Red Hat Enterprise Linux 8

创建自定义 RHEL 系统镜像

在 Red Hat Enterprise Linux 8 上使用 RHEL 镜像构建器创建自定义系统镜像

Last Updated: 2024-06-29

Red Hat Enterprise Linux 8 创建自定义 RHEL 系统镜像

在 Red Hat Enterprise Linux 8 上使用 RHEL 镜像构建器创建自定义系统镜像

法律通告

摘要

RHEL 镜像构建器是一个创建部署就绪的自定义系统镜像的工具：安装磁盘、虚拟机、特定于云提供商的镜像等。通过使用 RHEL 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，因为它消除了每种输出类型所需的特定配置。

目录

对红帽文档提供反馈

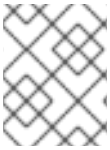
我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 RHEL 镜像构建器描述

要部署一个系统，请创建一个系统镜像。要创建 RHEL 系统镜像，请使用 RHEL 镜像构建器工具。您可以使用 RHEL 镜像构建器创建 RHEL 的自定义系统镜像，包括为在云平台上的部署准备的系统镜像。RHEL 镜像构建器自动为每种输出类型处理配置详情，因此比手动创建镜像方法更容易和更快地使用。您可以使用 **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访问 RHEL 镜像构建器功能。



注意

从 RHEL 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。

1.1. RHEL 镜像构建器术语

RHEL 镜像构建器使用以下概念：

Blueprint

Blueprint 是自定义系统镜像的描述。它列出了将成为系统一部分的软件包和自定义。您可以使用自定义编辑蓝图，并将其保存为特定版本。当从蓝图创建系统镜像时，镜像与 RHEL 镜像构建器界面中的蓝图相关联。
以 TOML 格式创建蓝图。

Compose

Compose 是基于特定蓝图的特定版本的系统镜像的单独构建。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。

Customizations

Customizations 是不是软件包的镜像的规范。这包括用户、组和 SSH 密钥。

1.2. RHEL 镜像构建器输出格式

RHEL 镜像构建器可以以下表中显示的多种输出格式创建镜像。

表 1.1. RHEL 镜像构建器输出格式

描述	CLI 名称	文件扩展名
QEMU 镜像	**qcow2**	**.qcow2**
磁盘归档	**tar**	**.tar**
Amazon Web Services	**raw**	**.raw**
Microsoft Azure	**vhd**	**.vhd**
Google Cloud Platform	**gce**	**.tar.gz**
VMware vSphere	**vmdk**	**.vmdk**

描述	CLI 名称	文件扩展名
VMware vSphere	**ova**	**.ova**
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**
RHEL for Edge Raw 镜像	**edge-raw-image**	**.raw.xz**
用于边缘简化安装程序的 RHEL	**edge-simplified-installer**	**.iso**
RHEL for Edge AMI	**edge-ami**	**.ami**
RHEL for Edge VMDK	**edge-vsphere**	**.vmdk**
RHEL 安装程序	**image-installer**	**.iso**
Oracle 云基础架构	**.oci**	**.qcow2**

要检查支持的类型，请运行以下命令：

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

第 2 章 安装 RHEL 镜像构建器

在使用 RHEL 镜像构建器前，您必须安装它。

2.1. RHEL 镜像构建器系统要求

运行 RHEL 镜像构建器的主机必须满足以下要求：

表 2.1. RHEL 镜像构建器系统要求

参数	最低要求值
系统类型	一个专用的主机或虚拟机。请注意，RHEL 镜像构建器在容器中不支持，包括 Red Hat Universal Base Images (UBI)。
处理器	2 个内核
内存	4 GiB
磁盘空间	'/var/cache/' 文件系统中有 20 GiB 可用空间
访问权限	root
Network	到 Red Hat Content Delivery Network (CDN)的互联网连接。

注意

如果您没有互联网连接，请在隔离网络中使用 RHEL 镜像构建器。为此，您必须覆盖默认存储库以指向本地存储库，来不连接到 Red Hat Content Delivery Network (CDN)。确保您的内容在内部镜像了或使用 Red Hat Satellite。

其他资源

- 配置 RHEL 镜像构建器存储库
- 使用红帽镜像构建器镜像置备 Satellite

2.2. 安装 RHEL 镜像构建器

安装 RHEL 镜像构建器以访问所有 **osbuild-composer** 软件包功能。

先决条件

- 您已登陆到要在其上安装 RHEL 镜像构建器的 RHEL 主机。
- 主机已订阅到 Red Hat Subscription Manager (RHSM)或 Red Hat Satellite。
- 您已启用了 **BaseOS** 和 **AppStream** 存储库，以便能安装 RHEL 镜像构建器软件包。

流程

1. 安装 RHEL 镜像构建器和其他必要的软件包：

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

   - **osbuild-composer** - 一个构建自定义 RHEL 操作系统镜像的服务。
   - **composer-cli**- 这个软件包提供对 CLI 界面的访问。
   - **cockpit-composer** - 这个软件包提供对 Web UI 界面的访问。Web 控制台作为 **cockpit-composer** 软件包的依赖项安装。

2. 启用并启动 RHEL 镜像构建器套接字：

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

3. 如果要在 web 控制台中使用 RHEL 镜像构建器，请启用并启动它。

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

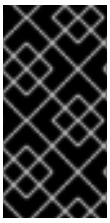
   osbuild-composer 和 **cockpit** 服务在第一次访问时自动启动。

4. 加载 shell 配置脚本，以便 **composer-cli** 命令的自动完成功能立即开始工作，而无需退出，然后再次登录：

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

5. 在 RHEL 主机上重启运行的 **osbuild-composer** 服务。

```
# systemctl restart osbuild-composer
```



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

- 通过运行 **composer-cli** 来验证安装是否正常工作：

```
# composer-cli status show
```

故障排除

您可以使用系统日志来跟踪 RHEL 镜像构建器活动。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
```

2.3. 回到 LORAX-COMPOSER RHEL 镜像构建器后端

osbuild-composer 后端虽然具有更大的可扩展性，但目前还无法与之前的 **lorax-composer** 后端功能相媲美。

要回到以前的后端，请按照以下步骤操作：

先决条件

- **osbuild-composer** 软件包已安装

流程

1. 删除 osbuild-composer 后端。

```
# yum remove osbuild-composer
# yum remove weldr-client
```

2. 在 **/etc/yum.conf** 文件中，为 **osbuild-composer** 软件包添加一个排除项。

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

3. 安装 **lorax-composer** 软件包。

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

4. 启用并启动 **lorax-composer** 服务，以便在每次重启后启动。

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

其他资源

- 创建红帽支持问题单。

第 3 章 配置 RHEL 镜像构建器存储库

要使用 RHEL 镜像构建器，您必须确保配置了存储库。您可以在 RHEL 镜像构建器中使用以下类型的存储库：

官方存储库覆盖

如果您要从 Red Hat Content Delivery Network (CDN)官方存储库（如网络中的自定义镜像）以外的其他位置下载基础系统 RPM。使用官方存储库覆盖会禁用默认存储库，您的自定义镜像必须包含所有必需的软件包。

自定义第三方存储库

使用这些存储库来包括官方 RHEL 软件仓库中没有的软件包。

3.1. 在 RHEL 镜像构建器中添加自定义第三方存储库

您可以将自定义第三方源添加到存储库中，并使用 **composer-cli** 管理这些存储库。

先决条件

- 您有自定义第三方存储库的 URL。

流程

1. 创建一个存储库源文件，如 **/root/repo.toml**。例如：

```
id = "k8s"
name = "Kubernetes"
type = "yum-baseurl"
url = "https://server.example.com/repos/company_internal_packages/"
check_gpg = false
check_ssl = false
system = false
```

type 字段接受以下有效值 **yum-baseurl**、**yum-mirrorlist** 和 **yum-metalink**。

2. 以 TOML 格式保存文件。
3. 将新的第三方源添加到 RHEL 镜像构建器中：

```
$ composer-cli sources add <file-name>.toml
```

验证

1. 检查新源是否已成功添加：

```
$ composer-cli sources list
```

2. 检查新源内容：

```
$ composer-cli sources info <source_id>
```

3.2. 将带有特定发行版的第三方存储库添加到 RHEL 镜像构建器中

您可以使用可选字段 **distro** 在自定义第三方源文件中指定发行版列表。在镜像构建期间解析依赖项时，存储库文件使用分发字符串列表。

任何指定 **rhel-8** 的请求都使用这个源。例如，如果您列出软件包并指定 **rhel-8**，它将包含此源。但是，列出主机发行版的软件包不包括此源。

先决条件

- 您有自定义第三方存储库的 URL。
- 您有要指定的发行版的列表。

流程

1. 创建一个存储库源文件，如 **/root/repo.toml**。例如，要指定发行版：

```
check_gpg = true
check_ssl = true
distros = ["rhel-8"]
id = "rh9-local"
name = "packages for RHEL"
system = false
type = "yum-baseurl"
url = "https://local/repos/rhel8/projectrepo/"
```

2. 以 TOML 格式保存文件。
3. 将新的第三方源添加到 RHEL 镜像构建器中：

```
$ composer-cli sources add <file-name>.toml
```

验证

1. 检查新源是否已成功添加：

```
$ composer-cli sources list
```

2. 检查新源内容：

```
$ composer-cli sources info <source_id>
```

3.3. 使用 GPG 检查存储库元数据

要检测和避免损坏的软件包，您可以使用 DNF 软件包管理器检查 RPM 软件包上的 GNU Privacy Guard (GPG)签名，并检查存储库元数据是否已使用 GPG 密钥进行了签名。

您可以使用密钥 URL 设置 **gpgkeys** 字段，输入您要通过 **https** 进行检查的 **gpgkey**。或者，为了提高安全性，您还可以将整个密钥嵌入到 **gpgkeys** 字段中，来直接导入它，而不是直接从 URL 获取密钥。

先决条件

- 您要用作存储库的目录存在，且包含软件包。

流程

1. 访问您要创建存储库的文件夹：

```
$ cd repo/
```

2. 运行 **createrepo_c** 来从 RPM 软件包创建存储库：

```
$ createrepo_c .
```

3. 访问 repodata 所在的目录：

```
$ cd repodata/
```

4. 为 **repomd.xml** 文件签名：

```
$ gpg -u <_gpg-key-email_> --yes --detach-sign --armor /srv/repo/example/repomd.xml
```

5. 要在存储库中启用 GPG 签名检查：

   a. 在存储库源中设置 **check_repogpg = true**。

   b. 输入您要进行检查的 **gpgkey**。如果您的密钥可以通过 **https** 使用，请使用密钥的密钥 URL 设置 **gpgkeys** 字段。您可以根据需要添加任意数量的 URL 密钥。
   以下是一个示例：

```
check_gpg = true
check_ssl = true
id = "signed local packages"
name = "repository_name"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
check_repogpg = true
gpgkeys=["https://local/keys/repokey.pub"]
```

   作为替代方案，直接在 **gpgkeys** 字段中添加 GPG 密钥，例如：

```
check_gpg = true
check_ssl = true
check_repogpg
id = "custom-local"
name = "signed local packages"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
gpgkeys=["https://remote/keys/other-repokey.pub",
'"-----BEGIN PGP PUBLIC KEY BLOCK-----
…
-----END PGP PUBLIC KEY BLOCK-----"']
```

   - 如果测试没有找到签名，GPG 工具会显示一个类似如下的错误：

```
$ GPG verification is enabled, but GPG signature is not available.
This may be an error or the repository does not support GPG verification:
Status code: 404 for http://repo-server/rhel/repodata/repomd.xml.asc (IP:
```

```
192.168.1.3)
```

- 如果签名无效，GPG 工具会显示一个类似如下的错误：

```
repomd.xml GPG signature verification error: Bad GPG signature
```

验证

- 手动测试存储库的签名：

```
$ gpg --verify /srv/repo/example/repomd.xml.asc
```

3.4. RHEL 镜像构建器官方存储库覆盖

RHEL 镜像构建器 **osbuild-composer** 后端不继承位于 **/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。这包括红帽官方存储库，其中包含要安装附加软件或将已安装的程序基础更新到新版本的基础系统 RPM。如果要覆盖官方存储库，您必须在 **/etc/osbuild-composer/repositories/** 中定义覆盖。此目录用于用户定义的覆盖，这里的文件优先于 **/usr/share/osbuild-composer/repositories/** 目录中的文件。

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

3.5. 覆盖系统存储库

您可以在 **/etc/osbuild-composer/repositories** 目录中为 RHEL 镜像构建器配置自己的存储库覆盖。

注意

在 RHEL 8.5 发行版本中，仓库覆盖的名称为 **rhel-8.json**。从 RHEL 8.5 开始，名称也会尊重次要版本： **rhel-84.json**、**rhel-85.json** 等等。

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 **/etc/osbuild-composer/repositories/** 目录来存储存储库覆盖：

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

2. 使用与 RHEL 版本对应的名称，创建一个 JSON 文件。或者，您还可以从 **/usr/share/osbuild-composer/** 中复制用于分发的文件，并修改其内容。
   对于 RHEL 8.9，请使用 **/etc/osbuild-composer/repositories/rhel-89.json**。

3. 将以下结构添加到 JSON 文件中。仅以字符串格式指定以下属性之一：

   - **baseurl** - 存储库的基本 URL。
   - **metalink** - 包含有效镜像存储库列表的 metalink 文件的 URL。
   - **mirrorlist** - 包含有效镜像存储库列表的 mirrorlist 文件的 URL。其余的字段（如 **gpgkey**）和 **metadata_expire** 是可选的。

例如：

```
{
    "x86_64": [
      {
          "name": "baseos",
          "baseurl": "http://mirror.example.com/composes/released/RHEL-
8/8.0/BaseOS/x86_64/os/",
          "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (…)",
          "check_gpg": true
      }
   ]
}
```

或者，您可以通过将 **rhel-version.json** 替换为 RHEL 版本（例如：rhel-8.json）来复制用于分发的 JSON 文件。

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

4. 可选：验证 JSON 文件：

```
$ json_verify  /etc/osbuild-composer/repositories/<file>.json
```

5. 编辑 **rhel-8.json** 文件中的 **baseurl** 路径，并保存。例如：

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

6. 重启 **osbuild-composer.service** ：

```
$ sudo systemctl restart osbuild-composer.service
```

验证

- 检查存储库是否指向正确的 URL：

```
$ cat /etc/yum.repos.d/redhat.repo
```

您可以看到仓库指向从 **/etc/yum.repos.d/redhat.repo** 文件复制的正确 URL。

其他资源

- 存储库中可用的最新的 RPM 版本对 **osbuild-composer** 不可见。

3.6. 覆盖需要订阅的系统存储库

您可以设置 **osbuild-composer** 服务，以使用 **/etc/yum.repos.d/redhat.repo** 文件中定义的系统订阅。要使用 **osbuild-composer** 中的系统订阅，请定义一个具有以下详情的存储库覆盖：

- 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。

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** 将其指定为附加 **源**。"BaseOS"和"AppStream"存储库通常使用系统存储库覆盖，而所有其他存储库则使用 **composer-cli** 源。

先决条件

- 您的系统有一个在 **/etc/yum.repos.d/redhat.repo** 中定义的订阅
- 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** ：

