of the specific server to make it use the database.
   Each server has a storage configuration section.

   - Manufacturer server: **ownership_voucher_store_driver**

   - Owner server: **ownership_voucher_store_driver**

   - Rendezvous server: **storage_driver**

     a. For the **manufacturing-server.yml** file, open it in your editor and change the store database:

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

     b. Change the **ownership_voucher_store_driver** configuration under the Directory section:

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

     c. Specify the following details:

   - The database type that you are using: SQLite or PostgreSQL

   - The server type: for example, when using PostgreSQL, set the following configuration:

     ```
     ownership_voucher_store_driver:
       Postgres:
         Manufacturer
     ```

     a. Repeat the steps to configure the Owner server and Rendezvous server.

6. Run the FDO onboarding service. See Automatically provisioning and onboarding RHEL for Edge devices with FDO for more details. Start the FDO onboarding process device initialization by running the manufacturing server:

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

   The onboarding process happens in two phases:

   - The device initialization phase that usually happens in the manufacturing site.
   - The device onboarding process that happens at the final destination of the device. As a result, the Ownership Vouchers stored in the Manufacturing Server's Database must be exported and transferred to the final Owner Database.

7. To export the Ownership Vouchers, from the manufacturing-vouchers database file, copy the Owner Voucher to the owner to continue the FDO onboarding protocol.

   a. Create a folder **export**:

```
$ mkdir export
```

   b. Export the Owner Voucher present in the Manufacturing Database by providing all the variables required for the command.

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

   DB_TYPE

   The type of the Manufacturing DB holding the Owner Vouchers: sqlite, postgres

   DB_URL

   The database connection URL, or path to the database file

   PATH

   The path to the directory where the Owner Vouchers will be exported

   GUID

   GUID of the owner voucher to be exported. If you do not provide a GUID, all the Owner Vouchers will be exported.

8. The OVs must be delivered to the final Owner's database. For that, use the **fdo-owner-tool** to import the Owner vouchers. Change to owner database. Import the Ownership Vouchers by running the following command:

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

   DB_TYPE

   Type of the Owner DB to import the OVs. Possible values: sqlite, postgres

   DB_URL

   DB connection URL or path to DB file

   SOURCE_PATH

   Path to the OV to be imported, or path to a directory where all the OVs to be imported are located

The command reads each OV that is specified in the <SOURCE_PATH> once, one by one, and tries to import them into the database. If the command finds errors, it returns an output with the GUIDs of the OVs that are faulty and specified with information about what caused the error. The non-faulty OVs are imported into the database. The device receives the configuration from the onboarding server. Then, the device receives an SSH key and starts to install the operating system on the device. Finally, the operating system automatically reboots in the device, and encrypts the device with a strong key stored at TPM.

CHAPTER 13. DEPLOYING A RHEL FOR EDGE IMAGE IN A NETWORK-BASED ENVIRONMENT

You can deploy a RHEL for Edge image using the RHEL installer graphical user interface or a Kickstart file. The overall process for deploying a RHEL for Edge image depends on whether your deployment environment is network-based or non-network-based.

NOTE

To deploy the images on bare metal, use a Kickstart file.

Network-based deployments

Deploying a RHEL for Edge image in a network-based environment involves the following high-level steps:

a. Extract the image file contents.

b. Set up a web server

c. Install the image

13.1. EXTRACTING THE RHEL FOR EDGE IMAGE COMMIT

After you download the commit, extract the **.tar** file and note the ref name and the commit ID.

The downloaded commit file consists of a **.tar** file with an **OSTree** repository. The **OSTree** repository has a commit and a **compose.json** file.

The **compose.json** file has information metadata about the commit with information such as the "Ref", the reference ID and the commit ID. The commit ID has the RPM packages.

To extract the package contents, perform the following the steps:

Prerequisites

- Create a Kickstart file or use an existing one.

Procedure

1. Extract the downloaded image **.tar** file:

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

2. Go to the directory where you have extracted the **.tar** file.
   It has a **compose.json** file and an OSTree directory. The **compose.json** file has the commit number and the **OSTree** directory has the RPM packages.

3. Open the **compose.json** file and note the commit ID number. You need this number handy when you proceed to set up a web server.
   If you have the **jq** JSON processor installed, you can also retrieve the commit ID by using the **jq** tool:

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

4. List the RPM packages in the commit.

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

5. Use a Kickstart file to run the RHEL installer. Optionally, you can use any existing file or can create one by using the Kickstart Generator tool.
   In the Kickstart file, ensure that you include the details about how to provision the file system, create a user, and how to fetch and deploy the RHEL for Edge image. The RHEL installer uses this information during the installation process.

   The following is a Kickstart file example:

```
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
```

   The OStree-based installation uses the **ostreesetup** command to set up the configuration. It fetches the OSTree commit, by using the following flags:

   - **--nogpg** - Disable GNU Privacy Guard (GPG) key verification.
   - **--osname** - Management root for the operating system installation.
   - **--remote** - Management root for the operating system installation
   - **--url** - URL of the repository to install from.
   - **--ref** - Name of the branch from the repository that the installation uses.
   - **--url=http://mirror.example.com/repo/** - is the address of the host system where you extracted the edge commit and served it over **nginx**. You can use the address to reach the host system from the guest computer.
     For example, if you extract the commit image in the **/var/www/html** directory and serve the commit over **nginx** on a computer whose hostname is **www.example.com**, the value of the **--url** parameter is **http://www.example.com/repo**.

NOTE

Use the http protocol to start a service to serve the commit, because https is not enabled on the Apache HTTP Server.

Additional resources

- Downloading a RHEL for Edge image
- Creating Kickstart files

13.2. SETTING UP A WEB SERVER TO INSTALL RHEL FOR EDGE IMAGES

After you have extracted the RHEL for Edge image contents, set up a web server to provide the image commit details to the RHEL installer by using HTTP.

The following example provides the steps to set up a web server by using a container.

Prerequisites

- You have installed Podman on your system. See the Red Hat Knowledgebase solution How do I install Podman in RHEL.

Procedure

1. Create the **nginx** configuration file with the following instructions:

```
events {

}

http {
    server{
        listen 8080;
        root /usr/share/nginx/html;
                }
         }

pid /run/nginx.pid;
daemon off;
```

2. Create a Dockerfile with the following instructions:

```
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/
```

Where,

- ***kickstart.ks*** is the name of the Kickstart file from the RHEL for Edge image. The Kickstart file includes directive information. To help you manage the images later, it is advisable to include the checks and settings for greenboot checks. For that, you can update the

Kickstart file to include the following settings:

```
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
```

Any HTTP service can host the OSTree repository, and the example, which uses a container, is just an option for how to do this. The Dockerfile performs the following tasks:

a. Uses the latest Universal Base Image (UBI)

b. Installs the web server (nginx)

c. Adds the Kickstart file to the server

d. Adds the RHEL for Edge image commit to the server

3. Build a Docker container

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

4. Run the container

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

As a result, the server is set up and ready to start the RHEL Installer by using the **commit.tar** repository and the Kickstart file.

13.3. PERFORMING AN ATTENDED INSTALLATION TO AN EDGE DEVICE BY USING KICKSTART

For an attended installation in a network-based environment, you can install the RHEL for Edge image to a device by using the RHEL Installer ISO, a Kickstart file, and a web server. The web server serves the RHEL for Edge Commit and the Kickstart file to boot the RHEL Installer ISO image.

Prerequisites

- You have made the RHEL for Edge Commit available by running a web server. See Setting up a web server to install RHEL for Edge images.
- You have created a **.qcow2** disk image to be used as the target of the attended installation. See Creating a virtual disk image by using qemu-img .

Procedure

1. Create a Kickstart file. The following is an example in which the **ostreesetup** directive instructs the Anaconda Installer to fetch and deploy the commit. Additionally, it creates a user and a password.

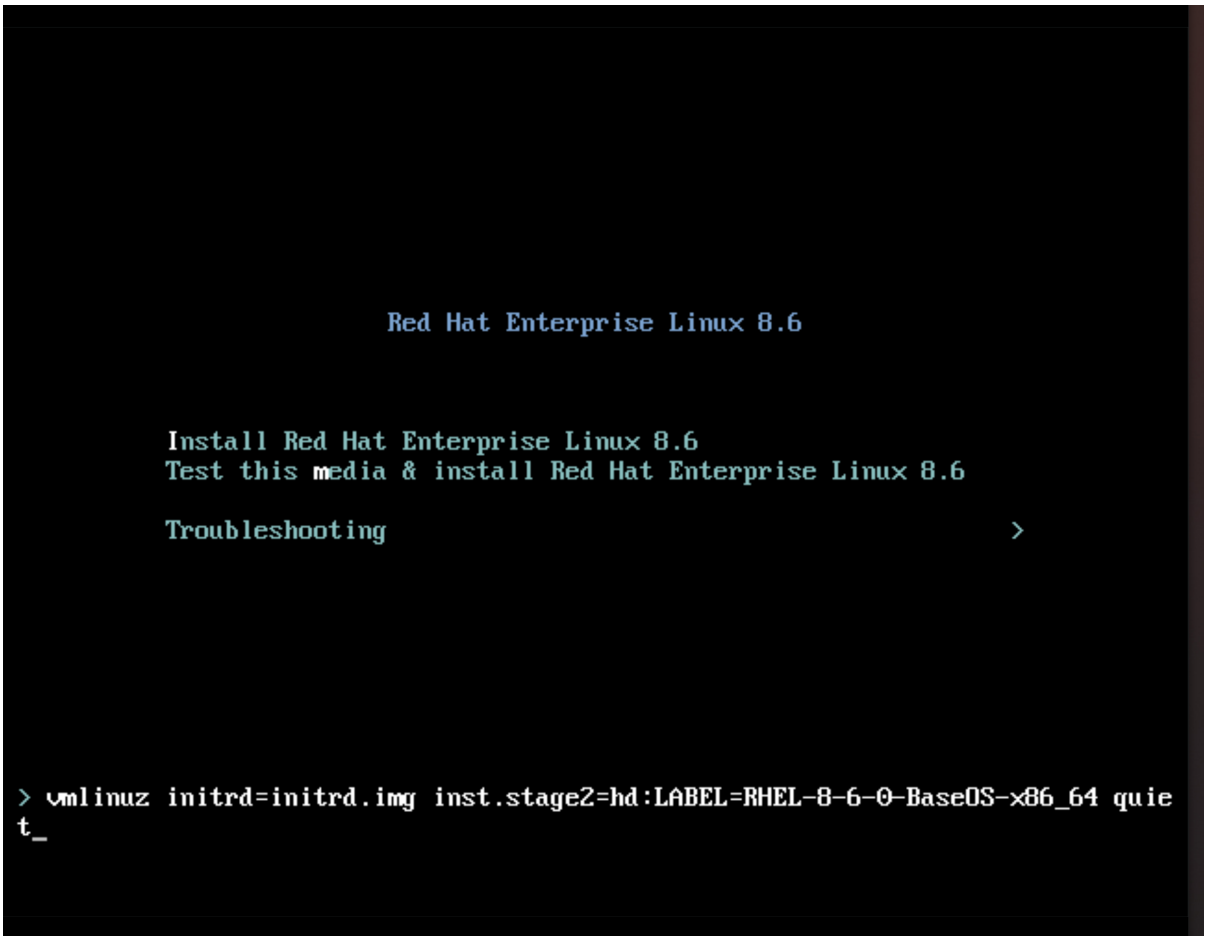
```
lang en_US.UTF-8
keyboard us
timezone UTC
zerombr
clearpart --all --initlabel
autopart --type=plain --fstype=xfs --nohome
reboot
text
network --bootproto=dhcp
user --name=core --groups=wheel --password=edge
services --enabled=ostree-remount
ostreesetup --nogpg --url=http://edge_device_ip:port/repo/ --osname=rhel --remote=edge --ref=rhel/9/x86_64/edge
```

