



Red Hat Enterprise Linux 8

Composición de una imagen de sistema RHEL personalizada

Creación de imágenes de sistema personalizadas con Image Builder en Red Hat Enterprise Linux 8

Red Hat Enterprise Linux 8 Composición de una imagen de sistema RHEL personalizada

Creación de imágenes de sistema personalizadas con Image Builder en Red Hat Enterprise Linux 8

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

Legal Notice

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Composing_a_customized_RHEL_system_image.ent file |.

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.

Resumen

Image Builder es una herramienta para crear imágenes del sistema personalizadas y listas para el despliegue: discos de instalación, máquinas virtuales, imágenes específicas del proveedor de la nube y otras. Image Builder le permite crear estas imágenes más rápidamente en comparación con los procedimientos manuales, ya que abstrae las particularidades de cada tipo de salida. Aprenda a configurar Image Builder y a crear imágenes con él.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	3
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	4
CAPÍTULO 1. IMAGE BUILDER DESCRIPTION	5
1.1. INTRODUCTION TO IMAGE BUILDER	5
1.2. IMAGE BUILDER TERMINOLOGY	5
1.3. IMAGE BUILDER OUTPUT FORMATS	5
1.4. IMAGE BUILDER SYSTEM REQUIREMENTS	6
CAPÍTULO 2. INSTALLING IMAGE BUILDER	7
2.1. INSTALLING IMAGE BUILDER IN A VIRTUAL MACHINE	7
2.2. REVERTING TO LORAX-COMPOSER IMAGE BUILDER BACKEND	8
CAPÍTULO 3. CREATING SYSTEM IMAGES WITH IMAGE BUILDER COMMAND-LINE INTERFACE	9
3.1. IMAGE BUILDER COMMAND-LINE INTERFACE	9
3.2. CREATING AN IMAGE BUILDER BLUEPRINT WITH COMMAND-LINE INTERFACE	9
3.3. EDITING AN IMAGE BUILDER BLUEPRINT WITH COMMAND-LINE INTERFACE	10
3.4. CREATING A SYSTEM IMAGE WITH IMAGE BUILDER IN THE COMMAND-LINE INTERFACE	11
3.5. BASIC IMAGE BUILDER COMMAND-LINE COMMANDS	12
3.6. IMAGE BUILDER BLUEPRINT FORMAT	13
3.7. SUPPORTED IMAGE CUSTOMIZATIONS	14
3.8. INSTALLED PACKAGES	18
3.9. ENABLED SERVICES	19
3.10. DISKS AND PARTITIONS CONFIGURATION USING IMAGE BUILDER	20
CAPÍTULO 4. CREATING SYSTEM IMAGES WITH IMAGE BUILDER WEB CONSOLE INTERFACE	21
4.1. ACCESSING IMAGE BUILDER GUI IN THE RHEL 8 WEB CONSOLE	21
4.2. CREATING AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE	21
4.3. EDITING AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE	22
4.4. ADDING USERS AND GROUPS TO AN IMAGE BUILDER BLUEPRINT IN THE WEB CONSOLE INTERFACE	24
4.5. CREATING A SYSTEM IMAGE WITH IMAGE BUILDER IN THE WEB CONSOLE INTERFACE	26
4.6. ADDING A SOURCE TO A BLUEPRINT	27
4.7. CREATING A USER ACCOUNT FOR A BLUEPRINT	28
4.8. CREATING A USER ACCOUNT WITH SSH KEY	30
CAPÍTULO 5. PREPARING AND UPLOADING CLOUD IMAGES WITH IMAGE BUILDER	33
5.1. PREPARING FOR UPLOADING AWS AMI IMAGES	33
5.2. UPLOADING AN AMI IMAGE TO AWS	34
5.3. PUSHING IMAGES TO AWS CLOUD AMI	35
5.4. PREPARING FOR UPLOADING AZURE VHD IMAGES	38
5.5. UPLOADING VHD IMAGES TO AZURE	39
5.6. UPLOADING VMDK IMAGES TO VSPHERE	40
5.7. PUSHING VHD IMAGES TO AZURE CLOUD	42
5.8. UPLOADING QCOW2 IMAGE TO OPENSTACK	45
5.9. PREPARING FOR UPLOADING IMAGES TO ALIBABA	47
5.10. UPLOADING IMAGES TO ALIBABA	48
5.11. IMPORTING IMAGES TO ALIBABA	49
5.12. CREATING AN INSTANCE OF A CUSTOM IMAGE USING ALIBABA	50

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 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**.

CAPÍTULO 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.

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 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.

Tabla 1.1. Image Builder output formats

Description	CLI name	file extension
QEMU QCOW2 Image	qcow2	.qcow2
TAR Archive	tar	.tar

Description	CLI name	file extension
Amazon Machine Image Disk	ami	.ami
Azure Disk Image	vhd	.vhd
VMware Virtual Machine Disk	vmdk	.vmdk
Openstack	openstack	.qcow2
RHEL for Edge	rhel-edge-commit	.tar

1.4. IMAGE BUILDER SYSTEM REQUIREMENTS

The **osbuild-composer** 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.

Tabla 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



NOTA

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

CAPÍTULO 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:

- `osbuild-composer`
- `composer-cli`
- `cockpit-composer`
- `bash-completion`

```
# yum 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 cockpit.socket
```

The **osbuild-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
```



IMPORTANTE

The **osbuild-composer** 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** 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**.

2.2. REVERTING TO LORAX-COMPOSER IMAGE BUILDER BACKEND

The **osbuild-composer** backend, though much more extensible, does not currently achieve feature parity with the previous **lorax-composer** backend.

To revert to the previous backend, follow the steps:

Prerequisites

- You have installed the **osbuild-composer** package

Procedure

1. Remove the **osbuild-composer** backend.