```
# cat /etc/yum.repos.d/redhat.repo
[AppStream]
name = AppStream mirror example
baseurl = https://mirror.example.com/RHEL-8/8.0/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-8/8.0/AppStream/x86_64/os/",
         "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (…)",
         "check_gpg": true,
         "rhsm": true
      }
   ]
}
```

3. 重启 **osbuild-composer.service** ：

```
$ sudo systemctl restart osbuild-composer.service
```

其他资源

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

3.7. 配置和使用 SATELLITE CV 作为内容源

您可以使用 Satellite 的内容视图(CV)作为存储库来使用 RHEL 镜像构建器构建镜像。为此，请在注册到 Satellite 的主机上手动配置存储库引用，以便您可以从 Satellite 存储库检索，而不是 Red Hat Content Delivery Network (CDN)官方存储库。

先决条件

- 您在注册到 Satellite 6 的主机上使用 RHEL 镜像构建程序。

流程

1. 从您当前配置的软件仓库中找到存储库 URL：

```
$ sudo yum -v repolist rhel-8-for-x86_64-baseos-rpms | grep repo-baseurl
  Repo-baseurl :
```

以下输出是一个示例：

```
https://satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV/content/dist/rhel8/8/x86_64/baseos/os
```

2. 将硬编码的存储库修改为 Satellite 服务器。

   a. 创建具有 **0755** 权限的存储库目录：

```
$ sudo mkdir -pvm 0755 /etc/osbuild-composer/repositories
```

   b. 将 **/usr/share/osbuild-composer/repositories suit.json** 中的内容复制到您创建的目录中：

```
$ sudo cp /usr/share/osbuild-composer/repositories/*.json /etc/osbuild-composer/repositories/
```

   c. 通过 **/content/dist** configuration 行更新 Satellite URL 和文件内容：

```
$ sudo sed -i -e 's|cdn.redhat.com|satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV|' /etc/osbuild-composer/repositories/*.json
```

   d. 验证配置是否已正确替换：

```
$ sudo vi /etc/osbuild-composer/repositories/rhel-8.json
```

3. 重启服务：

```
$ sudo systemctl restart osbuild-worker@1.service osbuild-composer.service
```

4. 覆盖红帽镜像构建器配置中所需的系统存储库，并将 Satellite 存储库的 URL 用作 baseurl。请参阅 覆盖系统存储库。

其他资源

- 当在 Satellite 上定义了多个自定义软件仓库时，RHEL 镜像构建器会失败

3.8. 使用 SATELLITE CV 作为软件仓库在 RHEL 镜像构建器中构建镜像

配置 RHEL 镜像构建器，以使用 Satellite 的内容视图(CV)作为存储库来构建您的自定义镜像。

先决条件

- 您已将 Satellite 与 RHEL web 控制台集成。请参阅 在 Satellite 中启用 RHEL web 控制台

流程

1. 在 Satellite Web UI 中，导航到 **Content > Products**，选择您的 **Product** 并点您要使用的存储库。
2. 在 Published 字段中搜索安全 URL (HTTPS)并复制它。
3. 使用您复制的 URL 作为红帽镜像构建器存储库的 baseurl。请参阅 在 RHEL 镜像构建器中添加自定义第三方存储库。

后续步骤

- 构建镜像。请参阅 在 web 控制台界面中使用 RHEL 镜像构建器 创建系统镜像 。

第 4 章 使用 RHEL 镜像构建器 CLI 创建系统镜像

RHEL 镜像构建器是一个创建自定义系统镜像的工具。要控制 RHEL 镜像构建器并创建自定义系统镜像，您可以使用命令行界面(CLI)或 Web 控制台界面。

4.1. RHEL 镜像构建器命令行界面简介

您可以使用 RHEL 镜像构建器命令行界面(CLI)创建蓝图，方法是运行带有合适的选项和子命令的 **composer-cli** 命令。

命令行界面的工作流总结如下：

1. 创建蓝图或将现有蓝图定义导出(*保存*)到纯文本文件中
2. 在文本编辑器中编辑这个文件
3. 将蓝图文本文件重新导回到镜像构建器中
4. 运行 compose 来从蓝图构建镜像
5. 导出镜像文件以下载它

除了创建蓝图的基本子命令外，**composer-cli** 命令还提供了许多子命令来检查配置的蓝图和组成的状态。

4.2. 以非 ROOT 用户身份使用 RHEL 镜像构建器

要以非 root 身份运行 **composer-cli** 命令，该用户必须在 **weldr** 组中。

先决条件

- 您已创建了一个用户

流程

- 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
```

4.3. 使用命令行界面创建蓝图

您可以使用命令行界面(CLI)创建一个新的 RHEL 镜像构建器蓝图。蓝图描述了最终的镜像及其自定义，如软件包和内核自定义。

前提条件

- 您以 root 用户身份或以 welder 组成员的用户身份登录，

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**。

可选，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 8.7.0。
- 对于最新可用版本，请使用星号 *。
- 对于最新的次版本，请使用以下格式，如 8.*。

3. 自定义蓝图以满足您的需要。例如，禁用 Simultaneous Multi Threading (SMT)，在蓝图文件中添加以下行：

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

有关其他可用的定制信息，请参阅 支持的镜像自定义。

4. 将文件保存为 例如 *BLUEPRINT-NAME*.toml，并关闭文本编辑器。

5. 推送蓝图：

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

将 *BLUEPRINT-NAME* 替换为您在前面步骤中使用的值。

注意

以非 root 身份运行 **composer-cli** 命令创建镜像，请将您的用户添加到 **weldr** 或 **root** 组中。

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

验证

- 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
```

- 显示您刚刚添加的蓝图配置：

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

- 检查蓝图中列出的组件和版本是否有效：

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

如果 RHEL 镜像构建器无法从自定义存储库中解决软件包的依赖项，请删除 **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

4.4. 使用命令行界面编辑蓝图

您可以在命令行(CLI)界面中编辑现有蓝图，例如，来添加新软件包或定义新组，并创建自定义镜像。为此，请按照以下步骤操作：

先决条件

- 您已创建了一个蓝图

流程

1. 列出现有的蓝图：

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

2. 将蓝图保存到一个本地文本文件中：

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

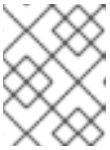
3. 使用文本编辑器编辑 *BLUEPRINT-NAME*.toml 文件并进行更改。
4. 在完成编辑前，请验证该文件是否是一个有效的蓝图：
   a. 如果存在，从蓝图中删除以下行：

```
packages = []
```

   b. 增加版本号，例如，从 0.0.1 增加到 0.1.0。请记住，RHEL 镜像构建器蓝图版本必须使用 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。

5. 保存文件并关闭文本编辑器。
6. 将蓝图推送回 RHEL 镜像构建器：

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



注意

要将蓝图导回到 RHEL 镜像构建器，请提供包括 **.toml** 扩展名的文件名，而在其他命令中则只使用蓝图名称。

验证

1. 要验证上传到 RHEL 镜像构建器的内容是否与您编辑的内容匹配，请列出蓝图的内容：

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

2. 检查蓝图中列出的组件和版本是否有效：

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

其他资源

- 支持的镜像自定义

4.5. 在命令行界面中使用 RHEL 镜像构建器创建一个系统镜像

您可以使用 RHEL 镜像构建器命令行界面构建一个自定义 RHEL 镜像。为此，您必须指定蓝图和镜像类型。可选，您还可以指定一个发行版。如果没有指定发行版，它将使用与主机系统一样的发行版和版本。架构也与主机上的架构一样。

先决条件

- 您已为镜像准备了蓝图。请参阅 使用命令行界面创建一个 RHEL 镜像构建器蓝图。

流程

1. 可选：列出您可以创建的镜像格式：

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

2. 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。有关可用值，请查看 **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)。

3. 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 **FINISHED** 状态值。要识别列表中您的 compose，请使用其 UUID。

4. 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：

- 下载镜像的元数据以获取 compose 的元数据的 **.tar** 文件：

```
$ 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
```

- 检查镜像的清单：

```
$ 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 RHEL 镜像构建器

4.6. 基本 RHEL 镜像构建器命令行命令

RHEL 镜像构建器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（导入）到 RHEL 镜像构建器

```
# 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>
```

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>
```

更多子命令和选项

```
# composer-cli help
```

其他资源

- *composer-cli*(1) man page

4.7. RHEL 镜像构建器蓝图格式

RHEL 镜像构建器蓝图以 TOML 格式的纯文本向用户显示。

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>"
```

BLUEPRINT-NAME 和 *LONG FORM DESCRIPTION TEXT* 字段是您的蓝图的名称和描述。

VERSION 是根据 Semantic Versioning 方案的版本号，只对整个蓝图文件显示一次。

镜像中包含的组

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

group 条目描述要安装到镜像中的一组软件包。组使用以下软件包类别：

- Mandatory（必需）
- Default（默认）
- 选填

  group-name 是组的名称，例如 **anaconda-tools**, **widget**, **wheel** 或 **users**。蓝图安装必需的和默认的软件包。没有选择可选软件包的机制。

镜像中包含的软件包

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

package-name 是软件包的名称，如 **httpd**、**gdb-doc** 或 **coreutils**。

package-version 是要使用的版本。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 8.7.0。
- 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.*。

为每个要包括的软件包重复这个块。

注意

RHEL 镜像构建器工具中的软件包和模块之间没有区别。两者都被视为 RPM 软件包依赖项。

4.8. 支持的镜像自定义

您可以通过向蓝图中添加自定义来自定义镜像，例如：

- 添加一个额外的 RPM 软件包
- 启用一个服务
- 自定义一个内核命令行参数.

在其他参数之间。您可以在蓝图中使用多个镜像自定义。通过使用自定义，您可以将软件包和组添加到默认软件包中不可用的镜像中。要使用这些选项，请在蓝图中配置自定义，并将其导入(推送)到 RHEL 镜像构建器中。

其他资源

- 在添加文件系统自定义 "size" 后，蓝图导入会失败。

4.8.1. 选择一个发行版

您可以使用 **distro** 字段选择在制作镜像时使用的发布，或者解决蓝图中的依赖项。如果 **distro** 留空，它将使用主机分布。如果没有指定发行版，则蓝图将使用主机分发。如果您升级主机操作系统，则没有发行版集合的蓝图使用新的操作系统版本构建镜像。您无法构建一个与 RHEL 镜像构建器主机不同的操作系统镜像。

流程

- 使用 RHEL 发行版本自定义蓝图，以始终构建指定的 RHEL 镜像：

```
name = "blueprint_name"
description = "blueprint_version"
version = "0.1"
distro = "different_minor_version"
```

将 "*different_minor_version*" 替换为构建不同的次版本，例如，如果要构建 RHEL 8.10 镜像，请使用 **distro** = "rhel-810"。在 RHEL 8.10 镜像中，您可以构建次版本，如 RHEL 8.9 及更早的版本。

4.8.2. 选择一个软件包组

自定义带有软件包和模块的蓝图。**name** 属性是必需的字符串。**version** 属性是一个可选字符串，如果未提供，则使用存储库中的最新版本。

注意

目前，**osbuild-composer** 中的软件包和模块之间没有区别。两者都被视为 RPM 软件包依赖项。

流程

- 自定义带有软件包的蓝图：

```
[[packages]]
name = "package_group_name"
```

将 "*package_group_name*" 替换为组的名称。例如，"tmux"。

```
[[packages]]
name = "tmux"
version = "2.9a"
```

4.8.3. 设置镜像主机名

customizations.hostname 是一个可选字符串，您可以用来配置最终镜像主机名。此自定义是可选的，如果您未设置它，则蓝图使用默认主机名。

流程

- 自定义蓝图以配置主机名：

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

4.8.4. 指定其他用户

将用户添加到镜像中，并可选择设置其 SSH 密钥。本节的所有字段都是可选的，但 **name** 除外。

流程

- 自定义蓝图来将用户添加到镜像中：

```
[[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
```

GID 是可选的，且必须在镜像中已存在。（可选）软件包会创建它，或者蓝图使用 **[[customizations.group]]** 条目创建 GID。

将 *PASSWORD-HASH* 替换为实际的 **密码哈希**。要生成 **密码哈希**，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())'
```

使用适当的值替换其他占位符。

输入 **name** 值，并省略您不需要的任何行。

为每个要包含的用户重复这个块。

4.8.5. 指定附加组

为生成的系统镜像指定一个组。**name** 和 **gid** 属性都是必需的。

流程

- 自定义带有组的蓝图：

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

为每个组重复此块。

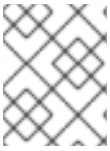
4.8.6. 为现有用户设置 SSH 密钥

您可以使用 **customizations.sshkey** 为最终镜像中的现有用户设置 SSH 密钥。 **user** 和 **key** 属性是必需的。

流程

- 通过为现有用户设置 SSH 密钥来自定义蓝图：

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



注意

您只能为现有用户配置 **customizations.sshkey** 自定义。要创建用户并设置 SSH 密钥，请参阅 **生成的系统镜像的用户规格** 自定义。

4.8.7. 附加一个内核参数

您可以向引导装载程序内核命令行中附加参数。默认情况下，RHEL 镜像构建器将默认内核构建到镜像中。但是，您可以通过在蓝图中配置它来自定义内核。

流程

- 在默认值中附加内核引导选项：

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

- 定义一个要在镜像中使用的内核名称

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

4.8.8. 设置时区和 NTP

您可以自定义蓝图来配置时区和 *网络时间协议* (NTP)。**timezone** 和 **ntpservers** 属性是可选字符串。如果您没有自定义时区，系统将使用 *Universal Time, Coordinated* (UTC)。如果您没有设置 NTP 服务器，系统将使用默认发行版。

流程

- 自定义带有您想要的 **timezone** 和 **ntpservers** 的蓝图：

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

例如：

```
[customizations.timezone]
timezone = "US/Eastern"
ntpservers = ["0.north-america.pool.ntp.org", "1.north-america.pool.ntp.org"]
```

注意

有些镜像类型，如 Google Cloud，已经建立了 NTP 服务器。您无法覆盖它，因为镜像需要 NTP 服务器来在所选环境中引导。但是，您可以在蓝图中自定义时区。

4.8.9. 自定义区域设置

您可以为生成的系统镜像自定义区域设置。**language** 和 **keyboard** 属性是必需的。您可以添加许多其他语言。您添加的第一个语言是主语言，其他语言是次要语言。

流程

- 设置区域设置：

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

例如：

```
[customizations.locale]
languages = ["en_US.UTF-8"]
keyboard = "us"
```

- 要列出语言支持的值，请运行以下命令：

```
$ localectl list-locales
```

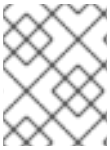
- 要列出键盘支持的值，请运行以下命令：

