



Red Hat Enterprise Linux 9

Composing a customized RHEL system image

Creating customized system images with Image Builder on Red Hat Enterprise Linux

9

Red Hat Enterprise Linux 9 Composing a customized RHEL system image

Creating customized system images with Image Builder on Red Hat Enterprise Linux 9

Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

Image Builder is a tool for creating deployment-ready customized system images: installation disks, virtual machines, cloud vendor-specific images, and others. Using Image Builder, you can create these images faster if compared to manual procedures, because it eliminates the specific configurations required for each output type. This document describes how to set up Image Builder and create images with it.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	5
CHAPTER 1. IMAGE BUILDER DESCRIPTION	6
1.1. INTRODUCTION TO IMAGE BUILDER	6
1.2. IMAGE BUILDER TERMINOLOGY	6
1.3. IMAGE BUILDER OUTPUT FORMATS	6
1.4. IMAGE BUILDER SYSTEM REQUIREMENTS	7
CHAPTER 2. INSTALLING IMAGE BUILDER	9
2.1. INSTALLING IMAGE BUILDER IN A VIRTUAL MACHINE	9
CHAPTER 3. MANAGING REPOSITORIES	11
3.1. IMAGE BUILDER DEFAULT SYSTEM REPOSITORIES	11
3.2. OVERRIDING A SYSTEM REPOSITORY	11
3.3. OVERRIDING A SYSTEM REPOSITORY WITH SUPPORT FOR SUBSCRIPTIONS	12
CHAPTER 4. CREATING SYSTEM IMAGES WITH IMAGE BUILDER COMMAND-LINE INTERFACE	14
4.1. IMAGE BUILDER COMMAND-LINE INTERFACE	14
4.2. CREATING AN IMAGE BUILDER BLUEPRINT WITH COMMAND-LINE INTERFACE	14
4.3. EDITING AN IMAGE BUILDER BLUEPRINT WITH COMMAND-LINE INTERFACE	16
4.4. CREATING A SYSTEM IMAGE WITH IMAGE BUILDER IN THE COMMAND-LINE INTERFACE	17
4.5. BASIC IMAGE BUILDER COMMAND-LINE COMMANDS	18
4.6. IMAGE BUILDER BLUEPRINT FORMAT	20
4.7. SUPPORTED IMAGE CUSTOMIZATIONS	21
4.8. INSTALLED PACKAGES	25
4.9. ENABLED SERVICES	28
CHAPTER 5. CREATING SYSTEM IMAGES WITH IMAGE BUILDER WEB CONSOLE INTERFACE	30
5.1. ACCESSING IMAGE BUILDER GUI IN THE RHEL WEB CONSOLE	30
5.2. CREATING AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE	30
5.3. EDITING AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE	31
5.4. ADDING USERS AND GROUPS TO AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE	33
5.5. CREATING A SYSTEM IMAGE WITH IMAGE BUILDER IN THE WEB CONSOLE INTERFACE	35
5.6. ADDING A SOURCE TO A BLUEPRINT	36
5.7. CREATING A USER ACCOUNT FOR A BLUEPRINT	37
5.8. CREATING A USER ACCOUNT WITH SSH KEY	39
CHAPTER 6. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER	42
6.1. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER IN THE COMMAND-LINE INTERFACE	42
6.2. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER IN THE GUI	43
6.3. INSTALLING THE ISO IMAGE TO A BARE METAL SYSTEM	44
CHAPTER 7. PREPARING AND DEPLOYING KVM GUEST IMAGES WITH IMAGE BUILDER	46
7.1. CREATING CUSTOMIZED KVM GUEST IMAGES WITH IMAGE BUILDER	46
7.2. CREATING A VIRTUAL MACHINE FROM A KVM GUEST IMAGE	47
CHAPTER 8. PREPARING AND UPLOADING CLOUD IMAGES WITH IMAGE BUILDER	50
8.1. PREPARING FOR UPLOADING AWS AMI IMAGES	50
8.2. UPLOADING AN AMI IMAGE TO AWS IN THE CLI	51
8.3. PUSHING IMAGES TO AWS CLOUD AMI	52

8.4. PREPARING FOR UPLOADING AZURE VHD IMAGES	54
8.5. UPLOADING VHD IMAGES TO AZURE	56
8.6. UPLOADING VMDK IMAGES TO VSPHERE	57
8.7. PUSHING VMWARE IMAGES TO VSPHERE	59
8.8. PUSHING VHD IMAGES TO AZURE CLOUD	61
8.9. PUSHING CUSTOMIZED IMAGES TO OCI	64
8.10. UPLOADING QCOW2 IMAGE TO OPENSTACK	66
8.11. PREPARING FOR UPLOADING IMAGES TO ALIBABA	68
8.12. UPLOADING IMAGES TO ALIBABA	69
8.13. IMPORTING IMAGES TO ALIBABA	70
8.14. CREATING AN INSTANCE OF A CUSTOM IMAGE USING ALIBABA	71

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Bugzilla (account required)

1. Log in to the [Bugzilla](#) website.
2. Select the correct version from the **Version** menu.
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 **Submit Bug**.

CHAPTER 1. IMAGE BUILDER DESCRIPTION

1.1. INTRODUCTION TO IMAGE BUILDER

You can use Image Builder to create customized system images of Red Hat Enterprise Linux, including system images prepared for deployment on cloud platforms. Image Builder automatically handles details of setup for each output type and is thus easier to use and faster to work with than manual methods of image creation. You can access Image Builder functionality through a command-line interface in the **composer-cli** tool, or a graphical user interface in the RHEL web console.

As of Red Hat Enterprise Linux 8.3, the **osbuild-composer** backend replaces **lorax-composer**. The new service provides REST APIs for image building. As a result, users can benefit from a more reliable backend and more predictable output images.

Image Builder runs as a system service **osbuild-composer**. You can interact with this service through two interfaces:

- CLI tool **composer-cli** for running commands in the terminal. This method is preferred.
- GUI plugin for the RHEL web console.

1.2. IMAGE BUILDER TERMINOLOGY

Blueprint

Blueprints define customized system images by listing packages and customizations that will be part of the system. Blueprints can be edited and they are versioned. When a system image is created from a blueprint, the image is associated with the blueprint in the Image Builder interface of the RHEL web console.

Blueprints are presented to the user as plain text in the TOML format.

Compose

Composes are individual builds of a system image, based on a particular version of a particular blueprint. Compose as a term refers to the system image, the logs from its creation, inputs, metadata, and the process itself.

Customizations

Customizations are specifications for the system, which are not packages. This includes users, groups, and SSH keys.

1.3. IMAGE BUILDER OUTPUT FORMATS

Image Builder can create images in multiple output formats shown in the following table. To check the supported types, run the command:

```
# composer-cli compose types
```

Table 1.1. Image Builder output formats

Description	CLI name	file extension
QEMU QCOW2 Image	qcow2	.qcow2

Description	CLI name	file extension
TAR Archive	tar	.tar
Amazon Machine Image Disk	ami	.raw
Azure Disk Image	vhd	.vhd
VMware Virtual Machine Disk	vmdk	.vmdk
Openstack	openstack	.qcow2
RHEL for Edge Commit	edge-commit	.tar
RHEL for Edge Container	edge-container	.tar
RHEL for Edge Installer	edge-installer	.iso
RHEL for Edge Raw	edge-raw-image	.tar
RHEL for Edge Simplified Installer	edge-simplified-installer	.iso
ISO image	image-installer	.iso

1.4. IMAGE BUILDER SYSTEM REQUIREMENTS

The environment where Image Builder runs, for example a dedicated virtual machine, must meet requirements listed in the following table.

Table 1.2. Image Builder system requirements

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



NOTE

Internet connectivity is not a prerequisite. You can use Image Builder in isolated networks if you reconfigure it to not connect to Red Hat CDN.

CHAPTER 2. INSTALLING IMAGE BUILDER

Before using Image Builder, you must install Image Builder in a virtual machine.

2.1. INSTALLING IMAGE BUILDER IN A VIRTUAL MACHINE

To install Image Builder on a dedicated virtual machine, follow these steps:

Prerequisites

- Connect to the virtual machine.
- The virtual machine for Image Builder must be installed, subscribed to Red Hat Subscription Manager (RHSM) or Red Hat Satellite, and running.

Procedure

1. Install the Image Builder and other necessary packages on the virtual machine:

- **osbuild-composer** – supported from RHEL 8.3 onward
- **composer-cli**
- **cockpit-composer**
- **bash-completion**

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

The web console is installed as a dependency of the *cockpit-composer* package.

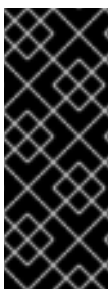
2. Enable Image Builder to start after each reboot:

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

The **osbuild-composer** and **cockpit** services start automatically on first access.

3. 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
```



IMPORTANT

The **osbuild-composer** package is the new backend engine that will be the preferred default and focus of all new functionality beginning with Red Hat Enterprise Linux 8.3 and later. The previous backend **lorax-composer** package is considered deprecated, will only receive select fixes for the remainder of the Red Hat Enterprise Linux 8 life cycle and will be omitted from future major releases. It is recommended to uninstall **lorax-composer** in favor of **osbuild-composer**.

Verification

You can use a system journal to track Image Builder service activities. Additionally, you can find the log messages in the file.

- To find the journal output for traceback, run the following commands:

```
$ journalctl | grep osbuild
```

- To show both remote or local workers:

```
$ journalctl -u osbuild-worker*
```

- To show the running services:

```
$ journalctl -u osbuild-composer.service
```

CHAPTER 3. MANAGING REPOSITORIES

3.1. IMAGE BUILDER DEFAULT SYSTEM REPOSITORIES

The **osbuild-composer** backend 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. 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 here take precedence over those in the **/usr** directory.

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

3.2. OVERRIDING A SYSTEM REPOSITORY

You can configure a repository override in the **/etc/osbuild-composer/repositories** directory with the following steps.

Prerequisites

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

Procedure

1. Create a directory that contains the repository overrides you want to use:

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

2. Create a JSON file with the following structure, for example:

```
{
  "<ARCH>": [
    {
      "name": "baseos",
      "metalink": "",
      "baseurl": "http://mirror.example.com/composes/released/RHEL-
9/9.0/BaseOS/x86_64/os/",
      "mirrorlist": "",
      "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (...)",
      "check_gpg": true,
      "metadata_expire": ""
    }
  ]
}
```

Specify only one of the following attributes: **metalink**, **mirrorlist**, or **baseurl**. The remaining fields are optional.

3. Save the JSON file using a name corresponding to your RHEL version, for example:

```
/etc/osbuild-composer/repositories/rhel-90.json
```

- a. Alternatively, you can copy the JSON file for your distribution from **/usr/share/osbuild-composer/** and modify its content.
- i. Copy the repository file to the directory you created.

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

Replace `rhel-version.json` with your RHEL version, for example: `rhel-9.json`

4. Using a text editor, edit the **baseurl** paths in the **rhel-9.json** file. For example:

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

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

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

As a result, the repository points to the correct URLs which are copied from the **/etc/yum.repos.d/redhat.repo** file.

Additional resources

- [latest RPMs version available in repository not visible for osbuild-composer.](#)

3.3. OVERRIDING A SYSTEM REPOSITORY WITH SUPPORT FOR SUBSCRIPTIONS

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

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

Prerequisites

- System with 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:

```
[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
    }
  ]
}
```



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** using **composer-cli**. System repository overrides are usually used for “BaseOS” and “AppStream” repositories, whereas **composer-cli** sources are used for all the other repositories.

Additional resources

- [image builder uses CDN repositories when host is registered to Satellite 6](#)

CHAPTER 4. CREATING SYSTEM IMAGES WITH IMAGE BUILDER COMMAND-LINE INTERFACE

Image Builder is a tool for creating custom system images. To control Image Builder and create your custom system images, use the command-line interface which is currently the preferred method to use Image Builder.

4.1. IMAGE BUILDER COMMAND-LINE INTERFACE

Image Builder command-line interface is currently the preferred method to use Image Builder. It offers more functionality than the [web console interface](#). To use this interface, run the **composer-cli** command with suitable options and subcommands.

The workflow for the command-line interface can be summarized as follows:

1. Export (save) the blueprint definition to a plain text file
2. Edit this file in a text editor
3. Import (*push*) the blueprint text file back into Image Builder
4. Run a compose to build an image from the blueprint
5. Export the image file to download it

Apart from the basic subcommands to achieve this procedure, the **composer-cli** command offers many subcommands to examine the state of configured blueprints and composes.

To run the **composer-cli** commands as non-root, user must be in the **weldr** or **root** groups.

- To add a user to the **weldr** or **root** groups, run the following commands:

```
$ sudo usermod -a -G weldr user
$ newgrp weldr
```

4.2. CREATING AN IMAGE BUILDER BLUEPRINT WITH COMMAND-LINE INTERFACE

This procedure describes how to create a new Image Builder blueprint using the command-line interface.

Procedure

1. Create a plain text file with the following contents:

```
name = "BLUEPRINT-NAME"
description = "LONG FORM DESCRIPTION TEXT"
version = "0.0.1"
modules = []
groups = []
```

Replace *BLUEPRINT-NAME* and *LONG FORM DESCRIPTION TEXT* with a name and description for your blueprint.

Replace *0.0.1* with a version number according to the [Semantic Versioning](#) scheme.

- For every package that you want to be included in the blueprint, add the following lines to the file:

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

Replace *package-name* with name of the package, such as **httpd**, **gdb-doc**, or **coreutils**.

Replace *package-version* with a version to use. This field supports **dnf** version specifications:

- For a specific version, use the exact version number such as **8.6.0**.
 - For latest available version, use the asterisk *****.
 - For the latest minor version, use formats such as **8.***.
- You can customize your blueprints in a number of ways. For this example, Simultaneous Multi Threading (SMT) can be disabled by performing the steps below. For additional customizations available, see [Supported Image Customizations](#).

