Red Hat Enterprise Linux 9

RHEL システムイメージのカスタマイズ

Red Hat Enterprise Linux 9 で RHEL Image Builder を使用してカスタマイズしたシステムイメージを作成する

Last Updated: 2024-12-25

Red Hat Enterprise Linux 9 RHEL システムイメージのカスタマイズ

Red Hat Enterprise Linux 9 で RHEL Image Builder を使用してカスタマイズしたシステムイメージを作成する

法律上の通知

概要

RHEL Image Builder は、デプロイメント可能なカスタムシステムイメージ (インストールディスク、仮想マシン、クラウドベンダー固有のイメージなど) を作成するツールです。RHEL Image Builder を使用すると、出力タイプごとに必要な特定の設定が不要になるため、手動の手順と比較してこれらのイメージをより迅速に作成できます。このドキュメントでは、RHEL Image Builder を設定してイメージを作成する方法を説明します。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 RHEL IMAGE BUILDER の説明

システムをデプロイするには、システムイメージを作成します。RHEL システムイメージを作成するには、RHEL Image Builder ツールを使用します。RHEL Image Builder を使用することで、RHEL のカスタマイズされたシステムイメージを作成できます。これには、クラウドプラットフォームへのデプロイメント用に準備されたシステムイメージが含まれます。RHEL Image Builder は、各出力タイプのセットアップの詳細を自動的に処理するため、手動でイメージを作成する方法よりも使いやすく、作業が高速です。RHEL Image Builder の機能には、**composer-cli** ツールのコマンドラインインターフェイス、または RHEL Web コンソールのグラフィカルユーザーインターフェイスを使用してアクセスできます。

1.1. RHEL IMAGE BUILDER の用語

RHEL Image Builder は次の概念を使用します。

ブループリント

ブループリントは、カスタマイズされたシステムイメージの説明です。システムの一部となるパッケージとカスタマイズが一覧表示されます。ブループリントをカスタマイズして編集し、特定のバージョンとして保存できます。ブループリントからシステムイメージを作成すると、イメージは RHEL Image Builder インターフェイスでブループリントに関連付けられます。
ブループリントは TOML 形式で作成します。

Compose

コンポーズは、特定のブループリントの特定のバージョンに基づいた、システムイメージの個別のビルドです。用語としての Compose は、システムイメージと、その作成、入力、メタデータ、およびそのプロセス自体のログを指します。

カスタマイズ

カスタマイズは、パッケージではないイメージの仕様です。これには、ユーザー、グループ、および SSH 鍵が含まれます。

1.2. RHEL IMAGE BUILDER の出力形式

RHEL Image Builder は、次の表に示す複数の出力形式でイメージを作成できます。

表1.1 RHEL Image Builder の出力形式

説明	CLI 名	ファイル拡張子
QEMU イメージ	**qcow2**	**.qcow2**
ディスクアーカイブ	**tar**	**.tar**
Amazon Web Services	**raw**	**.raw**
Microsoft Azure	**vhd**	**.vhd**
Google Cloud Platform	**gce**	**.tar.gz**
VMware vSphere	**vmdk**	**.vmdk**
VMware vSphere	**ova**	**.ova**

説明	CLI 名	ファイル拡張子
Openstack	**openstack**	**.qcow2**
RHEL for Edge Commit	**edge-commit**	**.tar**
RHEL for Edge Container	**edge-container**	**.tar**
RHEL for Edge Installer	**edge-installer**	**.iso**
RHEL for Edge Raw イメージ	**edge-raw-image**	**.raw.xz**
RHEL for Edge Simplified Installer	**edge-simplified-installer**	**.iso**
RHEL for Edge AMI	**edge-ami**	**.ami**
RHEL for Edge VMDK	**edge-vsphere**	**.vmdk**
RHEL インストーラー	**image-installer**	**.iso**
Oracle Cloud Infrastructure	**.oci**	**.qcow2**

サポートされているタイプを確認するには、次のコマンドを実行します。

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

1.3. イメージビルドでサポートされているアーキテクチャー

RHEL Image Builder は、次のアーキテクチャーのイメージのビルドをサポートしています。

- AMD および Intel 64 ビット (**x86_64**)
- ARM64 (**aarch64**)
- IBM Z (**s390x**)
- IBM POWER システム

ただし、RHEL Image Builder はマルチアーキテクチャービルドをサポートしていません。実行しているのと同じシステムアーキテクチャーのイメージのみをビルドします。たとえば、RHEL Image Builder が **x86_64** システムで実行している場合は、**x86_64** アーキテクチャーのイメージのみをビルドできます。

1.4. 関連情報

- RHEL image builder additional documentation index

第2章 RHEL IMAGE BUILDER のインストール

RHEL Image Builder を使用する前に、RHEL Image Builder をインストールする必要があります。

2.1. RHEL IMAGE BUILDER のシステム要件

RHEL Image Builder を実行するホストは、次の要件を満たしている必要があります。

表2.1 RHEL Image Builder のシステム要件

パラメーター	最低要求値
System type	専用のホストまたは仮想マシン。RHEL Image Builder は、Red Hat Universal Base Images (UBI) などのコンテナーではサポートされていないことに注意してください。
プロセッサー	2 コア
メモリー	4 GiB
ディスク領域	`/var/cache/` ファイルシステムに 20 GiB の空き領域
アクセス権限	root
Network	Red Hat コンテンツ配信ネットワーク (CDN) へのインターネット接続

注記

インターネットに接続できない場合は、分離されたネットワークで RHEL Image Builder を使用してください。そのためには、Red Hat コンテンツ配信ネットワーク (CDN) に接続しないように、ローカルリポジトリーを参照するようにデフォルトのリポジトリーをオーバーライドする必要があります。コンテンツが内部でミラーリングされていることを確認するか、Red Hat Satellite を使用してください。

関連情報

- RHEL Image Builder リポジトリーの設定
- Provisioning to Satellite using a Red Hat image builder image

2.2. RHEL IMAGE BUILDER のインストール

RHEL Image Builder をインストールして、**osbuild-composer** パッケージのすべての機能にアクセスできるようにします。

前提条件

- RHEL Image Builder をインストールする RHEL ホストにログインしている。
- ホストが Red Hat Subscription Manager (RHSM) または Red Hat Satellite にサブスクライブしている。

- RHEL Image Builder パッケージをインストールできるように、**BaseOS** リポジトリーおよび **AppStream** リポジトリーを有効化している。

手順

1. RHEL Image Builder とその他の必要なパッケージをインストールします。

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

   - **osbuild-composer** - カスタマイズした RHEL オペレーティングシステムイメージをビルドするサービス。
   - **composer-cli** - このパッケージにより、CLI インターフェイスへのアクセスが可能になります。
   - **cockpit-composer** - このパッケージにより、Web UI インターフェイスへのアクセスが可能になります。Web コンソールは、**cockpit-composer** パッケージの依存関係としてインストールされます。

2. RHEL Image Builder ソケットを有効にして起動します。

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

3. Web コンソールで RHEL Image Builder を使用する場合は、それを有効にして起動します。

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

   osbuild-composer サービスと **cockpit** サービスは、最初のアクセス時に自動的に起動します。

4. ログアウトおよびログインしなくても **composer-cli** コマンドのオートコンプリート機能がすぐに動作するように、シェル設定スクリプトをロードします。

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

5. RHEL ホストで実行中の **osbuild-composer** サービスを再起動します。

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

検証

- **composer-cli** を実行して、インストールが動作することを確認します。

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

トラブルシューティング

システムジャーナルを使用して、RHEL Image Builder のアクティビティーを追跡できます。さらに、ファイル内のログメッセージを見つけることができます。

- トレースバックのジャーナル出力を見つけるには、次のコマンドを実行します。

```
$ journalctl | grep osbuild
```

- リモートワーカーとローカルワーカーの両方を表示するには:

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

- 実行中のサービスを表示するには:

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

第3章 RHEL IMAGE BUILDER リポジトリーの設定

RHEL Image Builder を使用するには、確実にリポジトリーを設定する必要があります。RHEL Image Builder では、以下のタイプのリポジトリーを使用できます。

公式リポジトリーのオーバーライド

Red Hat Content Delivery Network (CDN) 公式リポジトリー以外の場所 (ネットワーク内のカスタムミラーなど) からベースシステム RPM をダウンロードする場合は、これらを使用します。公式リポジトリーのオーバーライドを使用するとデフォルトのリポジトリーが無効になるため、カスタムミラーには必要なパッケージがすべて含まれている必要があります。

カスタムサードパーティーリポジトリー

これらを使用して、公式の RHEL リポジトリーで利用できないパッケージを含めます。

3.1. RHEL IMAGE BUILDER へのカスタムサードパーティーリポジトリーの追加

カスタムのサードパーティーソースをリポジトリーに追加し、**composer-cli** 使用してこれらのリポジトリーを管理できます。

前提条件

- カスタムサードパーティーリポジトリーの URL を持っている。

手順

1. **/root/repo.toml** などのリポジトリーソースファイルを作成します。以下に例を示します。

```
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 フィールドは、有効な値 **yum-baseurl**、**yum-mirrorlist**、および **yum-metalink** を受け入れます。

2. ファイルを TOML 形式で保存します。
3. 新しいサードパーティーソースを RHEL Image Builder に追加します。

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

検証

1. 新しいソースが正常に追加されたかどうかを確認します。

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

2. 新しいソースコンテンツを確認します。

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

3.2. 特定のディストリビューションを使用した RHEL IMAGE BUILDER へのサードパーティーリポジトリーの追加

オプションのフィールド **distro** を使用して、カスタムサードパーティーソースファイル内のディストリビューションのリストを指定できます。リポジトリーファイルは、イメージのビルド中に依存関係を解決する際にディストリビューション文字列リストを使用します。

rhel-9 を指定するリクエストはすべて、このソースを使用します。たとえば、パッケージをリストして **rhel-9** を指定すると、このソースが含まれます。ただし、ホストディストリビューションのパッケージのリストには、このソースは含まれません。

前提条件

- カスタムサードパーティーリポジトリーの URL を持っている。
- 指定するディストリビューションのリストがある。

手順

1. **/root/repo.toml** などのリポジトリーソースファイルを作成します。たとえば、ディストリビューションを指定するには、次のようにします。

```
check_gpg = true
check_ssl = true
distros = ["rhel-9"]
id = "rh9-local"
name = "packages for RHEL"
system = false
type = "yum-baseurl"
url = "https://local/repos/rhel9/projectrepo/"
```

2. ファイルを TOML 形式で保存します。
3. 新しいサードパーティーソースを RHEL Image Builder に追加します。

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

検証

1. 新しいソースが正常に追加されたかどうかを確認します。

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

2. 新しいソースコンテンツを確認します。

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

3.3. GPG を使用したリポジトリーのメタデータの確認

破損したパッケージを検出して回避するために、DNF パッケージマネージャーを使用して RPM パッケージの GNU Privacy Guard (GPG) 署名を確認でき、リポジトリーのメタデータが GPG キーで署名されているかどうかも確認できます。

gpgkeys フィールドにキー URL を設定することで、 **https** 経由でチェックを行う **gpgkey** を入力できます。あるいは、セキュリティーを向上させるために、キー全体を **gpgkeys** フィールドに埋め込んで、キーを URL から取得する代わりに直接インポートすることもできます。

前提条件

- リポジトリーとして使用するディレクトリーが存在し、パッケージが含まれている。

手順

1. リポジトリーを作成するフォルダーにアクセスします。

```
$ cd repo/
```

2. **createrepo_c** を実行して、RPM パッケージからリポジトリーを作成します。

```
$ createrepo_c .
```

3. リポデータがあるディレクトリーにアクセスします。

```
$ cd repodata/
```

4. **repomd.xml** ファイルに署名します。

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

5. リポジトリーで GPG 署名チェックを有効にするには、以下を行います。

   a. リポジトリーソースで **check_repogpg = true** を設定します。

   b. チェックを行う **gpgkey** を入力します。キーが **https** 経由で利用できる場合は、**gpgkeys** フィールドにキーのキー URL を設定します。URL キーは必要なだけ追加できます。
   以下に例を示します。

```
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"]
```

   代わりに、たとえば次のように GPG キーを **gpgkeys** フィールドに直接追加します。

```
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-----''']
```

- テストで署名が見つからない場合、GPG ツールは次のようなエラーを表示します。

```
$ 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)
```

- 署名が無効な場合、GPG ツールは次のようなエラーを表示します。

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

検証

- リポジトリーの署名を手動でテストします。

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

3.4. RHEL IMAGE BUILDER 公式リポジトリーのオーバーライド

RHEL Image Builder の **osbuild-composer** バックエンドは、**/etc/yum.repos.d/** にあるシステムのリポジトリーを継承しません。代わりに、**/usr/share/osbuild-composer/repositories** ディレクトリーに定義された独自の公式リポジトリーのセットがあります。これには、追加のソフトウェアをインストールしたり、すでにインストールされているプログラムを新しいバージョンに更新したりするためのベースシステム RPM が含まれている Red Hat 公式リポジトリーが含まれます。公式リポジトリーをオーバーライドするには、**/etc/osbuild-composer/repositories** でオーバーライドを定義する必要があります。このディレクトリーはユーザー定義のオーバーライド用であり、ここにあるファイルは **/usr/share/osbuild-composer/repositories/** ディレクトリー内のファイルよりも優先されます。

設定ファイルは **/etc/yum.repos.d/** 内のファイルから知られている通常の DNF リポジトリー形式ではありません。それらは JSON ファイルです。

3.5. システムリポジトリーのオーバーライド

/etc/osbuild-composer/repositories ディレクトリーで、RHEL Image Builder 用のリポジトリーオーバーライドを独自に設定できます。

前提条件

- ホストシステムからアクセスできるカスタムリポジトリーがある。

手順

1. リポジトリーのオーバーライドを保存する **/etc/osbuild-composer/repositories/** ディレクトリーを作成します。

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

2. RHEL バージョンに対応する名前を使用して、JSON ファイルを作成します。または、配布用のファイルを **/usr/share/osbuild-composer/** からコピーして、その内容を変更することもできます。
   RHEL 9.3 の場合は、**/etc/osbuild-composer/repositories/rhel-93.json** を使用します。

3. 次の構造を JSON ファイルに追加します。次の属性から 1 つだけ文字列形式で指定します。

   - **baseurl** - リポジトリーのベース URL。
   - **metalink** - 有効なミラーリポジトリーのリストを含む metallink ファイルの URL。
   - **mirrorlist** - 有効なミラーリポジトリーのリストを含む mirrorlist ファイルの URL。 **gpgkey** や **metadata_expire** などの残りのフィールドはオプションです。
     以下に例を示します。

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

     あるいは、**rhel-version.json** を RHEL のバージョン (例: rhel-9.json) に置き換えて、ディストリビューション用の JSON ファイルをコピーすることもできます。

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

4. オプション: JSON ファイルを確認します。

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

5. **rhel-9.json** ファイル内の **baseurl** パスを編集して保存します。以下に例を示します。

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

6. **osbuild-composer.service** を再起動します。

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

検証

- リポジトリーが正しい URL を指しているか確認します。

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

  リポジトリーは **/etc/yum.repos.d/redhat.repo** ファイルからコピーされた正しい URL を指していることが分かります。

関連情報

- リポジトリーで利用可能な最新の RPM バージョンは、**osbuild-composer** では表示されません。(Red Hat ナレッジベース)

3.6. サブスクリプションが必要なシステムリポジトリーのオーバーライド

/etc/yum.repos.d/redhat.repo ファイルで定義されているシステムサブスクリプションを使用するように **osbuild-composer** サービスをセットアップできます。**osbuild-composer** でシステムサブスクリプションを使用するには、次の詳細を含むリポジトリーオーバーライドを定義します。

- **/etc/yum.repos.d/redhat.repo** で定義されているリポジトリーと同じ **baseurl**。
- **"rhsm": true** の値は、JSON オブジェクトで定義されます。

注記

osbuild-composer は、**/etc/yum.repos.d/** で定義されたリポジトリーを自動的に使用するわけではありません。リポジトリーは、システムリポジトリーオーバーライドとして、または追加の **source** として、**composer-cli** を使用して手動で指定する必要があります。"BaseOS" および "AppStream" リポジトリーは通常、システムリポジトリーオーバーライドを使用しますが、他のすべてのリポジトリーは **composer-cli** ソースを使用します。

前提条件

- システムに **/etc/yum.repos.d/redhat.repo** で定義されたサブスクリプションがある。
- リポジトリーオーバーライドを作成している。システムリポジトリーのオーバーライド を参照してください。

手順

1. **/etc/yum.repos.d/redhat.repo** ファイルから **baseurl** を取得します。

```
# cat /etc/yum.repos.d/redhat.repo
[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. 同じ **baseurl** を使用するようにリポジトリーオーバーライドを設定し、**rhsm** を true に設定します。

```
{
  "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
        }
    ]
}
```

3. **osbuild-composer.service** を再起動します。

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

関連情報

- RHEL image builder uses CDN repositories when host is registered to Satellite 6 (Red Hat ナレッジベース)

3.7. SATELLITE CV をコンテンツソースとして設定および使用する

RHEL Image Builder を使用してイメージをビルドするためのリポジトリーとして、Satellite のコンテンツビュー (CV) を使用できます。これを行うには、Satellite に登録されているホストで、Red Hat Content Delivery Network (CDN) の公式リポジトリーではなく、Satellite リポジトリーから取得できるようにリポジトリー参照を手動で設定します。

前提条件

- Satellite 6 に登録されたホストで RHEL Image Builder を使用している。

手順

1. 現在設定されているリポジトリーからリポジトリー URL を見つけます。

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

次の出力は例です。

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

2. ハードコードされたリポジトリーを Satellite Server に変更します。

   a. **0755** 権限を持つリポジトリーディレクトリーを作成します。

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

   b. **/usr/share/osbuild-composer/repositories/*.json** の内容を、作成したディレクトリーにコピーします。

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

   c. **/content/dist/*** 行を通じて Satellite URL とファイルの内容を更新します。

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

d. 設定が正しく置き換えられたことを確認します。

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

3. サービスを再起動します。

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

4. Red Hat Image Builder 設定で必要なシステムリポジトリーをオーバーライドし、Satellite リポジトリーの URL をベース URL として使用します。システムリポジトリーのオーバーライド を参照してください。

関連情報

- Composer RHEL image builder fails when multiple custom repositories are defined on the Satellite (Red Hat ナレッジベース)

3.8. RHEL IMAGE BUILDER でイメージをビルドするためのリポジトリーとして SATELLITE CV を使用する

カスタムイメージをビルドするためのリポジトリーとして Satellite のコンテンツビュー (CV) を使用するように RHEL Image Builder を設定します。

前提条件

- RHEL Web コンソールに Satellite が統合されている。Satellite での RHEL Web コンソールの有効化 を参照してください。

手順

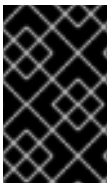
1. Satellite Web UI で、**Content > Products** に移動し、**Product** を選択して、使用するリポジトリーをクリックします。
2. Published フィールドで、保護された URL (HTTPS) を検索してコピーします。
3. コピーした URL を Red Hat Image Builder リポジトリーのベース URL として使用します。RHEL Image Builder へのカスタムサードパーティーリポジトリーの追加 を参照してください。