```
$ localectl list-keymaps
```

4.8.10. 自定义防火墙

为生成的系统镜像设置防火墙。默认情况下，防火墙阻止进入的连接，但明确启用其端口的服务除外，如 **sshd**。

如果您不想使用 **[customizations.firewall]** 或 **[customizations.firewall.services]**，可以删除属性，或者将它们设置为空列表 []。如果您只想使用默认的防火墙设置，您可以从蓝图中省略自定义。



注意

Google 和 OpenStack 模板为其环境明确禁用防火墙。您无法通过设置蓝图来覆盖此行为。

流程

- 使用以下设置自定义蓝图，以打开其他端口和服务：

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

其中 **ports** 是包含要打开的端口或一系列端口和协议的可选字符串列表。您可以使用以下格式配置端口：**port:protocol** 格式。您可以使用 **portA-portB:protocol** 格式配置端口范围。例如：

```
[customizations.firewall]
ports = ["22:tcp", "80:tcp", "imap:tcp", "53:tcp", "53:udp", "30000-32767:tcp", "30000-32767:udp"]
```

您可以使用数字端口或 **/etc/services** 中的名称来启用或禁用端口列表。

- 在 **customizations.firewall.service** 部分中指定要启用或禁用哪个防火墙服务：

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

- 您可以检查可用的防火墙服务：

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

例如：

```
[customizations.firewall.services]
enabled = ["ftp", "ntp", "dhcp"]
disabled = ["telnet"]
```

注意

firewall.services 中列出的服务与 **/etc/services** 文件中提供的 **service-names** 不同。

4.8.11. 启用或禁用服务

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服务，以确保镜像正常工作，您无法覆盖此设置。蓝图中的 **[customizations.services]** 设置不会替代这些服务，但可以将服务添加到已在镜像模板中存在的服务列表中。

流程

- 自定义在引导时要启用哪些服务：

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

例如：

```
[customizations.services]
enabled = ["sshd", "cockpit.socket", "httpd"]
disabled = ["postfix", "telnetd"]
```

4.8.12. 指定分区模式

使用 **partitioning_mode** 变量选择如何对您要构建的磁盘镜像进行分区。您可以使用以下支持的模式自定义镜像：

- **auto-lvm** ：它使用原始分区模式，除非有一个或多个文件系统自定义。在这种情况下，它使用 LVM 分区模式。
- **LVM** ：它总是使用 LVM 分区模式，即使没有额外的挂载点。
- Raw ：即使有一个或多个挂载点，它使用原始分区。
- **您可以使用以下自定义使用 partitioning_mode 变量自定义蓝图：**

```
[customizations]
partitioning_mode = "lvm"
```

4.8.13. 指定一个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，因此可以创建具有特定磁盘布局，而不是默认布局配置的镜像。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**

- 提高的性能
- 与现有设置的一致性

注意

OSTree 系统不支持文件系统自定义，因为 OSTree 镜像有自己的挂载规则，如只读。不支持以下镜像类型：

- image-installer
- edge-installer
- edge-simplified-installer

另外，以下镜像类型不支持文件系统自定义，因为这些镜像类型不会创建分区的操作系统镜像：

- edge-commit
- edge-container
- tar
- container

对于 RHEL 8.10 和 9.4 之前的发行版本，蓝图支持以下 挂载点 及其子目录：

- / - root 挂载点

- /var
- /home
- /opt
- /srv
- /usr
- /app
- /data
- /tmp

在 RHEL 9.4 和 8.10 发行版本中，您可以指定任意自定义挂载点，除了为操作系统保留的特定路径除外。

您不能在以下挂载点及其子目录中指定任意自定义挂载点：

- /bin
- /boot/efi
- /dev
- /etc

- /lib
- /lib64
- /lost+found
- /proc
- /run
- /sbin
- /sys
- /sysroot
- /var/lock
- /var/run

您可以在蓝图中为 /usr 自定义挂载点自定义文件系统，但不允许其子目录。

注意

从 RHEL 8.5 发行版开始，才支持使用 CLI 自定义挂载点。在之前的发行版本中，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。从 RHEL 8.6 开始，对于 osbuild-composer-46.1-1.el8 RPM 及更新版本，物理分区不再可用，文件系统自定义会创建逻辑卷。

如果您在自定义镜像中有多个分区，您可以在 LVM 上创建带有自定义文件系统分区的镜像，并在运行时调整这些分区大小。为此，您可以在蓝图中指定自定义的文件系统配置，因此可以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，cloud-init 会调整 root 分区的大小。

蓝图自动将文件系统自定义转换为 LVM 分区。

您可以使用自定义文件蓝图自定义来创建新文件或替换现有文件。您指定的文件的父目录必须存在，否则镜像构建会失败。通过在 [[customizations.directories]] 自定义中指定它来确保父目录存在。

警告

如果您将文件自定义与其他蓝图自定义相结合，这可能会影响其他自定义的功能，或者可能会覆盖当前的文件自定义。

4.8.13.1. 在蓝图中指定自定义文件

使用 [[customizations.files]] 蓝图自定义，您可以：

- 创建新文本文件。
- 修改现有文件。警告：这可能会覆盖现有内容。
- 为您要创建的文件设置用户和组所有权。
- 以八进制格式设置 mode 权限。

您无法创建或替换以下文件：

- /etc/fstab

- /etc/shadow
- /etc/passwd
- /etc/group

您可以使用 [[customizations.files]] 和 [[customizations.directories]] 蓝图自定义在镜像中创建自定义文件和目录。您只能在 /etc 目录中使用这些自定义。

注意

所有镜像类型都支持这些蓝图自定义，但部署 OSTree 提交的镜像类型除外，如 edge-raw-image、edge-installer、edge-simplified-installer。

警告

如果您使用带有目录路径的 customizations.directories，且其在设置了 mode、user 或 group 的镜像中已存，则镜像构建无法防止更改现有目录的所有权或权限。

4.8.13.2. 在蓝图中指定自定义目录

使用 [[customizations.directory]] 蓝图自定义，您可以：

- 创建新目录。
- 为您要创建的目录设置用户和组所有权。
- 以八进制格式设置目录模式权限。

- 确保根据需要创建父目录。

使用 [[customizations.files]] 蓝图自定义，您可以：

- 创建新文本文件。
- 修改现有文件。警告：这可能会覆盖现有内容。
- 为您要创建的文件设置用户和组所有权。
- 以八进制格式设置 mode 权限。

注意

您无法创建或替换以下文件：

- /etc/fstab
- /etc/shadow
- /etc/passwd
- /etc/group

可用的自定义如下：

- 在蓝图中自定义文件系统配置：

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

MINIMUM-PARTITION-SIZE 值没有默认大小格式。蓝图自定义支持以下值和单位：kB 到 TB 以及 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"
minsize = 1073741824
```

- **使用单位定义挂载点大小。例如：**

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

```
[[customizations.filesystem]]
mountpoint = "/boot"
minsize = "1 GiB"
```

- **通过设置 minsize 来定义最小分区。例如：**

```
[[customizations.filesystem]]
mountpoint = "/var"
minsize = 2147483648
```

- **使用 [[customizations.directories]]，在 /etc 目录下为镜像创建自定义目录：**

```
[[customizations.directories]]
path = "/etc/directory_name"
mode = "octal_access_permission"
user = "user_string_or_integer"
group = "group_string_or_integer"
ensure_parents = boolean
```

蓝图条目如下所述：

  - **path - 必需 - 输入您要创建的目录的路径。它必须是 /etc 目录下的绝对路径。**
  - **mode - 可选 - 以八进制格式设置目录的访问权限。如果没有指定权限，则默认为**

0755。前面的零是可选的。

- user - 可选 - 将用户设为目录的所有者。如果没有指定用户，则默认为 root。您可以将用户指定为字符串或整数。
- group - 可选 - 将组设为目录的所有者。如果没有指定组，则默认为 root。您可以将组指定为字符串或整数。
- ensure_parents - 可选 - 指定是否根据需要创建父目录。如果没有指定值，则默认为 false。

- 使用 [[customizations.directories]]，在 /etc 目录下为镜像创建自定义文件：

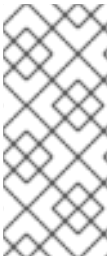
```
[[customizations.files]]
path = "/etc/directory_name"
mode = "octal_access_permission"
user = "user_string_or_integer"
group = "group_string_or_integer"
data = "Hello world!"
```

蓝图条目如下所述：

- path - 必需 - 输入您要创建的文件的路径。它必须是 /etc 目录下的绝对路径。
- mode 可选 - 以八进制格式设置文件的访问权限。如果没有指定权限，则默认为 0644。前面的零是可选的。
- user - 可选 - 将用户设为文件的所有者。如果没有指定用户，则默认为 root。您可以将用户指定为字符串或整数。
- group - 可选 - 将组设为文件的所有者。如果没有指定组，则默认为 root。您可以将组指定为字符串或整数。
- data - 可选 - 指定纯文本文件的内容。如果没有指定内容，它会创建一个空文件。

4.9. RHEL 镜像构建器安装的软件包

当使用 RHEL 镜像构建器创建系统镜像时，系统会安装一组基本软件包组。



注意

当您在蓝图中添加其他组件时，请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，系统无法解决依赖项并创建自定义镜像失败。您可以通过运行以下命令检查软件包之间没有冲突：

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

默认情况下，RHEL 镜像构建器使用 Core 组作为软件包的基本列表。

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, yum, nfs-utils, dnf-utils, cloud-init, python3-jsonschema, qemu-guest-agent, cloud-utils-growpart, dracut-norescue, tar, tcpdump, rsync, dnf-plugin-spacewalk, rhn-client-tools, rhnlib, rhnsd, rhn-setup, NetworkManager, dhcp-client, cockpit-ws, cockpit-system, subscription-manager-cockpit, redhat-release, redhat-release-eula, rng-tools, insights-client
tar	policycoreutils, selinux-policy-targeted
vhd	@core, langpacks-en
vmdk	@core, chrony, cloud-init, firewalld, langpacks-en, open-vm-tools, selinux-policy-targeted

镜像类型	默认软件包
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, 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, iwl6000g2b-firmware, jomolhari-fonts, kacst-farsi-fonts, kacst-qurn-fonts, kbd, kbd-misc, kdump-anaconda-addon, khmeros-base-fonts, libblockdev-lvm-dbus, libertas-sd8686-firmware, libertas-sd8787-firmware, libertas-usb8388-firmware, libertas-usb8388-olpc-firmware, libibverbs, libreport-plugin-bugzilla, libreport-plugin-reportuploader, libreport-rhel-anaconda-bugzilla, librsvg2, linux-firmware, 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-abyssinica-fonts, sil-padauk-fonts, sil-scheherazade-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**

镜像类型	默认软件包
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, policycoreutils, policycoreutils-python-utils, procps-ng, rootfiles, setools-console, sudo, traceroute, util-linux
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, 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
gce	@core, langpacks-en, acpid, dhcp-client, dnf-automatic, net-tools, python3, rng-tools, tar, vim

其他资源

- RHEL 镜像构建器描述

4.10. 在自定义镜像中启用服务

当使用镜像构建器配置自定义镜像时，镜像使用的默认服务由以下内容决定：

- 使用 osbuild-composer 工具的 RHEL 发行版本

- 镜像类型

例如，ami 镜像类型默认启用 sshd、chronyd 和 cloud-init 服务。如果没有启用这些服务，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	默认启用的服务
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
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

注：您可以自定义在系统引导过程中要启用的服务。但是，自定义不会覆盖上述镜像类型默认启用的服务。

其他资源

- 支持的镜像自定义

第 5 章 使用 RHEL 镜像构建器 WEB 控制台界面创建系统镜像

RHEL 镜像构建器是一个创建自定义系统镜像的工具。要控制 RHEL 镜像构建器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 RHEL 镜像构建器仪表盘

使用 RHEL web 控制台的 cockpit-composer 插件，您可以使用图形界面管理镜像构建器蓝图和 compose。

先决条件

- 您必须有对系统的 root 权限。
- 您已安装了 RHEL 镜像构建器。
- 您已安装了 cockpit-composer 软件包。

流程

1. 在主机上，在网页浏览器中打开 https://<_localhost_>:9090/。
2. 以 root 用户身份登录 Web 控制台。
3. 要显示 RHEL 镜像构建器控制，请单击窗口左上角的 Image Builder 按钮。

   RHEL 镜像构建器仪表盘打开，列出现有的蓝图（若有的话）。

其他资源

- 使用 RHEL 8 web 控制台管理系统

5.2. 在 WEB 控制台界面中创建蓝图

在创建自定义 RHEL 系统镜像前，创建蓝图是一个必要的步骤。所有自定义都是可选的。

注意

Red Hat Enterprise Linux 9.2 或更高版本，以及 Red Hat Enterprise Linux 8.8 或更高版本提供这些蓝图自定义。

先决条件

- 您已在浏览器中从 web 控制台打开了 RHEL 镜像构建器应用程序。请参阅 在 RHEL web 控制台中访问 RHEL 镜像构建器 GUI。

流程

1. 点右上角的 Create Blueprint。

   此时会打开一个对话框向导，其中包含蓝图名称和描述字段。

2. 在 Details 页面中：

   a. 输入蓝图的名称，以及可选的描述。

   b. 点击 Next。

3. 可选：在 Packages 页面中：

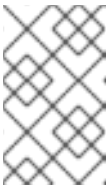
   a. 在 Available packages 搜索中，输入软件包名称

   b. 点 > 按钮将其移到 Chosen packages 字段中。

   c. 重复前面的步骤，以搜索并包含尽可能多的软件包。

d.

点击 **Next**。



注意

除非另有指定，否则这些自定义都是可选的。

4.

在 **Kernel** 页面中，输入内核名称和命令行参数。

5.

在 **File system** 页面中，您可以对镜像文件系统选择 **Use automatic partitioning** 或 **Manually configure partitions**。要手动配置分区，请完成以下步骤：

a.

点 **Manually configure partitions** 按钮。

此时会打开 **Configure partitions** 部分，显示基于红帽标准和安全指南的配置。

b.

在下拉菜单中，提供配置分区的详情：

i.