```
[customizations.kernel]
append = "nosmt=force"
```

- Save the file as *BLUEPRINT-NAME.toml* and close the text editor.
- Push (import) the blueprint:

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

Replace *BLUEPRINT-NAME* with the value you used in previous steps.

- List the existing blueprints to verify that the blueprint has been pushed and exists:

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

- To display the blueprint configuration you have just added, run the command:

```
# composer-cli blueprints show
```

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

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



NOTE

To create images using the **composer-cli** command as non-root, add your user to the **weldr** or **root** groups.

Verification

If Image Builder is unable to depolve a package from your custom repositories, follow the steps:

- Remove the osbuild-composer cache:

```
$ sudo rm -rf /var/cache/osbuild-composer/*
$ sudo systemctl restart osbuild-composer
```

Additional resources

- [osbuild-composer is unable to depolve a package from my custom repository](#)
- [Composing a customized RHEL system image with proxy server](#)

4.3. EDITING AN IMAGE BUILDER BLUEPRINT WITH COMMAND-LINE INTERFACE

To edit an existing Image Builder blueprint in the command-line interface, follow the steps.

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 and make your changes.
3. Before finishing with the edits, make sure the file is a valid blueprint:

- a. Remove this line, if present:

```
packages = []
```

- b. Increase the version number. Remember that Image Builder blueprint versions must use the [Semantic Versioning](#) scheme. Note also that if you do not change the version, the **patch** version component increases automatically.
- c. 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>

4. Save the file and close the text editor.
5. Push (import) the blueprint back into Image Builder:

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

**NOTE**

To import the blueprint back into Image Builder, supply the file name including the **.toml** extension, while in other commands use only the blueprint name.

6. To verify that the contents uploaded to Image Builder match your edits, list the contents of blueprint:

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

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

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

4.4. CREATING A SYSTEM IMAGE WITH IMAGE BUILDER IN THE COMMAND-LINE INTERFACE

This procedure shows how to build a custom image using the Image Builder command-line interface.

Prerequisites

- You have a blueprint prepared for the image.

Procedure

1. Start the compose:

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE
```

Replace *BLUEPRINT-NAME* with name of the blueprint, and *IMAGE-TYPE* with the type of image. For possible values, see output of the **composer-cli compose types** command.

The compose process starts in the background and shows the composer Universally Unique Identifier (UUID).

2. Wait until the compose process is finished. The image creation can take up to ten minutes to complete.

To check the status of the compose:

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

A finished compose shows a status value **FINISHED**. Identify the compose in the list by its UUID.

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

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

Replace *UUID* with the UUID value shown in the previous steps.

Verification

After you create your image, you can check the image creation progress using the following commands:

- Check the compose status:

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

- Download the metadata of the image:

```
$ sudo composer-cli compose metadata UUID
```

- Download the logs of the image:

```
$ sudo composer-cli compose logs UUID
```

The command creates a **.tar** file that contains the logs for the image creation. If the logs are empty, you can check the journal.

- Check the journal:

```
$ journalctl | grep osbuild
```

- Check the manifest:

```
$ sudo cat /var/lib/osbuild-composer/jobs/job_UUID.json
```

You can find the *job_UUID.json* in the journal.

Additional resources

- [Tracing Image Builder](#)

4.5. BASIC IMAGE BUILDER COMMAND-LINE COMMANDS

The Image Builder command-line interface offers the following subcommands.

Blueprint manipulation

List all available blueprints

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

Show a blueprint contents in the TOML format

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

Save (export) blueprint contents in the TOML format into a file *BLUEPRINT-NAME.toml*

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

Remove a blueprint

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

Push (import) a blueprint file in the TOML format into Image Builder

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

Composing images from blueprints

List the available image types

```
# composer-cli compose types
```

Start a compose

```
# composer-cli compose start BLUEPRINT COMPOSE-TYPE
```

Replace *BLUEPRINT* with name of the blueprint to build and *COMPOSE-TYPE* with the output image type.

List all composes

```
# composer-cli compose list
```

List all composes and their status

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

Cancel a running compose

```
# composer-cli compose cancel COMPOSE-UUID
```

Delete a finished compose

```
# composer-cli compose delete COMPOSE-UUID
```

Show detailed information about a compose

```
# composer-cli compose info COMPOSE-UUID
```

Download image file of a compose

```
# composer-cli compose image COMPOSE-UUID
```

Additional resources

- The *composer-cli(1)* manual page provides a full list of the available subcommands and options:

```
$ man composer-cli
```

- The **composer-cli** command provides help on the subcommands and options:

```
# composer-cli help
```

4.6. IMAGE BUILDER BLUEPRINT FORMAT

Image Builder blueprints are presented to the user as plain text in the TOML format.

The elements of a typical blueprint file include:

The blueprint metadata

```
name = "BLUEPRINT-NAME"
description = "LONG FORM DESCRIPTION TEXT"
version = "VERSION"
```

Replace *BLUEPRINT-NAME* and *LONG FORM DESCRIPTION TEXT* with a name and description for your blueprint.

Replace *VERSION* with a version number according to the [Semantic Versioning](#) scheme.

This part is present only once for the whole blueprint file.

The entry *modules* describe the package names and matching version glob to be installed into the image.

The entry *group* describes a group of packages to be installed into the image. Groups categorize their packages in:

- Mandatory
 - Default
 - Optional
- Blueprints installs the mandatory packages. There is no mechanism for selecting optional packages.

Groups to include in the image

```
[[groups]]
name = "group-name"
```

Replace *group-name* with the name of the group, such as **anaconda-tools**, **widget**, **wheel** or **users**.

Packages to include in the image

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

Replace *package-name* with the name of the package, such as **httpd**, **gdb-doc**, or **coreutils**.

Replace *package-version* with a version to use. This field supports **dnf** version specifications:

- For a specific version, use the exact version number such as **8.30**.

- For latest available version, use the asterisk `*`.
- For the latest minor version, use format such as `8.*`.

Repeat this block for every package to include.

4.7. SUPPORTED IMAGE CUSTOMIZATIONS

A number of image customizations are supported within blueprints. To make use of these options, you must initially configure the customizations in the blueprint and import (push) it to Image Builder.



NOTE

These customizations are not currently supported within the web console.

Set the image hostname

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

User specifications for the resulting system 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
```



NOTE

The GID is optional and must already exist in the image, be created by a package, or be created by the blueprint `[[customizations.group]]` entry.



IMPORTANT

To generate the hash, you must install **python3** on your system. The following command installs the **python3** package.

```
# dnf install python3
```

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

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

Replace *PUBLIC-SSH-KEY* with the actual public key.

Replace the other placeholders with suitable values.

Leave out any of the lines as needed, only the user name is required.

Repeat this block for every user to include.

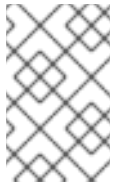
Group specifications for the resulting system image

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

Repeat this block for every group to include.

Set an existing users ssh key

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



NOTE

This option is only applicable for existing users. To create a user and set an ssh key, see the **User specifications for the resulting system image** customization in this section.

Append a kernel boot parameter option to the defaults

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

By default, Image Builder builds a default kernel into the image. But, you can customize the kernel with the following configuration in blueprint

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

Define a kernel name to be used in an image

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

Set the timezone and the *Network Time Protocol* (NTP) servers for the resulting system image

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

If you do not set a timezone, the system uses *Universal Time, Coordinated (UTC)* as default. Setting NTP servers is optional.

Set the locale settings for the resulting system image

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

Setting both language and keyboard options is mandatory. You can add multiple languages. The first language you add will be the primary language and the other languages will be secondary.

Set the firewall for the resulting system image

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

You can use the numeric ports, or their names from the `/etc/services` file to enable lists.

Customize the firewall services

Review the available firewall services.

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

In the blueprint, under section **customizations.firewall.service**, specify the firewall services that you want to customize.

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

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

You can optionally customize the firewall services for the system image that you plan to create.



NOTE

If you do not want to customize the firewall services, omit the **[customizations.firewall]** and **[customizations.firewall.services]** sections from the blueprint.

Set which services to enable during the boot time

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

You can control which services to enable during the boot time. Some image types already have services enabled or disabled so that the image works correctly and this setup cannot be overridden.

**NOTE**

Each time a build starts, it clones the repository. If you refer to a repository with a large amount of history, it might take a while to clone and use a significant amount of disk space. Also, the clone is temporary and is removed after the RPM package is created.

Specify a custom filesystem configuration

You can specify a custom filesystem configuration in your blueprints and thus create images with a specific disk layout, instead of using 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
- performance
- consistency with existing setups

Customize the filesystem configuration in your blueprint:

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

The following **mountpoints** and their sub-directories are supported:

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

**NOTE**

Customizing mountpoints is only supported from RHEL 8.5 and RHEL 9.0 distributions onward, using the CLI. In early 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, you can create images with a customized file system partition on LVM and resize those partitions at runtime. For that, you can specify a customized filesystem configuration in your blueprint and then create images with the desired disk layout. The default filesystem layout remains unchanged - if you use plain images without file system customization, the root partition is resized by **cloud-init**.

After you add a file system customization to your blueprint, the file system is converted to a LVM partition.

When you define the *MINIMUM-PARTITION-SIZE*, there is no default size format. The following values and units are supported: kB to TB and KiB to TiB. For example, you can define the mountpoint size in bytes:

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

You can also define the mountpoint size using units.



NOTE

You can only define the mountpoint size using units for package version provided for RHEL 8.6 and RHEL 9.0 distributions onward.

For example:

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

Additional resources

- [Blueprint import fails after adding filesystem customization "size" .](#)

4.8. INSTALLED PACKAGES

When you create a system image using Image Builder, by default, the system installs a set of base packages. The base list of packages are the members of the **comps core** group. By default, Image Builder uses the **core dnf** group.

Table 4.1. 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

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

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, mada-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
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, fdisk, gzip, ima-evm-utils, iproute, iptables, iputils, iscsi-initiator-utils, keyutils, lldpad, lvm2, passwd, policycoreutils, policycoreutils-python-utils, procs-ng, rootfiles, setools-console, sudo, traceroute, util-linux

Image type	Default Packages
image-installer	anaconda-dracut, curl, dracut-config-generic, dracut-network, hostname, iwl100-firmware, iwl1000-firmware, iwl105-firmware, iwl135-firmware, iwl2000-firmware, iwl2030-firmware, iwl3160-firmware, iwl5000-firmware, iwl5150-firmware, iwl6000-firmware, iwl6050-firmware, iwl7260-firmware, kernel, less, nfs-utils, openssh-clients, ostree, plymouth, prefixdevname, rng-tools, rpcbind, selinux-policy-targeted, systemd, tar, xfsprogs, xz
edge-raw-image	dnf, dosfstools, e2fsprogs, glibc, lorax-templates-generic, lorax-templates-rhel, lvm2, policycoreutils, python36, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz

**NOTE**

When you add additional components to your blueprint, you must make sure that the packages in the components you added do not conflict with any other package components, otherwise the system fails to solve dependencies. As a consequence, you are not able to create your customized image.

Additional resources

- [Image Builder description](#)

4.9. ENABLED SERVICES

When you configure the custom image, the services enabled are the defaults services for the RHEL release you are running **osbuild-composer** from, additionally the services enabled for specific image types.

For example, the **.ami** image type enables the services **sshd**, **chronyd** and **cloud-init** and without these services, the custom image does not boot.

Table 4.2. Enabled services to support image type creation

Image type	Enabled Services
ami	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
openstack	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
qcow2	cloud-init
rhel-edge-commit	No extra service enables by default

Image type	Enabled Services
tar	No extra service enables by default
vhd	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk	sshd, chronyd, vmtoolsd, cloud-init

Note: You can customize which services to enable during the system boot. However, for image types with services enabled by default, the customization does not override this feature.

Additional resources

- [Supported Image Customizations](#)

CHAPTER 5. CREATING SYSTEM IMAGES WITH IMAGE BUILDER WEB CONSOLE INTERFACE

Image Builder is a tool for creating custom system images. To control Image Builder and create your custom system images, you can use the web console interface. Note that the [command-line interface](#) is the currently preferred alternative, because it offers more features.

5.1. ACCESSING IMAGE BUILDER GUI IN THE RHEL WEB CONSOLE

The **cockpit-composer** plugin for the RHEL web console enables users to manage Image Builder blueprints and composes with a graphical interface. Note that the preferred method for controlling Image Builder is at the moment using the command-line interface.

Prerequisites

- You must have root access to the system.

Procedure

1. Open <https://localhost:9090/> in a web browser on the system where Image Builder is installed. For more information on how to remotely access Image Builder, see [Managing systems using the RHEL web console](#) document.
2. Log into the web console with credentials for a user account with sufficient privileges on the system.
3. To display the Image Builder controls, click the **Image Builder** icon, which is in the upper-left corner of the window.
The Image Builder view opens, listing existing blueprints.

Additional resources

- [Creating system images with Image Builder command-line interface](#)

5.2. CREATING AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE

To describe the customized system image, create a blueprint first.

Prerequisites

- You have opened the Image Builder interface of the RHEL web console in a browser.

Procedure

1. Click **Create Blueprint** in the top right corner.
A pop-up appears with fields for the blueprint name and description.
2. Fill in the name of the blueprint, its description, then click **Create**.
The screen changes to blueprint editing mode.
3. Add components that you want to include in the system image:

On the left, click the **Available Components** tab. On the right, click the **Available Components** tab.

- a. On the left, enter all or part of the component name in the **Available Components** field and press **Enter**.

The search is added to the list of filters under the text entry field, and the list of components below is reduced to those that match the search.

If the list of components is too long, add further search terms in the same way.

- b. The list of components is paged. To move to other result pages, use the arrows and entry field above the component list.
- c. Click the name of the component you intend to use to display its details. The right pane fills with details of the components, such as its version and dependencies.
- d. Select the version you want to use in the **Component Options** box, with the **Version Release** dropdown.
- e. Click **Add** in the top left.
- f. If you added a component by mistake, remove it by clicking the ... button at the far right of its entry in the right pane, and select **Remove** in the menu.



NOTE

If you do not intend to select a version for some components, you can skip the component details screen and version selection by clicking the **+** buttons on the right side of the component list.

4. To save the blueprint, click **Commit** in the top right. A dialog with a summary of the changes pops up. Click **Commit**.
A small pop-up on the right informs you of the saving progress and then the result.
5. To exit the editing screen, click **Back to Blueprints** in the top left.
The Image Builder view opens, listing existing blueprints.

5.3. EDITING AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE

To change the specifications for a custom system image, edit the corresponding blueprint.

Prerequisites

- You have opened the Image Builder interface of the RHEL web console in a browser.
- A blueprint exists.

Procedure

1. Locate the blueprint that you want to edit by entering its name or a part of it into the search box at top left, and press **Enter**.
The search is added to the list of filters under the text entry field, and the list of blueprints below is reduced to those that match the search.

If the list of blueprints is too long, add further search terms in the same way.

2. On the right side of the blueprint, press the **Edit Blueprint** button that belongs to the blueprint.

The view changes to the blueprint editing screen.

3. Remove unwanted components by clicking their  button at the far right of its entry in the right pane, and select **Remove** in the menu.
4. Change version of existing components:
 - a. On the Blueprint Components search field, enter component name or a part of it into the field under the heading **Blueprint Components** and press **Enter**.
The search is added to the list of filters under the text entry field, and the list of components below is reduced to those that match the search.

If the list of components is too long, add further search terms in the same way.
 - b. Click the  button at the far right of the component entry, and select **View** in the menu.
A component details screen opens in the right pane.
 - c. Select the desired version in the **Version Release** drop-down menu and click **Apply Change** in top right.
The change is saved and the right pane returns to listing the blueprint components.

5. Add new components:

- a. On the left, enter component name or a part of it into the field under the heading **Available Components** and press **Enter**.
The search is added to the list of filters under the text entry field, and the list of components below is reduced to those that match the search.

If the list of components is too long, add further search terms in the same way.
- b. The list of components is paged. To move to other result pages, use the arrows and entry field above the component list.
- c. Click the name of the component you intend to use to display its details. The right pane fills with details of the components, such as its version and dependencies.
- d. Select the version you want to use in the **Component Options** box, with the **Version Release** drop-down menu.
- e. Click **Add** in the top right.
- f. If you added a component by mistake, remove it by clicking the  button at the far right of its entry in the right pane, and select **Remove** in the menu.



NOTE

If you do not intend to select a version for some components, you can skip the component details screen and version selection by clicking the **+** buttons on the right side of the component list.

6. Commit a new version of the blueprint with your changes:
 - a. Click the **Commit** button in top right.
A pop-up window with a summary of your changes appears.
 - b. Review your changes and confirm them by clicking **Commit**.

A small pop-up on the right informs you of the saving progress and the results. A new version of the blueprint is created.

- c. In the top left, click **Back to Blueprints** to exit the editing screen.
The Image Builder view opens, listing existing blueprints.

5.4. ADDING USERS AND GROUPS TO AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE

Adding customizations such as users and groups to blueprints in the web console interface is currently not possible. To work around this limitation, use the **Terminal** tab in web console to use the command-line interface (CLI) workflow.

Prerequisites

- A blueprint must exist.
- A CLI text editor such as **vim**, **nano**, or **emacs** must be installed. To install them:

```
# dnf install editor-name
```

Procedure

1. Find out the name of the blueprint: Open the Image Builder (**Image builder**) tab on the left in the RHEL web console to see the name of the blueprint.
2. Navigate to the CLI in web console: Open the system administration tab on the left, then select the last item **Terminal** from the list on the left.
3. Enter the super-user (root) mode:

```
$ sudo bash
```

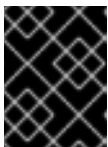
Provide your credentials when asked. Note that the terminal does not reuse your credentials you entered when logging into the web console.

A new shell with root privileges starts in your home directory.

4. Export the blueprint to a file:

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

5. Edit the file *BLUEPRINT-NAME.toml* with a text editor and add the users and groups.



IMPORTANT

RHEL web console does not have any built-in feature to edit text files on the system, so the use of a CLI text editor is required for this step.

- a. For every user to be added, add this block to the file:

```
[[customizations.user]]
name = "USER-NAME"
```

```
description = "USER-DESCRIPTION"
password = "PASSWORD-HASH"
key = "ssh-rsa (...) key-name"
home = "/home/USER-NAME/"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = NUMBER
gid = NUMBER
```

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

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

Replace *ssh-rsa (...) key-name* with the actual public key.

Replace the other placeholders with suitable values.

Leave out any of the lines as needed, only the user name is required.

- b. For every user group to be added, add this block to the file:

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

- c. Increase the version number.
- d. Save the file and close the text editor.

6. Import the blueprint back into Image Builder:

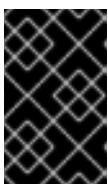
```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

Note that you must supply the file name including the **.toml** extension, while in other commands you use only the name of the blueprint.

7. To verify that the contents uploaded to Image Builder match your edits, list the contents of blueprint:

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

Check if the version matches what you put in the file and if your customizations are present.



IMPORTANT

The Image Builder plugin for RHEL web console does not show any information that could be used to verify that the changes have been applied, unless you also edited the packages included in the blueprint.

8. Exit the privileged shell:

```
# exit
```

9. Open the Image Builder (**Image builder**) tab on the left and refresh the page, in all browsers and all tabs where it was opened.

This prevents state cached in the loaded page from accidentally reverting your changes.

Additional resources

- [Image Builder blueprint format](#)
- [Editing an Image Builder blueprint with command-line interface](#)

5.5. CREATING A SYSTEM IMAGE WITH IMAGE BUILDER IN THE WEB CONSOLE INTERFACE

The following steps below describe creating a system image.

Prerequisites

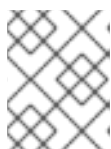
- You have opened the Image Builder interface of the RHEL web console in a browser.
- A blueprint exists.

Procedure

1. Locate the blueprint that you want to build an image by entering its name or a part of it into the search box at top left, and press **Enter**.
The search is added to the list of filters under the text entry field, and the list of blueprints below is reduced to those that match the search.

If the list of blueprints is too long, add further search terms in the same way.

2. On the right side of the blueprint, press the **Create Image** button that belongs to the blueprint. A pop-up window appears.
3. Select the image type and press **Create**.
A small pop-up in the top right informs you that the image creation has been added to the queue.
4. Click the name of the blueprint.
A screen with details of the blueprint opens.
5. Click the **Images** tab to switch to it. The image that is being created is listed with the status **In Progress**.



NOTE

Image creation takes a longer time, measured in minutes. There is no indication of progress while the image is created.

To abort image creation, press its **Stop** button on the right.

6. After the image is successfully created, the **Stop** button is replaced by a **Download** button. Click this button to download the image to your system.

5.6. ADDING A SOURCE TO A BLUEPRINT

The sources defined in Image Builder provide the contents that you can add to blueprints. These sources are global and therefore available to all blueprints. The System sources are repositories that are set up locally on your computer and cannot be removed from Image Builder. You can add additional custom sources and thus be able to access other contents than the System sources available on your system.

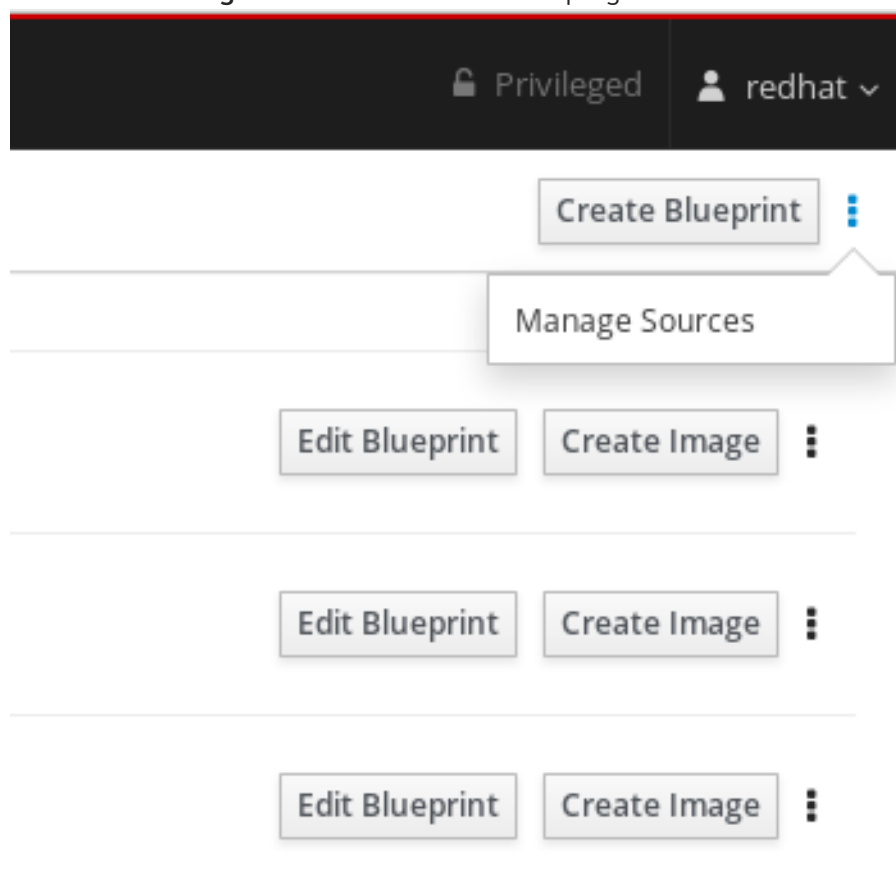
The following steps describe how to add a Source to your local system.

Prerequisites

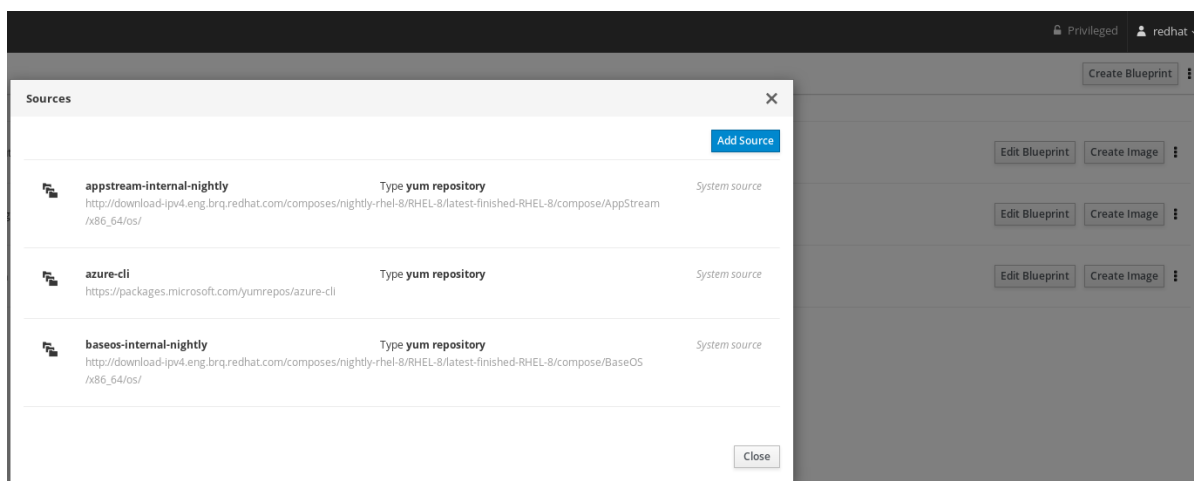
- You have opened the Image Builder interface of the RHEL web console in a browser.

