Red Hat Enterprise Linux 9.0 Beta

Composing a customized RHEL system image

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

Last Updated: 2021-11-15

Red Hat Enterprise Linux 9.0 Beta Composing a customized RHEL system image

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

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

RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.
- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.

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**.

CHAPTER 1. IMAGE BUILDER DESCRIPTION

1.1. INTRODUCTION TO IMAGE BUILDER

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

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

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

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

1.2. IMAGE BUILDER TERMINOLOGY

Blueprint

Blueprints define customized system images by listing packages and customizations that will be part of the system. Blueprints can be edited and they are versioned. When a system image is created from a blueprint, the image is associated with the blueprint in the Image Builder interface of the RHEL web console.
Blueprints are presented to the user as plain text in the 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**
TAR Archive	**tar**	**.tar**

Description	CLI name	file extension
Amazon Machine Image Disk	**ami**	**.raw**
Azure Disk Image	**vhd**	**.vhd**
VMware Virtual Machine Disk	**vmdk**	**.vmdk**
Openstack	**openstack**	**.qcow2**
RHEL for Edge Commit	**rhel-edge-commit**	**.tar**
RHEL for Edge Container	**rhel-edge-container**	**.tar**
RHEL for Edge Installer	**rhel-edge-installer**	**.iso**

1.4. IMAGE BUILDER SYSTEM REQUIREMENTS

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

Table 1.2. Image Builder system requirements

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

NOTE

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

CHAPTER 2. INSTALLING IMAGE BUILDER

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

2.1. INSTALLING IMAGE BUILDER IN A VIRTUAL MACHINE

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

Prerequisites

- Connect to the virtual machine.
- The virtual machine for Image Builder must be installed, subscribed, and running.

Procedure

1. Install the Image Builder and other necessary packages on the virtual machine:
   - **osbuild-composer** - supported from RHEL 8.3 onward
   - **composer-cli**
   - **cockpit-composer**
   - **bash-completion**

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

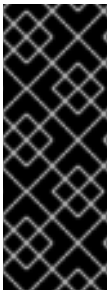
   The web console is installed as a dependency of the *cockpit-composer* package.
2. Enable Image Builder to start after each reboot:

   ```
   # 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
   ```



IMPORTANT

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

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 .