2. Run the RHEL Anaconda Installer by using the **libvirt virt-install** utility to create a virtual machine (VM) with a RHEL operating system. Use the **.qcow2** disk image as the target disk in the attended installation:

```
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. On the installation screen:

Figure 13.1. Red Hat Enterprise Linux boot menu



a. Press the **e** key to add an additional kernel parameter:

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

The kernel parameter specifies that you want to install RHEL by using the Kickstart file and not the RHEL image contained in the RHEL Installer.

b. After adding the kernel parameters, press **Ctrl**+**X** to boot the RHEL installation by using the Kickstart file.
The RHEL Installer starts, fetches the Kickstart file from the server (HTTP) endpoint and executes the commands, including the command to install the RHEL for Edge image commit from the HTTP endpoint. After the installation completes, the RHEL Installer prompts you for login details.

Verification

1. On the Login screen, enter your user account credentials and click **Enter**.
2. Verify whether the RHEL for Edge image is successfully installed.

```
$ rpm-ostree status
```

The command output provides the image commit ID and shows that the installation is successful.

Following is a sample output:

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

Additional resources

- How to embed a Kickstart file into an ISO image (Red Hat Knowledgebase)
- Booting the installation

13.4. PERFORMING AN UNATTENDED INSTALLATION TO AN EDGE DEVICE BY USING KICKSTART

For an unattended installation in a network-based environment, you can install the RHEL for Edge image to an Edge device by using a Kickstart file and a web server. The web server serves the RHEL for Edge Commit and the Kickstart file, and both artifacts are used to start the RHEL Installer ISO image.

Prerequisites

- You have the **qemu-img** utility installed on your host system.
- You have created a **.qcow2** disk image to install the commit you created. See Creating a system image with RHEL image builder in the CLI.
- You have a running web server. See Creating a RHEL for Edge Container image for non-network-based deployments.

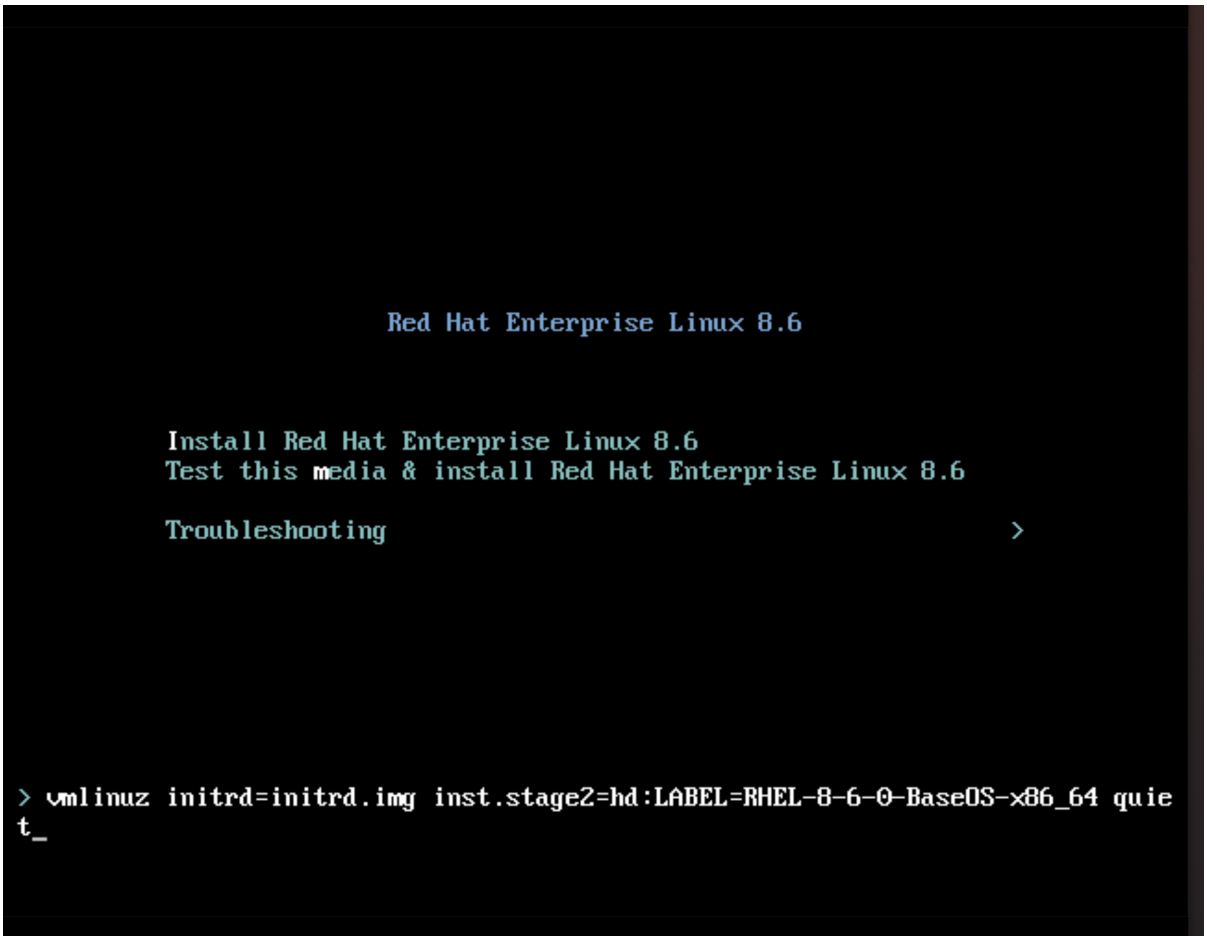
Procedure

1. Run a RHEL for Edge Container image to start a web server. The server fetches the commit in the RHEL for Edge Container image and becomes available and running.
2. Run the RHEL Anaconda Installer, passing the customized **.qcow2** disk image, by using **libvirt virt-install**:

```
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. On the installation screen:

Figure 13.2. Red Hat Enterprise Linux boot menu



a. Press the **TAB** key and add the Kickstart kernel argument:

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

The kernel parameter specifies that you want to install RHEL by using the Kickstart file and not the RHEL image contained in the RHEL Installer.

b. After adding the kernel parameters, press **Ctrl**+**X** to boot the RHEL installation by using the Kickstart file.
The RHEL Installer starts, fetches the Kickstart file from the server (HTTP) endpoint, and executes the commands, including the command to install the RHEL for Edge image commit from the HTTP endpoint. After the installation completes, the RHEL Installer prompts you for login details.

Verification

1. On the Login screen, enter your user account credentials and click **Enter**.
2. Verify whether the RHEL for Edge image is successfully installed.

```
$ rpm-ostree status
```

The command output provides the image commit ID and shows that the installation is successful.

The following is a sample output:

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

Additional resources

- How to embed a Kickstart file into an ISO image (Red Hat Knowledgebase)
- Booting the installation

CHAPTER 14. DEPLOYING A RHEL FOR EDGE IMAGE IN A NON-NETWORK-BASED ENVIRONMENT

The RHEL for Edge Container (**.tar**) in combination with the RHEL for Edge Installer (**.iso**) image type result in a ISO image. The ISO image can be used in disconnected environments during the image deployment to a device. However, network access might require network access to build the different artifacts.

Deploying a RHEL for Edge image in a non-network-based environment involves the following high-level steps:

1. Download the RHEL for Edge Container. See Downloading a RHEL for Edge image for information about how to download the RHEL for Edge image.
2. Load the RHEL for Edge Container image into Podman
3. Run the RHEL for Edge Container image into Podman
4. Load the RHEL for Edge Installer blueprint
5. Build the RHEL for Edge Installer image
6. Prepare a **.qcow2** disk
7. Boot the Virtual Machine (VM)
8. Install the image

14.1. CREATING A RHEL FOR EDGE CONTAINER IMAGE FOR NON-NETWORK-BASED DEPLOYMENTS

You can build a running container by loading the downloaded RHEL for Edge Container OSTree commit into Podman. For that, follow the steps:

Prerequisites

- You created and downloaded a RHEL for Edge Container OSTree commit.
- You have installed **Podman** on your system. See the Red Hat Knowledgebase solution How do I install Podman in RHEL.

Procedure

1. Navigate to the directory where you have downloaded the RHEL for Edge Container OSTree commit.
2. Load the RHEL for Edge Container OSTree commit into **Podman**.

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

The command output provides the image ID, for example:
@8e0d51f061ff1a51d157804362bc875b649b27f2ae1e66566a15e7e6530cec63

3. Tag the new RHEL for Edge Container image, using the image ID generated by the previous step.

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

The **podman tag** command assigns an additional name to the local image.