次のステップ

- イメージをビルドします。Web コンソールインターフェイスで RHEL Image Builder を使用してシステムイメージを作成する を参照してください。

第4章 RHEL IMAGE BUILDER CLI を使用したシステムイメージの作成

RHEL Image Builder は、カスタムのシステムイメージを作成するツールです。RHEL Image Builder を制御し、カスタムシステムイメージを作成するには、コマンドラインインターフェイス (CLI) または Web コンソールインターフェイスを使用できます。



重要

RHEL Image Builder ツールは、RHEL 8 システム上での RHEL 9 イメージのビルドをサポートしていません。イメージは、ホスト配布で使用する以前のバージョンのイメージからのみビルドできます。

4.1. RHEL IMAGE BUILDER コマンドラインインターフェイスの紹介

RHEL Image Builder コマンドラインインターフェイス (CLI) を使用してブループリントを作成するには、適切なオプションとサブコマンドを指定して **composer-cli** コマンドを実行します。

コマンドラインインターフェイスのワークフローの概要は次のようになります。

1. ブループリントを作成するか、既存のブループリント定義をプレーンテキストファイルにエクスポート (**保存**) します。
2. テキストエディターでこのファイルを編集します。
3. ブループリントテキストファイルを Image Builder にインポートします。
4. Compose を実行して、ブループリントからイメージをビルドします。
5. イメージファイルをエクスポートしてダウンロードします。

composer-cli コマンドには、ブループリントを作成する基本的なサブコマンドとは別に、設定したブループリントと Compose の状態を調べるための多くのサブコマンドが用意されています。

4.2. RHEL IMAGE BUILDER を非 ROOT ユーザーとして使用する

非 root として **composer-cli** コマンドを実行するには、ユーザーが **weldr** グループに属している必要があります。

前提条件

- ユーザーを作成している。

手順

- ユーザーを **weldr** または **root** グループに追加するには、次のコマンドを実行します:

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

4.3. コマンドラインインターフェイスを使用したブループリントの作成

RHEL Image Builder のコマンドラインインターフェイス (CLI) を使用して、新しいブループリントを作成できます。ブループリントには、最終的なイメージと、パッケージやカーネルのカスタマイズなどのそのカスタマイズが記述されています。

前提条件

- root ユーザーまたは **weldr** グループのメンバーであるユーザーとしてログインしている。

手順

1. 次の内容のプレーンテキストファイルを作成します。

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

BLUEPRINT-NAME および LONG FORM DESCRIPTION TEXT は、ブループリントの名前および説明に置き換えます。

0.0.1 は、セマンティックバージョニングスキームに従ってバージョン番号に置き換えます。

2. ブループリントに含まれるすべてのパッケージに、次の行をファイルに追加します。

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

package-name は、**httpd**、**gdb-doc**、**coreutils** などのパッケージ名に置き換えます。

必要に応じて、package-version を使用するバージョンに置き換えます。このフィールドは、**dnf** バージョンの指定に対応します。

- 特定のバージョンについては、8.7.0 などの正確なバージョン番号を使用してください。
- 利用可能な最新バージョンについては、アスタリスク * を使用してください。
- 最新のマイナーバージョンを指定する場合は、8.* などの形式を使用してください。

3. ニーズに合わせてブループリントをカスタマイズします。たとえば、Simultaneous Multi Threading (SMT) を無効にするには、ブループリントファイルに次の行を追加します。

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

その他に利用できるカスタマイズについては、サポートされているイメージのカスタマイズ を参照してください。

[] と **[[]]** は TOML で表現される異なるデータ構造であることに注意してください。

- **[customizations.kernel]** ヘッダーは、キーと各値のペアの集合によって定義される単一のテーブルを表します (例: **append = "nosmt=force"**)。
- **[[packages]]** ヘッダーはテーブルの配列を表します。最初のインスタンスで、配列とその最初のテーブル要素を定義します (例: **name = "package-name"**、**version = "package-**

version")。後続の各インスタンスで、指定の順序で、その配列内に新しいテーブル要素を作成して定義します。

4. たとえば、ファイルを **BLUEPRINT-NAME**.toml として保存し、テキストエディターを閉じます。

5. ブループリントをプッシュします。

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

BLUEPRINT-NAME は、前の手順で使用した値に置き換えます。

注記

composer-cli を非 root として使用してイメージを作成するには、ユーザーを **weldr** または **root** グループに追加します。

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

検証

- ブループリントがプッシュされ存在していることを確認するには、既存のブループリントを一覧表示します。

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

- 追加したばかりのブループリント設定を表示します。

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

- ブループリントに記載されているコンポーネントおよびバージョンと、その依存関係が有効かどうかを確認します。

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

RHEL Image Builder がカスタムリポジトリーのパッケージの依存関係を解決できない場合は、**osbuild-composer** キャッシュを削除します。3

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

関連情報

- osbuild-composer is unable to depsolve a package from my custom repository Red Hat ナレッジベース
- Composing a customized RHEL system image with proxy server (Red Hat ナレッジベース)

4.4. コマンドラインインターフェイスを使用したブループリントの編集

コマンドライン (CLI) インターフェイスで既存のブループリントを編集して、たとえば、新しいパッケージを追加したり、新しいグループを定義したり、カスタマイズしたイメージを作成したりできます。そのためには、以下の手順に従います。

前提条件

- ブループリントを作成している。

手順

1. 既存のブループリントをリストします。

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

2. ブループリントをローカルテキストファイルに保存します。

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

3. **BLUEPRINT-NAME** .toml ファイルをテキストエディターで編集し、変更を加えます。
4. 編集を完了する前に、ファイルが有効なブループリントであることを確認します。
   a. ブループリントに次の行が存在する場合は、それを削除します。

```
packages = []
```

   b. たとえば、バージョン番号を 0.0.1 から 0.1.0 に増やします。RHEL Image Builder のブループリントバージョンでは、セマンティックバージョニングスキームを使用する必要があります。また、バージョンを変更しない場合、**パッチ**バージョンコンポーネントが自動的に増加することにも注意してください。
5. ファイルを保存し、テキストエディターを閉じます。
6. ブループリントを RHEL Image Builder にプッシュして戻します。

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

注記

ブループリントを RHEL Image Builder にインポートして戻すには、**.toml** 拡張子を含むファイル名を指定しますが、他のコマンドではブループリント名のみを使用します。

検証

1. RHEL Image Builder にアップロードした内容が編集内容と一致することを確認するには、ブループリントの内容をリストします。

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

2. ブループリントに記載されているコンポーネントおよびバージョンと、その依存関係が有効かどうかを確認します。

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

関連情報

- サポートされているイメージのカスタマイズ

4.5. RHEL IMAGE BUILDER コマンドラインインターフェイスでのシステムイメージの作成

RHEL Image Builder のコマンドラインインターフェイスを使用して、カスタマイズした RHEL イメージをビルドできます。そのためには、ブループリントとイメージタイプを指定する必要があります。必要に応じて、ディストリビューションを指定することもできます。ディストリビューションを指定しない場合は、ホストシステムと同じディストリビューションとバージョンが使用されます。アーキテクチャーもホストのアーキテクチャーと同じになります。

前提条件

- イメージにブループリントを用意している。コマンドラインインターフェイスを使用した RHEL Image Builder ブループリントの作成 を参照してください。

手順

1. オプション: 作成できるイメージ形式をリストします。

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

2. Compose を起動します。

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

BLUEPRINT-NAME をブループリントの名前に、**IMAGE-TYPE** をイメージのタイプに置き換えます。使用可能な値は、**composer-cli compose types** コマンドの出力を参照してください。

作成プロセスはバックグラウンドで開始され、作成者の Universally Unique Identifier (UUID) が表示されます。

3. イメージの作成が完了するまでに最大 10 分かかる場合があります。
Compose のステータスを確認するには、以下のコマンドを実行します。

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

Compose が完了すると、ステータスが **FINISHED** になります。リスト内の Compose を識別するには、その UUID を使用します。

4. Compose プロセスが完了したら、作成されたイメージファイルをダウンロードします。

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

UUID は、前の手順で示した UUID 値に置き換えます。

検証

イメージを作成したら、次のコマンドを使用してイメージ作成の進行状況を確認できます。

- イメージのメタデータをダウンロードして、Compose 用のメタデータの **.tar** ファイルを取得します。

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

- イメージのログをダウンロードします。

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

このコマンドは、イメージ作成のログを含む **.tar** ファイルを作成します。ログが空の場合は、ジャーナルを確認できます。

- ジャーナルを確認してください。

```
$ journalctl | grep osbuild
```

- イメージのマニフェストを確認します。

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

ジャーナルで **job_UUID** .json を見つけることができます。

関連情報

- Tracing RHEL image builder (Red Hat ナレッジベース)

4.6. RHEL IMAGE BUILDER コマンドラインの基本的なコマンド

RHEL Image Builder コマンドラインインターフェイスでは、以下のサブコマンドを利用できます。

ブループリント操作

利用可能なブループリント一覧の表示

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

TOML 形式でブループリントの内容の表示

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

TOML 形式のブループリントの内容を BLUEPRINT-NAME.toml ファイルに保存 (エクスポート)

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

ブループリントの削除

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

TOML 形式のブループリントファイルを RHEL Image Builder へプッシュ (インポート)

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

ブループリントでイメージの設定

利用可能なイメージタイプをリスト表示します。

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

Compose の起動

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

Compose のリスト表示

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

Compose、およびそのステータスのリスト表示

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

実行中の Compose のキャンセル

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

完了した Compose の削除

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

Compose の詳細情報の表示

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

Compose のイメージファイルのダウンロード

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

サブコマンドとオプションをもっと見る

```
# composer-cli help
```

関連情報

- システムの **composer-cli**(1) の man ページ

4.7. RHEL IMAGE BUILDER のブループリント形式

RHEL Image Builder のブループリントは、TOML 形式のプレーンテキストとしてユーザーに表示されます。

一般的なブループリントファイルの要素には、次のものが含まれます。

ブループリントのメタデータ

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

BLUEPRINT-NAME および LONG FORM DESCRIPTION TEXT フィールドは、ブループリントの名前と説明です。

VERSION は、セマンティックバージョニングスキームに従ったバージョン番号であり、ブループリントファイル全体で 1 つだけ存在するものです。

イメージに追加するグループ

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

group エントリーは、イメージにインストールするパッケージのグループを説明します。グループは、次のパッケージカテゴリーを使用します。

- 必須
- デフォルト
- 任意
  group-name は、anaconda-tools、widget、wheel または users などのグループの名前です。ブループリントは、必須パッケージとデフォルトパッケージをインストールします。オプションパッケージを選択するメカニズムはありません。

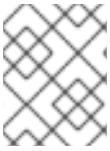
イメージに追加するパッケージ

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

package-name は、httpd、gdb-doc、または coreutils などのパッケージの名前です。

package-version は使用するバージョンです。このフィールドは、**dnf** バージョンの指定に対応します。

- 特定のバージョンについては、8.7.0 などの正確なバージョン番号を使用してください。
- 利用可能な最新バージョンを指定する場合は、アスタリスク (*) を使用します。
- 最新のマイナーバージョンの場合は、8.* などの形式を使用します。
  追加するすべてのパッケージにこのブロックを繰り返します。



注記

RHEL Image Builder ツールのパッケージとモジュールの間に違いはありません。どちらも RPM パッケージの依存関係として扱われます。

4.8. サポートされているイメージのカスタマイズ

ブループリントに次のようなカスタマイズを追加することで、イメージをカスタマイズできます。

- RPM パッケージの追加
- サービスの有効化
- カーネルコマンドラインパラメーターのカスタマイズ

とりわけ、ブループリント内ではいくつかのイメージのカスタマイズを使用できます。カスタマイズを使用すると、デフォルトのパッケージでは使用できないパッケージやグループをイメージに追加できます。これらのオプションを使用するには、ブループリントでカスタマイズを設定し、それを RHEL Image Builder にインポート (プッシュ) します。

関連情報

- Blueprint import fails after adding filesystem customization "size" (Red Hat ナレッジベース)

4.8.1. ディストリビューションの選択

distro フィールドを使用して、イメージを作成するときやブループリント内の依存関係を解決するときに使用するディストリビューションを選択できます。**distro** が空白のままの場合、ホストのディストリビューションが使用されます。ディストリビューションを指定しない場合、ブループリントはホストディストリビューションを使用します。ホストオペレーティングシステムをアップグレードする場合、ディストリビューションが設定されていないブループリントでは、新しいオペレーティングシステムのバージョンを使用してイメージがビルドされます。RHEL Image Builder ホストとは異なるオペレーティングシステムイメージをビルドすることはできません。

- 指定の RHEL イメージを常にビルドするように、RHEL ディストリビューションを使用してブループリントをカスタマイズします。

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

以下に例を示します。

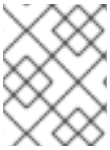
```
name = "tmux"
description = "tmux image with openssh"
version = "1.2.16"
distro = "rhel-9.5"
```

別のマイナーバージョンをビルドするには、**"different_minor_version"** を置き換えます。たとえば、RHEL 9.5 イメージをビルドする場合は、**distro** = "rhel-94" を使用します。RHEL 9.3 イメージでは、RHEL 9.3、RHEL 8.10、およびそれ以前のリリースなどのマイナーバージョンをビルドできます。

4.8.2. パッケージグループの選択

パッケージグループを使用してブループリントをカスタマイズします。イメージにインストールするパッケージのグループを、**groups** リストに記述します。パッケージグループはリポジトリーのメタデータで定義されます。各グループには、主にユーザーインターフェイスでの表示に使用されるわかりやすい名前と、キックスタートファイルでよく使用される ID があります。この場合、ID を使用してグループをリストする必要があります。グループでは、必須、デフォルト、任意の 3 つの方法でパッケージを分類できます。ブループリントには、必須パッケージとデフォルトパッケージのみがインストールされます。任意のパッケージを選択することはできません。

name 属性は、必須の文字列であり、リポジトリー内のパッケージグループ ID と正確に一致する必要があります。



注記

現在、**osbuild-composer** のパッケージとモジュールの間に違いはありません。どちらも RPM パッケージの依存関係として扱われます。

- パッケージを使用してブループリントをカスタマイズします。

```
[[groups]]
name = "group_name"
```

group_name は、グループの名前に置き換えます。たとえば、**anaconda-tools** です。

```
[[groups]]
name = "anaconda-tools"
```

4.8.3. コンテナーの埋め込み

ブループリントをカスタマイズして、最新の RHEL コンテナーを埋め込むことができます。ソースを含むオブジェクトと、必要に応じて **tls-verify** 属性をコンテナーリストに含めます。

イメージに埋め込むコンテナーイメージを、コンテナーリストのエントリーに記述します。

- **source** - 必須フィールド。これは、レジストリーにあるコンテナーイメージへの参照です。この例では、**registry.access.redhat.com** レジストリーを使用します。タグのバージョンを指定できます。デフォルトのタグバージョンは latest です。
- **name** - ローカルレジストリー内のコンテナーの名前。
- **tls-verify** - ブールフィールド。tls-verify ブールフィールドは、Transport Layer Security を制御します。デフォルト値は true です。

埋め込まれたコンテナーは自動的に起動しません。これを起動するには、**systemd** ユニットファイルまたは **quadlets** を作成し、ファイルをカスタマイズします。

- **registry.access.redhat.com/ubi9/ubi:latest** のコンテナーとホストのコンテナーを埋め込むには、ブループリントに次のカスタマイズを追加します。

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

```
[[containers]]
source = "localhost/test:latest"
local-storage = true
```

containers-auth.json ファイルを使用して、保護されたコンテナーリソースにアクセスできます。コンテナーレジストリーの認証情報 を参照してください。

4.8.4. イメージのホスト名の設定

customizations.hostname は、最終イメージのホスト名を設定するために使用できるオプションの文字列です。このカスタマイズはオプションであり、設定しない場合、ブループリントはデフォルトのホスト名を使用します。

- ブループリントをカスタマイズしてホスト名を設定します。

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

4.8.5. 追加ユーザーの指定

ユーザーをイメージに追加し、必要に応じて SSH キーを設定します。このセクションのフィールドは、**name** を除いてすべてオプションです。

手順

- ブループリントをカスタマイズして、イメージにユーザーを追加します。

```
[[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
```

```
[[customizations.user]]
name = "admin"
description = "Administrator account"
password = "$6$CHO2$3rN8eviE2t50lmVyBYihTgVRHcaecmeCk31L..."
key = "PUBLIC SSH KEY"
home = "/srv/widget/"
shell = "/usr/bin/bash"
groups = ["widget", "users", "wheel"]
uid = 1200
gid = 1200
expiredate = 12345
```

GID はオプションであり、イメージにすでに存在している必要があります。GID は、必要に応じて、パッケージによって作成されるか、ブループリントによって **[[customizations.group]]** エントリーを使用して作成されます。

PASSWORD-HASH は、実際の **password hash** に置き換えます。**password hash** を生成するには、次のようなコマンドを使用します。

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

その他のプレースホルダーを、適切な値に置き換えます。

name の値を入力し、不要な行は省略します。

追加するすべてのユーザーにこのブロックを繰り返します。

4.8.6. 追加グループの指定

作成されるシステムイメージのグループを指定します。**name** 属性と **gid** 属性は両方とも必須です。

- グループを使用してブループリントをカスタマイズします。

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

追加するすべてのグループにこのブロックを繰り返します。以下に例を示します。

```
[[customizations.group]]
name = "widget"
gid = 1130
```

4.8.7. 既存ユーザーの SSH キーの設定

customizations.sshkey を使用して、最終イメージ内の既存ユーザーの SSH キーを設定できます。**user** 属性と **key** 属性は両方とも必須です。

- 既存ユーザーの SSH キーを設定してブループリントをカスタマイズします。

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

以下に例を示します。

```
[[customizations.sshkey]]
user = "root"
key = "SSH key for root"
```

注記

既存ユーザーに対してのみ、**customizations.sshkey** カスタマイズを設定できます。ユーザーを作成して SSH キーを設定するには、追加ユーザーの指定 のカスタマイズを参照してください。

4.8.8. カーネル引数の追加

ブートローダーのカーネルコマンドラインに引数を追加できます。デフォルトでは、RHEL Image Builder はデフォルトのカーネルをイメージにビルドします。ただし、ブループリントでカーネルを設定することでカーネルをカスタマイズできます。

- デフォルト設定にカーネルの起動パラメーターオプションを追加します。

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

以下に例を示します。

```
[customizations.kernel]
name = "kernel-debug"
append = "nosmt=force"
```

4.8.9. リアルタイムカーネルを使用した RHEL イメージのビルド

リアルタイムカーネル (**kernel-rt**) を使用して RHEL イメージをビルドするには、デフォルトカーネルとして **kernel-rt** が正しく選択されたイメージをビルドできるように、リポジトリーをオーバーライドする必要があります。**/usr/share/osbuild-composer/repositories/** ディレクトリーの **.json** を使用してください。その後、ビルドしたイメージをシステムにデプロイし、リアルタイムカーネル機能を使用できます。

注記

リアルタイムカーネルは、Red Hat Enterprise Linux の動作認定を受けた AMD64 および Intel 64 サーバープラットフォームで動作します。

前提条件

- システムが登録され、RHEL が RHEL for Real Time サブスクリプションに割り当てられている。dnf を使用した RHEL for Real Time のインストール を参照してください。

