Red Hat Enterprise Linux 9

创建自定义 RHEL 系统镜像

在 Red Hat Enterprise Linux 9 中使用镜像构建器创建自定义系统镜像

Last Updated: 2022-08-15

Red Hat Enterprise Linux 9 创建自定义 RHEL 系统镜像

在 Red Hat Enterprise Linux 9 中使用镜像构建器创建自定义系统镜像

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Composing_a_customized_RHEL_system_image.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

摘要

镜像构建器是用于创建部署就绪的自定义系统镜像的工具：安装磁盘、虚拟机、特定于云供应商的镜像等。使用镜像构建器，如果与手动过程相比，您可以更快地创建这些镜像，因为它消除了每个输出类型所需的特定配置。本文档描述了如何设置镜像构建器并使用它创建镜像。

目录

使开源包含更多 4

对红帽文档提供反馈 5

第 1 章 镜像构建器描述 6
1.1. IMAGE BUILDER 简介 6
1.2. 镜像构建器术语 6
1.3. 镜像构建器输出格式 6
1.4. 镜像构建器系统要求 7

第 2 章 安装镜像构建器 8
2.1. 在虚拟机中安装镜像构建器 8

第 3 章 管理存储库 10
3.1. 镜像构建器默认系统存储库 10
3.2. 覆盖系统存储库 10
3.3. 覆盖支持订阅的系统存储库 11

第 4 章 使用镜像构建器命令行界面创建系统镜像 13
4.1. 镜像构建器命令行界面 13
4.2. 使用命令行界面创建镜像构建器蓝图 13
4.3. 使用命令行界面编辑镜像构建器蓝图 15
4.4. 在命令行界面中使用镜像构建器创建系统镜像 16
4.5. 基本镜像构建器命令行命令 17
4.6. 镜像构建器蓝图格式 18
4.7. 支持的镜像自定义 19
4.8. 安装的软件包 23
4.9. 已启用的服务 26

第 5 章 使用 IMAGE BUILDER WEB 控制台界面创建系统镜像 28
5.1. 在 RHEL WEB 控制台中访问镜像构建器 GUI 28
5.2. 在 WEB 控制台界面中创建 IMAGE BUILDER 蓝图 28
5.3. 在 WEB 控制台界面中编辑 IMAGE BUILDER 蓝图 29
5.4. 将用户和组添加到 WEB 控制台界面的 IMAGE BUILDER 蓝图中 30
5.5. 在 WEB 控制台界面中使用 IMAGE BUILDER 创建系统镜像 32
5.6. 在蓝图中添加源 33
5.7. 为蓝图创建用户帐户 35
5.8. 使用 SSH 密钥创建用户帐户 36

第 6 章 使用镜像构建器创建引导 ISO 安装程序镜像 39
6.1. 在命令行界面上使用镜像构建器创建一个引导 ISO 安装程序镜像 39
6.2. 在 GUI 中使用镜像构建器创建引导 ISO 安装程序镜像 40
6.3. 将 ISO 镜像安装到裸机系统 41

第 7 章 使用镜像构建器准备和部署 KVM 客户机镜像 42
7.1. 使用镜像构建器创建自定义 KVM 客户机镜像 42
7.2. 从 KVM 客户机镜像创建虚拟机 43

第 8 章 使用镜像构建器准备并上传云镜像 45
8.1. 准备上传 AWS AMI 镜像 45
8.2. 在 CLI 中将 AMI 镜像上传到 AWS 46
8.3. 将镜像推送到 AWS CLOUD AMI 47
8.4. 准备上传 AZURE VHD 镜像 49
8.5. 将 VHD 镜像上传到 AZURE 51

8.6. 将 VMDK 镜像上传到 VSPHERE 51
8.7. 将 VMWARE 镜像推送到 VSPHERE 53
8.8. 将 VHD 镜像推送到 AZURE 云 56
8.9. 将 QCOW2 镜像上传到 OPENSTACK 58
8.10. 准备将镜像上传到 ALIBABA 61
8.11. 将镜像上传到 ALIBABA 62
8.12. 将镜像导入到 ALIBABA 63
8.13. 使用 ALIBABA 创建自定义镜像实例 64

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请让我们了解如何改进文档。

- 关于特定内容的简单评论：
  1. 请确定您使用 *Multi-page HTML* 格式查看文档。另外，确定 **Feedback** 按钮出现在文档页的右上方。
  2. 用鼠标指针高亮显示您想评论的文本部分。
  3. 点在高亮文本上弹出的 **Add Feedback**。
  4. 按照显示的步骤操作。
- 要通过 Bugzilla 提交反馈，请创建一个新的 ticket：
  1. 进入 Bugzilla 网站。
  2. 在 Component 中选择 **Documentation**。
  3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
  4. 点 **Submit Bug**。

第 1 章 镜像构建器描述

1.1. IMAGE BUILDER 简介

您可以使用镜像构建器（Image Builder）创建 Red Hat Enterprise Linux 的自定义系统镜像，包括准备在云平台上部署的系统镜像。镜像构建器会自动处理每种输出类型的设置详情，因此比手动创建镜像的方法更易于使用，使用起来也更快。您可以通过 **composer-cli** 工具中的命令行界面或 RHEL web 控制台中的图形用户界面来访问镜像构建器的功能。

从 Red Hat Enterprise Linux 8.3 开始，**osbuild-composer** 后端替换了 **lorax-composer**。新服务为镜像构建提供 REST API。因此，用户可从更可靠的后端及更可预测的输出镜像中受益。

镜像构建器作为系统服务 **osbuild-composer** 运行。您可以通过两个接口与这个服务交互：

- 用于在终端中运行命令的 CLI 工具 **composer-cli**。这个方法是首选的。
- RHEL web 控制台的 GUI 插件。

1.2. 镜像构建器术语

蓝图（Blueprint）

蓝图通过列出将属于系统一部分的软件包和自定义来定义自定义系统镜像。蓝图可以被编辑并被版本化。从蓝图创建系统镜像时，该镜像与 RHEL web 控制台的镜像构建器界面中的蓝图相关联。
蓝图以纯文本形式向用户显示，格式为 TOML 格式。

组合（Compose）

Compose 是基于特定蓝图的特定版本的系统镜像的单个构建。作为一个术语，Compose 代表系统镜像以及来自其创建、输入、元数据和进程本身的日志。

自定义（Customizations）

自定义是系统的规范，而不是软件包。这包括用户、组和 SSH 密钥。

1.3. 镜像构建器输出格式

镜像构建器可以以多种输出格式创建镜像，如下表中所示。要检查支持的类型，请运行以下命令：

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

表 1.1. 镜像构建器输出格式

描述	CLI 名称	文件扩展
QEMU QCOW2 镜像	**qcow2**	**.qcow2**
TAR 归档	**tar**	**.tar**
Amazon 机器镜像磁盘	**ami**	**.raw**
Azure 磁盘镜像	**vhd**	**.vhd**

描述	CLI 名称	文件扩展
VMware 虚拟机磁盘	**vmdk**	**.vmdk**
Openstack	**openstack**	**.qcow2**
用于边缘提交的 RHEL	**edge-commit**	**.tar**
用于边缘容器的 RHEL	**edge-container**	**.tar**
用于边缘安装程序的 RHEL	**edge-installer**	**.iso**
用于 Edge Raw 的 RHEL	**edge-raw-image**	**.tar**
用于边缘简化安装程序的 RHEL	**edge-simplified-installer**	**.iso**
ISO 镜像	**image-installer**	**.iso**

1.4. 镜像构建器系统要求

镜像构建器运行的环境（如专用的虚拟机）必须满足下表中列出的要求。

表 1.2. 镜像构建器系统要求

参数	最低要求值
系统类型	专用虚拟机
处理器	2 个内核
内存	4 GiB
磁盘空间	20 GiB 可用空间（位于 **/var** 文件系统）
访问权限	管理员级别(root)
网络	连接至互联网

注意

互联网连接不是前提条件。如果您将其重新配置为不连接到 Red Hat CDN，您可以在隔离的网络中使用镜像构建器。

第 2 章 安装镜像构建器

在使用镜像构建器之前，您必须在虚拟机中安装镜像构建器。

2.1. 在虚拟机中安装镜像构建器

要在专用的虚拟机上安装镜像构建器，请按照以下步骤操作：

先决条件

- 连接到虚拟机。
- Image Builder 的虚拟机必须被安装，并订阅到 Red Hat Subscription Manager(RHSM)或 Red Hat Satellite，并正在运行。

流程

1. 在虚拟机上安装 Image Builder 和其他必要的软件包：
   - **osbuild-composer** - 从 RHEL 8.3 开始支持
   - **composer-cli**
   - **cockpit-composer**
   - **bash-completion**

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

   Web 控制台作为 *cockpit-composer* 软件包的依赖项安装。
2. 在每次重启后启动镜像构建器：

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

   osbuild-composer 和 **cockpit** 服务在第一次访问时自动启动。
3. 载入 shell 配置脚本，以便在不重启的情况下立即启动 **composer-cli** 命令的自动完成功能：

   ```
   $ source /etc/bash_completion.d/composer-cli
   ```

重要

osbuild-composer 软件包是新的后端引擎，它将是从 Red Hat Enterprise Linux 8.3 及更高版本开始的所有新功能的首选默认和重点。之前的后端 **lorax-composer** 软件包被视为已弃用，将只接收 Red Hat Enterprise Linux 8 生命周期剩余部分所选定的修复，并将在以后的主发行版本中被忽略。建议卸载 **lorax-composer**，使用 osbuild-composer。

验证

您可以使用系统日志来跟踪镜像构建器服务活动。此外，您还可以在 文件中找到日志消息。

- 要查找回溯的日志输出，请运行以下命令：

```
$ journalctl | grep osbuild
```

- 显示远程或本地 worker：

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

- 显示运行的服务：

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

第 3 章 管理存储库

3.1. 镜像构建器默认系统存储库

osbuild-composer 后端不会继承位于 **/etc/yum.repos.d/** 目录中的系统存储库。相反，它拥有自己的一组在 **/usr/share/osbuild-composer/repositories** 目录中定义的官方存储库。要覆盖官方存储库，您必须在 **/etc/osbuild-composer/repositories** 中定义覆盖。这个目录用于用户定义的覆盖，这里的文件优先于 **/usr** 目录中的文件。

配置文件不是 **/etc/yum.repos.d/** 中常见的 DNF 存储库格式。相反，它们是简单的 JSON 文件。

3.2. 覆盖系统存储库

您可以按照以下步骤在 **/etc/osbuild-composer/repositories** 目录中配置存储库覆盖。

先决条件

- 您有一个可从主机系统访问的自定义存储库

流程

1. 创建包含您要使用的存储库覆盖的目录：

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

2. 创建具有以下结构的 JSON 文件，例如：

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

仅指定以下属性之一： **metalink**、**mirrorlist** 或 **baseurl**。剩余的字段是可选的。

3. 使用与 RHEL 版本对应的名称保存 JSON 文件，例如：

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

   a. 或者，您可以从 **/usr/share/osbuild-composer/** 复制发行版的 JSON 文件，并修改其内容。

      i. 将存储库文件复制到您创建的目录中。

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

将 rhel-version.json 替换为您的 RHEL 版本，例如： rhel-9.json

4. 使用文本编辑器，编辑 **rhel-9.json** 文件中的 **baseurl** 路径。例如：

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

因此，存储库指向从 **/etc/yum.repos.d/redhat.repo** 文件中复制过来的正确的 URL。

3.3. 覆盖支持订阅的系统存储库

osbuild-composer 服务可以使用 **/etc/yum.repos.d/redhat.repo** 文件中定义的系统订阅。要在 **osbuild-composer** 中使用系统订阅，您需要定义一个具有以下特点的存储库覆盖：

- 与 **/etc/yum.repos.d/redhat.repo** 中定义的存储库相同的 **baseurl**。
- JSON 对象中定义的 **"rhsm": true** 值。

先决条件

- 在 **/etc/yum.repos.d/redhat.repo** 中定义了订阅的系统
- 您已创建了一个存储库覆盖。请参阅 覆盖系统存储库。

流程

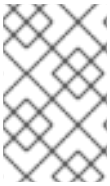
1. 从 **/etc/yum.repos.d/redhat.repo** 文件中获取 **baseurl** ：

```
[AppStream]
name = AppStream mirror example
baseurl = https://mirror.example.com/RHEL-9/9.0/AppStream/x86_64/os/
enabled = 1
gpgcheck = 0
sslverify = 1
sslcacert = /etc/pki/ca1/ca.crt
sslclientkey = /etc/pki/ca1/client.key
sslclientcert = /etc/pki/ca1/client.crt
metadata_expire = 86400
enabled_metadata = 0
```

2. 配置存储库覆盖以使用相同的 **baseurl**，并将 **rhsm** 设为 true：

```
{
  "x86_64": [
    {
      "name": "AppStream mirror example",
      "baseurl": "https://mirror.example.com/RHEL-9/9.0/AppStream/x86_64/os/",
      "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (…)",
      "check_gpg": true,
      "rhsm": true
```

```
        }
    ]
}
```



注意

osbuild-composer 不自动使用 **/etc/yum.repos.d/** 中定义的存储库。您需要手动将它们指定为系统存储库覆盖，或使用 **composer-cli** 指定为额外的 **源**。系统存储库覆盖通常用于"BaseOS"和"AppStream"存储库，而 **composer-cli** 源则用于所有其他存储库。

其他资源

- 当主机注册到 Satellite 6 时，Composer 镜像构建器使用 CDN 存储库

第 4 章 使用镜像构建器命令行界面创建系统镜像

镜像构建器是创建自定义系统镜像的工具。要控制镜像构建器并创建自定义系统镜像，请使用命令行界面，这是当前使用镜像构建器的首选方法。

4.1. 镜像构建器命令行界面

镜像构建器命令行界面目前是使用镜像构建器的首选方法。它提供比 Web 控制台界面更多的功能。要使用此界面，请使用适当的选项和子命令来运行 **composer-cli** 命令。

命令行界面的工作流总结如下：

1. 将蓝图定义导出（*保存*）到纯文本文件
2. 在文本编辑器中编辑这个文件
3. 将蓝图文本文件导入（*推送*）到镜像构建器
4. 运行 compose 来从蓝图构建镜像
5. 导出镜像文件以下载它

除了实现此过程的基本子命令外，**composer-cli** 命令还提供多个子命令来检查配置的蓝图和组合的状态。

要以非 root 用户身份运行 **composer-cli** 命令，用户必须位于 usld **r** 或 **root** 组中。

- 要在 **weldr** 或 **root** 组中添加用户，请运行以下命令：

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

4.2. 使用命令行界面创建镜像构建器蓝图

此流程描述了如何使用命令行界面创建新镜像构建器蓝图。

流程

1. 创建一个包含以下内容的纯文本文件：

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

用您的蓝图的名称和描述替换 *BLUEPRINT-NAME* 和 *LONG FORM DESCRIPTION TEXT*。

根据 Semantic Versioning 方案，将 *0.0.1* 替换为一个版本号。

2. 对于您要包含在蓝图中的每个软件包，请在文件中添加以下行：

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

使用软件包名称替换 *package-name*，比如 **httpd**、**gdb-doc** 或者 **coreutils**。

将 *package-version* 替换为要使用的版本。此字段支持 **dnf** 版本规范：

- 对于特定版本，请使用确切的版本号，如 8.6.0。
- 对于最新的可用版本，请使用星号 *。
- 对于最新的次版本，请使用以下格式，如 8.*。

3. 您可以通过多种方式自定义您的蓝图。在本例中，可以通过执行以下步骤禁用 Simultaneous Multi Threading(SMT)。如需了解更多自定义配置，请参阅支持的镜像自定义。

