Table of Contents

RHEL 8 Readiness Training

Image Builder (Composer)

Course: CEE-RL-804
Version: 2.0, April 2019

How to use this module:

- Look for gray < and > marks on either the bottom or the left and right sides of this pane, depending on the size of the window. Click those to navigate to the previous or next page, respectively.
- Jump to a specific page using the navigation links at the left.
- Play audio for a page using the player at the top of that page. Audio often provides more complete information than the text and graphics alone. A transcript is available from a link on the same page.

Objectives

On completing this training, you should be able to:

- Define Image Builder in Red Hat Enterprise Linux 8 and list some of its features
- Install and configure Image Builder and its web console plugin in RHEL 8
- Use Image Builder in the web console to modify packages selected for building images
- Customize a blueprint for post-installation tasks like adding users and SSH keys
- Identify and investigate common issues with Image Builder

Prerequisites

This training assumes that you are a Red Hat Certified System Administrator (RHCSA) or that you have equivalent experience with RHEL.

▼ Show transcript

Welcome to this training, RHEL 8 Readiness Training: Image Builder.

On completing this training, you should be able to define Image Builder in Red Hat Enterprise Linux 8 and list some of its features. You should also be able to install and configure Image Builder and its web console plugin in RHEL 8, and to use Image Builder in the web console to modify packages selected for building images. You should be able to customize a blueprint for post-installation tasks like adding users and SSH keys. In addition, you should be able to identify and investigate common issues with Image Builder.

This training assumes that you are a Red Hat Certified System Administrator (RHCSA), or that you have equivalent experience with RHEL.

Lab Environment

Successful completion for this training includes hands-on lab activities hosted in a cloud-based lab environment.

PROVISIONING

(1) Log in to the OpenTLC lab portal.

(2) On the far left, mouse over **Services** and select **Catalogs** from the pop-up menu.

(3) Select to expand **All Services** and **Support Labs**.

(4) Select **cee-rl-804** under that list.

(5) Select **Order**.

(6) Complete the application request: read the **Runtime Warning**, check the box to confirm the runtime and expiration dates, and select **Submit**.

IMPORTANT: Expect **up to 20 minutes** to provision your lab environment.

(7) Look for information on how to access your lab environment from one of two places:

- **Information email**
  Look for an email from *Red Hat OPENTLC <noreply@opentlc.com>* with the Subject similar to: *Your Red Hat OPENTLC service provision request for OTLC-LAB_COMPLETED has completed.* This email may arrive before the environment is ready to use. If you don't receive this email within 15 minutes, you can generate a new one from OpenTLC: **Services > Active Services > OTLC-LAB-NAME* > App Control > Status > Submit**
- **The OpenTLC UI**
  Look in the *Custom Attributes* section on the right in OpenTLC: **Services > Active Services > OTLC-LAB-*NAME***

SYSTEM INFO

System	IP	Credentials	Description
servera.example.com	172.25.250.10	root/redhat	Server to use with the web console

SSH ACCESS

(1) Use the SSH command shown here to access your environment, modifying the command based on the information you received by email:

```
$ ssh flastname-redhat.com@classroom-guid.red.osp.opentlc.com
```

(2) When prompted, log in to your lab environment using one of these options:

- A password set by OpenTLC and provided in the information email.
- An SSH key pair configured as described here: http://www.opentlc.com/ssh.html

```
$ ssh flastname-redhat.com@classroom-guid.red.osp.opentlc.com
The authenticity of host 'classroom-guid.red.osp.opentlc.com (169.47.191.199)' can't be established.
ECDSA key fingerprint is SHA256:v01n4XWXr0lphfGpBiSSvbasmrlQZul2ntS8g0Kbmdk.
Are you sure you want to continue connecting (yes/no)? yes

flastname-redhat.com@classroom-guid.red.osp.opentlc.com's password: <PASSWORD>

[flastname-redhat.com@classroom-guid ~]$ sudo su -
Last login: Thu Oct 24 14:19:41 EDT 2019 from 61.0.147.106 on pts/0
[root@classroom-guid ~]#
```

CONSOLE ACCESS

If you need console access to any of the machines in this environment, follow these steps:

(1) Retrieve the **Master Console** URL from the information email you received on provisioning your lab environment. Look for a line that's similar to this one:

```
Master Console: https://console-redvnc.apps.shared.na.openshift.opentlc.com
```

(2) Open this console URL in your web browser, and select **Log in with OpenShift**.

(3) Enter your OpenTLC username and password at the OpenShift login prompt.

(4) If a dialog appears requiring you to *Authorize Access* for a service account, choose to allow the selected permissions to continue.

(5) Select **Access Console** for a given virtual machine to open a VNC console session with that system.

LOCAL WEB BROWSER ACCESS (HOSTED WEB UI)

(1) Use the same *ssh* command from your local system as for command line access, but add the argument **-CfND 8080**

```
[user1@laptop ~]$ ssh flastname-redhat.com@classroom-guid.red.osp.opentlc.com -CfnND 8080
```

(2) Configure your local web browser to send all web traffic through **localhost:8080**.

```
[user1@laptop ~]$ google-chrome --proxy-server="socks5://127.0.0.1:8080" --host-resolver-rules="MAP * 0.0.0.0 , EXCLUDE localh
ost" &
```

▼ Show transcript

Successful completion for this training includes hands-on lab activities. Use the information on this page to launch you cloud-based lab environment, locate the URLs and credentials to access that environment, familiarize yourself with the network setup, and use SSH or a local web browser to access lab systems.