4. Run the container named **edge-container**.

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

The **podman run -d --name=edge-container** command assigns a name to your container-based on the **localhost/edge-container** image.

5. List containers:

```
$ 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
```

As a result, **Podman** runs a container that serves an OSTree repository with the RHEL for Edge Container commit.

14.2. CREATING A RHEL FOR EDGE INSTALLER IMAGE FOR NON-NETWORK-BASED DEPLOYMENTS

After you have built a running container to serve a repository with the **RHEL for Edge Container** commit, create an **RHEL for Edge Installer (.iso)** image. The **RHEL for Edge Installer (.iso)** pulls the commit served by **RHEL for Edge Container (.tar)**. After the **RHEL for Edge Container** commit is loaded in Podman, it exposes the **OSTree** in the URL format.

To create the RHEL for Edge Installer image in the CLI, follow the steps:

Prerequisites

- You created a blueprint for RHEL for Edge image.
- You created a RHEL for Edge Container image and deployed it using a web server.

Procedure

1. Begin to create the RHEL for Edge Installer image.

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

Where,

- *ref* is the same value that customer used to build OSTree repository

- *URL-OSTree-repository* is the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. See Creating a RHEL for Edge Container image for non-network-based deployments.
- *blueprint-name* is the RHEL for Edge Installer blueprint name.
- *image-type* is **edge-installer**.
  A confirmation that the composer process has been added to the queue appears. It also shows a Universally Unique Identifier (UUID) number for the image created. Use the UUID number to track your build. Also keep the UUID number handy for further tasks.

2. Check the image compose status.

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

The command output displays the status in the following format:

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

NOTE

The image creation process takes a few minutes to complete.

To interrupt the image creation process, run:

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

To delete an existing image, run:

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

RHEL image builder pulls the commit that is being served by the running container during the image build.

After the image build is complete, you can download the resulting ISO image.

3. Download the image. See Downloading a RHEL for Edge image.
   After the image is ready, you can use it for **non-network deployments**. See Installing the RHEL for Edge image for non-network-based deployments.

Additional resources

- Creating a RHEL for Edge Installer image by using the command-line interface for non-network-based deployments

14.3. INSTALLING THE RHEL FOR EDGE IMAGE FOR NON-NETWORK-BASED DEPLOYMENTS

To install the RHEL for Edge image, follow the steps:

Prerequisites

- You created a RHEL for Edge Installer ISO image.

- You stopped the running container.
- A disk image to install the commit you created.
- You installed the **edk2-ovmf** package.
- You installed the **virt-viewer** package.
- You customized your blueprint with a user account. See Creating a blueprint for a RHEL for Edge image using RHEL image builder in RHEL web console.

WARNING

If you do not define a user account customization in your blueprint, you will not be able to login to the ISO image.

Procedure

1. Create a **qcow** VM disk file to install the (**.iso**) image. That is an image of a hard disk drive for the virtual machine (VM). For example:

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

2. Use the **virt-install** command to boot the VM using the disk as a drive and the installer ISO image as a CD-ROM. For example:

```
$ 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
```

This command instructs **virt-install** to:

- Instructs the VM to use UEFI to boot, instead of the BIOS.
- Mount the installation ISO.
- Use the hard disk drive image created in the first step.
  It gives you an Anaconda Installer. The RHEL Installer starts, loads the Kickstart file from the ISO and executes the commands, including the command to install the RHEL for Edge image commit. Once the installation is complete, the RHEL installer prompts for login details.

NOTE

Anaconda is preconfigured to use the Container commit during the installation. However, you need to set up system configurations, such as disk partition, timezone, between others.

3. Connect to Anaconda GUI with **virt-viewer** to setup the system configuration:

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

4. Reboot the system to finish the installation.
5. On the login screen, specify your user account credentials and click **Enter**.

Verification

1. Verify whether the RHEL for Edge image is successfully installed.

```
$  rpm-ostree status
```

The command output provides the image commit ID and shows that the installation is successful.

CHAPTER 15. MANAGING RHEL FOR EDGE IMAGES

To manage the RHEL for Edge images, you can perform any of the following administrative tasks:

- Edit the RHEL for Edge image blueprint by using image builder in RHEL web console or in the command-line
- Build a commit update by using image builder command-line
- Update the RHEL for Edge images
- Configure **rpm-ostree** remotes on nodes, to update node policy
- Restore RHEL for Edge images manually or automatically by using greenboot

15.1. EDITING A RHEL FOR EDGE IMAGE BLUEPRINT BY USING IMAGE BUILDER

You can edit the RHEL for Edge image blueprint to:

- Add additional components that you might require
- Modify the version of any existing component
- Remove any existing component

15.1.1. Adding a component to RHEL for Edge blueprint using image builder in RHEL web console

To add a component to a RHEL for Edge image blueprint, ensure that you have met the following prerequisites and then follow the procedure to edit the corresponding blueprint.

Prerequisites

- On a RHEL system, you have accessed the RHEL image builder dashboard.
- You have created a blueprint for RHEL for Edge image.

Procedure

1. On the RHEL image builder dashboard, click the blueprint that you want to edit.
   To search for a specific blueprint, enter the blueprint name in the filter text box, and click **Enter**.
2. On the upper right side of the blueprint, click **Edit Packages**.
   The **Edit blueprints** wizard opens.
3. On the **Details** page, update the blueprint name and click **Next**.
4. On the **Packages** page, follow the steps:
   a. In the **Available Packages**, enter the package name that you want to add in the filter text box, and click **Enter**.
      A list with the component name appears.
   b. Click **>** to add the component to the blueprint.

5. On the **Review** page, click **Save**.
   The blueprint is now updated with the new package.

15.1.2. Removing a component from a blueprint using RHEL image builder in the web console

To remove one or more unwanted components from a blueprint that you created by using RHEL image builder, ensure that you have met the following prerequisites and then follow the procedure.

Prerequisites

- On a RHEL system, you have accessed the RHEL image builder dashboard.
- You have created a blueprint for RHEL for Edge image.
- You have added at least one component to the RHEL for Edge blueprint.

Procedure

1. On the RHEL image builder dashboard, click the blueprint that you want to edit.
   To search for a specific blueprint, enter the blueprint name in the filter text box, and click **Enter**.
2. On the upper right side of the blueprint, click **Edit Packages**.
   The **Edit blueprints** wizard opens.
3. On the **Details** page, update the blueprint name and click **Next**.
4. On the **Packages** page, follow the steps:
   a. From the **Chosen packages**, click **<** to remove the chosen component. You can also click **<<** to remove all the packages at once.
5. On the **Review** page, click **Save**.
   The blueprint is now updated.

15.1.3. Editing a RHEL for Edge image blueprint by using the command line

You can change the specifications for your RHEL for Edge image blueprint by using the RHEL image builder command-line interface. To do so, ensure that you have met the following prerequisites and then follow the procedure to edit the corresponding blueprint.

Prerequisites

- You have access to the RHEL image builder command-line.
- You have created a RHEL for Edge image blueprint.

Procedure

1. Save (export) the blueprint to a local text file:

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

2. Edit the **BLUEPRINT-NAME.toml** file with a text editor of your choice and make your changes.

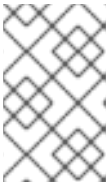
Before finishing with the edits, verify that the file is a valid blueprint:

3. Increase the version number.
Ensure that you use a Semantic Versioning scheme.

NOTE

if you do not change the version, the patch component of the version is increased automatically.

4. Check if the contents are valid TOML specifications. See the TOML documentation for more information.



NOTE

TOML documentation is a community product and is not supported by Red Hat. You can report any issues with the tool at https://github.com/toml-lang/toml/issues.

5. Save the file and close the editor.

6. Push (import) the blueprint back into RHEL image builder server:

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

NOTE

When pushing the blueprint back into the RHEL image builder server, provide the file name including the **.toml** extension.

7. Verify that the contents uploaded to RHEL image builder match your edits:

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

8. Check whether the components and versions listed in the blueprint and their dependencies are valid:

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

15.2. UPDATING RHEL FOR EDGE IMAGES

15.2.1. How RHEL for Edge image updates are deployed

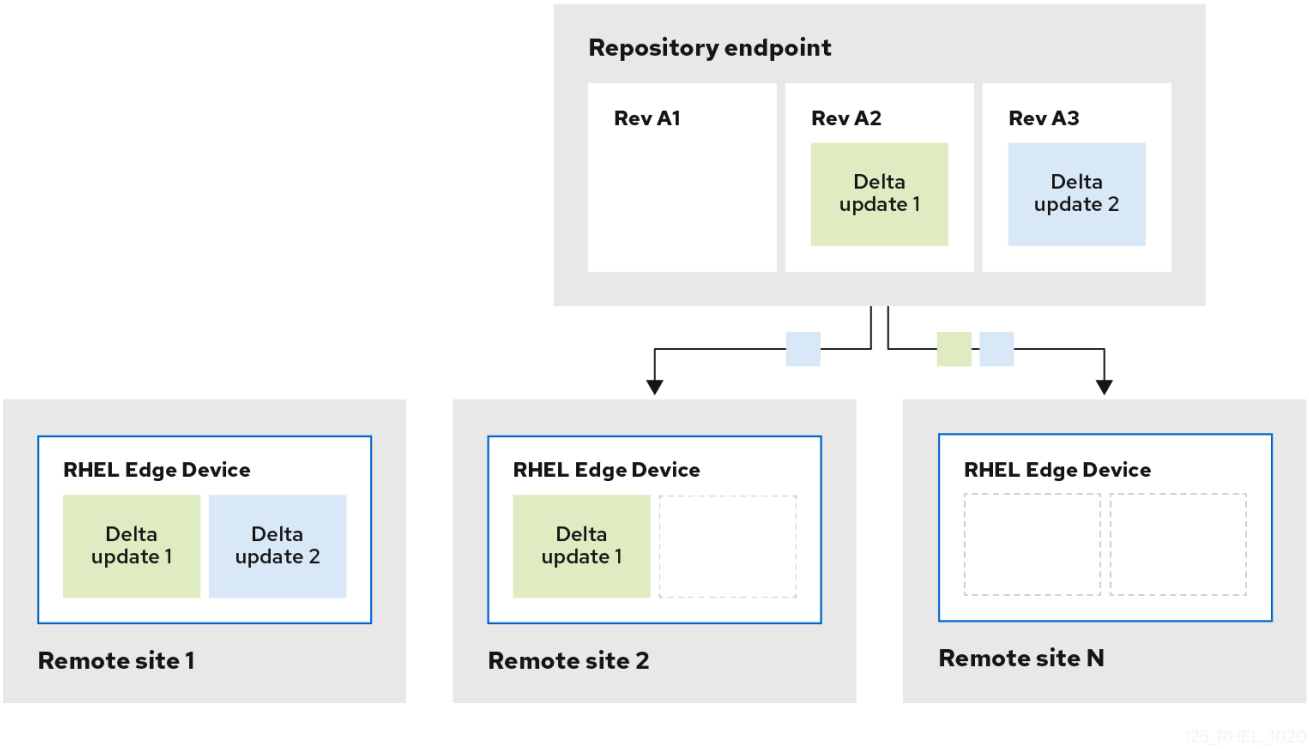
With RHEL for Edge images, you can either deploy the updates manually or can automate the deployment process. The updates are applied in an atomic manner, where the state of each update is known, and the updates are staged and applied only upon reboot. Because no changes are seen until you reboot the device, you can schedule a reboot to ensure the highest possible uptime.