Procedure

1. Click the **Manage Sources** button in the top right corner.



A pop-up window appears with the available sources, their names and descriptions.



2. On the right side of the pop-up window, click the **Add Source** button.
3. Add the desired **Source name**, the **Source path**, and the **Source Type**. The **Security** field is optional.

4. Click **Add Source** button. The screen shows the available sources window and lists the source you have added.

As a result, the new System source is available and ready to be used or edited.

5.7. CREATING A USER ACCOUNT FOR A BLUEPRINT

The images created by Image Builder have the root account locked and no other accounts included. Such configuration is provided in order to ensure that you cannot accidentally build and deploy an image without a password. Image Builder enables you to create a user account with password for a blueprint so that you can log in to the image created from the blueprint.

Prerequisites

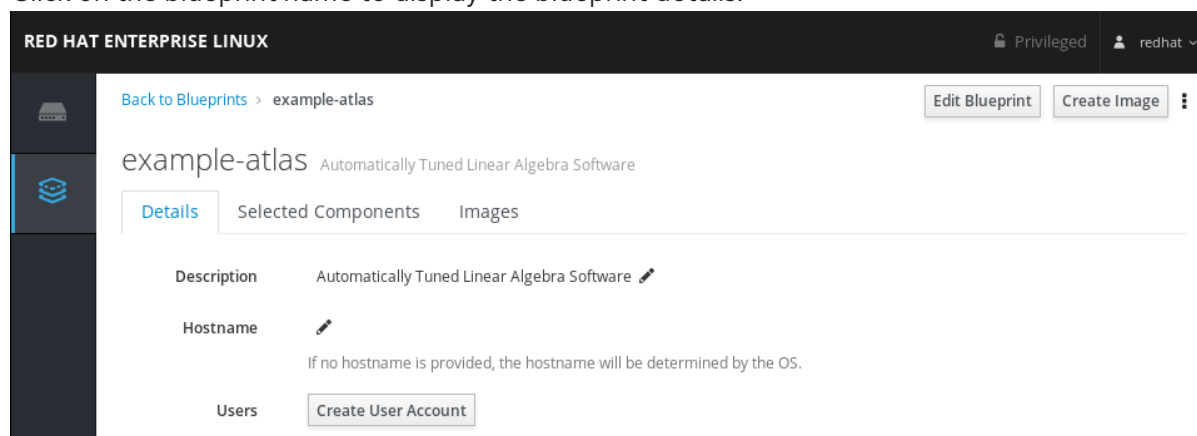
- You have opened the Image Builder interface of the RHEL web console in a browser.
- You have an existing blueprint.

Procedure

1. Locate the blueprint that you want to create a user account for by entering its name or a part of it into the search box at the top left, and press **Enter**.

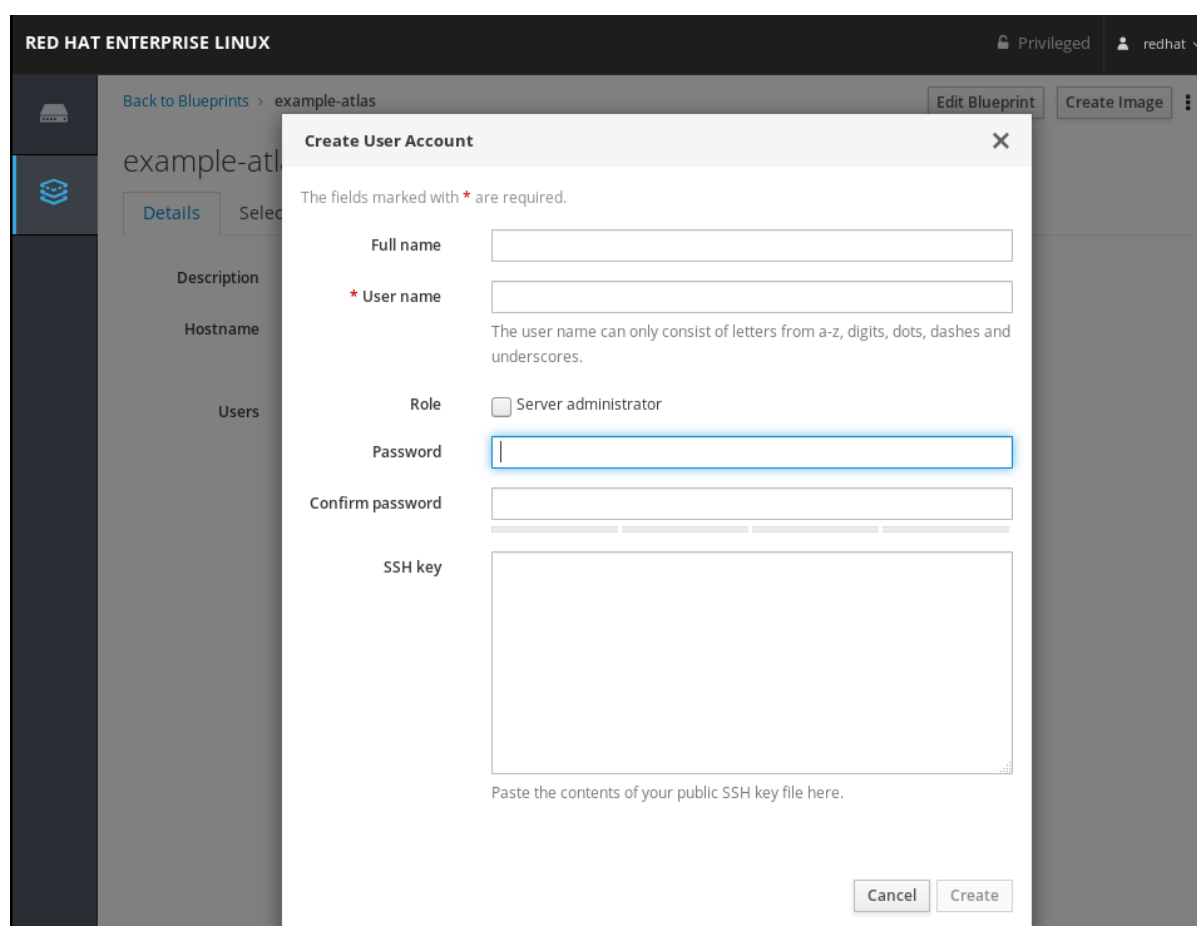
The search is added to the list of filters under the text entry field, and the list of blueprints below is reduced to those that match the search.

- Click on the blueprint name to display the blueprint details.

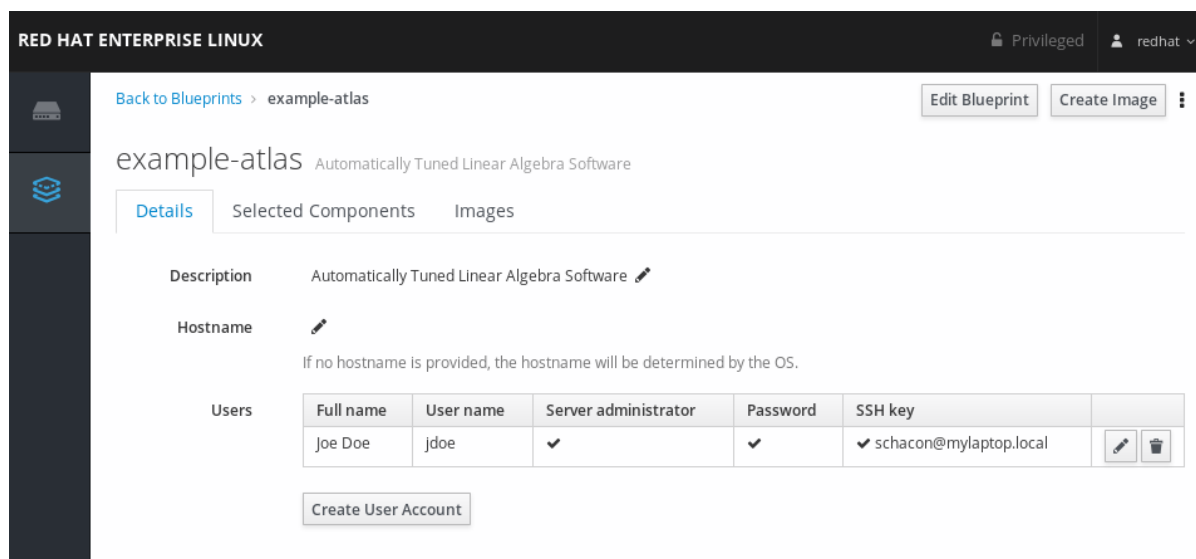


- Click **Create User Account**.

This will open a window with fields for user account creation.



- Fill in the details. Notice that when you insert the name, the **User name** field autocompletes, suggesting a username.
- After you have inserted all the desired details, click **Create**.
- The created user account appears showing all the information you have inserted.



7. To create further user accounts for the blueprint, repeat the process.

5.8. CREATING A USER ACCOUNT WITH SSH KEY

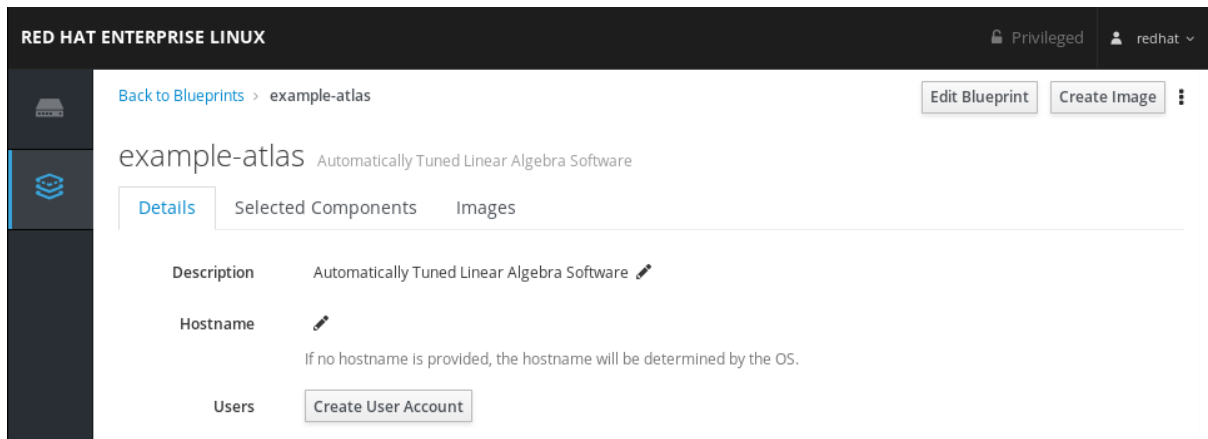
The images created by Image Builder have the root account locked and no other accounts included. Such configuration is provided in order to ensure that images are secure, by not having a default password. Image Builder enables you to create a user account with SSH key for a blueprint so that you can authenticate to the image that you created from the blueprint. To do so, first, create a blueprint. Then, you will create a user account with a password and an SSH key. The following example shows how to create a Server administrator user with an SSH key configured.