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 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
```

NOTE

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.

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.

   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**.

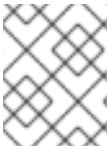
Replace *package-version* with a version to use. This field supports **dnf** version specifications:

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

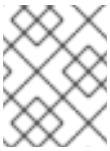
Procedure

Set the image hostname

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

User specifications for the resulting system image

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



NOTE

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

IMPORTANT

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

Define a kernel name to be used in an image

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

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

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

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

Set the locale settings for the resulting system image

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

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

Set the firewall for the resulting system image

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

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

Customize the firewall services

Review the available firewall services.

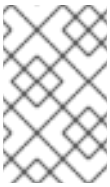
```
$ firewall-cmd --get-services
```

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

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

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

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



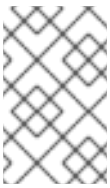
NOTE

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

Set which services to enable during the boot time

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

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



NOTE

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

Specify a custom filesystem configuration

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

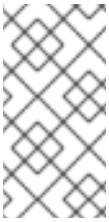
- security benchmark compliance
- protection against out-of-disk errors

- performance
- consistency with existing setups
  Customize the filesystem configuration in your blueprint:

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

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

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



NOTE

Customizing mountpoints is only supported in the RHEL8.5 and RHEL 9.0 distributions, using the CLI. In early distributions, you can only specify the **root** partition as a mountpoint and specify the **size** argument as an alias for the image size.

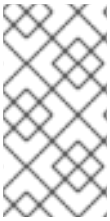
3.8. INSTALLED PACKAGES

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.

Table 3.1. Default packages to support image type creation

Image type	Default Packages
ami	checkpolicy, chrony, cloud-init, cloud-utils-growpart, @Core, dhcp-client, gdisk, insights-client, kernel, langpacks-en, net-tools, NetworkManager, redhat-release, redhat-release-eula, rng-tools, rsync, selinux-policy-targeted, tar, Yum-utils
openstack	@Core, langpacks-en

Image type	Default Packages
qcow2	@Core, chrony, dnf, kernel, yum, nfs-utils, dnf-utils, cloud-init, python3-jsonschema, qemu-guest-agent, cloud-utils-growpart, dracut-norescue, tar, tcpdump, rsync, dnf-plugin-spacewalk, rhn-client-tools, rhnlib, rhnsd, rhn-setup, NetworkManager, dhcp-client, cockpit-ws, cockpit-system, subscription-manager-cockpit, redhat-release, redhat-release-eula, rng-tools, insights-client
rhel-edge-commit	glibc, glibc-minimal-langpack, nss-altfiles, kernel, dracut-config-generic, dracut-network, basesystem, bash, platform-python,shadow-utils, chrony, setup, shadow-utils, sudo, systemd, coreutils, util-linux, curl, vim-minimal, rpm, rpm-ostree, polkit, lvm2, cryptsetup, pinentry, e2fsprogs, dosfstools, keyutils, gnupg2, attr, xz, gzip, firewalld, iptables, NetworkManager, NetworkManager-wifi, NetworkManager-wwan, wpa_supplicant, dnsmasq, traceroute, hostname, iproute, iputils, openssh-clients, procps-ng, rootfiles, openssh-server, passwd, policycoreutils, policycoreutils-python-utils, selinux-policy-targeted, setools-console, less, tar, rsync, fwupd, usbguard, bash-completion, tmux, ima-evm-utils, audit, rng-tools, podman, container-selinux, skopeo, criu, slirp4netns, fuse-overlayfs, clevis, clevis-dracut, clevis-luks, greenboot, greenboot-grub2, greenboot-rpm-ostree-grub2, greenboot-reboot, greenboot-status
tar	policycoreutils, selinux-policy-targeted
vhd	@Core, langpacks-en
vmdk	@Core, chrony, firewalld, kernel, langpacks-en, open-vm-tools, selinux-policy-targeted



NOTE

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

Additional resources

- Image Builder description

3.9. ENABLED SERVICES

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

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

Table 3.2. Enabled services to support image type creation

Image type	Enabled Services
ami	No default service
openstack	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
qcow2	No default service
rhel-edge-commit	No default service
tar	No default service
vhd	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk	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

- Supported Image Customizations

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 WEB CONSOLE

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

Prerequisites

- You must have root access to the system.

Procedure

1. Open **https://localhost:9090/** in a web browser on the system where Image Builder is installed. For more information on how to remotely access Image Builder, see *Managing systems using the RHEL web console* document.
2. Log into the web console with credentials for 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 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:

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

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

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

c. Click the name of the component you intend to use to display its details. The right pane fills with details of the components, such as its version and dependencies.

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

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

f. If you added a component by mistake, remove it by clicking the **...** button at the far right of its entry in the right pane, and select **Remove** in the menu.

NOTE

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

4. To save the blueprint, click **Commit** in the top right. A dialog with a summary of the changes pops up. Click **Commit**.
A small pop-up on the right informs you of the saving progress and then the result.

5. To exit the editing screen, click **Back to Blueprints** in the top left.
The Image Builder view opens, listing existing blueprints.

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 web console in a browser.
- A blueprint exists.

Procedure

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

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

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

The view changes to the blueprint editing screen.

3. Remove unwanted components by clicking their button at the far right of its entry in the right pane, and select **Remove** in the menu.

4. Change version of existing components:

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

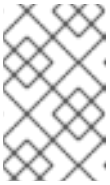
   b. The list of components is paged. To move to other result pages, use the arrows and entry field above the component list.

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

8. Exit the privileged shell:

```
# exit
```

9. Open the Image Builder (**Image builder**) tab on the left and refresh the page, in all browsers and all tabs where it was opened.
   This prevents state cached in the loaded page from accidentally reverting your changes.

Additional 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 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.

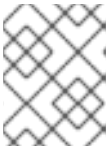
   If the list of blueprints is too long, add further search terms in the same way.

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.

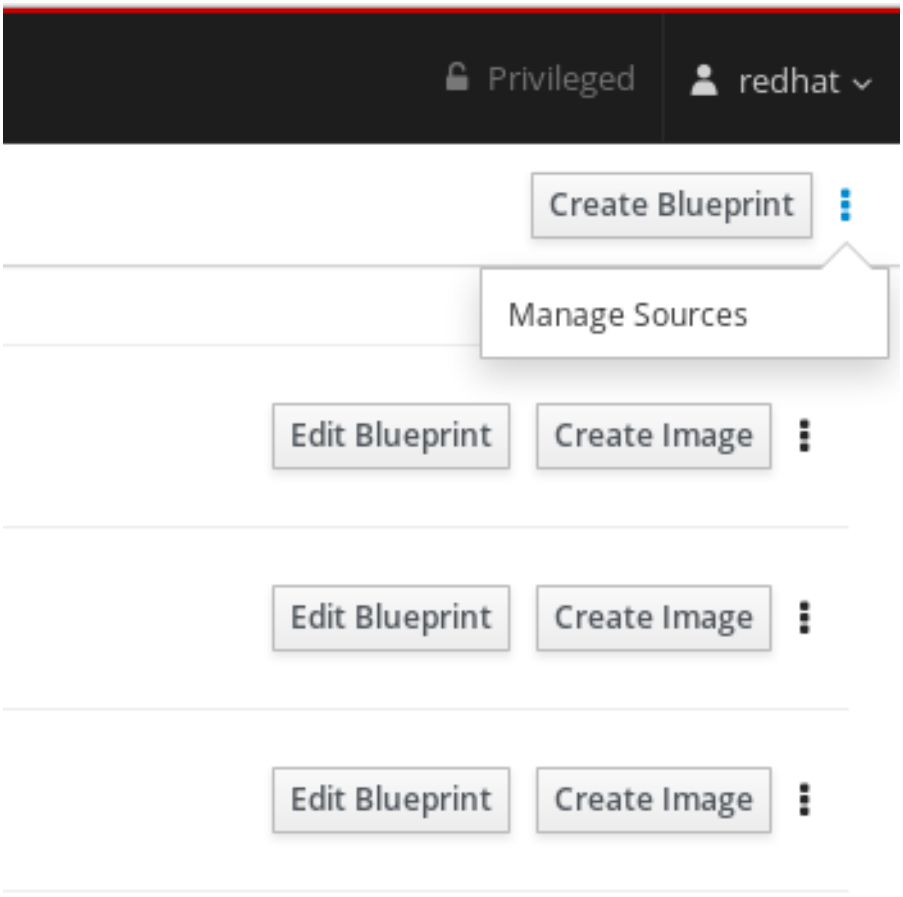
The following steps describe how to add a Source to your local system.

Prerequisites

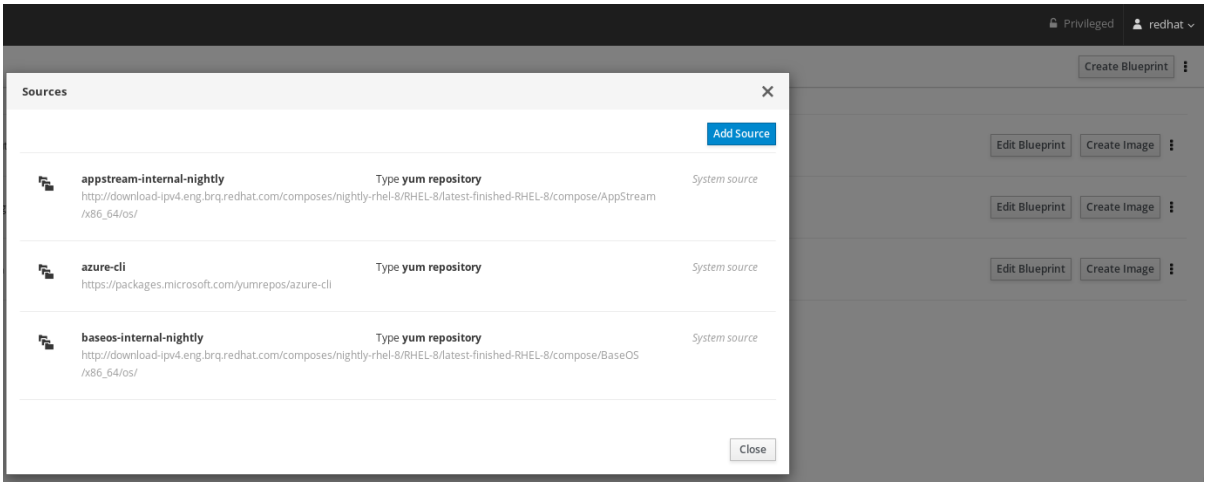
- You have opened the Image Builder interface of the RHEL web console in a browser.