During the image update, only the updated operating system content is transferred over the network. This makes the deployment process more efficient compared to transferring the entire image. The operating system binaries and libraries in **/usr** are **read-only**, and the **read and write** state is maintained

in **/var** and **/etc** directories.

When moving to a new deployment, the **/etc** and the **/var** directories are copied to the new deployment with **read and write** permissions. The **/usr** directory is copied as a soft link to the new deployment directory, with **read-only** permissions.

The following diagram illustrates the RHEL for Edge image update deployment process:



By default, the new system is booted using a procedure similar to a **chroot** operation, that is, the system enables control access to a filesystem while controlling the exposure to the underlying server environment. The new **/sysroot** directory mainly has the following parts:

- Repository database at the **/sysroot/ostree/repo** directory.
- File system revisions at the **/sysroot/ostree/deploy/rhel/deploy** directory, which are created by each operation in the system update.
- The **/sysroot/ostree/boot** directory, which links to deployments on the previous point. Note that **/ostree** is a soft link to **/sysroot/ostree**. The files from the **/sysroot/ostree/boot** directory are not duplicated. The same file is used if it is not changed during the deployment. The files are hard-links to another file stored in the **/sysroot/ostree/repo/objects** directory.

The operating system selects the deployment in the following way:

1. The **dracut** tool parses the **ostree** kernel argument in the **initramfs root** file system and sets up the **/usr** directory as a **read-only** bind mount.
2. Bind the deployment directory in **/sysroot** to / directory.
3. Re-mount the operating system already mounted **dirs** using the **MS_MOVE** mount flag

If anything goes wrong, you can perform a deployment rollback by removing the old deployments with the **rpm-ostree** cleanup command. Each client machine contains an **OSTree** repository stored in **/ostree/repo**, and a set of deployments stored in **/ostree/deploy/$STATEROOT/$CHECKSUM**.

With the deployment updates in RHEL for Edge image, you can benefit from a better system consistency across multiple devices, easier reproducibility, and better isolation between the pre and post system states change.

15.2.2. Building a commit update

You can build a commit update after making a change in the blueprint, such as:

- Adding an additional package that your system requires
- Modifying the package version of any existing component
- Removing any existing package.

Prerequisites

- You have updated a system which is running RHEL image builder.
- You created a blueprint update.
- You have previously created an OSTree repository and served it through HTTP. See Setting up a web server to install RHEL for Edge images.

Procedure

1. Start the compose of the new commit image, with the following arguments: **--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
   ```

   The command instructs the compose process to fetch the metadata from the OStree repo before starting the compose. The resulting new OSTree commit contains a reference of the original OSTree commit as a parent image.

2. After the compose process finishes, fetch the **.tar** file.

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

3. Extract the commit to a temporary directory, so that you can store the commit history in the OSTree repo.

   ```
   $ tar -xf UUID.tar -C /var/tmp
   ```

4. Inspect the resulting OSTree repo commit, by using the **tar -xf** command. It extracts the tar file to disk so you can inspect the resulting OSTree repo:

   ```
   $ ostree --repo=/var/tmp/repo log rhel/9/x86_64/edge
   commit d523ef801e8b1df69ddbf73ce810521b5c44e9127a379a4e3bba5889829546fa
   Parent:  f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
   ContentChecksum:
   ```

```
f0f6703696331b661fa22d97358db48ba5f8b62711d9db83a00a79b3ae0dfe16
Date:  2023-06-04 20:22:28 /+0000
Version: 9
```

In the output example, there is a single OSTree commit in the repo that references a parent commit. The parent commit is the same checksum from the original OSTree commit that you previously made.

5. Merge the two commits by using the **ostree pull-local** command:

```
$ sudo ostree --repo=/var/srv/httpd/repo pull-local /var/tmp/repo
20 metadata, 22 content objects imported; 0 bytes content written
```

This command copies any new metadata and content from the location on the disk, for example, **/var/tmp**, to a destination OSTree repo in **/var/srv/httpd**.

Verification

1. Inspect the target OSTree repo:

```
$ 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)
```

You can see that the target OSTree repo now contains two commits in the repository, in a logical order. After successful verification, you can update your RHEL for Edge system.

15.2.3. Deploying RHEL for Edge image updates manually

After you have edited a RHEL for Edge blueprint, you can update the image commit. RHEL image builder generates a new commit for the updated RHEL for Edge image. Use this new commit to deploy the image with latest package versions or with additional packages.

To deploy RHEL for Edge images updates, ensure that you meet the prerequisites and then follow the procedure.

Prerequisites

- On a RHEL system, you have accessed the RHEL image builder dashboard.
- You have created a RHEL for Edge image blueprint.
- You have edited the RHEL for Edge image blueprint.

Procedure

1. On the RHEL image builder dashboard click **Create Image**.
2. On the **Create Image** window, perform the following steps:
   a. In the Image output page:
      i. From the **Select a blueprint** dropdown list, select the blueprint that you edited.
      ii. From the **Image output type** dropdown list, select **RHEL for Edge Commit (.tar)**. Click **Next**.
   b. In the **OSTree settings** page, enter:
      i. In the **Repository URL** field, enter the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. See Setting up a web server to install RHEL for Edge image.
      ii. In the **Parent commit** field, specify the parent commit ID that was previously generated. See Extracting RHEL for Edge image commit .
      iii. In the **Ref** field, you can either specify a name for your commit or leave it empty. By default, the web console specifies the **Ref** as **rhel/9/arch_name/edge**. Click **Next**.
   c. In the **Review** page, check the customizations and click **Create image**. RHEL image builder starts to create a RHEL for Edge image for the updated blueprint. The image creation process takes a few minutes to complete.
      To view the RHEL for Edge image creation progress, click the blueprint name from the breadcrumbs, and then click the **Images** tab.

      The resulting image includes the latest packages that you have added, if any, and have the original **commit ID** as a parent.
3. Download the resulting RHEL for Edge Commit (**.tar**) image.
   a. From the **Images** tab, click **Download** to save the RHEL for Edge Commit (**.tar**) image to your system.
4. Extract the OSTree commit (**.tar**) file.

```
# tar -xf UUID-commit.tar -C UPGRADE_FOLDER
```

5. Upgrade the OSTree repo:

```
# ostree --repo=/usr/share/nginx/html/repo pull-local UPGRADE_FOLDER
# ostree --repo=/usr/share/nginx/html/repo summary -u
```

6. On the RHEL system provisioned, from the original edge image, verify the current status.

```
$ rpm-ostree status
```

   If there is no new commit ID, run the following command to verify if there is any upgrade available:

```
$ rpm-ostree upgrade --check
```

The command output provides the current active OSTree commit ID.

7. Update OSTree to make the new OSTree commit ID available.

```
$ rpm-ostree upgrade
```

OSTree verifies if there is an update on the repository. If yes, it fetches the update and requests you to reboot your system so that you can activate the deployment of this new commit update.

8. Check the current status again:

```
$ rpm-ostree status
```

You can now see that there are 2 commits available:

- The active parent commit.
- A new commit that is not active and contains 1 added difference.

9. To activate the new deployment and to make the new commit active, reboot your system.

```
# systemctl reboot
```

The Anaconda Installer reboots into the new deployment. On the login screen, you can see a new deployment available for you to boot.

10. If you want to boot into the newest deployment (commit), the **rpm-ostree** upgrade command automatically orders the boot entries so that the new deployment is first in the list. Optionally, you can use the arrow key on your keyboard to select the GRUB menu entry and press **Enter**.

11. Provide your login user account credentials.

12. Verify the OSTree status:

```
$ rpm-ostree status
```

The command output provides the active commit ID.

13. To view the changed packages, if any, run a diff between the parent commit and the new commit:

```
$ rpm-ostree db diff parent_commit new_commit
```

The update shows that the package you have installed is available and ready for use.

15.2.4. Deploying RHEL for Edge image updates manually using the command-line

After you have edited a RHEL for Edge blueprint, you can update the image commit. RHEL image builder generates a new commit for the updated RHEL for Edge image. Use the new commit to deploy the image with latest package versions or with additional packages using the CLI.

To deploy RHEL for Edge image updates using the CLI, ensure that you meet the prerequisites, and then follow the procedure.

Prerequisites

- You created the RHEL for Edge image blueprint.
- You edited the RHEL for Edge image blueprint. See Editing a RHEL for Edge image blueprint by using the command line.

Procedure

1. Create the RHEL for Edge Commit (**.tar**) image with the following arguments:

```
# composer-cli compose start-ostree --ref ostree_ref --url URL-OSTree-repository -
blueprint_name_ image-type
```

   where

   - ***ref*** is the reference you provided during the creation of the RHEL for Edge Container commit. For example, **rhel/9/x86_64/edge**.
   - ***URL-OSTree-repository*** is the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. See Setting up a web server to install RHEL for Edge image.
   - ***image-type*** is **edge-commit**.
     RHEL image builder creates a RHEL for Edge image for the updated blueprint.

2. Check the RHEL for Edge image creation progress:

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

   NOTE

   The image creation processes can take up to ten to thirty minutes to complete.

   The resulting image includes the latest packages that you have added, if any, and has the original **commit ID** as a parent.

3. Download the resulting RHEL for Edge image. For more information, see Downloading a RHEL for Edge image using the RHEL image builder command-line interface.

4. Extract the OSTree commit.

