Red Hat Enterprise Linux 8

RHEL システムイメージのカスタマイズ

Red Hat Enterprise Linux 8 の Image Builder でシステムイメージのカスタマイズ

Last Updated: 2020-09-04

Red Hat Enterprise Linux 8 RHEL システムイメージのカスタマイズ

Red Hat Enterprise Linux 8 の Image Builder でシステムイメージのカスタマイズ

法律上の通知

概要

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。Image Builder を使用すると、各出力タイプの詳細を抽象化するため、手動でイメージを作成するよりも時間が短縮できます。本書は、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) 3

第1章 IMAGE BUILDER の説明 4
1.1. IMAGE BUILDER の概要 4
1.2. IMAGE BUILDER の用語 4
1.3. IMAGE BUILDER の出力形式 4
1.4. IMAGE BUILDER のシステム要件 5

第2章 IMAGE BUILDER のインストール 6
2.1. 仮想マシンへの IMAGE BUILDER のインストール 6

第3章 IMAGE BUILDER コマンドラインインターフェースでシステムイメージの作成 7
3.1. IMAGE BUILDER コマンドラインインターフェース 7
3.2. コマンドラインインターフェースで IMAGE BUILDER の BLUEPRINT の作成 7
3.3. コマンドラインインターフェースで IMAGE BUILDER の BLUEPRINT の編集 8
3.4. IMAGE BUILDER コマンドラインインターフェースでシステムイメージの作成 9
3.5. IMAGE BUILDER コマンドラインの基本的なコマンド 10
3.6. IMAGE BUILDER の BLUEPRINT 形式 12
3.7. サポートされているイメージのカスタマイズ 13
3.8. インストール済みパッケージ 16
3.9. 有効なサービス 17
3.10. IMAGE BUILDER を使用したディスクおよびパーティション設定 18

第4章 IMAGE BUILDER WEB コンソールインターフェースでシステムイメージの作成 19
4.1. RHEL 8 WEB コンソールで IMAGE BUILDER GUI へのアクセス 19
4.2. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT の作成 19
4.3. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT の編集 20
4.4. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT にユーザーおよびグループを追加 22
4.5. WEB コンソールインターフェースで IMAGE BUILDER を使用したシステムイメージの作成 24
4.6. BLUEPRINT へのソースの追加 25
4.7. BLUEPRINT のユーザーアカウントの作成 26
4.8. SSH 鍵を持つユーザーアカウントの作成 28

第5章 IMAGE BUILDER でクラウドイメージの準備およびアップロード 31
5.1. AWS AMI イメージのアップロードの準備 31
5.2. AWS への AMI イメージのアップロード 32
5.3. AZURE VHD イメージのアップロードの準備 34
5.4. VHD イメージの AZURE へのアップロード 35
5.5. VMDK イメージの VSPHERE へのアップロード 36
5.6. QCOW2 イメージの OPENSTACK へのアップロード 38
5.7. AZURE VHD イメージのアップロードの準備 40
5.8. ALIBABA へのイメージのアップロード 41
5.9. イメージの ALIBABA へのインポート 42
5.10. ALIBABA を使用したカスタムイメージのインスタンスの作成 43

RED HAT ドキュメントへのフィードバック (英語のみ)

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。改善点を報告する場合は、以下のように行います。

- 特定の文章に簡単なコメントを記入する場合は、以下の手順を行います。
    1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上端に **Feedback** ボタンがあることを確認してください。
    2. マウスカーソルで、コメントを追加する部分を強調表示します。
    3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
    4. 表示される手順に従ってください。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
    1. Bugzilla の Web サイトにアクセスします。
    2. Component で **Documentation** を選択します。
    3. **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記入してください。
    4. **Submit Bug** をクリックします。

第1章 IMAGE BUILDER の説明

1.1. IMAGE BUILDER の概要

Image Builder を使用して、Red Hat Enterprise Linux システムイメージをカスタマイズできます。たとえば、クラウドプラットフォームへのデプロイに使用するシステムイメージを作成できます。Image Builder は、出力の各タイプに対する設定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、作業も速くなります。**composer-cli** ツールのコマンドラインインターフェースで Image Builder 機能、または RHEL 8 Web コンソールでグラフィカルインターフェースにアクセスできます。

Image Builder は、システムサービスの **lorax-composer** として実行します。このサービスは、以下の 2 つのインターフェースを介してこのサービスを利用できます。

- 端末でコマンドを実行する CLI ツールの **composer-cli**。これは、推奨される方法です。
- RHEL 8 Web コンソールの GUI プラグイン。

1.2. IMAGE BUILDER の用語

Blueprint

Blueprint は、システムに追加されるパッケージおよびカスタマイズの一覧を表示して、カスタマイズされたシステムのイメージを定義します。Blueprint は編集でき、バージョン管理が行われています。Blueprint からシステムイメージを作成すると、イメージは、RHEL 8 Web コンソールの Image Builder インターフェースにある Blueprint に関連付けられます。
Blueprint は、TOML (Tom's Obvious, Minimal Language) 形式の平文テキストとして表示されます。

Compose

Compose は、特定の Blueprint の特定のバージョンに基づくシステムイメージの個々のビルドです。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

カスタマイズはシステムの仕様で、パッケージではありません。これには、ユーザー、グループ、および SSH 鍵が含まれます。

1.3. IMAGE BUILDER の出力形式

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

表1.1 Image Builder の出力形式

説明	CLI 名	ファイル拡張子
QEMU QCOW2 イメージ	**qcow2**	**.qcow2**
Ext4 ファイルシステムイメージ	**ext4-filesystem**	**.img**
raw パーティション設定ディスクイメージ	**partitioned-disk**	**.img**
ライブ起動可能 ISO	**live-iso**	**.iso**

説明	CLI 名	ファイル拡張子
tar アーカイブ	**tar**	**.tar**
Amazon Machine Image ディスクの作成	**ami**	**.ami**
Azure ディスクイメージ	**vhd**	**.vhd**
VMware 仮想マシンディスク	**vmdk**	**.vmdk**
Openstack	**openstack**	**.qcow2**

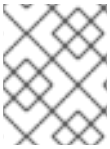
1.4. IMAGE BUILDER のシステム要件

Image Builder の基盤になっている **lorax** ツールは、システムイメージを作成している間に、潜在的にセキュリティーが保護されていないため、安全でない操作を多数実行します。このため、仮想マシンを使用して Image Builder を実行します。

Image Builder が実行する環境 (仮想マシンなど) は、次の表に記載されている要件を満たす必要があります。

表1.2 Image Builder のシステム要件

パラメーター	最低要求値
システムのタイプ	専用の仮想マシン
プロセッサー	2 コア
メモリー	4 GiB
ディスク容量	20 GiB
アクセス権限	管理者レベル (root)
ネットワーク	インターネットへの接続



注記

UEFI システムに直接インストールした仮想マシンにイメージを作成することは対応していません。

第2章 IMAGE BUILDER のインストール

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

2.1. 仮想マシンへの IMAGE BUILDER のインストール

Image Builder を専用の仮想マシンにインストールするには、以下の手順を行います。

前提条件

- 仮想マシンに接続している。
- Image Builder 用の仮想マシンがインストールされ、サブスクライブされ、実行している。

手順

1. Image Builder および必要なパッケージを仮想マシンにインストールします。
   - lorax-composer
   - composer-cli
   - cockpit-composer
   - bash-completion

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

   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. システムを再起動するたびに、Image Builder が起動するようにします。

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

   lorax-composer サービスおよび **cockpit** サービスは、最初のアクセスで自動的に起動します。

3. Web コンソールへのアクセスを許可するように、システムのファイアウォールを設定します。

   ```
   # firewall-cmd --add-service=cockpit && firewall-cmd --add-service=cockpit --permanent
   ```

4. システムを再起動しなくても、**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読み込みます。

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

第3章 IMAGE BUILDER コマンドラインインターフェースでシステムイメージの作成

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。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、現在 Image Builder を使用する方法として推奨されているコマンドラインインターフェースを使用します。