Procedure

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



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



2. On the right side of the pop-up window, click the **Add Source** button.

3. Add the desired **Source name**, the **Source path**, and the **Source Type**. The **Security** field is optional.

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

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

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

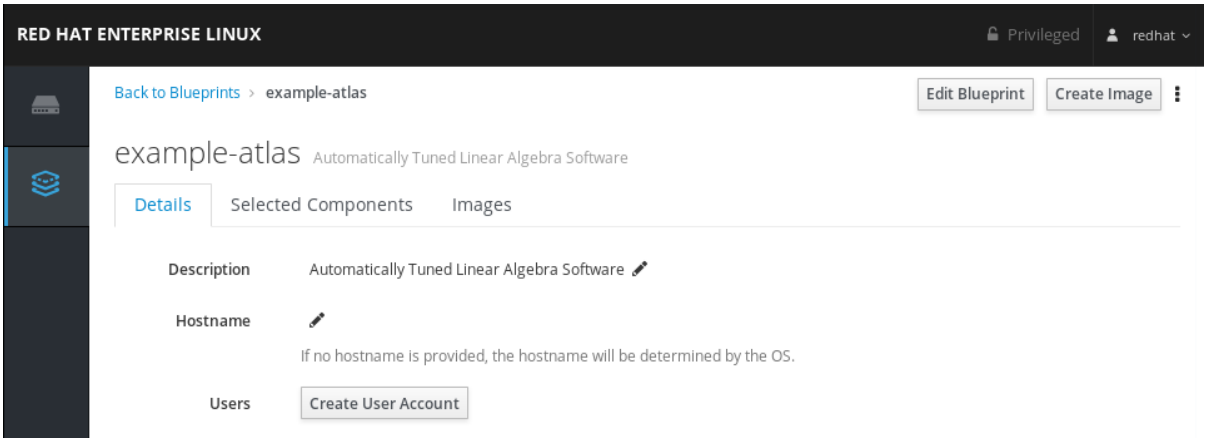
- You have opened the Image Builder interface of the RHEL web console in a browser.
- You have an existing blueprint.

Procedure

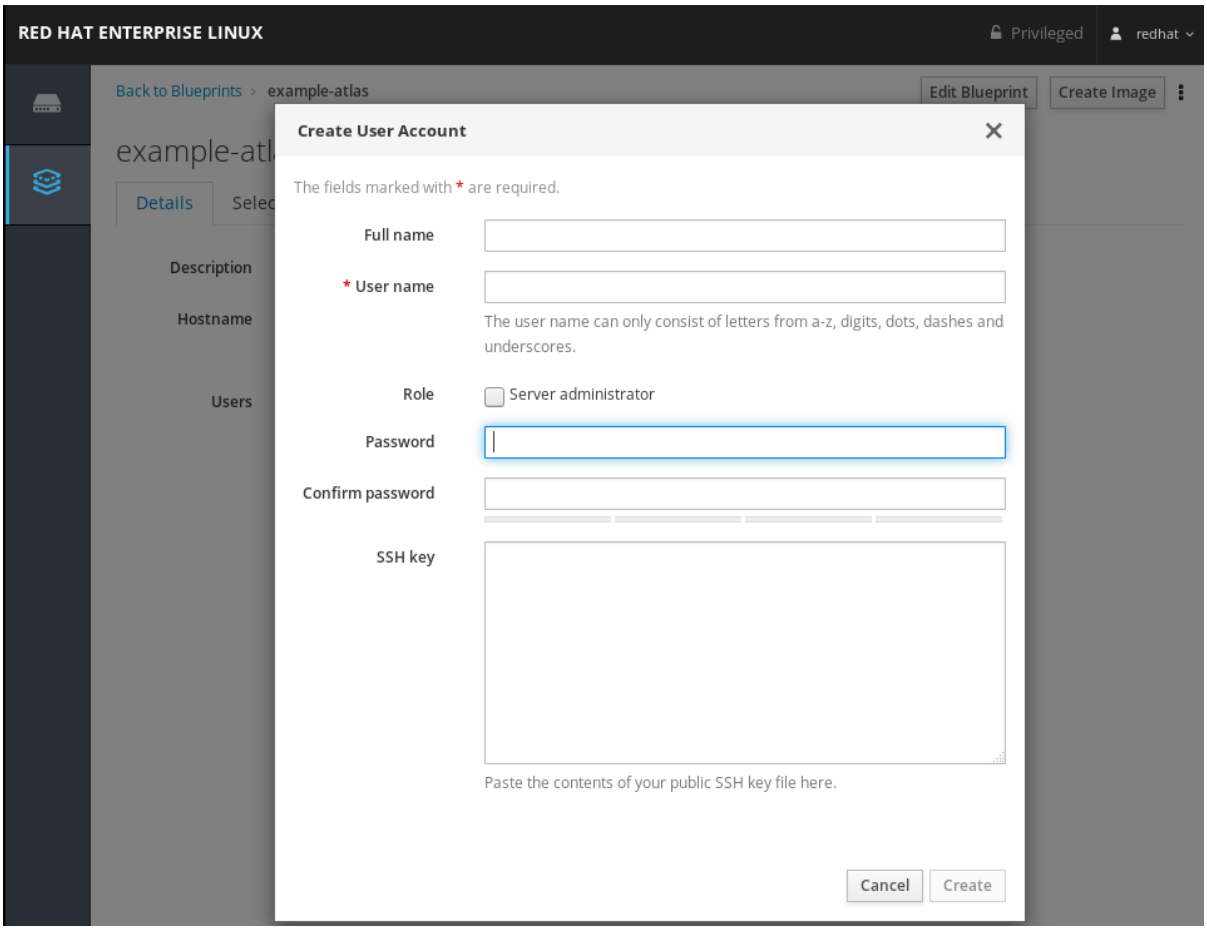
1. Locate the blueprint that you want to create a user account for by entering its name or a part of it into the search box at the top left, and press **Enter**.