```
# tar -xf UUID-commit.tar -C upgrade_folder
```

5. Serve the OSTree commit by using httpd. See Setting up a web server to install RHEL for Edge image.

6. Upgrade the OSTree repo:

```
# ostree --repo=/var/www/html/repo pull-local /tmp/ostree-commit/repo
# ostree --repo=/var/www/html/repo summary -u
```

7. On the RHEL system provisioned from the original edge image, verify the current status:

```
$ rpm-ostree status
```

If there is no new commit ID, run the following command to verify if there is any upgrade available:

```
$ rpm-ostree upgrade --check
```

The command output provides the current active OSTree commit ID.

8. Update OSTree to make the new OSTree commit ID available:

```
$ rpm-ostree upgrade
```

OSTree verifies if there is an update on the repository. If yes, it fetches the update and requests you to reboot your system so that you can activate the deployment of the new commit update.

9. Check the current status again:

```
$ rpm-ostree status
```

You should now see that there are 2 commits available:

- The active parent commit
- A new commit that is not active and contains 1 added difference

10. To activate the new deployment and make the new commit active, reboot your system:

```
# systemctl reboot
```

The Anaconda Installer reboots into the new deployment. On the login screen, you can see a new deployment available for you to boot.

11. If you want to boot into the newest deployment, the **rpm-ostree upgrade** command automatically orders the boot entries so that the new deployment is first in the list. Optionally, you can use the arrow key on your keyboard to select the GRUB menu entry and press **Enter**.

12. Log in using your account credentials.

13. Verify the OSTree status:

```
$ rpm-ostree status
```

The command output provides the active commit ID.

14. To view the changed packages, if any, run a diff between the parent commit and the new commit:

```
$ rpm-ostree db diff parent_commit new_commit
```

The update shows that the package you have installed is available and ready for use.

15.2.5. Deploying RHEL for Edge image updates manually for non-network-base deployments

After editing a RHEL for Edge blueprint, you can update your RHEL for Edge Commit image with those

updates. Use RHEL image builder to generate a new commit to update your RHEL for Edge image that is already deployed in a VM, for example. Use this new commit to deploy the image with latest package versions or with additional packages.

To deploy RHEL for Edge images updates, ensure that you meet the prerequisites and then follow the procedure.

Prerequisites

- On your host, you have opened the RHEL image builder app from the web console in a browser.
- You have a RHEL for Edge system provisioned that is up and running.
- You have an OSTree repository that is being served over HTTP.
- You have edited a previously created RHEL for Edge image blueprint.

Procedure

1. On your system host, on the RHEL image builder dashboard, click **Create Image**.
2. On the **Create Image** window, perform the following steps:
   a. In the **Image output** page:
      i. From the **Select a blueprint** dropdown list, select the blueprint that you edited.
      ii. From the **Image output type** dropdown list, select **RHEL for Edge Container (.tar)**.
      iii. Click **Next**.
   b. In the **OSTree settings** page, enter:
      i. In the **Repository URL** field, enter the URL to the OSTree repository of the commit to embed in the image. For example, http://10.0.2.2:8080/repo/. See Setting up a web server to install RHEL for Edge image.
      ii. In the **Parent commit** field, specify the parent commit ID that was previously generated. See Extracting RHEL for Edge image commit .
      iii. In the **Ref** field, you can either specify a name for your commit or leave it empty. By default, the web console specifies the **Ref** as **rhel/9/arch_name/edge**.
      iv. Click **Next**.
   c. In the **Review** page, check the customizations and click **Create**.
   RHEL image builder creates a RHEL for Edge image for the updated blueprint.
   d. Click the **Images** tab to view the progress of RHEL for Edge image creation.

NOTE

The image creation process takes a few minutes to complete.

The resulting image includes the latest packages that you have added, if any, and has the original **commit ID** as a parent.

3. Download the resulting RHEL for Edge image on your host:

   a. From the **Images** tab, click **Download** to save the RHEL for Edge Container (**.tar**) image to your host system.

4. On the RHEL system provisioned from the original edge image, perform the following steps:

   a. Load the RHEL for Edge Container image into Podman, serving the child commit ID this time.

   ```
   $ cat ./child-commit_ID-container.tar | sudo podman load
   ```

   b. Run **Podman**.

   ```
   #  sudo podman run -p 8080:8080 localhost/edge-test
   ```

   c. Upgrade the OSTree repo:

   ```
   # ostree --repo=/var/www/html/repo pull-local /tmp/ostree-commit/repo
   # ostree --repo=/var/www/html/repo summary -u
   ```

   d. On the RHEL system provisioned, from the original edge image, verify the current status.

   ```
   $ rpm-ostree status
   ```

   If there is no new commit ID, run the following command to verify if there is any upgrade available:

   ```
   $ rpm-ostree upgrade --check
   ```

   If there are updates available, the command output provides information about the available updates in the OSTree repository, such as the current active OSTree commit ID. Else, it prompts a message informing that there are no updates available.

   e. Update OSTree to make the new OSTree commit ID available.

   ```
   $ rpm-ostree upgrade
   ```

   OSTree verifies if there is an update on the repository. If yes, it fetches the update and requests you to reboot your system so that you can activate the deployment of this new commit update.

   f. Check the current system status:

   ```
   $ rpm-ostree status
   ```

   You can now see that there are 2 commits available:

   - The active parent commit.
   - A new commit that is not active and contains 1 added difference.

   g. To activate the new deployment and to make the new commit active, reboot your system.

```
# systemctl reboot
```

The Anaconda Installer reboots into the new deployment. On the login screen, you can see a new deployment available for you to boot.

h. To boot into the newest commit, run the following command to automatically order the boot entries so that the new deployment is first in the list:

```
$ rpm-ostree upgrade
```

Optionally, you can use the arrow key on your keyboard to select the GRUB menu entry and press **Enter**.

5. Provide your login user account credentials.
6. Verify the OSTree status:

```
$ rpm-ostree status
```

The command output provides the active commit ID.

7. To view the changed packages, if any, run a diff between the parent commit and the new commit:

```
$ rpm-ostree db diff parent_commit new_commit
```

The update shows that the package you have installed is available and ready for use.

15.3. UPGRADING RHEL FOR EDGE SYSTEMS

15.3.1. Upgrading your RHEL 8 system to RHEL 9

You can upgrade your RHEL 8 system to RHEL 9 by using the **rpm-ostree rebase** command. The command fully supports the default package set of RHEL for Edge upgrades from the most recent updates of RHEL 8 to the most recent updates of RHEL 9. The upgrade downloads and installs the RHEL 9 image in the background. After the upgrade finishes, you must reboot your system to use the new RHEL 9 image.

WARNING

The upgrade does not support every possible **rpm** package version and inclusions. You must test your package additions to ensure that these packages works as expected.

Prerequisites

- You have a running RHEL for Edge 8 system
- You have an OSTree repository server by HTTP

- You created a blueprint for RHEL for Edge 9 image that you will upgrade

Procedure

1. On the system where RHEL image builder runs, create a RHEL for Edge 9 image:
   a. Start the image compose:

   ```
   $ sudo composer-cli compose start blueprint-name edge-commit
   ```

   Optionally, you can also create the new RHEL for Edge 9 image by using a pre-existing OSTree repository, by using the following command:

   ```
   $ sudo composer-cli compose start-ostree --ref rhel/8/x86_64/edge --parent parent-OSTree-REF --url URL blueprint-name edge-commit
   ```

   b. After the compose finishes, download the image.
   c. Extract the downloaded image to **/var/www/html/** folder:

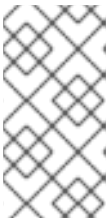
   ```
   $ sudo tar -xf image_file -C /var/www/html
   ```

   d. Start the **httpd** service:

   ```
   $ systemctl start httpd.service
   ```

2. On the RHEL for Edge device, check the current remote repository configuration:

   ```
   $ sudo cat /etc/ostree/remotes.d/edge.conf
   ```

   NOTE

   Depending on how your Kickstart file is configured, the **/etc/ostree/remotes.d** repository can be empty. If you configured your remote repository, you can see its configuration. For the **edge-installer**, **raw-image**, and **simplified-installer** images, the remote is configured by default.

3. Check the current URL repository:

   ```
   $ sudo ostree remote show-url edge
   ```

   edge is the of the Ostree repository.

4. List the remote reference branches:

   ```
   $ ostree remote refs edge
   ```

   You can see the following output:

   ```
   Error: Remote refs not available; server has no summary file
   ```

5. To add the new repository:

a. Configure the URL key to add a remote repository. For example:

```
$ sudo ostree remote add \ --no-gpg-verify rhel9 http://192.168.122.1/repo/
```

b. Configure the URL key to point to the RHEL 9 commit for the upgrade. For example:

```
$ sudo cat /etc/ostree/remotes.d/edge.conf

[remote "edge"]
url=http://192.168.122.1/ostree/repo/
gpg-verify=false
```

c. Confirm if the URL has been set to the new remote repository:

```
$ sudo cat /etc/ostree/remotes.d/rhel9.conf

[remote "edge"]
url=http://192.168.122.1/repo/
gpg-verify=false
```

d. See the new URL repository:

```
$ sudo ostree remote show-url rhel9 http://192.168.122.1/ostree-rhel9/repo/
```

e. List the current remote list options:

```
$ sudo ostree remote list

output:
edge
rhel9
```

6. Rebase your system to the RHEL version, providing the reference path for the RHEL 9 version:

```
$ rpm-ostree rebase rhel9:rhel/9/x86_64/edge
```

7. Reboot your system.

```
$ systemctl reboot
```

8. Enter your username and password.

9. Check the current system status:

```
$ rpm-ostree status
```

Verification

1. Check the current status of the currently running deployment:

```
$ rpm-ostree status
```

2. Optional: List the processor and tasks managed by the kernel in real-time.