3.1. IMAGE BUILDER コマンドラインインターフェース

Image Builder コマンドラインインターフェースは、現在 Image Builder を使用するのに推奨される方法です。Web コンソールインターフェース よりも多くの機能を提供します。このインターフェースを使用するには、**composer-cli** コマンドに、適切なオプションとサブコマンドを付けて実行します。

コマンドラインインターフェースのワークフローの概要は次のようになります。

1. 平文テキストファイルに Blueprint 定義をエクスポート (**保存**) する。
2. テキストエディターでこのファイルを編集する。
3. Image Builder で Blueprint のテキストファイルをインポート (**プッシュ**) する。
4. compose を実行して、Blueprint からイメージを構築する。
5. イメージファイルをエクスポートして、ダウンロードする。

この手順を実行する基本的なサブコマンドとは別に、**composer-cli** コマンドには、設定した Blueprint と Compose の状態を調べるサブコマンドが多数あります。

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

3.2. コマンドラインインターフェースで IMAGE BUILDER の BLUEPRINT の作成

この手順では、コマンドラインインターフェースを使用して Image Builder の Blueprint を新たに作成する方法を説明します。

手順

1. 以下の内容で平文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** を、Blueprint の名前および説明に置き換えます。

   Semantic Versioning スキームに従って、**0.0.1** をバージョン番号に置き換えます。

2. Blueprint 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 を、パッケージ名 (**httpd**、**gdb-doc**、**coreutils** など) に置き換えます。

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。

- 特定のバージョンを指定する場合は、**8.30** のように、バージョン番号を正確に指定してください。
- 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を指定する場合の形式は、**8.*** のようになります。

3. Blueprints は、さまざまな方法でカスタマイズできます。たとえば、同時マルチスレッド (SMT) は以下の手順で無効にできます。その他に利用できるカスタマイズは、「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。

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

4. 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. Blueprint をプッシュ (インポート) します。

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

BLUEPRINT-NAME を、前の手順で使用した値に置き換えます。

6. Blueprint 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 Blueprint を一覧表示します。

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

7. Blueprint に一覧表示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
```

3.3. コマンドラインインターフェースで IMAGE BUILDER の BLUEPRINT の編集

この手順は、コマンドラインインターフェースで既存の Image Builder の Blueprint を編集する方法を説明します。

手順

1. ローカルのテキストファイルに Blueprint を保存 (エクスポート) します。

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

2. **BLUEPRINT-NAME**.toml ファイルを、選択したテキストエディターで編集し、変更を加えます。

3. 編集を終了する前に、ファイルが有効な Blueprint であることを確認してください。

   a. 次の行がある場合は削除します。

   ```
   packages = []
   ```

   b. バージョン番号を大きくしてください。Image Builder の Blueprint バージョンは Semantic Versioning スキームを使用する必要があります。バージョンを変更しないと、バージョンの **patch** コンポーネントが自動的に増えます。

   c. コンテンツが有効な TOML 仕様かどうかを確認します。詳細は、TOML のドキュメント を参照してください。

   注記

   TOML のドキュメントはコミュニティーが提供しているため、Red Hat のサポート対象外となります。このツールの問題は、https://github.com/toml-lang/toml/issues から報告できます。

4. ファイルを保存してエディターを閉じます。

5. Blueprint を Image Builder にプッシュ (インポート) します。

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

   .toml 拡張子を含むファイル名を指定する必要がありますが、他のコマンドでは Blueprint の名前のみを使用することに注意してください。

6. Image Builder にアップロードしたコンテンツが編集内容と一致することを確認するには、Blueprint のコンテンツの一覧を表示します。

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

7. Blueprint に一覧表示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
   ```

3.4. IMAGE BUILDER コマンドラインインターフェースでシステムイメージの作成

この手順では、Image Builder コマンドラインインターフェースを使用して、カスタムイメージを作成する方法を説明します。

前提条件

- イメージに Blueprint を用意している。

手順

1. Compose を起動します。

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

BLUEPRINT-NAME を、Blueprint の名前に置き換え、**IMAGE-TYPE** を、イメージのタイプに置き換えます。設定できる値は、**composer-cli compose types** コマンドの出力を参照してください。

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。

2. Compose が完成するまで待ちます。完了には数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** となります。リスト内の Compose をその UUID で識別します。

3. Compose 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 を、前の手順で示した UUID 値に置き換えます。

あるいは、**/var/lib/lorax/composer/results/UUID/** の配下にあるイメージファイルに直接アクセスできます。

composer-cli compose logs UUID コマンドを使用してログをダウンロードし、**composer-cli compose metadata UUID** コマンドを使用してメタデータをダウンロードすることもできます。

3.5. IMAGE BUILDER コマンドラインの基本的なコマンド

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

Blueprint 操作

利用可能な Blueprint 一覧の表示

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

TOML 形式で Blueprint の内容の表示

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

TOML 形式の Blueprint 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
```

Blueprint の削除

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

TOML 形式の Blueprint ファイルを Image Builder へプッシュ (インポート)

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

Blueprint でイメージの構成

Compose の起動

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

BLUEPRINT を、構築する Blueprint の名前に置き換え、 **COMPOSE-TYPE** を、出力イメージタイプに置き換えます。

Compose の一覧表示

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

Compose、およびそのステータスの一覧表示

```
# 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
```

関連情報

- man ページの **composer-cli**(1) は、利用可能なサブコマンドとオプションの完全リストを提供します。

  ```
  $ man composer-cli
  ```

- **composer-cli** コマンドは、サブコマンドとオプションに関するヘルプを提供します。

  ```
  # composer-cli help
  ```

3.6. IMAGE BUILDER の BLUEPRINT 形式

TOML (Tom's Obvious Minimal Language) 形式の平文テキストとして、Image Builder の Blueprint をユーザーに提示します。

一般的な Blueprint ファイルの要素は次のとおりです。

Blueprint 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** を、Blueprint の名前および説明に置き換えます。

Semantic Versioning スキームに従い、**VERSION** をバージョン番号に置き換えます。

この部分は、Blueprint ファイル全体に対して一度だけ提示します。

modules エントリーは、パッケージの名前と、イメージにインストールするバージョンと一致する glob を説明します。

group エントリーは、イメージにインストールされるパッケージのグループを説明します。グループはパッケージを以下のように分類します。

- 必須
- デフォルト
- 任意
  Blueprint は、必須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グループ

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

group-name を、**anaconda-tools**、**widget**、**wheel**、**users** などのグループ名に置き換えます。

イメージに追加するパッケージ

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

package-name を、パッケージ名 (**httpd**、**gdb-doc**、**coreutils** など) に置き換えます。

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。

- 特定のバージョンを指定する場合は、8.30 のように、バージョン番号を正確に指定してください。

- 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を指定する場合の形式は、8.* のようになります。

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。

3.7. サポートされているイメージのカスタマイズ

現在、多くのイメージを Blueprint でカスタマイズできます。このオプションを使用できるようにするには、最初に Blueprint で設定し、Image Builder にインポート (プッシュ) する必要があります。

注記

このようなカスタマイズは、現在、添付の cockpit-composer GUI ではサポートされていません。

イメージのホスト名の設定

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

作成されるシステムイメージに対するユーザー指定

```
[[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
```

重要

ハッシュを生成する場合は、システムに **python3** をインストールする必要があります。以下のコマンドで、**python3** パッケージをインストールします。

```
# yum install python3
```

PASSWORD-HASH を、パスワードハッシュに置き換えます。ハッシュを生成するには、次のようなコマンドを実行します。

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

PUBLIC-SSH-KEY を、実際の公開鍵に置き換えます。

その他のプレースホルダーを、適切な値に置き換えます。

必要に応じて任意の行を省略します。ユーザー名のみが必須となります。

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。

作成されるシステムイメージに対するグループ指定

```
[[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。

既存ユーザーの SSH 鍵の設定

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

注記

このオプションは、既存ユーザーにのみ適用されます。ユーザーを作成して ssh 鍵を設定するには、**作成されたシステムイメージのカスタマイズのユーザー仕様** を使用します。

