Red Hat Enterprise Linux 8

Composing a customized RHEL system image

Creating customized system images with Image Builder on Red Hat Enterprise Linux 8

Last Updated: 2020-09-13

Red Hat Enterprise Linux 8 Composing a customized RHEL system image

Creating customized system images with Image Builder on Red Hat Enterprise Linux 8

Legal Notice

Abstract

Image Builder is a tool for creating deployment-ready customized system images: installation disks, virtual machines, cloud vendor-specific images, and others. Image Builder enables you to create these images faster compared to manual procedures, because it abstracts away the specifics of each output type. Learn how to set up Image Builder and create images with it.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. 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 8 web console.

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

- CLI tool **composer-cli** for running commands in the terminal. Prefer this method.
- GUI plugin for the RHEL 8 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 8 web console.
Blueprints are presented to the user as plain text in the Tom's Obvious, Minimal Language (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.

Table 1.1. Image Builder output formats

Description	CLI name	file extension
QEMU QCOW2 Image	**qcow2**	**.qcow2**
Ext4 File System Image	**ext4-filesystem**	**.img**
Raw Partitioned Disk Image	**partitioned-disk**	**.img**
Live Bootable ISO	**live-iso**	**.iso**

Description	CLI name	file extension
TAR Archive	**tar**	**.tar**
Amazon Machine Image Disk	**ami**	**.ami**
Azure Disk Image	**vhd**	**.vhd**
VMware Virtual Machine Disk	**vmdk**	**.vmdk**
Openstack	**openstack**	**.qcow2**

1.4. IMAGE BUILDER SYSTEM REQUIREMENTS

The **lorax** tool underlying Image Builder performs a number of potentially insecure and unsafe actions while creating the system images. For this reason, use a virtual machine to run Image Builder.

The environment where Image Builder runs, for example the 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
Access privileges	Administrator level (root)
Network	Connectivity to Internet

NOTE

There is no support for creating images on virtual machine directly installed on UEFI systems.

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, and running.

Procedure

1. Install the Image Builder and other necessary packages on the virtual machine:
   - lorax-composer
   - composer-cli
   - cockpit-composer
   - bash-completion

   ```
   # yum install lorax-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 lorax-composer.socket
   # systemctl enable cockpit.socket
   ```

   The **lorax-composer** and **cockpit** services start automatically on first access.
3. Configure the system firewall to allow access to the web console:

   ```
   # firewall-cmd --add-service=cockpit && firewall-cmd --add-service=cockpit --permanent
   ```
4. 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
   ```

CHAPTER 3. 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.

3.1. 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** command as non-root, user must be in the **weldr** or **root** groups.

3.2. 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.

2. 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 **8.30**.
- For latest available version, use the asterisk *****.
- For a latest minor version, use format such as **8.***.

3. Blueprints can be customized in a number of ways. For this example, Simultaneous Multi Threading (SMT) can be disabled by performing the steps below. For additional customizations available, please see Section 3.7, "Supported Image Customizations".

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

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

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

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

6. To verify that the blueprint has been pushed and exists, list the existing blueprints:

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

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

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

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

This procedure describes how to edit an existing Image Builder blueprint in the command-line interface.

Procedure

1. Save (export) the blueprint to a local text file:

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

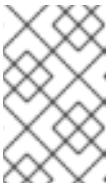
2. Edit the *BLUEPRINT-NAME*.toml file with a text editor of your choice and make your changes.
3. Before finishing with the edits, make sure the file is a valid blueprint:

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** component of version is increased 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 editor.

5. Push (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.

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
```

3.4. 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 the UUID of the compose is shown.

2. Wait until the compose is finished. Please, notice that this may take several minutes.
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. Once 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.

Alternatively, you can access the image file directly under the path **/var/lib/lorax/composer/results/*UUID*/**.

You can also download the logs using the **composer-cli compose logs *UUID*** command, or the metadata using the **composer-cli compose metadata *UUID*** command.

3.5. 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

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
```

Related information

- 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
  ```

3.6. IMAGE BUILDER BLUEPRINT FORMAT

Image Builder blueprints are presented to the user as plain text in the Tom's Obvious, Minimal Language (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**.

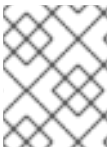
Replace *package-version* with a version to use. This field supports **dnf** version specifications:

- For a specific version, use the exact version number such as **8.30**.
- For latest available version, use the asterisk *****.
- For a latest minor version, use format such as **8.***.

Repeat this block for every package to include.

3.7. SUPPORTED IMAGE CUSTOMIZATIONS

A number of image customizations are supported at this time within blueprints. In order to make use of these options, they must be initially configured in the blueprint and imported (pushed) to Image Builder.



NOTE

These customizations are not currently supported within the accompanying cockpit-composer GUI.

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
```

IMPORTANT

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

```
# yum install python3
```

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

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

Replace *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, use the **User specifications for the resulting system image** customization.

Append a kernel boot parameter option to the defaults

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

Set the image host name

```
[customizations]
hostname = "BASE-IMAGE"
```

Add a group for the resulting system image

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

Only the name is required and GID is optional.

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]
language = ["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 theirs names from the /**etc**/**services** file to enable lists.

Customize the firewall services

Review the available firewall services.

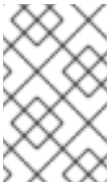
```
$ firewall-cmd --get-services
```

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.

Add files from a git repository to your blueprint