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

4. 将文件保存为 *BLUEPRINT-NAME*.toml，并关闭文本编辑器。
5. 推送（导入）蓝图：

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

将 *BLUEPRINT-NAME* 替换为您在前面步骤中使用的值。

6. 列出现有的蓝图以验证蓝图是否已推送并已存在：

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

7. 要显示您刚刚添加的蓝图配置，请运行以下命令：

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

8. 检查蓝图中列出的组件和版本是否有效：

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

注意

要将 **composer-cli** 命令用作非 root 命令创建镜像，请将您的用户添加到 **weldr** 或 **root** 组中。

验证

如果 Image Builder 无法从自定义软件仓库中解决软件包，请按照以下步骤执行：

- 删除 osbuild-composer 缓存：

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

其他资源

- osbuild-composer 无法从我的自定义存储库中解决软件包
- 使用代理服务器编写自定义的 RHEL 系统镜像

4.3. 使用命令行界面编辑镜像构建器蓝图

要在命令行界面中编辑现有镜像构建器蓝图，请按照以下步骤操作。

流程

1. 将蓝图保存（导出）到本地文本文件：

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

2. 使用文本编辑器编辑 *BLUEPRINT-NAME*.toml 文件并进行更改。
3. 在完成编辑前，请确保文件是一个有效的蓝图：
   a. 如果存在，删除此行：

```
packages = []
```

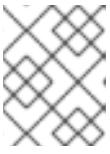
   b. 增加版本号。请记住, Image Builder 蓝图版本必须使用 Semantic Versioning 方案。请注意，如果您没有更改版本，补**丁版本** 组件会自动增加。
   c. 检查内容是否是有效的 TOML 规格。如需更多信息，请参阅 TOML 文档。

注意

TOML 文档是一款社区产品，红帽不支持。您可以使用该工具报告任何问题：https://github.com/toml-lang/toml/issues

4. 保存文件并关闭文本编辑器。
5. 将蓝图推送到镜像构建器：

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



注意

要将蓝图导入镜像构建器，请提供文件名，包括 **.toml** 扩展，而其他命令中则只使用蓝图名称。

6. 要验证上传到镜像构建器的内容是否与您的编辑匹配，请列出蓝图的内容：

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

7. 检查蓝图中列出的组件和版本是否有效：

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

4.4. 在命令行界面中使用镜像构建器创建系统镜像

此流程演示了如何使用 Image Builder 命令行界面构建自定义镜像。

先决条件

- 您已为镜像准备了蓝图。

流程

1. 启动 compose：

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

将 *BLUEPRINT-NAME* 替换为蓝图的名称，将 *IMAGE-TYPE* 替换为镜像的类型。对于可能的值，请参阅 **composer-cli compose types** 命令的输出。

compose 进程在后台启动，并显示 composer Universally Unique Identifier(UUID)。

2. 等待 compose 过程完成。镜像创建最多可能需要十分钟才能完成。
检查 Compose 的状态：

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

完成的 compose 显示状态值 **FINISHED**。根据 UUID 识别列表中的内容。

3. 完成 compose 过程后，下载生成的镜像文件：

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

使用前面步骤中显示的 *UUID* 值替换 UUID。

验证

创建镜像后，您可以使用以下命令检查镜像创建进度：

- 检查 compose 状态：

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

- 下载镜像元数据：

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

- 下载镜像的日志：

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

该命令会创建一个 **.tar** 文件，其中包含创建镜像的日志。如果日志为空，您可以检查日志。

- 检查日志：

```
$ journalctl | grep osbuild
```

- 检查清单：

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

您可以在日志中找到 *job_UUID*.json。

其他资源

- 追踪镜像构建器

4.5. 基本镜像构建器命令行命令

Image Builder 命令行界面提供以下子命令。

蓝图操作

列出所有可用的蓝图

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

显示 TOML 格式的蓝图内容

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

将蓝图内容以 TOML 格式保存（导出）到文件 BLUEPRINT-NAME.toml中

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

删除蓝图

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

将 TOML 格式的蓝图文件推送到镜像构建器中

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

从蓝图制作镜像

列出可用的镜像类型

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

启动一个目录

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

将 *BLUEPRINT* 替换为要构建的蓝图名称，将 *COMPOSE-TYPE* 替换为输出镜像类型。

列出所有 compose

-

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

列出所有 compose 及其状态

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

取消正在运行的 compose

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

删除完成的 compose

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

显示有关 compose 的详细信息

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

下载 compose 的镜像文件

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

其他资源

- *composer-cli*(1)书册页提供了可用子命令和选项的完整列表：

```
$ man composer-cli
```

- **composer-cli** 命令提供有关子命令和选项的帮助：

```
# composer-cli help
```

4.6. 镜像构建器蓝图格式

镜像构建器蓝图以 TOML 格式作为纯文本向用户显示。

典型的蓝图文件元素包括：

蓝图元数据

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

用您的蓝图的名称和描述替换 *BLUEPRINT-NAME* 和 *LONG FORM DESCRIPTION TEXT*。

根据 Semantic Versioning 方案，将 *VERSION* 替换为版本号。

这部分只针对整个蓝图文件显示一次。

条目 *modules* 描述了要安装到镜像中的软件包名称和与 glob 相匹配的版本。

条目 *group* 描述了要安装到镜像中的一组软件包。group 将其软件包分类如下：

- Mandatory（必需）
- Default（默认）
- Optional（可选）
  蓝图会安装必需的软件包。没有选择可选软件包的机制。

镜像中包含的组

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

使用组群名称替换 *group-name*，比如 **anaconda-tools**、**widget**、**wheel** 或者 **users**。

镜像中包含的软件包

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

使用软件包名称替换 *package-name*，比如 **httpd**、**gdb-doc** 或者 **coreutils**。

将 *package-version* 替换为要使用的版本。此字段支持 **dnf** 版本规范：

- 对于特定版本，请使用具体的版本号，如 8.30。
- 对于最新的可用版本，请使用星号 *。
- 对于最新的次版本，请使用类似 8.* 的格式。

为每个要包括的软件包重复这个块。

4.7. 支持的镜像自定义

蓝图中支持一些镜像自定义。要使用这些选项，您必须首先在蓝图中配置自定义，并将它导入到镜像构建器。

注意

目前在 web 控制台中不支持这些自定义。

设置镜像主机名

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

生成系统镜像的用户规范

```
[[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
```

注意

GID 是可选的，必须已存在于镜像中、由软件包创建，或者由蓝图 **[[customizations.group]]** 条目创建。

重要

若要生成哈希，您必须在您的系统上安装 **python3**。以下命令将安装 **python3** 软件包。

```
# dnf install python3
```

将 *PASSWORD-HASH* 替换为实际密码散列。要生成哈希，请使用如下命令：

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

将 *PUBLIC-SSH-KEY* 替换为实际公钥。

使用适当的值替换其他占位符。

根据需要可以省略任何行，只需要用户名。

为每个要包含的用户重复这个块。

生成系统镜像的组规范

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

为每个组重复此块。

设置现有用户的 ssh 密钥

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

注意

这个选项仅适用于现有用户。要创建用户并设置 ssh 密钥，请参阅本节中自定义的**系统镜像自定义的用户规格**。

在默认值中附加一个内核引导参数选项

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

默认情况下，Image Builder 将默认内核构建到镜像中。但是，您可以使用蓝图中的以下配置自定义内核

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

定义要在镜像中使用的内核名称

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

为生成的系统镜像设置时区和 *网络时间协议* (NTP) 服务器

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

如果您没有设置时区，系统将默认使用 *Universal Time, Coordinated* （**UTC**）。设置 NTP 服务器是可选的。

为生成的系统镜像设置区域设置

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

设置语言和键盘选项是必须的。您可以添加多种语言。您添加的第一个语言是主要语言，其他语言为次要语言。

为生成的系统镜像设置防火墙

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

您可以使用数字端口或 **/etc/services** 文件中的名称来启用列表。

自定义防火墙服务

查看可用的防火墙服务。

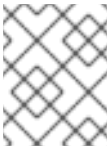
```
$ firewall-cmd --get-services
```

在蓝图中，在 **customs.firewall.service** 部分下指定要自定义的防火墙服务。

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

firewall.services 中列出的服务与 **/etc/services** 文件中提供的名称不同。

您可选择为计划创建的系统镜像自定义防火墙服务。



注意

如果您不想自定义防火墙服务，请省略蓝图中的 **[customizations.firewall]** 和 **[customizations.firewall.services]** 部分。

设置在引导时要启用哪个服务

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

您可以控制在引导期间要启用哪些服务。有些镜像类型已经启用或禁用了特定服务，以便镜像正常工作，此设置无法覆盖。

注意

每次构建启动时，它都会克隆存储库。如果您引用具有大量历史记录的存储库，则克隆可能会需要一段时间，并使用大量的磁盘空间。另外，克隆是临时的，在创建 RPM 软件包后会被删除。

指定自定义文件系统配置

您可以在蓝图中指定自定义文件系统配置，因此创建带有特定磁盘布局的镜像，而不是使用默认的布局。通过使用蓝图中的非默认布局配置，您可以受益于：

- 安全基准合规性
- 防止磁盘不足错误
- performance
- 与现有设置的一致性

  在蓝图中自定义文件系统配置：

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

  支持以下 **挂载点** 及其子目录：

  - / - root 挂载点
  - **/var**
  - **/home**

- **/opt**
- **/srv**
- **/usr**
- **/app**
- **/data**

注意

只有在使用 CLI，RHEL 8.5 和 RHEL 9.0 发行版才支持自定义挂载点。在早期发行本中，您只能将 **root** 分区指定为挂载点，并将 **size** 参数指定为镜像大小的别名。

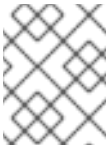
如果您有一个以上的分区，您可以在 LVM 中创建带有自定义文件系统分区的镜像，并在运行时重新定义这些分区大小。为此，您可以在蓝图中指定自定义的文件系统配置，然后使用所需的磁盘布局创建镜像。默认文件系统布局保持不变 - 如果您使用没有文件系统自定义的普通镜像，则 **cloud-init** 会调整 root 分区。

在蓝图中添加了文件系统自定义后，该文件系统将被转换为 LVM 分区。

当您定义 *MINIMUM-PARTITION-SIZE* 时，没有默认大小格式。支持以下值和单位： kB 到 TB，KiB 到 TiB。例如，您可以以字节为单位定义挂载点大小：

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

您还可以使用单元定义挂载点大小。



注意

您只能对 RHEL 8.6 和 RHEL 9.0 发行版提供的软件包版本的单元定义挂载点大小。

例如：

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

其他资源

- 在添加文件系统自定义 "size" 后，蓝图导入会失败。

4.8. 安装的软件包

当使用镜像构建器创建系统镜像时，系统默认安装一组基础软件包。软件包的基础列表是 **comps core** 组的成员。默认情况下，Image Builder 使用 **core dnf** 组。

表 4.1. 支持创建镜像类型的默认软件包

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

镜像类型	默认软件包
edge-installer	aajohan-comfortaa-fonts, abattis-cantarell-fonts, alsa-firmware, alsa-tools-firmware, anaconda, anaconda-install-env-deps, anaconda-widgets, audit, bind-utils, bitmap-fangsongti-fonts, bzip2, cryptsetup, dbus-x11, dejavu-sans-fonts, dejavu-sans-mono-fonts, device-mapper-persistent-data, dnf, dump, ethtool, fcoe-utils, ftp, gdb-gdbserver, gdisk, gfs2-utils, glibc-all-langpacks, google-noto-sans-cjk-ttc-fonts, gsettings-desktop-schemas, hdparm, hexedit, initscripts, ipmitool, iwl3945-firmware, iwl4965-firmware, iwl6000g2a-firmware, iwl6000g2bfirmware, jomolhari-fonts, kacst-farsi-fonts, kacst-qurn-fonts, kbd, kbd-misc, kdump-anaconda-addon, khmeros-base-fonts, libblockdev-lvm-dbus, libertas-sd8686-firm, libertas-sd87-firmware, libertas-usb8388-firmware, libertas-usb8388-olpc-firmware, libibverbs, libreport-plugin-bugzilla, libreport-plugin-reportuploader, libreport-rhel-anaconda-bugzilla, librsvg2, linuxfirmware, lklug-fonts, lldpad, lohit-assamese-fonts, lohit-bengali-fonts, lohit-devanagari-fonts, lohit-gujarati-fonts, lohit-gurmukhi-fonts, lohit-kannada-fonts, lohit-odia-fonts, lohit-tamil-fonts, lohit-telugu-fonts, lsof, madan-fonts, metacity, mtr, mt-st, net-tools, nmap-ncat, nm-connection-editor, nss-tools, openssh-server, oscap-anaconda-addon, pciutils, perl-interpreter, pigz, python3-pyatspi, rdma-core, redhat-release-eula, rpm-ostree, rsync, rsyslog, sg3_utils, sil-abysinica-fonts, sil-padauk-fonts, sil-scheherade-fonts, smartmontools, smc-meera-fonts, spice-vdagent, strace, system-storage-manager, thai-scalable-waree-fonts, tigervnc-server-minimal, tigervnc-server-module, udisks2, udisks2-iscsi, usbutils, vim-minimal, volume_key, wget, xfsdump, xorg-x11-drivers,xorg-x11-fonts-misc,xorg-x11-server-utils,xorg-x11-server-Xorg, xorg-x11-xauth
edge-simplified-installer	attr, basesystem, binutils, bsdtar, clevis-dracut, clevis-luks, cloud-utils-growpart, coreos-installer, coreos-installer-dracut, coreutils, device-mapper-multipath, dnsmasq, dosfstools, dracut-live, e2fsprogs, fcoe-utils, fdo-init, gzip, ima-evm-utils, iproute, iptables, iputils, iscsi-initiator-utils, keyutils, lldpad, lvm2, passwd, passwd, policycoreutils, policycoreutils-python-utils, procps-ng, rootfiles, setools-console, sudo, traceroute, util-linux

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

注意

当您在蓝图中添加其他组件时，您必须确保添加的组件中的软件包不会与任何其他软件包组件冲突，否则系统将无法解决依赖项问题。因此，您无法创建自定义镜像。

其他资源

- 镜像构建器描述

4.9. 已启用的服务

当您配置自定义镜像时，启用的服务是您运行 **osbuild-composer** 的 RHEL 版本的默认服务，另外还会为特定镜像类型启用服务。

例如，**.ami** 镜像类型启用服务 **sshd**、**chronyd** 和 **cloud-init**，如果没有这些服务，则自定义镜像不会引导。

表 4.2. 启用服务来支持镜像类型创建

镜像类型	已启用的服务
ami	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
openstack	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
qcow2	cloud-init
rhel-edge-commit	默认没有启用任何额外服务