Image Builder Overview

▼ Show transcript

This section provides an overview of the Image Builder in RHEL 8, including what it is and its supported image formats.

What is Image Builder?

Image Builder is an image-building tool and live bootable image creator introduced in RHEL 7.6 and RHEL 8.

- Create custom deployable images (selected packages, post-install configuration, etc.)
- Customize images for third-party packages and updated RHEL Errata content
- Create images in a variety of formats for deployment to a variety of environments

Image Builder was called ***Composer*** through the RHEL 8 Beta.

Before Image Builder:

- Creating customized RHEL images was unsupported by Red Hat
- Clients and partners often requested the ability to customize
- Customization grew in importance for cloud environments

▼ Show transcript

What is Image Builder? It is an image-building tool and live bootable image creator introduced in RHEL 7.6 and RHEL 8. Image builder is used to create custom deployable images. This customization can include selected packages, post-installation configuration, and more. You can also customize images to have third-party packages and updated RHEL Errata content. You can create images in a variety of formats for deployment to a variety of environments.

Note that Image Builder was called "Composer" through the RHEL 8 Beta, and the name changed just prior to the GA release of RHEL 8. You will still see package names, service names, and artifacts reflecting the "composer" name.

Before Image Builder, customization meant getting media from the Customer Portal and manually applying changes. This was unsupported by Red Hat, who only supported the ISO and virtual machines images provided directly from the Customer Portal. Clients and partners often made feature requests to Red Hat seeking a supported ability to customize RHEL images for their specific needs.

The need for image customization also grew in importance with the increased use of RHEL in cloud environments. With Red Hat's growth in the cloud market came the need for Red Hat to change its approach to supporting custom RHEL images.

Features & benefits

- Provides an end user with the ability to create supported custom RHEL images according to their needs
- Reduces deployment and configuration time on public cloud services
- Supports various output image formats for various environments
- Can be used to create images for deployment in a disconnected environment
- Output images can be configured for custom repositories (diverge from the Red Hat Content Delivery Network defaults)
- Provides package selection and configuration from a user-friendly web UI in the RHEL 8 web console
- Allows users to save and alter image configuration to create multiple replicas with a few clicks

▼ Show transcript

Listed here are some of the features and benefits of Image Builder in RHEL 8. Image Builder provides an end user with the ability to create supported custom RHEL images according to their needs. As discussed in upcoming pages, Image Builder takes the names of the packages a user wants to install and creates a bootable RHEL image file that has those packages pre-installed. It is designed to save customers time in both creating custom images and deploying those images in specific environments.

Because of its features, Image Builder reduces the deployment and configuration time on public cloud services. It also supports various output image formats for various environments. For customers with secure or offline environments, Image Builder can be used to create images with the latest packages and errata for deployment in those disconnected environments.

Image Builder's output images can be configured for custom repositories that diverge from the Red Hat Content Delivery Network defaults.

Image Builder provides package selection and configuration from a user-friendly web UI in the RHEL 8 web console. See a separate training in this series for more about the RHEL 8 web console. This UI allows users to save and alter image configuration to create multiple replicas with a few clicks.

Output image formats

Image Builder allows you to build custom images in a various formats, including:

- Raw disk (.img)
- Live ISO (.iso)
- File system (.img)
- Tarball (.tar.xz)
- VMDK (VMware® vSphere® Hypervisor)
- AMI (Amazon Web Services®)
- VHD (Microsoft® Azure®)
- qcow2 for KVM, Red Hat Virtualization, Red Hat Satellite, and Red Hat CloudForms
- qcow2 for OpenStack

▼ Show transcript

Listed here are the image formats you can output from Image Builder. This includes raw disk images, live ISO images, file system images, and tarball images. Also listed are common virtual and cloud formats for specific platforms: VMDK for the VMware vSphere Hypervisor, AMI for Amazon Web Services, VHD for Microsoft Azure, and qcow2 for KVM, Red Hat Virtualization, Satellite, and CloudForms. There is also a separate qcow2 optimized for OpenStack.

All of the virtual machine images created using Image Builder are partitioned and pre-installed with the correct packages. They are also pre-configured with the correct services and include the custom configuration files that support that specific virtual or cloud environment. These images can be directly deployed to their respective virtual and cloud environments for a variety of deployment use cases.