```
[[repos.git]]
rpmname = "RPM-NAME"
rpmversion = "RPM-VERSION"
rpmrelease = "RPM-RELEASE"
summary = "RPM-SUMMARY"
repo = "REPO-URL"
ref = "GIT-REF"
destination = "SERVICES"
```

You can use entries to add files from a git repository to the created image.

For example, to create an RPM package named **server-config-1.0-1.noarch.rpm**, add the following information to your blueprint:

Replace *_RPM-NAME* with the name of the RPM package to create. This is also the prefix name in the resulting tar archive.

Replace *RPM-VERSION* with the version of the RPM package, for example, “1.0.0”.

Replace *RPM-RELEASE* with the version of the RPM package release, for example, “1”.

Replace *RPM-SUMMARY* with the summary string for the RPM package.

Replace *REPO-URL* with the URL of the get repository to clone and create the archive from it.

Replace *GIT-REF* with the git reference to check out, for example, **origin/branch-name**, **git tag**, or **git commit hash**.

Replace *SERVICES* with the path to install the directory of the git repository when installing the RPM package.

As a consequence, the git repository you provided is cloned, the specified git reference is checked out and an RPM package is created to install the files to a destination path, for example, **/opt/server/**. The RPM includes a summary with the details of the repository and reference used to create it. The RPM package is also included in the image build metadata.

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 once the RPM package is created.

3.8. 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.

Table 3.1. Default packages to support image type creation

Image type	Default Packages
alibaba.ks	kernel, selinux-policy-targeted, cloud-init
ami.ks	kernel, selinux-policy-targeted, chrony, cloud-init
ext4-filesystem.ks	policycoreutils, selinux-policy-targeted, kernel
google.ks	kernel, selinux-policy-targeted
live-iso.ks	isomd5sum, kernel, dracut-config-generic, dracut-live, system-logos, selinux-policy-targeted
openstack.ks	kernel, selinux-policy-targeted
partitioned-disk.ks	kernel, selinux-policy-targeted
qcow2.ks	kernel, selinux-policy-targeted

Image type	Default Packages
tar.ks	policycoreutils, selinux-policy-targeted
vhd.ks	kernel, selinux-policy-targeted, chrony, WALinuxAgent, python3, net-tools, cloud-init, cloud-utils-growpart, gdisk
vmdk.ks	kernel, selinux-policy-targeted, chrony, open-vm-tools

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

- For more information about the available image types, see Section 1.3, "Image Builder output formats"

3.9. ENABLED SERVICES

When you configure the custom image, the services enabled are the defaults services for the RHEL8 release you are running lorax-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 3.2. Enabled services to support image type creation

Image type	Enabled Services
alibaba.ks	sshd,cloud-init
ami.ks	No default service
ext4-filesystem.ks	No default service
google.ks	No default service
live-iso.ks	NetworkManager
openstack.ks	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
partitioned-disk.ks	No default service

Image type	Enabled Services
qcow2.ks	No default service
tar.ks	No default service
vhd.ks	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk.ks	sshd, chronyd, vmtoolsd

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

- For more information about service customization, see Section 3.7, "Supported Image Customizations"

3.10. DISKS AND PARTITIONS CONFIGURATION USING IMAGE BUILDER

Image Builder does not allow disks to be partitioned. The output types that have a partitioned disk will have a single partition and additionally any platform-specific partitions that are required to boot the system image. For example, **qcow2** image type has a single root partition, and possibly a platform specific boot partition - like **PReP** for **PPC64** system - that the image requires to boot.

CHAPTER 4. 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.

4.1. ACCESSING IMAGE BUILDER GUI IN THE RHEL 8 WEB CONSOLE

The **cockpit-composer** plugin for the RHEL 8 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 how to remotely access Image Builder, see *Managing systems using the RHEL 8 web console* document.
2. Log into the web console with credentials for an 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.

Related information

- Chapter 3, *Creating system images with Image Builder command-line interface*

4.2. 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 8 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:

1. 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 these that match the search.

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

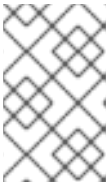
2. The list of components is paged. To move to other result pages, use the arrows and entry field above the component list.

3. 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.

4. Select the version you want to use in the **Component Options** box, with the **Version Release** dropdown.

5. Click **Add** in the top left.

6. 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.

4.3. 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 8 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 these 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 these 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 these that match the search.

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

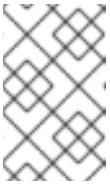
   b. The list of components is paged. To move to other result pages, use the arrows and entry field above the component list.

   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.

4.4. 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:

```
# yum 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 8 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 CLI text editor of your choice and add the users and groups.

IMPORTANT

RHEL 8 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 editor.