手順

1. 以下のディレクトリーを作成します。

```
# mkdir /etc/osbuild-composer/repositories/
```

2. **/usr/share/osbuild-composer/repositories/rhel-9.version.json** ファイルの内容を新しいディレクトリーにコピーします。

```
# cp /usr/share/osbuild-composer/repositories/rhel-9.version.json /etc/osbuild-composer/repositories
```

3. **/etc/osbuild-composer/repositories/rhel-9.version.json** ファイルを編集して、RT カーネルリポジトリーを含めます。

```
# grep -C 6 kernel-rt /etc/osbuild-composer/repositories/rhel-9.version.json
      "baseurl": "https://cdn.redhat.com/content/dist/rhel9/9.version/x86_64/appstream/os",
      "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\nm……..=\n=UZd/\n-----END PGP PUBLIC KEY BLOCK-----\n",
      "rhsm": true,
```

```
      "check_gpg": true
    },
    {
      "name": "kernel-rt",
      "baseurl": "https://cdn.redhat.com/content/dist/rhel9/9.version/x86_64/rt/os",
      "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\nmQINBEr……fg==\n=UZd/\n-
----END PGP PUBLIC KEY BLOCK-----\n",
      "rhsm": true,
      "check_gpg": true
    },
```

4. サービスを再起動します。

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

5. **kernel-rt** が **.json** ファイルに含まれていることを確認します。

```
# composer-cli sources list
# composer-cli sources info kernel-rt
```

以前に設定した URL が表示されます。

6. ブループリントを作成します。ブループリントに、"[customizations.kernel]" カスタマイズを追加します。以下は、ブループリントに "[customizations.kernel]" を追加した例です。

```
name = "rt-kernel-image"
description = ""
version = "2.0.0"
modules = []
groups = []
distro = "rhel-9_version_"
[[customizations.user]]
name = "admin"
password = "admin"
groups = ["users", "wheel"]
[customizations.kernel]
name = "kernel-rt"
append = ""
```

7. ブループリントをサーバーにプッシュします。

```
# composer-cli blueprints push rt-kernel-image.toml
```

8. 作成したブループリントからイメージをビルドします。次の例では、(**.qcow2**) イメージをビルドします。

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

9. ビルドしたイメージを、リアルタイムカーネル機能を使用するシステムにデプロイします。

検証

- イメージから仮想マシンを起動した後、デフォルトのカーネルとして **kernel-rt** が正しく選択されたイメージがビルドされていることを確認します。

```
$ cat /proc/cmdline
BOOT_IMAGE=(hd0,got3)/vmlinuz-5.14.0-362.24.1..el9_version_.x86_64+rt...
```

4.8.10. タイムゾーンと NTP の設定

ブループリントをカスタマイズして、タイムゾーンと **Network Time Protocol** (NTP) を設定できます。**timezone** 属性と **ntpservers** 属性は両方ともオプションの文字列です。タイムゾーンをカスタマイズしない場合、システムは **協定世界時** (UTC) を使用します。NTP サーバーを設定しない場合、システムはデフォルトのディストリビューションを使用します。

- 必要な **timezone** と **ntpservers** を使用してブループリントをカスタマイズします。

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

以下に例を示します。

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

注記

Google Cloud などの一部のイメージタイプには、すでに NTP サーバーがセットアップされています。そのようなイメージでは、選択されている環境で NTP サーバーを起動する必要があるため、これをオーバーライドすることはできません。ただし、ブループリントでタイムゾーンをカスタマイズできます。

4.8.11. ロケール設定のカスタマイズ

作成されるシステムイメージのロケール設定をカスタマイズできます。**language** 属性と **keyboard** 属性は両方とも必須です。他の多くの言語を追加できます。最初に追加する言語はプライマリー言語で、他の言語はセカンダリー言語です。

手順

- ロケール設定を行います。

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

以下に例を示します。

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

- 言語でサポートされている値を一覧表示するには、以下のコマンドを実行します。

```
$ localectl list-locales
```

- キーボードでサポートされている値を一覧表示するには、以下のコマンドを実行します。

```
$ localectl list-keymaps
```

4.8.12. ファイアウォールのカスタマイズ

生成されたシステムイメージのファイアウォールを設定します。デフォルトでは、ファイアウォールは、**sshd** など、ポートを明示的に有効にするサービスを除き、着信接続をブロックします。

[customizations.firewall] または **[customizations.firewall.services]** を使用したくない場合は、属性を削除するか、空のリスト [] に設定します。デフォルトのファイアウォールセットアップのみを使用する場合は、ブループリントからカスタマイズを省略できます。

注記

Google および OpenStack テンプレートは、環境のファイアウォールを明示的に無効にします。ブループリントを設定してこの動作をオーバーライドすることはできません。

手順

- 他のポートとサービスを開くには、次の設定を使用してブループリントをカスタマイズします。

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

ここで、**ports** は、ポート、または開くポートとプロトコルの範囲を含む文字列のオプションのリストです。**port:protocol** 形式を使用してポートを設定できます。**portA-portB:protocol** 形式を使用してポート範囲を設定できます。以下に例を示します。