Image Builder Interfaces

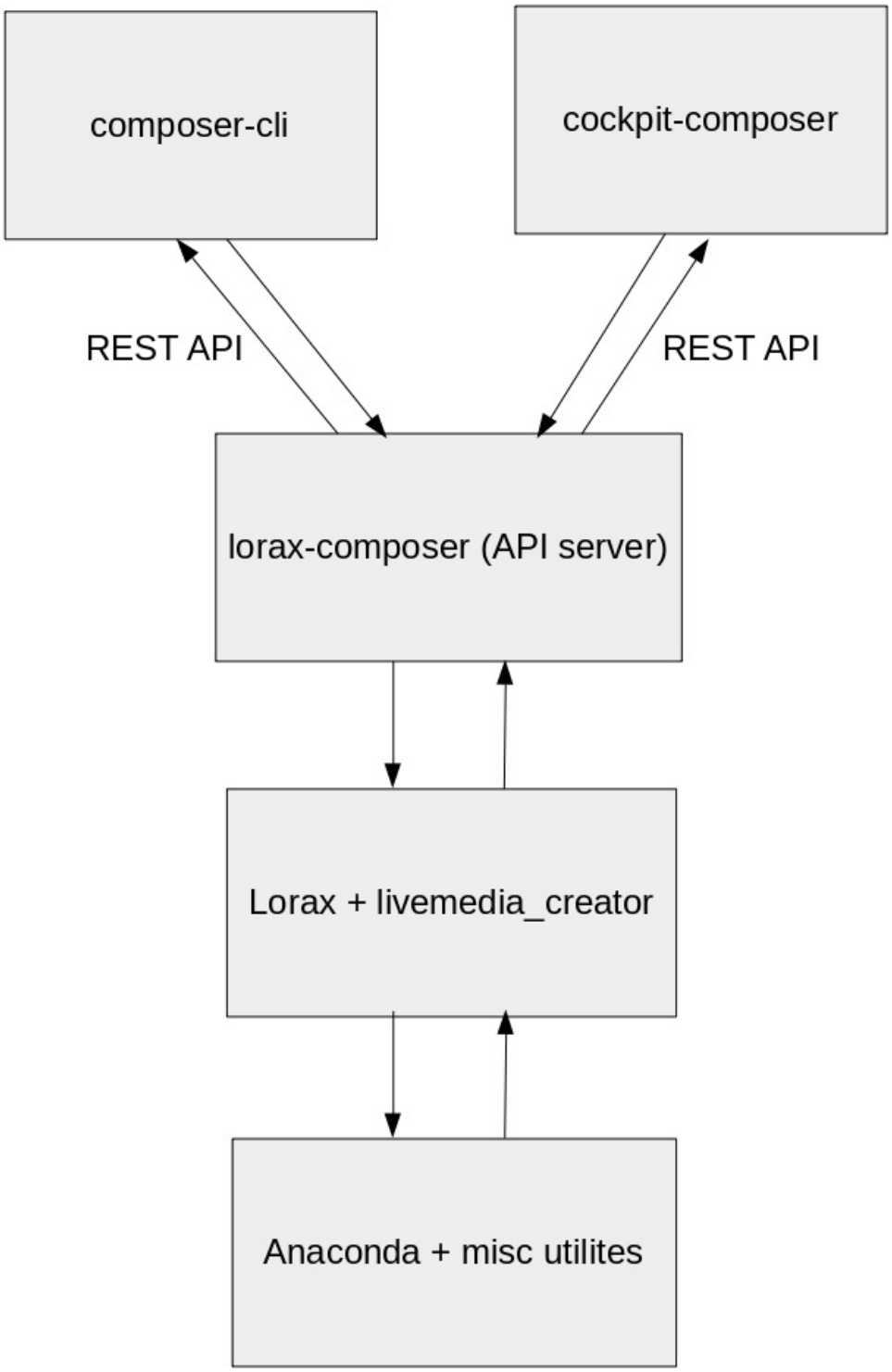
▼ Show transcript

This section covers details about the Image Builder interfaces in RHEL 8.

Image Builder API & back end

- **Image Builder is an application programming interface (API) server for building disk images.**
- Image Builder provides its functionality through API endpoints.
- Two front-end tools use this API:
  - Command-line interface (*composer-cli*)
  - Plugin for the RHEL 8 web console (*cockpit-composer*)

Behind the API: Image Builder uses the Lorax and *livemedia-creator* tool, which uses Anaconda and other utilities to create images:



▼ Show transcript

Image Builder is actually an application programming interface server for building disk images. It provides its functionality through callable API endpoints. Two front-end tools use this API: a command line interface, and a plugin for the RHEL 8 web console.

As you see here, the package names "composer-cli" and "cockpit-composer" reflect older names for both the Image Builder and the web console. It's also possible to create a custom front-end for calling these API endpoints.

Behind the API, Image Builder uses the Lorax and "livemedia-creator" tool, which, in turn, uses Anaconda and other utilities to create images. The diagram here shows how all these parts work together.

Image Builder CLI

- Command: ***composer-cli***
- Provides a command-line interface for using Image Builder
- Has some functions that are only available in the CLI (e.g. post-install configuration)