デフォルトにカーネルの起動パラメーターオプションを追加

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

イメージのホスト名の設定

```
[customizations]
hostname = "BASE-IMAGE"
```

作成されるシステムイメージのグループを追加

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

名前のみが必要であり、GID は任意です。

作成されたシステムイメージにタイムゾーンおよび Network Time Protocol (NTP) サーバーを設定

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

タイムゾーンを設定しないと、システムはデフォルトとして Universal Time, Coordinated (UTC) を使用します。NTP サーバーの設定はオプションです。

作成されたシステムイメージのロケール設定

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

言語とキーボードの両方のオプションを設定することが必要です。複数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になります。

作成されたシステムイメージのファイアウォールを設定

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

/etc/services ファイルの数値ポートまたは名前を使用して、一覧を有効または無効にすることができます。

システムの起動時に有効にするサービスの設定

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

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。イメージタイプによっては、サービスがすでに有効または無効になっているため、イメージが正常に機能し、この設定はオーバーライドできません。

Git リポジトリーから Blueprint へのファイルの追加

```
[[repos.git]]
rpmname = "_RPM-NAME"
rpmversion = "[RPM-VERSION]"
rpmrelease = "[RPM-RELEASE]"
summary = "[RPM-SUMMARY]"
repo = "[REPO-URL]"
ref = "[GIT-REF]"
destination = "[SERVICES]"
```

エントリーを使用して、git リポジトリーから作成されたイメージにファイルを追加できます。

たとえば、**server-config-1.0-1.noarch.rpm** という名前の RPM パッケージを作成するには、Blueprint に以下の情報を追加します。

_RPM-NAME を、作成する RPM パッケージの名前に置き換えます。これは、生成される tar アーカイブの接頭辞名でもあります。

RPM-VERSION を、RPM パッケージのバージョン (「1.0.0」など) に置き換えます。

RPM-RELEASE は、RPM パッケージリリースのバージョンに置き換えます (「1」など)。

RPM-SUMMARY は、RPM パッケージの要約文字列に置き換えます。

REPO-URL を、クローンを作成する get リポジトリーの URL に置き換え、そこからアーカイブを作成します。

GIT-REF を git 参照に置き換え、たとえば **origin/branch-name**、**git tag**、または **git commit hash** を確認します。

RPM パッケージのインストール時に、**SERVICES** を git リポジトリーのディレクトリーをインストールするパスに置き換えます。

これにより、指定した git リポジトリーがクローンされ、指定された git 参照が確認され、ファイルを宛先パス (/**opt/server/** など) にインストールする RPM パッケージが作成されます。RPM には、リポジトリーの詳細と、リポジトリーの作成に使用する参照に関するサマリーが含まれます。RPM パッケージは、イメージビルドのメタデータにも含まれています。

注記

ビルドが起動するたびに、リポジトリーのクローンが作成されます。大量の履歴が含まれるリポジトリーを参照する場合は、大量のディスク領域のクローンを作成して使用するのに時間がかかる場合があります。また、クローンは一時的なもので、RPM パッケージが作成されると削除されます。

3.8. インストール済みパッケージ

Image Builder を使用してシステムイメージを作成すると、システムはデフォルトでベースパッケージのセットをインストールします。パッケージのベースリストは、**comps core** グループのメンバーです。

表3.1 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
alibaba.ks	kernel, selinux-policy-targeted, cloud-init
ami.ks	kernel, selinux-policy-targeted, chrony, cloud-init
ext4-filesystem.ks	policycoreutils, selinux-policy-targeted, kernel
google.ks	kernel, selinux-policy-targeted
live-iso.ks	isomd5sum, kernel, dracut-config-generic, dracut-live, system-logos, selinux-policy-targeted
openstack.ks	kernel, selinux-policy-targeted
partitioned-disk.ks	kernel, selinux-policy-targeted
qcow2.ks	kernel, selinux-policy-targeted
tar.ks	policycoreutils, selinux-policy-targeted
vhd.ks	kernel, selinux-policy-targeted, chrony, WALinuxAgent, python3, net-tools, cloud-init, cloud-utils-growpart, gdisk

イメージタイプ	デフォルトパッケージ
vmdk.ks	kernel, selinux-policy-targeted, chrony, open-vm-tools

注記

Blueprint 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する必要があります。競合すると、システムが依存関係の解決に失敗します。これにより、カスタマイズしたイメージを作成することはできなくなります。

関連情報

- 利用可能なイメージタイプの詳細は、「Image Builder の出力形式」を参照してください。

3.9. 有効なサービス

有効なサービスとは、カスタムイメージを設定する際に、lorax-composer を実行している RHEL8 リリースのデフォルトサービス、および特定のイメージタイプに対して有効になっているサービスです。

たとえば、**.ami** イメージタイプは **sshd**、**chronyd**、および **cloud-init** のサービスを有効にし、このサービスがないと、カスタムイメージは起動しません。

表3.2 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	有効なサービス
alibaba.ks	sshd,cloud-init
ami.ks	デフォルトサービスなし
ext4-filesystem.ks	デフォルトサービスなし
google.ks	デフォルトサービスなし
live-iso.ks	NetworkManager
openstack.ks	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
partitioned-disk.ks	デフォルトサービスなし
qcow2.ks	デフォルトサービスなし
tar.ks	デフォルトサービスなし
vhd.ks	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final

イメージタイプ	有効なサービス
vmdk.ks	sshd, chronyd, vmtoolsd

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、サービスがデフォルトで有効になっているイメージタイプの場合、カスタマイズはこの機能をオーバーライドしません。

関連情報

- サービスのカスタマイズに関する詳細は、「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。

3.10. IMAGE BUILDER を使用したディスクおよびパーティション設定

Image Builder では、ディスクのパーティション設定はできません。パーティションが設定されたディスクを持つ出力タイプには、1つのパーティションと、システムイメージの起動に必要なプラットフォーム固有のパーティションがあります。たとえば、**qcow2** イメージタイプには root パーティションが1つあり、場合によっては、**PPC64** システムの **PReP** のように、イメージが起動するために必要なプラットフォーム固有のブートパーティションがあります。

第4章 IMAGE BUILDER WEB コンソールインターフェースでシステムイメージの作成

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。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ースを使用できます。ただし、コマンドラインインターフェース 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ースを使用することが推奨されます。

4.1. RHEL 8 WEB コンソールで IMAGE BUILDER GUI へのアクセス

RHEL 8 Web コンソールの **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ースを使用して、Image Builder の Blueprint と Compose を管理できるようになります。現在、Image Builder を制御するのに推奨される方法は、コマンドラインインターフェースを使用することです。

前提条件

- システムへの root アクセス権限がある。

手順

1. Image Builder がインストールされている Web ブラウザーで **https://localhost:9090/** を開きます。
   Image Builder にリモートでアクセスする方法は**『RHEL 8 で Web コンソールを使用したシステムの管理』**を参照してください。

2. そのシステムへの権限が十分なユーザーアカウントの認証情報で、Web コンソールにログインします。

3. Image Builder コントロールを表示するには、ウィンドウの左上にある **Image Builder** アイコンをクリックします。
   Image Builder ビューが開き、既存の Blueprint の一覧が表示されます。

関連情報

- *3章Image Builder コマンドラインインターフェースでシステムイメージの作成*

4.2. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT の作成

カスタマイズしたシステムイメージを説明するには、最初に Blueprint を作成します。

前提条件

- ブラウザーで、RHEL 8 Web コンソールの Image Builder インターフェースを開いている。

手順

1. 右上隅の **Blueprint の作成** をクリックします。
   ポップアップに Blueprint 名および説明のフィールドが表示されます。

2. Blueprint の名前およびその説明を入力したら、**作成** をクリックします。
   ウィンドウが Blueprint の編集モードに切り替わります。

3. システムイメージに追加するコンポーネントを追加します。

    1. 左側の **利用可能なコンポーネント** フィールドにコンポーネント名またはその一部を入力し、**Enter** を押します。
    テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下のコンポーネントのリストが、検索条件と一致するものに絞られます。