对于 **Mount point** 字段，选择以下挂载点类型选项之一：

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

- /usr
- /app
- /data
- /tmp
- /usr/local

您还可以向 Mount point 添加额外的路径，如 /tmp。例如：/var 作为前缀，/tmp 作为附加路径会产生 /var/tmp。

注意

根据您选择的挂载点类型，文件系统类型变为 xfs。

ii. 对于文件系统的 Minimum size partition 字段，请输入所需的最小分区大小。在 Minimum size 下拉菜单中，您可以使用通用大小单位，如 GiB、MiB 或 KiB。默认单位为 GiB。

注意

Minimum size 意味着 RHEL 镜像构建器仍然可以增加分区大小，以防它们太小而不能创建可正常工作的镜像。

c. 要添加更多分区，请点击 Add partition 按钮。如果您看到以下错误信息：Duplicate partitions: Only one partition at each mount point can be created.，您可以：

i. 点 Remove 按钮删除重复的分区。

ii. 为您要创建的分区选择一个新的挂载点。

d. 完成分区配置后，点 Next。

6. 在 Services 页面中，您可以启用或禁用服务：

a. 输入您要启用或禁用的服务名称，用逗号、空格或按 Enter 键分隔它们。点击 Next。

7. 在 Firewall 页面上，设置防火墙设置：

a. 输入 Ports，以及您要启用或禁用的防火墙服务。

b. 点 Add zone 按钮，为每个区域单独管理您的防火墙规则。点击 Next。

8. 在 Users 页面上，按照以下步骤添加用户：

a. 单击 Add user。

b. 输入 Username、password，以及 SSH key。您还可以单击 Server administrator 复选框，将用户标记为特权用户。点击 Next。

9. 在 Groups 页面上，完成以下步骤来添加组：

a. 点 Add groups 按钮：

i. 输入 Group name 和 Group ID。您可以添加多个组。点击 Next。

10. 在 SSH keys 页面上，添加一个密钥：

a. 点 Add key 按钮。

i. 输入 SSH 密钥

输入 SSH 密钥。

ii. 输入 User。点击 Next。

11. 在 Timezone 页面上，设置您的时区设置：

a. 在 Timezone 字段上，输入您要添加到系统镜像的时区。例如，添加以下时区格式："US/Eastern"。

如果您没有设置时区，系统将使用 Universal Time, Coordinated (UTC)作为默认值。

b. 输入 NTP 服务器。点击 Next。

12. 在 Locale 页面上，完成以下步骤：

a. 在 Keyboard 搜索字段中，输入您要添加到系统镜像的软件包名称。例如：["en_US.UTF-8"]。

b. 在 Languages 搜索字段中，输入您要添加到系统镜像的软件包名称。例如："us"。点击 Next。

13. 在 Others 页面上，完成以下步骤：

a. 在 Hostname 字段上，输入您要添加到系统镜像的主机名。如果没有添加主机名，操作系统会决定主机名。

b. 仅对 Simplifier Installer 镜像是必需的：在 Installation Devices 字段上，输入您系统镜像的有效节点。例如：dev/sda1。点击 Next。

14. 仅在构建 FIDO 镜像时才是必需的：在 FIDO device onboardin 页面上，完成以下步骤：

a. 在 Manufacturing server URL 字段中输入以下信息：

i. 在 DIUN public key insecure 字段中，输入不安全的公钥。

ii. 在 DIUN public key hash 字段中，输入公钥哈希。

iii. 在 DIUN public key root certs 字段中，输入公钥根证书。点击 Next。

15. 在 OpenSCAP 页面上，完成以下步骤：

a. 在 Datastream 字段上，输入您要添加到系统镜像的 datastream 补救指令。

b. 在 Profile ID 字段删，输入您要添加到系统镜像的 profile_id 安全配置文件。点击 Next。

16. 仅在构建 Ignition 镜像时必需：在 Ignition 页面上，完成以下步骤：

a. 在 Firstboot URL 字段上，输入您要添加到系统镜像的软件包名称。

b. 在 Embedded Data 字段上，拖动或上传您的文件。点击 Next。

17. .在 Review 页面上，查看蓝图的详情。点 Create。

RHEL 镜像构建器视图打开，列出现有的蓝图。

5.3. 在 RHEL 镜像构建器 WEB 控制台界面中导入蓝图

您可以导入并使用已存在的蓝图。系统会自动解决所有依赖项。

先决条件

- 您已在浏览器中从 web 控制台打开了 RHEL 镜像构建器应用程序。

- 您有一个要导入到 RHEL 镜像构建器 web 控制台界面中使用的蓝图。

流程

1. 在 RHEL 镜像构建器仪表盘上，点 Import blueprint。此时会打开 Import blueprint 向导。
2. 在 Upload 字段中，拖放或上传现有的蓝图。此蓝图可以是 TOML 或 JSON 格式。
3. 点 Import。仪表盘列出了您导入的蓝图。

验证

当您点您导入的蓝图时，您可以访问带有您导入蓝图的所有自定义的仪表盘。

- 要验证已为导入蓝图选择的软件包，请导航到 Packages 选项卡。
  - 要列出所有软件包依赖项，请单击 All。列表是可搜索的，并可排序。

后续步骤

- 可选：要修改任何自定义：
  - 在 Customizations 仪表盘中，点击您要更改的自定义。另外，您可以点击 Edit blueprint 来导航到所有可用的自定义选项。

其他资源

- 在 web 控制台界面中使用 RHEL 镜像构建器创建系统镜像。

5.4. 从 RHEL 镜像构建器 WEB 控制台界面导出蓝图

您可以导出蓝图以便在另一个系统中使用自定义。您可以以 TOML 或 JSON 格式导出蓝图。这两个格式在 CLI 和 API 接口中都可以工作。

先决条件

- 您已在浏览器中从 web 控制台打开了 RHEL 镜像构建器应用程序。
- 您有一个要导出的蓝图。

流程

1. 在镜像构建器仪表盘上，选择您要导出的蓝图。
2. 点 Export blueprint。此时会打开 Export blueprint 向导。
3. 点 Export 按钮将蓝图下载为一个文件，或者点击 Copy 按钮将蓝图复制到剪贴板。
   a. （可选）点击 Copy 按钮来复制蓝图。

验证

- 在文本编辑器中打开导出的蓝图，以检查并审核它。

5.5. 在 WEB 控制台界面中使用 RHEL 镜像构建器创建系统镜像

您可以通过完成以下步骤，从蓝图创建一个自定义 RHEL 系统镜像。

先决条件

- 在浏览器中从 web 控制台打开 RHEL 镜像构建器应用程序。
- 您创建了蓝图。

流程

1. 在 RHEL 镜像构建器仪表盘中，点 blueprint 标签页。

2. 在蓝图表中，找到您要构建镜像的蓝图。

3. 在所选蓝图的右侧，点 Create Image。Create image 对话框向导将打开。

4. 在 Image 输出页面中完成以下步骤：

   a. 从 Select a blueprint 列表中，选择您想要的镜像类型。

   b. 从 Image output type 列表中，选择您想要的镜像输出类型。

   根据您选择的镜像类型，您需要添加更多详细信息。

5. 点击 Next。

6. 在 Review 页面上，查看关于镜像创建的详情，并点 Create image。

   镜像构建启动，需要 20 分钟完成。

验证

镜像完成构建后，您可以：

- 下载镜像。
  - 在 RHEL 镜像构建器仪表盘上，点 Node options( ) 菜单，然后选择 Download image。
- 下载镜像的日志以检查元素，并验证是否发现了任何问题。
  - 在 RHEL 镜像构建器仪表盘上，点 Node options( ) 菜单，然后选择 Download logs。

第 6 章 使用 RHEL 镜像构建器从不同的版本创建系统镜像

您可以使用 RHEL 镜像构建器创建与主机不同的多个 RHEL 次版本的镜像，如 RHEL 8.8 和 RHEL 8.7。为此，您可以使用设置发行版本发布字段添加源系统仓库，也可以使用正确的发行版本发布字段创建蓝图。

另外，如果您以旧格式拥有现有的蓝图或源系统程序库，您可以使用正确的发行版本发布字段创建新蓝图。

- 要列出支持的发行版本发行版本，您可以运行以下命令：

```
$ composer-cli distros list
```

输出显示支持的发行版本分布名称的列表：

```
rhel-8
rhel-84
rhel-85
rhel-86
rhel-87
rhel-88
rhel-89
```

注意

不支持跨发布镜像构建，如在 RHEL 上构建 CentOS 镜像。

6.1. 通过 CLI 创建一个使用不同发布的镜像

要在 RHEL 镜像构建器 CLI 中制作镜像时选择要使用的发行版，您必须在蓝图中设置 distro 字段。为此，请按照以下步骤操作：

流程

如果要创建新蓝图

1. 创建一个蓝图。例如：

```
name = "<blueprint_name>"
```

```
description = "<image-description>"
version = "0.0.1"
modules = []
groups = []
distro = "<distro-version>"
```

通过将 distro 字段设置为 "rhel-88"，您可以确保它始终构建 RHEL 8.8 镜像，无论主机上运行的是哪个版本。

注意

如果 distro 字段为空，它会使用相同的主机分布。

如果要更新现有蓝图

1. **将现有蓝图保存（导出）到本地文本文件：**

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

1. **使用您选择的文本编辑器编辑现有蓝图文件，使用您选择的发布设置 distro 字段，例如：**

```
name = "blueprint_84"
description = "A 8.8 base image"
version = "0.0.1"
modules = []
groups = []
distro = "rhel-88"
```

2. **保存文件并关闭编辑器。**

3. **将蓝图推送（导入）回 RHEL 镜像构建器：**

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

4. **启动镜像创建：**

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

等待 compose 完成。

5. 检查 compose 的状态：

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

完成 compose 后，它会显示一个 FINISHED 状态值。根据 UUID 识别列表中的内容。

6. 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。

6.2. 使用特定发行版本的系统仓库

您可以在解析依赖项和构建镜像时指定系统存储库源使用的分发字符串列表。为此，请按照以下步骤执行：

流程

- 创建一个具有以下结构的 TOML 文件，例如：

```
check_gpg = true
check_ssl = true
distros = ["<distro-version>"]
id = "<image-id>"
name = "<image-name>_"
system = false
type = "<image-type>"
url = "\http://local/repos/rhel-<distro-version>_/<project-repo>/"
```

例如：

```
check_gpg = true
check_ssl = true
distros = ["rhel-84"]
```

```
id = "rhel-84-local"
name = "local packages for rhel-84"
system = false
type = "yum-baseurl"
url = "\http://local/repos/rhel-84/projectrepo/"
```

其他资源

- 管理存储库.

第 7 章 使用 RHEL 镜像构建器创建引导 ISO 安装程序镜像

您可以使用 RHEL 镜像构建器创建可引导的 ISO 安装程序镜像。这些镜像由一个带有根文件系统的 .tar 文件组成。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。

RHEL 镜像构建器构建一个清单，该清单创建一个包含根文件系统的引导 ISO。要创建 ISO 镜像，请选择映像类型 image-installer。RHEL 镜像构建器构建一个具有以下内容的 .tar 文件：

- 标准 Anaconda 安装程序 ISO
- 嵌入式 RHEL 系统 tar 文件
- 安装提交的默认 Kickstart 文件，其要求最小

创建的安装程序 ISO 镜像包含一个预先配置的系统镜像，您可以直接安装到裸机服务器。

7.1. 使用 RHEL 镜像构建器 CLI 创建一个引导 ISO 安装程序镜像

您可以使用 RHEL 镜像构建器命令行界面创建一个自定义的引导 ISO 安装程序镜像。因此，镜像构建器构建一个包含 .tar 文件的 .iso 文件，您可以为操作系统安装该文件。.iso 文件被设置为引导 Anaconda，并安装 .tar 文件来建立系统。您可以使用在硬盘上创建的 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。

警告

安装程序(.iso)镜像类型不接受分区自定义。如果您尝试手动配置文件系统自定义，它不会应用到安装程序镜像构建的任何系统。挂载使用 RHEL 镜像构建器文件系统自定义构建的 ISO 镜像会在 Kickstart 中导致一个错误，且安装不会自动重启。如需更多信息，请参阅 自动化由镜像构建器产生的 RHEL ISO 安装，并 自动化镜像构建器产生的 RHEL ISO 安装。

先决条件

-

您已为镜像创建了一个蓝图，并使用包含的用户自定义了它，并将其推送回 RHEL 镜像构建器。请参阅 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。等待 compose 完成。这可能需要几分钟。

2. 检查 compose 的状态：

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

完成的 compose 显示 FINISHED 状态值。

3. 根据 UUID 识别列表中的内容。

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

4. 完成 compose 后，将创建的镜像文件下载到当前目录中：

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

将 ***UUID*** 替换为前面步骤中获取的 UUID 值。

RHEL 镜像构建器构建一个包含 .tar 文件的 .iso 文件。.tar 文件是要为操作系统安装的镜像。. iso 被设置为引导 Anaconda，并安装 .tar 文件来建立系统。

后续步骤

在下载镜像文件的目录中。

1. 找到您下载的 .iso 镜像。
2. 挂载 ISO。

```
$ mount -o ro path_to_ISO /mnt
```

您可以在 /mnt/liveimg.tar.gz 目录中找到 .tar 文件。

3. 列出 .tar 文件内容：

```
$ tar ztvf /mnt/liveimg.tar.gz
```

其他资源

- 使用 RHEL 镜像构建器命令行界面创建系统镜像
- 为 RHEL 创建一个可引导的安装介质

7.2. 在 GUI 中使用 RHEL 镜像构建器创建一个引导 ISO 安装程序镜像

您可以使用 RHEL 镜像构建器 GUI 构建一个自定义引导 ISO 安装程序镜像。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它。例如，在 HTTP Boot 或者 USB 安装中。

警告

安装程序(.iso)镜像类型不接受分区自定义。如果您尝试手动配置文件系统自定义，它不会应用到安装程序镜像构建的任何系统。挂载使用 RHEL 镜像构建器文件系统自定义构建的 ISO 镜像会在 Kickstart 中导致一个错误，且安装不会自动重启。如需更多信息，请参阅 自动化由镜像构建器产生的 RHEL ISO 安装，并 自动化镜像构建器产生的 RHEL ISO 安装。

先决条件

- 您已在浏览器中从 web 控制台打开了 RHEL 镜像构建器应用程序。
- 您已为镜像创建了一个蓝图。请参阅在 web 控制台界面中创建 RHEL 镜像构建器蓝图。

流程

1. 在 RHEL 镜像构建器仪表盘上，找到您要用来构建镜像的蓝图。可选，在左上角的搜索框中输入蓝图名称或部分名称，然后点 Enter。
2. 在蓝图的右侧，点对应的 Create Image 按钮。

   Create image 对话框向导将打开。
3. 在 Create image 对话框向导中：
   a. 在 Image Type 列表中，选择 "RHEL Installer (.iso) "。
   b. 点击 Next。
   c. 在 Review 选项卡中，点 Create。

   RHEL 镜像构建器将 RHEL ISO 镜像的组成添加到队列中。

过程完成后，您可以看到镜像构建完成状态。RHEL 镜像构建器创建 ISO 镜像。

验证

成功创建镜像后，您可以下载该镜像。

1. 点 Download 将 "RHEL Installer (.iso) " 镜像保存到您的系统中。
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。

```
$ tar -xf content.tar
```

其他资源

- 在 web 控制台界面中创建一个 RHEL 镜像构建器蓝图
- 使用 RHEL 镜像构建器命令行界面创建系统镜像
- 为 RHEL 创建一个可引导的安装介质

7.3. 将可引导 ISO 安装到介质并引导它

将您使用 RHEL 镜像构建器创建的可引导 ISO 镜像安装到裸机系统。

先决条件

- 您已使用 RHEL 镜像构建器创建了可引导 ISO 镜像。请参阅 在命令行界面中使用 RHEL 镜像构建器创建一个引导 ISO 安装程序镜像。

- 您已下载了可引导 ISO 镜像。
- 已安装 dd 工具。
- 您有一个有足够容量的 USB 闪存驱动器 ISO 镜像。所需的大小因您在蓝图中选择的软件包而异，但推荐的最小值为 8 GB。