镜像类型	已启用的服务
tar	默认没有启用任何额外服务
vhd	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk	sshd、chronyd、vmtoolsd、cloud-init

备注：您可以自定义在系统引导过程中要启用的服务。但是，对于默认启用了服务的映像类型，自定义不会覆盖此功能。

其他资源

- 支持的镜像自定义

第 5 章 使用 IMAGE BUILDER WEB 控制台界面创建系统镜像

镜像构建器是创建自定义系统镜像的工具。要控制 Image Builder 并创建自定义系统镜像，您可以使用 Web 控制台界面。请注意, 命令行界面 目前为首选，因为它提供了更多的功能。

5.1. 在 RHEL WEB 控制台中访问镜像构建器 GUI

RHEL web 控制台的 **cockpit-composer** 插件使用户能够管理镜像构建器蓝图，以及带有图形界面的 compose。请注意,控制镜像构建器的首选方法是使用命令行界面。

先决条件

- 您必须有对该系统的根权限。

流程

1. 在安装了镜像构建器的系统上，在 web 浏览器中打开 **https://localhost:9090/**。
   有关如何远程访问镜像构建器的更多信息，请参阅 *使用 RHEL web 控制台管理系统* 文档。
2. 使用系统中有足够权限的用户帐户登录到 web 控制台。
3. 要显示镜像构建器控制，请点击位于窗口左上角的 **Image Builder** 图标。
   Image Builder 视图会打开，它会列出现有的蓝图。

其他资源

- 使用镜像构建器命令行界面创建系统镜像

5.2. 在 WEB 控制台界面中创建 IMAGE BUILDER 蓝图

要描述自定义的系统镜像，请首先创建一个蓝图。

先决条件

- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。

流程

1. 点击右上角的 **Create Blueprint**。
   此时会出现一个蓝图名称和描述字段。
2. 填写蓝图名称及其描述，然后点击 **Create**。
   屏幕更改为蓝图编辑模式。
3. 添加您要包含在系统镜像中的组件：
   a. 在左侧，在 **Available Components** 字段中输入所有或部分组件的名称，然后按 **Enter**。
   搜索会被添加到文本条目字段下的过滤器列表中，以下组件列表将缩减为与搜索匹配的组件列表。

   如果组件列表过长，以同样的方式添加进一步的搜索词。

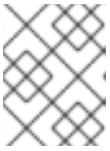
   b. 组件列表会进行分页。要移动到其他结果页面，请使用组件列表上方的箭头和条目字段。

c. 点击要用来显示其详情的组件名称。右侧窗格中会填充组件的详细信息，如组件的版本和依赖项。

d. 在 **Component Options** 框中，使用 **Version Release** 下拉菜单选择要使用的版本。

e. 点击左上角的 **Add**。

f. 如果您错误地添加了一个组件，删除它，点 **...** 按钮（位于右侧窗格中的条目右侧），然后在菜单中选择 **Remove**。



注意

如果您不想为某些组件选择版本，您可以通过点击组件列表右侧的 **+** 按钮来跳过组件详情屏幕和版本选择。

4. 要保存蓝图，请点击右上角的 **Commit**。弹出一个包含更改摘要的对话框。点 **Commit**。
右侧的一个小弹窗会告知您保存进度和结果。

5. 要退出编辑屏幕，请单击左上角的 **Back to Blueprints**。
Image Builder 视图会打开，它会列出现有的蓝图。

5.3. 在 WEB 控制台界面中编辑 IMAGE BUILDER 蓝图

要更改自定义系统镜像的规范，请编辑对应的蓝图。

先决条件

- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 一个蓝图已存在。

流程

1. 通过在左上角的搜索框中输入名称或部分名称来找到您要编辑的蓝图，然后按 **Enter**。
搜索会被添加到文本条目字段下的过滤器列表中，以下蓝图列表会缩减成与搜索匹配的列表。

   如果蓝图列表太长，以同样的的方式添加进一步的搜索。

2. 在蓝图的右侧，按属于该蓝图的 **Edit Blueprint** 按钮。
蓝图编辑屏幕会显示。

3. 点击右窗格中最右边条目处的 按钮，并在菜单中选择 **Remove** 来删除不需要的组件。

4. 更改现有组件的版本：

   a. 在蓝图组件搜索字段中，在标题 **Blueprint Components** 下的字段中输入组件名称或部分名称，然后按 **Enter**。
   搜索会被添加到文本条目字段下的过滤器列表中，以下组件列表将缩减为与搜索匹配的组件列表。

   如果组件列表过长，以同样的方式添加进一步的搜索词。

   b. 点击组件条目最右侧的 按钮，并在菜单中选择 **View**。
   在右窗格中打开组件详情屏幕。

c. 在 **Version Release** 下拉菜单中选择所需的版本，并点击右上角的 **Apply Change**。
保存更改，右侧窗格返回到列出蓝图组件。

5. 添加新组件：

a. 在左侧，在标题 **Available Components** 下输入组件名称或部分名称，然后按 **Enter** 。
搜索会被添加到文本条目字段下的过滤器列表中，以下组件列表将缩减为与搜索匹配的组件列表。

如果组件列表过长，以同样的方式添加进一步的搜索词。

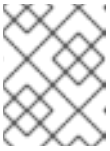
b. 组件列表会进行分页。要移动到其他结果页面，请使用组件列表上方的箭头和条目字段。

c. 点击要用来显示其详情的组件名称。右侧窗格中会填充组件的详细信息，如组件的版本和依赖项。

d. 通过 **Version Release** 下拉菜单，在 **Component Options** 框中选择您要使用的版本。

e. 点击右上角的 **Add**。

f. 如果您错误地添加了一个组件，请点击右侧窗格中最右边条目的 按钮，并在菜单中选择 **Remove**来删除它。



注意

如果您不想为某些组件选择版本，您可以通过点击组件列表右侧的 **+** 按钮来跳过组件详情屏幕和版本选择。

6. 提交带有您的更改的蓝图的新版本：

a. 点击右上角的 **Commit** 按钮。
此时会出现一个带有您更改小结的弹出窗口。

b. 点 **Commit** 检查您的更改并进行确认。
右侧的一个小弹出会告诉您保存的进程和结果。创建了新版本的蓝图。

c. 在左上角，单击 **Back to Blueprints** 退出编辑屏幕。
Image Builder 视图会打开，它会列出现有的蓝图。

5.4. 将用户和组添加到 WEB 控制台界面的 IMAGE BUILDER 蓝图中

目前无法在 web 控制台界面中将自定义（如用户和组）添加到蓝图中。要临时解决这个局限，请使用 web 控制台中的 **Terminal** 选项卡来使用命令行界面(CLI)工作流。

先决条件

- 蓝图必须存在。
- 必须安装 CLI 文本编辑器，如 **vim**、**nano** 或 **emacs**。要安装它们：

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

流程

1. 查找蓝图的名称：在 RHEL web 控制台中，打开左侧的（**Image Builder**）标签页，以查看蓝图的名称。

2. 在 Web 控制台中导航到 CLI：打开左侧的系统管理选项卡，然后从左侧的列表中选择最后一个项目**终端**。

3. 进入超级用户(root)模式：

```
$ sudo bash
```

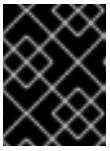
当被提示时提供您的凭证。请注意，终端不会重复使用您在登录 web 控制台时输入的凭证。

在您的主目录中会启动一个具有 root 特权的新 shell。

4. 将蓝图导出到一个文件中：

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

5. 使用文本编辑器编辑 *BLUEPRINT-NAME*.toml 文件并添加用户和组。



重要

RHEL web 控制台没有任何内置功能来编辑系统上的文本文件，因此此步骤需要使用 CLI 文本编辑器。

a. 对于要添加的每个用户，请将此块添加到文件中：

```
[[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
```

将 *PASSWORD-HASH* 替换为实际密码散列。要生成散列，请使用如下命令：

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

替换 *ssh-rsa（...）key-name*，使用实际的公钥。

使用适当的值替换其他占位符。

根据需要可以省略任何行，只需要用户名。

b. 对于要添加的每个用户组，请将此块添加到文件中：

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

c. 增加版本号。

d. 保存文件并关闭文本编辑器。

6. 将蓝图重新导入到镜像构建器中：

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

请注意，您必须提供包括 **.toml** 扩展名的文件名，而在其他命令中，您只能使用蓝图的名称。

7. 要验证上传到镜像构建器的内容是否与您的编辑匹配，请列出蓝图的内容：

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

检查版本是否与您在文件中放入的内容相匹配，以及您的自定义是否存在。

重要

RHEL web 控制台的镜像构建器插件不会显示任何信息，它们可用来验证更改是否已应用，除非您也编辑了蓝图中包含的软件包。

8. 退出特权 shell:

```
# exit
```

9. 打开左侧的镜像构建器(**Image builder**)选项卡，然后在所有浏览器和所有打开的选项卡中刷新页面。
这可防止在载入页面中缓存的状态意外恢复您的更改。

其他资源

- 镜像构建器蓝图格式
- 使用命令行界面编辑镜像构建器蓝图

5.5. 在 WEB 控制台界面中使用 IMAGE BUILDER 创建系统镜像

以下步骤描述了创建系统镜像。

先决条件

- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 一个蓝图已存在。

流程

1. 通过在左上角的搜索框中输入镜像的名称或部分名称，找到您要构建镜像的蓝图，然后按 **Enter**。
搜索会被添加到文本条目字段下的过滤器列表中，以下蓝图列表会缩减成与搜索匹配的列表。

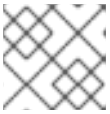
如果蓝图列表太长，以同样的的方式添加进一步的搜索。

2. 在蓝图的右侧，按属于该蓝图的 **Create Image** 按钮。
   此时会出现弹出窗口。

3. 选择镜像类型并按 **Create**。
   右上角有一个小弹出通知您已将镜像创建添加到队列中。

4. 点蓝图的名称。
   此时会打开一个包含详细蓝图的屏幕。

5. 点击 **Images** 选项卡切换到它。正在创建的镜像状态为 **In Progress**。

   注意

   镜像创建需要较长的时间（以分钟为单位）。在创建镜像时没有显示进度。

   要取消创建镜像，按右侧的 **Stop** 按钮。

6. 成功创建镜像后，**Stop** 按钮会被一个 **Download** 按钮替代。点击这个按钮将镜像下载到您的系统。

5.6. 在蓝图中添加源

Image Builder 中定义的源提供您可以添加到蓝图中的内容。这些源是全局的，因此适用于所有蓝图。系统源是本地设置且无法从 Image Builder 中删除的软件仓库。您可以添加额外的自定义源，从而能够访问系统上可用系统源之外的其他内容。

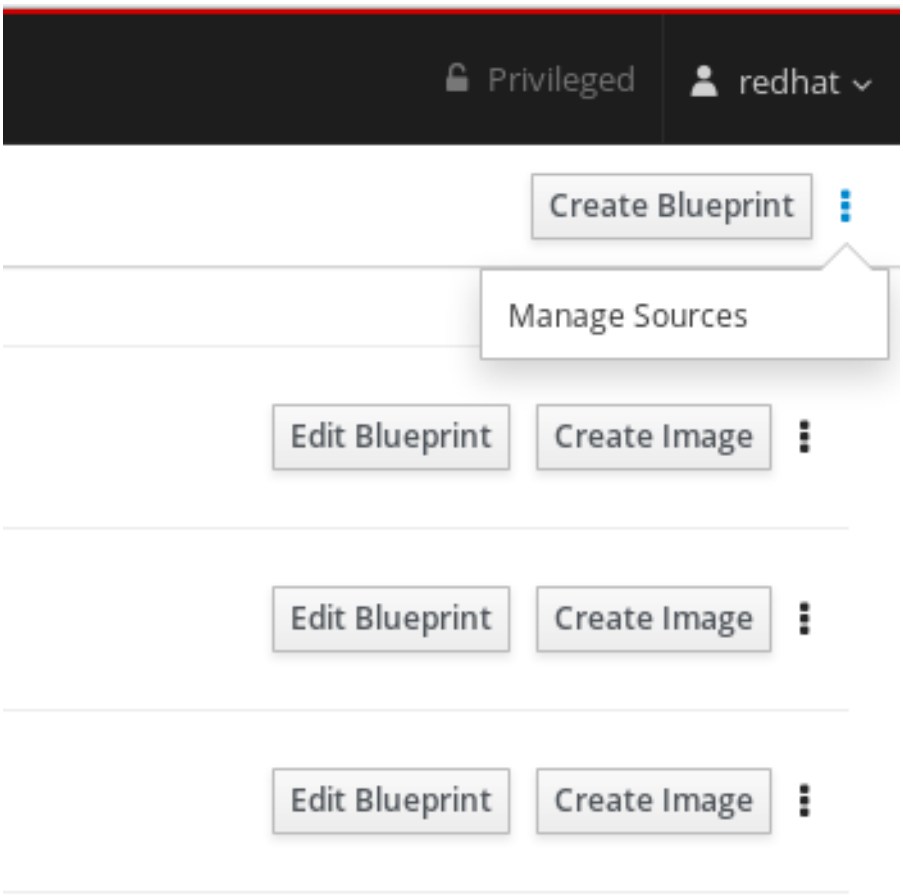
以下步骤描述了如何在本地系统中添加源。

先决条件

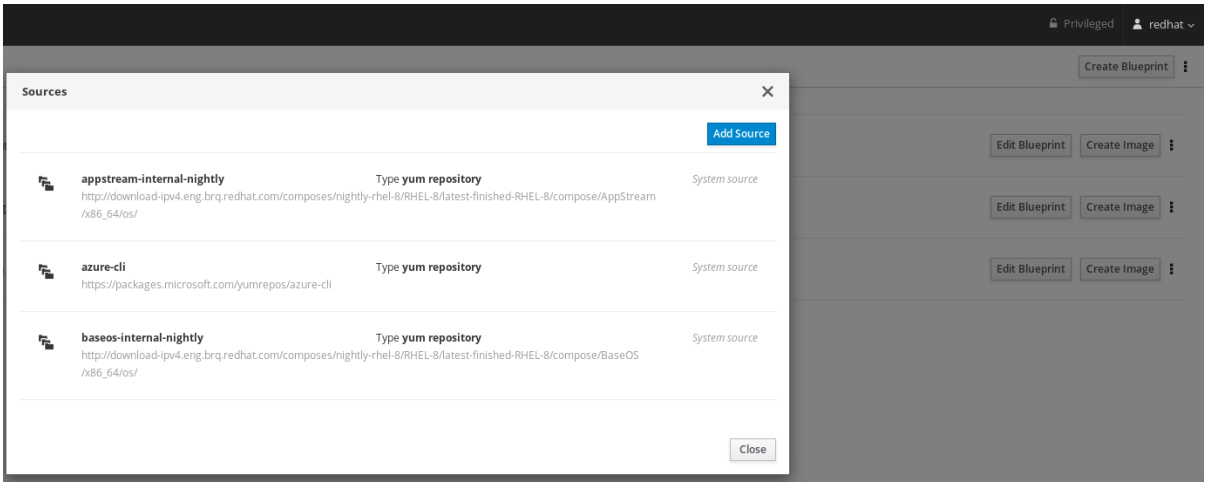
- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。