To get help on how to use *composer-cli*:

```
composer-cli -h
```

▼ Show transcript

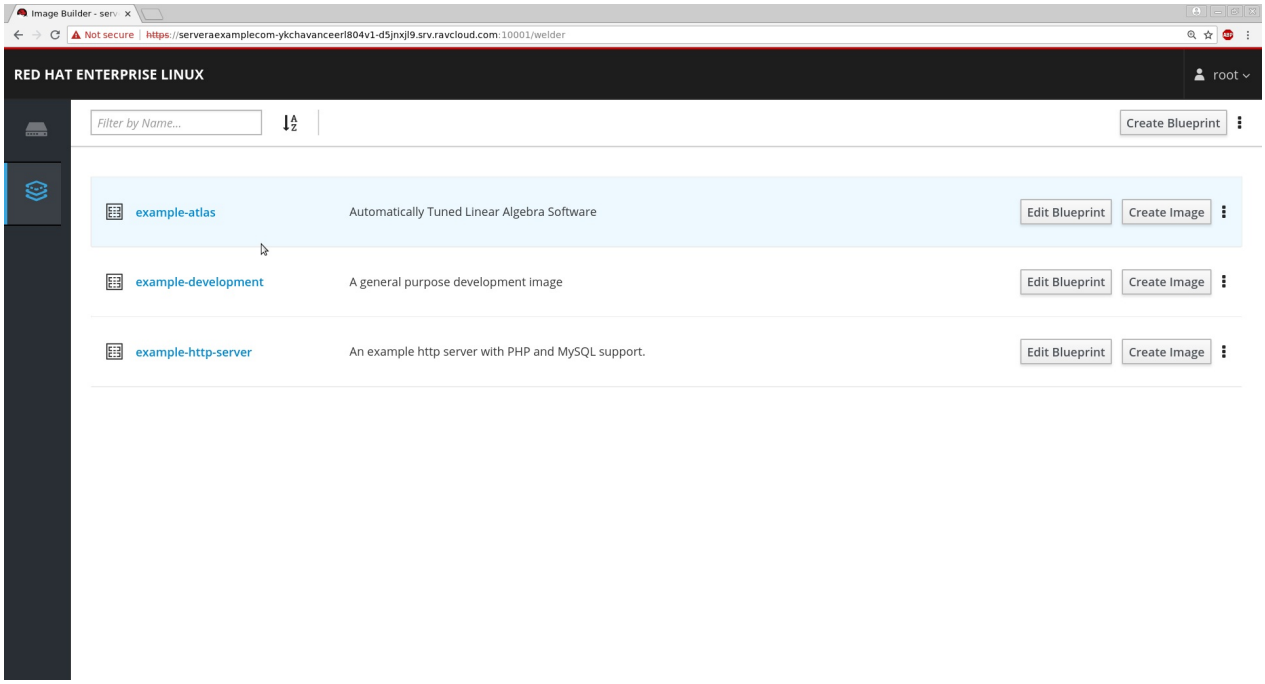
The Image Builder CLI provides a command line interface for using Image Builder. Some functions are only available in the CLI, such as adding a post-install configuration.

With the CLI installed, you can use the command shown here to output a help page for the "composer-cli" command.

Image Builder plugin for the RHEL 8 web console

- Plugin: **cockpit-composer**
- A GUI for Image Builder within the RHEL 8 web console
- Allows using Image Builder remotely from a web interface
- Does not require having the GUI packages installed on the RHEL system
- Currently: the Image Builder functions available in the web console are more limited than in the CLI

Click the image to see at full size:



Note

The default port to access the web console is **9090**. This lab setup uses custom port settings to address limitations and security. Remember to use port 9090 in a default install.

▼ Show transcript

The cockpit-composer plugin provides a GUI for Image Builder within the RHEL 8 web console. This interface allows for using Image Builder remotely from a web browser interface. Note again that there is a separate training in this series on the RHEL 8 web console, which includes how to set that up on a RHEL system. The web console is already set up for you in your labs for this training.

As a web application, this plugin does not require that you have the GUI packages installed on a RHEL 8 system in order to access it.

As of this writing, the Image Builder functions available through the web console are more limited than in the Image Builder CLI.

The image here shows what the Image Builder looks like within the web console. This view shows a list of blueprints inside the Image Builder. The next section covers more about blueprints.

Read the note on this page about the port access for the web console used in this training. Then, go on to the next page.

How to Install Image Builder

Before installing Composer, ensure that you have:

- Installed RHEL 8
- Enabled networking
- Registered the system and attached a valid subscription

Install composer with both the CLI and the web console plugin:

```
# yum install lorax lorax-composer composer-cli cockpit-composer
```

Recall: Lorax is a back-end tool that Image Builder uses to create Anaconda install images.

Enable and start the *lorax-composer* service:

```
# systemctl enable --now lorax-composer.socket
```

Restart the *cockpit* service to load the newly installed Image Builder plugin in the web console:

```
# systemctl restart cockpit.service
```

Note

If you use the terminal in the web console to restart *cockpit.service*, you will be disconnected from the console. Log in again after the service restart to resume using the console.

▼ Show transcript

Before installing the Image Builder, ensure that you have installed RHEL 8, enabled networking, registered the system, and attached a valid subscription. To install Image Builder with both the command line interface and the web console plugin, use the first command shown here. Notice the "lorax" packages in the command, and recall that Lorax is a back-end tool that Image Builder uses to create Anaconda install images.

After install, use the second command to enable and start the "lorax-composer" service. Keep this service name in mind when you need to support Image Builder.

Then wrap up the install by restarting the "cockpit" service so that it loads the newly installed Image Builder plugin in the web console. As noted here, if you run this from the terminal inside the web console, you'll be disconnected, and you'll need to log in to the console again.