流程

1. 使用 dd 工具将可引导 ISO 镜像直接写入 USB 驱动器。例如：

```
dd if=installer.iso of=/dev/sdX
```

   其中 installer.iso 是 ISO 镜像文件名，/dev/sdX 是您的 USB 闪存驱动器设备路径。

2. 将闪存驱动器插入到您要引导的计算机的 USB 端口中。
3. 从 USB 闪存引导 ISO 镜像。

   当安装环境启动时，您可能需要手动完成安装，类似于默认的 Red Hat Enterprise Linux 安装。

其他资源

- 引导安装
- 自定义安装
- 在 Linux 中创建可引导 USB 设备

第 8 章 使用 RHEL 镜像构建器 OPENSCAP 集成创建预强化的镜像

内部 RHEL 镜像构建器支持 OpenSCAP 集成。这个集成启用了预强化的 RHEL 镜像的生产。通过设置蓝图，您可以执行以下操作：

- 使用一组预定义的安全配置文件对其进行自定义
- 添加一组软件包或附加文件
- 构建一个更适合您的环境、准备部署到所选平台上的自定义 RHEL 镜像

红帽为构建系统时可以选择的安全强化配置集定期更新版本，以便您能够满足您的当前部署指南。

8.1. KICKSTART 和预先强化的镜像之间的区别

对于使用 Kickstart 文件创建的传统镜像，您需要选择要安装哪些软件包，并确保系统不受安全漏洞的影响。通过 RHEL 镜像构建器 OpenSCAP 集成，您可以构建强化安全的镜像。在镜像构建过程中，OSBuild oscap.remediation 阶段 在 chroot （文件系统树）中运行 OpenSCAP 工具。OpenSCAP 工具为您选择的配置集运行标准评估，并将补救应用于镜像。因此，您可以在镜像首次引导前根据安全配置集要求配置的镜像。

8.2. 安装 OPENSCAP

安装 OpenSCAP 工具可访问 SCAP 工具，以帮助您为您的系统创建标准安全清单。

流程

1. 在您的系统上安装 OpenSCAP ：

```
# *yum install openscap-scanner*
```

2. 安装 scap-security-guide 软件包：

```
# *yum install scap-security-guide*
```

安装完成后，您可以使用 oscap 命令行工具开始。带有安全配置集的 SCAP 内容将安装在 /usr/share/xml/scap/ssg/content/ 目录中。

8.3. OPENSCAP 蓝图自定义

通过对蓝图自定义的 OpenSCAP 支持，您可以为特定安全配置集生成来自 scap-security-guide 内容的蓝图，然后使用它们构建自己的预先强化的镜像。要创建自定义预先强化的镜像，您可以修改挂载点并根据具体要求配置文件系统布局。选择 OpenSCAP 配置集后，OpenSCAP 蓝图配置镜像，以根据所选配置集在镜像构建过程中触发补救。在镜像构建过程中，OpenSCAP 应用预启动修复。

要在镜像蓝图中使用 OpenSCAP 蓝图，您需要提供以下信息：

- 数据流路径到 datastream 修复指令。scap-security-guide 软件包中的数据流文件位于 /usr/share/xml/scap/ssg/content/ 目录中。1

- 所需的安全配置集的 profile_id。profile_id 字段的值接受长和短形式，例如，以下是可接受的：cis 或 xccdf_org.ssgproject.content_profile_cis。如需了解更多详细信息，请参阅 RHEL 8 支持的 SCAP 安全指南配置集。

  以下示例是带有 OpenSCAP 补救阶段的蓝图：

```
[customizations.openscap]
# If you want to use the data stream from the 'scap-security-guide' package
# the 'datastream' key could be omitted.
# datastream = "/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml"
profile_id = "xccdf_org.ssgproject.content_profile_cis"
```

  您可以使用以下命令，从 scap-security-guide 软件包中查找 SCAP 源数据流的更多详细信息，包括它提供的安全配置集列表：

```
# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

为方便起见，OpenSCAP 工具可以为 scap-security-guide 数据流中的任何配置集生成强化蓝图。

例如，命令

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

将为 CIS 配置文件生成蓝图，类似于

```
# Blueprint for CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server
#
# Profile Description:
# This profile defines a baseline that aligns to the "Level 2 - Server"
# configuration from the Center for Internet Security® Red Hat Enterprise
# Linux 8 Benchmark™, v3.0.0, released 2023-10-30.
# This profile includes Center for Internet Security®
# Red Hat Enterprise Linux 8 CIS Benchmarks™ content.
#
# Profile ID:  xccdf_org.ssgproject.content_profile_cis
# Benchmark ID:  xccdf_org.ssgproject.content_benchmark_RHEL-8
# Benchmark Version:  0.1.74
# XCCDF Version:  1.2

name = "hardened_xccdf_org.ssgproject.content_profile_cis"
description = "CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server"
version = "0.1.74"

[customizations.openscap]
profile_id = "xccdf_org.ssgproject.content_profile_cis"
# If your hardening data stream is not part of the 'scap-security-guide' package
# provide the absolute path to it (from the root of the image filesystem).
# datastream = "/usr/share/xml/scap/ssg/content/ssg-xxxxx-ds.xml"

[[customizations.filesystem]]
mountpoint = "/home"
size = 1073741824

[[customizations.filesystem]]
mountpoint = "/tmp"
size = 1073741824

[[customizations.filesystem]]
mountpoint = "/var"
size = 3221225472

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

[[packages]]
name = "aide"
version = "*"

[[packages]]
name = "libselinux"
version = "*"
```

```
[[packages]]
name = "audit"
version = "*"

[customizations.kernel]
append = "audit_backlog_limit=8192 audit=1"

[customizations.services]
enabled = ["auditd","crond","firewalld","systemd-journald","rsyslog"]
disabled = []
masked = ["nfs-server","rpcbind","autofs","bluetooth","nftables"]
```

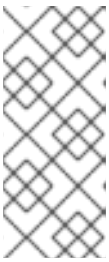
注意

不要将此精确的蓝图片断用于镜像强化。它没有反映完整的配置集。当红帽不断更新并优化 scap-security-guide 软件包中每个配置集的安全要求时，始终使用为您的系统提供的最最新的数据流版本来重新生成初始模板。

现在，您可以自定义蓝图，或者在构建镜像时使用它。

RHEL 镜像构建器根据您的蓝图自定义，为 osbuild 阶段产生所需的配置。另外，RHEL 镜像构建器向镜像中添加了两个软件包：

- openscap-scanner - OpenSCAP 工具。
- scap-security-guide - 包含修复和评估指令的软件包。



注意

补救阶段将 scap-security-guide 软件包用于 datastream，因为这个软件包默认安装在镜像中。如果要使用不同的数据流，将必要的软件包添加到蓝图中，并在 oscap 配置中指定到 datastream 的路径。

其他资源

- RHEL 8 支持的 SCAP 安全指南配置文件。

8.4. 使用 RHEL 镜像构建器创建一个预强化的镜像

有了 OpenSCAP 和 RHEL 镜像构建器集成，您可以创建可在虚拟机中部署的预强化镜像。

前提条件

- 您以 root 用户身份或 welder 组成员的身份登录。

流程

1. 以 TOML 格式创建强化蓝图，使用 OpenSCAP 工具和 scap-security-guide 内容，并根据需要进行修改：

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml > cis.toml
```

2. 使用 composer-cli 工具将蓝图推送到 osbuild-composer ：

```
# composer-cli blueprints push cis.toml
```

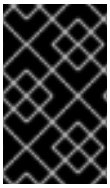
3. 启动强化镜像的构建：

```
# composer-cli compose start hardened_xccdf_org.ssgproject.content_profile_cis qcow2
```

镜像构建就绪后，您可以在部署中使用预先强化的镜像。请参阅创建虚拟机。

验证

在虚拟机中部署预先强化的镜像后，可以执行配置合规性扫描，以验证镜像是否与所选安全配置集一致。



重要

执行配置合规性扫描不能保证系统是合规的。如需更多信息，请参阅配置合规性扫描。

1. 使用 SSH 连接到虚拟机。

2. 运行 oscap 扫描程序。

```
# oscap xccdf eval --profile=cis --report=/tmp/compliance-report.html
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml
```

3. 获取 compliance-report.html 并检查结果。

其他资源

- 扫描系统的配置合规性和漏洞。

8.5. 在蓝图中为配置集添加自定义定制选项

通过 OpenSCAP 和 RHEL 镜像构建器集成，您可以使用以下选项将配置集的自定义定制选项添加到蓝图自定义中：

- selected 用于您要添加的规则的列表
- unselected 用于您要删除的规则的列表

使用默认的 org.ssgproject.content 规则命名空间，您可以在此命名空间下省略规则的前缀。例如，org.ssgproject.content_grub2_password 和 grub2_password 的功能相当。

当您从该蓝图构建镜像时，它会使用新的定制配置文件 ID 创建一个定制文件，并将其保存为 /usr/share/xml/osbuild-oscap-tailoring/tailoring.xml 镜像。新配置文件 ID 将 _osbuild_tailoring 作为后缀附加到基本配置文件。例如，如果您使用 CIS (cis)基础配置集，配置集 ID 为 xccdf_org.ssgproject.content_profile_cis_osbuild_tailoring。

先决条件

- 您以 root 用户身份或 welder 组成员的身份登录。

流程

1. 以 TOML 格式创建强化蓝图，使用 OpenSCAP 工具和 scap-security-guide 内容，并根据需要进行修改：

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml > cis_tailored.toml
```

2. **将 tailoring 部分的自定义规则集附加到蓝图中：**

```
# Blueprint for CIS Red Hat Enterprise Linux {ProductNumber} Benchmark for Level 2 -
Server
# ...

[customizations.openscap.tailoring]
selected = [ "xccdf_org.ssgproject.content_bind_crypto_policy" ]
unselected = [ "grub2_password" ]
```

3. **使用 composer-cli 工具将蓝图推送到 osbuild-composer ：**

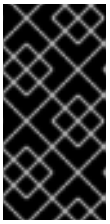
```
# composer-cli blueprints push cis_tailored.toml
```

4. **启动强化镜像的构建：**

```
# composer-cli compose start hardened_xccdf_org.ssgproject.content_profile_cis qcow2
```

镜像构建就绪后，请在部署中使用预先强化的镜像来创建虚拟机。如需更多信息，请参阅 创建虚拟机。

在虚拟机中部署预先强化的镜像后，可以执行配置合规性扫描，以验证镜像是否与所选安全配置集一致。



重要

执行配置合规性扫描不能保证系统是合规的。如需更多信息，请参阅配置合规性扫描。

验证

在部署预先强化的镜像的虚拟机中，请按照以下步骤执行：

1. 使用 SSH 连接到虚拟机。

2. 运行 oscap 扫描程序。

```
# oscap xccdf eval --profile=cis --report=/tmp/compliance-report.html
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml
```

3. 获取 compliance-report.html 并检查结果。

其他资源

- 扫描系统的配置合规和漏洞。

第 9 章 使用 RHEL 镜像构建器准备并部署 KVM 客户机镜像

使用 RHEL 镜像构建器创建一个专门构建的 .qcow2，您可以在基于 hypervisor 的基于 Kernel 的虚拟机(KVM)上部署。

创建自定义 KVM 客户机镜像涉及以下高级别步骤：

1. 为 .qcow2 镜像创建一个蓝图。
2. 使用 RHEL 镜像构建器创建一个 .qcow2 镜像。
3. 从 KVM 客户机镜像创建一个虚拟机。

9.1. 使用 RHEL 镜像构建器创建自定义 KVM 客户机镜像

您可以使用 RHEL 镜像构建器创建一个自定义 .qcow2 KVM 客户机镜像。以下流程演示了 GUI 上的步骤，但您也可以使用 CLI。

先决条件

- 您必须位于 root 或 weldr 组中，以访问系统。
- cockpit-composer 软件包已安装。
- 在 RHEL 系统上，您已打开了 web 控制台的 RHEL 镜像构建器仪表盘。
- 您已创建了蓝图。请参阅 在 web 控制台界面中创建蓝图。

流程

1. 点您创建的蓝图名称。

2. 选择 **Images** 选项卡。

3. 单击 **Create Image** 来创建自定义镜像。**Create Image** 窗口打开。

4. 在 **Type** 下拉菜单中选择 **QEMU Image (.qcow2)**。

5. 在实例化时设置您想要的镜像大小，并点击 **Create**。

6. 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

- 点击 **breadcrumbs** 图标，并选择 **Download** 选项。**RHEL** 镜像构建器将 **KVM** 客户机镜像 **.qcow2** 文件下载到默认下载位置。

其他资源

- 在 web 控制台界面中创建蓝图。

9.2. 从 KVM 客户机镜像创建虚拟机

使用 **RHEL** 镜像构建器，您可以构建一个 **.qcow2** 镜像，并使用 **KVM** 客户机镜像创建虚拟机。使用 **RHEL** 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

- 您已使用 **RHEL** 镜像构建器创建了一个 **.qcow2** 镜像。请参阅 在 web 控制台界面中创建一个蓝图。

- 您已在系统上安装了 **qemu-kvm** 软件包。您可以检查 **/dev/kvm** 设备是否在您的系统上可用，并且 **BIOS** 中是否已启用了虚拟化功能。

- 在您的系统上已安装了 libvirt 和 virt-install 软件包。
- 您有 genisoimage 工具，其由 xorriso 软件包提供，已安装在您的系统上。

流程

1. 将您使用 RHEL 镜像构建器创建的 .qcow2 镜像移到 /var/lib/libvirt/images/ 目录中。
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: vmname
```

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: password
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
  - ssh-rsa AAA...fhHQ== your.email@example.com
```

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

5. 使用 genisoimage 工具创建一个包含 user-data 和 meta-data 文件的 ISO 镜像。