Prerequisites

- You have created an SSH key that will be paired with the created user later on in the process.
- You have opened the Image Builder interface of the RHEL web console in a browser.
- You have an existing blueprint

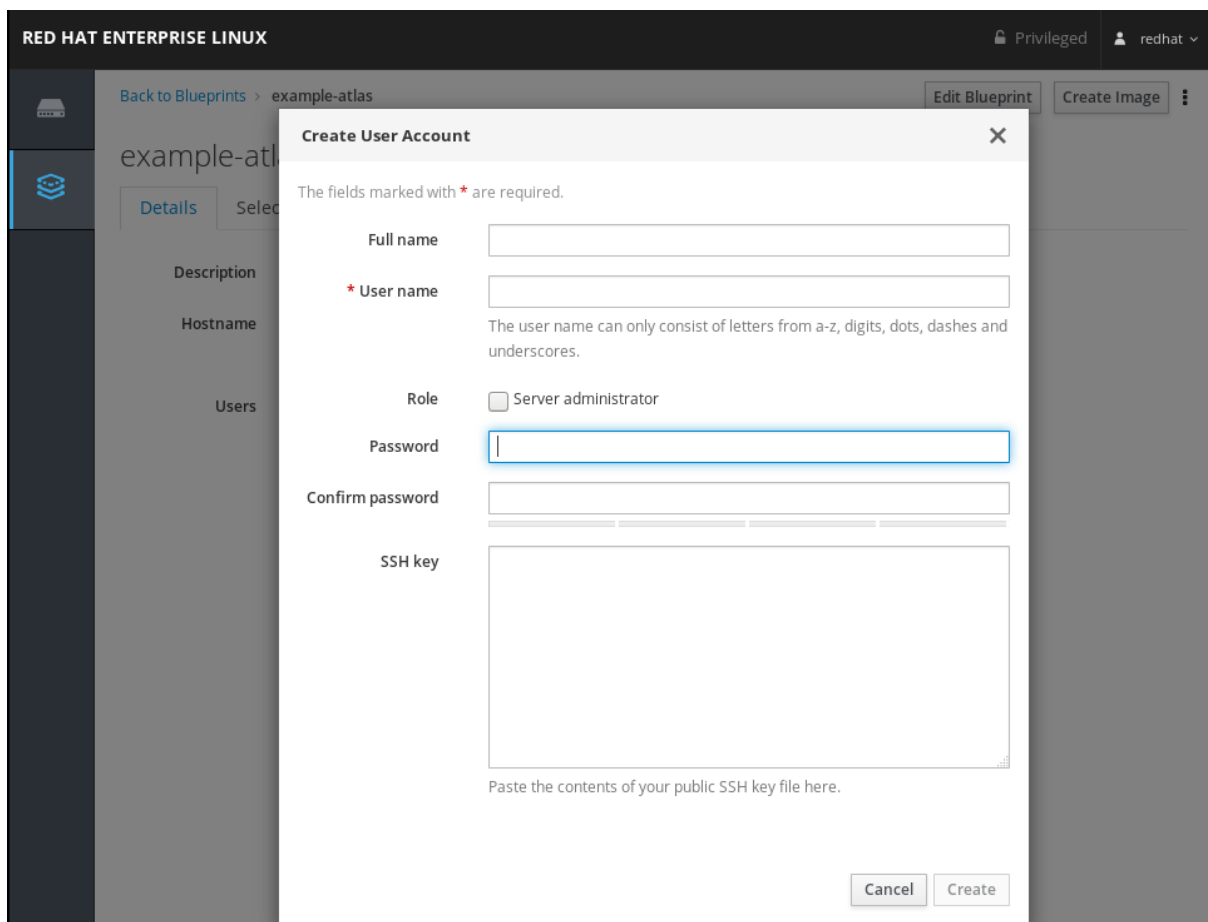
Procedure

1. Locate the blueprint that you want to create a user account for by entering its name or a part of it into the search box at the top left, and press **Enter**.
The search is added to the list of filters under the text entry field, and the list of blueprints below is reduced to those that match the search.
2. Click on the blueprint name to display the blueprint details.

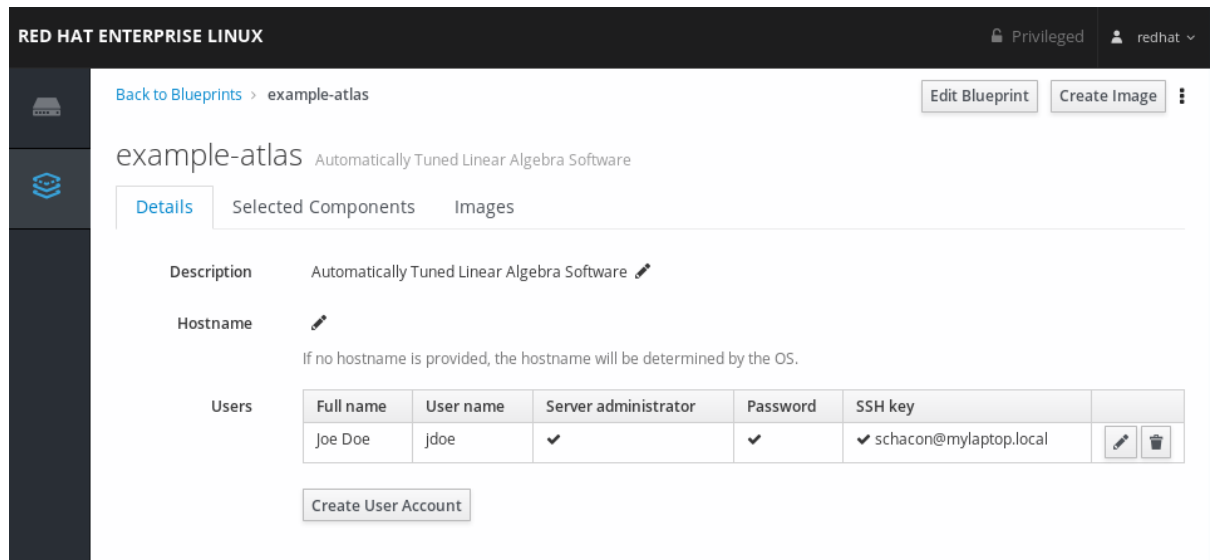


3. Click **Create User Account**.

This will open a window with fields for user account creation



4. Fill in the details. Notice that when you insert the name, the **User name** field autocompletes, suggesting a username.
If you want to provide administrators rights to the user account you are creating, check the **Role** field.
- Paste the content of your public SSH key file.
5. After you have inserted all the desired details, click **Create**.
6. The new user account will appear in the user list, showing all the information you have inserted.



7. If you want to create more user accounts for the blueprint, repeat the process.

Additional resources

- [Generating SSH key pairs](#)

CHAPTER 6. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER

You can use Image Builder to create bootable ISO Installer images. These images consist of a tarball that contains a root file system. You can use the bootable ISO image to install the file system to a bare metal server.

Image Builder builds a manifest that creates a boot ISO that contains the commit and a root file system. To create the ISO image, choose the new image type **image-installer**. Image Builder builds a **.tar** file, which contains:

- a standard Anaconda installer ISO
- an embedded RHEL system tarball
- a default kickstart file that installs the commit with minimal default requirements

The created installer ISO image embeds a pre-configured system image that you can install directly to a bare metal server.

6.1. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER IN THE COMMAND-LINE INTERFACE

This procedure shows how to build a custom boot ISO installer image using the Image Builder command-line interface.

Prerequisites

- You created a blueprint for the image with a user included and pushed it back into Image Builder. See [Blueprint customization for users](#).

Procedure

1. Create the ISO image:

```
# composer-cli compose start BLUEPRINT-NAME image-installer
```

- *BLUEPRINT-NAME* with name of the blueprint you created
 - *image-installer* is the image type
- The compose process starts in the background and the UUID of the compose is shown.

2. Wait until the compose is finished. Note that this may take several minutes.
To check the status of the compose:

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

A finished compose shows a status value of **FINISHED**. Identify the compose in the list by its UUID.

3. After the compose is finished, download the resulting image file:

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

Replace *UUID* with the UUID value shown in the previous steps.

As a result, Image Builder builds a **.tar** file that contains the ISO Installer image.

Verification

1. Navigate to the folder where you downloaded the image file.
2. Locate the **.tar** image you downloaded.
3. Extract the **.tar** content.

You can use the resulting ISO image file on a hard drive or to boot in a virtual machine, for example, in an HTTP Boot or a USB installation.

Additional resources

- [Creating system images with Image Builder command-line interface](#)
- [Creating a bootable installation medium for RHEL](#)

6.2. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER IN THE GUI

You can build a customized boot ISO installer image using Image Builder GUI.

Prerequisites

- You created a blueprint for your image.

Procedure

1. Open the Image Builder interface of the RHEL web console in a browser.
2. Locate the blueprint that you want to build your image by entering its name or a part of it into the search box at top left, and click **Enter**.
3. On the right side of the blueprint, click the **Create Image** button that belongs to the blueprint. The **Create image** dialog wizard opens.
4. On the **Create image** dialog wizard, from the **Image Type** list:
 - a. Select the **"RHEL Installer (.iso)"** image type.
 - b. Click **Create**.

Image Builder adds the compose of a RHEL **.iso** image to the queue.



NOTE

The image build process take a few minutes to complete.

After the process is complete, you can see the Image build complete status. Image Builder creates the **.iso** image.

Verification

After the image is successfully created, you can download your image button.

1. Click **Download** to save the "RHEL Installer (.iso)" image to your system.
2. Navigate to the folder where you downloaded the "**RHEL Installer (.iso)**" image.
3. Locate the .tar image you downloaded.
4. Extract the "**RHEL Installer (.iso)**" image content.

You can use the resulting ISO image file on a hard drive or to boot in a virtual machine, for example, in an HTTP Boot or a USB installation.

Additional resources

- [Creating an Image Builder blueprint in the web console interface](#) .
- [Creating system images with Image Builder command-line interface](#) .
- [Creating a bootable installation medium for RHEL](#) .

6.3. INSTALLING THE ISO IMAGE TO A BARE METAL SYSTEM

This procedure shows how to install the bootable ISO image you created by using Image Builder to a bare metal system, using the command-line interface.

Prerequisites

- You created the bootable ISO image using Image Builder.
- You have downloaded and extracted the bootable ISO image.
- You have a 8 GB USB flash drive.



NOTE

The ISO size can be bigger depending on the packages that you selected in your blueprint.

Procedure

1. Place the bootable ISO image file on 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.
4. Perform the steps to install the customized bootable ISO image.
The boot screen shows you the following options:
 - Install Red Hat Enterprise Linux 9
 - Test this media & install Red Hat Enterprise Linux 9

Additional resources

- [Booting the installation](#)

CHAPTER 7. PREPARING AND DEPLOYING KVM GUEST IMAGES WITH IMAGE BUILDER

Customers can manually create images from an ISO or have a purpose-built image created using Image Builder. This procedure describes steps to create a purpose-built image using Image Builder. This is limited to **rhel-guest-image** support to Red Hat Virtualization (RHV).

Creating a customized KVM guest image following involves the following high-level steps:

1. Creating a KVM guest Image **.qcow2** image using Image Builder.
2. Creating a virtual machine from the KVM guest image.

7.1. CREATING CUSTOMIZED KVM GUEST IMAGES WITH IMAGE BUILDER

This describes steps to create a **.qcow2** KVM guest image using Image Builder.

Prerequisites

- You must have root or wheel group user access to the system.
- The **cockpit-composer** package is installed.
- On a RHEL system, you have opened the Image Builder dashboard of Cockpit UI.

Procedure

1. Click **Create blueprint** to create a blueprint. See [Creating an Image Builder blueprint in the web console interface](#).
2. Select the components and packages that you want as part of the KVM guest image you are creating.
3. Click **Commit** to commit the changes you made to the blueprint. A small pop-up on the superior right side informs you of the saving progress and then the result of the changes you committed.
4. Click the blueprint name link on the left banner.
5. Select the tab **Images**.
6. Click **Create Image** to create your customized image. A pop-up window opens.
7. From the **Type** drop-down menu list, select the **`QEMU Image(.qcow2)`** image.
8. Set the size that you want the image to be when instantiated and click **Create**.
9. A small pop-up on the upper right side of the window informs you that the image creation has been added to the queue. After the image creation process is complete, you can see the **Image build complete** status.

Verification steps

1. Click the breadcrumbs icon and select the **Download** option. Image Builder downloads the KVM guest image **.qcow2** file at your default download location.

Additional resources

- [Creating an Image Builder blueprint in the web console interface](#) .

7.2. CREATING A VIRTUAL MACHINE FROM A KVM GUEST IMAGE

With Image Builder, you can build a **.qcow2** image, and use a KVM guest image to create a VM. The KVM guest images created using Image Builder already have **cloud-init** installed and enabled.

Prerequisites

- You created a **.qcow2** image using Image Builder. See [Creating an Image Builder blueprint in the web console interface](#).
- The **qemu-kvm** package is installed on your system. You can check if the **/dev/kvm** folder is available on your system.
- You have **libvirt** installed on your system.
- You have **virt-install** installed on your system.
- The **genisoimage** utility is installed on your system.