6. Import the blueprint back into Image Builder:

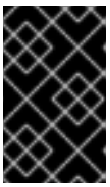
```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

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 8 web console does not show any information that could be used to verify that the changes have been applied, unless you edited also 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 information

- Section 3.6, "Image Builder blueprint format"
- Section 3.3, "Editing an Image Builder blueprint with command-line interface"

4.5. 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 8 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 these 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. Once the image is successfully created, the **Stop** button is replaced by a **Download** button. Click this button to download the image to your system.

4.6. 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.

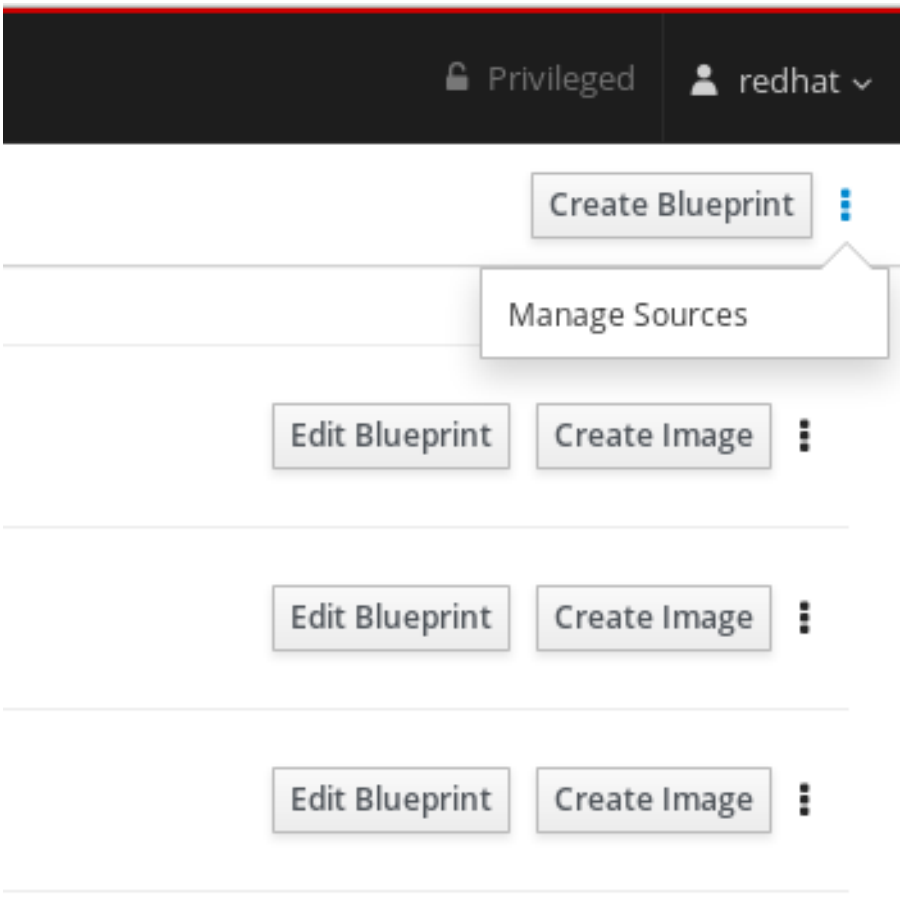
The following steps describe how to add a Source to your local system.

Prerequisites

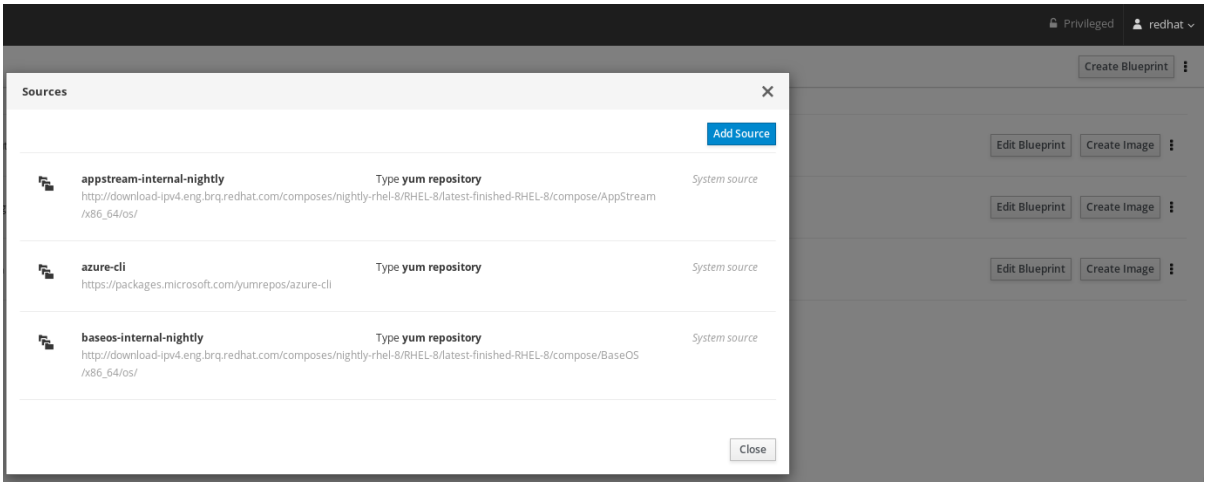
- You have opened the Image Builder interface of the RHEL 8 web console in a browser.

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 list the source you have added.

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

4.7. 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

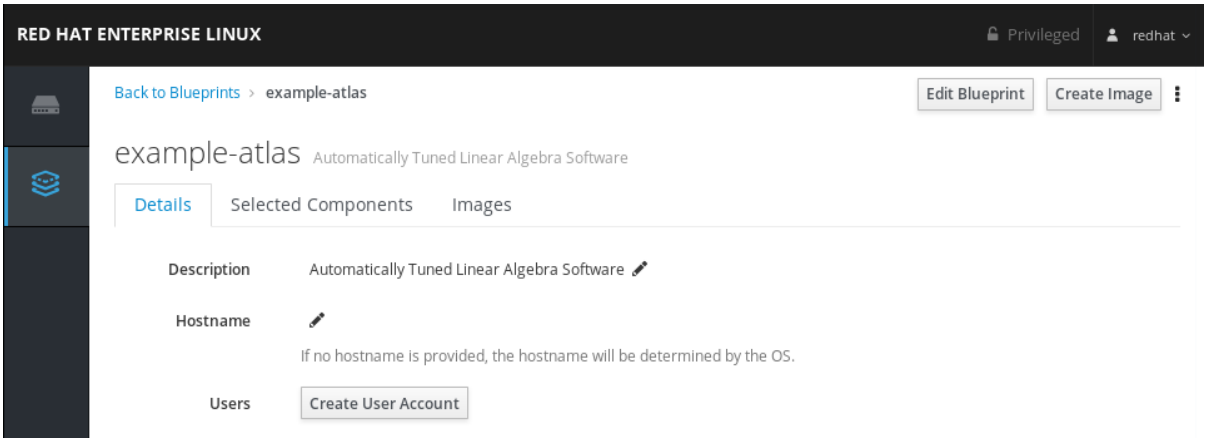
- You have opened the Image Builder interface of the RHEL 8 web console in a browser.
- You have an existing blueprint.

Procedure

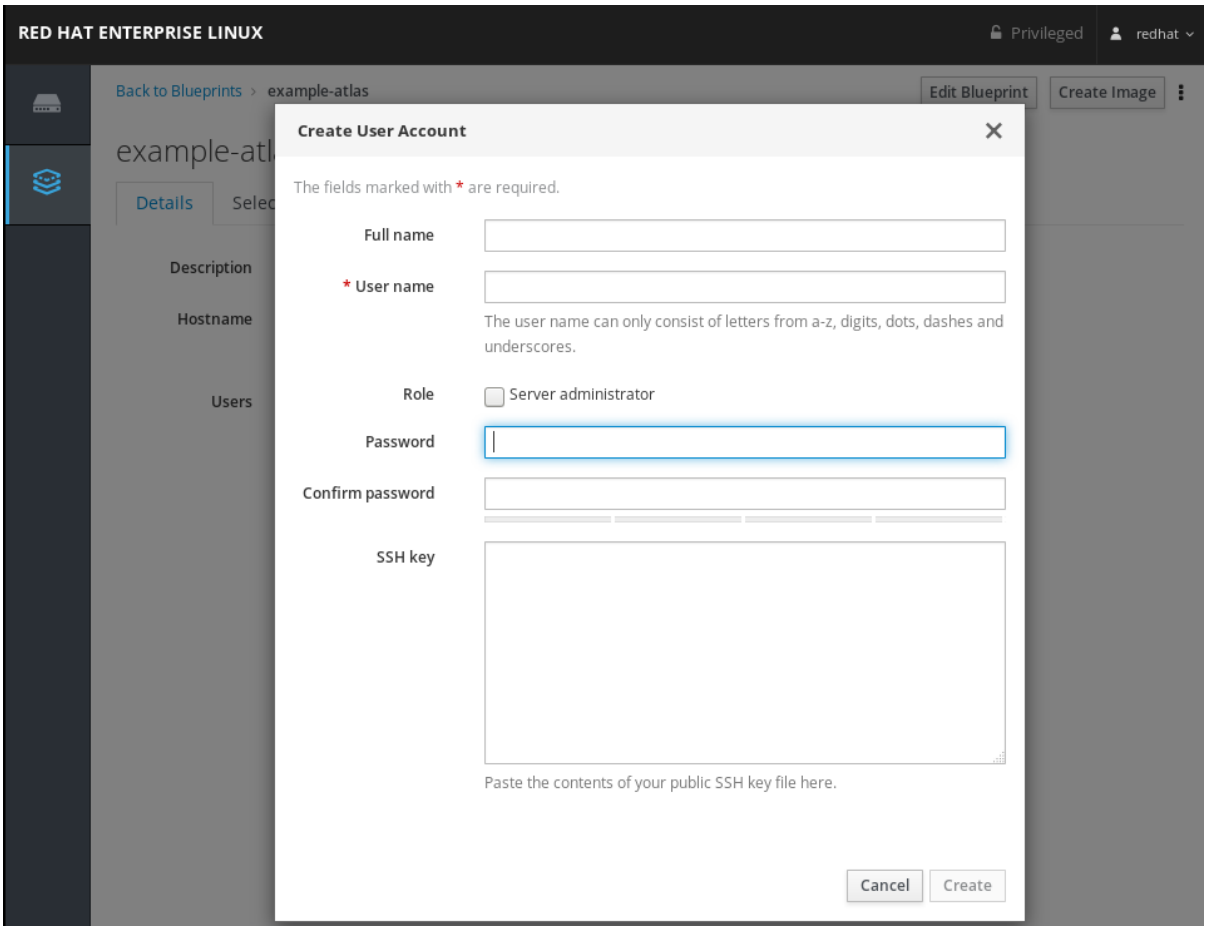