```
# genisoimage -output cloud-init.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 myvm \
    --disk rhel-8-x86_64-kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
    --disk cloud-init.iso,device=cdrom \
    --os-variant rhel 8 \
    --virt-type kvm \
    --graphics none \
    --import
```

- **--graphics none - 表示它是一个无头的 RHEL 8 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 (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

验证

引导完成后，虚拟机会显示文本登录界面。要登录到虚拟机的本地控制台，请使用 user-data 文件中的详情：

1. **输入 admin 作为用户名，然后按 Enter 键。**

2. 输入 password 作为密码，然后按 Enter 键。

登录身份验证完成后，您可以使用 CLI 访问虚拟机。

其他资源

- 启用虚拟化.
- 为 RHEL 8 配置和管理 cloud-init。
- cloud-init 重要目录和文件.

第 10 章 将容器推送到 REGISTRY 中并将其嵌入到镜像中

使用 RHEL 镜像构建器，您可以使用 OpenSCAP 工具构建强化安全的镜像。您可以在蓝图中利用容器自定义支持来创建容器，并将其直接嵌入到您创建的镜像中。

10.1. 蓝图自定义，将容器嵌入到镜像中

要从 registry.access.redhat.com registry 中嵌入容器，您必须在蓝图中添加容器自定义。例如：

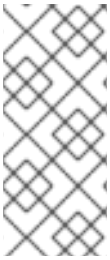
```
[[containers]]
source = "registry.access.redhat.com/ubi9/ubi:latest"
name =  "local-name"
tls-verify = true
```

- source - 必需的字段。它是对 registry 中的容器镜像的引用。这个示例使用 registry.access.redhat.com registry。您可以指定标签版本。默认标签版本为 latest。
- name - 本地注册中心中的容器的名称。
- tls-verify - 布尔值字段。tls-verify 布尔值字段控制传输层安全性。默认值为 true。

RHEL 镜像构建器在镜像构建过程中拉取容器，并将容器存储为镜像。默认本地容器存储位置取决于镜像类型，因此都支持 container-tools，如 Podman 可以使用它。嵌入的容器没有启动。要访问受保护的容器资源，您可以使用 containers-auth.json 文件。

10.2. 容器 REGISTRY 凭证

osbuild-worker 服务负责与容器 registry 的通信。要启用此功能，您可以设置 /etc/osbuild-worker/osbuild-worker.toml 配置文件。



注意

在设置 /etc/osbuild-worker/osbuild-worker.toml 配置文件后，您必须重启 osbuild-worker 服务，因为在 osbuild-worker 服务启动时，它只会读取 /etc/osbuild-worker/osbuild-worker.toml 配置文件一次。

/etc/osbuild-worker/osbuild-worker.toml 配置文件有一个 containers 部分，它带有

auth_field_path 条目，其值是是一个字符串代表用于访问受保护的资源的 containers-auth.json 文件路径。容器 registry 凭证仅在将容器嵌入到镜像中时用于从 registry 中拉取容器镜像。

例如：

```
[containers]
auth_file_path = "/etc/osbuild-worker/containers-auth.json"
```

其他资源

- containers-auth.json 手册页

10.3. 将容器工件直接推送到容器 REGISTRY

您可以使用 RHEL 镜像构建器 CLI 将容器工件（如 RHEL for Edge 容器镜像）直接推送到容器注册中心。

先决条件

- 访问 quay.io registry。本例使用 quay.io 容器 registry 作为目标 registry，但您可以使用您选择的容器 registry。

流程

1. 设置 registry-config.toml 文件以选择容器提供程序。凭证是可选的。

```
provider = "container_provider"

[settings]
tls_verify = false
username = "admin"
password = "your_password"
```

2. 使用 .toml 格式创建蓝图。这是您在蓝图中安装 nginx 软件包的容器蓝图。

```
name = "simple-container"
description = "Simple RHEL container"
version = "0.0.1"
```

```
[[packages]]
name = "nginx"
version = "*"
```

3. **推送蓝图：**

```
# composer-cli blueprints push blueprint.toml
```

4. **通过将注册中心和存储库作为参数传给 composer-cli 来构建容器镜像。**

```
# composer-cli compose start simple-container container "quay.io:8080/osbuild/repository"
registry-config.toml
```

- **simple-container - 是蓝图名称。**
- **Container - 是镜像类型。**
- **"quay.io:8080/osbuild/*repository*" - quay.io 是目标 registry, osbuild 是机构，repository 是在构建完成后要推送到的容器的位置。另外，您可以设置一个 tag。如果没有为 :tag 设置值，则默认为 :latest 标签。**

注意

构建容器镜像需要一些时间，因为需要解析自定义软件包的依赖项。

5. **在镜像构建完成后，您创建的容器将出现在 quay.io 中。**

验证

1. **打开 quay.io。然后单击 Repository Tags。**

```
You can see details about the container you created, such as:
- last modified
- image size
- the `manifest ID`, that you can copy to the clipboard.
```

2. 复制 清单 ID 值以构建您要嵌入容器的镜像。

其他资源

- Quay.io - 使用标签。

10.4. 构建镜像并将容器提取到镜像中

创建容器镜像后，您可以构建自定义镜像并将容器镜像提取到其中。为此，您必须在蓝图中指定 容器自定义，以及最终镜像的 容器名称。在构建过程中，会获取容器镜像，并放置在本地 Podman 容器存储中。

先决条件

- 您创建了容器镜像并将其推送到本地 quay.io 容器 registry 实例。请参阅 将容器工件直接推送到容器注册中心。
- 您可以访问 registry.access.redhat.com。
- 您有一个容器 清单 ID。
- 已安装 qemu-kvm 和 qemu-img 软件包。

流程

1. 创建蓝图来构建 qcow2 镜像。蓝图必须包含 自定义。

```
name = "image"
description = "A qcow2 image with a container"
version = "0.0.1"
distro = "rhel-90"

[[packages]]
name = "podman"
version = "*"

[[containers]]
source = "registry.access.redhat.com/ubi9:8080/osbuild/container/container-
```

```
image@sha256:manifest-ID-from-Repository-tag: tag-version"
name =  "source-name"
tls-verify = true
```

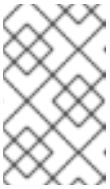
2. **推送蓝图：**

```
# composer-cli blueprints push blueprint-image.toml
```

3. **构建容器镜像：**

```
# composer-cli start compose image qcow2
```

- ***image* 是蓝图名称。**
- **qcow2 是镜像类型。**



注意

构建镜像需要一些时间，因为它将检查 quay.io registry 上的容器。

4. **检查 Compose 的状态：**

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

完成的 compose 显示 FINISHED 状态值。要识别列表中您的 compose，请使用其 UUID。

5. **compose 进程完成后，将生成的镜像文件下载到您默认的下载位置：**

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

使用前面步骤中显示的 UUID 值来替换 UUID。

您可以使用您创建并下载的 qcow2 镜像来创建虚拟机。

验证

从下载的 qcow2 镜像执行以下步骤：

1. 在虚拟机中启动 qcow2 镜像。请参阅从 KVM 客户机镜像创建虚拟机。

2. qemu 向导将打开。登录到 qcow2 镜像。

   a. 输入用户名和密码。这些可以是您在 "customizations.user" 部分的 .qcow2 蓝图中设置的用户名和密码，或是在引导时使用 cloud-init 创建的。

3. 运行容器镜像，并在容器内打开 shell 提示符：

```
# podman run -it registry.access.redhat.com/ubi9:8080/osbuild/repository /bin/bash/
```

   registry.access.redhat.com 是目标 registry，osbuild 是机构，repository 则是在完成构建时推送容器的位置。

4. 检查您添加到蓝图中的软件包是否可用：

```
# type -a nginx
```

   输出显示 nginx 软件包路径。

其他资源

- 红帽容器 Registry 身份验证.
- 访问和配置红帽 Registry。
- 基本 Podman 命令。

- 在容器中运行 **Skopeo**。

第 11 章 使用 RHEL 镜像构建器准备并上传云镜像

RHEL 镜像构建器可以创建准备在各种云平台上使用的自定义系统镜像。要在云中使用自定义的 RHEL 系统镜像，请使用 RHEL 镜像构建器，使用所选输出类型创建系统镜像，配置您的系统以上传镜像，并将镜像上传到您的云帐户。您可以通过 RHEL web 控制台中的 Image Builder 应用程序将自定义镜像推送到云，其可用于我们支持的服务提供商的子集，如 AWS 和 Microsoft Azure 云。请参阅 将镜像推送到 AWS Cloud AMI 和 将 VHD 镜像推送到 Microsoft Azure 云。

11.1. 准备上传 AWS AMI 镜像

在上传 AWS AMI 镜像前，您必须配置系统来上传镜像。

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：

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

2. 使用 pip 安装 AWS 命令行工具 ：

```
# pip3 install awscli
```

3. 设置您的配置集。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. 要授予访问 S3 存储桶的权限，如果您还没有这样做，请在 AWS Identity and Access Management (IAM) 中创建一个 vmimport S3 角色：

   a. 创建一个 JSON 格式的带有信任策略配置的 trust-policy.json 文件。例如：

```
{
   "Version": "2022-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Principal": {
         "Service": "vmie.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
         "StringEquals": {
            "sts:Externalid": "vmimport"
         }
      }
   }]
}
```

   b. 创建一个 JSON 格式的带有角色策略配置的 role-policy.json 文件。例如：

```
{
   "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
```

   c. 使用 trust-policy.json 文件为您的 Amazon Web Services 帐户创建一个角色：

```
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-policy.json
```

d. **使用 role-policy.json 文件嵌入一个内联策略文档：**

```
$ 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

11.2. 使用 CLI 将 AMI 镜像上传到 AWS

您可以使用 RHEL 镜像构建器构建 ami 镜像，并使用 CLI 将它们直接推送到 Amazon AWS Cloud 服务提供商。

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 和 region 的凭

证。**IMAGE_KEY** 值是要上传到 **EC2** 的虚拟机镜像的名称。

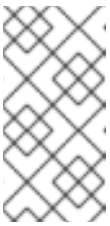
2. 将文件保存为 ***CONFIGURATION-FILE*.toml**，然后关闭文本编辑器。

3. 启动 **compose** 来将其上传到 **AWS**：

```
# composer-cli compose start blueprint-name image-type image-key configuration-file.toml
```

替换：

- 使用您创建的蓝图名称替换 ***blueprint-name***
- 使用 **ami** 镜像类型替换 ***image-type***。
- 使用要上传到 **EC2** 的虚拟机镜像的名称替换 ***image-key***。
- ***configuration-file*.toml**，使用云供应商的配置文件的名称。



注意

您必须有要将自定义镜像发送到的存储桶的正确 **AWS** 身份和访问管理 **(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：

```
# 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，以指示它已被成功上传。

2. 在控制面板中，选择您的镜像并点 Launch。

其它资源

- 导入虚拟机所需的服务角色

11.3. 将镜像推送到 AWS CLOUD AMI

您可以使用 RHEL 镜像构建器创建一个 (.raw) 镜像，并选择 Upload to AWS 复选框，来将您创建的结果镜像直接推送到 Amazon AWS Cloud AMI 服务提供商。

先决条件

- 您必须有访问系统的 root 或 wheel 组用户权限。
- 您已在浏览器中打开了 RHEL web 控制台的 RHEL 镜像构建器界面。
- 您已创建了蓝图。请参阅 在 web 控制台界面中创建蓝图。
- 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. 在 RHEL 镜像构建器仪表盘中，点之前创建的 blueprint name。

2. 选择选项卡 Images。

3. 点 Create Image 创建自定义镜像。

   Create Image 窗口打开。

   a. 在 Type 下拉菜单中选择 Amazon Machine Image Disk (.raw)。

   b. 选中 Upload to AWS 复选框，将您的镜像上传到 AWS 云，然后点 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 字段中输入镜像名称，在 Amazon S3 bucket name 字段中输入 Amazon bucket 名称，并为您要添加自定义镜像的存储桶输入 AWS region 字段。点击 Next。

   e. 查看信息并点 Finish。

   可选，点 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。此流程使用 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 策略。如需更多信息，请参阅 IAM 用户所需的权限。

4. 右上角的弹出窗口告诉您保存的进度。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。

   完成这个过程后，您可以看到 镜像构建完成状态。

5. 在浏览器中，访问 Service→EC2。

   a. 在 AWS 控制台仪表盘菜单中，选择 correct region。镜像必须具有 Available 状态，以指示它已被上传。

   b. 在 AWS 仪表盘中，选择您的镜像并点 Launch。

6. 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。

7. 查看您的实例启动详情。如果需要进行任何更改，您可以编辑任何部分。点 Launch

8. 在启动实例之前，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。

   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 "下载密钥对" 在您的本地系统中保存新密钥对。

9. 然后，您可以点 Launch Instance 来启动您的实例。

   您可以检查实例的状态，它显示为 Initializing。

10. 实例状态变为 running 后，连接按钮将变为可用。

11. 点 连接。此时会出现一个窗口，其中包含有关如何使用 SSH 进行连接的说明。

a. 选择 A standalone SSH client 作为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_&gt;
```

c. 使用其公共 DNS 连接到您的实例：

```
$ ssh -i &lt;_your-instance-name.pem_&gt; ec2-user@&lt;_your-instance-IP-address_&gt;
```

d. 键入 yes 以确认您要继续连接。

因此，您通过 SSH 连接到您的实例。

验证

- 检查在使用 SSH 连接到您的实例的过程中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

11.4. 准备上传 MICROSOFT AZURE VHD 镜像

您可以使用 RHEL 镜像构建器创建一个可上传到 Microsoft Azure 云的 VHD 镜像。

先决条件

- 您必须有 Microsoft Azure 资源组和存储帐户。
- 您已安装了 Python。AZ CLI 工具依赖 python。

流程

1. 导入 Microsoft 存储库密钥：

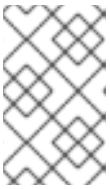
```
# rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

2. 使用以下信息创建一个本地 azure-cli.repo 存储库：将 azure-cli.repo 存储库保存在 /etc/yum.repos.d/ 下：

```
[azure-cli]
name=Azure CLI
baseurl=https://packages.microsoft.com/yumrepos/vscode
enabled=1
gpgcheck=1
gpgkey=https://packages.microsoft.com/keys/microsoft.asc
```

3. 安装 Microsoft Azure CLI：

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



注意

下载的 Microsoft Azure CLI 软件包版本可能会因当前可用的版本而异。

4. 运行 Microsoft Azure CLI：

```
$ az login
```

终端会显示以下信息 Note, we have launched a browser for you to login.For old experience with device code, use "az login --use-device-code.然后，终端会打开浏览器，其中包含可从其登录 https://microsoft.com/devicelogin 的链接。

注意

如果您正在运行一个远程(SSH)会话，则登录页面链接不会在浏览器中打开。在这种情况下，您可以将链接复制到浏览器，并登录以验证您的远程会话。要登录，使用网页浏览器打开页面 https://microsoft.com/devicelogin 并输入要进行身份验证的设备代码。

5. 列出 Microsoft Azure 中存储帐户的密钥：

```
$ az storage account keys list --resource-group <resource_group_name> --account-name
<storage_account_name>
```

将 ***resource-group-name*** 替换为 Microsoft Azure 资源组的名称，将 ***storage-account-name*** 替换为 Microsoft 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
```

记录上一命令输出中 key1 的值。

1. 创建存储容器：

```
$ az storage container create --account-name <storage_account_name>\
--account-key <key1_value> --name <storage_account_name>
```

将 ***storage-account-name*** 替换为存储帐户的名称。

其他资源

- Microsoft Azure CLI。

11.5. 将 VHD 镜像上传到 MICROSOFT AZURE 云

创建了自定义 VHD 镜像后，您可以将其上传到 Microsoft Azure 云。

先决条件

- 必须设置您的系统以上传 Microsoft Azure VHD 镜像。请参阅 准备上传 Microsoft Azure VHD 镜像。
- 您必须有由 RHEL 镜像构建器创建的 Microsoft Azure VHD 镜像。
  - 在 CLI 中，使用 vhd 输出类型。
  - 在 GUI 中，使用 Azure Disk Image (.vhd) 镜像类型。

流程

1. 将镜像推送到 Microsoft Azure 并从中创建一个实例：

```
$ az storage blob upload --account-name <account_name> --container-name
<container_name> --file <_image_-disk.vhd> --name <_image_-disk.vhd> --type page
...
```

2. 上传到 Microsoft Azure Blob 存储后，从其创建一个 Microsoft Azure 镜像：

```
$ az image create --resource-group &lt; _resource_group_name_&gt; --name &lt;_image_-
disk.vhd&gt;  --os-type linux --location eastus --source
https://$ACCOUNT.blob.core.windows.net/&lt;_container_name_&gt; &lt;_image_-
disk.vhd&gt;
 - Running ...
```

验证

1. 使用 Microsoft Azure 门户创建实例，或者使用以下命令：

```
$ az vm create --resource-group &lt;_resource_group_name_&gt; --location eastus --name
&lt;_image_-disk.vhd&gt; --image &lt;_image_-disk.vhd&gt; --admin-username azure-user --
generate-ssh-keys
 - Running ...
```

2.