How to Use Image Builder

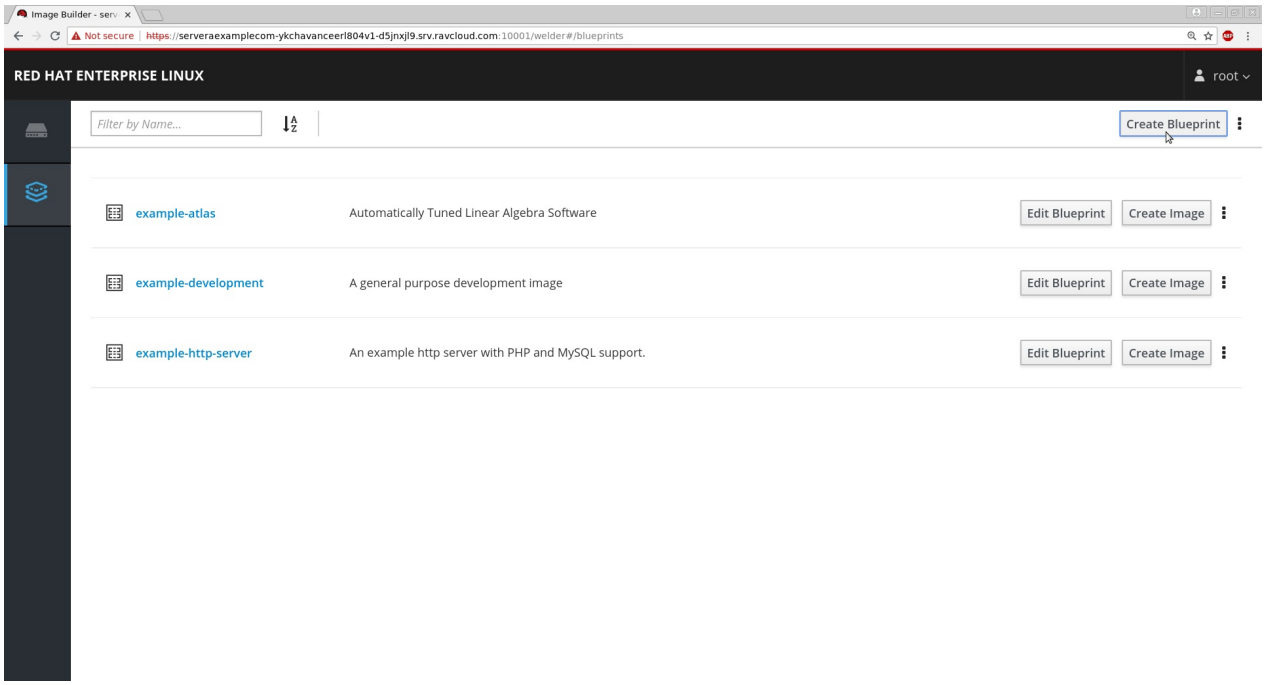
▼ Show transcript

This section covers how to use Image Builder to create custom RHEL 8 images in various formats.

Create blueprints

- Blueprint = a list of preselected components (RPM packages and software groups) that form a template for a custom image
- Create multiple images in multiple supported formats from the same blueprint
- A blueprint saves a record of the inputs and instructions for an image build

Click the image to see at full size:



▼ Show transcript

When an infrastructure administrator realizes that an application needs to be added to that infrastructure, they can use Image Builder to design a single system to deploy that application, and to generate images that can be used to install as many of those systems as needed. The feature that holds that customized plan for the system is called a blueprint or recipe.

A blueprint is a list of preselected components for a system that forms the template for a custom image. These components consist of packages or software groups.

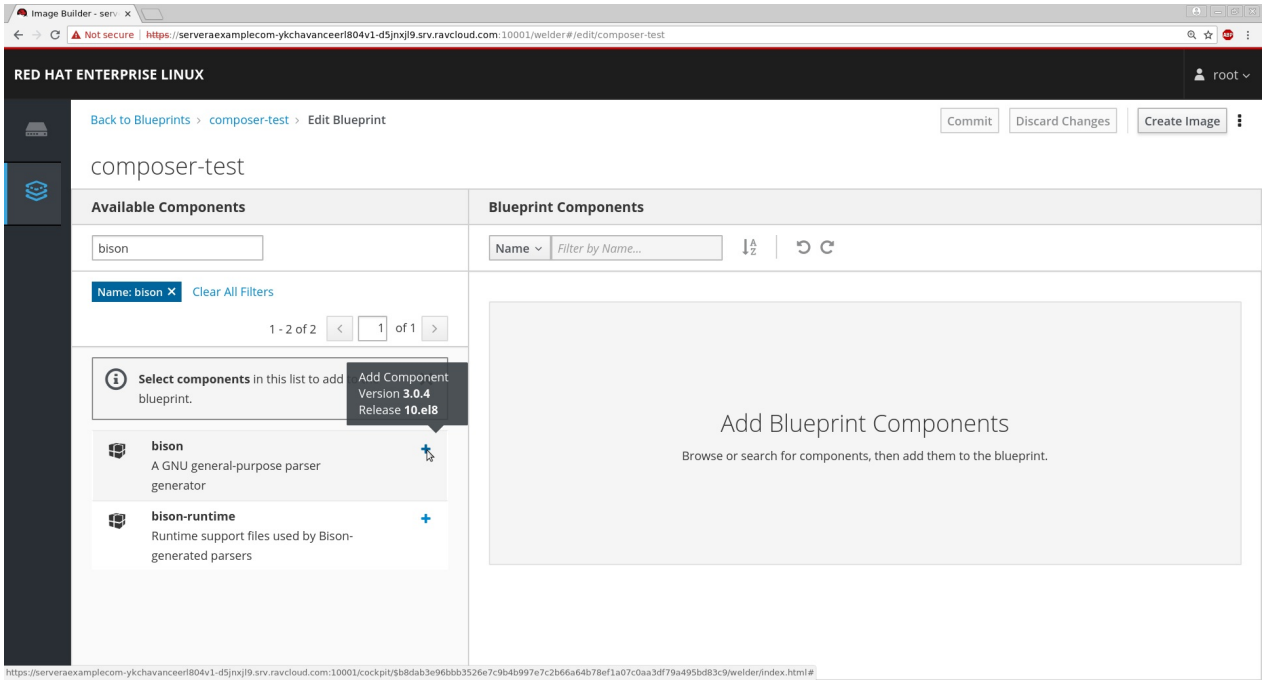
You can create multiple images in multiple supported formats from the same blueprint. A blueprint saves a record of the inputs and instructions for an image build so it can be reproduced at a later date.