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

/etc/services の数値ポートまたはその名前を使用して、ポートリストを有効または無効にすることができます。

- **customizations.firewall.service** セクションで、どのファイアウォールサービスを有効または無効にするかを指定します。

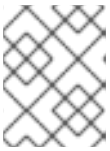
```
[customizations.firewall.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

- 利用可能なファイアウォールサービスを確認できます。

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

以下に例を示します。

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



注記

firewall.services にリストされているサービスは、**/etc/services** ファイルで使用可能な **service-names** とは異なります。

4.8.13. サービスの有効化または無効化

システムの起動時に有効にするサービスを制御することができます。一部のイメージタイプでは、イメージが正しく機能するようにすでにサービスが有効または無効になっており、このセットアップをオーバーライドすることができません。ブループリントの **[customizations.services]** 設定はこれらのサービスを置き換えるものではありませんが、イメージテンプレートにすでに存在するサービスのリストにサービスを追加します。

- 起動時に有効にするサービスをカスタマイズします。

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

以下に例を示します。

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

4.8.14. パーティションモードの指定

ビルドするディスクイメージをパーティション設定する方法を選択するには、**partitioning_mode** 変数を使用します。サポートされている次のモードでイメージをカスタマイズできます。

- **auto-lvm**: 1 つ以上のファイルシステムのカスタマイズがない限り、RAW パーティションモードを使用します。ある場合は、LVM パーティションモードを使用します。
- **lvm**: 追加のマウントポイントがない場合でも、常に LVM パーティションモードを使用します。
- **raw**: マウントポイントが 1 つ以上ある場合でも、RAW パーティションを使用します。
- 次のカスタマイズを使用して、**partitioning_mode** 変数を使用してブループリントをカスタマイズできます。

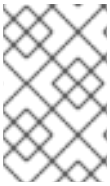
```
[customizations]
partitioning_mode = "lvm"
```

4.8.15. カスタムファイルシステム設定の指定

ブループリントでカスタムファイルシステム設定を指定できるため、デフォルトのレイアウト設定ではなく、特定のディスクレイアウトでイメージを作成できます。ブループリントでデフォルト以外のレイアウト設定を使用すると、次の利点が得られます。

- セキュリティーベンチマークへの準拠
- ディスク外エラーに対する保護

- パフォーマンスの向上
- 既存のセットアップとの整合性



注記

OSTree イメージには読み取り専用などの独自のマウントルールがあるため、OSTree システムではファイルシステムのカスタマイズはサポートされません。次のイメージタイプはサポートされません。

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

さらに、次のイメージタイプはパーティション設定されたオペレーティングシステムイメージを作成しないため、ファイルシステムのカスタマイズをサポートしません。

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

ただし、次のイメージタイプではファイルシステムのカスタマイズがサポートされています。

- **simplified-installer**
- **edge-raw-image**
- **edge-ami**
- **edge-vsphere**

OSTree システムの一部例外を除き、ファイルシステムの **/root** レベルで任意のディレクトリー名を選択できます (例: `/local`、`/mypartition`、**/$PARTITION**)。論理ボリュームでは、これらの変更は LVM パーティションシステム上で行われます。別の論理ボリューム上の **/var**、`/var/log`、および **/var/lib/containers** ディレクトリーがサポートされています。root レベルでの例外は次のとおりです。

- "/home": {Deny: true},
- "/mnt": {Deny: true},
- "/opt": {Deny: true},
- "/ostree": {Deny: true},
- "/root": {Deny: true},
- "/srv": {Deny: true},
- "/var/home": {Deny: true},
- "/var/mnt": {Deny: true},

- "/var/opt": {Deny: true},
- "/var/roothome": {Deny: true},
- "/var/srv": {Deny: true},
- "/var/usrlocal": {Deny: true},

RHEL 8.10 および 9.5 より前のリリースディストリビューションの場合、ブループリントは次の **mountpoints** とそのサブディレクトリーをサポートしています。

- / - ルートマウントポイント
- **/var**
- **/home**
- **/opt**
- **/srv/**
- **/usr**
- **/app**
- **/data**
- **/tmp**

RHEL 9.5 および 8.10 以降のリリースディストリビューションでは、オペレーティングシステム用に予約されている特定のパスを除き、任意のカスタムマウントポイントを指定できます。

次のマウントポイントとそのサブディレクトリーに任意のカスタムマウントポイントを指定することはできません。

- **/bin**
- **/boot/efi**
- **/dev**
- **/etc**
- **/lib**
- **/lib64**
- **/lost+found**
- **/proc**
- **/run**
- **/sbin**
- **/sys**
- **/sysroot**

- **/var/lock**
- **/var/run**

ブループリントで **/usr** カスタムマウントポイントのファイルシステムはカスタマイズできますが、そのサブディレクトリーはカスタマイズできません。

注記

マウントポイントのカスタマイズは、RHEL 9.0 ディストリビューション以降、CLI を使用した場合のみサポートされます。以前のディストリビューションでは、**root** パーティションをマウントポイントとして指定し、size 引数をイメージ **size** のエイリアスとして指定することしかできません。

カスタマイズされたイメージに複数のパーティションがある場合、LVM でカスタマイズされたファイルシステムパーティションを使用してイメージを作成し、実行時にそれらのパーティションのサイズを変更できます。これを行うには、ブループリントでカスタマイズされたファイルシステム設定を指定して、必要なディスクレイアウトでイメージを作成します。デフォルトのファイルシステムレイアウトは変更されません。ファイルシステムをカスタマイズせずにプレーンイメージを使用すると、**cloud-init** によってルートパーティションのサイズが変更されます。

ブループリントは、ファイルシステムのカスタマイズを LVM パーティションに自動的に変換します。

カスタムファイルブループリントのカスタマイズを使用して、新しいファイルを作成したり、既存のファイルを置き換えたりできます。指定するファイルの親ディレクトリーが存在している必要があります。存在しない場合、イメージのビルドが失敗します。**[[customizations.directories]]** のカスタマイズで親ディレクトリーを指定して、親ディレクトリーが存在することを確認してください。

警告

ファイルのカスタマイズを他のブループリントのカスタマイズと組み合わせると、他のカスタマイズの機能に影響が生じたり、現在のファイルのカスタマイズがオーバーライドされる可能性があります。

4.8.15.1. カスタマイズされたファイルをブループリントで指定する

[[customizations.files]] ブループリントのカスタマイズを使用すると、次のことが可能になります。

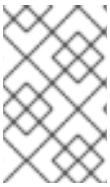
- 新しいテキストファイルを作成する。
- 既存のファイルを変更する。警告: これにより、既存のコンテンツが上書きされる可能性があります。
- 作成するファイルのユーザーとグループの所有権を設定する。
- モード許可を 8 進数形式で設定する。

以下のファイルは作成または置き換えることはできません。

- **/etc/fstab**

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

[[customizations.files]] および **[[customizations.directories]]** ブループリントのカスタマイズを使用して、イメージ内にカスタマイズされたファイルとディレクトリーを作成できます。これらのカスタマイズは、**/etc** ディレクトリーでのみ使用できます。



注記

これらのブループリントのカスタマイズは、OSTree コミットをデプロイするイメージタイプ (**edge-raw-image**、**edge-installer**、**edge-simplified-installer** など) を除く、すべてのイメージタイプでサポートされます。

警告

mode、**user**、または **group** がすでに設定されているイメージ内にすでに存在するディレクトリーパスで **customizations.directories** を使用すると、イメージビルドで既存のディレクトリーの所有権または権限の変更を防ぐことができません。

4.8.15.2. カスタマイズされたディレクトリーをブループリントで指定する

[[customizations.directories]] ブループリントのカスタマイズを使用すると、以下を行うことができます。

- 新しいディレクトリーを作成する。
- 作成するディレクトリーのユーザーとグループの所有権を設定する。
- ディレクトリーモードのパーミッションを 8 進数形式で設定する。
- 必要に応じて親ディレクトリーを作成する。

[[customizations.files]] ブループリントのカスタマイズを使用すると、次のことが可能になります。

- 新しいテキストファイルを作成する。
- 既存のファイルを変更する。警告: これにより、既存のコンテンツが上書きされる可能性があります。
- 作成するファイルのユーザーとグループの所有権を設定する。
- モード許可を 8 進数形式で設定する。

注記

以下のファイルは作成または置き換えることはできません。

- **/etc/fstab**

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

以下のカスタマイズが可能です。

- ブループリントのファイルシステム設定をカスタマイズします。

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

MINIMUM-PARTITION-SIZE 値には、デフォルトのサイズ形式はありません。ブループリントのカスタマイズでは、kB から TB、および KiB から TiB の値と単位がサポートされています。たとえば、マウントポイントのサイズをバイト単位で定義できます。

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

- 単位を使用してマウントポイントのサイズを定義します。以下に例を示します。

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

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

- **minsize** を設定して最小パーティションを定義します。以下に例を示します。

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

- **[[customizations.directories]]** を使用して、イメージ用にカスタマイズされたディレクトリーを **/etc** ディレクトリーの下に作成します。

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

ブループリントの各エントリーを説明します。

  - **path** - 必須 - 作成するディレクトリーへのパスを入力します。**/etc** ディレクトリー下の絶対パスである必要があります。

- **mode** - オプション - ディレクトリーのアクセスパーミッションを 8 進数形式で設定します。パーミッションを指定しない場合、デフォルトで 0755 に設定されます。先頭のゼロは任意です。
- **user** - オプション - ユーザーをディレクトリーの所有者として設定します。ユーザーを指定しない場合は、デフォルトで **root** に設定されます。ユーザーは文字列または整数として指定できます。
- **group** - オプション - グループをディレクトリーの所有者として設定します。グループを指定しない場合は、デフォルトで **root** になります。グループは文字列または整数として指定できます。
- **ensure_parents** - オプション - 必要に応じて親ディレクトリーを作成するかどうかを指定します。値を指定しない場合は、デフォルトで **false** に設定されます。

- **[[customizations.directories]]** を使用して、イメージ用にカスタマイズされたファイルを **/etc** ディレクトリーの下に作成します。

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

ブループリントの各エントリーを説明します。

- **path** - 必須 - 作成するファイルへのパスを入力します。**/etc** ディレクトリー下の絶対パスである必要があります。
- **mode** - オプション - ファイルのアクセスパーミッションを 8 進数形式で設定します。パーミッションを指定しない場合、デフォルトで 0644 に設定されます。先頭のゼロは任意です。
- **user** - オプション - ユーザーをファイルの所有者として設定します。ユーザーを指定しない場合は、デフォルトで **root** に設定されます。ユーザーは文字列または整数として指定できます。
- **group** - オプション - グループをファイルの所有者として設定します。グループを指定しない場合は、デフォルトで **root** になります。グループは文字列または整数として指定できます。
- **data** - オプション - プレーンテキストファイルの内容を指定します。コンテンツを指定しない場合は、空のファイルが作成されます。

4.9. RHEL IMAGE BUILDER によってインストールされるパッケージ

RHEL Image Builder を使用してシステムイメージを作成すると、システムは一連のベースパッケージグループをインストールします。

注記

ブループリントにコンポーネントを追加する場合は、追加したコンポーネント内のパッケージが他のパッケージコンポーネントと競合しないようにしてください。そうしないと、システムは依存関係を解決できず、カスタマイズされたイメージの作成に失敗します。次のコマンドを実行して、パッケージ間に競合がないかどうかを確認できます。

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

表4.1イメージタイプの作成をサポートするデフォルトパッケージ

イメージタイプ	デフォルトパッケージ
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-util、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**

イメージタイプ	デフォルトパッケージ
edge-commit	redhat-release、glibc、glibc-minimal-langpack、nss-altfiles、dracut-config-generic、dracut-network、basesystem、bash、platform-python、shadow-utils、chrony、setup、shadow-utils、sudo、systemd、coreutils、util-linux、curl、vim-minimal、rpm、rpm-ostree、polkit、lvm2、cryptsetup、pinentry、e2fsprogs、dosfstools、keyutils、gnupg2、attr、xz、gzip、firewalld、iptables、NetworkManager、NetworkManager-wifi、NetworkManager-wwan、wpa_supplicant、traceroute、hostname、iproute、iputils、openssh-clients、procps-ng、rootfiles、openssh-server、passwd、policycoreutils、policycoreutils-python-utils、selinux-policy-targeted、setools-console、less、tar、rsync、usbguard、bash-completion、tmux、ima-evm-utils、audit、podman、containernetworking-plugins、container-selinux、skopeo、criu、slirp4netns、fuse-overlayfs、clevis、clevis-dracut、clevis-luks、greenboot、greenboot-default-health-checks、fdo-client、fdo-owner-cli、sos
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

イメージタイプ	デフォルトパッケージ
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**

イメージタイプ	デフォルトパッケージ
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	**aajohan-comfortaa-fonts、abattis-cantarell-fonts、alsa-firmware、alsa-tools-firmware、anaconda、anaconda-dracut、anaconda-install-env-deps、anaconda-widgets、audit、bind-utils、bitmap-fangsongti-fonts、bzip2、cryptsetup、curl、dbus-x11、dejavu-sans-fonts、dejavu-sans-mono-fonts、device-mapper-persistent-data、dmidecode、dnf、dracut-config-generic、dracut-network、efibootmgr、ethtool、fcoe-utils、ftp、gdb-gdbserver、gdisk、glibc-all-langpacks、gnome-kiosk、google-noto-sans-cjk-ttc-fonts、grub2-tools、grub2-tools-extra、grub2-tools-minimal、grubby、gsettings-desktop-schemas、hdparm、hexedit、hostname、initscripts、ipmitool、iwl1000-firmware、iwl100-firmware、iwl105-firmware、iwl135-firmware、iwl2000-firmware、iwl2030-firmware、iwl3160-firmware、iwl5000-firmware、iwl5150-firmware、iwl6000g2a-firmware、iwl6000g2b-firmware、iwl6050-firmware、iwl7260-firmware、jomolhari-fonts、kacst-farsi-fonts、kacst-qurn-fonts、kbd、kbd-misc、kdump-anaconda-addon、kernel、khmeros-base-fonts、less、libblockdev-lvm-dbus、libibverbs、libreport-plugin-bugzilla、libreport-plugin-reportuploader、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、mtr、mt-st、net-tools、nfs-utils、nmap-ncat、nm-connection-editor、nss-tools、openssh-clients、openssh-server、oscap-anaconda-**

イメージタイプ	デフォルトパッケージ
	addon、ostree、pciutils、perl-interpreter、pigz、plymouth、prefixdevname、python3-pyatspi、rdma-core、redhat-release-eula、rng-tools、rpcbind、rpm-ostree、rsync、rsyslog、selinux-policy-targeted、sg3_utils、sil-abyssinica-fonts、sil-padauk-fonts、sil-scheherazade-fonts、smartmontools、smc-meera-fonts、spice-vdagent、strace、systemd、tar、thai-scalable-waree-fonts、tigervnc-server-minimal、tigervnc-server-module、udisks2,udisks2-iscsi,usbutils,vim-minimal、volume_key、wget、xfsdump、xfsprogs、xorg-x11-drivers、xorg-x11-fonts-misc、xorg-x11-server-utils、xorg-x11-server-Xorg、xorg-x11-xauth、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**

関連情報

- RHEL Image Builder の説明

4.10. カスタムイメージで有効なサービス

Image Builder を使用してカスタムイメージを設定する場合、イメージが使用するデフォルトのサービスは次のように決定されます。

- **osbuild-composer** ユーティリティーを使用する RHEL リリース
- イメージの種類

たとえば、**ami** イメージタイプは、デフォルトで **sshd**、**chronyd**、および **cloud-init** サービスを有効にします。これらのサービスが有効になっていない場合、カスタムイメージは起動しません。

表4.2 イメージタイプの作成をサポートするために有効になっているサービス

イメージタイプ	デフォルトで有効化されているサービス
ami	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
openstack	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final

イメージタイプ	デフォルトで有効化されているサービス
qcow2	cloud-init
rhel-edge-commit	デフォルトでは、追加のサービスは有効になりません。
tar	デフォルトでは、追加のサービスは有効になりません。
vhd	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk	sshd、chronyd、vmtoolsd、cloud-init

注記: システムの起動時に有効にするサービスをカスタマイズできます。ただし、カスタマイズは、前述のイメージタイプに対してデフォルトで有効になっているサービスを上書きしません。

関連情報

- サポートされているイメージのカスタマイズ

第5章 RHEL IMAGE BUILDER WEB コンソールインターフェイスを使用したシステムイメージの作成

RHEL Image Builder は、カスタムのシステムイメージを作成するツールです。RHEL Image Builder を制御してカスタムシステムイメージを作成する場合は、Web コンソールインターフェイスを使用できます。ただし、コマンドラインインターフェイス の方が提供している機能が多いため、コマンドラインインターフェイスを使用することが推奨されます。

5.1. RHEL WEB コンソールでの RHEL IMAGE BUILDER ダッシュボードへのアクセス

RHEL Web コンソール用の **cockpit-composer** プラグインを使用すると、グラフィカルインターフェイスを使用して Image Builder のブループリントと設定を管理できます。

前提条件

- システムへの root アクセス権限がある。
- RHEL Image Builder がインストールされている。
- **cockpit-composer** パッケージがインストールされている。

手順

1. ホスト上で、Web ブラウザーで **https://<_localhost_>:9090/** を開きます。
2. root ユーザーとして Web コンソールにログインします。
3. RHEL Image Builder のコントロールを表示するには、ウィンドウの左上隅にある **Image Builder** ボタンをクリックします。
   RHEL Image Builder ダッシュボードが開き、既存のブループリントがあればそれがリストされます。

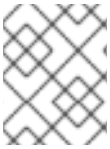
関連情報

- RHEL 9 Web コンソールを使用したシステムの管理

5.2. WEB コンソールインターフェイスでのブループリントの作成

カスタマイズした RHEL システムイメージをビルドするには、ブループリントを作成する必要があります。利用可能なカスタマイズはすべて任意です。次の方法を使用して、カスタマイズしたブループリントを作成できます。

- CLI を使用する。サポートされているイメージのカスタマイズ を参照してください。
- Web コンソールを使用する。次の手順に従ってください。



注記

これらのブループリントのカスタマイズは、Red Hat Enterprise Linux 9.2 以降のバージョンおよび Red Hat Enterprise Linux 8.8 以降のバージョンで利用できます。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。RHEL Web コンソールでの RHEL Image Builder GUI へのアクセス を参照してください。

手順

1. 右上隅にある **Create Blueprint** をクリックします。
ブループリントの名前と説明のフィールドを含むダイアログウィザードが開きます。

2. **Details** ページで以下を行います。

   a. ブループリントの名前と、必要に応じてその説明を入力します。

   b. **Next** をクリックします。

3. オプション: **Packages** ページで、以下を行います。

   a. **Available packages** の検索で、パッケージ名を入力します。

   b. **>** ボタンをクリックして **Chosen packages** フィールドに移動します。

   c. 前の手順を繰り返して、必要な数のパッケージを検索して含めます。

   d. **Next** をクリックします。

注記

特に指定がない限り、これらのカスタマイズはすべてオプションです。

4. **Kernel** ページで、カーネル名とコマンドライン引数を入力します。

5. **File system** ページで、お使いのイメージファイルシステムに合わせて **Use automatic partitioning** または **Manually configure partitions** を選択します。パーティションを手動で設定するには、次の手順を実行します。

   a. **Manually configure partitions** ボタンをクリックします。
   Configure partitions セクションが開き、Red Hat 標準およびセキュリティーガイドに基づく設定が表示されます。

   b. ドロップダウンメニューから、パーティションを設定するための詳細を入力します。

      i. **Mount point** フィールドに、以下のマウントポイントタイプオプションのいずれかを選択します。

      - / - ルートマウントポイント
      - **/app**
      - **/boot**
      - **/data**
      - **/home**
      - **/opt**
      - **/srv/**

- **/usr**
- **/usr/local**
- **/var**
  /tmp などの追加のパスを **Mount point** に追加することもできます。例: 接頭辞 **/var** と、追加パス **/tmp** で、**/var/tmp** になります。

注記

選択した Mount point のタイプに応じて、ファイルシステムのタイプが **xfs** に変わります。

ii. ファイルシステムの **Minimum size partition** フィールドに、必要な最小パーティションサイズを入力します。Minimum size ドロップダウンメニューでは、**GiB**、**MiB**、**KiB** などの一般的なサイズ単位を使用できます。デフォルトの単位は **GiB** です。

注記

Minimum size とは、作業用イメージの作成には小さすぎる場合にも RHEL Image Builder がパーティションサイズを増加できるという意味です。

c. さらにパーティションを追加するには、**Add partition** ボタンをクリックします。エラーメッセージ **Duplicate partitions: Only one partition at each mount point can be created.** が表示された場合は、以下を行います。

i. **Remove** ボタンをクリックして、重複したパーティションを削除します。

ii. 作成するパーティションの新しいマウントポイントを選択します。

d. パーティションの設定が完了したら、**Next** をクリックします。

6. **Services** ページで、サービスを有効または無効にします。

a. 有効または無効にするサービス名をコンマまたはスペースで区切るか、**Enter** キーを押して入力します。**Next** をクリックします。

b. **Enabled services** を入力します。

c. **Disabled services** を入力します。

7. **Firewall** ページで、ファイアウォールを設定します。

a. **Ports** と、有効または無効にするファイアウォールサービスを入力します。

b. **Add zone** ボタンをクリックして、各ゾーンのファイアウォールルールを個別に管理します。**Next** をクリックします。

8. **Users** ページで、以下の手順に従ってユーザーを追加します。

a. **Add user** をクリックします。

b. **Username**、**Password**、および **SSH key** を入力します。**Server administrator** チェックボックスをクリックして、ユーザーを特権ユーザーとしてマークすることもできます。**Next** をクリックします。

9. **Groups** ページで、次の手順を実行してグループを追加します。
    a. **Add groups** ボタンをクリックします。
        i. **Group name** と **Group ID** を入力します。グループをさらに追加できます。**Next** をクリックします。
10. **SSH keys** ページで、キーを追加します。
    a. **Add key** ボタンをクリックします。
        i. SSH キーを入力します。
        ii. **User** を入力します。**Next** をクリックします。
11. **Timezone** ページで、タイムゾーンを設定します。
    a. **Timezone** フィールドに、システムイメージに追加するタイムゾーンを入力します。たとえば、タイムゾーン形式 "US/Eastern" を追加します。
    タイムゾーンを設定しない場合、システムはデフォルトとして協定世界時 (UTC) を使用します。
    b. **NTP servers** を入力します。**Next** をクリックします。
12. **Locale** ページで、以下の手順を実行します。
    a. **Keyboard** 検索フィールドに、システムイメージに追加するパッケージ名を入力します。たとえば、["en_US.UTF-8"] と入力します。
    b. **Languages** 検索フィールドに、システムイメージに追加するパッケージ名を入力します。たとえば、"us" と入力します。**Next** をクリックします。
13. **Others** ページで、次の手順を実行します。
    a. **Hostname** フィールドに、システムイメージに追加するホスト名を入力します。ホスト名を追加しない場合、オペレーティングシステムによってホスト名が決定されます。
    b. Simplifier インストーラーイメージの場合のみ必須: **Installation Devices** フィールドに、システムイメージの有効なノードを入力します。たとえば、**dev/sda1** と入力します。**Next** をクリックします。
14. FDO 用のイメージをビルドする場合のみ必須: **FIDO Device Onboarding** ページで、次の手順を実行します。
    a. **Manufacturing server URL** フィールドに、次の情報を入力します。
        i. **DIUN public key insecure** フィールドに、セキュアでない公開鍵を入力します。
        ii. **DIUN public key hash** フィールドに、公開鍵ハッシュを入力します。
        iii. **DIUN public key root certs** フィールドに、公開鍵ルート証明書を入力します。**Next** をクリックします。
15. **OpenSCAP** ページで、次の手順を実行します。
    a. **Datastream** フィールドに、システムイメージに追加する **datastream** 修復手順を入力します。

b. **Profile ID** フィールドに、システムイメージに追加する **profile_id** セキュリティープロファイルを入力します。**Next** をクリックします。

16. Ignition を使用するイメージをビルドする場合にのみ必須: **Ignition** ページで、次の手順を実行します。

    a. **Firstboot URL** フィールドに、システムイメージに追加するパッケージ名を入力します。

    b. **Embedded Data** フィールドに、ファイルをドラッグまたはアップロードします。**Next** をクリックします。

17. .**Review** ページで、ブループリントの詳細を確認します。**Create** をクリックします。

RHEL Image Builder ビューが開き、既存のブループリントのリストが表示されます。

5.3. RHEL IMAGE BUILDER WEB コンソールインターフェイスでのブループリントのインポート

既存のブループリントをインポートして使用できます。システムがすべての依存関係を自動的に解決します。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- RHEL Image Builder Web コンソールインターフェイスで使用するためにインポートするブループリントがある。

手順

1. RHEL Image Builder ダッシュボードで、**Import blueprint** をクリックします。**Import blueprint** が開きます。
2. **Upload** フィールドから、既存のブループリントをドラッグまたはアップロードします。**TOML** と **JSON** のどちらかの形式のブループリントを使用できます。
3. **Import** をクリックします。ダッシュボードに、インポートしたブループリントがリストされます。

検証

インポートしたブループリントをクリックすると、インポートしたブループリントのすべてのカスタマイズを含むダッシュボードにアクセスできます。

- インポートしたブループリント用に選択されているパッケージを確認するには、**Packages** タブに移動します。
  - すべてのパッケージの依存関係を一覧表示するには、**All** をクリックします。リストは検索可能で、並べ替えることもできます。

次のステップ

- オプション: カスタマイズを変更するには、以下を実行します。

- **Customizations** ダッシュボードから、変更するカスタマイズをクリックします。必要に応じて、**Edit blueprint** をクリックして、利用可能なすべてのカスタマイズオプションに移動できます。

関連情報

- Web コンソールインターフェイスで RHEL Image Builder を使用してシステムイメージを作成する

5.4. RHEL IMAGE BUILDER WEB コンソールインターフェイスからのブループリントのエクスポート

ブループリントをエクスポートして、別のシステムでカスタマイズを使用できます。ブループリントは **TOML** または **JSON** 形式でエクスポートできます。どちらの形式も CLI だけでなく API インターフェイスでも使用できます。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- エクスポートするブループリントがある。

手順

1. Image Builder ダッシュボードで、エクスポートするブループリントを選択します。
2. **Export blueprint** をクリックします。**Export blueprint** が開きます。
3. **Export** ボタンをクリックしてブループリントをファイルとしてダウンロードするか、**Copy** ボタンをクリックしてブループリントをクリップボードにコピーします。
   a. オプション: ブループリントをコピーするには、**Copy** ボタンをクリックします。

検証

- エクスポートしたブループリントをテキストエディターで開き、検査して確認します。

5.5. WEB コンソールインターフェイスで RHEL IMAGE BUILDER を使用してシステムイメージを作成する

次の手順を実行すると、ブループリントからカスタマイズされた RHEL システムイメージを作成できます。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- ブループリントを作成している。

手順

1. RHEL Image Builder ダッシュボードで、blueprint タブをクリックします。
2. ブループリントのテーブルで、イメージをビルドするブループリントを見つけます。

3. 選択したブループリントの右側で、**Create Image** をクリックします。Create image ダイアログウィザードが開きます。

4. **Image output** ページで、次の手順を実行します。

   a. **Select a blueprint** リストから、必要なイメージのタイプを選択します。

   b. **Image output type** リストから、必要なイメージの出力タイプを選択します。
   選択したイメージの種類に応じて、さらに詳細を追加する必要があります。

5. **Next** をクリックします。

6. **Review** ページで、イメージの作成に関する詳細を確認し、**Create image** をクリックします。
   イメージのビルドが開始され、完了するまでに最大 20 分かかります。

検証

イメージのビルドが完了したら、次のことが可能になります。

- イメージをダウンロードします。
  - RHEL Image Builder ダッシュボードで、**Node options (▮)** メニューをクリックし、**Download image** を選択します。
- イメージのログをダウンロードして要素を検査し、問題がないかどうかを確認します。
  - RHEL Image Builder ダッシュボードで、**Node options (▮)** メニューをクリックし、**Download logs** を選択します。

第6章 RHEL IMAGE BUILDER を使用したブート ISO インストーラーイメージの作成

RHEL Image Builder を使用して、起動可能な ISO インストーラーイメージを作成できます。これらのイメージは、ルートファイルシステムを含む **.tar** ファイルで構成されます。起動可能な ISO イメージを使用して、ファイルシステムをベアメタルサーバーにインストールすることができます。

RHEL Image Builder は、ルートファイルシステムを含むブート ISO を作成するマニフェストを作成します。ISO イメージを作成するには、イメージタイプ **image-installer** を選択します。RHEL Image Builder は、次の内容の **.tar** ファイルをビルドします。

- 標準の Anaconda インストーラー ISO
- 組み込みの RHEL システム tar ファイル
- 最小限のデフォルト要件でコミットをインストールするデフォルトのキックスタートファイル

作成されたインストーラー ISO イメージには、ベアメタルサーバーに直接インストールできる設定済みのシステムイメージが含まれています。

6.1. RHEL IMAGE BUILDER CLI を使用したブート ISO インストーラーイメージの作成

RHEL Image Builder コマンドラインインターフェイスを使用して、カスタマイズしたブート ISO インストーラーイメージを作成できます。その結果、Image Builder は、オペレーティングシステム用にインストールできる **.tar** ファイルを含む **.iso** ファイルをビルドします。**.iso** ファイルは、Anaconda を起動し、**tar** ファイルをインストールしてシステムをセットアップするように設定されています。作成した ISO イメージファイルをハードディスク上で使用したり、HTTP ブートや USB インストールなどで仮想マシンを起動したりできます。

警告

インストーラー (**.iso**) イメージタイプはパーティションのカスタマイズを受け入れません。ファイルシステムのカスタマイズを手動で設定しようとしても、インストーラーイメージによってビルドされたシステムには適用されません。RHEL Image Builder ファイルシステムのカスタマイズでビルドした ISO イメージをマウントすると、キックスタートでエラーが発生し、インストールは自動的に再起動しません。詳細は、Red Hat ナレッジベースソリューション Automate a RHEL ISO installation generated by image builder を参照してください。

前提条件

- ユーザーを含むイメージのブループリントを作成してカスタマイズし、RHEL Image Builder にプッシュしている。ブループリントのカスタマイズ を参照してください。

手順

1. ISO イメージを作成します。

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

- **BLUEPRINT-NAME** は作成したブループリントの名前です。
- **image-installer** はイメージタイプです。
  Compose プロセスはバックグラウンドで開始し、Compose の UUID が表示されます。
  Compose が完成するまで待ちます。これには数分かかる場合があります。

2. Compose のステータスを確認します。

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

Compose が完了すると、ステータスが **FINISHED** となります。

3. リスト内の Compose をその UUID で識別します。

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

4. Compose が完了したら、作成したイメージファイルをカレントディレクトリーにダウンロードします。

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

UUID は、前の手順で取得した UUID 値に置き換えます。

RHEL Image Builder は **.tar** ファイルを含む **.iso** ファイルをビルドします。**.tar** ファイルは、オペレーティングシステム用にインストールされるイメージです。**.iso** は、Anaconda を起動し、**tar** ファイルをインストールしてシステムをセットアップするように設定されています。

次のステップ

イメージファイルをダウンロードしたディレクトリーで、以下を実行します。

1. ダウンロードした **.iso** イメージを見つけます。
2. ISO をマウントします。

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

.tar ファイルは **/mnt/liveimg.tar.gz** ディレクトリーにあります。

3. **.tar** ファイルの内容をリスト表示します。

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

関連情報

- RHEL Image Builder コマンドラインインターフェイスでのシステムイメージの作成
- 起動可能な RHEL 用インストールメディアの作成

6.2. GUI で RHEL IMAGE BUILDER を使用してブート ISO インストーラーイメージを作成する

RHEL Image Builder GUI を使用して、カスタマイズしたブート ISO インストーラーイメージをビルドできます。作成された ISO イメージファイルは、ハードディスク上で使用することも、仮想マシンで起動することもできます。たとえば、HTTP ブートや USB インストールなどで起動できます。

警告

インストーラー (**.iso**) イメージタイプはパーティションのカスタマイズを受け入れません。ファイルシステムのカスタマイズを手動で設定しようとしても、インストーラーイメージによってビルドされたシステムには適用されません。RHEL Image Builder ファイルシステムのカスタマイズでビルドした ISO イメージをマウントすると、キックスタートでエラーが発生し、インストールは自動的に再起動しません。詳細は、Red Hat ナレッジベースソリューション Automate a RHEL ISO installation generated by image builder を参照してください。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- イメージのブループリントを作成している。Web コンソールインターフェイスでの RHEL Image Builder ブループリントの作成 を参照してください。

手順

1. RHEL Image Builder のダッシュボードで、イメージのビルドに使用するブループリントを見つけます。必要に応じて、ブループリント名またはその一部を左上の検索ボックスに入力し、**Enter** をクリックします。
2. ブループリントの右側で、対応する **Create Image** ボタンをクリックします。**Create image** ダイアログウィザードが開きます。
3. **Create image** ダイアログウィザードで:
   a. **Image Type** リストで、**"RHEL Installer (.iso)"** を選択します。
   b. **Next** をクリックします。
   c. **Review** タブで **Create** をクリックします。
   RHEL Image Builder は、RHEL .iso イメージの Compose をキューに追加します。

   プロセスが完了すると、Image build complete のステータスが表示されます。RHEL Image Builder は ISO イメージを作成します。

検証

イメージが正常に作成されたら、ダウンロードできます。

1. **Download** をクリックして、**"RHEL Installer (.iso)"** イメージをシステムに保存します。
2. **"RHEL Installer (.iso)"** イメージをダウンロードしたフォルダーに移動します。
3. ダウンロードした .tar イメージを見つけます。

4. **"RHEL Installer (.iso)"** イメージの内容を展開します。

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

関連情報

- Web コンソールインターフェイスでの RHEL Image Builder ブループリントの作成
- RHEL Image Builder コマンドラインインターフェイスでのシステムイメージの作成
- 起動可能な RHEL 用インストールメディアの作成

6.3. 起動可能な ISO をメディアにインストールして起動する

RHEL Image Builder を使用して作成した起動可能な ISO イメージをベアメタルシステムにインストールします。

前提条件

- RHEL Image Builder を使用して起動可能な ISO イメージを作成している。コマンドラインインターフェイスで RHEL Image Builder を使用してブート ISO インストーラーイメージを作成する を参照してください。
- 起動可能な ISO イメージをダウンロードしました。
- **dd** ツールをインストールしました。
- ISO イメージに十分な容量の USB フラッシュドライブがある。必要なサイズはブループリントで選択したパッケージによって異なりますが、推奨される最小サイズは 8 GB です。

手順

1. **dd** ツールを使用して、ブート可能な ISO イメージを USB ドライブに直接書き込みます。以下に例を示します。

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

ここで、**installer.iso** は ISO イメージファイル名、 **/dev/sdX** は USB フラッシュドライブのデバイスパスです。

2. 起動するコンピューターの USB ポートにフラッシュドライブを挿入します。
3. USB フラッシュドライブから ISO イメージを起動します。
インストール環境が開始したら、デフォルトの Red Hat Enterprise Linux インストールと同様に、インストールの手動完了が必要になることがあります。

関連情報

- インストールメディアの起動
- インストールのカスタマイズ
- Linux で起動可能な USB デバイスの作成

第7章 RHEL IMAGE BUILDER OPENSCAP 統合によるハードニング済みイメージの作成

OpenSCAP 統合の RHEL Image Builder オンプレミスサポートを使用すると、特定のセキュリティープロファイルを使用してカスタマイズしたブループリントを作成し、そのブループリントを使用してハードニング済みイメージをビルドできます。このハードニング済みイメージを使用して、特定のプロファイルに準拠させる必要があるシステムをデプロイできます。ブループリントは、パッケージやアドオンファイルのセットを追加してカスタマイズできます。これにより、カスタマイズしたハードニング済み RHEL イメージをビルドし、プロファイルに準拠したシステムをデプロイする準備を整えることができます。

イメージのビルドプロセス中に、OSBuild **oscap.remediation** ステージが、ファイルシステムツリー上の **chroot** 環境で OpenSCAP ツールを実行します。OpenSCAP ツールは、選択したプロファイルの標準評価を実行し、修復をイメージに適用します。これにより、イメージを初めて起動する前でも、セキュリティープロファイルの要件に従って設定されたイメージをビルドできます。

Red Hat は、現在のデプロイメントガイドラインを満たすことができるように、システムを構築するときに選択できるセキュリティーハードニングプロファイルの定期的に更新されたバージョンを提供します。

7.1. OPENSCAP ブループリントのカスタマイズ

OpenSCAP によるブループリントのカスタマイズのサポートにより、特定のセキュリティープロファイルの `scap-security-guide` コンテンツからブループリントを生成し、そのブループリントを使用して独自のハードニング済みイメージをビルドできます。

OpenSCAP を使用してカスタマイズしたブループリントを作成するには、次の主な手順を実行します。

- 特定の要件に応じてマウントポイントを変更し、ファイルシステムレイアウトを設定します。
- ブループリントで、OpenSCAP プロファイルを選択します。これにより、選択したプロファイルに従って、イメージのビルド中に修復をトリガーするようにイメージが設定されます。また、イメージのビルド中に、OpenSCAP が初回起動前の修復を適用します。

イメージブループリントで **OpenSCAP** ブループリントのカスタマイズを使用するには、次の情報を提供する必要があります。

- **datastream** 修復手順へのデータストリームパス。**scap-security-guide** パッケージのデータストリームファイルは、**/usr/share/xml/scap/ssg/content/** ディレクトリーにあります。
- 必要なセキュリティープロファイルの **profile_id**。**profile_id** フィールドの値は、長い形式と短い形式の両方を受け入れます。たとえば、**cis** や **xccdf_org.ssgproject.content_profile_cis** を受け入れることができます。詳細は、RHEL 9 でサポートされている SCAP セキュリティーガイドプロファイル を参照してください。

次の例は、**OpenSCAP** 修復ステージのスニペットです。