```
$ top
```

3. If the upgrade does not support your requirements, you have the option to manually rollback to the previous stable deployment RHEL 8 version:

```
$ sudo rpm-ostree rollback
```

4. Reboot your system. Enter your username and password:

```
$ systemctl reboot
```

After rebooting, your system runs RHEL 9 successfully.

NOTE

If your upgrade succeeds and you do not want to use the previous deployment RHEL 8 version, you can delete the old repository:

```
$ sudo ostree remote delete edge
```

Additional resources

- rpm-ostree update and rebase fails with failed to find kernel error (Red Hat Knowledgebase)

15.4. DEPLOYING RHEL FOR EDGE AUTOMATIC IMAGE UPDATES

After you install a RHEL for Edge image on an Edge device, you can check for image updates available, if any, and can auto-apply them.

The **rpm-ostreed-automatic.service** (systemd service) and **rpm-ostreed-automatic.timer** (systemd timer) control the frequency of checks and upgrades. The available updates, if any, appear as staged deployments.

Deploying automatic image updates involves the following high-level steps:

- Update the image update policy
- Enable automatic download and staging of updates

15.4.1. Updating the RHEL for Edge image update policy

To update the image update policy, use the **AutomaticUpdatePolicy** and an **IdleExitTimeout** setting from the **rpm-ostreed.conf** file at **/etc/rpm-ostreed.conf** location on an Edge device.

The **AutomaticUpdatePolicy** settings controls the automatic update policy and has the following update checks options:

- **none**: Disables automatic updates. By default, the **AutomaticUpdatePolicy** setting is set to **none**.
- **check**: Downloads enough metadata to display available updates with **rpm-ostree** status.

- **stage**: Downloads and unpacks the updates that are applied on a reboot.

The **IdleExitTimeout** setting controls the time in seconds of inactivity before the daemon exit and has the following options:

- 0: Disables auto-exit.
- 60: By default, the **IdleExitTimeout** setting is set to **60**.

To enable automatic updates, perform the following steps:

Procedure

1. In the **/etc/rpm-ostreed.conf** file, update the following:
   - Change the value of **AutomaticUpdatePolicy** to **check**.
   - To run the update checks, specify a value in seconds for **IdleExitTimeout**.
2. Reload the **rpm-ostreed** service and enable the **systemd** timer.

```
# systemctl reload rpm-ostreed
# systemctl enable rpm-ostreed-automatic.timer --now
```

3. Verify the **rpm-ostree** status to ensure the automatic update policy is configured and time is active.

```
# rpm-ostree status
```

   The command output shows the following:

```
State: idle; auto updates enabled (check; last run <minutes> ago)
```

   Additionally, the output also displays information about the available updates.

15.4.2. Enabling RHEL for Edge automatic download and staging of updates

After you update the image update policy to check for image updates, the updates if any are displayed along with the update details. If you decide to apply the updates, enable the policy to automatically download and stage the updates. The available image updates are then downloaded and staged for deployment. The updates are applied and take effect when you reboot the Edge device.

To enable the policy for automatic download and staging of updates, perform the following updates:

Procedure

1. In the **/etc/rpm-ostreed.conf** file, update "AutomaticUpdatePolicy" to **stage**.
2. Reload the **rpm-ostreed** service.

```
# systemctl enable rpm-ostreed-automatic.timer --now
```

3. Verify the **rpm-ostree** status

```
# rpm-ostree status
```

The command output shows the following:

```
State: idle
AutomaticUpdates: stage; rpm-ostreed-automatic.timer: last run <time> ago
```

4. To initiate the updates, you can either wait for the timer to initiate the updates, or can manually start the service.

```
# systemctl start rpm-ostreed-automatic.service
```

After the updates are initiated, the **rpm-ostree** status shows the following:

```
# rpm-ostree status
State: busy
AutomaticUpdates: stage; rpm-ostreed-automatic.service: running
Transaction: automatic (stage)
```

When the update is complete, a new deployment is staged in the list of deployments, and the original booted deployment is left untouched. You can decide if you want to boot the system using the new deployment or can wait for the next update.

To view the list of deployments, run the **rpm-ostree status** command.

Following is a sample output.

```
# rpm-ostree status
State: idle
AutomaticUpdates: stage; rpm-ostreed-automatic.timer: last run <time> ago
Deployments:
```

To view the list of deployments with the updated package details, run the **rpm-ostree status -v** command.

15.5. ROLLING BACK RHEL FOR EDGE IMAGES

Because RHEL for Edge applies transactional updates to the operating system, you can either manually or automatically roll back the unsuccessful updates to the last known good state, which prevents system failure during updates. You can automate the verification and rollback process by using the **greenboot** framework.

The **greenboot** health check framework leverages **rpm-ostree** to run custom health checks on system startup. In case of an issue, the system rolls back to the last working state. When you deploy a **rpm-ostree** update, it runs scripts to check that critical services can still work after the update. If the system does not work, for example, due to some failed package, you can roll back the system to a previous stable version of the system. This process ensures that your RHEL for Edge device is in an operational state.

After you update an image, it creates a new image deployment while preserving the previous image deployment. You can verify whether the update was successful. If the update is unsuccessful, for example, due to a failed package, you can roll back the system to a previous stable version.

15.5.1. Introducing the greenboot checks

Greenboot is a Generic Health Check Framework for **systemd** available on **rpm-ostree** based systems. It contains the following RPM packages that you can install on your system:

- **greenboot** - a package that contains the following functionalities:
  - Checking provided scripts
  - Reboot the system if the check fails
  - Rollback to a previous deployment the reboot did not solve the issue.
- **greenboot-default-health-checks** - a set of optional and selected health checks provided by your **greenboot** system maintainers.

Greenboot works in a RHEL for Edge system by using health check scripts that run on the system to assess the system health and automate a rollback to the last healthy state in case of some software fails. These health checks scripts are available in the **/etc/greenboot/check/required.d** directory. **Greenboot** supports shell scripts for the health checks. Having a health check framework is especially useful when you need to check for software problems and perform system rollbacks on edge devices where direct serviceability is either limited or non-existent. When you install and configure health check scripts, it triggers the health checks to run every time the system starts.

You can create your own health check scripts to assess the health of your workloads and applications. These additional health check scripts are useful components of software problem checks and automatic system rollbacks.

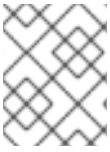
NOTE

You cannot use rollback in case of any health check failure on a system that is not using OSTree.

15.5.2. RHEL for Edge images roll back with greenboot

With RHEL for Edge images, only transactional updates are applied to the operating system. The transactional updates are atomic, which means that the updates are applied only if all the updates are successful, and there is support for rollbacks. With the transactional updates, you can easily rollback the unsuccessful updates to the last known good state, preventing system failure during updates.

Performing health checks is especially useful when you need to check for software problems and perform system rollbacks on edge devices where direct serviceability is limited or non-existent.



NOTE

You cannot use rollback in case of an update failure on a system that is not using OSTree, even if health checks might run.

You can use intelligent rollbacks with the **greenboot** health check framework to automatically assess system health every time the system starts. You can obtain pre-configured health from the **greenboot-default-health-checks** subpackage. These checks are located in the **/usr/lib/greenboot/check** read-only directory in **rpm-ostree** systems.

Greenboot leverages **rpm-ostree** and runs custom health checks that run on system startup. In case of an issue, the system rolls back the changes and preserves the last working state. When you deploy an

rpm-ostree update, it runs scripts to check that critical services can still work after the update. If the system does not work, the update rolls back to the last known working version of the system. This process ensures that your RHEL for Edge device is in an operational state.

You can obtain pre-configured health from the **greenboot-default-health-checks`subpackage. These checks are located in the `/usr/lib/greenboot/check** read-only directory in **rpm-ostree** systems. You can also configure shell scripts as the following types of checks:

Example 15.1. The greenboot directory structure