1. Locate the blueprint that you want to create a user account for by entering its name or a part of it into the search box at the top left, and press **Enter**.

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.



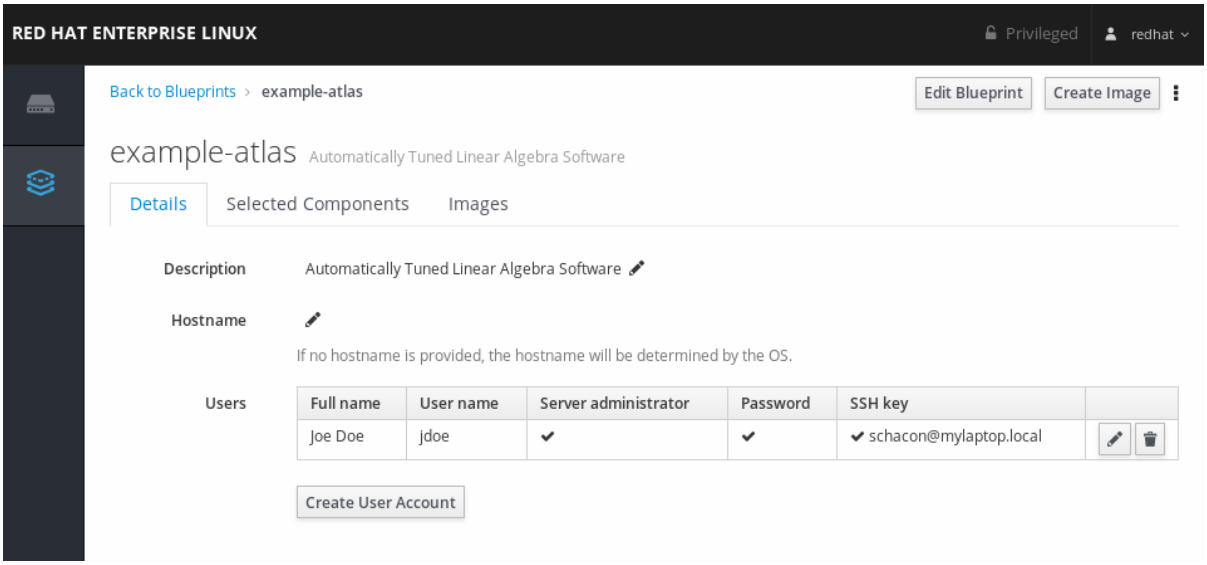
3. Click **Create User Account**.
This will open a window with fields for user account creation.



4. Fill in the details. Notice that when you insert the name, the **User name** field autocompletes, suggesting a username.

5. Once you have inserted all the desired details, click **Create**.

6. The created user account appears showing all the information you have inserted.



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

4.8. 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.

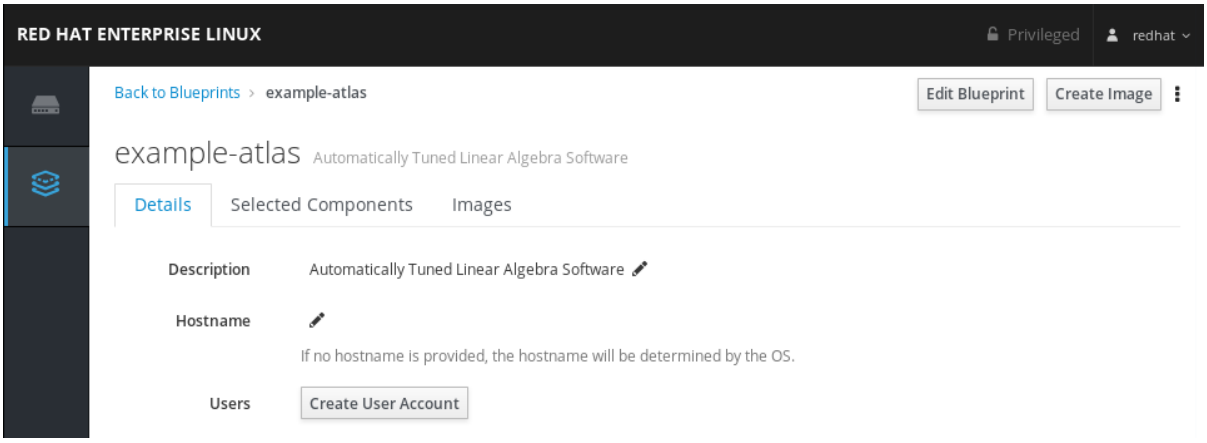
Prerequisites

- 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 8 web console in a browser.
- You have an existing blueprint

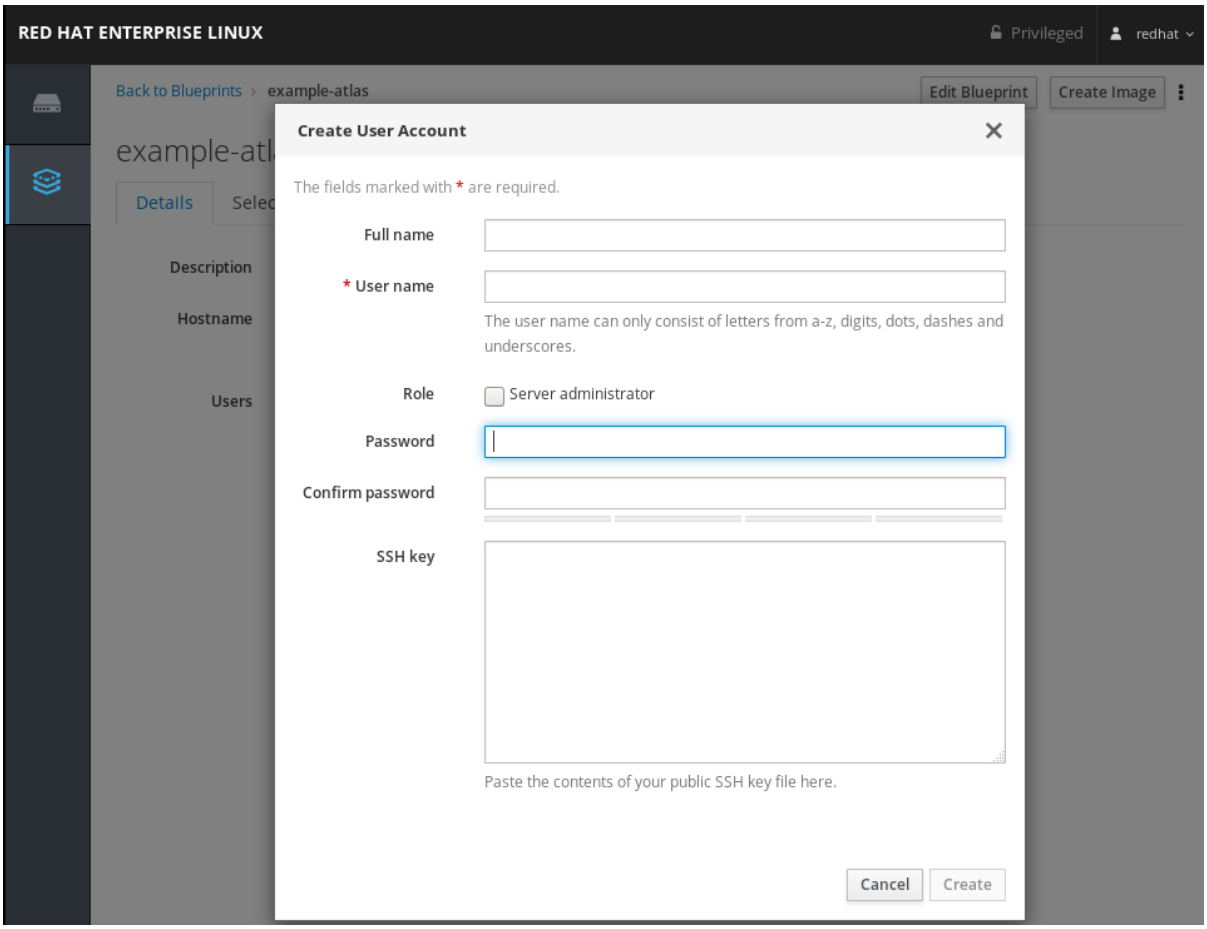
Procedure

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

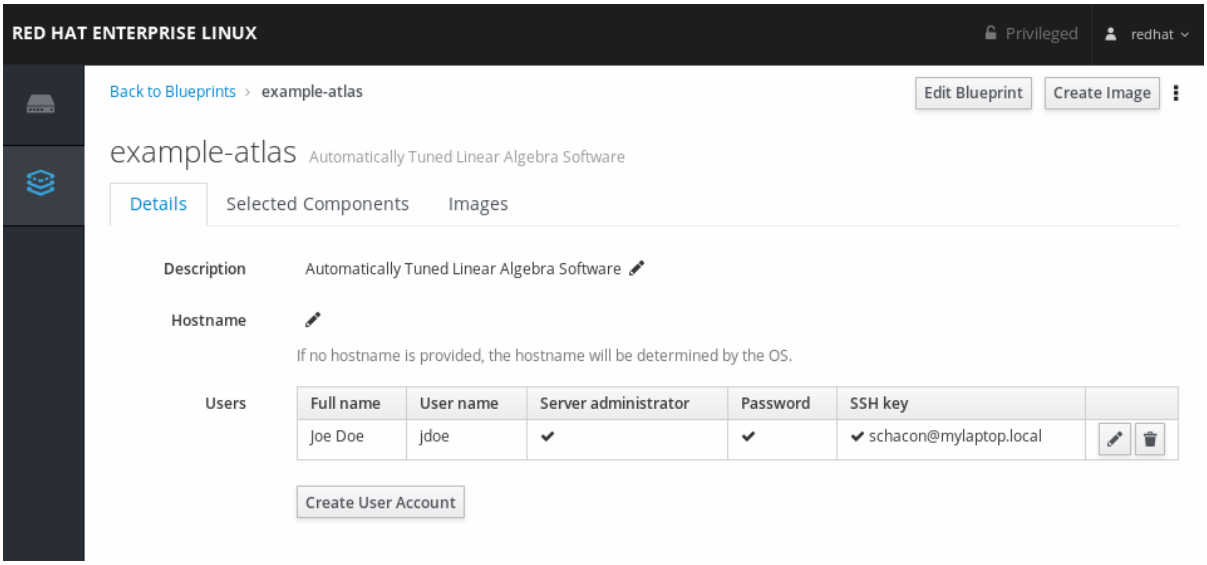


4. Fill in the details. Notice that when you insert the name, the **User name** field autocompletes, suggesting a username.
   If you want to provide administrators rights to the user account you are creating, check the **Role** field.

   Paste the content of your public SSH key file.

5. Once 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

- For more details on SSH keys, see the Using SSH Keys.

CHAPTER 5. 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.

5.1. 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:

```
# yum install python3
# yum install python3-pip
```

2. Install the AWS command-line tools with **pip**:

```
# pip3 install awscli
```

3. Configure the AWS command-line client according to your AWS access details:

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

4. Configure the AWS command-line client to use your bucket:

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

Replace *bucketname* with the actual bucket name.

5. Create a **vmimport** S3 Role in IAM and grant it permissions to access S3, 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
```

5.2. UPLOADING AN AMI IMAGE TO AWS

This section describes how to upload an AMI image to AWS.

Prerequisites

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

Procedure

1. Push the image to S3:

```
$ AMI=8db1b463-91ee-4fd9-8065-938924398428-disk.ami
$ aws s3 cp $AMI s3://$BUCKET
Completed 24.2 MiB/4.4 GiB (2.5 MiB/s) with 1 file(s) remaining
...
```

2. After the upload to S3 ends, import the image as a snapshot into EC2:

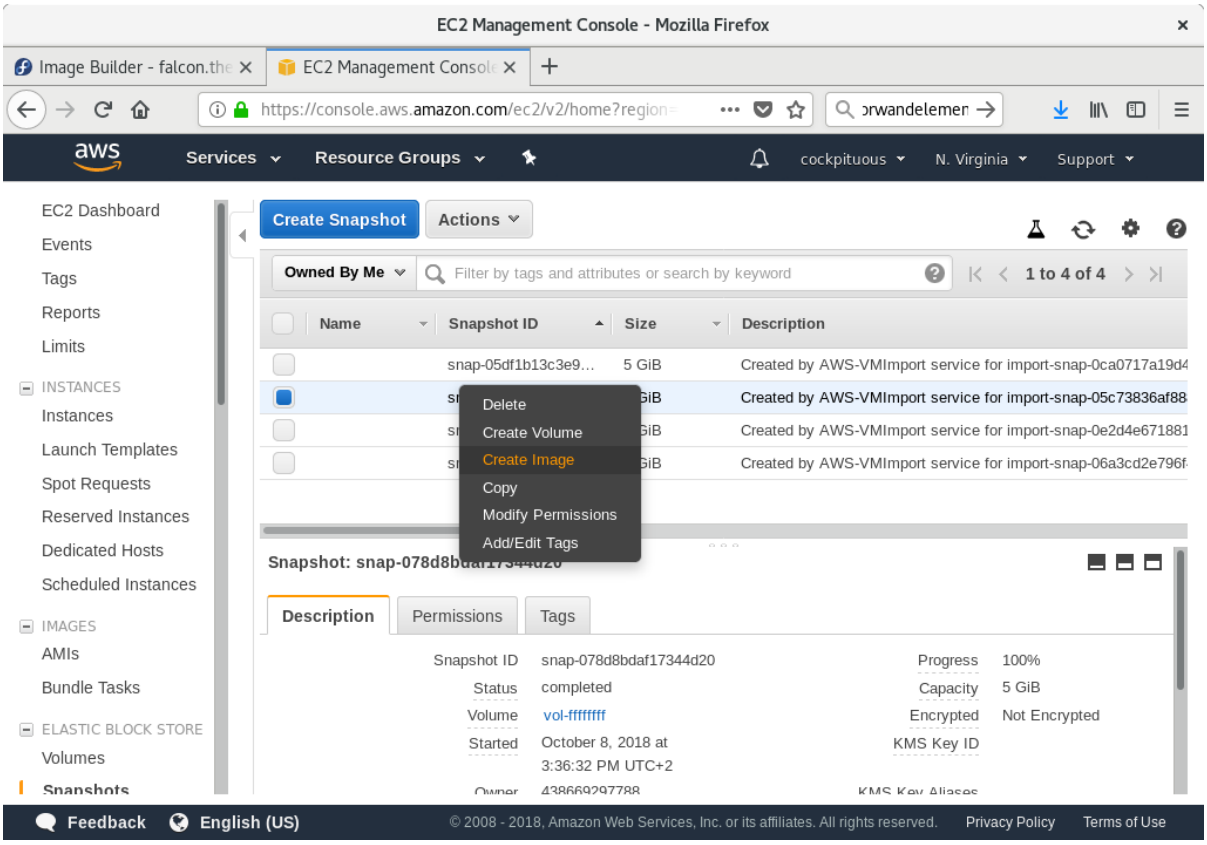
```
$ printf '{ "Description": "my-image", "Format": "raw", "UserBucket": { "S3Bucket": "%s",
"S3Key": "%s" } }' $BUCKET $AMI > containers.json
$ aws ec2 import-snapshot --disk-container file://containers.json
```

   Replace *my-image* with the name of the image.