流程

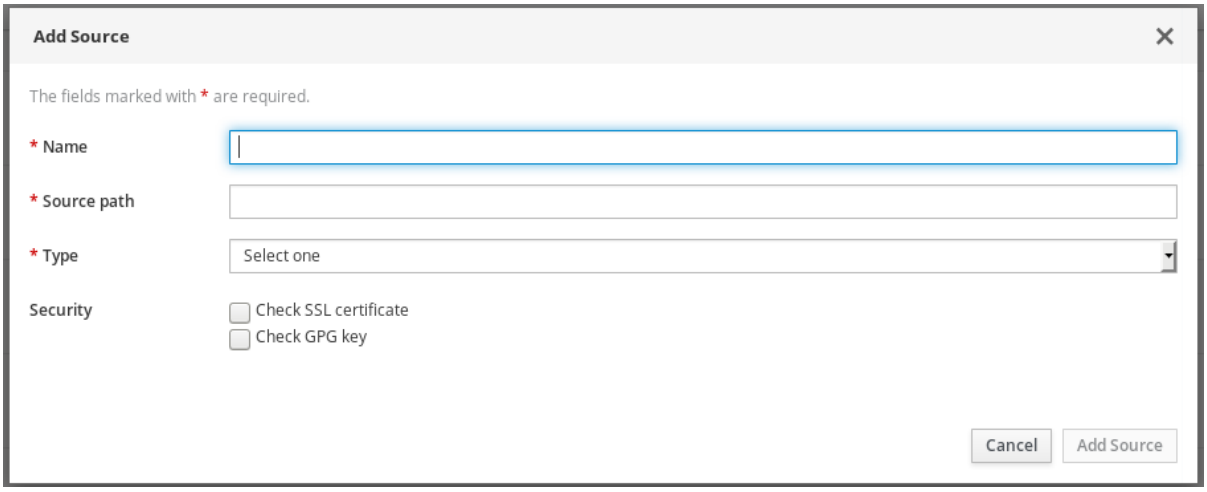
1. 点击右上角的 **Manage Sources** 按钮。



此时会出现一个带有可用源、名称和描述的弹出窗口。



2. 在弹出窗口的右侧点击 **Add Source** 按钮。
3. 添加所需的 **Source name**、**Source path** 和 **Source Type**。**Security** 字段是可选的。



4. 点击 **添加源** 按钮。屏幕中显示可用的源窗口并列出您添加的源。

因此，新的系统源可用，并可供使用或编辑。

5.7. 为蓝图创建用户帐户

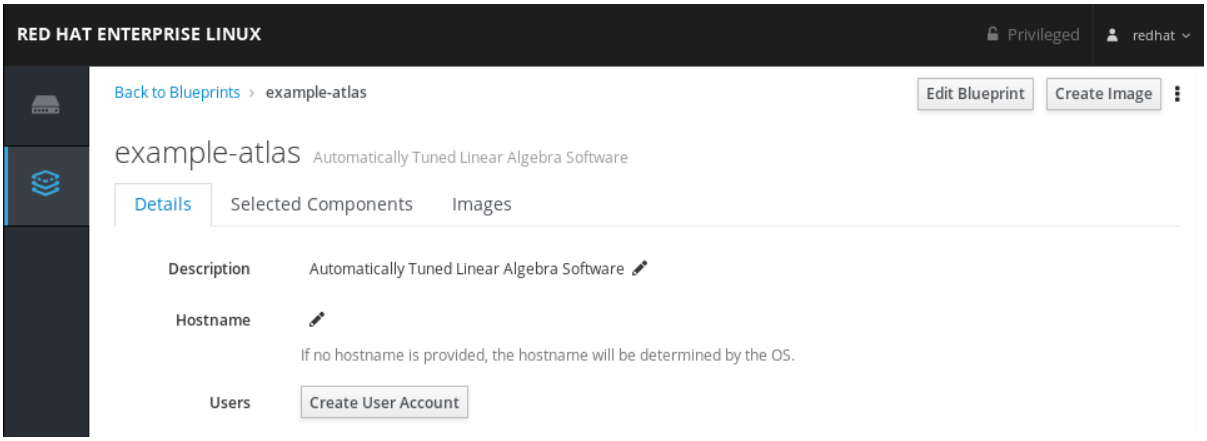
Image Builder 创建的镜像会将 root 帐户锁定，且不包括其他帐户。这样配置的目的是，避免意外地在没有密码的情况下构建和部署镜像。镜像构建器允许您为蓝图创建带有密码的用户帐户，以便您可以登录到从蓝图创建的镜像。

先决条件

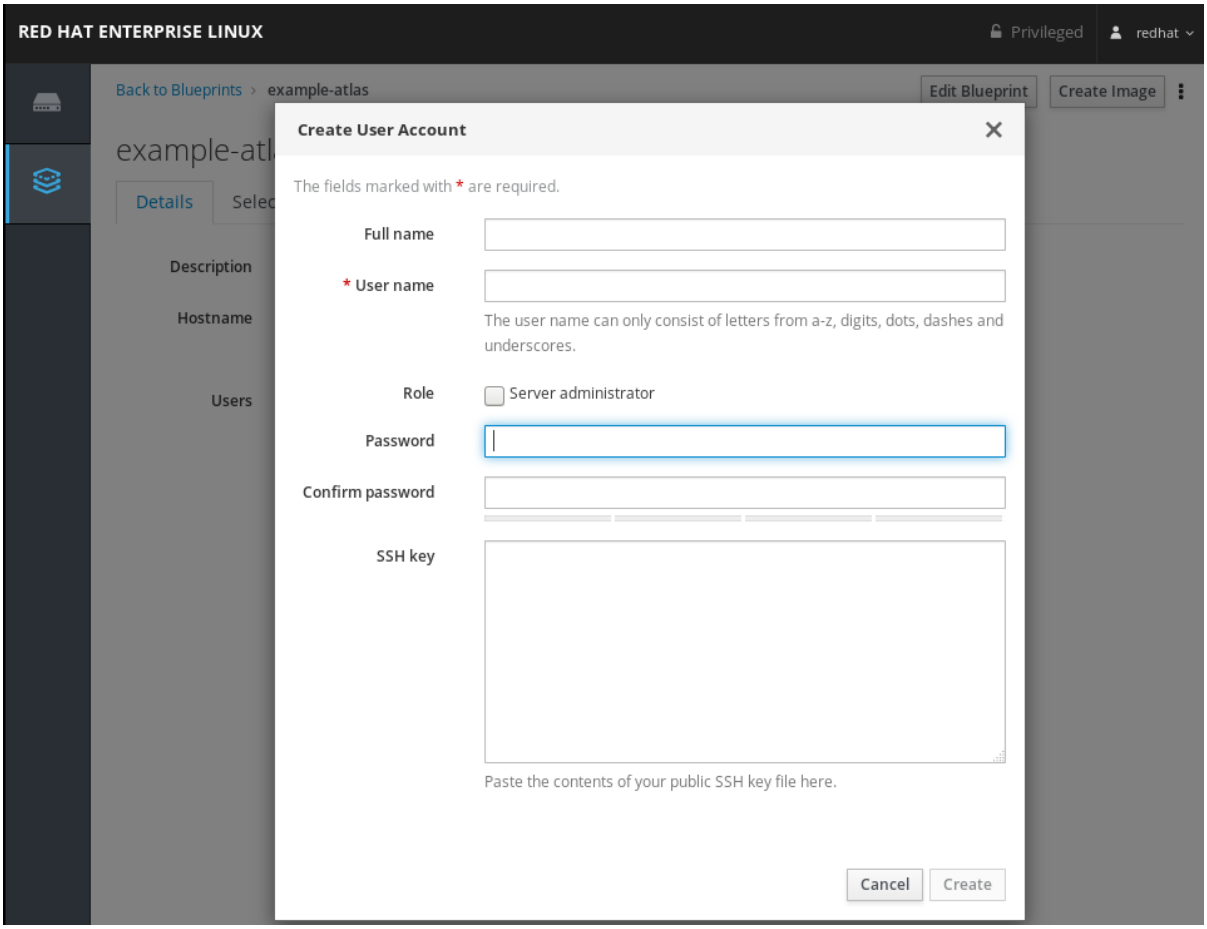
- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 您有一个现有的蓝图。

流程

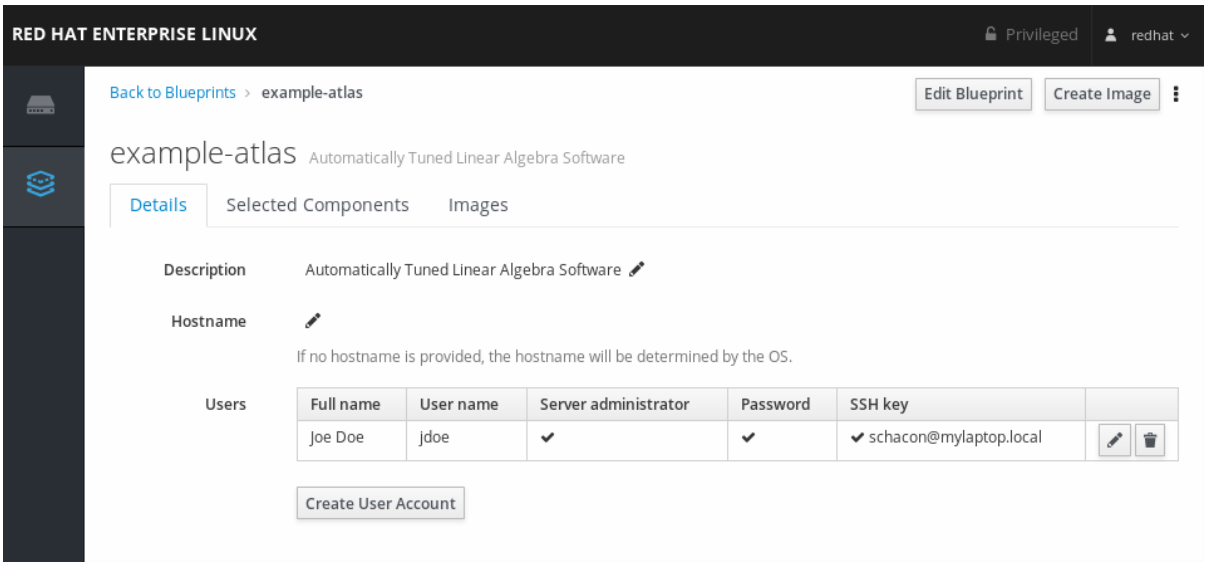
1. 通过在左上角的搜索框中输入名称或部分名称，找到您要为其创建用户帐户的蓝图，然后按 **Enter**。
   搜索会被添加到文本条目字段下的过滤器列表中，以下蓝图列表会缩减成与搜索匹配的列表。
2. 点击蓝图名称显示蓝图详情。



3. 点**创建用户帐户**。
   这将打开一个窗口，其中包含创建用户帐户的字段。



4. 填写详情。请注意，当您插入名称时，**用户名**字段会自动完成，建议使用的用户名。
5. 插入所有所需详情后，点 **Create**。
6. 创建的用户帐户会出现显示您插入的所有信息。



7. 要为蓝图创建其他用户帐户，请重复此过程。

5.8. 使用 SSH 密钥创建用户帐户

Image Builder 创建的镜像会将 root 帐户锁定，且不包括其他帐户。这个配置的目的是，确保镜像的安全，不使用默认密码。镜像构建器使您可以为蓝图创建带有 SSH 密钥的用户帐户，以便您可以对从蓝图中创建的镜像进行认证。为此，首先创建一个蓝图。然后，您将创建带有密码和 SSH 密钥的用户帐户。

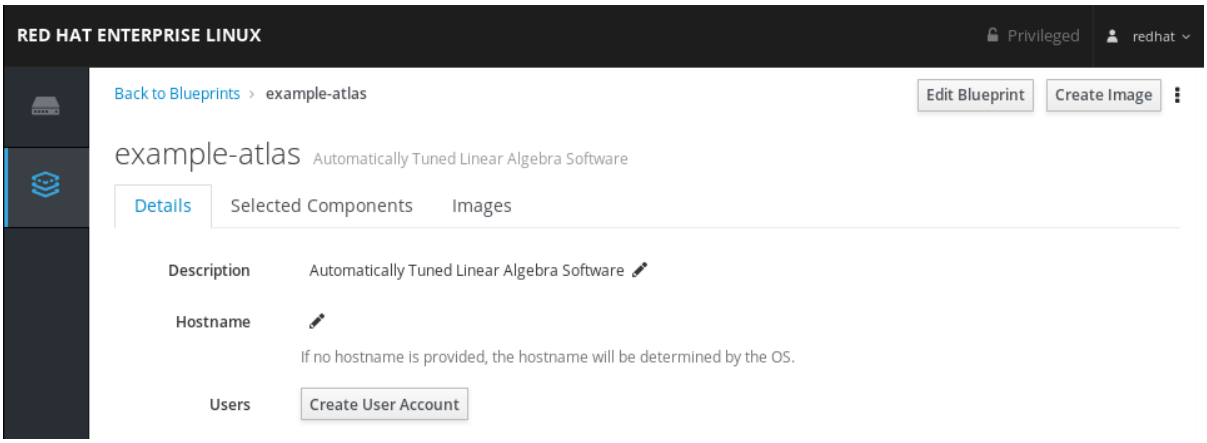
以下示例演示了如何创建配置了 SSH 密钥的服务器管理员用户。

先决条件

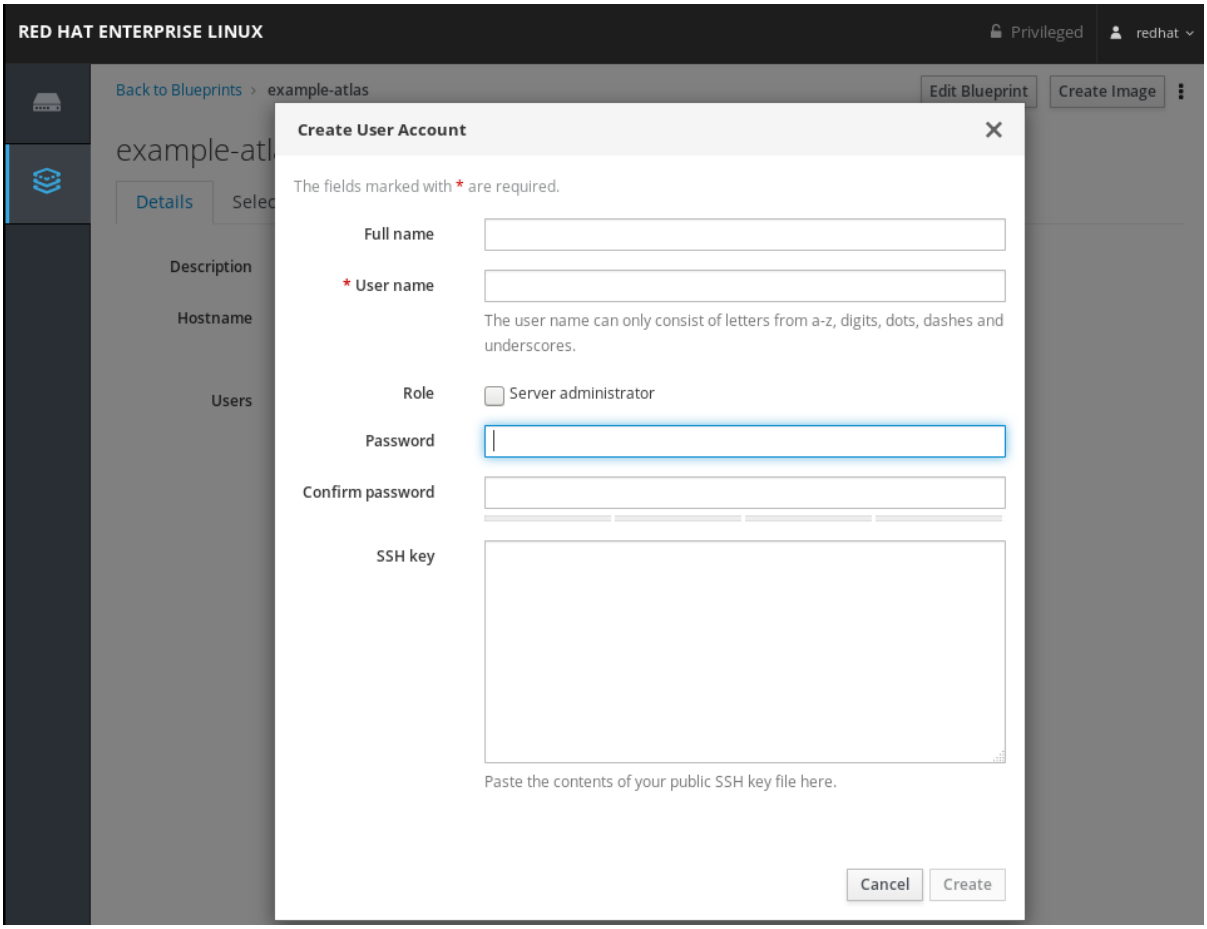
- 您已创建了在这个过程后会与创建的用户配对的 SSH 密钥。
- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 您有一个现有的蓝图