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.



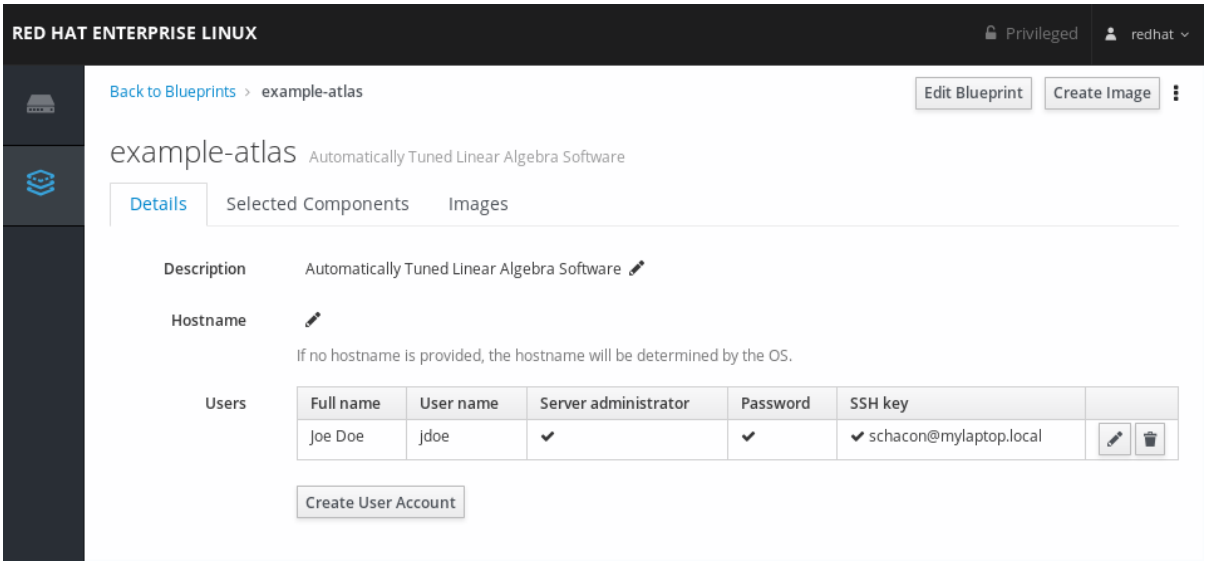
3. Click **Create User Account**.
This will open a window with fields for user account creation.



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.

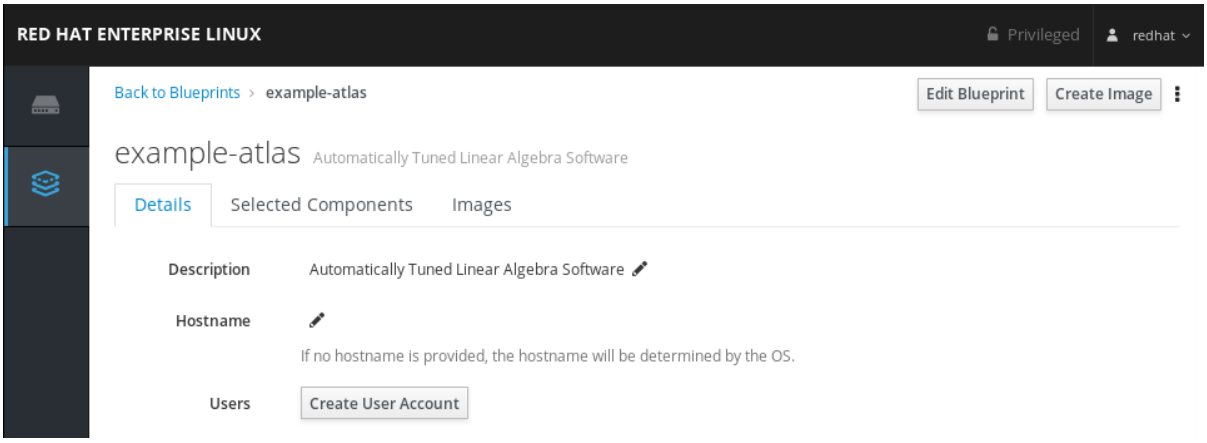
Prerequisites

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

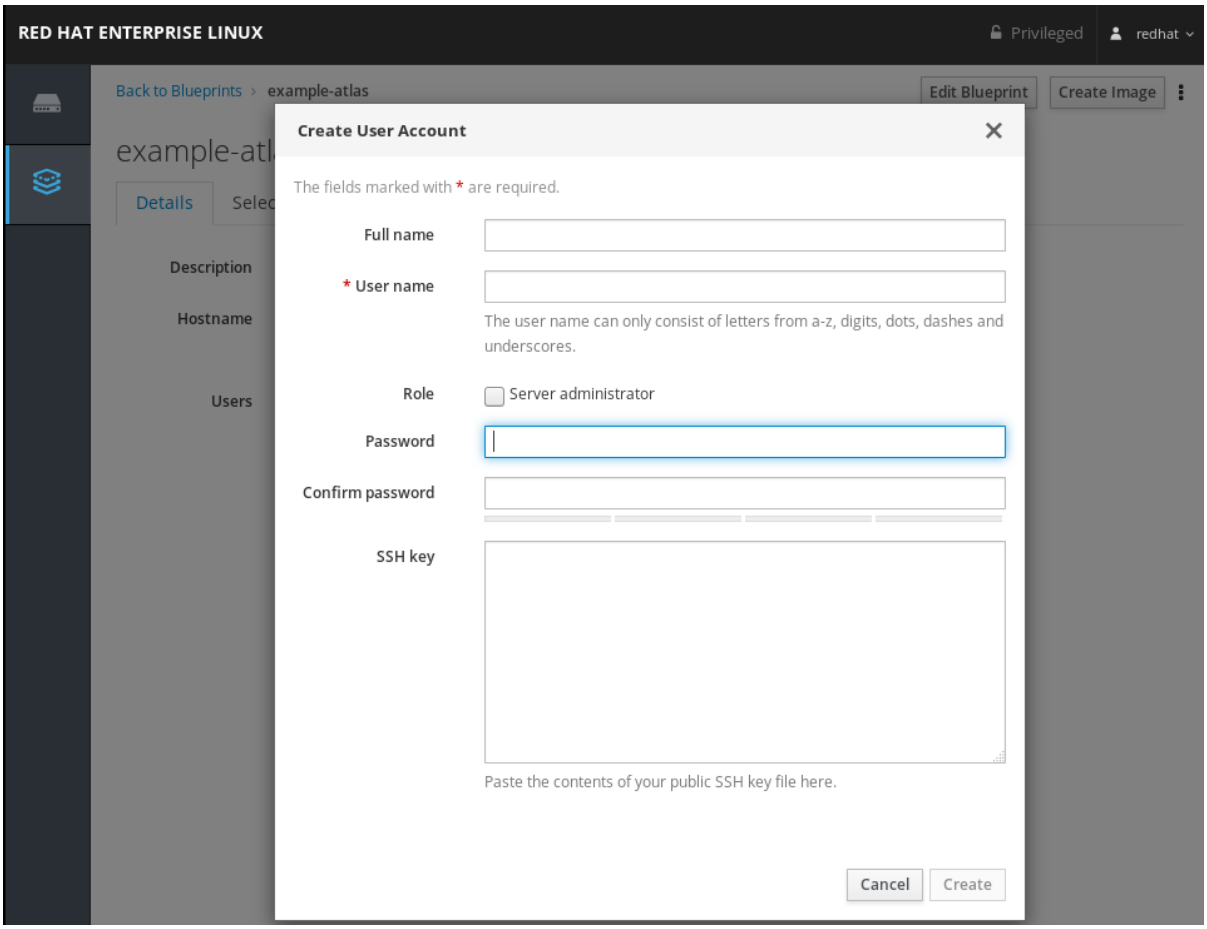
Procedure

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

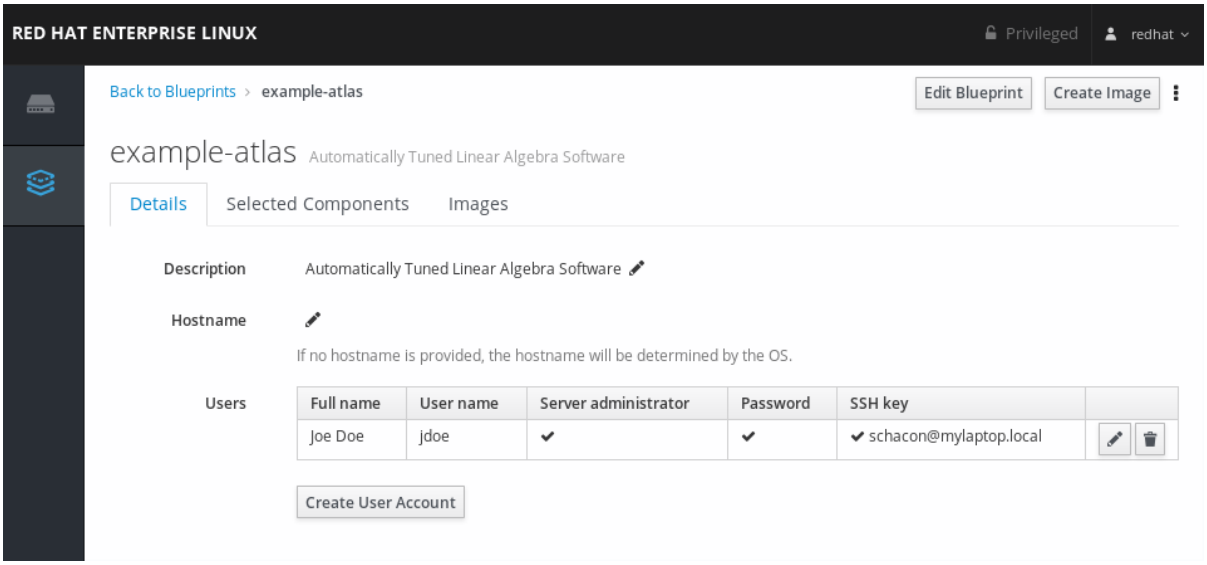


4. Fill in the details. Notice that when you insert the name, the **User name** field autocompletes, suggesting a username.
   If you want to provide administrators rights to the user account you are creating, check the **Role** field.

   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

- Generating SSH key pairs

CHAPTER 5. USING IMAGE BUILDER TO CREATE SYSTEM IMAGES FROM DIFFERENT RELEASES

You can use Image Builder to create images of multiple RHEL minor releases that are different from the host, such as RHEL 8.4 and RHEL 8.5. For that, you can add source system repositories with the release distribution fields set and also, you can create blueprints with the correct release distribution fields set.

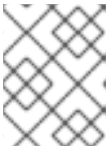
Additionally, if you have existing blueprint or source system repositories in an old format, you can create new blueprints with the correct release distribution fields set.

- To list the supported release distribution, you can run the following command:

```
$ curl --unix-socket /run/weldr/api.socket http://localhost/api/v1/distros/list
```

The output shows you a JSON string that lists the supported release distribution names:

```
{"distros":["rhel-84","rhel-85","rhel-86"]}
```



NOTE

Cross-distribution image building, such as building a CentOS image on RHEL is not supported.

5.1. CREATING AN IMAGE WITH A DIFFERENT DISTRIBUTION IN THE CLI

To select the distribution you want to use when composing an image in the CLI, you must set the new **distro** field in the blueprint. For that, follow the steps:

Procedure

If you are creating a new blueprint

1. Create a blueprint. For example:

```
name = "blueprint_84"
description = "A 8.5 base image"
version = "0.0.1"
modules = []
groups = []
distro = "rhel-84"
```

By attributing “rhel-84” to the **distro** field, you ensure that it always builds a RHEL 8.2 image, no matter which version is running on the host.

NOTE

If the **distro** field is blank, it uses the same distribution of the host.

If you are updating an existing blueprint

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

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

1. Edit the existing blueprint file with a text editor of your choice, setting the **distro** field with the distribution of your choice, for example:

```
name = "blueprint_84"
description = "A 8.4 base image"
version = "0.0.1"
modules = []
groups = []
distro = "rhel-84"
```

2. Save the file and close the editor.

3. Push (import) the blueprint back into Image Builder:

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

4. Start the image creation:

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

Wait until the compose is finished. Notice that this may take several minutes.

5. Check the status of the compose:

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

Once the compose finishes, it shows a FINISHED status value. Identify the compose in the list by its UUID.

6. Download the resulting image file:

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

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

5.2. USING SYSTEM REPOSITORIES WITH SPECIFIC DISTRIBUTIONS

You can specify a list of distribution strings that the system repository source uses when depsolving and building images. For that, see the following example:

Procedure

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