The image on this page shows what the list of blueprints looks like in the web console interface. The "Image Builder" navigation sidebar option presents an overview, which lists the blueprints that are already configured. A few blueprints are provided by default, and those are shown here. Notice that there are buttons for each blueprint to edit the blueprint or to create a new image based on that blueprint.

Customize blueprints

- *Create Blueprint* opens the interface shown here
- Select components to customize the system
- A component can be either an RPM package or a Yum module
- Click **Commit** to save that blueprint
- Edit the blueprint as needed to add or remove components

Click the image to see at full size:



Note

Recall that post-installation configuration is not yet implemented in the web console interface for Image Builder.

▼ Show transcript

Click "Create Blueprint" in the Image Builder interface to open the page shown in the screenshot here. On this page, you can select components to customize the system. A component can be either an RPM package or a Yum module. If you're not familiar with Yum modules, look for the training on Yum 4 features and modules as part of this RHEL 8 Readiness Training series.

After making your selections, click "Commit" to save the new blueprint. From here, you can edit the blueprint as needed without starting over, including adding and removing components.

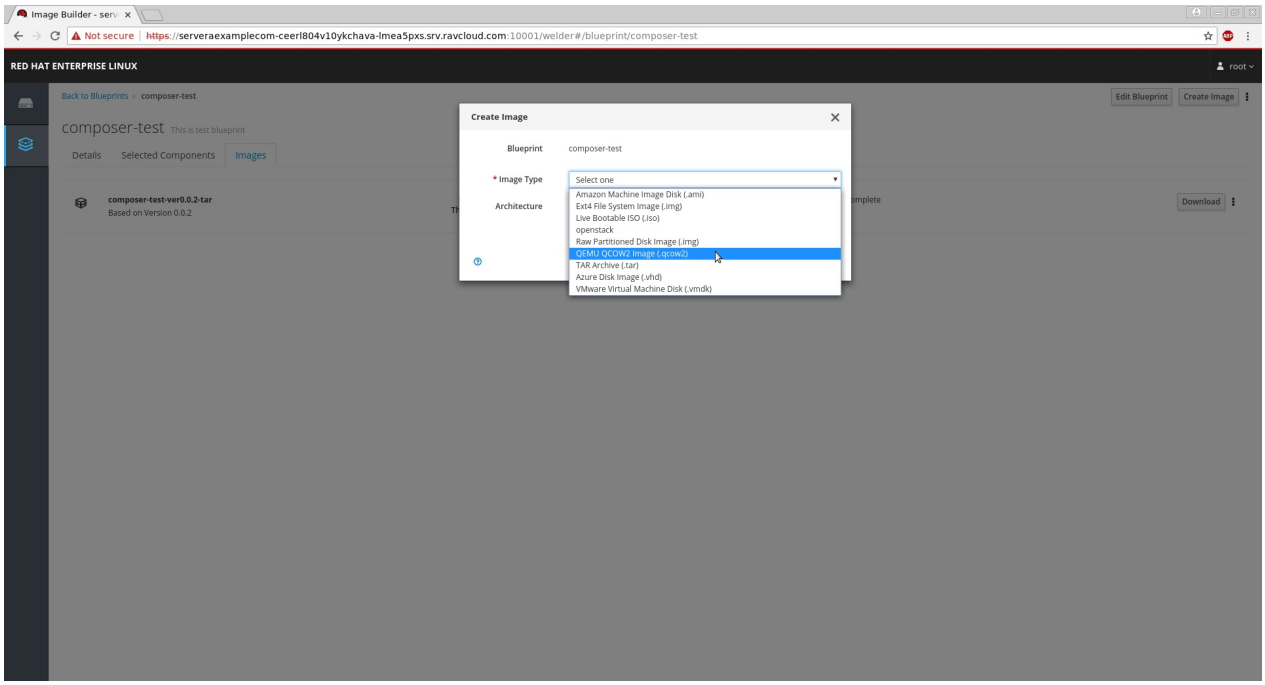
In the screenshot here, notice that when you mouse over the "plus" sign for a selected component on the left, it indicates the version and release for that component, which is locked in unless you choose a different version and release.

Recall from earlier that post-installation configuration is not yet implemented in the web console interface for Image Builder. That requires using the CLI, and it's covered later in this training.

Create images

- After creating a blueprint, you can create images based on that blueprint
- The pages for viewing and for editing a blueprint each include a button to build an image
- Image Builder allows building any of the supported images from a blueprint

Click the image to see at full size:



▼ Show transcript

After creating a blueprint, you can create images based on that blueprint. The previous pages, which show screenshots of viewing and editing a blueprint, each have a button for building an image from the selected blueprint. Image Builder allows building any of the supported image types mentioned earlier from a blueprint.

The image here shows selecting the output file format for the image you want to build.

Lab 1

In this lab, you will create one blueprint and one tar image using that blueprint. This lab is configured with a local repository, so there is no need to register the system to install packages.

To successfully complete this activity, you must:

- Install **lorax-composer**, **cockpit-composer**, and their dependencies on *servera.example.com* *https://servera-64a3.red.osp.opentlc.com:9090/*.
- Create a blueprint called **composer-test** in the web console.
- Add the **bison** package to the *composer-test* blueprint.
- Create a **tar** image using the *composer-test* blueprint. Please note that image creation takes more than five minutes.