流程

1. 通过在左上角的搜索框中输入名称或部分名称，找到您要为其创建用户帐户的蓝图，然后按 **Enter**。
   搜索会被添加到文本条目字段下的过滤器列表中，以下蓝图列表会缩减成与搜索匹配的列表。
2. 点击蓝图名称显示蓝图详情。



3. 点**创建用户帐户**。
   这将打开一个窗口，其中包含创建用户帐户的字段

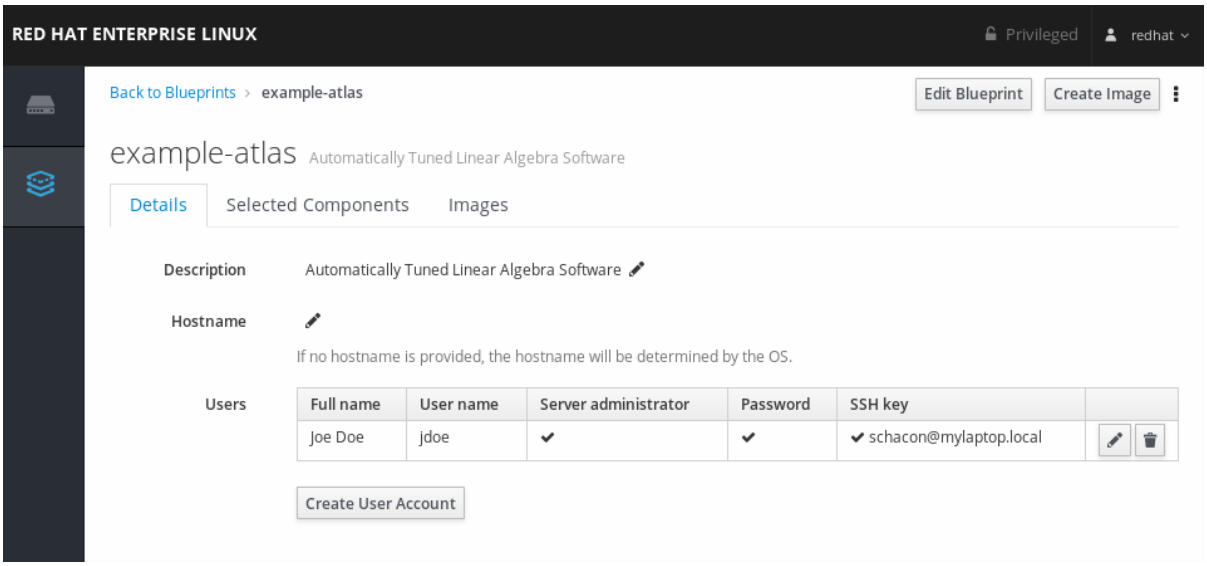


4. 填写详情。请注意，当您插入名称时，**用户名**字段会自动完成，建议使用的用户名。如果要为您正在创建的用户帐户提供管理员权限，请检查 **Role** 字段。

   粘贴 SSH 公钥文件的内容。

5. 插入所有所需详情后，点 **Create**。

6. 新用户帐户将显示在用户列表中，显示您插入的所有信息。



7. 如果要为蓝图创建更多用户帐户，请重复此过程。

其他资源

- 生成 SSH 密钥对

第 6 章 使用镜像构建器创建引导 ISO 安装程序镜像

您可以使用镜像构建器创建可引导的 ISO 安装程序镜像。这些镜像由包含根文件系统的 tar 包组成。您可以使用可引导的 ISO 镜像来将文件系统安装到裸机服务器上。

镜像构建器构建了一个清单，该清单会创建一个包含提交和根文件系统的引导 ISO。要创建 ISO 镜像，请选择新镜像类型 **image-installer**。镜像构建器构建一个 **.tar** 文件，其中包含：

- 标准 Anaconda 安装程序 ISO
- 嵌入式 RHEL 系统 tar 包
- 以最低默认要求安装提交的默认 kickstart 文件

创建的安装程序 ISO 镜像嵌入了一个预先配置的系统镜像，您可以直接将其安装到裸机服务器。

6.1. 在命令行界面上使用镜像构建器创建一个引导 ISO 安装程序镜像

此流程演示了如何使用镜像构建器命令行界面构建一个自定义引导 ISO 安装程序镜像。

先决条件

- 您为镜像创建了一个包含用户的蓝图，并将其推送回镜像构建器中。请参阅 用户的蓝图自定义。

流程

1. 创建 ISO 镜像：

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

   - 带有您创建的蓝图名称的 *BLUEPRINT-NAME*
   - *image-installer* 是镜像类型
     compose 进程在后台启动，并显示 Compose 的 UUID。

2. 等待 compose 完成。请注意，这可能需要几分钟时间。
   检查 Compose 的状态：

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

   完成的 compose 显示 **FINISHED** 状态值。根据 UUID 识别列表中的内容。

3. 完成 compose 后，下载生成的镜像文件：

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

   使用前面步骤中显示的 *UUID* 值替换 UUID。

   因此，镜像构建器会构建一个包含 ISO 安装程序镜像的 **.tar** 文件。

验证

1. 导航到下载镜像文件的文件夹。

2. 找到您下载的 **.tar** 镜像。
3. 提取 **.tar** 内容。

您可以使用在硬盘上生成的 ISO 镜像文件，或者在虚拟机中引导，例如在 HTTP 引导或 USB 安装中。

其他资源

- 使用镜像构建器命令行界面创建系统镜像
- 为 RHEL 创建可引导的安装介质

6.2. 在 GUI 中使用镜像构建器创建引导 ISO 安装程序镜像

您可以使用 Image Builder GUI 构建自定义引导 ISO 安装程序镜像。

先决条件

- 已为您的镜像创建一个蓝图。

流程

1. 在浏览器中打开 RHEL web 控制台的 Image Builder 界面。
2. 通过在左上角的搜索框中输入名称或部分来选择要构建镜像的蓝图，点 **Enter**。
3. 在蓝图的右侧，点属于蓝图的 **Create Image** 按钮。
   Create image 对话框向导将打开。
4. 在 **Create image** 对话框向导中，从 **Image Type** 列表：
   a. 选择 **"RHEL Installer (.iso)"** 镜像类型。
   b. 点 **Create**。

镜像构建器将 RHEL **.iso** 镜像的集合添加到队列中。

注意

完成镜像构建过程需要几分钟时间。

完成这个过程后，您可以看到镜像构建完成状态。镜像构建器创建 **.iso** 镜像。

验证

成功创建镜像后，您可以下载您的镜像按钮。

1. 点 **Download** 将 "RHEL Installer(.iso)" 镜像保存到您的系统。
2. 进入您下载 **"RHEL Installer(.iso)"** 镜像的文件夹。
3. 找到您下载的 .tar 镜像。
4. 提取 **"RHEL 安装程序(.iso)"** 镜像内容。

您可以使用在硬盘上生成的 ISO 镜像文件，或者在虚拟机中引导，例如在 HTTP 引导或 USB 安装中。

其他资源

- 在 web 控制台界面中创建镜像构建器蓝图。
- 使用镜像构建器命令行界面创建系统镜像。
- 为 RHEL 创建可引导安装介质。

6.3. 将 ISO 镜像安装到裸机系统

此流程演示了如何使用命令行界面，将使用镜像构建器创建的可引导 ISO 镜像安装到裸机系统上。

先决条件

- 已使用镜像构建器创建了可引导的 ISO 镜像。
- 您已下载并提取了可引导的 ISO 镜像。
- 您有一个 8 GB USB 闪存。

注意

ISO 可能很大，具体取决于您在蓝图中所选择的软件包。

流程

1. 将可引导的 ISO 镜像文件放在 USB 闪存中。
2. 将 USB 闪存连接到您要引导的计算机的端口。
3. 从 USB 闪存引导 ISO 镜像。
4. 执行步骤来安装自定义的可引导的 ISO 镜像。
   引导屏幕显示以下选项：
   - 安装 Red Hat Enterprise Linux 9
   - 测试这个介质并安装 Red Hat Enterprise Linux 9

其他资源

- 引导安装

第 7 章 使用镜像构建器准备和部署 KVM 客户机镜像

客户可以从 ISO 手动创建镜像，或者有一个使用镜像构建器创建的专用镜像。此流程描述了使用镜像构建器创建专用镜像的步骤。这仅限于对 Red Hat Virtualization(RHV)的 **rhel-guest-image** 支持。

创建自定义 KVM 客户机镜像涉及以下高级步骤：

1. 使用镜像构建器创建 KVM 客户机 **.qcow2** 镜像。
2. 从 KVM 客户机镜像创建虚拟机。

7.1. 使用镜像构建器创建自定义 KVM 客户机镜像

这描述了使用镜像构建器创建 **.qcow2** KVM 客户机镜像的步骤。

先决条件

- 您必须有访问系统的 root 或 wheel 组用户权限。
- **cockpit-composer** 软件包已安装。
- 在 RHEL 系统上，您已打开 Cockpit UI 的镜像构建器仪表盘。

流程

1. 点击 **Create blueprint** 来创建蓝图。请参阅 在 web 控制台界面中创建镜像构建器蓝图。
2. 选择作为您要创建的 KVM 客户机镜像一部分的组件和软件包。
3. 点击 **Commit** 提交您对蓝图所做的更改。右侧的一个小弹窗会告知您保存的进度，然后是提交的更改的结果。
4. 点击左侧横幅上的蓝图名称链接。
5. 选择 **Images** 选项卡。
6. 单击 **Create Image** 来创建自定义镜像。此时会打开弹出窗口。
7. 从 **Type** 下拉菜单中选择 'QEMU Image(.qcow2)'镜像。
8. 在实例化时设置您想要的镜像大小，并点击 **Create**。
9. 窗口右上角的小弹窗通知您已将镜像创建添加到队列中。镜像创建过程完成后，您可以看到**镜像构建完成**状态。

验证步骤

1. 点击 breadcrumbs 图标，并选择 **Download** 选项。镜像构建器会在您的默认下载位置下载 KVM 客户机镜像 **.qcow2** 文件。

其他资源

- 在 web 控制台界面中创建镜像构建器蓝图。

7.2. 从 KVM 客户机镜像创建虚拟机

要在主机上快速创建占用空间较少的虚拟机，您可以使用 KVM 客户机镜像。此流程使用镜像构建器生成的 KVM 客户机镜像作为 **.qcow2** 镜像格式来创建虚拟机(VM)。使用镜像构建器创建的 KVM 客户机镜像已安装并启用了 **cloud-init**。

先决条件

- 已使用镜像构建器创建了一个 **.qcow2** 镜像。请参阅 在 web 控制台界面中创建镜像构建器蓝图。
- **qemu-kvm** 软件包已安装在您的系统上。您可以检查 **/dev/kvm** 文件夹是否在您的系统上可用。
- 在您的系统上已安装了 **libvirt**。
- 在您的系统上已安装了 **virt-install**。
- **genisoimage** 工具已安装在您的系统上。

流程

1. 使用镜像构建器将您创建的 KVM 客户机镜像移到 **/var/lib/libvirt/images** 目录，并将镜像名称重命名为 **rhel-8.4-x86_64-kvm.qcow2**。
2. 创建一个目录，如 **cloudinitiso** ，并导航到这个新创建的目录：

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

3. 创建一个名为 **meta-data** 的文件。在此文件中添加以下信息：

```
instance-id: citest
local-hostname: citest-1
```

4. 创建一个名为 **user-data** 的文件。在文件中添加以下信息：

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

其中,

- **ssh_authorized_keys** 是您的 SSH 公钥。您可以在 **~/.ssh/id_rsa.pub** 中找到 SSH 公钥。

5. 使用 **genisoimage** 命令创建一个包含 **user-data** 和 **meta-data** 文件的 ISO 镜像。

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

I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 331
Total directory bytes: 0
```

```
Path table size(bytes): 10
Max brk space used 0
183 extents written (0 MB)
```

6. 使用 **virt-install** 命令从 KVM 客户机映像创建一个新虚拟机。将您在第 4 步中创建的 ISO 镜像作为虚拟机镜像的附件。

```
# virt-install \
    --memory 4096 \
    --vcpus 4 \
    --name mytestcivm \
    --disk /var/lib/libvirt/images/rhel-8.4-x86_64-
kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
    --disk /home/sample/cloudinitiso/ciiso.iso,device=cdrom \
    --os-variant rhel8.4 \
    --virt-type kvm \
    --graphics none \
    --import
```

其中,

- --graphics none - 表示它是一个无头的 RHEL 8.4 虚拟机。
- --vcpus 4 - 表示它使用 4 个虚拟 CPU。
- --memory 4096 - 表示它使用 4096 MB RAM。

7. 虚拟机安装开始：

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

Red Hat Enterprise Linux 8.4 Beta (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

验证

1. 使用 **cloud-user** 作为用户名登录到创建的虚拟机。您的密码为 **cilogon**。

其他资源

- 请参阅 启用虚拟化。
- 请参阅 为 RHEL 9 配置和管理 cloud-init 。
- 请参阅 cloud-init 重要目录和文件。

第 8 章 使用镜像构建器准备并上传云镜像

镜像构建器可以创建自定义系统镜像，以便用于各种供应商的云中。要在云中使用自定义的 RHEL 系统镜像，请使用相应的输出类型通过镜像构建器创建系统镜像，配置您的系统以上传镜像，并将镜像上传到您的云帐户。从 Red Hat Enterprise Linux 8.3 开始，我们支持的一些服务提供商（如 **AWS** 和 **Azure** 云）可以通过 RHEL web控制台中的 **Image Builder** 应用程序来推送定制的镜像云。请参阅 将镜像推送到 AWS Cloud AMI 和 将 VHD 镜像推送到 Azure 云。

8.1. 准备上传 AWS AMI 镜像

这描述了配置系统来上传 AWS AMI 镜像的步骤。

先决条件

- 您必须在 AWS IAM account manager 中配置了一个 Access Key ID。
- 您必须具有一个可写的 S3 存储桶。

流程

1. 安装 Python 3 和 **pip** 工具：

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

2. 使用 **pip** 安装 AWS 命令行工具 ：