```
check_gpg = true
check_ssl = true
distros = ["rhel-84"]
id = "rhel-84-local"
name = "local packages for rhel-84"
```

```
system = false
type = "yum-baseurl"
url = "http://local/repos/rhel-84/projectrepo/"
```

Additional resources

- For more details on overriding repositories, see Managing repositories.

CHAPTER 6. MANAGING REPOSITORIES

6.1. IMAGE BUILDER DEFAULT SYSTEM REPOSITORIES

The **osbuild-composer** backend does not inherit the system repositories located in the **/etc/yum.repos.d/** directory. Instead, it has its own set of official repositories defined in the **/usr/share/osbuild-composer/repositories** directory. To override the official repositories, you must define overrides in **/etc/osbuild-composer/repositories**. This directory is for user defined overrides and the files located here take precedence over those in the **/usr** directory.

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

6.2. OVERRIDING A SYSTEM REPOSITORY

You can configure a repository override in the **/etc/osbuild-composer/repositories** directory by following these steps. NOTE: Prior to RHEL 8.5 release, the name of the repository overrides is rhel-8.json. Starting from RHEL 8.5, the names also respect the minor version: rhel-84.json, rhel-85.json, and so on. .Prerequisites

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

Procedure

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

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

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

```
{
   "<ARCH>": [
      {
         "name": "baseos",
         "metalink": "",
         "baseurl": "http://mirror.example.com/composes/released/RHEL-
8/8.2.0/BaseOS/x86_64/os/",
         "mirrorlist": "",
         "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (...)",
         "check_gpg": true,
         "metadata_expire": "",
      }
   ]
}
```

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

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

```
/etc/osbuild-composer/repositories/rhel-84.json
/etc/osbuild-composer/repositories/rhel-85.json
/etc/osbuild-composer/repositories/rhel-90.json
```

Alternatively, you can copy the file for your distribution from **/usr/share/osbuild-composer/** and modify its content.

4. Copy the repository file to the directory you created.

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

Replace rhel-version.json with your RHEL version, for example: rhel-85.json

5. Using the editor of your choice, edit the **baseurl** paths in the **rhel-85.json** file. For example:

```
$ vi etc/osbuild-composer/repositories/rhel-85.json
```

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

6.3. OVERRIDING A SYSTEM REPOSITORY WITH SUPPORT FOR SUBSCRIPTIONS

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

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

Prerequisites

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

Procedure

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

```
[AppStream]
name = AppStream mirror example
baseurl = https://mirror.example.com/RHEL-8/8.5.0/AppStream/x86_64/os/
enabled = 1
gpgcheck = 0
sslverify = 1
sslcacert = /etc/pki/ca1/ca.crt
sslclientkey = /etc/pki/ca1/client.key
sslclientcert = /etc/pki/ca1/client.crt
metadata_expire = 86400
enabled_metadata = 0
```

2. Configure the repository override to use the same **baseurl** and set **rhsm** to true:

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

```
        {
            "name": "AppStream mirror example",
            "baseurl": "https://mirror.example.com/RHEL-8/8.5.0/AppStream/x86_64/os/",
            "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (...)",
            "check_gpg": true,
            “rhsm": true
        }
    ]
}
```

NOTE

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

CHAPTER 7. CREATING A BOOT ISO INSTALLER IMAGE WITH IMAGE BUILDER

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

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

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

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

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

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

Prerequisites

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

Procedure

1. Create the ISO image:

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

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

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

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

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

3. 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.

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

Verification

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

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

Additional resources

- Creating system images with Image Builder command-line interface
- Creating a bootable installation medium for RHEL

7.2. INSTALLING THE ISO IMAGE TO A BARE METAL SYSTEM

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

Prerequisites

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

NOTE

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

Procedure

1. Place the bootable ISO image file on a USB flash drive.
2. Connect the USB flash drive to the port of the computer you want to boot.
3. Boot the ISO image from the USB flash drive.
4. Perform the steps to install the customized bootable ISO image.
   The boot screen shows you the following options:
   - Install Red Hat Enterprise Linux 9
   - Test this media & install Red Hat Enterprise Linux 9

Additional resources

- Booting the installation