The following video demonstrates these steps. You can follow along using your RHEL 8 web console as described in the Lab Environment setup page.

NOTE: Use port *9090* to connect to the cockpit.service running in your lab environment. The full URL to use depends upon the FQDN provisioned by OpenTLC. The FQDN for your server will be listed in the *Environment Info* under the *Custom Attributes*. For example, https://servera-64a3.red.osp.opentlc.com:9090/

Right click and maximize the video as needed to see details.

When the image build is complete, access a *servera* terminal and run the following command as root:

```
# ./kc-rl-804.sh grade1
```

Submit the completion code from that output as prompted here:

> Completion code for Lab 1: _____
>
> ans: OMIT

Lab 2

In this lab, you will edit your previously created blueprint to add and remove packages. Then, you will create a qcow2 image using that modified blueprint.

To successfully complete this activity, you must:

- Open the **composer-test** blueprint for editing.
- Remove the **bison** package.
- Add these packages: **findutils**, **diffutils**, **pciutils**
- Create a **qcow2** image for KVM using the *composer-test* blueprint.

The following video demonstrates these steps. You can follow along using your web console as described in the Lab Environment setup page.

Right click and Maximize the video as needed to see details.

When the image build is complete, access a *servera* terminal and run the following command as root:

```
# ./kc-rl-804.sh grade2
```

Submit the completion code from that output as prompted here:

> Completion code for Lab 2: _____
>
> ans: OMIT

OPTIONAL:
Notice that when the image build is complete, the status is shown as *Complete*, and the *Download* button is enabled. If you are running Linux in your local system with virtualization software installed (e.g. *virt-install* and *virt-viewer* in RHEL or Fedora), then you can download the image file from your browser and test it from your local system. For *virt-install*, test the image using the following command:

```
# virt-install --name RHEL8Lab2 --memory 2048 --vcpus 2 --os-variant rhel8.0 --import --disk  <path to downloaded qcow2 file>
```

This command should open a *virt-viewer* window with the text-based login prompt. You won't be able to log in at this point, but the reason and solution for this is covered in the next section.

Issues & Troubleshooting

▼ Show transcript

This section provides information about common issues with Image Builder and how to troubleshoot Image Builder issues.

Where to investigate Image Builder issues

Configuration file:
/etc/lorax/composer.conf

Log files:
/var/log/lorax-composer/composer.log
/var/log/lorax-composer/dnf.log
/var/log/lorax-composer/program.log
/var/log/lorax-composer/server.log

Check the status and error messages for the *lorax-composer* service:

```
systemctl status lorax-composer.service
```

Get detailed logs about the *lorax-composer* service:

```
journalctl -fu lorax-composer
```

Get a list of blueprints:

```
composer-cli blueprints list
```

Confirm whether Image Builder is getting the Yum (DNF) sources correctly and not using a third-party image:

```
composer-cli sources list
```

▼ Show transcript
This page covers data to collect when investigating Image Builder issues. The configuration file and log files are listed here along with a command that shows the status and errors for the "lorax-composer" service. The log files mentioned here are collected by default with Sosreport in RHEL 8. The last two commands shown here are "composer-cli" commands used to get list of the blueprints and configured repository resources in Image Builder. Sosreport collects these and other troubleshooting commands under the "sos_commands/composer" directory of its report.

Common issues

- Misconfigured repository sources
- *lorax-composer.socket* is not enabled
  Error message in the web console: *'An error occurred. not-found'*
- Old version of Image Builder needed SELinux needs to be *Permissive*
  - This is fixed in the latest version
  - Bug addressing this limitation: https://bugzilla.redhat.com/show_bug.cgi?id=1645189

▼ Show transcript

This page lists some common issues you might have to address when working with or supporting Image Builder. As more Red Hat customers adopt this feature and encounter other issues over time, this list may grow.

One common issue is that Image Builder has misconfigured repository sources. The source list mentioned previously is important for identifying this issue.

Another issue that "lorax-composer.socket" is not enabled. This condition results in the error message shown here in the web console.

Older versions of Image Builder needed to have SELinux in permissive mode. This was a limitation, and it has been fixed in the latest versions of Image Builder. If you think you may be seeing this, though, see the bug linked here for more information about this limitation.

Known issue: user accounts on images generated from the web console

Images created with Image Builder in the web console:

- Have their root account locked for security purposes
- By default, do not have any other users configured
- Cannot have a user added using just the web console (requires the CLI)

This results in images that have no way to log in.

Workaround for clouds only:

- Deploy the image on a cloud and use *cloud_init* to add new users to them.
- See RHEL 8 Beta documentation for vendor-specific instructions:
  Installing and deploying RHEL (RHEL 8), Chapter 7, "Creating Cloud Images With Composer"

Workaround for qcow2 images only:

- Mount the qcow2 image offline using this knowledge base article and remove or change the password: Changing the password on the RHEL 7 kvm qcow2 download

Otherwise use the Image Builder CLI (*composer-cli*) or edit blueprints manually.

Track this bug: Bug 1655862 - Option to include username or custom settings in the blueprint

Note

You can view some screen captures of a proposed fix here on GitHub. The final fix released in RHEL may differ from this proposal.

▼ Show transcript

Though the images you created in your earlier labs were bootable, they were practically unusable because the root account is locked and no other users were added to the images.

On images created with Image Builder in the web console, the root account is locked for security purposes and, by default, they do not have any other users configured. Also, because of the web console's current limitations, you cannot add a user to the blueprint using just the web console. This results in images that have no way to log in.