```
# yum remove osbuild-composer
```

2. In the **/etc/yum.conf** file, add an exclude entry for **osbuild-composer** package.

```
# cat /etc/yum.conf
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
skip_if_unavailable=False
exclude=osbuild-composer
```

3. Install the "lorax-composer" package.

```
# yum install lorax-composer
```

Additional resources

- [Create a Case at Red Hat Support](#) .

CAPÍTULO 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 [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
```



NOTA

You are able to create images using the **composer-cli** command as non-root. To do so, add your user to the **weldr** or **root** groups. To add your user to the **weldr** group, perform the following steps:

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

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.



NOTA

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.

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
```

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
```

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.



NOTA

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

Procedure

1. Set the image hostname:

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

2. 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
```



IMPORTANTE

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.

3. Group specifications for the resulting system image:

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

Repeat this block for every group to include.

4. Set an existing users ssh key:

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



NOTA

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.

5. Append a kernel boot parameter option to the defaults:

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

6. Set the image host name:

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

7. Add a group for the resulting system image:

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

Only the name is required and GID is optional.

8. 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.

9. 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.

10. 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.

11. 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.



NOTA

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

12. 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.

13. 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.



NOTA

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. By default, Image Builder uses the **core yum** group.

Tabla 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	polycoreutils, 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

Image type	Default Packages
openstack.ks	kernel, selinux-policy-targeted
partitioned-disk.ks	kernel, selinux-policy-targeted
qcow2.ks	kernel, selinux-policy-targeted
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



NOTA

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](#)

3.9. ENABLED SERVICES

When you configure the custom image, the services enabled are the defaults services for the RHEL8 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.

Tabla 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

Image type	Enabled Services
live-iso.ks	NetworkManager
openstack.ks	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
partitioned-disk.ks	No default service
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, vmttoolsd

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](#)

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.

CAPÍTULO 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.

Additional resources

- [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:

○ **Available Components**

- 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 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** 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.



NOTA

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.



NOTA

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
```

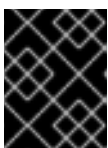
Provide your credentials when asked. Note that the terminal does not reuse your credentials you entered when logging into the web console.

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.