```
# pip3 install awscli
```

3. 运行以下命令设定您的配置集。终端提示您提供凭证、地区和输出格式：

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

4. 为存储桶定义名称，并使用以下命令创建存储桶：

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

使用实际存储桶名称替换 *bucketname*。它必须是全局唯一的名称。因此，您的存储桶会被创建。

5. 然后，要授予访问 S3 存储桶的权限，请在 IAM 中创建 **vmimport** S3 角色（如果您之前还没有这样做的话）：

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service":
"vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{
"sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
$ printf '{ "Version":"2012-10-17", "Statement":[ { "Effect":"Allow", "Action":[
"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource":[ "arn:aws:s3:::%s",
```

```
"arn:aws:s3:::%s/*" ] }, { "Effect":"Allow", "Action":[ "ec2:ModifySnapshotAttribute",
"ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource":"*" } ] }' $BUCKET
$BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

其他资源

- 使用 AWS CLI 中的高级(s3)命令

8.2. 在 CLI 中将 AMI 镜像上传到 AWS

您可以使用镜像构建器来构建 **.ami** 镜像，并使用 CLI 将其直接推送到 Amazon AWS 云服务供应商。

先决条件

- 您已在 AWS IAM 账号管理器中配置了一个 **Access Key ID**。
- 您已准备好了一个可写的 S3 存储桶。
- 您有一个定义的蓝图。

流程

1. 使用文本编辑器，使用以下内容创建配置文件：

```
provider = "aws"

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

   将字段中的值替换为您的 **accessKeyID**、**secretAccessKey**、**bucket**、**region** 的凭证。**IMAGE_KEY** 值是要上传到 EC2 的虚拟机镜像的名称。

2. 将文件保存为 *CONFIGURATION-FILE*.toml，然后关闭文本编辑器。

3. 启动 compose：

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

   替换：

   - 将 *BLUEPRINT-NAME* 替换为您创建的蓝图名称
   - 将 *IMAGE-TYPE* 替换为 **ami** 镜像类型。
   - 将 *IMAGE_KEY* 替换为要上传到 EC2 的虚拟机镜像的名称。

- 将 *CONFIGURATION-FILE*.toml 替换为云供应商的配置文件的名称。

注意

对于要将自定义镜像发送到的存储桶，您必须拥有正确的 IAM 设置。在将镜像上传到存储桶前，您必须对存储桶设置策略。

4. 检查镜像构建的状态，并将其上传到 AWS：

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

完成镜像上传过程后，您可以看到"FINISHED"状态。

验证

要确认镜像上传成功：

1. 访问菜单中的 EC2，并在 AWS 控制台中选择正确的区域。镜像必须具有 "available" 状态，以表示它已被成功上传。
2. 在仪表盘上，选择您的镜像并点击 **Launch**。

其它资源

- 所需的服务角色

8.3. 将镜像推送到 AWS CLOUD AMI

当前提供了将您创建的输出镜像推送到 **AWS Cloud AMI**的功能。这描述了将使用镜像构建器创建的 **.ami** 镜像推送到 Amazon AWS Cloud 服务提供商的步骤。

先决条件

- 您必须有访问系统的 **root** 或 **wheel** 组用户权限。
- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 您必须在 AWS IAM account manager 中配置了一个 Access Key ID。
- 您必须具有一个可写的 S3 存储桶。

流程

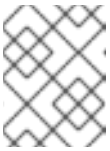
1. 点击 **Create blueprint** 来创建蓝图。请参阅 web 控制台界面中的创建镜像构建器蓝图。
2. 选择您要作为您要创建的镜像一部分的组件和软件包。
3. 点击 **Commit** 提交您对蓝图所做的更改。
   右侧的一个小弹窗会告知您保存的进度，然后是提交的更改的结果。
4. 点击左侧标题上的 **蓝图名称** 链接。
5. 选择选项卡 **Images**。
6. 单击 **Create Image** 来创建自定义镜像。

此时会打开弹出窗口。

a. 在 **"Type"** 下拉菜单中选择“Amazon Machine Image Disk(.ami)”镜像。

b. 选中 **"Upload to AWS"** 复选框，将您的镜像上传到 AWS 云,并点击 **Next**。

c. 要验证您是否可以访问 AWS ，请在对应的字段中输入您的"AWS access key ID"和"AWS secret access key"。点击 **Next**。



注意

您只能在创建新 Access Key ID 时查看 AWS secret access key 。如果您不知道您的 Secret Key，请生成一个新的 Access Key ID。

d. 在"Image name"字段中输入镜像的名称，在 "Amazon S3 bucket name" 字段中输入 Amazon 存储桶的名称，并为您要将自定义镜像添加到的存储桶输入"AWS region"字段。点击 **Next**。

e. 查看信息并点 **Finish**。
（可选）您可以点击 **Back** 来修改任何不正确的详情。

注意

您必须具有要发送自定义镜像的存储桶的正确 IAM 设置。我们使用 IAM 导入和导出，因此您必须在将镜像上传到存储桶前为存储桶设置 **一个策略** 。如需更多信息，请参阅 IAM 用户所需的权限。

7. 在右侧的一个小弹出可让您了解保存的过程。它还告知镜像创建过程、创建此镜像的过程以及后续的上传到 AWS Cloud。
过程完成后，您可以看到 **"Image build complete"**状态。

8. 点菜单中的 Service→EC2，然后在 AWS 控制台中选择 正确的区域。镜像必须具有"Available"状态才能指示它已被上传。

9. 在仪表盘上，选择您的镜像并点击 **Launch**。

10. 此时会打开一个新窗口。根据启动镜像所需的资源选择实例类型。点击 **Review and Launch**。

11. 查看您的实例启动详情。如果需要进行任何更改，您可以编辑每个部分。点击 **Launch**

12. 在启动实例之前，您必须选择一个访问它的公钥。
您可以使用您已有的密钥对，也可以创建一个新的密钥对。另外，您可以使用 **镜像构建器** 来将用户添加到带有预设公钥的镜像中。如需了解更多详细信息，请参阅 创建带有 SSH 密钥的用户帐户。

    按照以下步骤在 EC2 中创建新的密钥对，并将它连接到新实例。

    a. 在下拉菜单中选择 **"Create a new key pair"**。

    b. 输入新密钥对名称。它生成一个新的密钥对。

    c. 点击 **"下载密钥对"** 在您的本地系统中保存新密钥对。

13. 然后，您可以单击 **Launch Instance** 来启动您的实例。
您可以检查实例的状态，它显示为 **"Initializing"**。

14. 实例状态变为 **"运行"** 后，**连接**按钮将变为可用。
15. 点 **连接**。此时会出现一个弹出窗口并给出如何使用 SSH 连接的说明。
    a. 选择 **"A standalone SSH client"**的首选连接方法并打开终端。
    b. 在您存储私钥的位置，确保您的密钥是公开可见的，以便 SSH 可以正常工作。要做到这一点，请运行以下命令：

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

    c. 使用其公共 DNS 连接到您的实例：

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

    d. 输入 "yes" 来确认您要继续连接。
    因此，您使用 SSH 连接到实例。

验证步骤

1. 检查在使用 SSH 连接到实例后您是否能够执行任何操作。

其他资源

- 在红帽客户门户网站中创建一个问题单
- 使用 SSH 连接到您的 Linux 实例
- 创建支持问题单

8.4. 准备上传 AZURE VHD 镜像

这描述了将 VHD 镜像上传到 Azure 的步骤。

先决条件

- 您必须具有可用的 Azure 资源组和存储帐户。

流程

1. 安装 python2:

```
# dnf install python2
```

注意

必须安装 **python2** 软件包，因为 AZ CLI 特别依赖于 python 2.7

2. 导入 Microsoft 存储库密钥：

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

3. 创建本地 azure-cli 存储库信息：

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

4. 安装 Azure CLI：

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

注意

下载的 Azure CLI 软件包版本可能会根据当前下载的版本而有所不同。

5. 运行 Azure CLI：

```
$ az login
```

终端会显示信息，'Note, we have launched a browser for you to login.对于以前使用设备代码的经验，请使用 "az login --use-device-code"'，并打开您可以在其中登录的浏览器。

注意

如果您正在运行远程(SSH)会话，则该链接不会在浏览器中打开。在这种情况下，您可以使用提供的链接，从而可以登录并验证您的远程会话。要登录,请使用网页浏览器打开页面 https://microsoft.com/devicelogin 并输入代码 *XXXXXXXXX* 进行验证。

6. 列出 Azure 中存储帐户的密钥：

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

将 *resource-group-name* 替换为 Azure 资源组的名称, *storage-account-name* 替换为 Azure 存储帐户的名称。

注意

您可以使用以下命令列出可用资源：

```
$ az resource list
```

7. 记录上一个命令输出中的 **key1** 值，并将其分配给环境变量：

```
$ KEY1=value
```

8. 创建存储容器：

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

将 *storage-account-name* 替换为存储帐户的名称。

其他资源

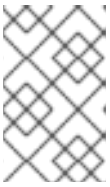
- Azure CLI

8.5. 将 VHD 镜像上传到 AZURE

这描述了将 VHD 镜像上传到 Azure 的步骤。

先决条件

- 必须设置您的系统才能上传 Azure VHD 镜像。
- 您必须具有 Image Builder 创建的 Azure VHD 镜像。在创建镜像时，在 CLI 中使用 **vhd** 输出类型或 在 GUI 中使用 **Azure Disk Image(.vhd)**。



注意

在使用 CLI 创建 **.vhd** 镜像时，Image Builder 会将临时文件写入 **/var** 子目录。为了避免 **.vhd** 镜像创建失败，请将 **/var** 子目录容量增加到至少 15 到 20 GB 的可用空间，以确保可用性。

流程

1. 将镜像推送到 Azure，并从其创建一个实例：

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

2. 上传到 Azure BLOB 后，从其中创建一个 Azure 镜像：

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

3. 使用 Azure 门户或类似如下的命令来创建实例：

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

4. 通过 SSH 使用您的私钥访问生成的实例。以 **azure-user** 用户身份登录。

8.6. 将 VMDK 镜像上传到 VSPHERE

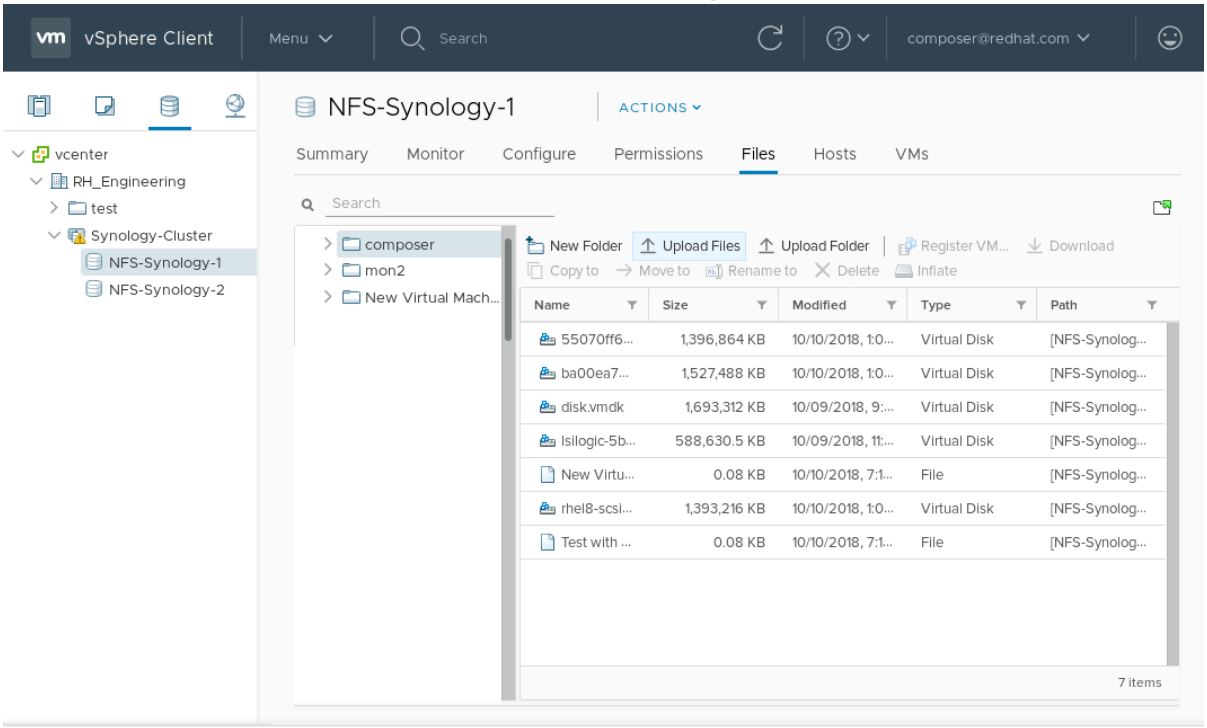
镜像构建器可以生成适合上传到 VMware ESXi 或 vSphere 系统的镜像。这描述了将 VMDK 镜像上传到 VMware vSphere 的步骤。