    コンポーネントのリストが長すぎる場合は、さらに検索の用語を追加します。

    2. コンポーネントのリストは、1ページずつ表示されます。別の結果ページに移動するには、コンポーネントリストの上にある矢印および入力フィールドを使用します。

    3. 使用するコンポーネントの名前をクリックし、その詳細を表示します。右側のペインに、コンポーネントの詳細 (バージョンや依存関係など) が表示されます。

    4. **コンポーネントのオプション** ボックスの **バージョンリリース** ドロップダウンメニューで、使用するバージョンを選択します。

    5. 左上の **追加** をクリックします。

    6. 誤ってコンポーネントを追加した場合は、 **...** ボタンをクリックし、メニューで **削除** を選択してそのコンポーネントを削除します。

注記

コンポーネントのバージョンを選択しない場合は、コンポーネントリストの右側の **+** ボタンをクリックすると、コンポーネントの詳細ウィンドウおよびバージョンの選択をスキップできます。

4. Blueprint を保存するには、変更の概要に関するダイアログがポップアップとして表示されるダイアログの右上にある **コミット** をクリックします。**コミット** をクリックします。
右側の小さなポップアップに保存の進捗が表示され、続いて結果が表示されます。

5. 編集画面を終了するには、左上で **Back to Blueprints** をクリックします。
Image Builder ビューが開き、既存の Blueprint の一覧が表示されます。

4.3. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT の編集

カスタムのシステムイメージの仕様を変更するには、対応する Blueprint を編集します。

前提条件

- ブラウザーで、RHEL 8 Web コンソールの Image Builder インターフェースを開いている。
- Blueprint が存在する。

手順

1. 編集する Blueprint を探します。左上の検索ボックスに Blueprint の名前またはその一部を入力し、**Enter** を押します。
テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下の Blueprint のリストが、検索条件と一致するものに絞られます。

    Blueprint のリストが長すぎる場合には、さらに検索の用語を追加します。

2. Blueprint の右側で、その Blueprint の **Blueprint の編集** ボタンを押します。
ウィンドウが Blueprint の編集ウィンドウに切り替わります。

3. 右側のペインで、不要なコンポーネントのエントリー右端の　ボタンをクリックし、メニューで **削除** を選択してそのコンポーネントを削除します。

4. 既存のコンポーネントのバージョンを変更します。

   a. Blueprint コンポーネント検索フィールドで、**Blueprint コンポーネント** の見出しの下にあるフィールドにコンポーネント名またはその一部を入力し、**Enter** を押します。
   テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下のコンポーネントのリストが、検索条件と一致するものに絞られます。

   コンポーネントのリストが長すぎる場合は、さらに検索の用語を追加します。

   b. コンポーネントのエントリー右端の　ボタンをクリックし、メニューで **表示** を選択します。
   右側のペインに、コンポーネントの詳細ウィンドウが表示されます。

   c. **バージョンリリース** ドロップダウンメニューで目的のバージョンを選択し、右上の **変更の適用** をクリックします。
   変更が保存され、右側のペインが Blueprint コンポーネントのリストに戻ります。

5. 新しいコンポーネントを追加します。

   a. 左側で、**利用可能なコンポーネント** の見出しの下にあるフィールドにコンポーネント名またはその一部を入力し、**Enter** を押します。
   テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下のコンポーネントのリストが、検索条件と一致するものに絞られます。

   コンポーネントのリストが長すぎる場合は、さらに検索の用語を追加します。

   b. コンポーネントのリストは、1ページずつ表示されます。別の結果ページに移動するには、コンポーネントリストの上にある矢印および入力フィールドを使用します。

   c. 使用するコンポーネントの名前をクリックし、その詳細を表示します。右側のペインに、コンポーネントの詳細 (バージョンや依存関係など) が表示されます。

   d. **コンポーネントのオプション** ボックスで、**バージョンリリース** ドロップダウンメニューを使用して、使用するバージョンを選択します。

   e. 右上の **追加** をクリックします。

   f. 誤ってコンポーネントを追加してしまった場合には、右側のペインでそのエントリー右端の　ボタンをクリックし、メニューで **削除** を選択してそのコンポーネントを削除します。

   注記

   コンポーネントのバージョンを選択しない場合は、コンポーネントリストの右側の **+** ボタンをクリックすると、コンポーネントの詳細ウィンドウおよびバージョンの選択をスキップできます。

6. 変更を加えた新しいバージョンの Blueprint をコミットします。

   a. 右上の **コミット** ボタンをクリックします。
   ポップアップウィンドウに変更の概要が表示されます。

b. 変更内容を確認し、**コミット** をクリックして確定します。
右側の小さなポップアップに保存の進捗が表示され、続いて結果が表示されます。新しいバージョンの Blueprint が作成されます。

c. 左上の **Blueprint に戻る** をクリックし、編集ウィンドウを終了します。
Image Builder ビューが開き、既存の Blueprint の一覧が表示されます。

4.4. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT にユーザーおよびグループを追加

現在、Web コンソールインターフェースの Blueprint に、ユーザーやグループなどのカスタマイズを追加することはできません。この制限を回避するには、Web コンソールの **Terminal** タブを使用して、コマンドラインインターフェース (CLI) ワークフローを使用します。

前提条件

- Blueprint が存在する。
- **vim**、**nano**、**emacs** などの CLI テキストエディターがインストールされている。インストールするには、以下のコマンドを実行します。

```
# yum install editor-name
```

手順

1. Blueprint の名前を確認します。RHEL 8 Web コンソールの左側にある Image Builder (**Image builder**) タブを開いて、Blueprint の名前を表示します。
2. Web コンソールの CLI に移動します。左側でシステム管理タブを開いて、左側のリストにある最後の項目 **Terminal** を選択します。
3. スーパーユーザー (root) モードに入ります。

```
$ sudo bash
```

認証情報を求められたら入力してください。端末は、Web コンソールにログインする際に入力した認証情報を再利用しません。

ホームディレクトリーで、root 権限を持つ新しいシェルが起動します。

4. Blueprint をファイルにエクスポートします。

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

5. **BLUEPRINT-NAME**.toml ファイルを、選択した CLI テキストエディターで編集し、ユーザーとグループを追加します。

重要

RHEL 8 の Web コンソールには、システムにあるテキストファイルを編集するのに使用する組み込み機能がないため、ここでは CLI テキストエディターを使用する必要があります。

a. 追加するすべてのユーザーに対して、このブロックをファイルに追加します。

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

PASSWORD-HASH を、パスワードハッシュに置き換えます。ハッシュを生成するには、以下のようなコマンドを実行します。

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

ssh-rsa (...) key-name を、実際の公開鍵に置き換えます。

その他のプレースホルダーを、適切な値に置き換えます。

必要に応じて任意の行を省略します。ユーザー名のみが必須となります。

b. 追加するすべてのユーザーグループに対して、このブロックをファイルに追加します。

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

c. バージョン番号を大きくしてください。

d. ファイルを保存してエディターを閉じます。

6. Blueprint を Image Builder にインポートします。

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

.toml 拡張子を含むファイル名を指定する必要がありますが、他のコマンドでは Blueprint の名前のみを使用することに注意してください。

7. Image Builder にアップロードしたコンテンツが編集内容と一致することを確認するには、Blueprint のコンテンツの一覧を表示します。

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

バージョンが、ファイルに指定したバージョンと一致するか、およびカスタマイズが存在するかどうかを確認します。

重要

Blueprint に含まれるパッケージも編集しないと、変更が適用されたことを確認するために使用できる情報が、RHEL 8 Web コンソールの Image Builder プラグインには表示されません。

8. 特権シェルを終了します。

```
# exit
```

9. 左側の Image Builder (**Image builder**) タブを開き、開いていたすべてのブラウザーとタブでページを更新します。
これにより、読み込まれたページにキャッシュされた状態が、誤って変更を元に戻すことを防ぎます。