Procedure

1. Move the KVM Guest Image you created using Image Builder to the **/var/lib/libvirt/images** directory and rename the image name to **rhel-8.6-x86_64-kvm.qcow2**.
2. Create a directory, for example, **cloudinitiso** and navigate to this newly created directory:

```
$ mkdir cloudinitiso
$ cd cloudinitiso
```

3. Create a file named **meta-data**. Add the following information to this file:

```
instance-id: citest
local-hostname: vmname
```

4. Create a file named **user-data**. Add the following information to the file:

```
#cloud-config
user: admin
password: password
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
- ssh-rsa AAA...fhHQ== your.email@example.com
```

Where,

- **ssh_authorized_keys** is your SSH public key. You can find your SSH public key in **~/.ssh/id_rsa.pub**.
5. Use the **genisoimage** command to create an ISO image that includes the **user-data** and **meta-data** files.

```
# genisoimage -output cloud-init.iso -volid cidata -joliet -rock user-data meta-data
```

```
l: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 331
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
183 extents written (0 MB)
```

6. Create a new VM from the KVM Guest Image using the **virt-install** command. Include the ISO image you created on step 4 as an attachment to the VM image.

```
# virt-install \
  --memory 4096 \
  --vcpus 4 \
  --name myvm \
  --disk rhel-8.6-x86_64-kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
  --disk cloud-init.iso,device=cdrom \
  --os-variant rhel8.6 \
  --virt-type kvm \
  --graphics none \
  --import
```

Where,

- `--graphics none` - means it is a headless RHEL 8.4 VM.
- `--vcpus 4` - means that it uses 4 virtual CPUs.
- `--memory 4096` - means it uses 4096 MB RAM.

7. The VM installation starts:

```
Starting install...
Connected to domain mytestcivm
...
[ OK ] Started Execute cloud user/final scripts.
[ OK ] Reached target Cloud-init target.

Red Hat Enterprise Linux 8.6 (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

Verification

After the boot is complete, the VM shows a text login interface. To log in to the VM:

1. Enter **admin** as a username and press **Enter**.
2. Enter **password** as password and press **Enter**.
After the login authentication is complete, you have access to the VM using the CLI.

Additional resources

- [Enabling virtualization](#).

- [Configuring and managing cloud-init for RHEL 9](#) .
- [cloud-init significant directories and files](#).

CHAPTER 8. PREPARING AND UPLOADING CLOUD IMAGES WITH IMAGE BUILDER

Image Builder can create custom system images ready for use in clouds of various providers. To use your customized RHEL system image in a cloud, create the system image with Image Builder using the respective output type, configure your system for uploading the image, and upload the image to your cloud account. From Red Hat Enterprise Linux 8.3, the ability to push customized image clouds through the **Image Builder** application in the RHEL web console is available for a subset of the service providers that we support, such as **AWS** and **Azure** clouds. See [Pushing images to AWS Cloud AMI](#) and [Pushing VHD images to Azure cloud](#).

8.1. PREPARING FOR UPLOADING AWS AMI IMAGES

This describes steps to configure a system for uploading AWS AMI images.

Prerequisites

- You must have an Access Key ID configured in the [AWS IAM account manager](#).
- You must have a writable [S3 bucket](#) prepared.

Procedure

1. Install Python 3 and the **pip** tool:

```
# dnf install python3
# dnf install python3-pip
```

2. Install the [AWS command-line tools](#) with **pip**:

```
# pip3 install awscli
```

3. Run the following command to set your profile. The terminal prompts you to provide your credentials, region and output format:

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

4. Define a name for your bucket and use the following command to create a bucket:

```
$ BUCKET=bucketname
$ aws s3 mb s3://$BUCKET
```

Replace *bucketname* with the actual bucket name. It must be a globally unique name. As a result, your bucket is created.

5. Then, to grant permission to access the S3 bucket, create a **vmimport** S3 Role in IAM, if you have not already done so in the past:

```
$ printf '{"Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service":
```

```
"vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{
"sts:Externalid": "vmimport" } } } }' > trust-policy.json
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [
"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource": [ "arn:aws:s3:::%s",
"arn:aws:s3:::%s/*" ] }, { "Effect": "Allow", "Action": [ "ec2:ModifySnapshotAttribute",
"ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource": "*" } ] }' $BUCKET
$BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

Additional resources

- [Using high-level \(s3\) commands with the AWS CLI](#)

8.2. UPLOADING AN AMI IMAGE TO AWS IN THE CLI

You can use Image Builder to build **.ami** images and push them directly to Amazon AWS Cloud service provider 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 a defined blueprint.

Procedure

1. Using the text editor, create a configuration file with the following content:

```
provider = "aws"

[settings]
accessKeyId = "AWS_ACCESS_KEY_ID"
secretAccessKey = "AWS_SECRET_ACCESS_KEY"
bucket = "AWS_BUCKET"
region = "AWS_REGION"
key = "IMAGE_KEY"
```

Replace values in the fields with your credentials for **accessKeyId**, **secretAccessKey**, **bucket**, **region**. The **IMAGE_KEY** value is the name of your VM Image to be uploaded to EC2.

2. Save the file as *CONFIGURATION-FILE.toml* and close the text editor.
3. Start the compose:

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE IMAGE_KEY
CONFIGURATION-FILE.toml
```

Replace:

- *BLUEPRINT-NAME* with the name of the blueprint you created
- *IMAGE-TYPE* with the **ami** image type.
- *IMAGE_KEY* with the name of your VM Image to be uploaded to EC2.
- *CONFIGURATION-FILE.toml* with the name of the configuration file of the cloud provider.



NOTE

You must have the correct IAM settings for the bucket you are going to send your customized image to. You have to set up a policy to your bucket before you are able to upload images to it.

4. Check the status of the image build and upload it to AWS:

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

After the image upload process is complete, you can see the "FINISHED" status.

Verification

To confirm that the image upload was successful:

1. Access [EC2](#) on the menu and choose 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**.

Additional Resources

- [Required service role to import a VM](#)

8.3. PUSHING IMAGES TO AWS CLOUD AMI

The ability to push the output image that you create to **AWS Cloud AMI** is available this time. This describes steps to push **.ami** images you create using Image Builder to Amazon AWS Cloud service provider.

Prerequisites

- You must have **root** or **wheel** group user access to the system.
- You have opened the Image Builder interface of the RHEL web console in a browser.
- You must have an Access Key ID configured in the [AWS IAM](#) account manager.
- You must have a writable [S3 bucket](#) prepared.

Procedure

1. Click **Create blueprint** to create a blueprint. See [Creating an Image Builder blueprint in the web console interface](#).
2. Select the components and packages that you want as part of the image you are creating.

3. Click **Commit** to commit the changes you made to the blueprint.
A small pop-up on the superior right side informs you of the saving progress and then the result of the changes you committed.
4. Click **blueprint name** link on the left banner.
5. Select the tab **Images**.
6. Click **Create Image** to create your customized image.
A pop-up window opens.
 - a. From the **"Type"** drop-down menu list, select the **"Amazon Machine Image Disk (.ami)"** image.
 - b. Check the **"Upload to AWS"** check box to upload your image to the AWS Cloud and click **Next**.
 - c. To authenticate your access to AWS, type your "AWS access key ID" and "AWS secret access key" in the corresponding fields. Click **Next**.

**NOTE**

You can view your AWS secret access key only when you create a new Access Key ID. If you do not know your Secret Key, generate a new Access Key ID.

- d. Type the name of the image in the "Image name" field, type the Amazon bucket name in the "Amazon S3 bucket name" field and type the "AWS region" field for the bucket you are going to add your customized image to. Click **Next**.
- e. Review the information and click **Finish**.
Optionally, you can click **Back** to modify any incorrect detail.

**NOTE**

You must have the correct IAM settings for the bucket you are going to send your customized image. We are using the IAM Import and Export, so you have to set up a **policy** to your bucket before you are able to upload images to it. For more information, see [Required Permissions for IAM Users](#).

7. A small pop-up on the superior right side informs you of the saving progress. It also informs that the image creation has been initiated, the progress of this image creation and the subsequent upload to the AWS Cloud.
After the process is complete, you can see the **"Image build complete"** status.
8. Click [Service→EC2](#) on the menu and choose the [correct region](#) in the AWS console. The image must have the "Available" status, to indicate that it is uploaded.
9. On the dashboard, select your image and click **Launch**.
10. A new window opens. Choose an instance type according to the resources you need to launch your image. Click **Review and Launch**.
11. Review your instance launch details. You can edit each section if you need to make any change. Click **Launch**.

12. Before you launch the instance, you must select a public key to access it.
You can either use the key pair you already have or you can create a new key pair. Alternatively, you can use **Image Builder** to add a user to the image with a preset public key. See [Creating a user account with SSH key](#) for more details.

Follow the next steps to create a new key pair in EC2 and attach it to the new instance.

- a. From the drop-down menu list, select "**Create a new key pair**".
 - b. Enter the name to the new key pair. It generates a new key pair.
 - c. Click "**Download Key Pair**" to save the new key pair on your local system.
13. Then, you can click **Launch Instance** to launch your instance.
You can check the status of the instance, it shows as "**Initializing**".
 14. After the instance status is "**running**", the **Connect** button becomes available.
 15. Click **Connect**. A popup window appears with instructions on how to connect using SSH.
 - a. Select the preferred connection method to "**A standalone SSH client**" and open a terminal.
 - b. In the location you store your private key, make sure that your key is publicly viewable for SSH to work. To do so, run the command:

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

- c. Connect to your instance using its Public DNS:

```
$ ssh -i "<_your-instance-name.pem_"> ec2-user@<_your-instance-IP-address_>
```

- d. Type "yes" to confirm that you want to continue connecting.
As a result, you are connected to your instance using SSH.

Verification steps

1. Check if you are able to perform any action while connected to your instance using SSH.

Additional resources

- [Open a case on Red Hat Customer Portal](#)
- [Connecting to your Linux instance using SSH](#)
- [Open support cases](#)

8.4. PREPARING FOR UPLOADING AZURE VHD IMAGES

This describes steps to upload an VHD image to Azure.

Prerequisites

- You must have a usable Azure resource group and storage account.

Procedure

1. Install python2:

```
# dnf install python2
```



NOTE

python2 package must be installed because since the AZ CLI depends specifically on python 2.7

2. Import the Microsoft repository key:

```
# rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

3. Create a local azure-cli repository information:

```
# sh -c 'echo -e "[azure-cli]\nname=Azure\ncli\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\nngpgcheck=1\nngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/azure-cli.repo'
```

4. Install the Azure CLI:

```
# dnfdownload azure-cli
# rpm -ivh --nodeps azure-cli-2.0.64-1.el7.x86_64.rpm
```



NOTE

The downloaded version of the Azure CLI package may vary depending on the current downloaded version.

5. Run the Azure CLI:

```
$ az login
```

The terminal shows the message 'Note, we have launched a browser for you to login. For old experience with device code, use "az login --use-device-code"' and open a browser where you can login.



NOTE

If you are running a remote (SSH) session, the link will not open in the browser. In this case, you can use the link provided and thus be able to login and authenticate your remote session. To sign in, use a web browser to open the page <https://microsoft.com/devicelogin> and enter the code XXXXXXXXXX to authenticate.

6. List the keys for the storage account in Azure:

```
$ GROUP=resource-group-name
$ ACCOUNT=storage-account-name
$ az storage account keys list --resource-group $GROUP --account-name $ACCOUNT
```

Replace *resource-group-name* with name of the Azure resource group and *storage-account-name* with name of the Azure storage account.



NOTE

You can list the available resources using the command:

```
$ az resource list
```

- Make note of value **key1** in the output of the previous command, and assign it to an environment variable:

```
$ KEY1=value
```

- Create a storage container:

```
$ CONTAINER=storage-account-name
$ az storage container create --account-name $ACCOUNT \
--account-key $KEY1 --name $CONTAINER
```

Replace *storage-account-name* with name of the storage account.

Additional resources

- [Azure CLI](#)

8.5. UPLOADING VHD IMAGES TO AZURE

This describes steps to upload an VHD image to Azure.

Prerequisites

- Your system must be set up for uploading Azure VHD images.
- You must have an Azure VHD image created by Image Builder. Use the **vhd** output type in CLI or **Azure Disk Image (.vhd)** in GUI when creating the image.



NOTE

When creating a **.vhd** image using the CLI, Image Builder writes temporary files to the **/var** subdirectory. To avoid the **.vhd** image creation to fail, increase the **/var** subdirectory capacity to at least 15 to 20 GB free space to ensure availability.