先决条件

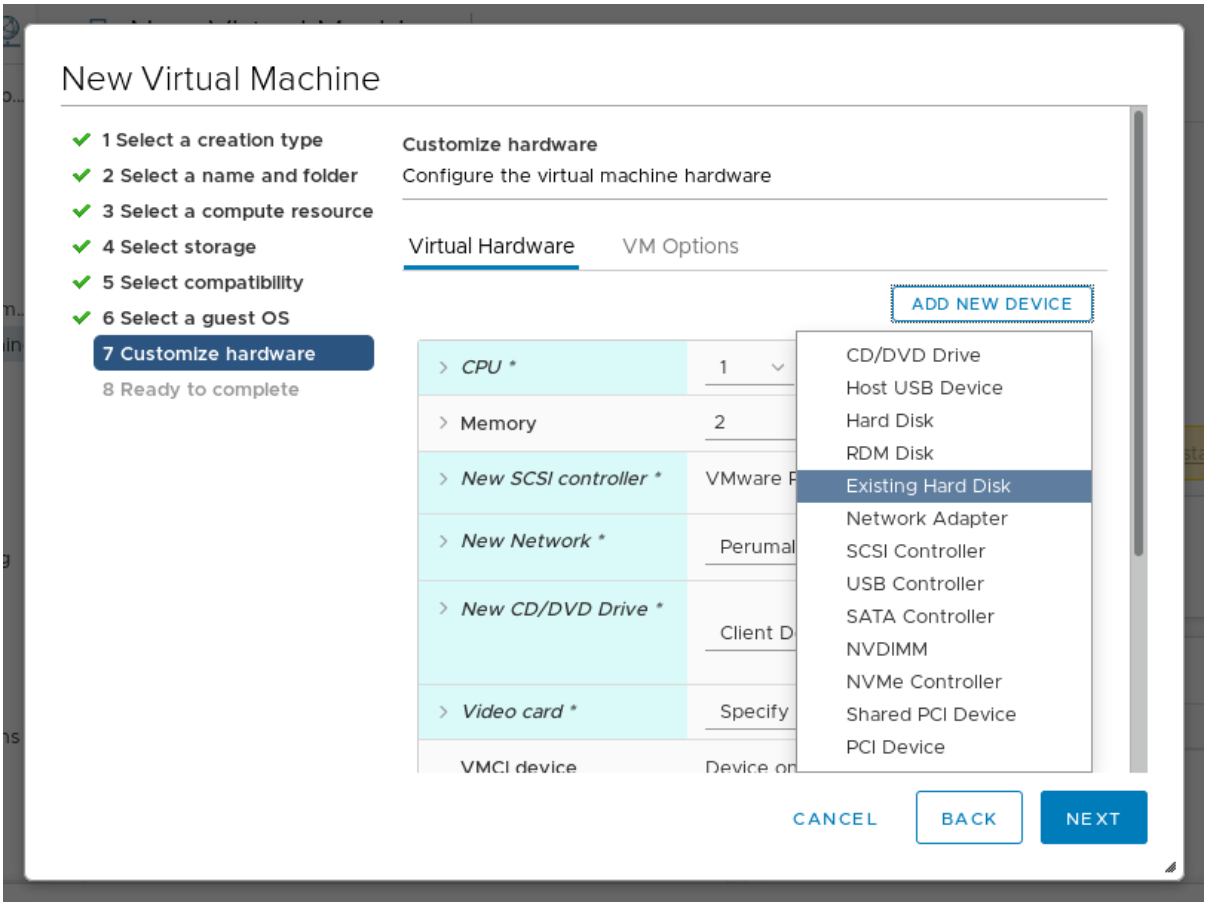
- 您必须具有由 Image Builder 创建的 VMDK 镜像。在创建镜像时，在 CLI 中使用**vmdk** 输出类型或 在 GUI 中使用 **VMware Virtual Machine Disk（.vmdk）**。

流程

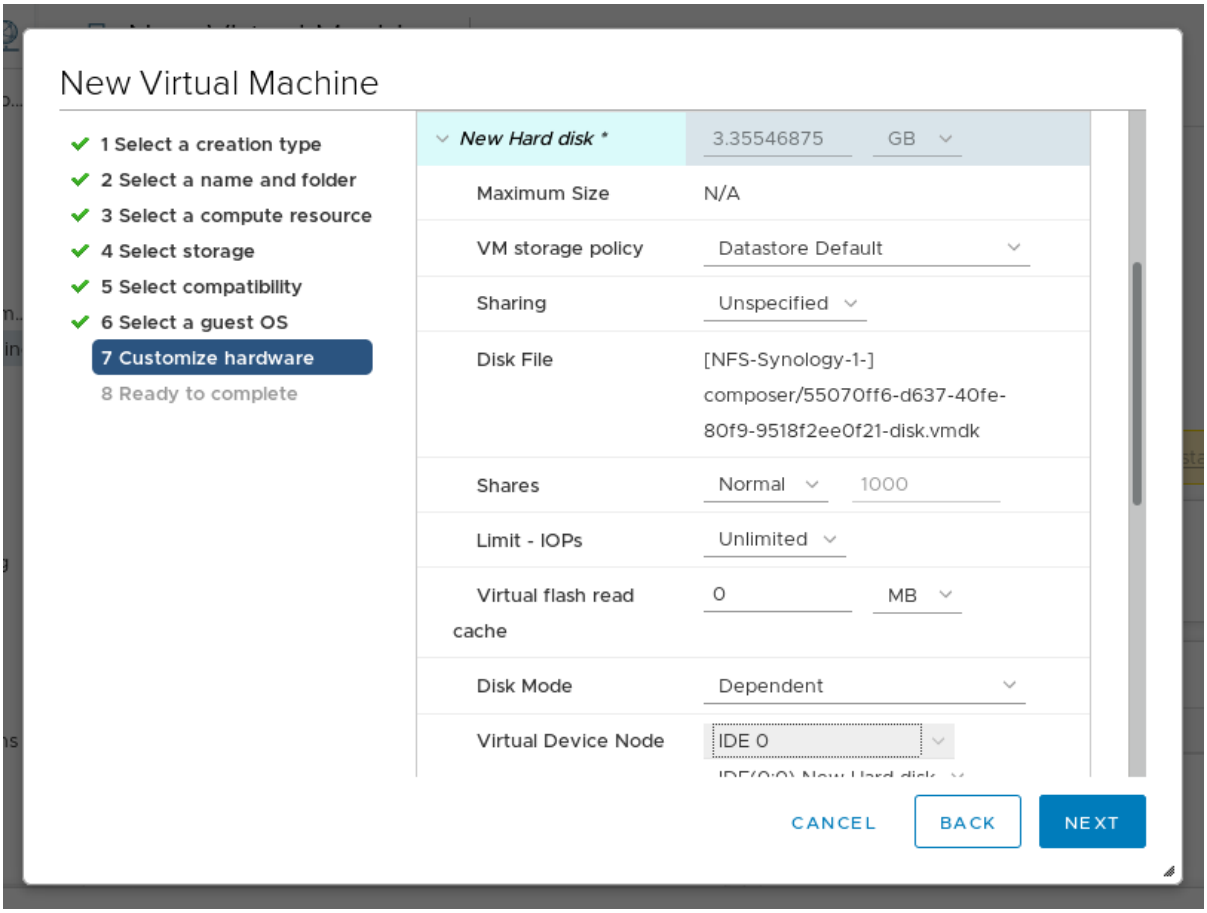
1. 通过 HTTP 将镜像上传到 vSphere。点击 vCenter 中的 **Upload Files**:



2. 当您创建虚拟机时，在 **Device Configuration** 上，删除默认的 **New Hard Disk**，并使用下拉菜单选择 **Existing Hard Disk** 磁盘镜像：



3. 确保使用 **IDE** 设备作为您所创建磁盘的 **虚拟磁盘节点** 。默认值 **SCSI** 会导致虚拟机无法启动。



8.7. 将 VMWARE 镜像推送到 VSPHERE

您可以构建 VMware 镜像，并将它们直接推送到 vSphere 实例，以避免下载镜像文件并手动推送它。这描述了将使用镜像构建器直接创建的 **.vmdk** 镜像推送到 vSphere 实例服务提供商的步骤。

先决条件

- 您有可以访问系统的 **root** 或 **wheel** 组用户权限。
- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 您有 vSphere 帐户。

流程

1. 点击 **Create blueprint**。
   请参阅 在 web 控制台界面中创建镜像构建器蓝图。
2. 选择您要作为您要创建的镜像一部分的组件和软件包。
3. 点击 **Commit** 提交您对蓝图所做的更改。
   右上角的弹窗会告知您保存的进度，然后显示您提交的更改的结果。
4. 点击左侧标题上的 **blueprint name** 链接。
5. 选择 **Customizations** 选项卡，来为蓝图创建用户帐户。
   请参阅 为蓝图创建用户帐户。
6. 选择 **Images** 选项卡。
7. 单击 **Create Image** 来创建自定义镜像。
   此时将打开镜像类型窗口。
8. 在 **Image type** 窗口中：
   a. 从下拉菜单中，选择 Type:VMware VSphere(.vmdk)。
   b. 选中 **Upload to VMware** 复选框，来将镜像上传到 vSphere。
   c. 可选：设置您要实例化的镜像的大小。最小的默认大小为 2GB。
   d. 点击 **Next**。
9. 在 **Upload to VMware** 窗口中，在 **Authentication** 下输入以下详情：
   a. Username：vSphere 帐户的用户名。
   b. Password：vSphere 帐户的密码。
10. 在 **Upload to VMware** 窗口中，在 **Destination** 下输入以下详情：
    a. **Image name**：要上传的镜像的名称。
    b. **主机** ：上传镜像的 VMware vSphere 的 URL。
    c. **集群** ：上传镜像的集群名称。
    d. **数据中心** ：上传镜像的数据中心的名称。

e. **Data store** ：镜像将要上传到的数据存储的名称。

f. 点击 **Next**。

11. 在 **Review** 窗口中，查看有关镜像创建的详情，并点 **Finish**。
您可以点击 **Back** 来修改任何不正确的详情。

镜像构建器将 RHEL vSphere 镜像的 compose 添加到队列中，并创建镜像，并将其上传到您指定的vSphere 实例的集群上。

注意

镜像构建和上传过程需要几分钟时间才能完成。

完成这个过程后，您可以看到 **镜像构建完成**状态。

验证

成功完成镜像上传后，您可以从上传的镜像创建虚拟机(VM)，并登录到虚拟机。按照以下步骤操作：

1. 访问 VMware vSphere 客户端。
2. 在您指定的 vSphere 实例上的集群中搜索镜像。
3. 您可以从上传的镜像创建新的虚拟机。为此：
   a. 选择您上传的镜像。
   b. 单击所选镜像右边的按钮。
   c. 点击 New Virtual Machine。
   此时 **New Virtual Machine** 窗口打开。

   New Virtual Machine 窗口中提供了以下详情：

      i. 选择创建类型：您可以选择创建新虚拟机。
      ii. 选择一个名称和文件夹：例如，虚拟机名称：*vSphere Virtual Machine* 以及 vSphere 客户端中您选择的位置。
      iii. 选择计算资源：为此操作选择一个目标计算资源。
      iv. 选择存储：例如，选择 NFS-Node1
      v. 选择兼容性：该镜像应仅为 BIOS。
      vi. 选择客户端操作系统：例如，选择 *Linux* 和 _Red Hat Fedora（64 位）。
      vii. **自定义硬件** ：创建虚拟机时，在右上角的 **Device Configuration**按钮，删除默认的 New Hard Disk，并使用下拉菜单来选择 Existing Hard Disk 磁盘镜像：
      viii. 准备完成：查看详情并点 **Finish** 创建镜像。

   d. 导航至 **VMs** 选项卡。
      i. 从列表中选择您创建的虚拟机。

ii. 单击面板上的 **Start** 按钮。此时会显示一个新窗口，显示正在加载 VM 镜像。

iii. 使用您为蓝图创建的凭证登录。

iv. 您可以验证添加到蓝图中的软件包是否已安装。例如：

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

其他资源

- 安装 vSphere 客户端。

8.8. 将 VHD 镜像推送到 AZURE 云

您可以使用 Image Builder 创建 **.vhd** 镜像。然后，您可以将 **.vhd** 镜像推送到 Azure Cloud 服务供应商的 Blob 存储。

先决条件

- 您必须有对该系统的根权限。
- 在浏览器中打开了 RHEL web 控制台的镜像构建器界面。
- 您必须创建一个存储帐户。
- 您必须准备可写入 Blob Storage。

流程

1. 点击 **Create blueprint** 来创建蓝图。请参阅 web 控制台界面中的创建镜像构建器蓝图。
2. 选择您要作为您要创建的镜像一部分的组件和软件包。
3. 点 **Commit** 提交您对蓝图所做的更改。
   右上角的一个小弹窗通知您保存的进度，然后通知您所提交的更改的结果。
4. 点击左侧标题上的 **blueprint name** 链接。
5. 选择选项卡 **Images**。
6. 单击 **Create Image** 来创建自定义镜像。
   此时会打开弹出窗口。
   a. 在 **"Type"** 下拉菜单中选择 **Azure Disk Image(.vhd)** 镜像。
   b. 选中 **"Upload to Azure"** 复选框，来将您的镜像上传到 Azure Cloud，并点 **Next**。
   c. 要验证您是否可以访问 Azure，请在对应的字段中输入您的"存储账号"和"存储访问密钥"。点击 **Next**。
      您可以在 Settings→Access Key 菜单列表中找到您的 存储帐户详情。
   d. 键入要用于上传的镜像文件的 **"Image name"**，以及您要将镜像推送到的 Blob "存储容器"。
      点击 **Next**。
   e. 查看您提供的信息，然后点 **Finish**。

（可选）您可以点击 **Back** 来修改任何不正确的详情。

7. 当镜像创建过程以消息开头时，右上角会显示一个小的弹出窗口："镜像创建已添加到队列"。
镜像创建过程完成后，点您从其创建镜像的蓝图。在 **Images** 选项卡中您可以看到所创建的镜像的 "Image build complete" 状态。

8. 要访问推送到 Azure Cloud 的镜像，请访问 Azure Portal。

9. 在搜索栏中，输入 Images，并选择 Services 下的第一项。您将被重定向到 **镜像仪表盘**。

10. 点 **+Add**。您将被重定向到 Create a Image 仪表盘。
插入以下详情：

    a. **Name**:为您的新镜像选择一个名称。

    b. **资源组** ：选择一个**资源**组。

    c. **位置** ：选择与分配给您的存储帐户的区域匹配的**位置**。否则您将无法选择 blob。

    d. **操作系统类型** ：将操作系统类型设置为 **Linux**。

    e. **虚拟机生成** ：在 **Gen 1** 上保留虚拟机生成集。

    f. **Storage Blob**:点 **Storage blob input** 右面的 **Browse**。使用对话框查找您之前上传的镜像。
    将剩余的字段保留为默认值。

11. 点 **Create** 来创建镜像。创建镜像后，您可以在右上角看到消息 "Successfully created image"。

12. 点 **Refresh** 查看新镜像并打开新创建的镜像。

13. 点 **+ Create VM**。您将被重定向到 Create a virtual machine 仪表盘。

14. 在 Basic 选项卡中，Project Details, your *Subscription 和 Resource Group 已被预先设置。
如果要创建新资源组

    a. 点 **Create new**。
    弹出提示您创建 Resource Group Name 容器。

    b. 插入名称并单击 **OK**。
    如果要保留已经预先设置的 Resource Group。

15. 在 Instance Details 下插入：

    a. **Virtual machine name**

    b. **区域**

    c. **镜像** ：您创建的镜像默认是预先选择的镜像。

    d. **大小** ：选择最适合您的需求的虚拟机大小。
    将剩余的字段保留为默认值。

16. 在 Administrator account 下输入以下详情：

    a. **Username** ：帐户管理员的名称。

    b. **SSH public key source** ：从下拉菜单中选择 **Generate new key pair**。

您可以使用您已有的密钥对，也可以创建一个新的密钥对。另外，您可以使用 **镜像构建器** 来将用户添加到带有预设公钥的镜像中。如需了解更多详细信息，请参阅 创建带有 SSH 密钥的用户帐户。

c. **Key pair name** ：插入密钥对的名称。

17. **Inbound port rules** 下选择：

a. **公共入站端口** ：**允许所选端口**。

b. **选择入站端口** ：使用默认设置 **SSH(22)**。

18. 点击 **Review + Create**。您将被重定向到 **Review + create** 选项卡，并收到验证通过的确认信息。

19. 检查详情并点击 **Create**。
（可选）您可以点击 **Previous** 来修复之前选择的选项。

20. 此时弹出窗口 **generates new key pair** 会打开。点 **Download private key and create resources**。
将密钥文件保存为 "*yourKey*.pem"。