IMPORTANTE

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
```

- 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

- [Image Builder blueprint format](#)
- [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.

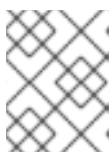
Prerequisites

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

Procedure

- 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.
- On the right side of the blueprint, press the **Create Image** button that belongs to the blueprint. A pop-up window appears.
- 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.
- Click the name of the blueprint.
A screen with details of the blueprint opens.
- Click the **Images** tab to switch to it. The image that is being created is listed with the status **In Progress**.



NOTA

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.

- 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.

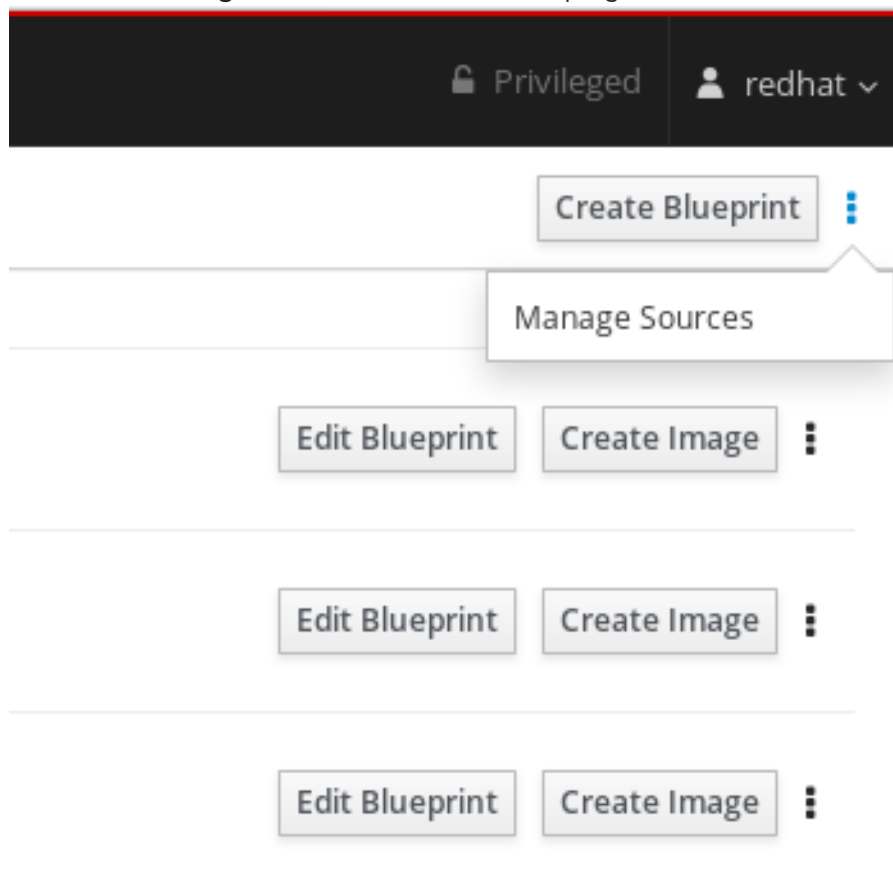
The following steps describe how to add a Source to your local system.