関連情報

- 「Image Builder の Blueprint 形式」
- 「コマンドラインインターフェースで Image Builder の Blueprint の編集」

4.5. WEB コンソールインターフェースで IMAGE BUILDER を使用したシステムイメージの作成

以下の手順では、システムイメージの作成について説明します。

前提条件

- ブラウザーで、RHEL 8 Web コンソールの Image Builder インターフェースを開いている。
- Blueprint が存在する。

手順

1. イメージをビルドする Blueprint を検索します。検索ボックスに Blueprint の名前またはその一部を入力し、**Enter** を押します。
テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下の Blueprint のリストが、検索条件と一致するものに絞られます。

   Blueprint のリストが長すぎる場合には、さらに検索の用語を追加します。

2. Blueprint の右側で、その Blueprint に関する **イメージの作成** ボタンを押します。
ポップアップウィンドウが表示されます。

3. イメージのタイプを選択し、**作成** を押します。
右上の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。

4. Blueprint の名前をクリックします。
Blueprint の詳細に関するウィンドウが表示されます。

5. **Images** タブをクリックして切り替えます。作成中のイメージは、**In Progress** ステータスで一覧表示されます。

注記

イメージの作成には数分かかります。イメージの作成中、進捗は表示されません。

イメージの作成を中止するには、右側の **停止** ボタンを押します。

6. イメージが正常に作成されると、**停止** ボタンが **ダウンロード** ボタンに変わります。このボタンをクリックして、システムにイメージをダウンロードします。

4.6. BLUEPRINT へのソースの追加

Image Builder で定義したソースは、Blueprint に追加できるコンテンツを提供します。これらのソースはグローバルであるため、すべての Blueprint で利用可能です。システムソースはコンピューターにローカルで設定されているリポジトリーで、Image Builder からは削除できません。カスタムソースを追加できるため、システムで利用できるシステムソース以外のコンテンツにアクセスできます。

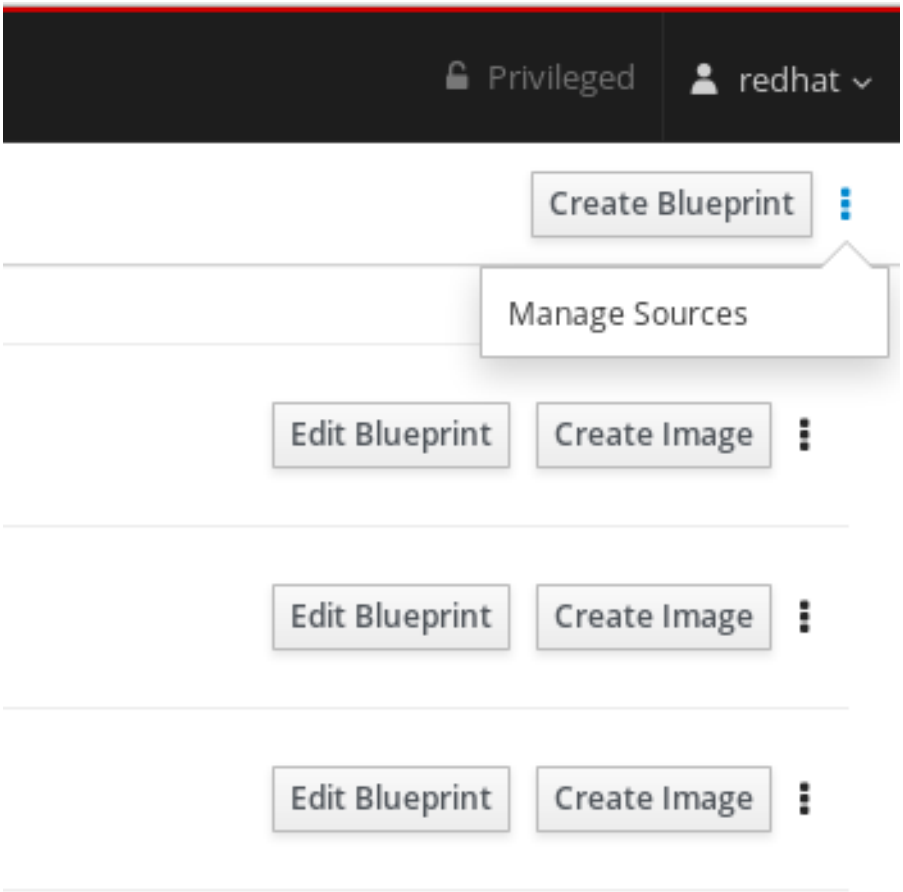
以下の手順では、ローカルシステムにソースを追加する方法を説明します。

前提条件

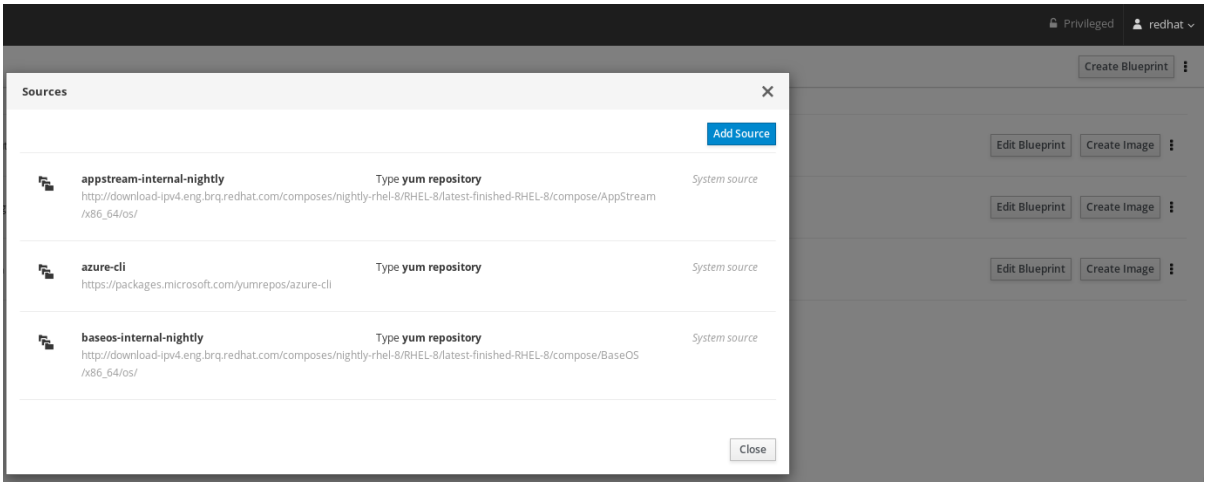
- ブラウザーで、RHEL 8 Web コンソールの Image Builder インターフェースを開いている。

手順

1. 右上端にある　**Manage Sources** ボタンをクリックします。



ポップアップウィンドウが表示され、利用可能なソースと、その名前と説明が表示されます。



2. ポップアップウィンドウの右側で、**ソースの追加** ボタンをクリックします。

3. 必要な **ソースの名前**、**ソースのパス**、および **ソースのタイプ** を追加します。**セキュリティー** フィールドはオプションです。

4. **ソースの追加** ボタンをクリックします。画面には、利用可能なソースウィンドウが表示され、追加したソースが一覧表示されます。

これにより、新しいシステムソースが利用可能になり、使用または編集できる状態になります。

4.7. BLUEPRINT のユーザーアカウントの作成

Image Builder で作成したイメージでは root アカウントがロックされ、他のアカウントは含まれていません。このような設定は、パスワードなしで誤ってイメージをビルドし、デプロイできないように提供されます。Image Builder を使用すると、Blueprint のパスワードでユーザーアカウントを作成でき、Blueprint から作成したイメージにログインできます。