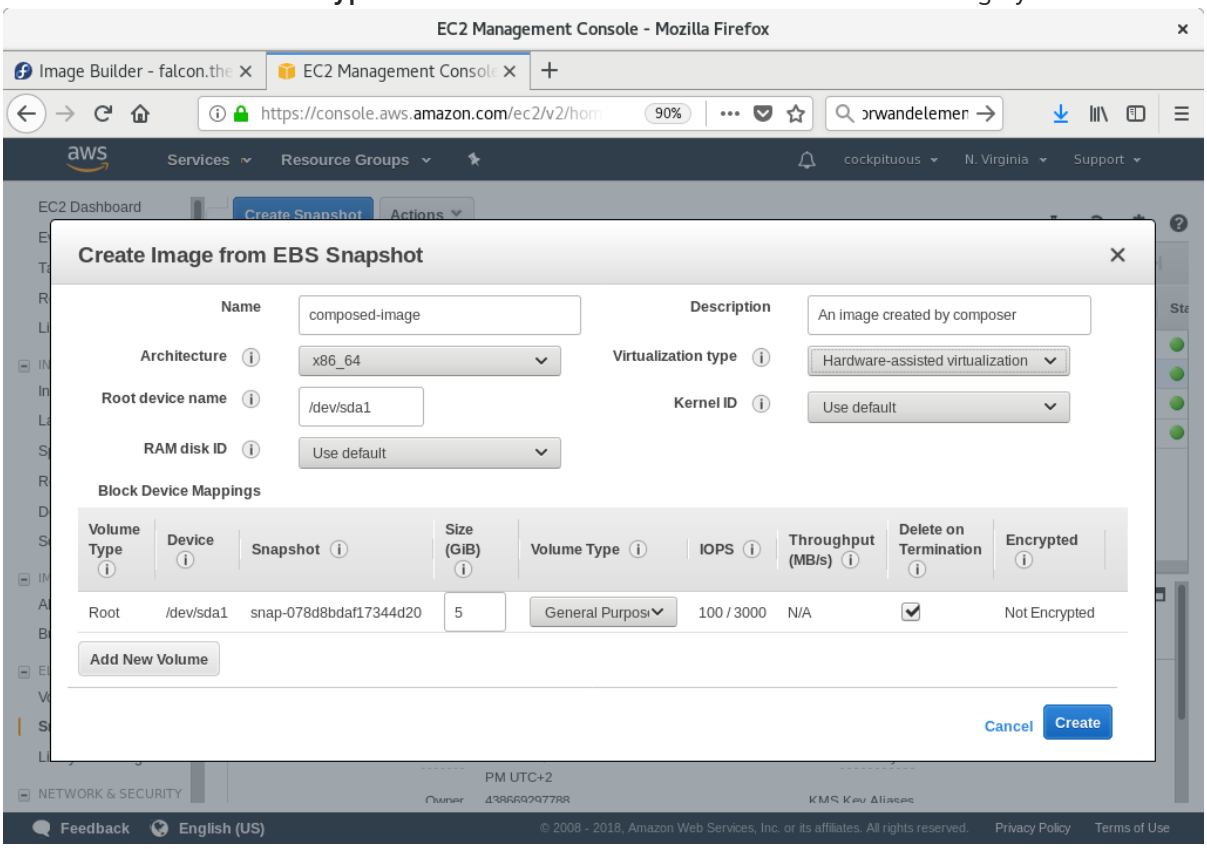
   To track progress of the import, run:

```
$ aws ec2 describe-import-snapshot-tasks --filters Name=task-state,Values=active
```

3. Create an image from the uploaded snapshot by selecting the snapshot in the EC2 console, right clicking on it and selecting **Create Image**:



4. Select the **Virtualization type** of **Hardware-assisted virtualization** in the image you create:



5. Now you can run an instance using whatever mechanism you like (CLI or AWS Console) from the snapshot. Use your private key via SSH to access the resulting EC2 instance. Log in as **ec2-user**.

5.3. 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:

```
# yum 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
CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/azure-cli.repo'
```

4. Install the Azure CLI:

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

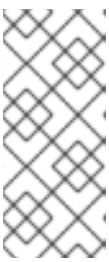
NOTE

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 opens a browser where you can login.



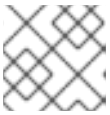
NOTE

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 autenticate your remote session. To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code *XXXXXXXXX* 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
```

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