Procedure

- Push the image to Azure and create an instance from it:

```
$ VHD=25ccb8dd-3872-477f-9e3d-c2970cd4bbaf-disk.vhd
$ az storage blob upload --account-name $ACCOUNT --container-name $CONTAINER --file
$VHD --name $VHD --type page
...
```

2. After the upload to the Azure BLOB completes, create an Azure image from it:

```
$ az image create --resource-group $GROUP --name $VHD --os-type linux --location eastus
--source https://$ACCOUNT.blob.core.windows.net/$CONTAINER/$VHD
- Running ...
```

3. Create an instance either with the Azure portal, or a command similar to the following:

```
$ az vm create --resource-group $GROUP --location eastus --name $VHD --image $VHD --
admin-username azure-user --generate-ssh-keys
- Running ...
```

4. Use your private key via SSH to access the resulting instance. Log in as **azure-user**.

Additional Resources

- [Composing an image for the .vhd format fails](#)

8.6. UPLOADING VMDK IMAGES TO VSPHERE

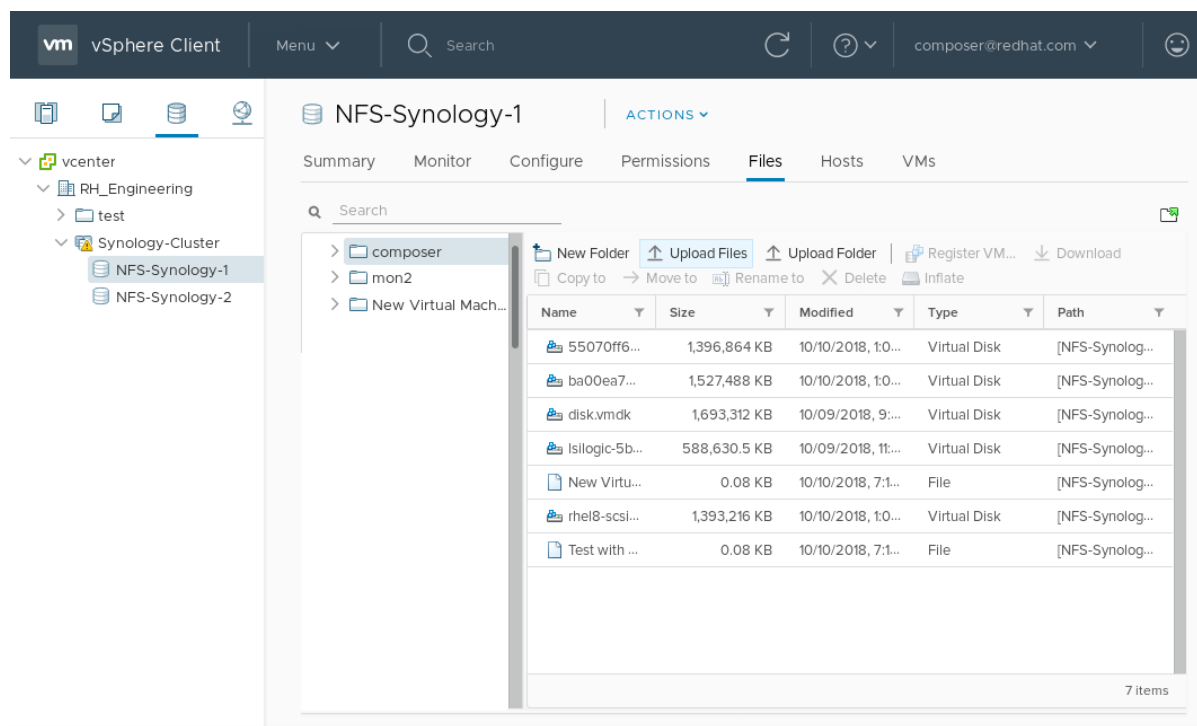
Image Builder can generate images suitable for uploading to a VMware ESXi or vSphere system. This describes steps to upload an VMDK image to VMware vSphere.

Prerequisites

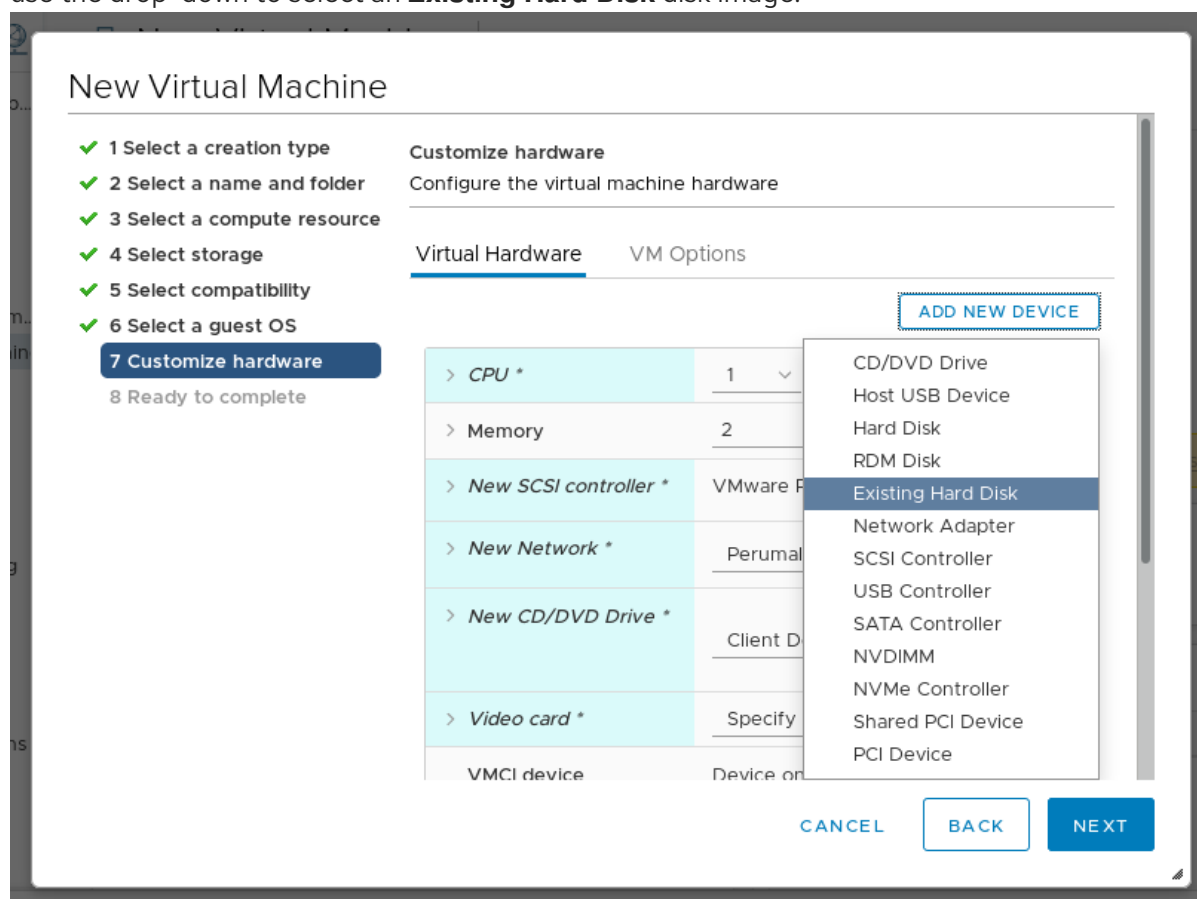
- You must have an VMDK image created by Image Builder. Use the **vmdk** output type in CLI or **VMware Virtual Machine Disk (.vmdk)** in GUI when creating the image.

Procedure

1. Upload the image into vSphere via HTTP. Click on **Upload Files** in the vCenter:



- When you create a VM, on the **Device Configuration**, delete the default **New Hard Disk** and use the drop-down to select an **Existing Hard Disk** disk image:



- Make sure you use an **IDE** device as the **Virtual Device Node** for the disk you create. The default value **SCSI** results in an unbootable virtual machine.

New Virtual Machine

- ✓ 1 Select a creation type
- ✓ 2 Select a name and folder
- ✓ 3 Select a compute resource
- ✓ 4 Select storage
- ✓ 5 Select compatibility
- ✓ 6 Select a guest OS
- 7 Customize hardware**
- 8 Ready to complete

▼ New Hard disk *	3.35546875	GB ▼
Maximum Size	N/A	
VM storage policy	Datastore Default ▼	
Sharing	Unspecified ▼	
Disk File	[NFS-Synology-1-] composer/55070ff6-d637-40fe-80f9-9518f2ee0f21-disk.vmdk	
Shares	Normal ▼	1000
Limit - IOPs	Unlimited ▼	
Virtual flash read cache	0	MB ▼
Disk Mode	Dependent ▼	
Virtual Device Node	IDE 0 ▼	

[CANCEL](#)
[BACK](#)
[NEXT](#)

8.7. PUSHING VMWARE IMAGES TO VSPHERE

You can build VMware images and push them directly to your vSphere instance, to avoid having to download the image file and push it manually. This describes steps to push **.vmdk** images you create using Image Builder directly to vSphere instances service provider.

Prerequisites

- You have **root** or **wheel** group user access to the system.
- You have opened the [Image Builder](#) interface of the RHEL web console in a browser.
- You have a [vSphere Account](#).

Procedure

1. Click **Create blueprint**.
See [Creating an Image Builder blueprint in the web console interface](#) .
2. Select the components and packages that you want as part of the image you are creating.
3. Click **Commit** to commit the changes you made to the blueprint.
A pop-up on the upper right side informs you of the saving progress and then the result of the changes you committed.
4. Click **blueprint name** link on the left banner.
5. Select the **Customizations** tab to create a user account for the blueprint.
See [Creating a user account for a blueprint](#) .

6. Select the **Images** tab .
7. Click **Create Image** to create your customized image.
The Image type window opens.
8. In the **Image type** window:
 - a. From the dropdown menu, select the Type: VMware VSphere (.vmdk).
 - b. Check the **Upload to VMware** checkbox to upload your image to the vSphere.
 - c. Optional: Set the size of the image you want to instantiate. The minimal default size is 2GB.
 - d. Click **Next**.
9. In the **Upload to VMware** window, under **Authentication**, enter the following details:
 - a. Username: username of the vSphere account.
 - b. Password: password of the vSphere account.
10. In the **Upload to VMware** window, under **Destination**, enter the following details:
 - a. **Image name**: a name for the image to be uploaded.
 - b. **Host**: The URL of your VMware vSphere where the image will be uploaded.
 - c. **Cluster**: The name of the cluster where the image will be uploaded.
 - d. **Data center**: The name of the datacenter where the image will be uploaded.
 - e. **Data store**: The name of the Data store where the image will be uploaded.
 - f. Click **Next**.
11. In the **Review** window, review the details about the image creation and click **Finish**.
You can click **Back** to modify any incorrect detail.

Image Builder adds the compose of a RHEL vSphere image to the queue, and creates and uploads the image to the Cluster on the vSphere instance you specified.



NOTE

The image build and upload processes take a few minutes to complete.

After the process is complete, you can see the **Image build complete** status.

Verification

After the image status upload is completed successfully, you can create a Virtual Machine (VM) from the image you uploaded and login into it. Follow the steps for that:

1. Access VMware vSphere Client.
2. Search for the image in the Cluster on the vSphere instance you specified.
3. You can create a new Virtual Machine from the image you uploaded. For that:

- a. Select the image you uploaded.
- b. Click the right button on the selected image.
- c. Click New Virtual Machine.

A **New Virtual Machine** window opens.

In the **New Virtual Machine** window, provide the following details:

- i. Select a creation type: You can choose to create a New Virtual Machine.
 - ii. Select a name and a folder: For example, Virtual Machine name: *vSphere Virtual Machine* and location of your choice inside vSphere Client.
 - iii. Select a computer resource: choose a destination computer resource for this operation.
 - iv. Select storage: For example, select NFS-Node1
 - v. Select compatibility: The image should be BIOS only.
 - vi. Select a guest OS: For example, select *Linux* and *_Red Hat Fedora (64-bit)*.
 - vii. **Customize hardware**: When you create a VM, on the **Device Configuration** button on the upper right, delete the default New Hard Disk and use the drop-down to select an Existing Hard Disk disk image:
 - viii. Ready to complete: Review the details and click **Finish** to create the image.
- d. Navigate to the **VMs** tab.
 - i. From the list, select the VM you created.
 - ii. Click the **Start** button from the panel. A new window appears, showing the VM image loading.
 - iii. Log in with the credentials you created for the blueprint.
 - iv. You can verify if the packages you added to the blueprint are installed. For example:

```
$ rpm -qa | grep firefox
```

Additional resources

- [Installing the vSphere Client.](#)

8.8. PUSHING VHD IMAGES TO AZURE CLOUD

You can create **.vhd** images using Image Builder. Then, you can push the **.vhd** images to a Blob Storage of the Azure Cloud service provider.

Prerequisites

- You must have root access to the system.
- You have opened the Image Builder interface of the RHEL web console in a browser.

- You must have a [Storage Account](#) created.
- You must have a writable [Blob Storage](#) prepared.