Prerequisites

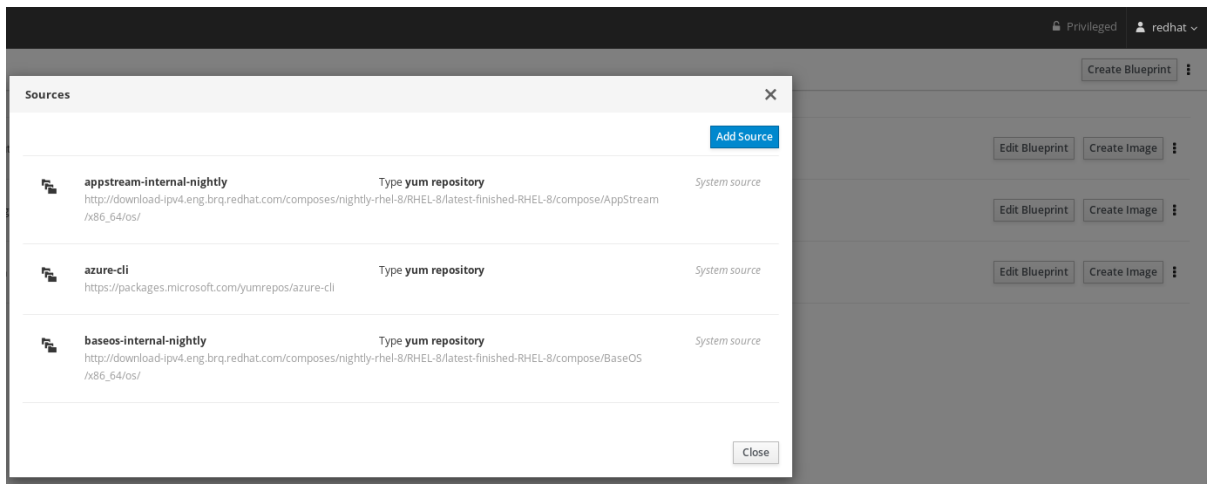
- You have opened the Image Builder interface of the RHEL 8 web console in a browser.

Procedure

- 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

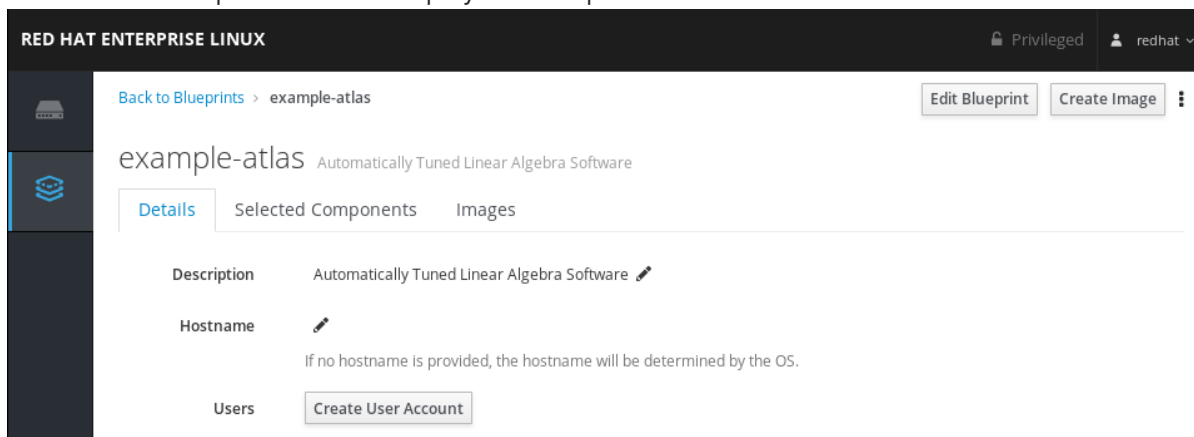
- You have opened the Image Builder interface of the RHEL 8 web console in a browser.
- You have an existing blueprint.

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**.

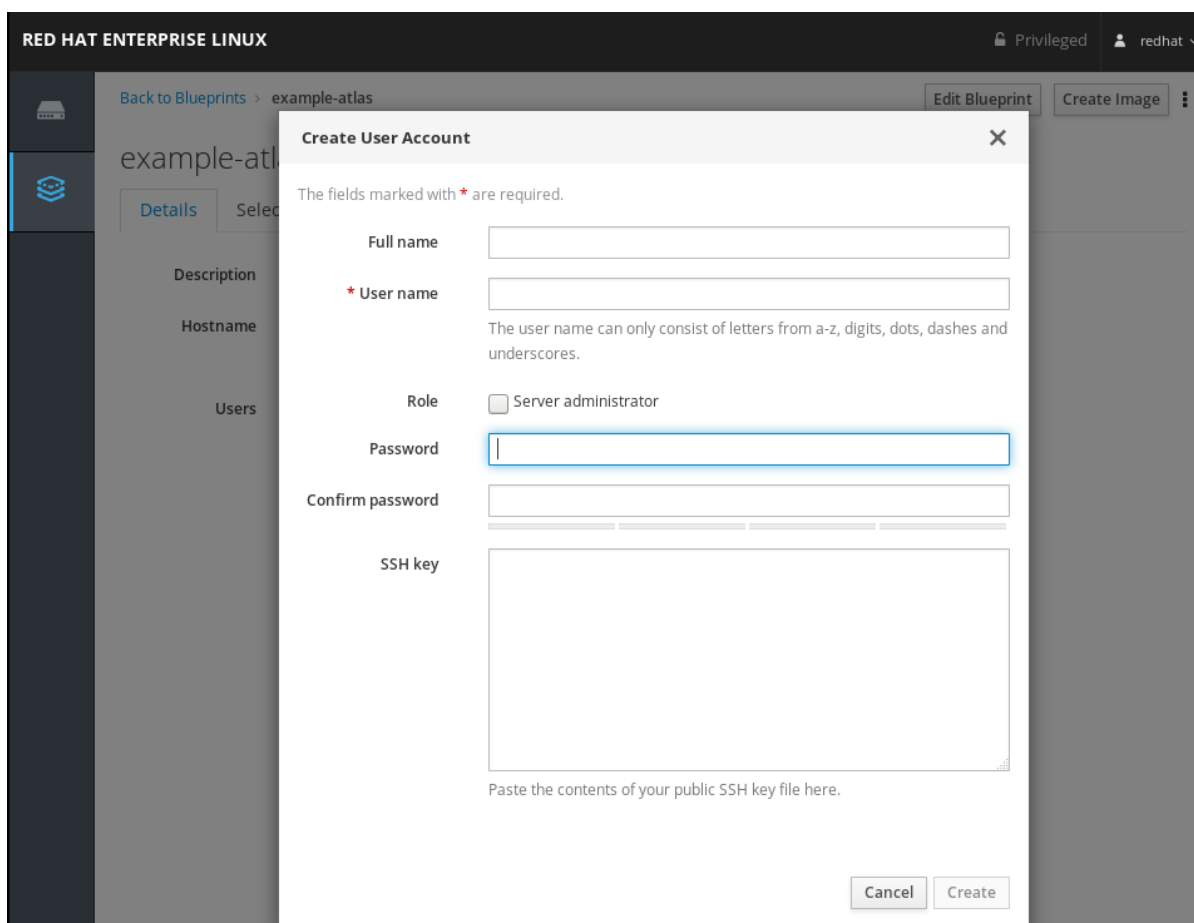
The search is added to the list of filters under the text entry field, and the list of blueprints below is reduced to those that match the search.

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

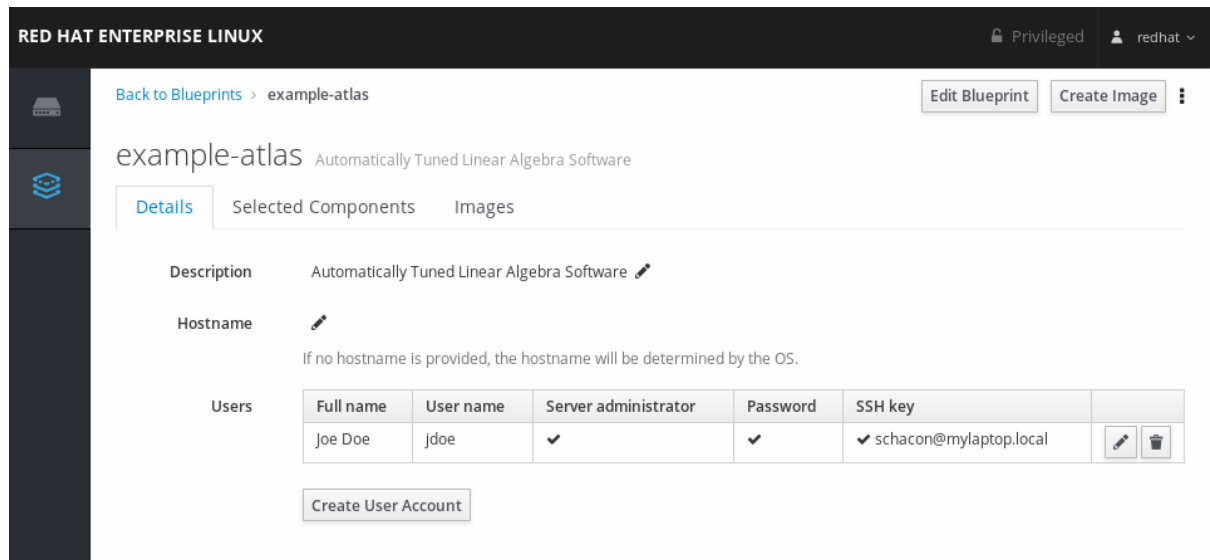


3. Click **Create User Account**.

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

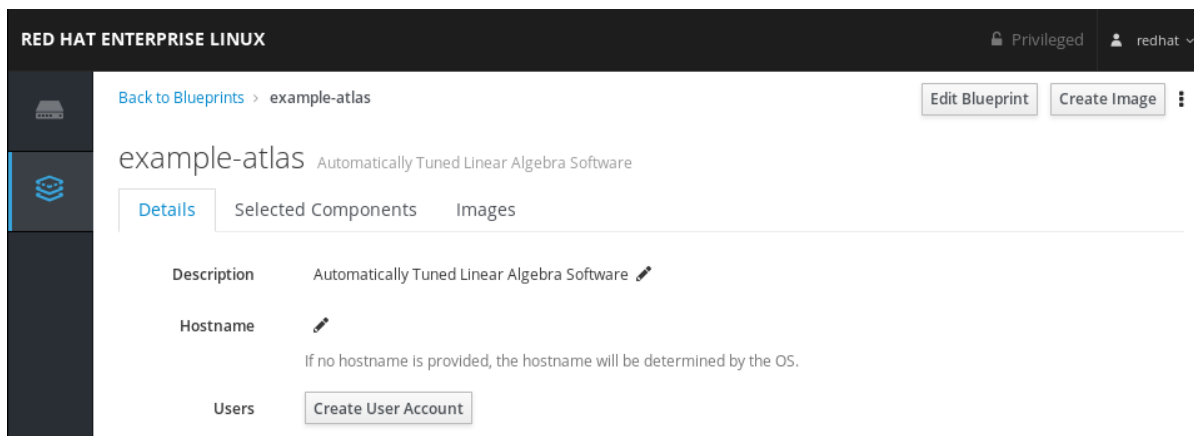
The images created by Image Builder have the root account locked and no other accounts included. Such configuration is provided in order to ensure that images are