```
$ KEY1=value
```

8. 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

5.4. 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.

Procedure

1. 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. Once 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**.

5.5. 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.

NOTE

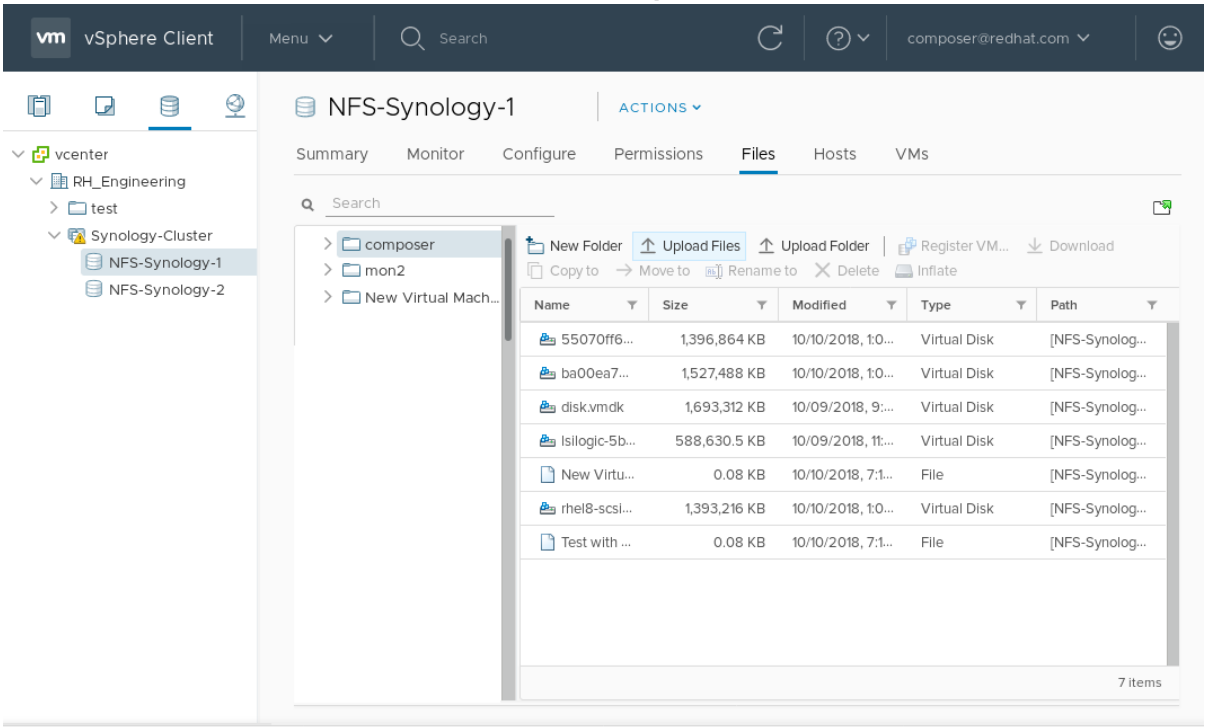
Because VMWare deployments typically does not have cloud-init configured to inject user credentials to virtual machines, we must perform that task ourselves on the blueprint.

Prerequisites

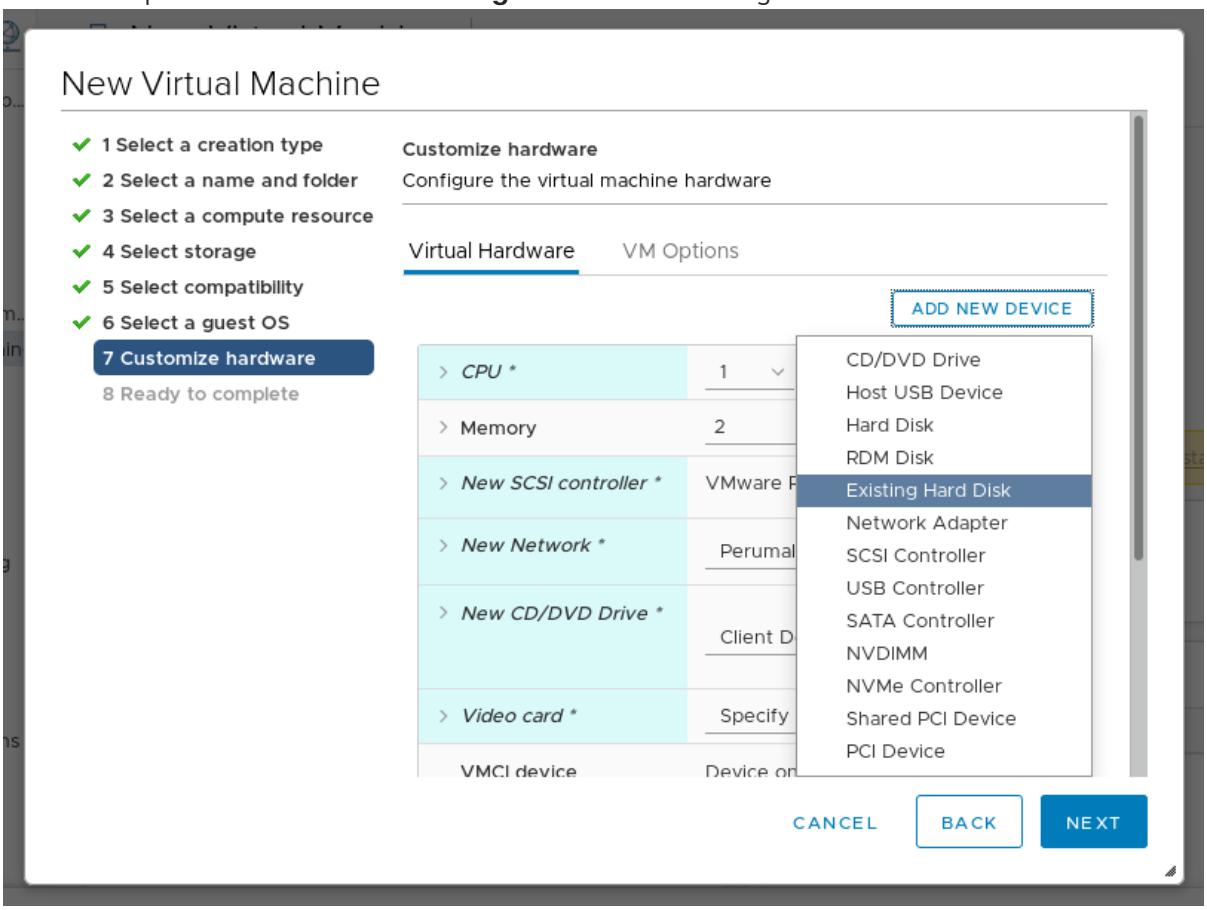
- You must have an VMDK image created by Image Builder. Use the **vmdk** output type in CLI or **VMware Virtual Machine Disk (.vmdk)**in GUI when creating the image.

Procedure

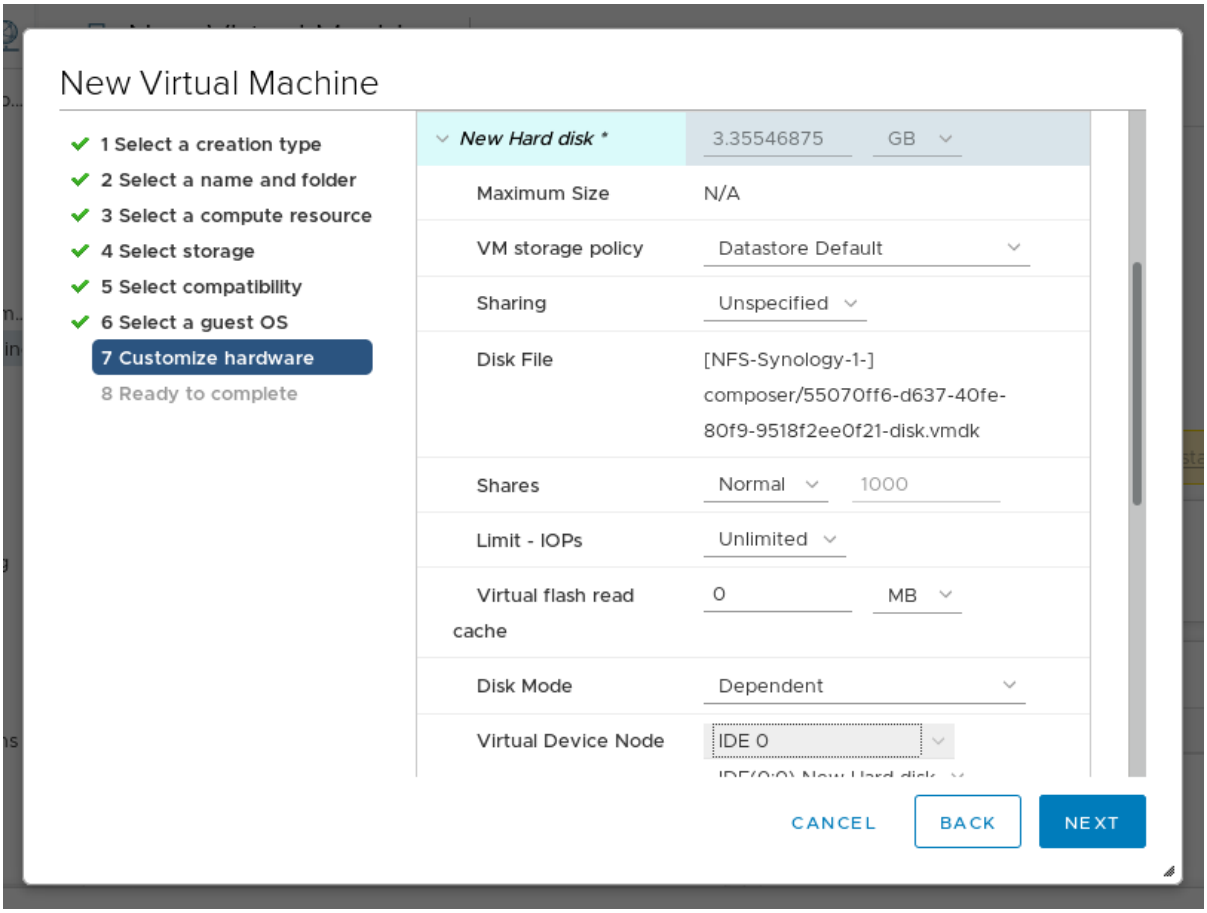
1. Upload the image into vSphere via HTTP. Click on **Upload Files** in the vCenter:



2. 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:



3. 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.



5.6. 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

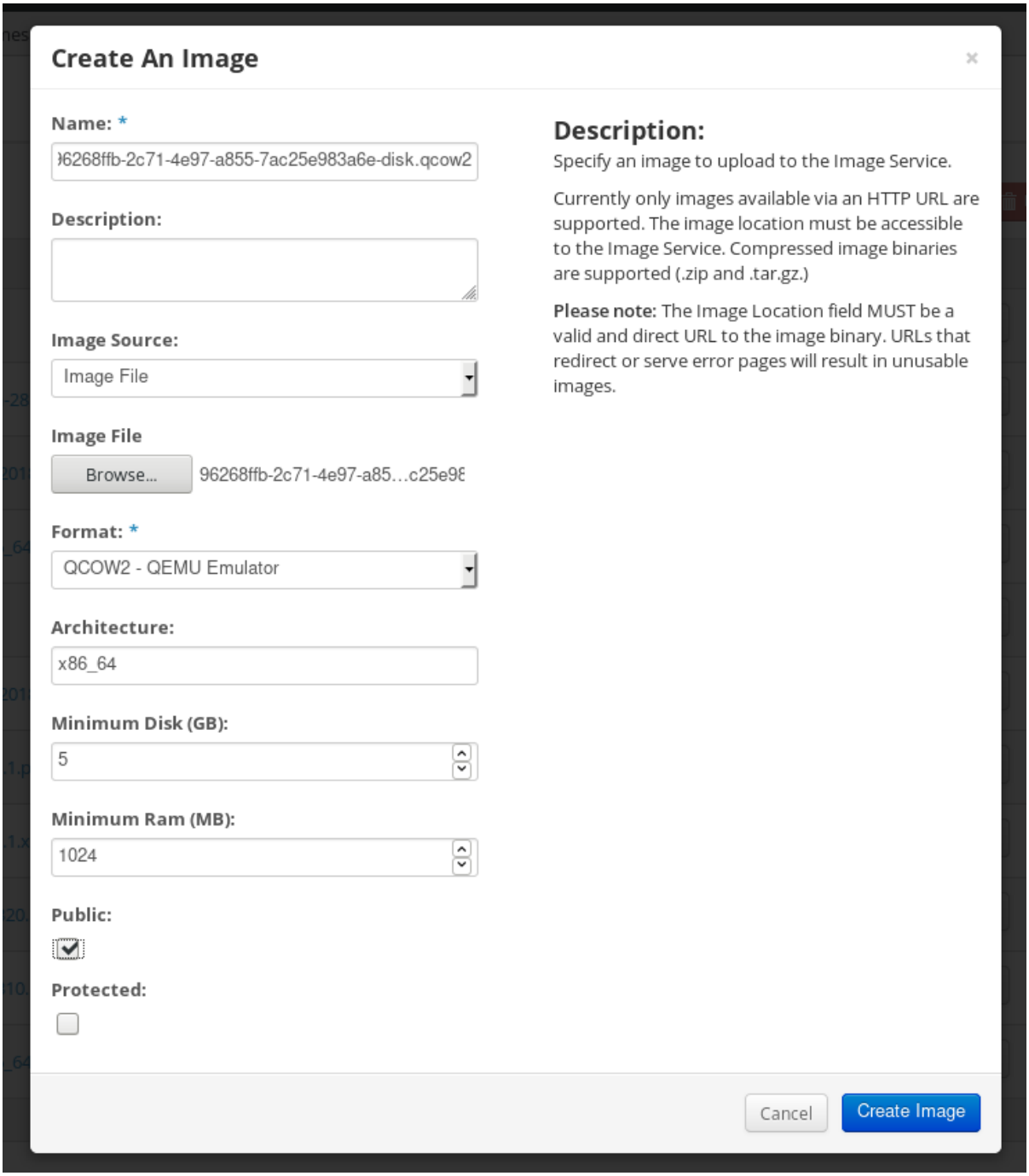
- You must have an OpenStack-specific image created by Image Builder. Use the **openstack** output type in CLI or **OpenStack Image (.qcow2)** in GUI when creating the image.

WARNING

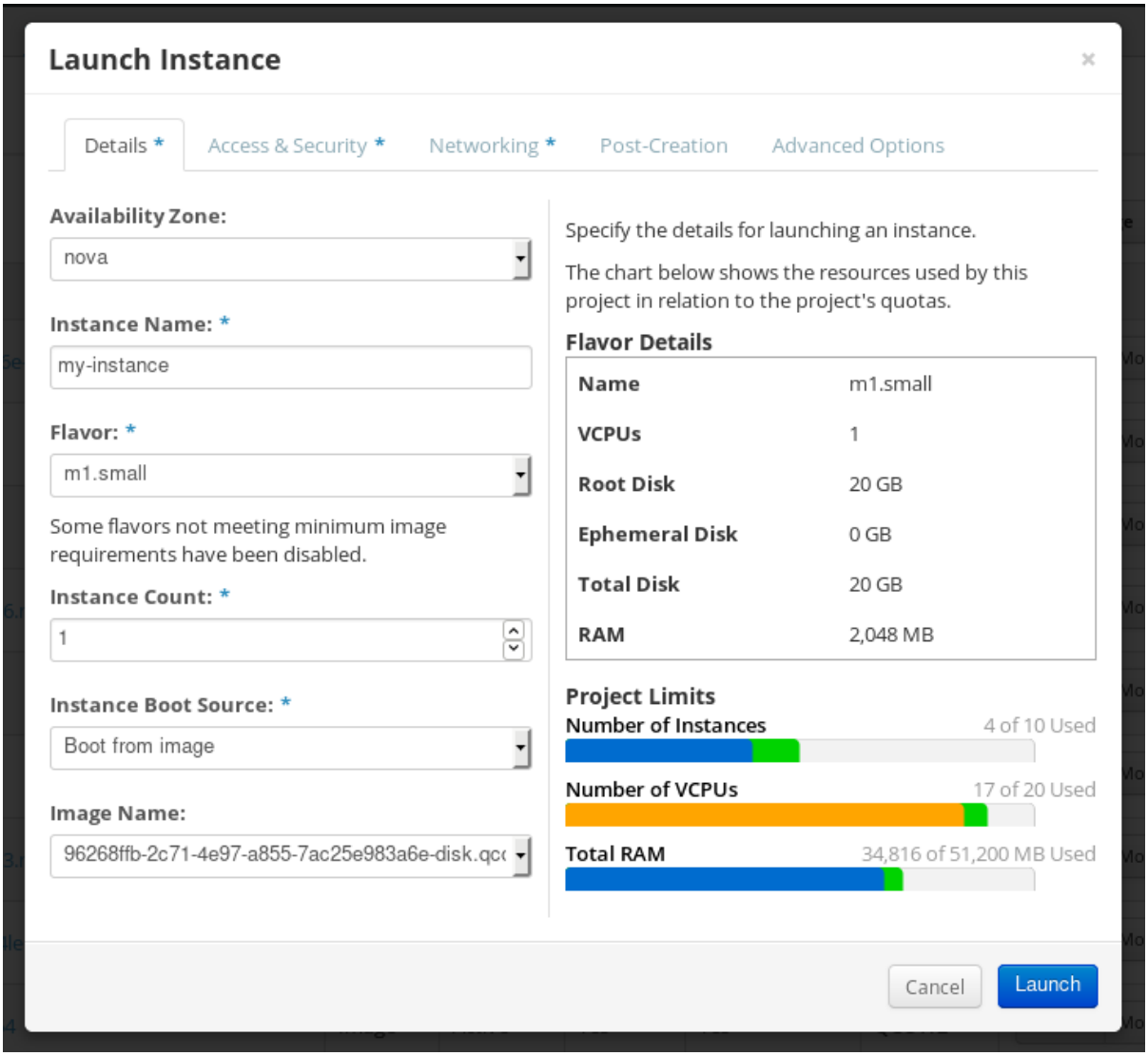
Image Builder also offers a generic QCOW2 image type output format as **qcow2** or **QEMU QCOW2 Image (.qcow2)**. Do not mistake it with the OpenStack image type which is also in the QCOW2 format, but contains further changes specific to OpenStack.

Procedure

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



2. Start an instance with that image:



3. 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**.

5.7. PREPARING FOR UPLOADING IMAGES TO ALIBABA

NOTE

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

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**.

Prerequisites

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

Procedure

1. 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 generate 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

- For more details, see Image Compliance Tool.

5.8. 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 **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

- For more details on uploading custom images to Alibaba Cloud, see Upload an object.
- For more details on creating instances from custom images, see Creating an instance from custom images.
- For more details on importing custom images to Alibaba, see Importing images.

5.9. 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 on **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

- For more details on importing custom images to Alibaba Cloud, see Notes for importing images.
- For more details on creating instances from custom images, see Creating an instance from custom images.
- For more details on creating instances from custom images, see Upload an object.

5.10. CREATING AN INSTANCE OF A CUSTOM IMAGE USING ALIBABA

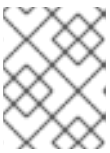
You can create instances of the custom image using Alibaba ECS Console.

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 Instace**, depending on your subscription.

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

Additional resources

- For further details on creating an instance, see Creating an instance by using a custom image .
- For more details on providing details when creating an instance, see Create an instance by using the wizard.