In cloud environments, you can make these images usable by using "cloud_init" to add new users to them. The steps to accomplish this are vendor-specific, so see the RHEL 8 Beta documentation link here about how to add cloud images in their respective environments.

Without cloud_init, there is no other way to add users in images created entirely from the web console. This function is planned for a future release to follow soon after the RHEL 8 GA release. In the meantime, Image Builder users should use the CLI to add users to an image, or modify a blueprint's configuration file manually. The next section covers how to do this.

Customize a blueprint

To customize a blueprint:

1. Download a copy of the blueprint configuration file from Image Builder:

```
composer-cli blueprints save <blueprint-name>
```

2. Edit that file using file editor like vi (e.g. *blueprint-name.toml*).

3. Push the revised blueprint configuration file back to Image Builder:

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

4. Verify that your changes appear in the configuration file using this command:

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

In your labs for this training, replace <*blueprint-name*> with **composer-test**.

Blueprint configuration files follow TOML (Tom's Obvious, Minimal Language) format, which uses key/value pairs. For more information, see TOML on GitHub.

▼ Show transcript

To customize a blueprint, download a copy of the blueprint configuration file from Image Builder, edit that file, and then push the revised blueprint configuration file back to Image Builder. The command syntax for downloading and pushing are shown here.

Blueprint configuration files follow "Tom's Obvious, Minimal Language" format, which uses key/value pairs. For more information, see the GitHub link here. The next couple of pages include some examples, also.

Add SSH keys for root

Append lines to the blueprint configuration file to add SSH keys for a user.

SSH keys for the root user:

```
[[customizations.sshkey]]
user = "root"
key = "<public SSH key>"
```

If you copy the contents of your personal public key file (e.g. *.ssh/id_rsa.pub*) to the value of *key*, you should be able to use SSH to log in as root on systems launched from the resulting image.

▼ Show transcript

You should already know that SSH key authentication is more secure than standard password-based authentication. When you want to add SSH keys for the root user to an image created by Image Builder, edit its blueprint configuration file to add the lines shown here. If you copy the contents of your personal public key file to the value of "key", you should be able to use SSH to log in as root on systems launched from the resulting image.

Add users

Append lines to the blueprint configuration file to add a new user.

New user *myuser*:

```
[[customizations.user]]
name = "myuser"
password = "<password in plain text OR an encrypted password string>"
key = "<public SSH key>"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = 1001
gid = 1001
```

To generate encrypted password string:

```
python3 -c "import crypt, getpass; print(crypt.crypt(getpass.getpass(), crypt.METHOD_SHA512))"
```

Note

Demonstration of editing a blueprint configuration:
RHEL 8 Beta - Building Custom RHEL Images With Image Builder (Composer) (YouTube)

▼ Show transcript
Similar to adding SSH keys for the root user, you can add a new user by adding lines to the blueprint configuration file. Edit the file to add the lines shown here. You can provide the password in plain text or as an encrypted password string. We've provided a Python command you can run on a RHEL system if you want to generate an encrypted password string for this field. Also notice that you can paste a public SSH key here as described on the previous page.

You should recognize that not all of these fields are required to create a new user in Linux. If desired, you could just provide the username and password.

For a demonstration of editing a blueprint to add users and SSH keys, see the video linked here from Red Hat's YouTube channel.

Lab 3

In this lab, you will modify the *composer-test* blueprint configuration file to add new user *user1* and to add a public SSH key for the root user. Then, you will create a qcow2 image using that modified blueprint.

To successfully complete this activity, you must:

- Save the **composer-test** blueprint to */root* on your *servera* system using the *composer-cli* command.
- Configure a new user **user1** with password **pass1**, and that user belongs to the **wheel** and **users** groups.
- Configure an SSH key for the root user. You can find your root user's public key file on *servera* at **/root/.ssh/composer_rsa.pub**.
- Push the modified *composer-test* blueprint, and confirm that your changes are applied.
- Use the RHEL 8 web console to create a **QEMU qcow2 Image (.qcow2)** using the *composer-test* blueprint.

The resulting image build should then have *user1* with sudo access (because of the *wheel* group), and it allows SSH for the root user with the correct key authentication.

When the image build is complete, access a *servera* terminal and run the following command as root:

```
# ./kc-rl-804.sh grade3
```

Submit the completion code from that output as prompted here:

> Completion code for Lab 3: _____
>
> ans: OMIT

OPTIONAL:
Like in Lab 2, you have the option to download and test this image on your local system. If you are running Linux with *virt-install*, use the following command to test the image:

```
# virt-install --name RHEL8Lab3 --memory 2048 --vcpus 2 --os-variant rhel8.0 --import --disk  <path to downloaded qcow2 file>
```

This command should open a *virt-viewer* window with the text-based login prompt. This time, you should be able to test the user login options you configured:

- Log in as *user1* with password *pass1*.
- Confirm that *user1* is a member of both the *users* and *wheel* groups by running the *id* command.
- Confirm that *user1* is able to run privileged commands with *sudo*.

Resources

Weldr Lorax composer homepage

Cockpit project homepage

Lorax upstream project (GitHub)

Red Hat developer blog on Composer

Feedback

Thank you for taking time to provide feedback about this training using the form below.

How likely are you to recommend this training module to other associates?

Not at all Likely 0 1 2 3 4 5 6 7 8 9 10 Extremely likely

Enter additional comments here...

Submit FeedbackReset