secure, by not having a default password. Image Builder enables you to create a user account with SSH key for a blueprint so that you can 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

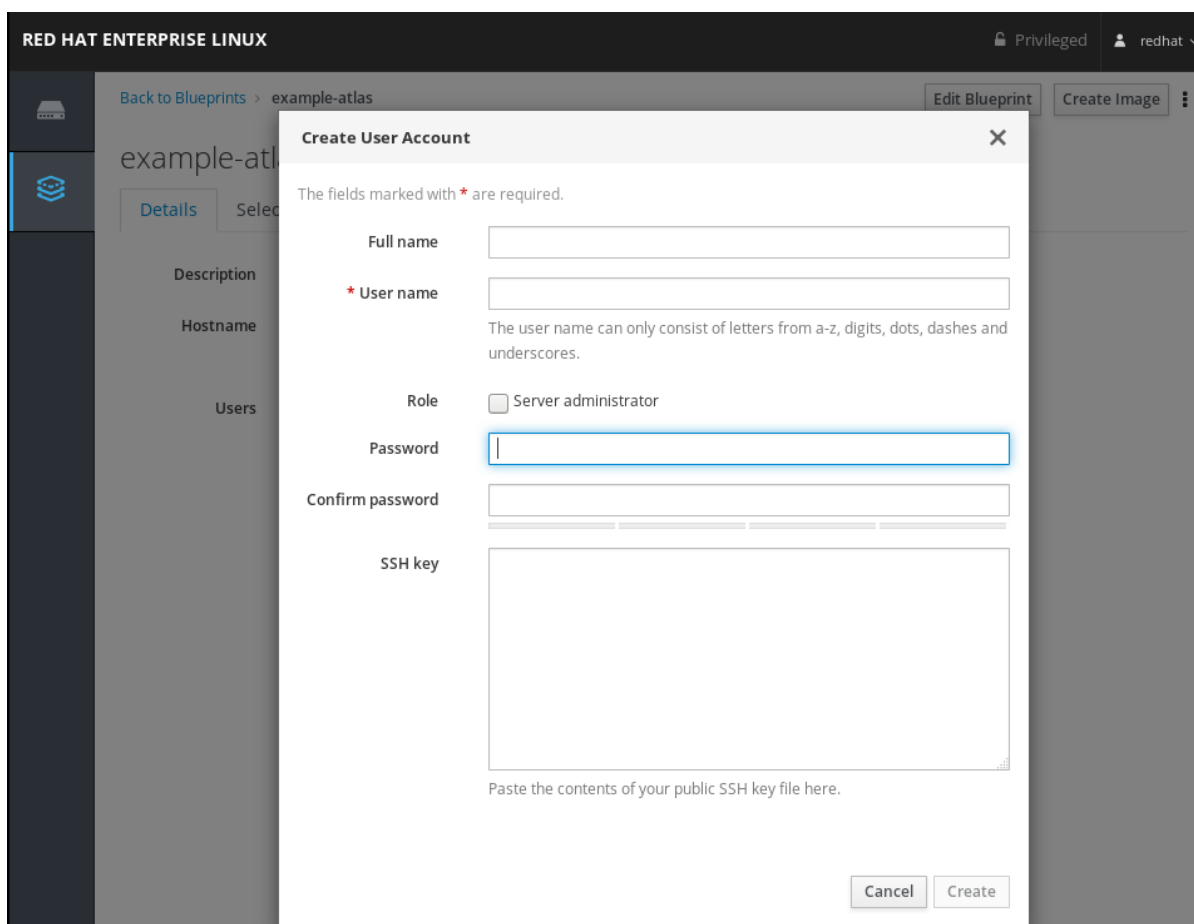
Procedure

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.

The screenshot shows the Red Hat Enterprise Linux 8 web interface. The top navigation bar includes 'RED HAT ENTERPRISE LINUX', 'Privileged', and 'redhat'. The main content area is titled 'example-atlas' with the subtitle 'Automatically Tuned Linear Algebra Software'. There are buttons for 'Edit Blueprint' and 'Create Image'. Below the title, there are tabs for 'Details', 'Selected Components', and 'Images'. The 'Details' tab is active, showing the following configuration:

- Description:** Automatically Tuned Linear Algebra Software
- Hostname:** (empty field) with a note: 'If no hostname is provided, the hostname will be determined by the OS.'
- Users:** A table with the following data:

Full name	User name	Server administrator	Password	SSH key	
Joe Doe	jdoe	✓	✓	✓ schacon@mylaptop.local	

Below the table is a 'Create User Account' button.

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

Additional resources

- [Using SSH Keys](#)

CAPÍTULO 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. From Red Hat Enterprise Linux 8.3, the ability to push customized images 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 imaged to Azure cloud](#).

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. 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](#)

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:

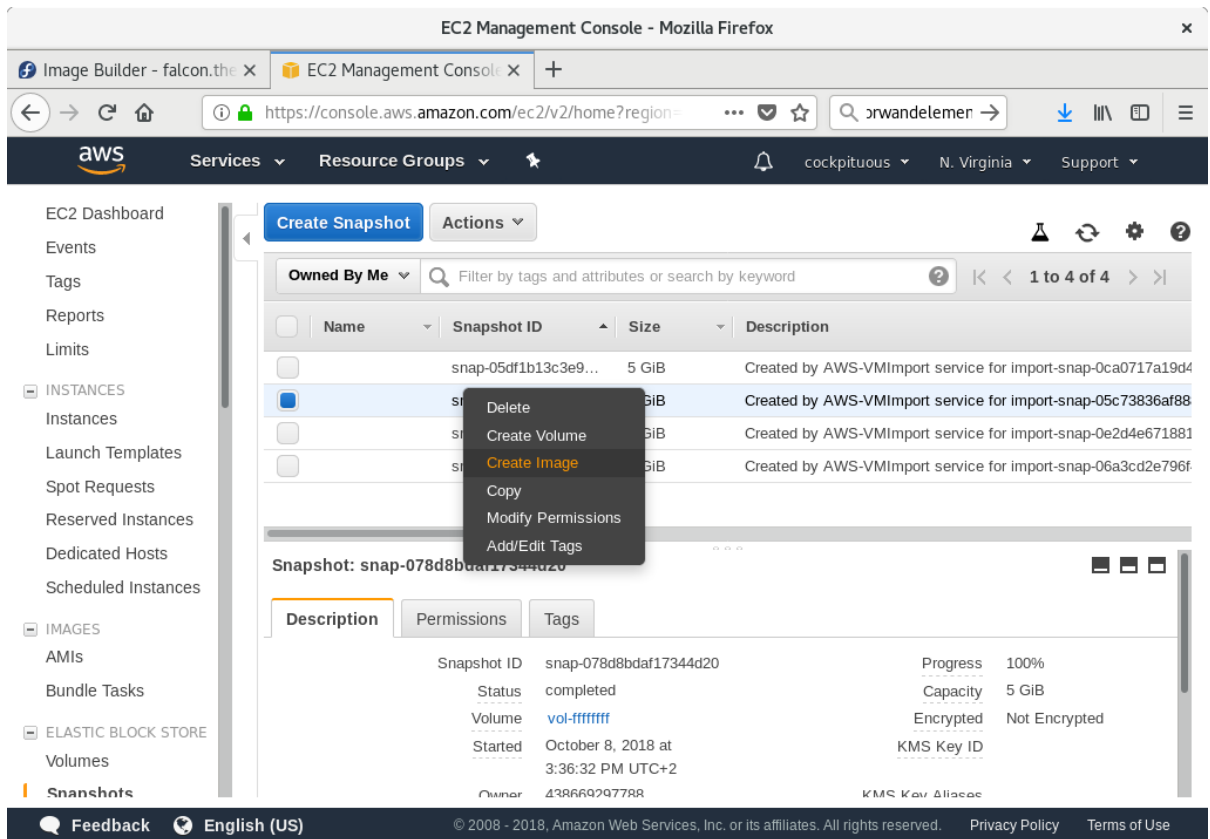
```
$ printf '{ "Description": "my-image", "Format": "raw", "UserBucket": { "S3Bucket": "%s",
"S3Key": "%s" } }' $BUCKET $AMI > containers.json
$ aws ec2 import-snapshot --disk-container file://containers.json
```

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

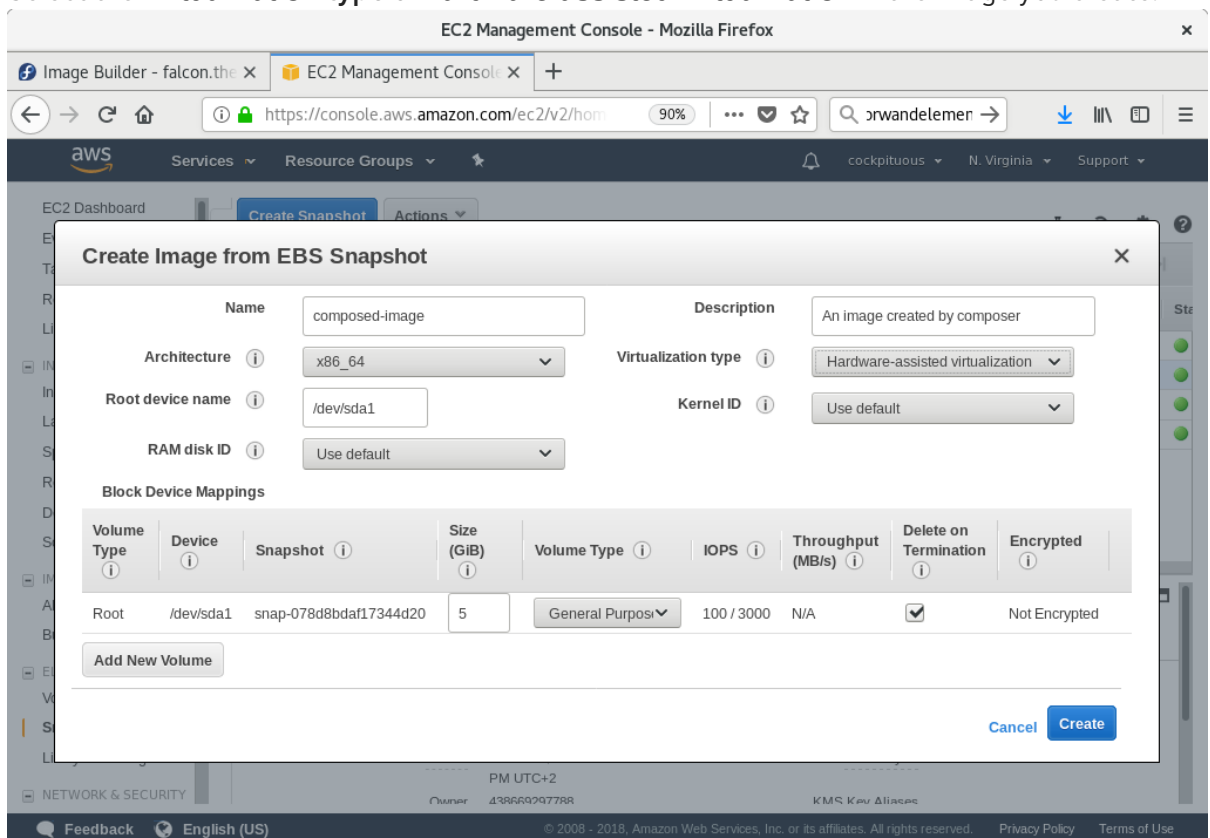
To track progress of the import, run:

```
$ 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. 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 8 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**.



NOTA

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 you provided and once you are satisfied, click **Finish**.
Optionally, you can click **Back** to modify any incorrect detail.



NOTA

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 setup 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.
Once 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. Once the instance status is **"running"**, the **Connect** button turns 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](#)

5.4. 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:

```
# yum install python2
```



NOTA

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:

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



NOTA

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

- 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.



NOTA

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.

- 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.



NOTA

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](#)

5.5. 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.

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 ...
```

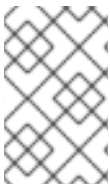
3. Create an instance either with the Azure portal, or a command similar to the following:

```
$ 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**.

5.6. 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.



NOTA

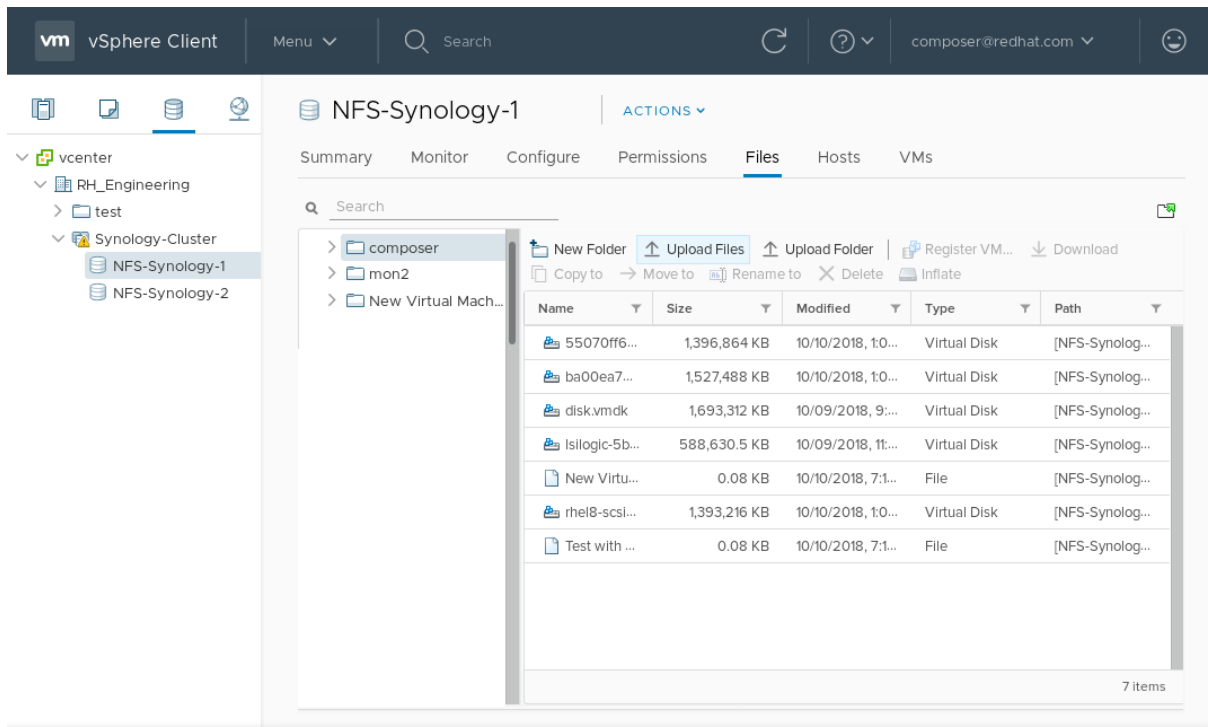
Because VMWare deployments typically does not have cloud-init configured to inject user credentials to virtual machines, we must perform that task ourselves on the blueprint.

Prerequisites

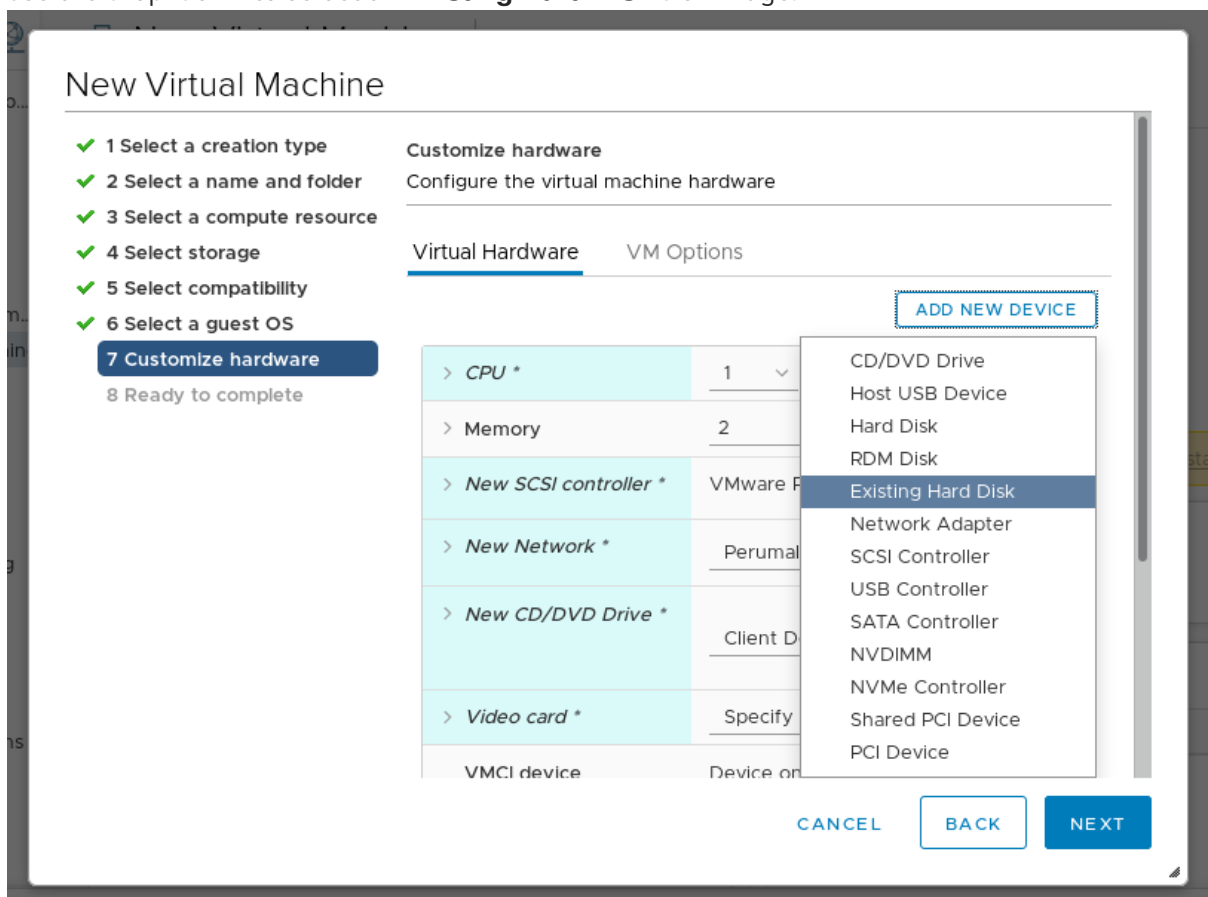
- You must have an VMDK image created by Image Builder. Use the **vmdk** output type in CLI or **VMware Virtual Machine Disk (.vmdk)** in GUI when creating the image.