前提条件

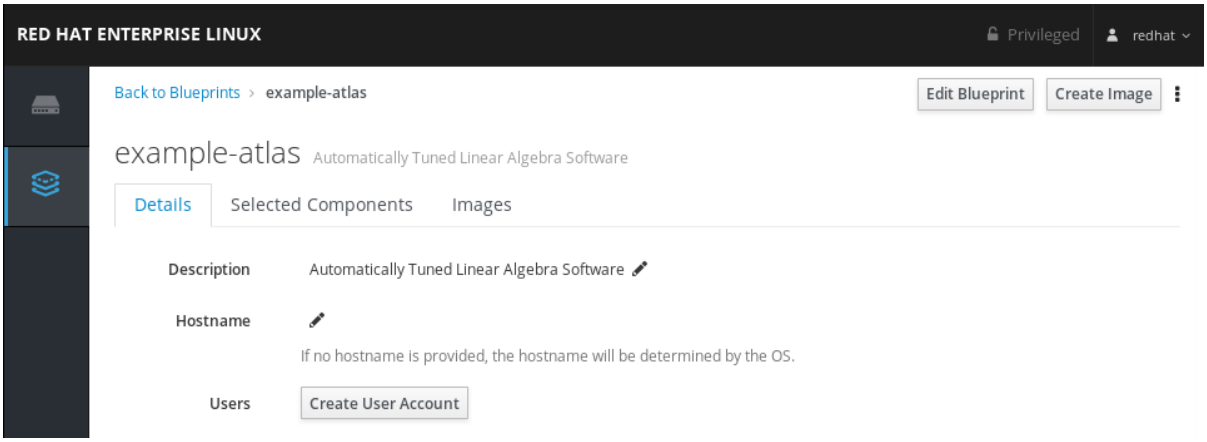
- ブラウザーで、RHEL 8 Web コンソールの Image Builder インターフェースを開いている。
- 既存の Blueprint がある。

手順

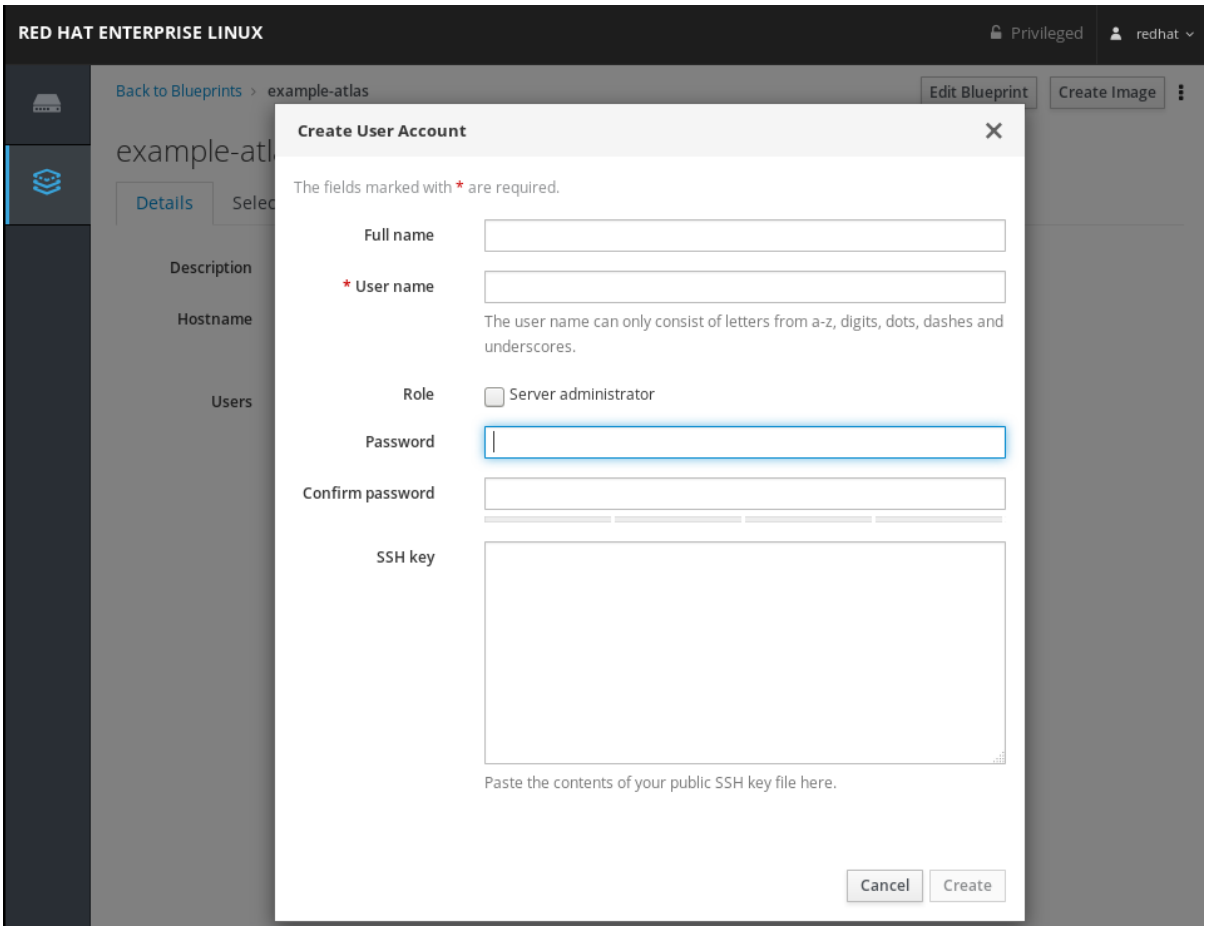
1. 左上の検索ボックスに Blueprint の名前またはその一部を入力し、**Enter** を押して、ユーザーアカウントを作成する Blueprint を検索します。

テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下の Blueprint のリストが、検索条件と一致するものに絞られます。

2. Blueprint 名をクリックして、Blueprint の詳細を表示します。



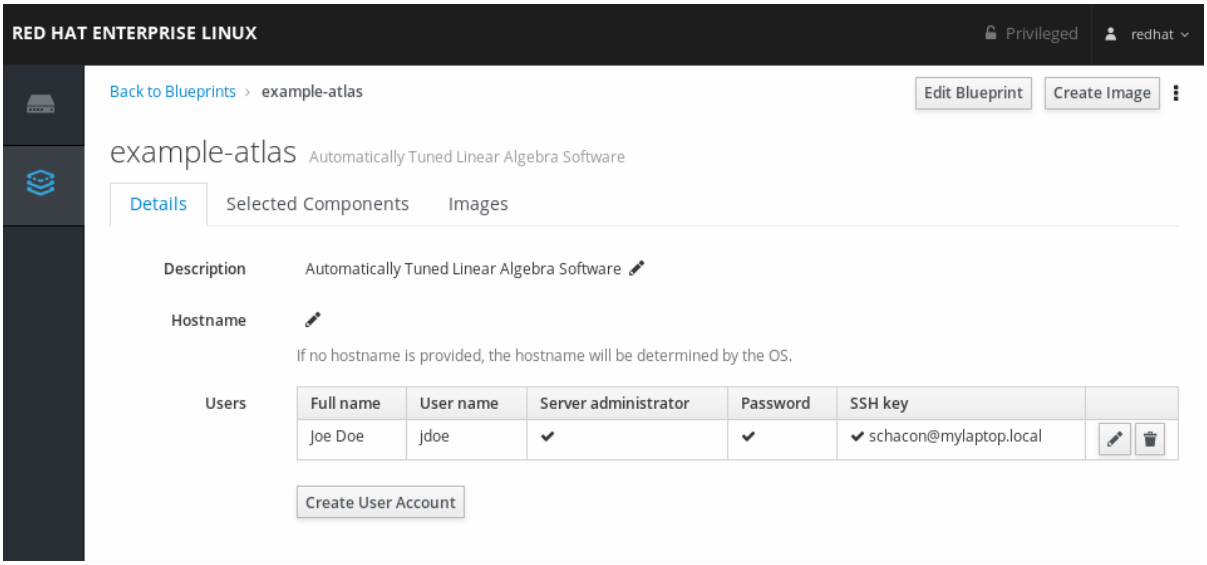
3. **ユーザーアカウントの作成** をクリックします。
これにより、ユーザーアカウントを作成するフィールドを含むウィンドウが開きます。