Procedure

1. Click **Create blueprint** to create a blueprint. See [Creating an Image Builder blueprint in the web console interface](#).
2. Select the components and packages that you want as part of the image you are creating.
3. Click **Commit** to commit the changes you made to the blueprint.
A small pop-up on the upper right side informs you of the saving progress and then the result of the changes you committed.
4. Click **blueprint name** link on the left banner.
5. Select the tab **Images**.
6. Click **Create Image** to create your customized image.
A pop-up window opens.
 - a. From the **"Type"** drop-down menu list, select the **Azure Disk Image (.vhd)** image.
 - b. Check the **"Upload to Azure"** check box to upload your image to the Azure Cloud and click **Next**.
 - c. To authenticate your access to Azure, type your "Storage account" and "Storage access key" in the corresponding fields. Click **Next**.
You can find your [Storage account details](#) in the Settings→Access Key menu list.
 - d. Type a **"Image name"** to be used for the image file that will be uploaded and the Blob "Storage container" in which the image file you want to push the image into. Click **Next**.
 - e. Review the information you provided and click **Finish**.
Optionally, you can click **Back** to modify any incorrect detail.
7. A small pop-up on the upper right side displays when the image creation process starts with the message: "Image creation has been added to the queue".
After the image process creation is complete, click the blueprint you created an image from.
You can see the **"Image build complete"** status for the image you created within the **Images** tab.
8. To access the image you pushed into **Azure Cloud**, access [Azure Portal](#).
9. On the search bar, type **Images** and select the first entry under **Services**. You are redirected to the **Image dashboard**.
10. Click **+Add**. You are redirected to the **Create an Image** dashboard.
Insert the below details:
 - a. **Name**: Choose a name for your new image.
 - b. **Resource Group**: Select a **resource group**.
 - c. **Location**: Select the **location** that matches the regions assigned to your storage account.
Otherwise you will not be able to select a blob.

- d. **OS Type:** Set the OS type to **Linux**.
 - e. **VM Generation:** Keep the VM generation set on **Gen 1**.
 - f. **Storage Blob:** Click **Browse** on the right of **Storage blob input**. Use the dialog to find the image you uploaded earlier.
Keep the remaining fields as in the default choice.
11. Click **Create** to create the image. After the image is created, you can see the message **"Successfully created image"** in the upper right corner.
 12. Click **Refresh** to see your new image and open your newly created image.
 13. Click **+ Create VM**. You are redirected to the **Create a virtual machine** dashboard.
 14. In the **Basic** tab, under **Project Details**, your ***Subscription** and the **Resource Group** are already pre-set.
If you want to create a new resource Group
 - a. Click **Create new**.
A pop-up prompts you to create the **Resource Group Name** container.
 - b. Insert a name and click **OK**.
If you want to keep the **Resource Group** that is already pre-set.
 15. Under **Instance Details**, insert:
 - a. **Virtual machine name**
 - b. **Region**
 - c. **Image:** The image you created is pre-selected by default.
 - d. **Size:** Choose a VM size that better suits your needs.
Keep the remaining fields as in the default choice.
 16. Under **Administrator account**, enter the below details:
 - a. **Username:** the name of the account administrator.
 - b. **SSH public key source:** from the drop-down menu, select **Generate new key pair**.
You can either use the key pair you already have or you can create a new key pair.
Alternatively, you can use **Image Builder** to add a user to the image with a preset public key. See [Creating a user account with SSH key](#) for more details.
 - c. **Key pair name:** insert a name for the key pair.
 17. Under **Inbound port rules**, select:
 - a. **Public inbound ports:** **Allow selected ports**
 - b. **Select inbound ports:** Use the default set **SSH (22)**.
 18. Click **Review + Create**. You are redirected to the **Review + create** tab and receive a confirmation that the validation passed.
 19. Review the details and click **Create**.
Optionally, you can click **Previous** to fix previous options selected.

20. A pop-up **generates new key pair** window opens. Click **Download private key and create resources**.
Save the key file as "*yourKey.pem*".
21. After the deployment is complete, click **Go to resource**.
22. You are redirected to a new window with your VM details. Select the public IP address on the top right side of the page and copy it to your clipboard.

Now, to create an SSH connection with the VM to connect to the Virtual Machine.

1. Open a terminal.
2. At your prompt, open an SSH connection to your virtual machine. Replace the IP address with the one from your VM, and replace the path to the .pem with the path to where the key file was downloaded.

```
# ssh -i ./Downloads/yourKey.pem azureuser@10.111.12.123
```

3. You are required to confirm if you want to continue to connect. Type yes to continue.

As a result, the output image you pushed to the Azure Storage Blob is ready to be provisioned.

Additional resources

- [Azure Storage Documentation](#).
- [Create an Azure Storage account](#).
- [Open a case on Red Hat Customer Portal](#).
- [Help + support](#).
- [Contacting Red Hat](#).

8.9. PUSHING CUSTOMIZED IMAGES TO OCI

With Image Builder, you can build customized images and push them directly to your Oracle Cloud Infrastructure (OCI) instance. Then, you can launch an image instance from the OCI dashboard.

Prerequisites

- You have **root** or **wheel** group user access to the system.
- You have an [Oracle Cloud](#) account.
- You must be granted security access in an **OCI policy** by your administrator.
- You have created an OCI Bucket in the **OCI_REGION** of your choice.

Procedure

1. Open the Image Builder interface of the web console in a browser.
2. Click **Create blueprint**.

See [Creating an Image Builder blueprint in the web console interface](#) .

3. Select the components and packages that you want to include in the image.
4. Click **Commit** to commit the changes you made to the blueprint.
A pop-up on the upper right side informs you of the saving progress and then the result of the changes you committed.
5. Click **blueprint name** link on the left banner.
6. Select the **Customizations** tab to create a user account for the blueprint.
See [Creating a user account for a blueprint](#) .
7. Select the **Images** tab.
8. Click **Create Image**. The **Create image** wizard opens.
9. On the **Image type** page, complete the following steps:
 - a. From the **"Type"** drop-down menu list, select the **Oracle Cloud Infrastructure (.qcow2)** image.
 - b. Check the **"Upload OCI"** check box to upload your image to the OCI.
 - c. Enter the **"image size"**. Click **Next**.
10. On the **Upload to OCI - Authentication** page, enter the following mandatory details:
 - a. User OCID: you can find it in the Console on the page showing the user's details.
 - b. Private key
 - c. Fingerprint
11. On the **Upload to OCI - Destination** page, enter the following mandatory details and click **Next**.:
 - a. Image name: a name for the image to be uploaded.
 - b. OCI bucket
 - c. Bucket namespace
 - d. Bucket region
 - e. Bucket compartment
 - f. Bucket tenancy
12. Review the details in the wizard and click **Finish**.

Image Builder adds the compose of a RHEL **.qcow2** image to the queue.

Verification

1. Access the [OCI dashboard](#) → Custom Images.

2. Select the Compartment you specified for the image and locate the image in the **Import image** table.
3. Click the image name and verify the image information.

Additional resources

- [Managing custom images in the OCI](#)
- [Managing buckets in the OCI](#)
- [Generating SSH keys](#)

8.10. UPLOADING QCOW2 IMAGE TO OPENSTACK

Image Builder can generate images suitable for uploading to OpenStack cloud deployments, and starting instances there. This describes steps to upload an QCOW2 image to OpenStack.

Prerequisites

- You must have an OpenStack-specific image created by Image Builder. Use the **openstack** output type in CLI or **OpenStack Image (.qcow2)** in GUI when creating the image.



WARNING

Image Builder also offers a generic QCOW2 image type output format as **qcow2** or **QEMU QCOW2 Image (.qcow2)**. Do not mistake it with the OpenStack image type which is also in the QCOW2 format, but contains further changes specific to OpenStack.

Procedure

1. Upload the image to OpenStack and start an instance from it. Use the **Images** interface to do this:

Create An Image

Name: *

96268ffb-2c71-4e97-a855-7ac25e983a6e-disk.qcow2

Description:

Image Source:

Image File

Image File

Browse...

96268ffb-2c71-4e97-a85...c25e98

Format: *

QCOW2 - QEMU Emulator

Architecture:

x86_64

Minimum Disk (GB):

5

Minimum Ram (MB):

1024

Public:

☒

Protected:

☐

Cancel

Create Image

Description:

Specify an image to upload to the Image Service.

Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)

Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

2. Start an instance with that image:

Launch Instance

Details *

Access & Security *

Networking *

Post-Creation

Advanced Options

Availability Zone:

nova

Instance Name: *

my-instance

Flavor: *

m1.small

Some flavors not meeting minimum image requirements have been disabled.

Instance Count: *

1

Instance Boot Source: *

Boot from image

Image Name:

96268ffb-2c71-4e97-a855-7ac25e983a6e-disk.qc

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.small
VCPUs	1
Root Disk	20 GB
Ephemeral Disk	0 GB
Total Disk	20 GB
RAM	2,048 MB

Project Limits

Number of Instances

4 of 10 Used

Number of VCPUs

17 of 20 Used

Total RAM

34,816 of 51,200 MB Used

Cancel

Launch

- You can run the instance using any mechanism (CLI or OpenStack web UI) from the snapshot. Use your private key via SSH to access the resulting instance. Log in as **cloud-user**.

8.11. PREPARING FOR UPLOADING IMAGES TO ALIBABA

This section describes steps to verify custom images that you can deploy on Alibaba Cloud. The images will need a specific configuration to boot successfully, because Alibaba Cloud requests the custom images to meet certain requirements before you use it. For this, it is recommended that you use the Alibaba **image_check tool**.



NOTE

The custom image verification is an optional task. Image Builder generates images that conform to Alibaba's requirements.

Prerequisites

- You must have an Alibaba image created by Image Builder.

Procedure

- Connect to the system containing the image you want to check it by the Alibaba **image_check tool**.

2. Download the **image_check** tool:

```
$ curl -O http://docs-aliyun.cn-hangzhou.oss.aliyun-inc.com/assets/attach/73848/cn_zh/1557459863884/image_check
```

3. Change the file permission of the image compliance tool:

```
# chmod +x image_check
```

4. Run the command to start the image compliance tool checkup:

```
# ./image_check
```

The tool verifies the system configuration and generates a report that is displayed on your screen. The `image_check` tool saves this report in the same folder where the image compliance tool is running.

5. If any of the **Detection Items** fail, follow the instructions to correct it. For more information, see link: [Detection items section](#).

Additional resources

- [Image Compliance Tool](#)

8.12. UPLOADING IMAGES TO ALIBABA

This section describes how to upload an Alibaba image to Object Storage Service (OSS).

Prerequisites

- Your system is set up for uploading Alibaba images.
- You must have an Alibaba image created by Image Builder. Use the **ami** output type on RHEL 7 or Alibaba on RHEL 8 when creating the image.
- You have a bucket. See [Creating a bucket](#).
- You have an [active Alibaba Account](#).
- You activated [OSS](#).

Procedure

1. Log in to the [OSS console](#).
2. On the left side Bucket menu, select the bucket to which you want to upload an image.
3. On the right upper menu, click the **Files** tab.
4. Click **Upload**. A window dialog opens on the right side. Choose the following information:
 - **Upload To:** Choose to upload the file to the **Current** directory or to a **Specified** directory.
 - **File ACL:** Choose the type of permission of the uploaded file.

5. Click **Upload**.
6. Choose the image you want to upload.
7. Click **Open**.

As a result, the custom image is uploaded to OSS Console.

Additional resources

- [Upload an object](#)
- [Creating an instance from custom images](#)
- [Importing images](#)

8.13. IMPORTING IMAGES TO ALIBABA

This section describes how to import an Alibaba image to Elastic Cloud Console (ECS).

Prerequisites

- You have uploaded the image to Object Storage Service (OSS).

Procedure

1. Log in to the [ECS console](#).
 - i. On the left side menu, click **Images**.
 - ii. On the right upper side, click **Import Image**. A window dialog opens.
 - iii. Confirm that you have set up the correct region where the image is located. Enter the following information:
 - a. **OSS Object Address**: See how to obtain [OSS Object Address](#).
 - b. **Image Name**:
 - c. **Operating System**:
 - d. **System Disk Size**:
 - e. **System Architecture**:
 - f. **Platform**: Red Hat
 - iv. Optionally, provide the following details:
 - g. **Image Format**: qcow2 or ami, depending on the uploaded image format.
 - h. **Image Description**:
 - i. **Add Images of Data Disks**

The address can be determined in the OSS management console after selecting the required bucket in the left menu, select Files section and then click the **Details** link on the right for the appropriate image. A window will appear on the right side of the screen,

showing image details. The OSS object address is in the URL box.

2. Click **OK**.



NOTE

The importing process time can vary depending on the image size.

As a result, the custom image is imported to ECS Console. You can create an instance from the custom image.

Additional resources

- [Notes for importing images](#)
- [Creating an instance from custom images](#)
- [Upload an object](#)

8.14. CREATING AN INSTANCE OF A CUSTOM IMAGE USING ALIBABA

You can create instances of the custom image using Alibaba ECS Console.

Prerequisites

- You have activated [OSS](#) and uploaded your custom image.
- You have successfully imported your image to ECS Console.

Procedure

1. Log in to the [ECS console](#).
2. On the left side menu, choose **Instances**.
3. In the top corner, click **Create Instance**. You are redirected to a new window.
4. Fill in all the required information. See [Creating an instance by using the wizard](#) for more details.
5. Click **Create Instance** and confirm the order.



NOTE

You can see the option **Create Order** instead of **Create Instance**, depending on your subscription.

As a result, you have an active instance ready for deployment.

Additional resources

- [Creating an instance by using a custom image](#)
- [Create an instance by using the wizard](#)