Procedure

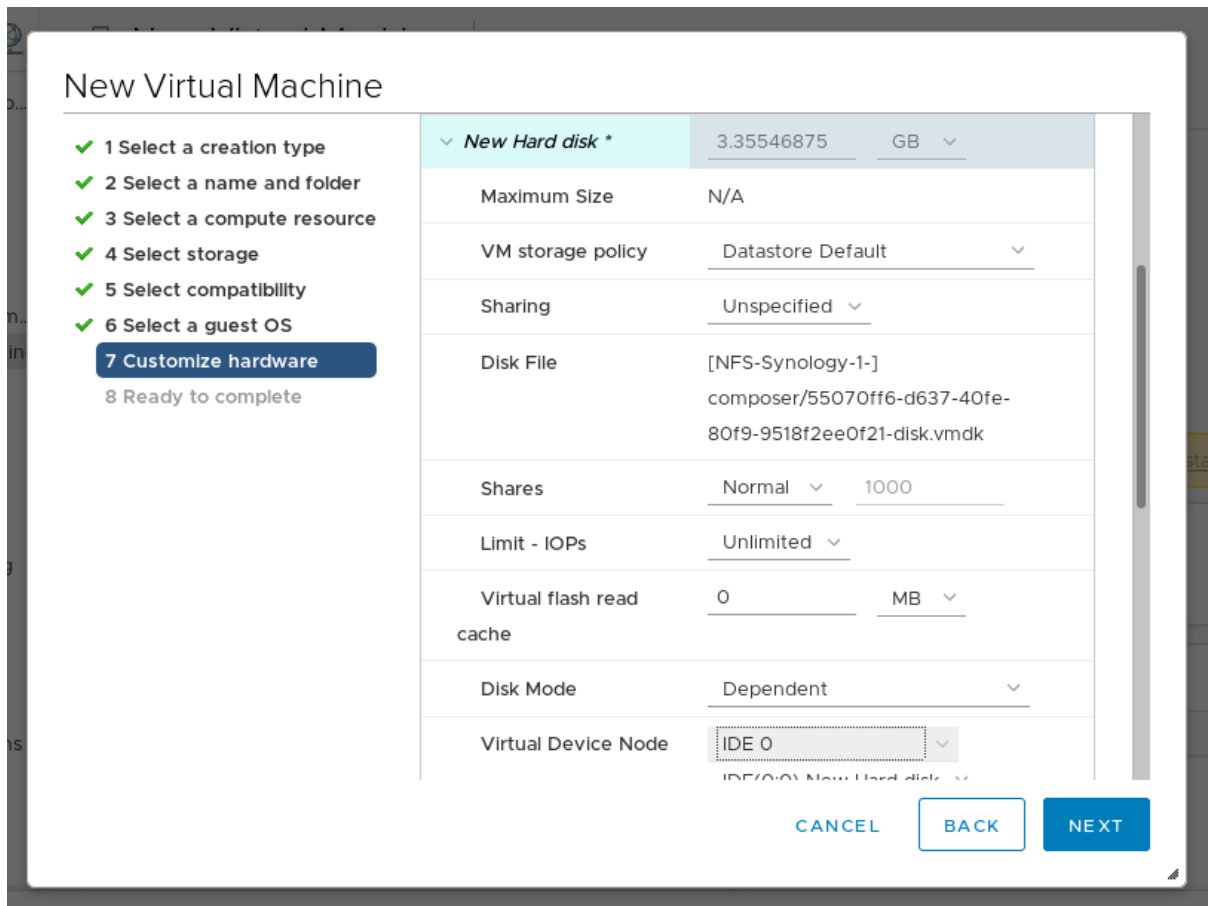
1. Upload the image into vSphere via HTTP. Click on **Upload Files** in the vCenter:



- 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.



5.7. PUSHING VHD IMAGES TO AZURE CLOUD

The ability to push the output image you create to the Azure Blob Storage is available. This section describes steps to push **.vhd** images you create using Image Builder to Azure Cloud service provider.

Prerequisites

- You must have root access to the system.
- You have opened the Image Builder interface of the RHEL 8 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 more at [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 once you are satisfied, 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".
Once 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 defaults choice.
11. Click **Create** to create the image. Once 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 are 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 defaults 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. Once 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
```

- 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](#).

5.8. 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 a 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.



AVISO

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

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**.

5.9. 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**.



NOTA

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 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

- [Image Compliance Tool](#)

5.10. 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

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

5.11. 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**.



NOTA

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](#)

5.12. CREATING AN INSTANCE OF A CUSTOM IMAGE USING ALIBABA

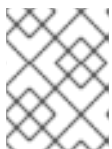
You can create instances of the custom image using Alibaba ECS Console.

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.



NOTA

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](#)