```
[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-rhel9-ds.xml"
profile_id = "xccdf_org.ssgproject.content_profile_cis"
```

次のコマンドを使用すると、**scap-security-guide** パッケージの **SCAP** ソースデータストリームの詳細 (データストリームによって提供されるセキュリティープロファイルのリストを含む) を確認できます。

```
# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

ユーザーの便宜のために、**OpenSCAP** ツールでは、**scap-security-guide** データストリームで利用可能な任意のプロファイルのハードニングブループリントを生成できます。

たとえば、次のコマンドを実行します。

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

次のような CIS プロファイルのブループリントを生成します。

```
# Blueprint for CIS Red Hat Enterprise Linux 9 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 9 Benchmark™, v3.0.0, released 2023-10-30.
# This profile includes Center for Internet Security®
# Red Hat Enterprise Linux 9 CIS Benchmarks™ content.
#
# Profile ID:  xccdf_org.ssgproject.content_profile_cis
# Benchmark ID:  xccdf_org.ssgproject.content_benchmark_RHEL-9
# Benchmark Version:  0.1.74
# XCCDF Version:  1.2

name = "hardened_xccdf_org.ssgproject.content_profile_cis"
description = "CIS Red Hat Enterprise Linux 9 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"]
```

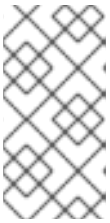
注記

このブループリントスニペットは、イメージのハードニングにそのまま使用しないでください。完全なプロファイルを反映するものではありません。Red Hat は **scap-security-guide** パッケージ内の各プロファイルのセキュリティー要件を継続的に更新および改良しています。そのため、システムに提供されるデータストリームの最新バージョンを使用して初期テンプレートを常に再生成することを推奨します。

これで、ブループリントをカスタマイズすることも、そのまま使用してイメージをビルドすることもできます。

RHEL Image Builder は、ブループリントのカスタマイズに基づいて、**osbuild** ステージに必要な設定を生成します。さらに、RHEL Image Builder は 2 つのパッケージをイメージに追加します。

- **openscap-scanner** - **OpenSCAP** ツール。
- **scap-security-guide** - 修復および評価の手順が含まれるパッケージ。



注記

このパッケージはデフォルトでイメージにインストールされるため、修復ステージではデータストリームに **scap-security-guide** パッケージを使用します。別のデータストリームを使用する場合は、必要なパッケージをブループリントに追加し、**oscap** 設定でデータストリームへのパスを指定します。

関連情報

- RHEL 9 でサポートされる SCAP Security Guide プロファイル

7.2. RHEL IMAGE BUILDER を使用したハードニング済みイメージの作成

OpenSCAP と RHEL Image Builder の統合により、特定のプロファイルに準拠したハードニング済みイメージを作成し、それを仮想マシンやベアメタル環境などにデプロイできます。

前提条件

- root ユーザーまたは **weldr** グループのメンバーであるユーザーとしてログインしている。
- **openscap** および **scap-security-guide** パッケージがインストールされている。

手順

1. **OpenSCAP** ツールと **scap-security-guide** コンテンツを使用して、TOML 形式でハードニングブループリントを作成し、必要に応じて変更します。

```
# oscap xccdf generate fix --profile=<profileID> --fix-type=<blueprint_name>.toml
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml > cis.toml
```

   <profileID> は、システムが準拠する必要があるプロファイル ID (例: **cis**) に置き換えます。

2. **composer-cli** ツールを使用して、ブループリントを **osbuild-composer** にプッシュします。

```
# composer-cli blueprints push <blueprint_name>.toml
```

3. ハードニング済みイメージのビルドを開始します。

```
# composer-cli compose start <blueprint_name> <image_type>
```

   <image_type> は、任意のイメージタイプ (例: **qcow2**) に置き換えます。

   イメージビルドの準備ができたら、デプロイメントでハードニング済みイメージを使用できます。仮想マシンの作成 を参照してください。

検証

ハードニング済みイメージをデプロイした後、設定コンプライアンススキャンを実行して、イメージが選択したセキュリティープロファイルに準拠していることを確認できます。

重要

設定コンプライアンススキャンを実行しても、システムが準拠しているとは限りません。詳細は、設定コンプライアンススキャン を参照してください。

関連情報

- 設定コンプライアンスおよび脆弱性スキャンの開始

7.3. RHEL IMAGE BUILDER を使用したハードニング済みイメージのカスタマイズ

セキュリティープロファイルをカスタマイズするには、特定のルール (パスワードの最小長など) のパラメーターを変更し、別の方法で対象とするルールを削除し、追加のルールを選択して内部ポリシーを実装できます。プロファイルをカスタマイズして新しいルールの定義はできません。

そのブループリントからイメージをビルドすると、新しいテーラリングプロファイル ID を持つテーラリングファイルが作成され、**/usr/share/xml/osbuild-oscap-tailoring/tailoring.xml** としてイメージに保存されます。新しいプロファイル ID は、ベース ID に **_osbuild_tailoring** 接尾辞を追加したものです。たとえば、CIS (**cis**) ベースプロファイルをカスタマイズする場合、プロファイル ID は **xccdf_org.ssgproject.content_profile_cis_osbuild_tailoring** になります。

前提条件

- root ユーザーまたは **weldr** グループのメンバーであるユーザーとしてログインしている。
- **openscap** および **scap-security-guide** パッケージがインストールされている。

手順

1. 選択したプロファイルから TOML 形式でハードニングブループリントを作成します。

```
# oscap xccdf generate fix --profile=<profileID> --fix-type=blueprint
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml > <profileID>-tailored.toml
```

2. テーラリングファイルをブループリントに追加します。テーラリングによるカスタマイズは、カスタマイズのベースとなるプロファイル内の選択済みおよび未選択のルールのデフォルト状態にのみ影響します。プロファイル内のルールの選択または選択解除は行いますが、他のルールの状態を変更することはありません。

```
# Blueprint for CIS Red Hat Enterprise Linux 9 Benchmark for Level 2 - Server
# ...
[customizations.openscap.tailoring]
selected = [ "xccdf_org.ssgproject.content_bind_crypto_policy" ]
unselected = [ "grub2_password" ]
```

3. **composer-cli** ツールを使用して、ブループリントを **osbuild-composer** にプッシュします。

```
# composer-cli blueprints push <blueprintProfileID>-tailored.toml
```

4. ハードニング済みイメージのビルドを開始します。

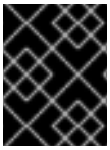
```
# composer-cli compose start <blueprintProfileID> <image_type>
```

<image_type> は、任意のイメージタイプ (例: **qcow2**) に置き換えます。

イメージのビルドが準備できたら、デプロイメントでハードニング済みイメージを使用します。

検証

ハードニング済みイメージをデプロイした後、設定コンプライアンススキャンを実行して、イメージが選択したセキュリティープロファイルに準拠していることを確認できます。



重要

設定コンプライアンススキャンを実行しても、システムが準拠しているとは限りません。詳細は、設定コンプライアンススキャン を参照してください。

関連情報

- 設定コンプライアンスおよび脆弱性スキャンの開始

第8章 RHEL IMAGE BUILDER を使用した FIPS モードの有効化

カスタマイズしたイメージを作成し、FIPS 対応の RHEL イメージを起動できます。イメージを作成する前に、ブループリント内の **fips** ディレクティブの値を変更する必要があります。

前提条件

- root ユーザーまたは **weldr** グループのメンバーであるユーザーとしてログインしている。

手順

1. TOML (Tom's Obvious Minimal Language) 形式で、以下のコンテンツのプレーンテキストファイルを作成します。

```
name = "system-fips-mode-enabled"
description = "blueprint with FIPS enabled "
version = "0.0.1"

[customizations]
fips = true

[[customizations.user]]
name = "admin"
password = "admin"
groups = ["users", "wheel"]
```

2. ブループリントを RHEL Image Builder サーバーにインポートします。

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

3. 既存のブループリントをリスト表示して、作成されたブループリントが正常にインポートされて存在するかどうかを確認します。

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

4. ブループリントに記載されているコンポーネントおよびバージョンと、その依存関係が有効かどうかを確認します。

```
# composer-cli blueprints depsolve <blueprint-name>
```

5. カスタマイズした RHEL イメージをビルドします。

```
# composer-cli compose start \ <blueprint-name> \ <image-type> \
```

6. イメージのステータスを確認します。

```
# composer-cli compose status
…
$ <UUID> FINISHED <date> <blueprint-name> <blueprint-version> <image-type>
…
```

7. イメージをダウンロードします。

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

RHEL Image Builder により、イメージがカレントディレクトリーパスにダウンロードされます。UUID 番号とともにイメージサイズが表示されます。

```
$ <UUID-image-name.type>: <size> MB
```

検証

1. ブループリントで設定したユーザー名とパスワードを使用してシステムイメージにログインします。
2. FIPS モードが有効になっているかどうかを確認します。

```
$ fips-mode-setup --check
FIPS mode is enabled.
```

第9章 RHEL IMAGE BUILDER を使用した KVM ゲストイメージの準備とデプロイ

RHEL Image Builder を使用して、カーネルベースの仮想マシン (KVM) をベースとしたハイパーバイザーにデプロイできる、専用の **.qcow2** を作成します。

カスタマイズされた KVM ゲストイメージの作成には、以下の手順が含まれます。

1. **.qcow2** イメージのブループリントの作成
2. RHEL Image Builder を使用した **.qcow2** イメージの作成
3. KVM ゲストイメージからの仮想マシンの作成

9.1. RHEL IMAGE BUILDER を使用したカスタマイズされた KVM ゲストイメージの作成

RHEL Image Builder を使用して、カスタマイズした **.qcow2** KVM ゲストイメージを作成できます。以下では GUI での手順を示していますが、CLI を使用することもできます。

前提条件

- システムにアクセスするには、**root** または **weldr** グループに属している必要があります。
- **cockpit-composer** パッケージがインストールされている。
- RHEL システムで、Web コンソールの RHEL Image Builder ダッシュボードを開いている。
- ブループリントを作成している。Web コンソールインターフェイスでのブループリントの作成を参照してください。

手順

1. 作成したブループリント名をクリックします。
2. **Images** タブを選択します。
3. **Create Image** をクリックして、カスタマイズしたイメージを作成します。**Create Image** ウィンドウが開きます。
4. **Type** ドロップダウンメニューリストから、**QEMU Image (.qcow2)** を選択します。
5. インスタンス化するときのイメージのサイズを設定し、**Create** をクリックします。
6. ウィンドウの右上にある小さなポップアップに、イメージの作成がキューに追加されたことが表示されます。イメージ作成プロセスが完了すると、**Image build complete** のステータスが表示されます。

検証

- パンくずリストのアイコンをクリックして、**Download** オプションを選択します。RHEL Image Builder は、KVM ゲストイメージ **.qcow2** ファイルをデフォルトのダウンロード場所にダウンロードします。

関連情報

- Web コンソールインターフェイスでのブループリントの作成

9.2. KVM ゲストイメージからの仮想マシンの作成

RHEL Image Builder を使用すると、**.qcow2** イメージをビルドし、KVM ゲストイメージを使用して仮想マシンを作成できます。RHEL Image Builder を使用して作成した KVM ゲストイメージでは、**cloud-init** がすでにインストールされ、有効になっています。

前提条件

- RHEL Image Builder を使用して **.qcow2** イメージを作成している。Web コンソールインターフェイスでのブループリントの作成 を参照してください。
- **qemu-kvm** パッケージがシステムにインストールされている。**/dev/kvm** デバイスがシステムで使用可能かどうか、および仮想化機能が BIOS で有効になっているかどうかを確認できる。
- システムに **libvirt** および **virt-install** パッケージがインストールされている。
- **xorriso** パッケージによって提供される **genisoimage** ユーティリティーがシステムにインストールされている。

手順

1. RHEL Image Builder を使用して作成した **.qcow2** イメージを **/var/lib/libvirt/images/** ディレクトリーに移動します。
2. ディレクトリー (**cloudinitiso** など) を作成し、新規に作成したそのディレクトリーに移動します。

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

3. **meta-data** という名前のファイルを作成します。このファイルに以下の情報を追加します。

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

4. **user-data** という名前のファイルを作成します。以下の情報をファイルに追加します。

```
#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 は、SSH 公開鍵になります。 **~/.ssh/<id_rsa.pub>**\ で SSH 公開鍵を確認できます。

5. **genisoimage** ユーティリティーを使用して、**user-data** ファイルおよび **meta-data** ファイルを含む ISO イメージを作成します。

```
# 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. **virt-install** コマンドを使用して、KVM ゲストイメージから新しい仮想マシンを作成します。仮想マシンイメージへのアタッチメントとして、手順 4 で作成した ISO イメージを含めます。

```
# virt-install \
    --memory 4096 \
    --vcpus 4 \
    --name myvm \
    --disk rhel-9-x86_64-kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
    --disk cloud-init.iso,device=cdrom \
    --os-variant rhel 9 \
    --virt-type kvm \
    --graphics none \
    --import
```

   - --graphics none - ヘッドレス RHEL 9 仮想マシンであることを意味します。
   - --vcpus 4 - 4 つの仮想 CPU を使用することを意味します。
   - --memory 4096 - 4096 MB のメモリーを使用することを意味します。

7. 仮想マシンのインストールが起動します。

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

Red Hat Enterprise Linux 9 (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

検証

起動が完了すると、仮想マシンにテキストログインインターフェイスが表示されます。仮想マシンのローカルコンソールにログインするには、**user-data** ファイルのユーザーの詳細を使用します。

1. ユーザー名として **admin** と入力し、**Enter** を押します。
2. **password** としてパスワードを入力し、**Enter** を押します。
   ログイン認証が完了すると、CLI を使用して仮想マシンにアクセスできます。

関連情報

- 仮想化を有効にする
- RHEL 9 の cloud-init の設定および管理

- cloud-init の重要なディレクトリーおよびファイル

第10章 コンテナーをレジストリーにプッシュしてイメージに埋め込む

RHEL Image Builder を使用すると、OpenSCAP ツールを使用してセキュリティーが強化されたイメージをビルドできます。ブループリントでコンテナーのカスタマイズがサポートされていることを利用して、コンテナーを作成し、作成したイメージに直接埋め込むことができます。

10.1. コンテナーをイメージに埋め込むブループリントのカスタマイズ

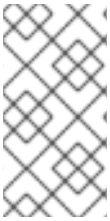
registry.access.redhat.com レジストリーからコンテナーを埋め込むには、ブループリントにコンテナーのカスタマイズを追加する必要があります。以下に例を示します。