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。此用户名在上一步中设置了。

其它资源

- 生成 .vhd 格式的镜像失败。

11.6. 将 VHD 镜像推送到 MICROSOFT AZURE 云

您可以使用 RHEL 镜像构建器创建 .vhd 镜像。然后，您可以将输出 .vhd 镜像推送到 Microsoft Azure Cloud 服务提供商的 Blob 存储。

先决条件

- 有对系统的 root 访问权限。
- 您可以访问 RHEL web 控制台的 RHEL 镜像构建器界面。
- 您创建了蓝图。请参阅 在 web 控制台界面中创建一个 RHEL 镜像构建器蓝图。
- 您已创建了 Microsoft 存储帐户。
- 您有一个可写入 Blob Storage。

流程

1. 在 RHEL 镜像构建器仪表盘中，选择要使用的蓝图。
2. 点 Images 选项卡。
3. 点 Create Image 创建自定义的 .vhd 镜像。

   Create image 向导将打开。

a. 从 Type 下拉菜单中选择 Microsoft Azure (.vhd)。

b. 选中 Upload to Azure 复选框，来将镜像上传到 Microsoft Azure Cloud。

c. 输入 Image Size ， 并点 Next。

4. 在 Upload to Azure 页面行，输入以下信息：

a. 在 Authentication 页面上，输入：

i. 您的 存储帐户 名称。您可以在 Microsoft Azure 门户中 中的 Storage account 页面上找到它。

ii. 您的 存储访问密钥 ：您可以在 Access Key 存储页面上找到它。

iii. 点击 Next。

b. 在 Authentication 页面上，输入：

i. 镜像名称。

ii. 存储容器.它是您将镜像上传到的 blob 容器。您可以在 Microsoft Azure 门户 的 Blob service 部分中找到它。

iii. 点击 Next。

5. 在 Review 页面上， 点 Create。RHEL 镜像构建器和上传进程启动。

访问推送到 Microsoft Azure Cloud 的镜像。

6. 访问 Microsoft Azure 门户网站。

7. 在搜索栏中，输入"storage account"，然后从列表中单击 Storage accounts。

8. 在搜索栏中输键入"Images"，然后在 Services 下选择第一个条目。您将被重定向到 镜像仪表盘。

9. 在导航面板上，单击 Containers。

10. 查找您创建的容器。容器中是您使用 RHEL 镜像构建器创建并推送的 .vhd 文件。

验证

1. 验证您能否创建虚拟机镜像并启动它。

   a. 在搜索栏中，输入镜像帐户，并单击列表中的 Images。

   b. 单击 +Create。

   c. 从下拉列表中，选择您之前使用的资源组。

   d. 输入镜像的名称。

   e. 对于 OS type，请选择 Linux。

   f. 对于 VM generation，请选择 Gen 2。

   g. 在 Storage Blob 下，点 Browse ，并点存储账户和容器，直到您到达 VHD 文件。

h. 点页面末尾的 Select。

i. 选择一个帐户类型，例如 Standard SSD。

j. 点 Review + Create，然后点 Create。等待几分钟，以便创建镜像。

2. 要启动虚拟机，请按照以下步骤执行：

a. 点 Go to resource。

b. 从标题的菜单栏中，单击 + Create VM。

c. 输入虚拟机的名称。

d. 完成 Size 和 Administrator account 部分。

e. 点 Review + Create，然后点 Create。您可以查看部署进度。

部署完成后，单击虚拟机名称，以检索使用 SSH 连接的实例的公用 IP 地址。

f. 打开一个终端，创建一个 SSH 连接来连接到虚拟机。

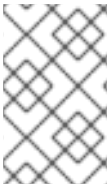
其他资源

- Microsoft Azure Storage 文档.
- 创建 Microsoft 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。

11.7. 上传 VMDK 镜像并在 VSPHERE 中创建 RHEL 虚拟机

使用 RHEL 镜像构建器，您可以创建自定义的 VMware vSphere 系统镜像，可以是 Open virtualization 格式(.ova)，也可以是 Virtual disk (.vmdk)格式。您可以将这些镜像上传到 VMware vSphere 客户端。您可以使用 govc import.vmdk CLI 工具将 .vmdk 或 .ova 镜像上传到 VMware vSphere。您创建的 vmdk 包含安装的 cloud-init 软件包，您可以使用它，通过使用用户数据来提供给用户：



注意

不支持使用 VMware vSphere GUI 上传 vmdk 镜像。

先决条件

- 您已使用用户名和密码自定义创建了一个蓝图。
- 您已使用 RHEL 镜像构建器创建了 .ova 或 .vmdk 格式的 VMware vSphere 镜像，并将其下载到主机系统。
- 您安装并配置了 govc CLI 工具，以便能够使用 import.vmdk 命令。

流程

1. 使用 GOVC 环境变量在用户环境中配置以下值：

```
GOVC_URL
GOVC_DATACENTER
GOVC_FOLDER
GOVC_DATASTORE
GOVC_RESOURCE_POOL
GOVC_NETWORK
```

2. **进入到您下载 VMware vSphere 镜像的目录。**

3. **按照以下步骤在 vSphere 上启动 VMware vSphere 镜像：**

   a. **将 VMware vSphere 镜像导入到 vSphere：**

   ```
   $ govc import.vmdk ./composer-api.vmdk foldername
   ```

   对于 .ova 格式：

   ```
   $ govc import.ova ./composer-api.ova foldername
   ```

   b. **在 vSphere 中创建虚拟机而不开机：**

   ```
   govc vm.create \
   -net.adapter=vmxnet3 \
   -m=4096 -c=2 -g=rhel8_64Guest \
   -firmware=efi -disk="foldername/composer-api.vmdk" \
   -disk.controller=scsi -on=false \
    vmname
   ```

   对于 .ova 格式，将行 -firmware=efi -disk="*foldername*/composer-api.vmdk" \ 替换为 `-firmware=efi -disk="*foldername*/composer-api.ova"

   c. **打开虚拟机：**

   ```
   govc vm.power -on vmname
   ```

   d. **检索虚拟机 IP 地址：**

   ```
   govc vm.ip vmname
   ```

   e. **使用您在蓝图中指定的用户名和密码，使用 SSH 登录到虚拟机：**

   ```
   $ ssh admin@<_ip_address_of_the_vm_>
   ```

注意

如果您使用 govc datastore.upload 命令将 .vmdk 镜像从本地主机复制到目的地，则不支持使用生成的镜像。在 vSphere GUI 中没有使用 import.vmdk 命令的选项，因此 vSphere GUI 不支持直接上传。因此，.vmdk 镜像无法从 vSphere GUI 使用。

11.8. 使用 RHEL 镜像构建器将镜像上传到 GCP

使用 RHEL 镜像构建器，您可以构建 gce 镜像，为用户或 GCP 服务帐户提供凭证，然后将 gce 镜像直接上传到 GCP 环境。

11.8.1. 使用 CLI 将 gce 镜像上传到 GCP

按照以下步骤使用凭证设置配置文件，将 gce 镜像上传到 GCP。

警告

您无法手动将 gce 镜像导入到 GCP，因为镜像不能引导。您必须使用 gcloud 或 RHEL 镜像构建器上传它。

先决条件

- 您有一个有效的 Google 帐户和凭证，以便将镜像上传到 GCP。凭据可以从用户帐户或服务帐户获取。与凭证关联的帐户必须至少分配以下 IAM 角色：
  - roles/storage.admin - 用于创建和删除存储对象
  - roles/compute.storageAdmin - 将虚拟机镜像导入到 Compute Engine。
- 您有一个现有的 GCP 存储桶。

流程

1. 通过使用文本编辑器，创建一个具有以下内容的 gcp-config.toml 配置文件：

```
provider = "gcp"

[settings]
bucket = "GCP_BUCKET"
region = "GCP_STORAGE_REGION"
object = "OBJECT_KEY"
credentials = "GCP_CREDENTIALS"
```

- GCP_BUCKET 指向现有的存储桶。它用于存储正在上传的镜像的中间存储对象。
- GCP_STORAGE_REGION 是常规的 Google 存储区域，是一个双区域或多区域。
- OBJECT_KEY 是中间存储对象的名称。它在上传过程前不能存在，并在上传过程完成后被删除。如果对象名称不以 .tar.gz 结尾，则扩展会自动添加到对象名称中。
- GCP_CREDENTIALS 是一个从 GCP 下载的凭证 JSON 文件的 Base64 编码方案。凭证决定了 GCP 将镜像上传到的项目。

注意

如果您使用不同的机制来使用 GCP 进行身份验证，在 gcp-config.toml 文件中指定 GCP_CREDENTIALS 是可选的。有关其他验证方法，请参阅 使用 GCP 进行验证 。

2. 从 GCP 下载的 JSON 文件中检索 GCP_CREDENTIALS。

```
$ sudo base64 -w 0 cee-gcp-nasa-476a1fa485b7.json
```

3. 使用附加镜像名称和云供应商配置集创建 compose：

```
$ sudo composer-cli compose start BLUEPRINT-NAME gce IMAGE_KEY gcp-config.toml
```

镜像构建、上传和云注册过程最多可能需要十分钟才能完成。

验证

验证

- 验证镜像状态为 **FINISHED**：

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

其他资源

- 身份和访问权限管理
- 创建存储桶

11.8.2. RHEL 镜像构建器如何对不同 GCP 凭证的身份验证顺序进行排序

您可以通过 RHEL 镜像构建器使用几种不同类型的凭证来使用 GCP 进行身份验证。如果 RHEL 镜像构建器配置被设置为使用多个凭证与 GCP 进行身份验证，它会按以下首选顺序使用凭证：

1. 在配置文件中，使用 **composer-cli** 命令指定的凭证。
2. 凭证在 **osbuild-composer worker** 配置中被配置。
3. Google GCP SDK 库中的 应用程序默认凭证，它尝试使用以下选项自动找到一个身份验证的方法：
   a. 如果设置了 *GOOGLE_APPLICATION_CREDENTIALS* 环境变量，应用程序默认凭据会尝试加载并从文件中使用由变量指向的凭证。
   b. 应用默认凭据尝试使用附加到运行代码的资源的服务帐户进行身份验证。例如，Google Compute Engine 虚拟机。

注意

您必须使用 GCP 凭证来决定将镜像上传到的 GCP 项目。因此，除非要将所有镜像上传到同一 GCP 项目，您必须使用 **composer-cli** 命令指定 **gcp-config.toml** 配置文件中的凭证。

11.8.2.1. 使用 composer-cli 命令指定 GCP 凭证

您可以在上传目标配置 gcp-config.toml 文件中指定 GCP 验证凭证。使用 Google 帐户凭证 JSON 文件的 Base64 编码方案来节省时间。

流程

1. 运行以下命令，使用存储在 GOOGLE_APPLICATION_CREDENTIALS 环境变量中的路径获取 Google 帐户凭证文件的编码内容：

```
$ base64 -w 0 "${GOOGLE_APPLICATION_CREDENTIALS}"
```

2. 在上传目标配置 gcp-config.toml 文件中，设置凭证：

```
provider = "gcp"

[settings]
provider = "gcp"

[settings]
...
credentials = "GCP_CREDENTIALS"
```

11.8.2.2. 在 osbuild-composer worker 配置中指定凭证

您可以将 GCP 身份验证凭据配置为全局用于 GCP 用于所有镜像构建。这样，如果您想要将镜像导入到同一 GCP 项目，则所有镜像都可以使用相同的凭证来上传到 GCP。

流程

- 在 /etc/osbuild-worker/osbuild-worker.toml worker 配置中，设置以下凭证值：

```
[gcp]
credentials = "PATH_TO_GCP_ACCOUNT_CREDENTIALS"
```

11.9. 使用 RHEL 镜像构建器 GUI 工具将 VMDK 镜像推送到 VSPHERE

您可以使用 RHEL 镜像构建器 GUI 工具构建 VMware 镜像，并将镜像直接推送到 vSphere 实例，以避免必须下载镜像文件并手动推送它。您创建的 vmdk 包含安装的 cloud-init 软件包，您可以使用它来使用用户数据提供给用户，例如：要使用 RHEL 镜像构建器构建 .vmdk 镜像,并将它们直接推送到 vSphere 实例服务提供商，请按照以下步骤执行：

先决条件

- 您是 root 或 weldr 组的成员。
- 您已在浏览器中打开了 link:https://localhost:9090/RHEL 镜像构建器。
- 您已创建了蓝图。请参阅 在 web 控制台界面中创建一个 RHEL 镜像构建器蓝图。
- 您有 vSphere 帐户。

流程

1. 对于您创建的蓝图，点 Images 选项卡。
2. 点 Create Image 创建自定义镜像。

   此时将打开镜像类型窗口。
3. 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 2 GB。
   d. 点击 Next。
4. 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。**

5. **在 Upload to VMware 窗口中，在 Destination 下输入有关镜像上传目的地的以下详情：**

a. **Image name ：镜像的名称。**

b. **Host ：VMware vSphere 的 URL。**

c. **Cluster ：集群的名称。**

d. **Data center ：数据中心的名称。**

e. **Data store ：数据存储的名称。**

f. **点 Next。**

6. **在 Review 窗口中，查看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。

RHEL 镜像构建器将 RHEL vSphere 镜像的组成添加到队列中，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。要做到这一点：

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. 选择您上传的镜像。
4. 右键点所选镜像。
5. 点 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：

   a. 选择 New Virtual Machine。
   b. 为您的虚拟机选择一个名称和文件夹。
   c. 选择计算资源：为此操作选择一个目标计算资源。
   d. 选择存储: 例如，选择 NFS-Node1
   e. 选择兼容性：镜像应仅限于 BIOS。
   f. 选择一个客户机操作系统：例如，选择 *Linux* 和 *Red Hat Fedora (64 位)*。

g. 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：

h. 准备完成：查看详情并点击 Finish 来创建镜像。

6. 导航至 VMs 选项卡。

a. 从列表中选择您创建的虚拟机。

b. 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。

c. 使用您为蓝图创建的凭证登录。

d. 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

- vSphere 安装和设置的简介.