4. 詳細を入力します。名前を入力すると、**ユーザー名** のフィールドが自動補完され、ユーザー名を提案することに注意してください。

5. 必要な情報をすべて入力したら、**作成** をクリックします。

6. 作成したユーザーアカウントが表示され、追加したすべての情報が表示されます。



7. Blueprint のユーザーアカウントをさらに作成する場合は、プロセスを繰り返します。

4.8. SSH 鍵を持つユーザーアカウントの作成

Image Builder で作成したイメージでは root アカウントがロックされ、他のアカウントは含まれていません。このような設定は、デフォルトパスワードを持たないイメージのセキュリティーを確保するために提供されます。Image Builder を使用すると、Blueprint の SSH 鍵でユーザーアカウントを作成し、Blueprint から作成したイメージに対して認証できるようにします。これを実行するには、まず Blueprint を作成します。次に、パスワードと SSH 鍵でユーザーアカウントを作成します。以下の例は、SSH キーを使用してサーバー管理ユーザーを作成する方法を示しています。

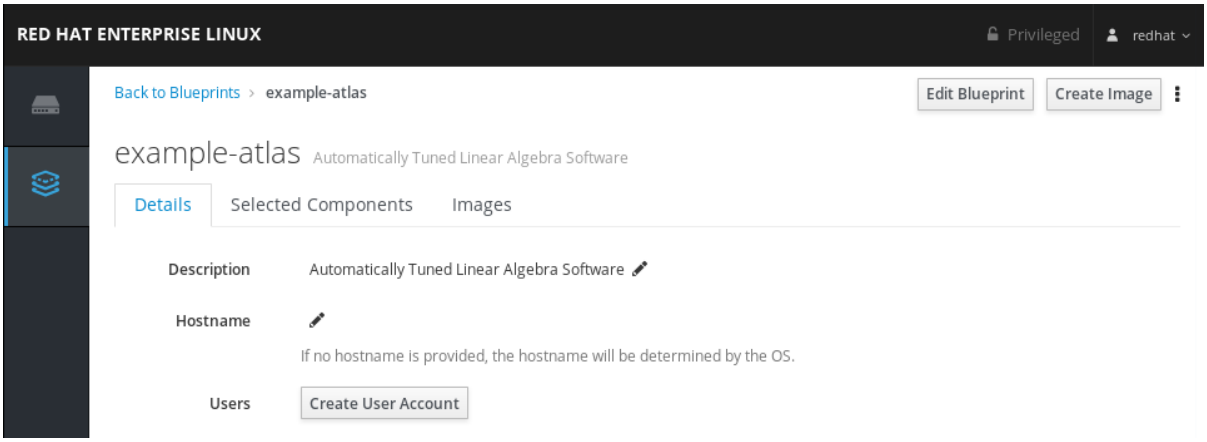
前提条件

- プロセスを後で作成したユーザーとペアにする SSH キーを作成している。
- ブラウザーで、RHEL 8 Web コンソールの Image Builder インターフェースを開いている。
- 既存の Blueprint がある。

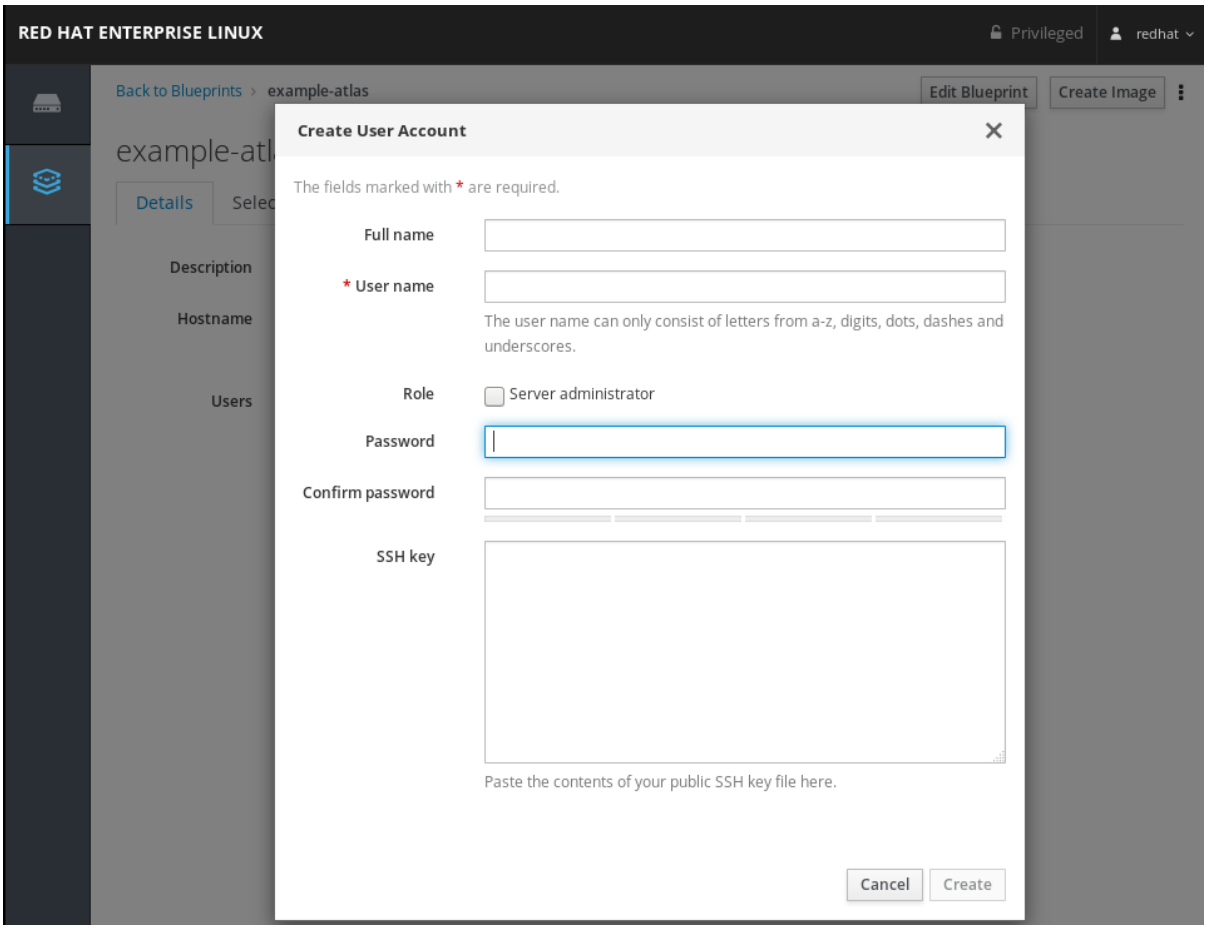
手順

1. 左上の検索ボックスに Blueprint の名前またはその一部を入力し、**Enter** を押して、ユーザーアカウントを作成する Blueprint を検索します。
   テキスト入力フィールドの下にあるフィルターリストに検索条件が追加され、その下の Blueprint のリストが、検索条件と一致するものに絞られます。