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

- **source** - 必須フィールド。これは、レジストリーにあるコンテナーイメージへの参照です。この例では、**registry.access.redhat.com** レジストリーを使用します。タグのバージョンを指定できます。デフォルトのタグバージョンは **latest** です。
- **name** - ローカルレジストリー内のコンテナーの名前。
- **tls-verify** - ブールフィールド。**tls-verify** ブールフィールドは、Transport Layer Security を制御します。デフォルト値は **true** です。
  RHEL Image Builder は、イメージのビルド中にコンテナーをプルし、コンテナーをイメージに格納します。デフォルトのローカルコンテナーストレージの場所は、イメージの種類によって異なります。そのため、Podman などのすべてのサポート **container-tools** がそれを使用できます。組み込みコンテナーは開始されません。保護されたコンテナーリソースにアクセスするには、**containers-auth.json** ファイルを使用できます。

10.2. コンテナーレジストリーの認証情報

osbuild-worker サービスは、コンテナーレジストリーとの通信を担当します。これを有効にするには、**/etc/osbuild-worker/osbuild-worker.toml** 設定ファイルをセットアップします。



注記

/etc/osbuild-worker/osbuild-worker.toml 設定ファイルを設定したら、**osbuild-worker** サービスを再起動する必要があります。これは、**osbuild-worker** サービス開始の際に、**/etc/osbuild-worker/osbuild-worker.toml** 設定ファイルを 1 度のみ読み込むためです。

/etc/osbuild-worker/osbuild-worker.toml 設定ファイルには、保護されたリソースへのアクセスに使用される、**containers-auth.json** ファイルのパスを参照する文字列である **auth_field_path** エントリーを含む container セクションがあります。コンテナーレジストリーの認証情報は、コンテナーをイメージに埋め込むときに、コンテナーイメージをレジストリーからプルするためにのみ使用されます。

以下に例を示します。

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

関連情報

- システムの **containers-auth.json** の man ページ

10.3. コンテナーアーティファクトをコンテナーレジストリーに直接プッシュする

RHEL for Edge Container イメージなどのコンテナーアーティファクトは、RHEL Image Builder CLI を使用して、ビルド後にコンテナーレジストリーに直接プッシュできます。

前提条件

- quay.io レジストリー へのアクセス。この例では、ターゲットレジストリーとして **quay.io** コンテナーレジストリーを使用していますが、任意のコンテナーレジストリーを使用できます。

手順

1. コンテナープロバイダーを選択するために、**registry-config.toml** ファイルをセットアップします。認証情報はオプションです。

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

2. ブループリントを **.toml** 形式で作成します。これは、ブループリントに **nginx** パッケージをインストールするコンテナーのブループリントです。

```
name = "simple-container"
description = "Simple RHEL container"
version = "0.0.1"
[[packages]]
name = "nginx"
version = "*"
```

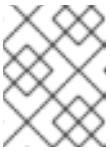
3. ブループリントをプッシュします。

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

4. レジストリーとリポジトリーを **composer-cli** ツールに引数として渡し、コンテナーイメージをビルドします。

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

   - simple-container - ブループリント名です。
   - コンテナー - イメージの種類です。
   - "quay.io:8080/osbuild/**repository**" - **quay.io** はターゲットレジストリー、**osbuild** は組織、**repository** はビルドが完了したときにコンテナーをプッシュする場所です。オプションで、**tag** を設定できます。**:tag** の値を設定しない場合、デフォルトで **:latest** タグが使用されます。



注記

コンテナーイメージのビルドには、カスタマイズされたパッケージの依存関係を解決するため時間がかかります。

5. イメージのビルドが完了すると、作成したコンテナーが quay.io で使用できるようになります。

検証

1. quay.io を開き、**Repository Tags** をクリックします。

```
 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. **manifest ID** の値をコピーして、コンテナーを埋め込むイメージをビルドします。

関連情報

- Quay.io - タグの操作

10.4. イメージのビルドとコンテナーのイメージへのプル

コンテナーイメージを作成したら、カスタマイズしたイメージをビルドし、コンテナーイメージをそこにプルできます。そのためには、ブループリントで **コンテナーのカスタマイズ** を指定し、最終的なイメージの **コンテナー名** を指定する必要があります。ビルドプロセス中に、コンテナーイメージが取得され、ローカルの Podman コンテナーストレージに配置されます。

前提条件

- コンテナーイメージを作成し、それをローカルの **quay.io** コンテナーレジストリーインスタンスにプッシュしました。コンテナーアーティファクトをコンテナーレジストリーに直接プッシュする を参照してください。
- registry.access.redhat.com にアクセスできます。
- コンテナー **manifest ID** を持っています。
- **qemu-kvm** および **qemu-img** パッケージがインストールされている。

手順

1. **qcow2** イメージをビルドするためのブループリントを作成します。このブループリントには、"" カスタマイズを含める必要があります。

```
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. ブループリントをプッシュします。

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

3. コンテナーイメージをビルドします。

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

- **image** はブループリントの名前です。
- **qcow2** はイメージタイプです。

注記

quay.io レジストリーでコンテナーをチェックするため、イメージのビルドに時間がかかります。

4. Compose のステータスを確認するには、以下のコマンドを実行します。

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

Compose が完了すると、ステータスが **FINISHED** になります。リスト内の Compose を識別するには、その UUID を使用します。

5. Compose プロセスが完了したら、作成したイメージファイルをデフォルトのダウンロード先にダウンロードします。

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

UUID は、前の手順で示した UUID 値に置き換えます。

作成してダウンロードした **qcow2** イメージを使用して、仮想マシンを作成できます。

検証

ダウンロードした **qcow2** イメージで、次の手順を実行します。

1. 仮想マシンで **qcow2** イメージを起動します。KVM ゲストイメージからの仮想マシンの作成 を参照してください。
2. **qemu** ウィザードが開きます。**qcow2** イメージにログインします。
   a. ユーザー名とパスワードを入力します。**.qcow2** ブループリントの "customizations.user" セクションで設定したユーザー名とパスワード、または **cloud-init** を使用して起動時に作成したユーザー名とパスワードを使用できます。
3. コンテナーイメージを実行し、コンテナー内でシェルプロンプトを開きます。

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

-

registry.access.redhat.com はターゲットレジストリー、**osbuild** は組織、**repository** はビルドの完了時にコンテナーをプッシュする場所です。

4. ブループリントに追加したパッケージが利用可能であることを確認します。

```
# type -a nginx
```

出力には、**nginx** パッケージのパスが表示されます。

関連情報

- Red Hat コンテナーレジストリーの認証
- Red Hat レジストリーへのアクセスおよびその設定
- 基本的な Podman コマンド
- コンテナーでの Skopeo の実行

第11章 RHEL IMAGE BUILDER を使用したクラウドイメージの準備とアップロード

RHEL Image Builder は、さまざまなクラウドプラットフォームですぐに使用できるカスタムシステムイメージを作成できます。カスタマイズした RHEL システムイメージをクラウドで使用するには、指定の出力タイプを使用して RHEL Image Builder でシステムイメージを作成し、イメージをアップロードするようにシステムを設定し、クラウドアカウントへイメージをアップロードします。RHEL Web コンソールの **Image Builder** アプリケーションを介して、カスタマイズされたイメージクラウドをプッシュできます。これは、**AWS** や **Microsoft Azure** クラウドなど、Red Hat サポート対象のサービスプロバイダーの一部で利用できます。AWS Cloud AMI に直接イメージを作成して自動的にアップロードする および Microsoft Azure クラウドに直接 VHD イメージを作成して自動的にアップロードする を参照してください。

11.1. AMI イメージの準備と AWS へのアップロード

RHEL Image Builder を使用してカスタムイメージを作成し、そのイメージを手動または自動で AWS クラウドにアップロードできます。

11.1.1. AWS AMI イメージを手動でアップロードする準備

AWS AMI イメージをアップロードする前に、イメージをアップロードするためのシステムを設定する必要があります。

前提条件

- AWS IAM アカウントマネージャー にアクセスキー ID を設定している。
- 書き込み可能な S3 バケット を準備している。

手順

1. Python 3 および **pip** ツールをインストールします。

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

2. pip で **AWS コマンドラインツール** をインストールします。

```
# pip3 install awscli
```

3. プロファイルを設定します。ターミナルで、認証情報、リージョン、および出力形式を指定するように求められます。

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

4. バケットの名前を定義し、バケットを作成します。

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

bucketname は、実際のバケット名に置き換えます。この名前は、グローバルで一意となるように指定する必要があります。上記で、バケットが作成されます。

5. S3 バケットへのアクセス許可を付与するには、AWS Identity and Access Management (IAM) で **vmimport** S3 ロールを作成します (まだ作成していない場合)。

   a. 信頼ポリシーの設定で、JSON 形式で **trust-policy.json** ファイルを作成します。以下に例を示します。

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

   b. ロールポリシーの設定を含む **role-policy.json** ファイルを JSON 形式で作成します。以下に例を示します。

```
{
   "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. **trust-policy.json** ファイルを使用して、Amazon Web Services アカウントのロールを作成します。

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

   d. **role-policy.json** ファイルを使用して、インラインポリシードキュメントを埋め込みます。

```
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-
document file://role-policy.json
```

関連情報

- Using high-level (s3) commands with the AWS CLI

11.1.2. CLI を使用して AMI イメージを AWS に手動でアップロードする

RHEL Image Builder を使用して **ami** イメージをビルドし、CLI を使用して Amazon AWS Cloud サービスプロバイダーに直接手動でアップロードすることができます。

前提条件

- AWS IAM アカウントマネージャーに **Access Key ID** を設定している。
- 書き込み可能な S3 バケット を準備している。
- 定義済みの青写真がある。

手順

1. テキストエディターを使用して、次の内容の設定ファイルを作成します。

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

   フィールドの値を **accessKeyID**、**secretAccessKey**、**bucket**、および **region** の認証情報に置き換えます。**IMAGE_KEY** 値は、EC2 にアップロードする仮想マシンイメージの名前です。

2. ファイルを **CONFIGURATION-FILE**.toml として保存し、テキストエディターを閉じます。

3. Compose を開始して AWS にアップロードします。

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

   以下を置き換えます。

   - **blueprint-name** は、作成したブループリントの名前に置き換えます。
   - **image-type** は、**ami** イメージタイプに置き換えます。
   - **image-key** は、EC2 にアップロードする仮想マシンイメージの名前に置き換えます。
   - **configuration-file**.toml は、クラウドプロバイダーの設定ファイルの名前に置き換えます。

注記

カスタマイズしたイメージの送信先となるバケットの正しい AWS Identity and Access Management (IAM) 設定が必要です。イメージをアップロードする前にバケットにポリシーを設定しておく必要があります。

4. イメージビルドのステータスを確認します。

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

イメージのアップロードプロセスが完了すると、"FINISHED" ステータスが表示されます。

検証

イメージのアップロードが成功したことを確認するには、以下を行います。

1. メニューで EC2 にアクセスし、AWS コンソールで正しいリージョンを選択します。イメージが正常にアップロードされたことを示すには、イメージが **available** ステータスになっている必要があります。
2. Dashboard でイメージを選択し、**起動** をクリックします。

関連情報

- 仮想マシンのインポートに必要なサービスロール

11.1.3. イメージを作成して AWS Cloud AMI に自動的にアップロードする

RHEL Image Builder を使用して **(.raw)** イメージを作成し、**Upload to AWS** チェックボックスをオンにして、作成した出力イメージを **Amazon AWS Cloud AMI** サービスプロバイダーに直接自動的にプッシュすることができます。

前提条件

- **root** または **wheel** グループでシステムにアクセスできる。
- ブラウザーで、RHEL Web コンソールの RHEL Image Builder インターフェイスを開いている。
- ブループリントを作成している。Web コンソールインターフェイスでのブループリントの作成を参照してください。
- AWS IAM アカウントマネージャーにアクセスキー ID を設定している。
- 書き込み可能な S3 バケット を準備している。

手順

1. RHEL Image Builder のダッシュボードで、以前に作成した **ブループリント名** をクリックします。
2. **Images** タブを選択します。
3. **Create Image** をクリックして、カスタマイズしたイメージを作成します。
   Create Image ウィンドウが開きます。
   a. **Type** ドロップダウンメニューから、**Amazon Machine Image Disk (.raw)** を選択します。
   b. イメージを AWS Cloud にアップロードするには、**Upload to AWS** チェックボックスをオンして、**Next** をクリックします。
   c. AWS へのアクセスを認証するには、対応するフィールドに **AWS access key ID** および **AWS secret access key** を入力します。**Next** をクリックします。

注記

新規アクセスキー ID を作成する場合にのみ、AWS シークレットアクセスキーを表示できます。秘密鍵が分からない場合は、新しいアクセスキー ID を生成します。

d. **Image name** フィールドにイメージ名を、**Amazon S3 bucket name** フィールドに Amazon バケット名を入力して、カスタマイズイメージを追加するバケットの **AWS region** フィールドを入力します。**Next** をクリックします。

e. 情報を確認し、**Finish** をクリックします。
必要に応じて、**Back** をクリックして誤った情報を変更します。

注記

カスタマイズイメージを送信するバケットの正しい IAM 設定が必要です。この手順では IAM のインポートとエクスポートを使用するため、バケットにイメージをアップロードする前にバケットに **ポリシー** を設定する必要があります。詳細は、IAM ユーザーの必要なパーミッション を参照してください。

4. 右上のポップアップで、保存の進行状況が通知されます。イメージ作成の開始、イメージ作成の進捗、およびその後の AWS Cloud にアップロードに関する情報も通知されます。
プロセスが完了すると、**Image build complete** のステータスが表示されます。

5. ブラウザーで、Service→EC2 にアクセスします。

   a. AWS コンソールのダッシュボードメニューで、正しいリージョン を選択します。イメージのステータスは、アップロードされていることを示す **Available** でなければなりません。

   b. AWS ダッシュボードでイメージを選択し、**Launch** をクリックします。

6. 新しいウィンドウが開きます。イメージを開始するために必要なリソースに応じて、インスタンスタイプを選択します。**Review and Launch** をクリックします。

7. インスタンスの開始の詳細を確認します。変更が必要な場合は、各セクションを編集できます。**起動** をクリックします。

8. インスタンスを起動する前に、インスタンスにアクセスするための公開鍵を選択します。
既存のキーペアを使用するか、キーペアーを新規作成します。

   次の手順に従って、EC2 で新規キーペアを作成し、新規インスタンスにアタッチします。

   a. ドロップダウンメニューリストから、**Create a new key pair**を選択します。

   b. 新しいキーペアに名前を入力します。新しいキーペアが生成されます。

   c. **Download Key Pair** をクリックして、新しいキーペアをローカルシステムに保存します。

9. **Launch Instance** をクリックしてインスタンスを起動します。
Initializing と表示されるインスタンスのステータスを確認できます。

10. インスタンスのステータスが **running** になると、**Connect** ボタンが有効になります。

11. **Connect** をクリックします。ウィンドウが表示され、SSH を使用して接続する方法の説明が表示されます。

a. 優先する接続方法として **スタンドアロン SSH クライアント** を選択し、ターミナルを開きます。

b. 秘密鍵を保存する場所で、SSH が機能するために鍵が公開されていることを確認してください。これには、以下のコマンドを実行します。

```
$ chmod 400 <_your-instance-name.pem_>
```

c. パブリック DNS を使用してインスタンスに接続します。

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

d. **yes** と入力して、接続の続行を確定します。
その結果、SSH 経由でインスタンスに接続されます。

検証

- SSH でインスタンスに接続している間にアクションが実行できるかどうかを確認します。

関連情報

- Red Hat カスタマーポータルでのケースの作成
- Connecting to your Linux instance by using SSH

11.2. VHD イメージを準備して MICROSOFT AZURE にアップロードする

RHEL Image Builder を使用すると、カスタムイメージを作成し、そのイメージを手動または自動で Microsoft Azure クラウドにアップロードできます。

11.2.1. Microsoft Azure VHD イメージを手動でアップロードする準備

Microsoft Azure クラウドに手動でアップロードできる VHD イメージを作成するには、RHEL Image Builder を使用できます。

前提条件

- Microsoft Azure リソースグループとストレージアカウントがある。
- Python がインストールされている。**AZ CLI** ツールは Python に依存しています。

手順

1. Microsoft リポジトリーキーをインポートします。

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

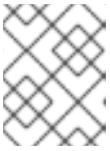
2. 次の情報を使用して、ローカルの **azure-cli.repo** リポジトリーを作成します。**azure-cli.repo** リポジトリーを **/etc/yum.repos.d/** に保存します。

```
[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. Microsoft Azure CLI をインストールします。

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



注記

Microsoft Azure CLI パッケージのダウンロードバージョンは、現在利用可能なバージョンによって異なる場合があります。

4. Microsoft Azure CLI を実行します。

```
$ az login
```

ターミナルに次のメッセージが表示されます。**Note, we have launched a browser for you to login.For old experience with device code, use "az login --use-device-code**次に、ターミナルは、ログインできる場所から https://microsoft.com/devicelogin へのリンクのあるブラウザーを開きます。

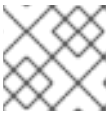
注記

リモート (SSH) セッションを実行している場合、ログインページのリンクはブラウザーで開きません。この場合、リンクをブラウザーにコピーしてログインし、リモートセッションを認証できます。サインインするには、Web ブラウザーを使用してページ https://microsoft.com/devicelogin を開き、デバイスコードを入力して認証します。

5. Microsoft Azure のストレージアカウントのキーをリスト表示します。

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

resource-group-name を Microsoft Azure リソースグループの名前に置き換え、**storage-account-name** を Microsoft Azure ストレージアカウントの名前に置き換えます。



注記

次のコマンドを使用して、使用可能なリソースを一覧表示できます。

```
$ az resource list
```

上記のコマンドの出力にある値 **key1** をメモします。

6. ストレージコンテナーを作成します。

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

storage-account-name は、ストレージアカウント名に置き換えます。

関連情報

- Microsoft Azure CLI.