21. 部署完成后，点 **Go to resource**。

22. 您会被重定向到带有虚拟机详情的新窗口。选择页面右上方的公共 IP 地址并将其复制到您的剪贴板中。

现在，要创建与虚拟机的 SSH 连接，来连接到虚拟机。

1. 打开终端。

2. 在提示符后打开到您的虚拟机的 SSH 连接。将 IP 地址替换为您虚拟机的 IP 地址，并将 .pem 的路径替换为下载密钥文件的路径。

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

3. 需要确认是否要继续连接。键入 yes 来继续。

您推送到 Azure Storage Blob 的输出镜像现已准备好，可以用于置备。

其他资源

- Azure Storage 文档。
- 创建 Azure Storage 帐户。
- 在红帽客户门户网站中创建问题单。
- 帮助 + 支持。
- 联系红帽。

8.9. 将 QCOW2 镜像上传到 OPENSTACK

镜像构建器可以生成适合于上传到 OpenStack 云部署的镜像，并在那里启动实例。这描述了将 QCOW2 镜像上传到 OpenStack 的步骤。

先决条件

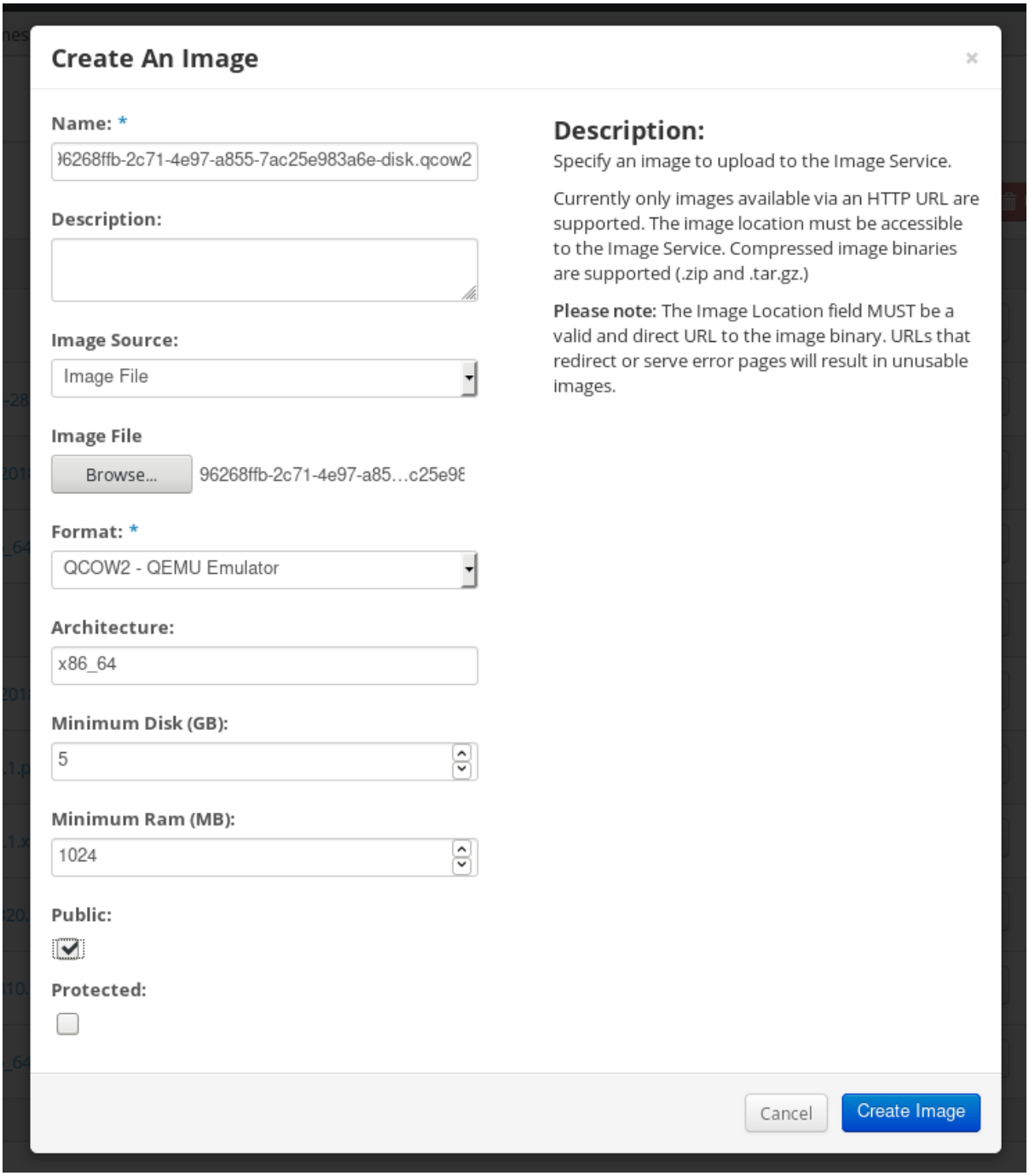
- 您必须具有由 Image Builder 创建的 OpenStack 特定镜像。在创建镜像时，请使用 CLI 中的 **openstack** 输出类型或 GUI 中的 **OpenStack Image(.qcow2)**。

警告

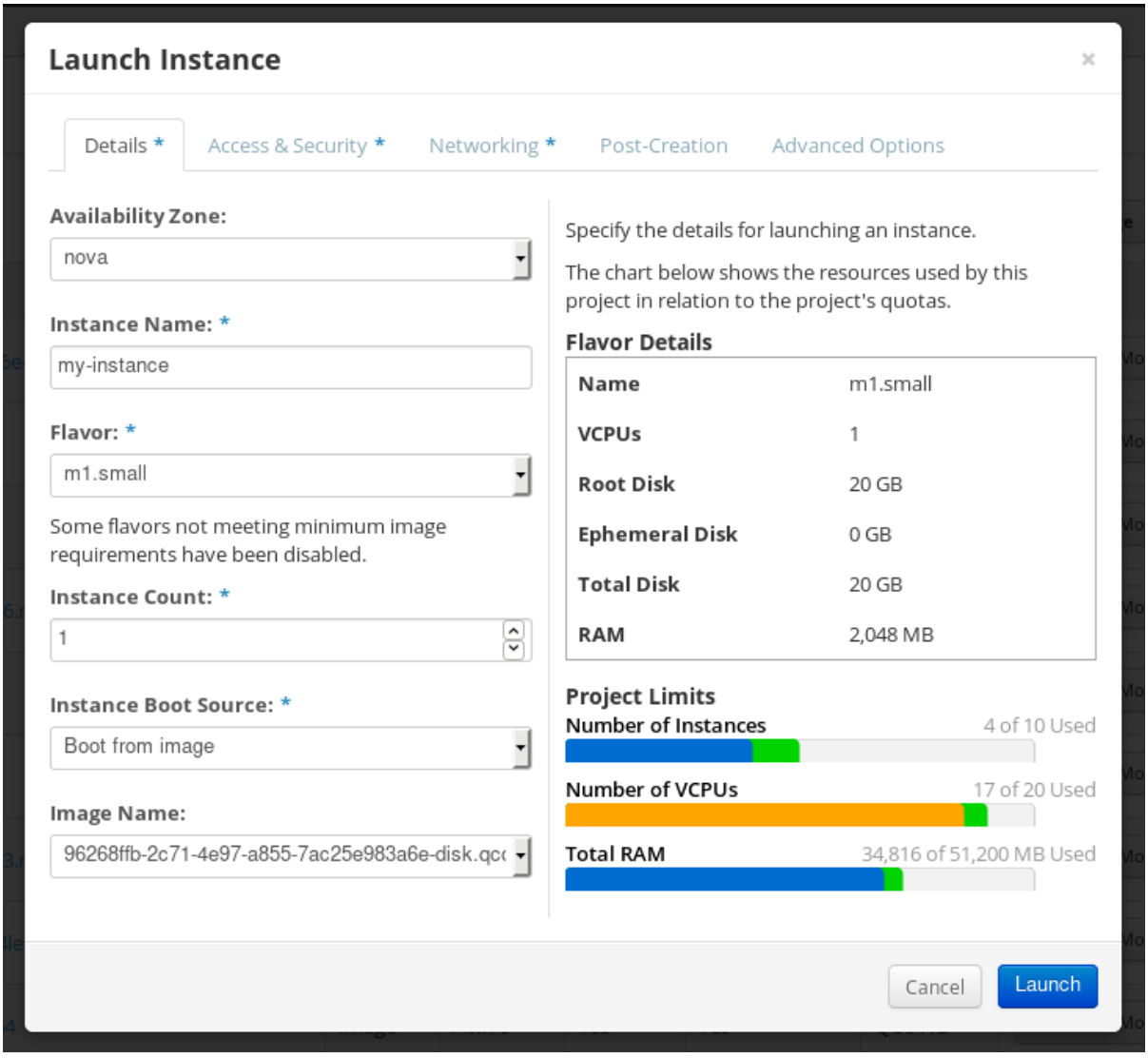
镜像构建器还提供通用 QCOW2 镜像类型输出格式，格式为 **qcow2** 或 **QEMU QCOW2 Image(.qcow2)**。不要将其与 QCOW2 格式的 OpenStack 镜像类型相混淆，后者包含针对 OpenStack 的进一步更改。

流程

1. 将映像上传到 OpenStack，并从其启动实例。使用 **Images** 界面进行此操作：


Create An Image
Name: *
96268ffb-2c71-4e97-a855-7ac25e983a6e-disk.qcow2
Description:
Image Source:
Image File
Image File
Browse...
96268ffb-2c71-4e97-a85...c25e98
Format: *
QCOW2 - QEMU Emulator
Architecture:
x86_64
Minimum Disk (GB):
5
Minimum Ram (MB):
1024
Public:
Protected:
Description:
Specify an image to upload to the Image Service.
Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)
Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.
Cancel
Create Image

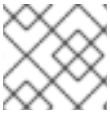
2. 使用该镜像启动实例：



3. 您可以使用快照的任何机制（CLI 或 OpenStack Web UI）来运行实例。通过 SSH 使用您的私钥访问生成的实例。以 **cloud-user** 用户身份登录。

8.10. 准备将镜像上传到 ALIBABA

本节论述了验证您可以在 Alibaba Cloud 上部署的自定义镜像的步骤。镜像需要一个特定的配置才能成功引导，因为 Alibaba Cloud 在使用前需要自定义镜像满足某些要求。因此，建议您使用 Alibaba 的 **image_check 工具**。



注意

自定义镜像验证是一个可选的任务。镜像构建器生成符合 Alibaba 要求的镜像。

先决条件

- 您必须具有由 Image Builder 创建的 Alibaba 镜像。

流程

1. 连接到包含您要通过 Alibaba **image_check 工具**检查的镜像的系统。
2. 下载 **image_check 工具**：

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

3. 更改镜像合规工具的文件权限：

```
# chmod +x image_check
```

4. 运行命令启动镜像合规工具检查：

```
# ./image_check
```

该工具会验证系统配置，并生成在屏幕上显示的报告。image_check 工具会在运行镜像合规工具的同一目录中保存此报告。

5. 如果任何检测项失败，请按照说明进行更正。如需更多信息，请参阅链接：检测项目部分。

其他资源

- 镜像合规工具

8.11. 将镜像上传到 ALIBABA

本节论述了如何将 Alibaba 镜像上传到对象存储服务（OSS）。

先决条件

- 设置您的系统以上传 Alibaba 镜像。
- 您必须具有由 Image Builder 创建的 Alibaba 镜像。在创建镜像时，请使用 RHEL 7 上的 **ami** 输出类型或 RHEL 8 上的 Alibaba 。
- 您有一个存储桶。请参阅创建存储桶。
- 您有一个活跃的 Alibaba 帐户。
- 已激活了 OSS。

流程

1. 登录到 OSS 控制台。
2. 在左侧 Bucket 菜单中，选择您要将镜像上传到的存储桶。
3. 在右上菜单中点击 **Files** 标签页。
4. 点 **Upload**。此时会在右侧打开窗口对话框。选择以下信息：
   - **上传到** ：选择将文件上传到 **Current** 目录或一个 **指定的**目录。
   - **文件 ACL** ：选择上传文件的权限类型。
5. 点 **Upload**。
6. 选择您要上传的镜像。

7. 点 **Open**。

自定义镜像被上传到 OSS 控制台。

其他资源

- 上传一个对象
- 从自定义镜像创建实例
- 导入镜像

8.12. 将镜像导入到 ALIBABA

本节描述了如何将 Alibaba 镜像导入到 Elastic Cloud Console(ECS)。

先决条件

- 您已将镜像上传到对象存储服务(OSS)。

流程

1. 登录到 ECS 控制台。
   i. 在左侧菜单中点击 **Images**。
   ii. 在右上方点击 **Import Image**。此时会打开一个窗口对话框。
   iii. 确认您已设置了镜像所在的正确区域。输入以下信息：
      a. **OSS 对象地址** ：了解如何获取 OSS 对象地址。
      b. **镜像名称**：
      c. **操作系统** ：
      d. **系统磁盘大小** ：
      e. **系统架构** ：
      f. **平台** ：Red Hat
   iv. 另外,还可提供以下详情：
      g. **镜像格式** ：qcow2 或 ami，具体取决于上传的镜像格式。
      h. **Image Description**：
      i. **Add Images of Data Disks**：
      在左侧菜单中选择了所需的存储桶后，可以在 OSS 管理控制台中确定地址，选择 Files 部分，然后单击相应镜像右侧的 **Details** 链接。此时会在屏幕右侧出现一个窗口，显示图像详情。OSS 对象地址位于 URL 框中。
2. 点 **确定**。

注意

导入过程的时间可能因镜像大小而异。

因此，自定义镜像被导入到 ECS 控制台。您可以从自定义镜像创建实例。

其他资源

- 导入镜像的备注
- 从自定义镜像创建实例
- 上传一个对象

8.13. 使用 ALIBABA 创建自定义镜像实例

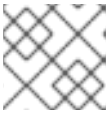
您可以使用 Alibaba ECS 控制台创建自定义镜像的实例。

先决条件

- 您已激活了 OSS 并上传您的自定义镜像。
- 您已成功将镜像导入到 ECS 控制台。

流程

1. 登录到 ECS 控制台。
2. 在左侧菜单中选择 **Instances**。
3. 在右上角，点 **Create Instance**。您会被重新定向到新窗口。
4. 填写所有需要的信息。如需了解更多详细信息，请参阅使用向导创建实例。
5. 点 **Create Instance** 并确认顺序。



注意

根据订阅，您可以看到 **Create Order** 选项，而不是 **Create Instance** 选项。

因此，您有一个活跃的实例准备用于部署。

其他资源

- 使用自定义镜像创建实例
- 使用向导创建一个实例