11.10. 将自定义镜像推送到 OCI

使用 RHEL 镜像构建器，您可以构建自定义镜像，并直接将它们推送到 Oracle Cloud Infrastructure (OCI)实例。然后，您可以从 OCI 仪表盘中启动镜像实例。

先决条件

- 您有访问系统的 root 或 weldr 组用户权限。
- 您有一个 Oracle Cloud 帐户。

- 您必须为管理员授予 OCI 策略中的安全访问权限。
- 您已在您选择的 OCI_REGION 中创建了 OCI Bucket。

流程

1. 在浏览器中打开 web 控制台的 RHEL 镜像构建器界面。
2. 点击 Create blueprint。Create blueprint 向导将打开。
3. 在 Details 页面中，输入蓝图的名称，以及可选的描述。点击 Next。
4. 在 Packages 页面中，选择要在镜像中包含的组件和软件包。点击 Next。
5. 在 Customizations 页面中，配置您要用于蓝图的自定义。点击 Next。
6. 在 Review 页面上，点 Create。
7. 要创建镜像，请单击 Create Image。Create image 向导将打开。
8. 在 Image 输出页面中完成以下步骤：
   a. 在 "Select a blueprint" 下拉菜单中选择您想要的蓝图。
   b. 从 "Image output type" 下拉菜单中选择 Oracle Cloud Infrastructure (.qcow2)。
   c. 选中 "Upload OCI 复选框，来将您的镜像上传到 OCI。
   d. 输入 "镜像大小 "。点击 Next。

9. 在 Upload to OCI - Authentication 页面中，输入以下强制详情：

   a. 用户 OCID：您可以在显示用户详情的页面的控制台中找到它。

   b. 私钥

10. 在 Upload to OCI - Destination 页面中，输入以下强制详情，并点 Next。

    a. 镜像名称：要上传的镜像的名称。

    b. OCI 存储桶

    c. bucket 命名空间

    d. bucket 区域

    e. bucket 划分

    f. bucket 租户

11. 查看向导中的详情并点 Finish。

RHEL 镜像构建器将 RHEL .qcow2 镜像的组成添加到队列中。

验证

1. 访问 OCI 仪表板 → Custom Images。

2. 选择您为镜像指定的 Compartment，并在 Import image 表中找到镜像。

3. 点镜像名称并验证镜像信息。

其他资源

- 在 OCI 中管理自定义镜像。
- 在 OCI 中管理存储桶。
- 生成 SSH 密钥。

11.11. 将 QCOW2 镜像上传到 OPENSTACK

使用 RHEL 镜像构建器工具，您可以创建适合上传到 OpenStack 云部署的自定义 .qcow2 镜像，并在那里启动实例。RHEL 镜像构建器以 QCOW2 格式创建镜像，但针对 OpenStack 有进一步的更改。

警告

不要将使用 RHEL 镜像构建器创建的通用 QCOW2 镜像类型输出格式与 OpenStack 镜像类型混淆，后者也是 QCOW2 格式，但包含特定于 OpenStack 的进一步更改。

先决条件

- 您已创建了一个蓝图。

流程

1. 启动 QCOW2 镜像的 compose。

```
# composer-cli compose start blueprint_name openstack
```

2. 检查构建的状态。

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

镜像构建完成后，您可以下载镜像。

3. **下载 QCOW2 镜像：**

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

4. **访问 OpenStack 仪表板，再单击 +Create Image。**

5. **在左侧菜单中，选择 Admin 选项卡。**

6. **从系统面板中，点镜像。**

   Create An Image 向导将打开。

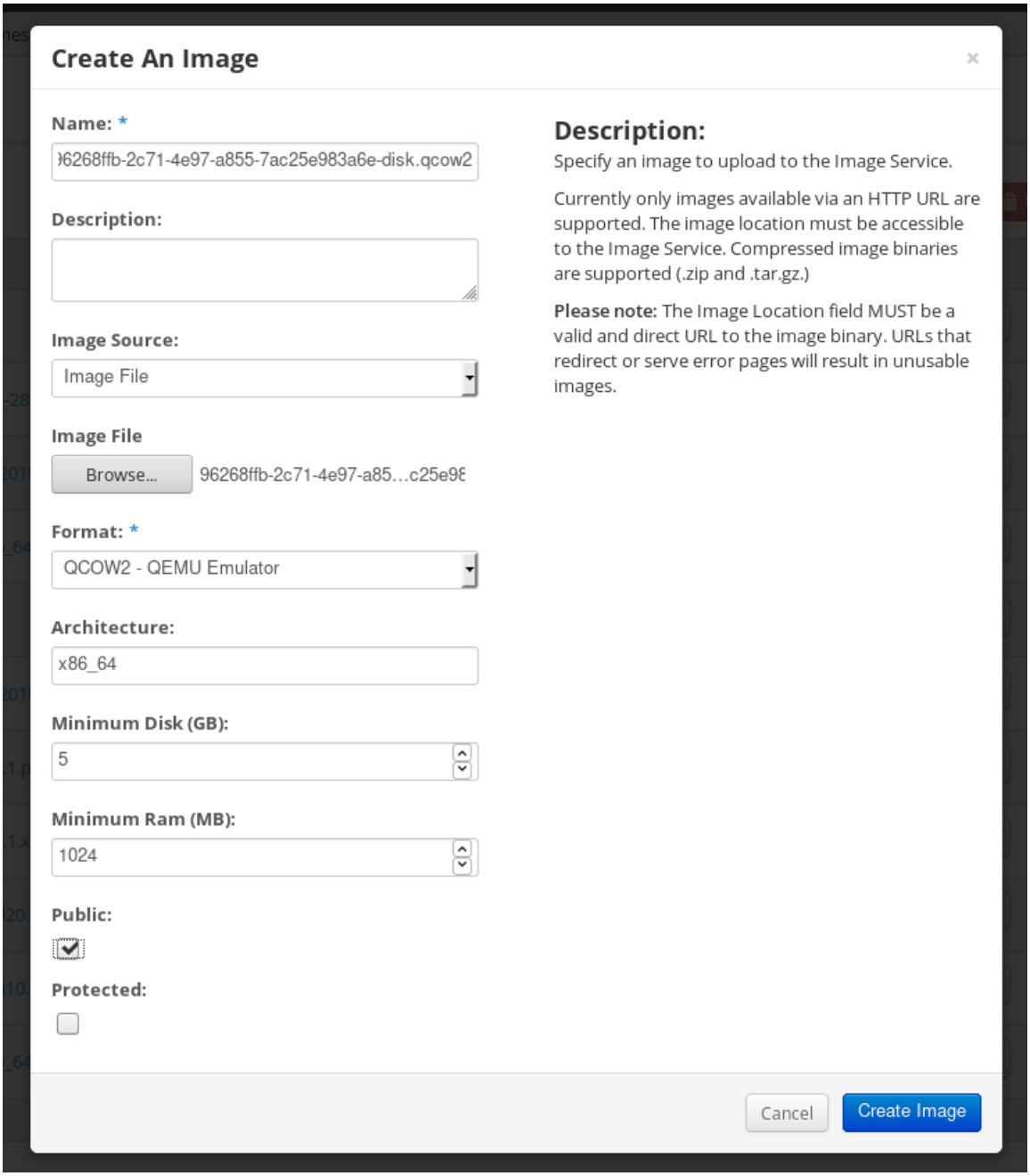
7. **在 Create An Image 向导中：**

   a. **输入镜像的名称**

   b. **点 Browse，上传 QCOW2 镜像。**

   c. **从 格式 下拉列表中，选择 QCOW2 - QEMU Emulator。**

   d. **点 Create Image。**



8. **在左侧菜单中，选择 Project 选项卡。**

   a. **从 Compute 菜单中，选择 Instances。**

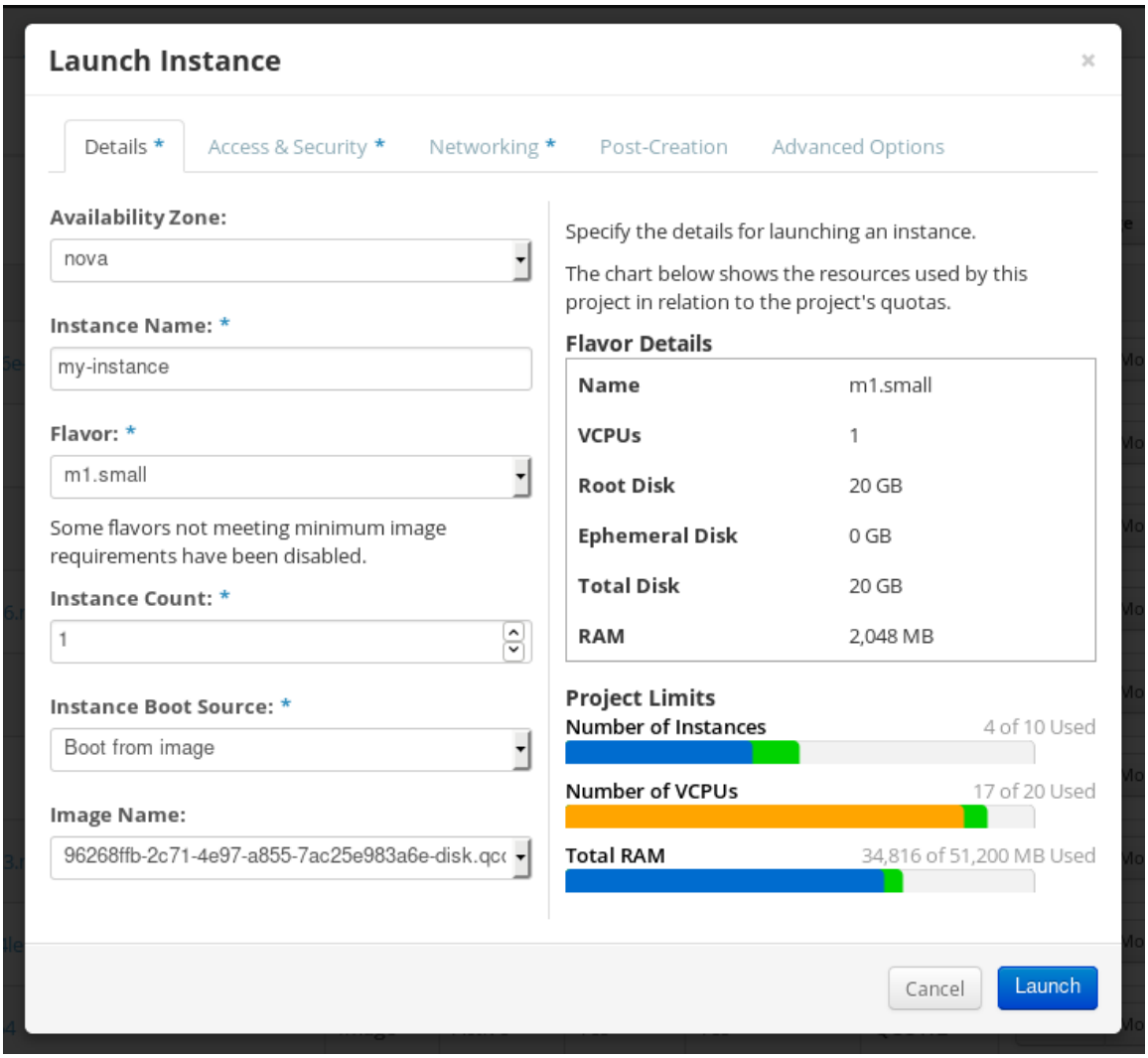
   b. **单击 Launch Instance 按钮。**

      此时会打开 Launch Instance 向导。

c. 在 Details 页面中，输入实例的名称。点击 Next。

d. 在 Source 页面中，选择您上传的镜像的名称。点击 Next。

e. 在 Flavor 页面中，选择最适合您的需要的机器资源。点 Launch。



9. 您可以使用任何机制（CLI 或 OpenStack Web UI）来从镜像运行镜像实例。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。

11.12. 准备将自定义 RHEL 镜像上传到 ALIBABA CLOUD

要将自定义 RHEL 镜像部署到 Alibaba Cloud 中，您首先需要验证自定义镜像。镜像需要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来满足某些要求。

注意

RHEL 镜像构建器生成符合 Alibaba 要求的镜像。但是，红帽建议使用 Alibaba image_check 工具来验证镜像的格式合规性。

先决条件

- 您必须已使用 RHEL 镜像构建器创建了一个 Alibaba 镜像。

流程

1. 使用 Alibaba image_check 工具连接到包含您要检查的镜像的系统。

2. 下载 image_check 工具：

```
$ curl -O https://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 工具将此报告保存在与运行镜像合规工具同样的文件夹中。

故障排除

如果任何检测项失败，请按照终端中的说明进行更正。

其他资源

- 镜像合规工具.

11.13. 将自定义 RHEL 镜像上传到 ALIBABA

您可以使用 RHEL 镜像构建器将您创建的自定义 AMI 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。请参阅准备将镜像上传到 Alibaba。
- 您已使用 RHEL 镜像构建器创建了一个 ami 镜像。
- 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. 在左侧的 Bucket 菜单中，选择要将镜像上传到的存储桶。
3. 在右菜单中点 Files 标签页。
4. 点 Upload。一个对话框窗口会在右侧打开。配置以下内容：
   - Upload To：选择将文件上传到 Current 目录或一个指定的目录。

- 文件 ACL：选择上传的文件的权限类型。

5. 点 Upload。

6. 选择您要上传到 OSS 控制台的镜像。

7. 点击 Open。

其他资源

- 上传一个对象。
- 从自定义镜像创建一个实例。
- 导入镜像。

11.14. 将镜像导入到 ALIBABA CLOUD

要将使用 RHEL 镜像构建器创建的自定义 Alibaba RHEL 镜像导入到 Elastic Compute Service (ECS)，请按照以下步骤操作：

先决条件

- 设置您的系统以上传 Alibaba 镜像。请参阅准备将镜像上传到 Alibaba。
- 您已使用 RHEL 镜像构建器创建了一个 ami 镜像。
- 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。
- 您已将镜像上传到对象存储服务(OSS)。请参阅将镜像上传到 Alibaba。

流程

1. 登录到 ECS 控制台。
   i. 在左侧菜单中，点 Images。
   ii. 在右上角，点 Import Image。此时会打开一个对话框窗口。
   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. Platform: 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. 镜像描述

i. 添加数据磁盘的镜像

地址可以在 OSS 管理控制台中确定。在左侧菜单中选择所需的存储桶之后：

2. 选择 Files 部分。

3. 点相应镜像右侧的 Details 链接。

窗口会出现在屏幕右侧，显示镜像详情。OSS 对象地址位于 URL 框中。

4. 点击 OK。

注意

导入过程的时间可能因镜像大小而异。

自定义镜像导入到 ECS 控制台。

其他资源

- 导入镜像的备注。
- 从自定义镜像创建一个实例。
- 上传一个对象。

11.15. 使用 ALIBABA CLOUD 创建自定义 RHEL 镜像的一个实例

您可以使用 Alibaba ECS 控制台创建自定义 RHEL 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。请参阅 将镜像导入到 Alibaba。

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. 完成所有必需的信息。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好从 Alibaba ECS 控制台进行部署。

其他资源

- 使用自定义镜像创建一个实例。
- 使用向导创建一个实例。