11.2.2. VHD イメージを Microsoft Azure クラウドに手動でアップロードする

カスタマイズした VHD イメージを作成したら、それを手動で Microsoft Azure クラウドにアップロードできます。

前提条件

- Microsoft Azure VHD イメージをアップロードするようにシステムを設定している。Microsoft Azure VHD イメージのアップロードの準備 を参照してください。
- RHEL Image Builder で Microsoft Azure VHD イメージを作成している。
  - GUI で、**Azure Disk Image (.vhd)** イメージタイプを使用します。
  - CLI で、**vhd** 出力タイプを使用します。

注記

CLI を使用して **.vhd** イメージを作成すると、Image Builder は一時ファイルを **/var** サブディレクトリーに書き込みます。**.vhd** イメージの作成が失敗しないようにするには、**/var** サブディレクトリーの容量を少なくとも 15 から 20 GB の空き領域に増やし、可用性を確保します。

手順

1. イメージを Microsoft Azure にプッシュし、そこからインスタンスを作成します。

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

2. Microsoft Azure Blob ストレージへのアップロードが完了したら、そこから Microsoft Azure イメージを作成します。

```
$ az image create --resource-group <_resource_group_name_> --name <_image_>-disk.vhd
--os-type linux --location <_location_> --source
https://$<_account_name_>.blob.core.windows.net/<_container_name_>/<_image_>-
disk.vhd
 - Running ...
```

注記

RHEL Image Builder で作成するイメージは、V1 = BIOS と V2 = UEFI の両方のインスタンスタイプをサポートするハイブリッドイメージを生成するため、**--hyper-v-generation** 引数を指定できます。デフォルトのインスタンスタイプは V1 です。

検証

1. Microsoft Azure ポータル、または以下のようなコマンドを使用して、インスタンスを作成します。

```
$ az vm create --resource-group <_resource_group_name_> --location <_location_> --name
<_vm_name_> --image <_image_>-disk.vhd --admin-username azure-user --generate-ssh-
keys
 - Running ...
```

2. 秘密鍵を使用して、SSH 経由で、作成されたインスタンスにアクセスします。**azure-user** としてログインします。このユーザー名は前の手順で設定したものです。

関連情報

- Composing an image for the **.vhd** format fails (Red Hat ナレッジベース)

11.2.3. VHD イメージを作成して Microsoft Azure クラウドに自動的にアップロードする

RHEL Image Builder を使用して **.vhd** イメージを作成すると、Microsoft Azure クラウドサービスプロバイダーの Blob Storage に自動的にアップロードされます。

前提条件

- システムへの root アクセス権があります。
- RHEL Web コンソールの RHEL Image Builder インターフェイスにアクセスできる。
- ブループリントを作成している。Web コンソールインターフェイスでの RHEL Image Builder ブループリントの作成 を参照してください。
- Microsoft ストレージアカウント が作成されました。
- 書き込み可能な Blob Storage が準備されました。

手順

1. RHEL Image Builder ダッシュボードで、使用するブループリントを選択します。
2. **Images** タブをクリックします。
3. **Create Image** をクリックして、カスタマイズした **.vhd** イメージを作成します。
   Create image ウィザードが開きます。
   a. **Type** ドロップダウンメニューリストから **Microsoft Azure (.vhd)** を選択します。
   b. イメージを Microsoft Azure クラウドにアップロードするには、**Upload to Azure** チェックボックスをオンします。
   c. **Image Size** を入力し、**Next** をクリックします。
4. **Upload to Azure** ページで、次の情報を入力します。
   a. 認証ページで、次のように入力します。

i. **Storage account** の名前。これは、Microsoft Azure portal の **Storage account** ページにあります。

ii. **Storage access key**. これは、**Access Key** ストレージページにあります。

iii. **Next** をクリックします。

b. **Authentication** ページで、次のように入力します。

i. イメージ名

ii. **Storage container**。これは、イメージのアップロード先の Blob コンテナーです。Microsoft Azure portal の **Blob service** セクションにあります。

iii. **Next** をクリックします。

5. **Review** ページで **Create** をクリックします。RHEL Image Builder が起動し、アップロードプロセスが開始します。
Microsoft Azure Cloud にプッシュしたイメージにアクセスします。

6. Microsoft Azure ポータル にアクセスします。

7. 検索バーに "storage account" と入力し、リストから **Storage accounts** をクリックします。

8. 検索バーに "Images" と入力し、**Services** の下にある最初のエントリーを選択します。**Image Dashboard** にリダイレクトされます。

9. ナビゲーションパネルで、**Containers** をクリックします。

10. 作成したコンテナーを見つけます。コンテナー内には、RHEL Image Builder を使用して作成およびプッシュした **.vhd** ファイルがあります。

検証

1. 仮想マシンイメージを作成して起動できることを確認します。

a. 検索バーに images account と入力し、リストから **Images** をクリックします。

b. **+Create** をクリックします。

c. ドロップダウンリストから、前に使用したリソースグループを選択します。

d. イメージの名前を入力します。

e. **OS type** で **Linux** を選択します。

f. **VM generation** で **Gen 2** を選択します。

g. **Storage Blob** で **Browse** をクリックし、VHD ファイルに到達するまでストレージアカウントとコンテナーをクリックします。

h. ページの最後にある **Select** をクリックします。

i. Account Type を選択します (例: **Standard SSD**)。

j. **Review + Create** をクリックし、**Create** をクリックします。イメージが作成されるまでしばらく待機します。

2. 仮想マシンを起動するには、次の手順に従います。

   a. **Go to resource** をクリックします。

   b. ヘッダーのメニューバーから **+ Create VM** をクリックします。

   c. 仮想マシンの名前を入力します。

   d. **Size** セクションと **Administrator account** セクションに入力します。

   e. **Review + Create** をクリックし、**Create** をクリックします。デプロイメントの進行状況を確認できます。
   デプロイメントが完了したら、仮想マシン名をクリックしてインスタンスのパブリック IP アドレスを取得し、SSH を使用して接続します。

   f. ターミナルを開いて SSH 接続を作成し、仮想マシンに接続します。

関連情報

- Microsoft Azure ストレージのドキュメント
- Microsoft Azure ストレージアカウントの作成
- Red Hat カスタマーポータルでのケースの作成
- Help + support
- Red Hat サポートへの問い合わせ

11.3. VMDK カスタムイメージの準備と VSPHERE へのアップロード

RHEL Image Builder を使用してカスタムイメージを作成し、そのイメージを手動または自動で VMware vSphere クラウドにアップロードできます。

11.3.1. Image Builder を使用してカスタマイズした RHEL VMDK イメージを作成し、自動的にアップロードする

RHEL Image Builder を使用すると、Open virtualization format (**.ova**) 形式のカスタマイズしたシステムイメージを作成し、そのイメージを VMware vSphere クライアントに自動的にアップロードできます。Open virtualization format (**.ova**) は、仮想ハードウェアに関する追加のメタデータを含む **.vmdk** イメージです。このイメージには、vSphere へのイメージのインポートを容易にする最小限のテンプレートが含まれています。vSphere **.ova** イメージには、**.ovf** (Open Virtualization Format) パッケージが含まれています。RHEL Image Builder による vSphere クライアントへの **.ova** イメージのインポートを完了したら、そのイメージをネットワーク、ディスク、CD-ROM などの追加ハードウェアを使用して設定できます。

Open virtualization format (**.ova**) イメージは、vSphere GUI または **govc** クライアントを使用してインポートできます。**govc** クライアントを使用してイメージをアップロードするには、VMDK イメージのアップロードと vSphere での RHEL 仮想マシンの作成 を参照してください。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- ブループリントを作成している。

手順

1. RHEL Image Builder ダッシュボードで、**Blueprint** タブをクリックします。
2. ブループリントのテーブルで、イメージをビルドするブループリントを見つけます。
3. 選択したブループリントの右側で、**Create Image** をクリックします。**Create image** ダイアログウィザードが開きます。
4. **Image output** ページで、次の手順を実行します。
   a. **Select a blueprint** リストから、必要なイメージのタイプを選択します。
   b. **Image output type** リストから、必要なイメージの出力タイプを選択します。
   c. オプション: イメージを VMware に直接アップロードするには、**Upload to VMware** チェックボックスをオンにします。
   d. イメージのサイズを入力します。
   e. **Next** をクリックします。
5. **Upload to VMware** ページで、次の情報を入力します。
   a. **Image name**: イメージ名を入力します。
   b. **Host**: イメージファイルをアップロードする VMware vSphere インスタンスの URL を入力します。
   c. **Cluster**: イメージをアップロードするクラスター名のページを入力します。
   d. **Datacenter**: イメージをアップロードするデータセンター名。
   e. **Datastore**: イメージをアップロードするデータストア名。
   f. **Folder**: イメージをアップロードするフォルダー名。
   g. **Next** をクリックします。
6. **Review** ページで、イメージの作成に関する詳細を確認し、**Create** をクリックします。
   イメージの作成が開始し、このイメージ作成の進捗が表示されます。その後、VMware vSphere クライアントへのアップロードが行われます。

11.3.2. VMDK イメージのアップロードと vSphere での RHEL 仮想マシンの作成

RHEL Image Builder を使用すると、カスタマイズした VMware vSphere システムイメージを Open virtualization format (**.ova**) または Virtual disk (**.vmdk**) 形式で作成できます。これらのイメージを VMware vSphere クライアントにアップロードできます。**govc import.vmdk** CLI ツールを使用して、**.vmdk** または **.ova** イメージを VMware vSphere にアップロードできます。作成した **vmdk** には、インストール済みの **cloud-init** パッケージが含まれています。このパッケージを使用して、たとえばユーザーデータを使用してユーザーをプロビジョニングできます。

注記

VMware vSphere GUI を使用した **vmdk** イメージのアップロードはサポートされていません。

前提条件

- ユーザー名とパスワードをカスタマイズしたブループリントを作成している。
- RHEL Image Builder を使用して VMware vSphere イメージを **.ova** または **.vmdk** 形式で作成し、ホストシステムにダウンロードしている。
- **govc** CLI ツールをインストールして設定し、**import.vmdk** コマンドが使用可能である。

手順

1. GOVC 環境変数を使用して、ユーザー環境で次の値を設定します。

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

2. VMware vSphere イメージをダウンロードしたディレクトリーに移動します。
3. 次の手順に従って、vSphere で VMware vSphere イメージを起動します。
   a. VMware vSphere イメージを vSphere にインポートします。

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

   .ova 形式の場合:

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

   b. 電源をオンにせずに vSphere に仮想マシンを作成します。

```
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
```

   .ova 形式の場合は、行 **-firmware=efi -disk="foldername/composer-api.vmdk"** \ を `-firmware=efi -disk="**foldername**/composer-api.ova" \ に置き換えます。

   c. 仮想マシンの電源をオンにします。

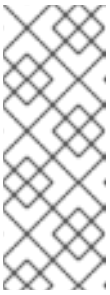
```
govc vm.power -on vmname
```

   d. 仮想マシンの IP アドレスを取得します。

```
govc vm.ip vmname
```

   e. ブループリントで指定したユーザー名とパスワードで、SSH を使用して、仮想マシンにログインします。

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



注記

govc datastore.upload コマンドを使用してローカルホストから宛先に **.vmdk** イメージをコピーしても、コピーして作成したイメージを使用することはできません。vSphere GUI には **import.vmdk** コマンドを使用するオプションがないため、vSphere GUI は直接アップロードをサポートしません。そのため、**.vmdk** イメージを vSphere GUI から使用することはできません。

11.3.3. Image Builder GUI を使用して VMDK イメージを作成し、vSphere に自動的にアップロードする

RHEL Image Builder GUI ツールを使用して VMware イメージをビルドし、そのイメージを vSphere インスタンスに直接自動的にプッシュできます。これにより、イメージファイルをダウンロードして手動でプッシュする必要がなくなります。作成した **vmdk** には、インストール済みの **cloud-init** パッケージが含まれています。このパッケージを使用して、たとえばユーザーデータを使用してユーザーをプロビジョニングできます。RHEL Image Builder を使用して **.vmdk** イメージをビルドし、vSphere インスタンスサービスプロバイダーに直接プッシュするには、次の手順に従います。

前提条件

- **root** または **weldr** グループのメンバーである。
- ブラウザーで https://localhost:9090/RHEL Image Builder を開いている。
- ブループリントを作成している。Web コンソールインターフェイスでの RHEL Image Builder ブループリントの作成 を参照してください。
- vSphere アカウント がある。

手順

1. 作成したブループリントの **Images** タブをクリックします。
2. **Create Image** をクリックして、カスタマイズしたイメージを作成します。
   イメージタイプウィンドウが開きます。
3. **Image type** ウィンドウで、以下を実行します。
   a. ドロップダウンメニューからタイプ (VMware vSphere (.vmdk)) を選択します。
   b. **Upload to VMware** チェックボックスをチェックして、イメージを vSphere にアップロードします。
   c. 必要に応じて、インスタンス化するイメージのサイズを設定します。最小のデフォルトサイズは 2 GB です。
   d. **Next** をクリックします。
4. **Upload to VMware** ウィンドウの **Authentication** の下に以下の情報を入力します。
   a. **ユーザー名**: vSphere アカウントのユーザー名。

b. **パスワード**: vSphere アカウントのパスワード。

5. **Upload to VMware** ウィンドウの **Destination** の下に、イメージのアップロード先に関する以下の情報を入力します。

   a. **Image name**: イメージの名前。

   b. **Host**: VMware vSphere の URL。

   c. **Cluster**: クラスターの名前。

   d. **Data center**: データセンターの名前。

   e. **Data store**: データストアの名前。

   f. **Next** をクリックします。

6. **確認** ウィンドウで、イメージ作成の詳細を確認し、**Finish** をクリックします。
   Back をクリックして、誤った情報を変更できます。

   RHEL Image Builder は、RHEL vSphere イメージの Compose をキューに追加し、指定した vSphere インスタンスのクラスターにイメージを作成してアップロードします。

注記

イメージビルドおよびアップロードプロセスの完了には数分かかります。

プロセスが完了すると、**Image build complete** のステータスが表示されます。

検証

イメージステータスのアップロードが正常に完了したら、アップロードしたイメージから仮想マシン (VM) を作成し、ログインできます。改善点を報告する場合は、以下のように行います。

1. VMware vSphere クライアントにアクセスします。

2. 指定した vSphere インスタンスのクラスターでイメージを検索します。

3. アップロードしたイメージを選択します。

4. 選択したイメージを右クリックします。

5. **New Virtual Machine** をクリックします。
   New Virtual Machine ウィンドウが開きます。

   New Virtual Machine ウィンドウで、以下の詳細を指定します。

   a. **New Virtual Machine** を選択します。

   b. 仮想マシンの名前とフォルダーを選択します。

   c. コンピューターリソースの選択: この操作の宛先コンピューターリソースを選択します

   d. ストレージの選択: たとえば NFS-Node1 を選択します。

   e. 互換性の選択: イメージは BIOS 専用でなければなりません。

f. ゲストオペレーティングシステムを選択します。たとえば、**Linux** および **Red Hat Fedora (64-bit)** を選択します。

g. **ハードウェアのカスタマイズ**: 仮想マシンを作成する場合は、右上の **Device Configuration** ボタンでデフォルトの New Hard Disk を削除し、ドロップダウンを使用して Existing Hard Disk ディスクイメージを選択します。

h. 準備完了: 詳細を確認し、**Finish** をクリックしてイメージを作成します。

6. **VMs** タブに移動します。

a. リストから、作成した仮想マシンを選択します。

b. パネルから **Start** ボタンをクリックします。仮想マシンイメージを読み込み中であることを示す新しいウィンドウが表示されます。

c. ブループリント用に作成した認証情報を使用してログインします。

d. ブループリントに追加したパッケージがインストールされていることを確認できます。以下に例を示します。

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

関連情報

- Introduction to vSphere Installation and Setup

11.4. カスタム GCE イメージの準備と GCP へのアップロードする

RHEL Image Builder を使用してカスタムイメージを作成し、そのイメージを Oracle Cloud Infrastructure (OCI) インスタンスに自動的にアップロードできます。

11.4.1. RHEL Image Builder を使用した GCP へのイメージのアップロード

RHEL Image Builder を使用すると、**gce** イメージをビルドし、ユーザーまたは GCP サービスアカウントの認証情報を指定して、**gce** イメージを GCP 環境に直接アップロードできます。

11.4.1.1. CLI を使用して gce イメージを設定して GCP にアップロードする

RHEL Image Builder CLI を使用して、**gce** イメージを GCP にアップロードするための認証情報を含む設定ファイルを設定します。

警告

イメージが起動しなくなるため、**gce** イメージを GCP に手動でインポートすることはできません。アップロードするには、**gcloud** または RHEL Image Builder を使用する必要があります。

前提条件

- イメージを GCP にアップロードするための有効な Google アカウントと認証情報がある。認証情報は、ユーザーアカウントまたはサービスアカウントから取得できます。認証情報に関連付けられたアカウントには、少なくとも次の IAM ロールが割り当てられている必要があります。
  - **roles/storage.admin** - ストレージオブジェクトの作成と削除
  - **roles/compute.storageAdmin** - 仮想マシンイメージの Compute Engine へのインポート
- 既存の GCP バケットがあります。

手順

1. テキストエディターを使用して、次の内容の **gcp-config.toml** 設定ファイルを作成します。

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

   - **GCP_BUCKET** は既存のバケットを指します。アップロード中のイメージの中間ストレージオブジェクトを格納するために使用されます。
   - **GCP_STORAGE_REGION** は、通常の Google ストレージリージョンであり、デュアルまたはマルチリージョンです。
   - **OBJECT_KEY** は、中間ストレージオブジェクトの名前です。アップロード前に存在してはならず、アップロードプロセスが完了すると削除されます。オブジェクト名が **.tar.gz** で終わらない場合、拡張子がオブジェクト名に自動的に追加されます。
   - **GCP_CREDENTIALS** は、GCP からダウンロードした認証情報 JSON ファイルの **Base64** エンコードされたスキームです。認証情報によって、GCP がイメージをアップロードするプロジェクトが決まります。

注記

GCP での認証に別のメカニズムを使用する場合、**gcp-config.toml** ファイルでの **GCP_CREDENTIALS** の指定は任意です。他の認証方法は、Authenticating with GCP を参照してください。

2. GCP からダウンロードした JSON ファイルから **GCP_CREDENTIALS** を取得します。

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

3. 追加のイメージ名とクラウドプロバイダープロファイルを使用して Compose を作成します。

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

   イメージビルド、アップロード、およびクラウド登録プロセスは、完了に最大 10 分かかる場合があります。

検証

- イメージのステータスが FINISHED であることを確認します。

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

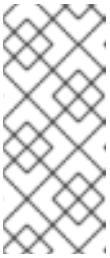
関連情報

- Identity and Access Management
- Create storage buckets

11.4.1.2. RHEL Image Builder によるさまざまな GCP 認証情報の認証順序の並べ替え

RHEL Image Builder でいくつかの異なる種類の認証情報を使用して、GCP で認証できます。複数の認証情報セットを使用して GCP で認証するように RHEL Image Builder が設定されている場合、次の優先順位で認証情報が使用されます。

1. 設定ファイルで **composer-cli** コマンドで指定された認証情報。
2. **osbuild-composer** ワーカー設定で設定された認証情報。
3. 次の方法で認証方法を自動的に見つけようとする、**Google GCP SDK** ライブラリーからの **Application Default Credentials**。
   a. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数が設定されている場合、Application Default Credentials は、変数が指すファイルから認証情報を読み込んで使用しようとします。
   b. Application Default Credentials は、コードを実行しているリソースに関連付けられたサービスアカウントを使用して認証を試みます。たとえば、Google Compute Engine 仮想マシンです。



注記

イメージをアップロードする GCP プロジェクトを決定するには、GCP 認証情報を使用する必要があります。したがって、すべてのイメージを同じ GCP プロジェクトにアップロードする場合を除き、**composer-cli** コマンドを使用して **gcp-config.toml** 設定ファイルに認証情報を指定する必要があります。

11.4.1.2.1. composer-cli コマンドで GCP 認証情報を指定する

アップロードターゲット設定の **gcp-config.toml** ファイルで、GCP 認証情報を指定できます。時間を節約するために、Google アカウント認証情報の JSON ファイルの **Base64** エンコードスキームを使用します。

手順

1. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数に保存されているパスを使用して、Google アカウント認証情報ファイルのエンコードされたコンテンツを取得するには、次のコマンドを実行します。

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

2. アップロードターゲット設定の **gcp-config.toml** ファイルで、認証情報を設定します。

```
provider = "gcp"
```

```
[settings]
provider = "gcp"

[settings]
credentials = "GCP_CREDENTIALS"
```