```
etc
└─ greenboot
  ├─ check
  |  └─ required.d
  |  └─ init.py
  └─ green.d
  └─ red.d
```

Required

Contains the health checks that must not fail. Place required shell scripts in the **/etc/greenboot/check/required.d** directory. If the scripts fail, greenboot retries them three times by default. You can configure the number of retries in the **/etc/greenboot/greenboot.conf** file by setting the **GREENBOOT_MAX_BOOTS** parameter to the number of retries you want.

After all retries fail, **greenboot** automatically initiates a rollback if one is available. If a rollback is not available, the system log output shows that you need to perform a manual intervention.

Wanted

Contains the health checks that might fail without causing the system to roll back. Place wanted shell scripts in the **/etc/greenboot/check/wanted.d** directory. **Greenboot** informs that the script fails, the system health status remains unaffected and it does not perform a rollback neither a reboot.

You can also specify shell scripts that will run after a check:

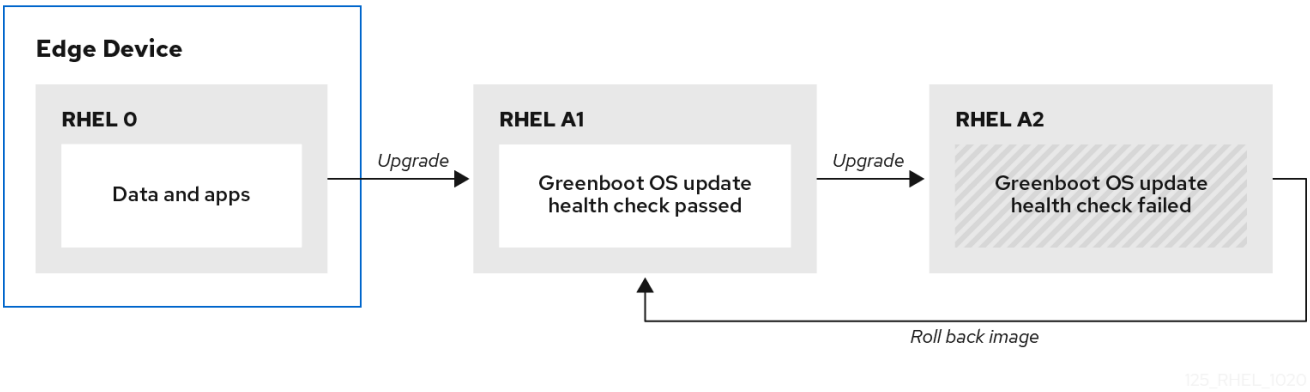
Green

Contains the scripts to run after a successful boot. Place these scripts into the **/etc/greenboot/green.d`directory**. **Greenboot** informs that the boot was successful.

Red

Contains the scripts to run after a failed boot. Place these scripts into the **/etc/greenboot/red.d** directory. The system attempts to boot three times and in case of failure, it executes the scripts. **Greenboot** informs that the boot failed.

The following diagram illustrates the RHEL for Edge image roll back process.



After booting the updated operating system, **greenboot** runs the scripts in the **required.d** and **wanted.d** directories. If any of the scripts fail in the **required.d** directory, **greenboot** runs any scripts in the **red.d** directory, and then reboots the system.

Greenboot makes 2 more attempts to boot on the upgraded system. If during the third boot attempt the scripts in **required.d** are still failing, **greenboot** runs the **red.d** scripts one last time, to ensure that the script in the **red.d** directory tried to make a corrective action to fix the issue and this was not successful. Then, **greenboot** rollbacks the system from the current **rpm-ostree** deployment to the previous stable deployment.

15.5.3. Greenboot health check status

When deploying your updated system, wait until the greenboot health checks have finished before making the changes to ensure that those changes are not lost if greenboot rolls the system back to an earlier state. If you want to make configuration changes or deploy applications you must wait until the greenboot health checks have finished. This ensures that your changes are not lost if greenboot rolls your **rpm-ostree** system back to an earlier state.

The **greenboot-healthcheck** service runs once and then exits. You can check the status of the service to know if it is done, and to know the outcome, by using the following commands:

systemctl is-active greenboot-healthcheck.service

This command reports **active** when the service has exited. If it the service did not even run it shows **inactive**.

systemctl show --property=SubState --value greenboot-healthcheck.service

Reports **exited** when done, **running** while still running.

systemctl show --property=Result --value greenboot-healthcheck.service

Reports **success** when the checks passed.

systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service

Reports the numerical exit code of the service, 0 means success and nonzero values mean a failure occurred.

cat /run/motd.d/boot-status

Shows a message, such as "Boot Status is GREEN - Health Check SUCCESS".

15.5.4. Checking greenboot health checks statuses

Check the status of greenboot health checks before making changes to the system or during troubleshooting. You can use any of the following commands to help you ensure that greenboot scripts have finished running.

- Use one of the following options to check the statuses:
  - To see a report of health check status, enter:

    ```
    $ systemctl show --property=SubState --value greenboot-healthcheck.service
    ```

    The following outputs are possible:

    - **start** means that greenboot checks are still running.
    - **exited** means that checks have passed and greenboot has exited. Greenboot runs the scripts in the **green.d** directory when the system is in a healthy state.
    - **failed** means that checks have not passed. Greenboot runs the scripts in **red.d** directory when the system is in this state and might restart the system.
  - To see a report showing the numerical exit code of the service, where **0** means success and nonzero values mean a failure occurred, use the following command:

    ```
    $ systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service
    ```
  - To see a report showing a message about boot status, such as **Boot Status is GREEN - Health Check SUCCESS**, enter:

    ```
    $ cat /run/motd.d/boot-status
    ```

15.5.5. Manually rolling back RHEL for Edge images

When you upgrade your operating system, a new deployment is created, and the **rpm-ostree** package also keeps the previous deployment. If there are issues on the updated version of the operating system, you can manually roll back to the previous deployment with a single **rpm-ostree** command, or by selecting the previous deployment in the GRUB boot loader.

To manually roll back to a previous version, perform the following steps.

Prerequisites

1. You updated your system and it failed.

Procedure

1. Optional: Check for the fail error message:

   ```
   $ journalctl -u greenboot-healthcheck.service.
   ```

2. Run the **rollback** command:

   ```
   # rpm-ostree rollback
   ```

   The command output provides details about the commit ID that is being moved and indicates a completed transaction with the details of the package being removed.

3. Reboot the system.

```
# systemctl reboot
```

The command activates the previous commit with the stable content. The changes are applied and the previous version is restored.

15.5.6. Rolling back RHEL for Edge images using an automated process

Greenboot checks provides a framework that is integrated into the boot process and can trigger **rpm-ostree** rollbacks when a health check fails. For the health checks, you can create a custom script that indicates whether a health check passed or failed. Based on the result, you can decide when a rollback should be triggered. The following procedure shows how to create an health check script example:

Procedure

1. Create a script that returns a standard exit code **0**.
   For example, the following script ensures that the configured DNS server is available:

```
#!/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. Include an executable file for the health checks at **/etc/greenboot/check/required.d/**.

```
chmod +x check-dns.sh
```

   During the next reboot, the script is executed as part of the boot process, before the system enters the **boot-complete.target** unit. If the health checks are successful, no action is taken. If the health checks fail, the system will reboot several times, before marking the update as failed and rolling back to the previous update.

Verification

To check if the default gateway is reachable, run the following health check script:

1. Create a script that returns a standard exit code **0**.

```
#!/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. Include an executable file for the health checks at **/etc/greenboot/check/required.d/** directory.

```
chmod +x check-gw.sh
```

CHAPTER 16. CREATING AND MANAGING OSTREE IMAGE UPDATES

You can easily create and manage OStree image updates for your RHEL for Edge systems and make them immediately available to RHEL for Edge devices. With OSTree, you can use image builder to create RHEL for Edge Commit or RHEL for Edge Container images as **.tar** files that contain OSTree commits. The OSTree update versioning system works as a "Git repository" that stores and versions the OSTree commits. The **rpm-ostree** image and package system then assembles the commits on the client device. When you create a new image with RHEL image builder to perform an update, RHEL image builder pulls updates from these repositories.

16.1. BASIC CONCEPTS FOR OSTREE

Basic terms that OSTree and **rpm-ostree** use during image updates.

rpm-ostree

The technology on the edge device that handles how the OSTree commits are assembled on the device. It works as a hybrid between an image and a package system. With the **rpm-ostree** technology, you can make atomic upgrades and rollbacks to your system.

OSTree

OSTree is a technology that enables you to create commits and download bootable file system trees. You can also use it to deploy the trees and manage the boot loader configuration.

Commit

An OSTree commit contains a full operating system that is not directly bootable. To boot the system, you must deploy it, for example, with a RHEL Installable image.

Reference

It is also known as **ref**. An OSTree ref is similar to a Git branch and it is a name. The following reference names examples are valid:

- **rhel/9/x86_64/edge**
- **ref-name**
- **app/org.gnome.Calculator/x86_64/stable**
- **ref-name-2**

By default, image builder specifies **rhel/9/$ARCH/edge** as a path. The "$ARCH" value is determined by the host machine.

Parent

The **parent** argument is an OSTree commit that you can provide to build a new commit with image builder. You can use the **parent** argument to specify an existing **ref** that retrieves a parent commit for the new commit that you are building. You must specify the parent commit as a ref value to be resolved and pulled, for example **rhel/9/x86_64/edge**. You can use the **--parent** commit for the RHEL for Edge Commit (**.tar**) and RHEL for Edge Container (**.tar**) image types.

Remote

The http or https endpoint that hosts the OSTree content. This is analogous to the baseurl for a yum repository.

Static delta

Static deltas are a collection of updates generated between two OSTree commits. This enables the system client to fetch a smaller amount of files, which are larger in size. The static deltas updates are more network efficient because, when updating an ostree-based host, the system client will only fetch the objects from the new OSTree commit which do not exist on the system. Typically, the new OSTree commit contains many small files, which requires multiple TCP connections.

Summary

The summary file is a concise way of enumerating refs, checksums, and available static deltas in an OSTree repo. You can check the state of all the refs and static deltas available in an Ostree repo. However, you must generate the summary file every time a new ref, commit, or static-delta is added to the OSTree repo.

16.2. CREATING OSTREE REPOSITORIES

You can create OSTree repos with RHEL image builder by using either **RHEL for Edge Commit (.tar)** or **RHEL for Edge Container (.tar)** image types. These image types contain an OSTree repo that contains a single OSTree commit.

- You can extract the **RHEL for Edge Commit (.tar)** on a web server and it is ready to be served.
- You must import the **RHEL for Edge Container (.tar)** to a local container image storage or push the image to a container registry. After you start the container, it serves the commit over an integrated **nginx** web server.

Use the **RHEL for Edge Container (.tar)** on a RHEL server with Podman to create an OSTree repo:

Prerequisites

- You created a **RHEL for Edge Container (.tar)** image.

Procedure

1. Download the container image from image builder:

```
$ composer-cli compose image _<UUID>
```

2. Import the container into Podman:

```
$ skopeo copy oci-archive:_<UUID>_-container.tar containers-storage:localhost/ostree
```

3. Start the container and make it available by using the port **8080**:

```
$ podman run -rm -p 8080:8080 ostree
```

Verification

- Check that the container is running:

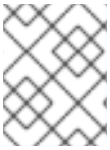
```
$ podman ps -a
```

16.3. MANAGING A CENTRALIZED OSTREE MIRROR

For production environments, having a central OSTree mirror that serves all the commits has several advantages, including:

- Deduplicating and minimizing disk storage
- Optimizing the updates to clients by using static delta updates
- Pointing to a single OSTree mirror for their deployment life.

To manage a centralized OSTree mirror, you must pull each commit from image builder into the centralized repository where it will be available to your users.



NOTE

You can also automate managing an OSTree mirror by using the **infra.osbuild** Ansible collection. See osbuild.infra Ansible.

To create a centralized repository you can run the following commands directly on a web server:

Procedure

1. Create an empty blueprint, customizing it to use "rhel-92" as the distro:

```
name = "minimal-rhel92"
description = "minimal blueprint for ostree commit"
version = "1.0.0"
modules = []
groups = []
distro = "rhel-92"
```

2. Push the blueprint to the server:

```
# composer-cli blueprints push minimal-rhel92.toml
```

3. Build a RHEL for Edge Commit (**.tar**) image from the blueprint you created:

```
# composer-cli compose start-ostree minimal-rhel92 edge-commit
```

4. Retrieve the **.tar file** and decompress it to the disk:

```
# composer-cli compose image _<rhel-92-uuid>
$ tar -xf <rhel-92-uuid>.tar -C /usr/share/nginx/html/
```

The **/usr/share/nginx/html/repo** location on disk will become the single OSTree repo for all refs and commits.

5. Create another empty blueprint, customizing it to use "rhel-87" as the distro:

```
name = "minimal-rhel87"
description = "minimal blueprint for ostree commit"
version = "1.0.0"
modules = []
groups = []
distro = "rhel-87"
```

6. Push the blueprint and create another RHEL for Edge Commit (**.tar**) image:

```
# composer-cli blueprints push minimal-rhel87.toml
# composer-cli compose start-ostree minimal-rhel87 edge-commit
```

7. Retrieve the **.tar file** and decompress it to the disk:

```
# composer-cli compose image <rhel-87-uuid>
$ tar -xf <rhel-87-uuid>.tar
```

8. Pull the commit to the local repo. By using **ostree pull-local**, you can copy the commit data from one local repo to another local repo.

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo
```

9. Optional: Inspect the status of the OSTree repo. The following is an output example:

```
$ 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. Update the RHEL 9.2 blueprint to include a new package and build a new commit, for example:

```
name = "minimal-rhel92"
description = "minimal blueprint for ostree commit"
version = "1.1.0"
modules = []
groups = []
distro = "rhel-92"
[[packages]]
name = "strace"
version = "*"
```

11. Push the updated blueprint and create a new RHEL for Edge Commit (**.tar**) image, pointing the compose to the existing OSTree repo:

```
# 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. Retrieve the **.tar** file and decompress it to the disk:

```
# rm -rf repo
# composer-cli compose image <rhel-92-uuid>
# tar -xf <rhel-92-uuid>.tar
```

13. Pull the commit to repo:

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo
```

14. Optional: Inspect the OSTree repo status again:

```
$ 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
```

16.4. PERFORMING UPDATES BY USING STATIC DELTAS

You can use static deltas to improve the speed of your **client** repository updates. If static deltas for an update exist, they are used. If not, the update still happens, but with a lower speed.

After building an OSTree system, if you want to retrieve the content from your client systems, you need to create a repository management.

By default, the repository you use for **production** requires one HTTP fetch transaction per client request. If you perform releases only once a week, use “static deltas” to get client repository updates faster.

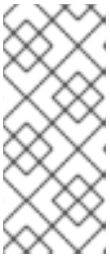
Prerequisites

- You have a RHEL **repo** repository running with Podman.
- You have created a centralized OSTree mirror to use as the production repository.

Procedure

1. Pull the content update from the **repo** repository into the **repo-prod** repository:

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo-prod
```



NOTE

If you do not specify the **refs** branch, the **ostree-pull-local** command retrieves all the branches. As an alternative to the **ostree-pull-local** command, you can also use the **ostree-pull** command. The difference is that **ostree-pull-local** is optimized for copies only between repositories on the same system, while **ostree-pull** can download data from a remote repository.

2. Apply a delta update against the previous commit. The static generation happens on the local repository, in this case, **prod**. After the delta update is generated, the update is done offline.

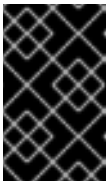
NOTE

You must have at least two commits in the repository where you want to generate static deltas.

```
# ostree --repo=/usr/share/nginx/html/client-server static-delta generate
exampleos/x86_64/standard
```

- Optional: If you want to upgrade from the two previous commits, run the following command:

```
# ostree --repo=repo-prod static-delta generate --from=exampleos/x86_64/standard^^ --
to=exampleos/x86_64/standard
```



IMPORTANT

Generating a full permutation of deltas across all previous versions can be a heavy workload for your systems. The OSTree core has some support for static deltas which refer to a parent.

3. Update the summary file:

```
# ostree --repo=repo-prod summary -u
```

You need to update the summary file, even if you choose to generate static deltas. The summary command cannot run concurrently. It must be triggered serially by other jobs. That is, you must generate a summary file after generating the static-deltas, because the previous command added a new static delta.

4. Use the generated static delta to apply to the existing Operating System and perform updates:

```
# ostree --repo=/ostree/repo-prod static-delta <delta-update-file-path>
```

5. Apply the OSTree delta update:

```
$ sudo rpm-ostree upgrade
```

6. Reboot into the new image to apply the updates:

```
$ systemctl reboot
```

APPENDIX A. TERMINOLOGY AND COMMANDS

Learn more about the **rpm ostree** terminology and commands.

A.1. OSTREE AND RPM-OSTREE TERMINOLOGY

Following are some helpful terms that are used in context to OSTree and **rpm-ostree** images.

Table A.1. OSTree and rpm-ostree terminology

Term	Definition
OSTree	A tool used for managing Linux-based operating system versions. The OSTree tree view is similar to Git and is based on similar concepts.
rpm-ostree	A hybrid image or system package that hosts operating system updates.
Commit	A release or image version of the operating system. RHEL image builder generates an OSTree commit for RHEL for Edge images. You can use these images to install or update RHEL on Edge servers.
Refs	Represents a branch in OSTree. Refs always resolve to the latest commit. For example, **rhel/9/x86_64/edge**.
Revision (Rev)	SHA-256 for a specific commit.
Remote	The http or https endpoint that hosts the OSTree content. This is analogous to the baseurl for a dnf repository.
static-delta	Updates to OSTree images are always delta updates. In case of RHEL for Edge images, the TCP overhead can be higher than expected due to the updates to number of files. To avoid TCP overhead, you can generate static-delta between specific commits, and send the update in a single connection. This optimization helps large deployments with constrained connectivity.

A.2. OSTREE COMMANDS

The following table provides a few OSTree commands that you can use when installing or managing OSTree images.

Table A.2. ostree commands

ostree pull	**ostree pull-local --repo [path] src** **ostree pull-local <path> <rev> --repo=<repo-path>** **ostree pull <URL> <rev> --repo=<repo-path>**
ostree summary	**ostree summary -u --repo=<repo-path>**
View refs	**ostree refs --repo ~/Code/src/osbuild-iot/build/repo/ --list**
View commits in repo	**ostree log --repo=/home/gicmo/Code/src/osbuild-iot/build/repo/ <REV>**
Inspect a commit	**ostree show --repo build/repo <REV>**
List remotes of a repo	**ostree remote list --repo <repo-path>**
Resolve a REV	**ostree rev-parse --repo ~/Code/src/osbuild-iot/build/repo fedora/x86_64/osbuild-demo** **ostree rev-parse --repo ~/Code/src/osbuild-iot/build/repo b3a008eceeddd0cfd**
Create static-delta	**ostree static-delta generate --repo=[path] --from=REV --to=REV**
Sign an **existing** ostree commit with a GPG key	**ostree gpg-sign --repo=<repo-path> --gpg-homedir <gpg_home> COMMIT KEY-ID…**

A.3. RPM-OSTREE COMMANDS

The following table provides a few **rpm-ostree** commands that you can use when installing or managing OSTree images.

Table A.3. rpm-ostree commands

Commands	Description
rpm-ostree --repo=/home/gicmo/Code/src/osbuild-iot/build/repo/ db list <REV>	This command lists the packages existing in the <REV> commit into the repository.
rpm-ostree install	This command installs one or more RPM packages. You cannot place binaries in the **/opt** directory by using **rpm-ostree install**. You must rebuild the RPM package so that it uses **/usr/opt** or **/usr/local/opt**.

Commands	Description
rpm-ostree rollback	OSTree manages an ordered list of boot loader entries, called **deployments**. The entry at index 0 is the default boot loader entry. Each entry has a separate **/etc** directory, but all the entries share a single **/var** directory. You can use the boot loader to choose between entries by pressing Tab to interrupt startup. This rolls back to the previous state, that is, the default deployment changes places with the non-default one.
rpm-ostree status	This command gives information about the current deployment in use. Lists the names and **refspecs** of all possible deployments in order, such that the first deployment in the list is the default upon boot. The deployment marked with * is the current booted deployment, and marking with 'r' indicates the most recent upgrade.
rpm-ostree db list	Use this command to see which packages are within the commit or commits. You must specify at least one commit, but more than one or a range of commits also work.
rpm-ostree db diff	Use this command to show how the packages are different between the trees in two revs (revisions). If no revs are provided, the booted commit is compared to the pending commit. If only a single rev is provided, the booted commit is compared to that rev.
rpm-ostree upgrade	This command downloads the latest version of the current tree, and deploys it, setting up the current tree as the default for the next boot. This has no effect on your running filesystem tree. You must reboot for any changes to take effect.

Additional resources

- **rpm-ostree** man page on your system

A.4. FDO AUTOMATIC ONBOARDING TERMINOLOGY

Learn more about the FDO terminology.

Table A.4. FDO terminology

Commands	Description
FDO	FIDO Device Onboarding.
Device	Any hardware, device, or computer.
Owner	The final owner of the device - a company or an IT department.

Commands	Description
Manufacturer	The device manufacturer.
Manufacturer server	Creates the device credentials for the device.
Manufacturer client	Informs the location of the manufacturing server.
Ownership Voucher (OV)	Record of ownership of an individual device. Contains the following information: * Owner (**fdo-owner-onboarding-service**) * Rendezvous Server - FIDO server (**fdo-rendezvous-server**) * Device (at least one combination) (**fdo-manufacturing-service**)
Device Credential (DC)	Key credential and rendezvous stored in the device at manufacture.
Keys	Keys to configure the manufacturing server * key_path * cert_path * key_type * mfg_string_type: device serial number * allowed_key_storage_types: Filesystem and Trusted Platform Module (TPM) that protects the data used to authenticate the device you are using.
Rendezvous server	Link to a server used by the device and later on, used on the process to find out who is the owner of the device

Additional resources

- FIDO IoT spec

A.5. FDO AUTOMATIC ONBOARDING TECHNOLOGIES

Following are the technologies used in context to FDO automatic onboarding.

Table A.5. OSTree and rpm-ostree terminology

Technology	Definition
UEFI	Unified Extensible Firmware Interface.
RHEL	Red Hat® Enterprise Linux® operating system
rpm-ostree	Background image-based upgrades.
Greenboot	**Healthcheck** framework for systemd on **rpm-ostree**.
Osbuild	Pipeline-based build system for operating system artifacts.
Container	A Linux® container is a set of 1 or more processes that are isolated from the rest of the system.
Coreos-installer	Assists installation of RHEL images, boots systems with UEFI.
FIDO FDO	Specification protocol to provision configuration and onboarding devices.