11.4.1.2.2. osbuild-composer ワーカー設定で認証情報を指定する

すべてのイメージビルドでグローバルに GCP に使用される GCP 認証認証情報を設定できます。このようにして、イメージを同じ GCP プロジェクトにインポートする場合、GCP へのすべてのイメージのアップロードに同じ認証情報を使用できます。

手順

- **/etc/osbuild-worker/osbuild-worker.toml** ワーカー設定で、次の認証情報の値を設定します。

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

11.5. カスタムイメージの準備と OCI への直接アップロード

RHEL Image Builder を使用してカスタムイメージを作成し、そのイメージを Oracle Cloud Infrastructure (OCI) インスタンスに自動的にアップロードできます。

11.5.1. カスタムイメージを作成して OCI に自動的にアップロードする

RHEL Image Builder を使用すると、カスタマイズしたイメージをビルドし、そのイメージを Oracle Cloud Infrastructure (OCI) インスタンスに直接自動的にプッシュできます。その後、OCI ダッシュボードからイメージインスタンスを開始できます。

前提条件

- システムに対して **root** または **weldr** グループのユーザーアクセスがある。
- Oracle Cloud アカウントを持っている。
- 管理者によって **OCI ポリシー** でセキュリティーアクセスが許可されている必要があります。
- 選択した **OCI_REGION** に OCI バケットを作成しました。

手順

1. ブラウザーで Web コンソールの RHEL Image Builder インターフェイスを開きます。
2. **Create blueprint** をクリックします。**Create blueprint** ウィザードが開きます。
3. **Details** ページで、ブループリントの名前を入力し、必要に応じて説明を入力します。**Next** をクリックします。
4. **Packages** ページで、イメージに含めるコンポーネントとパッケージを選択します。**Next** をクリックします。

5. **Customizations** ページで、ブループリントに必要なカスタマイズを設定します。**Next** をクリックします。
6. **Review** ページで **Create** をクリックします。
7. イメージを作成するには、**Create Image** をクリックします。Create image ウィザードが開きます。
8. **Image output** ページで、次の手順を実行します。
    a. **"Select a blueprint"** ドロップダウンメニューから、必要なブループリントを選択します。
    b. **"Image output type"** ドロップダウンメニューから、**Oracle Cloud Infrastructure (.qcow2)** を選択します。
    c. イメージを OCI にアップロードするには、**Upload OCI** チェックボックスをオンにします。
    d. **"image size"** を入力します。**Next** をクリックします。
9. **Upload to OCI - Authentication** ページで、次の必須の詳細を入力します。
    a. ユーザー OCID: ユーザーの詳細を表示するページのコンソールで確認できます。
    b. 秘密鍵
10. **Upload to OCI - Destination** ページで、次の必須の詳細を入力し、**Next** をクリックします。
    a. イメージ名: アップロードするイメージの名前。
    b. OCI バケット
    c. バケット namespace
    d. バケットリージョン
    e. バケットコンパートメント
    f. バケットテナンシー
11. ウィザードの詳細を確認し、**Finish** をクリックします。

RHEL Image Builder が、RHEL **.qcow2** イメージの Compose をキューに追加します。

検証

1. OCI ダッシュボード → カスタムイメージにアクセスします。
2. イメージに指定した **Compartment** を選択し、**Import image** テーブルでイメージを見つけます。
3. イメージ名をクリックして、イメージ情報を確認します。

関連情報

- OCI でのカスタムイメージの管理
- OCI でのバケットの管理

- SSH 鍵の生成

11.6. カスタマイズした QCOW2 イメージを準備して OPENSTACK に直接アップロードする

RHEL Image Builder を使用してカスタムの **.qcow2** イメージを作成し、OpenStack クラウドデプロイメントに手動でアップロードできます。

11.6.1. OpenStack への QCOW2 イメージのアップロード

RHEL Image Builder ツールを使用すると、OpenStack クラウドデプロイメントにアップロードし、そこでインスタンスを起動するのに適した、カスタマイズした **.qcow2** イメージを作成できます。RHEL Image Builder は QCOW2 フォーマットでイメージを作成しますが、OpenStack に固有の変更がさらに加えられています。

警告

RHEL Image Builder を OpenStack イメージタイプで使用して作成する一般的な **QCOW2** イメージタイプの出力フォーマットを間違えないでください。これも QCOW2 フォーマットですが、OpenStack に固有の変更がさらに含まれています。

前提条件

- ブループリントを作成している。

手順

1. **QCOW2** イメージの作成を開始します。

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

2. ビルドの状態を確認します。

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

   イメージのビルドが完了したら、イメージをダウンロードできます。

3. **QCOW2** イメージをダウンロードします。

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

4. OpenStack ダッシュボードにアクセスし、**+Create Image** をクリックします。
5. 左側のメニューで、**Admin** タブを選択します。
6. **System Panel** から **Image** をクリックします。
   Create An Image ウィザードが開きます。

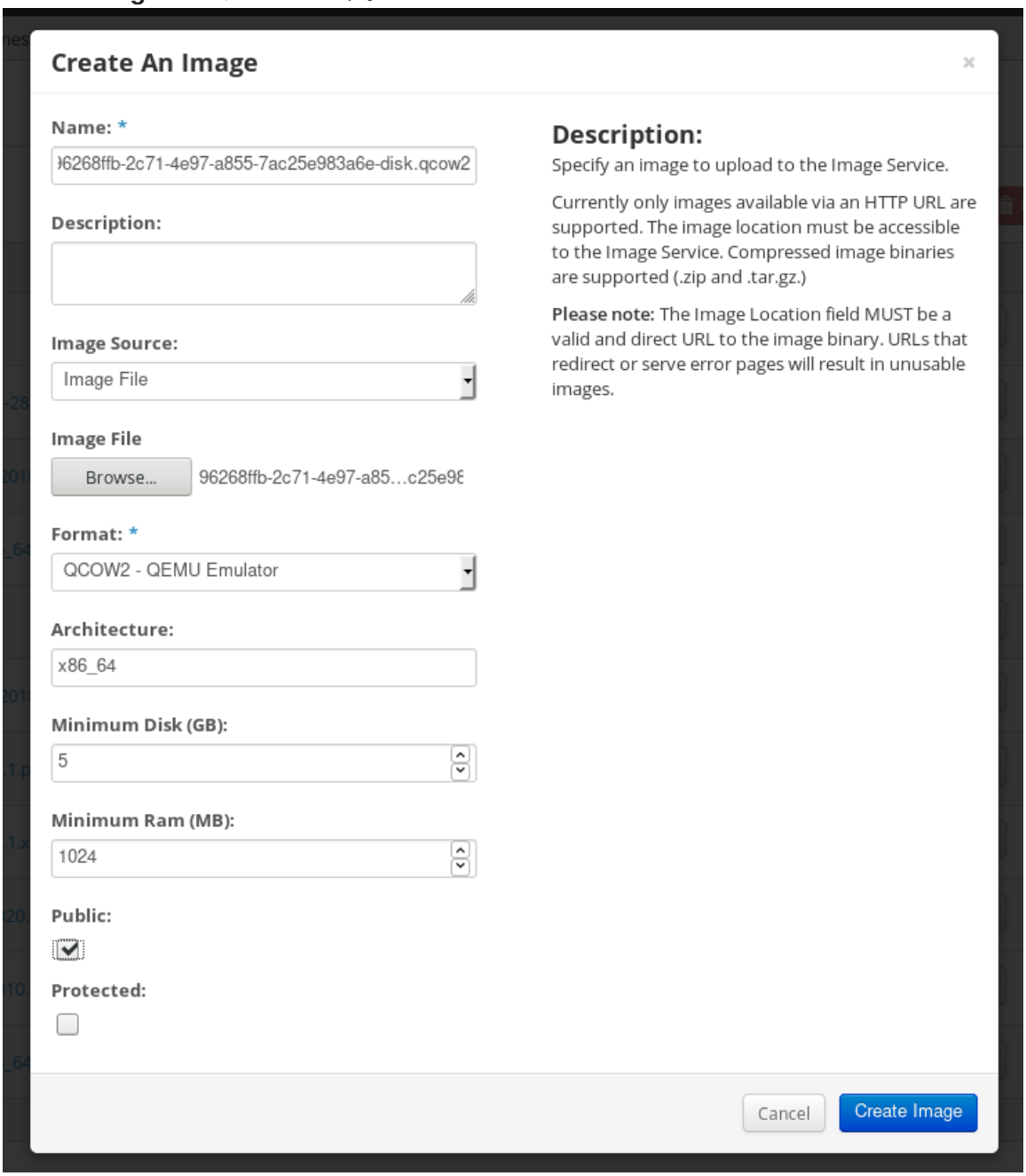
7. **Create An Image** ウィザードで、以下を行います。

   a. イメージの名前を入力します。

   b. **Browse** をクリックして **QCOW2** イメージをアップロードします。

   c. **Format** ドロップダウンリストから、**QCOW2 - QEMU Emulator** を選択します。

   d. **Create Image** をクリックします。



8. 左側のメニューで **Project** タブを選択します。

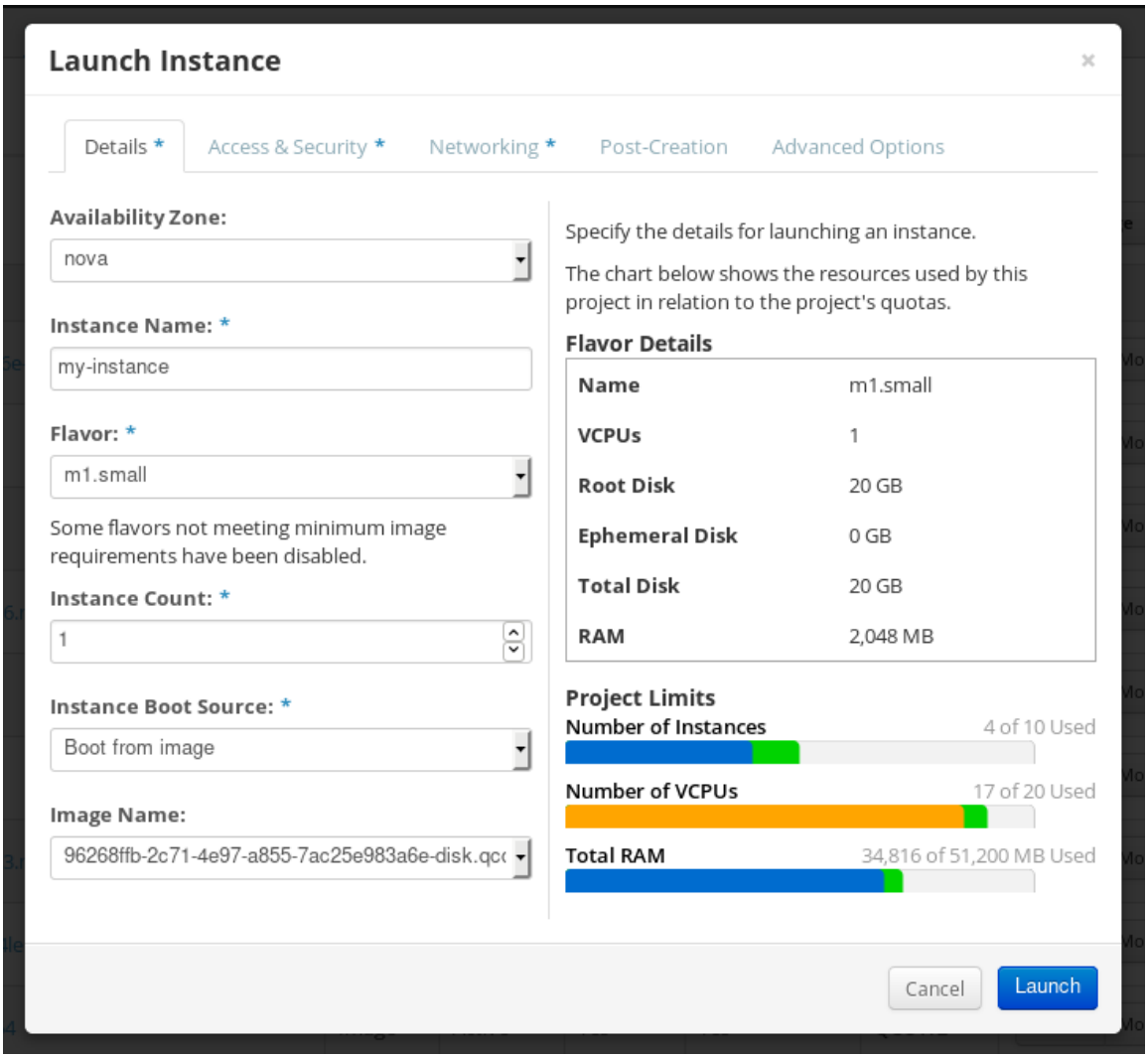
   a. **Compute** メニューから **Instances** を選択します。

   b. **Launch Instance** ボタンをクリックします。
   インスタンスの **Launch Instance** が開きます。

   c. **Details** ページで、インスタンスの名前を入力します。**Next** をクリックします。

d. **Source** ページで、アップロードしたイメージの名前を選択します。**Next** をクリックします。

e. **Flavor** ページで、ニーズに最適なマシンリソースを選択します。**Launch** をクリックします。



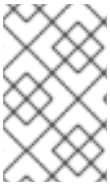
9. イメージから任意のメカニズム (CLI または OpenStack Web UI) を使用して、イメージインスタンスを実行できます。秘密鍵を使用して、SSH 経由で、作成されたインスタンスにアクセスします。**cloud-user** としてログインします。

11.7. カスタマイズした RHEL イメージを準備して ALIBABA CLOUD にアップロードする

RHEL Image Builder で作成した、カスタマイズしれた **.ami** イメージを Alibaba Cloud にアップロードできます。

11.7.1. カスタマイズされた RHEL イメージを Alibaba Cloud にアップロードする準備

カスタマイズされた RHEL イメージを Alibaba Cloud にデプロイするには、まずカスタマイズされたイメージを検証する必要があります。Alibaba Cloud は、イメージを使用する前に特定の要件を満たすようにカスタムイメージを要求するため、イメージが正常に起動するように特別な設定が必要になります。



注記

RHEL Image Builder は、Alibaba の要件に準拠したイメージを生成します。ただし、Red Hat は、Alibaba **image_check ツール** を使用して、イメージのフォーマット準拠を確認することも推奨します。

前提条件

- RHEL Image Builder を使用して Alibaba イメージを作成している。

手順

1. Alibaba の **image_check** ツールを使用して、チェックするイメージを含むシステムに接続します。
2. **image_check** ツールをダウンロードします。

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

3. イメージのコンプライアンスツールのファイルパーミッションを変更します。

```
# chmod +x image_check
```

4. 次のコマンドを実行して、イメージコンプライアンスツールのチェックを起動します。

```
# ./image_check
```

このツールは、システム設定を検証し、画面に表示されるレポートを生成します。**image_check** ツールは、イメージのコンプライアンスツールが実行されているフォルダーにこのレポートを保存します。

トラブルシューティング

いずれかの **検出項目** が失敗した場合は、ターミナルの指示に従って修正してください。

関連情報

- Image Compliance Tool

11.7.2. カスタマイズされた RHEL イメージを Alibaba にアップロードする

RHEL Image Builder で作成した、カスタマイズした **AMI** イメージを Object Storage Service (OSS) にアップロードできます。

前提条件

- Alibaba イメージのアップロードを設定している。Alibaba にイメージをアップロードするための準備 を参照してください。
- RHEL Image Builder を使用して **ami** イメージを作成している。
- バケットがある。Creating a bucket を参照してください。
- アクティブな Alibaba アカウント がある。

- OSS をアクティベートしている。

手順

1. OSS コンソール にログインします。
2. 左側のバケットメニューで、イメージをアップロードするバケットを選択します。
3. 右上のメニューで、**Files** タブをクリックします。
4. **Upload** をクリックします。右側のダイアログウィンドウが開きます。以下を設定します。
   - **アップロード先** - これを選択すると、**現在** のディレクトリーまたは **指定した** ディレクトリーにファイルをアップロードします。
   - **ファイル ACL** - アップロードしたファイルのパーミッションのタイプを選択します。
5. **Upload** をクリックします。
6. OSS コンソールにアップロードするイメージを選択します。
7. **Open** をクリックします。

関連情報

- Upload an object
- Creating an instance from custom images
- Importing images

11.7.3. Alibaba Cloud へのイメージのインポート

RHEL Image Builder で作成した、カスタマイズした Alibaba RHEL イメージを Elastic Compute Service (ECS) にインポートするには、次の手順に従います。

前提条件

- Alibaba イメージのアップロードを設定している。Alibaba にイメージをアップロードするための準備 を参照してください。
- RHEL Image Builder を使用して **ami** イメージを作成している。
- バケットがある。Creating a bucket を参照してください。
- アクティブな Alibaba アカウント がある。
- OSS をアクティベートしている。
- イメージを OSS (Object Storage Service) にアップロードしている。Alibaba へのイメージのアップロード を参照してください。

手順

1. ECS コンソール にログインします。

i. 左側のメニューで、**images** をクリックします。

ii. 右上にある **Import Image** をクリックします。ダイアログウィンドウが開きます。

iii. イメージが含まれる正しいリージョンを設定していることを確認します。以下の情報を入力します。

a. **OSS Object Address** - OSS Object Address を参照

b. **Image Name**

c. **オペレーティングシステム**

d. **System Disk Size**

e. **システムアーキテクチャー**

f. **Platform**: Red Hat

iv. オプション: 次の詳細を入力します。

g. **Image Format** - アップロードしたイメージの形式に応じて **qcow2** または **ami**。

h. **Image Description**

i. **Add Images of Data Disks**
アドレスは、OSS 管理コンソールで確認できます。左側のメニューで必要なバケットを選択した後:

2. **Files** セクションを選択します。

3. 適切なイメージの右側にある **Details** リンクをクリックします。
画面右側にウィンドウが表示され、イメージの詳細が表示されます。**OSS** オブジェクトアドレスは **URL** ボックスにあります。

4. **OK** をクリックします。

注記

インポートプロセスの時間は、イメージのサイズによって異なります。

カスタマイズされたイメージが **ECS** コンソールにインポートされます。

関連情報

- Notes for importing images
- Creating an instance from custom images
- Upload an object

11.7.4. Alibaba Cloud を使用したカスタマイズされた RHEL イメージのインスタンスの作成

Alibaba ECS コンソールを使用して、カスタマイズされた RHEL イメージのインスタンスを作成できます。

前提条件

- OSS をアクティベートして、カスタムイメージをアップロードしている。
- イメージを ECS コンソールに正常にインポートしている。Alibaba へのイメージのインポートを参照してください。

手順

1. ECS コンソール にログインします。
2. 左側のメニューで、**インスタンス** を選択します。
3. 右上隅にある **インスタンスの作成** をクリックします。新しいウィンドウにリダイレクトされます。
4. 必要な情報をすべて完了します。詳細は、Creating an instance by using the wizard を参照してください。
5. **Create Instance** をクリックして、順番を確認します。

注記

サブスクリプションによっては、**Create Instance** ではなく **Create Order** が表示されます。

その結果、アクティブなインスタンスを **Alibaba ECS Console** からデプロイする準備が整いました。

関連情報

- Creating an instance by using a custom image
- Create an instance by using the wizard