2. Blueprint 名をクリックして、Blueprint の詳細を表示します。



3. **ユーザーアカウントの作成** をクリックします。
これにより、ユーザーアカウントを作成するフィールドを含むウィンドウが開きます。

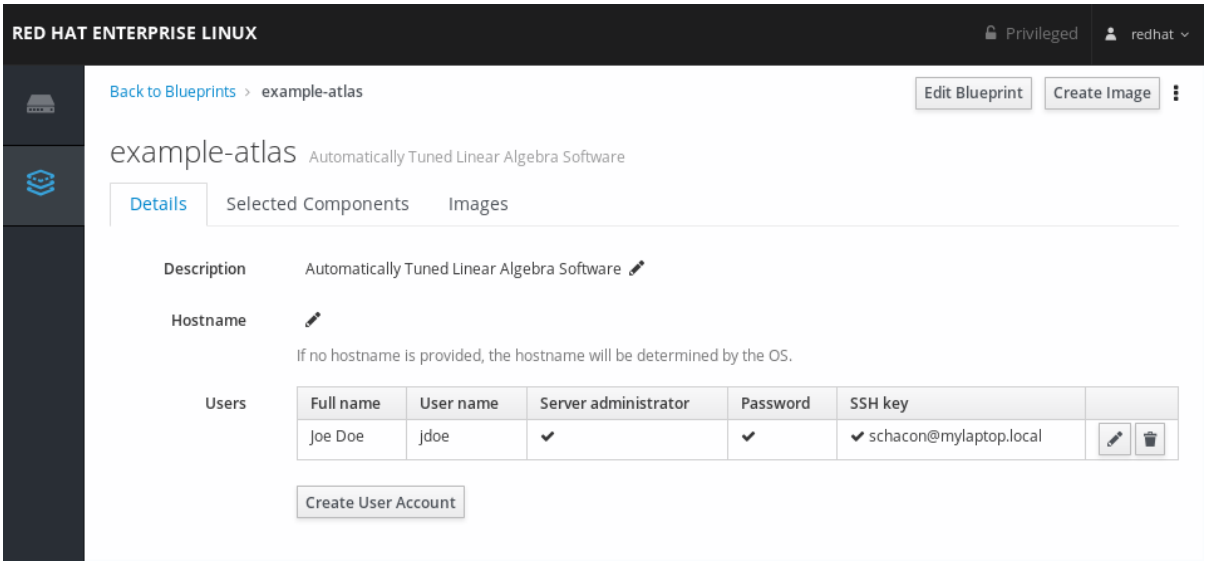


4. 詳細を入力します。名前を入力すると、**ユーザー名** のフィールドが自動補完され、ユーザー名を提案することに注意してください。
作成するユーザーアカウントに管理者権限を付与する場合は、**Role** フィールドを確認します。

   公開 SSH 鍵のファイルの内容を貼り付けます。

5. 必要な情報をすべて入力したら、**作成** をクリックします。

6. 新しいユーザーアカウントがユーザーリストに表示され、入力したすべての情報が表示されます。



7. Blueprint のユーザーアカウントをさらに作成する場合は、プロセスを繰り返します。

関連情報

- SSH キーの詳細は「スマートカードに保存された SSH 鍵の使用」を参照してください。

第5章 IMAGE BUILDER でクラウドイメージの準備およびアップロード

Image Builder を使用して、さまざまなプロバイダーのクラウドで使用できるようにカスタムの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、各出力タイプを使用して 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。

5.1. AWS AMI イメージのアップロードの準備

ここでは、AWS AMI イメージをアップロードするようにシステムを設定する手順を説明します。

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。

```
# yum install python3
# yum install 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. AWS アクセスの詳細に従って、AWS コマンドラインクライアントを設定します。

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

4. バケットを使用するように、AWS コマンドラインクライアントを設定します。

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

bucketname を、バケット名に置き換えます。

5. IAM で **vmimport** S3 Role を作成し、S3 にアクセスする権限を付与していない場合はここで付与します。

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service":
"vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{
"sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
$ printf '{ "Version":"2012-10-17", "Statement":[ { "Effect":"Allow", "Action":[
"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource":[ "arn:aws:s3:::%s",
"arn:aws:s3:::%s/*" ] }, { "Effect":"Allow", "Action":[ "ec2:ModifySnapshotAttribute",
```

```
"ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource":"*" } ] }' $BUCKET
$BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

5.2. AWS への AMI イメージのアップロード

本セクションは、AMI イメージを AWS にアップロードする手順を説明します。

前提条件

- AWS イメージのアップロードを設定している。
- Image Builder で AWS イメージを作成している。イメージの作成時に、CLI で **ami** 出力タイプ、または GUI で **Amazon Machine Image Disk (.ami)** を使用します。

手順

1. イメージを S3 にプッシュします。

```
$ AMI=8db1b463-91ee-4fd9-8065-938924398428-disk.ami
$ aws s3 cp $AMI s3://$BUCKET
Completed 24.2 MiB/4.4 GiB (2.5 MiB/s) with 1 file(s) remaining
...
```

2. S3 エンドへアップロードしたら、イメージをスナップショットとして EC2 にインポートします。

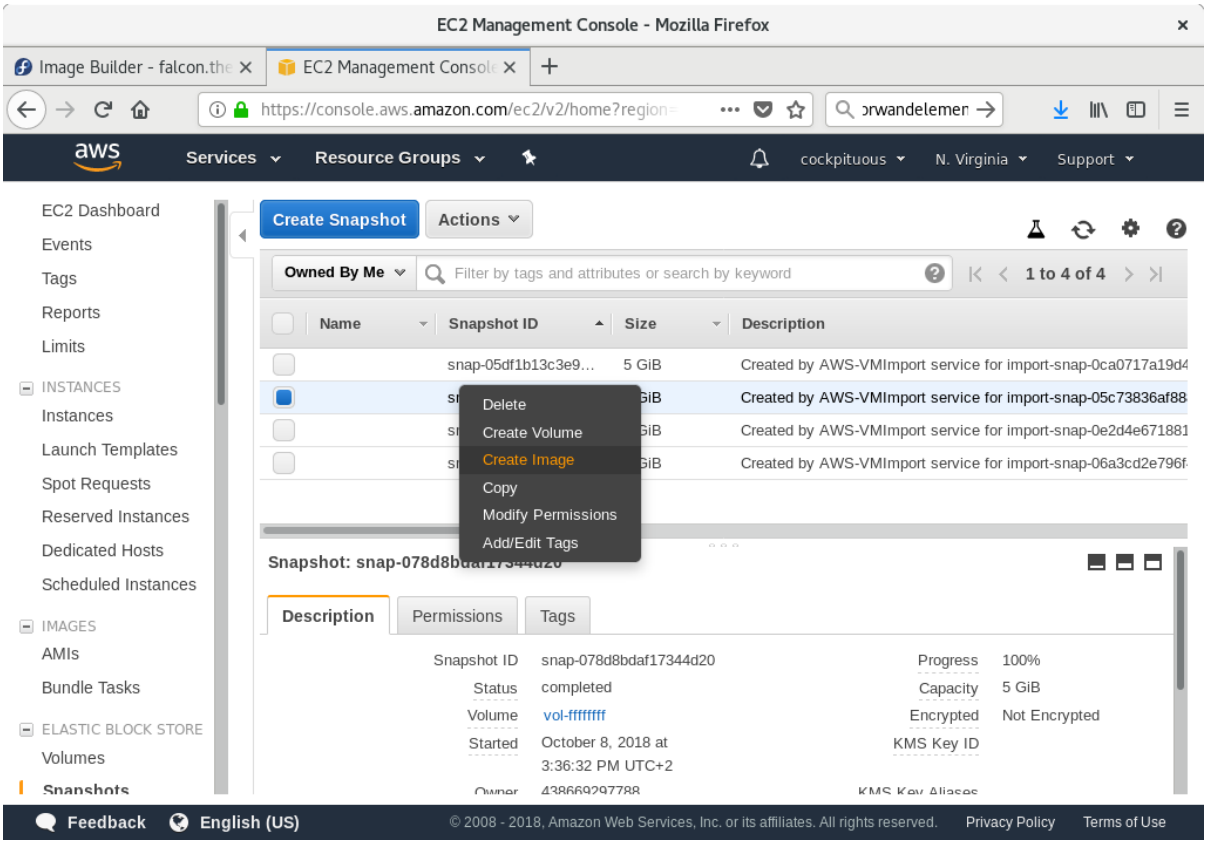
```
$ printf '{ "Description": "my-image", "Format": "raw", "UserBucket": { "S3Bucket": "%s",
"S3Key": "%s" } }' $BUCKET $AMI > containers.json
$ aws ec2 import-snapshot --disk-container file://containers.json
```

my-image を、イメージ名に置き換えます。

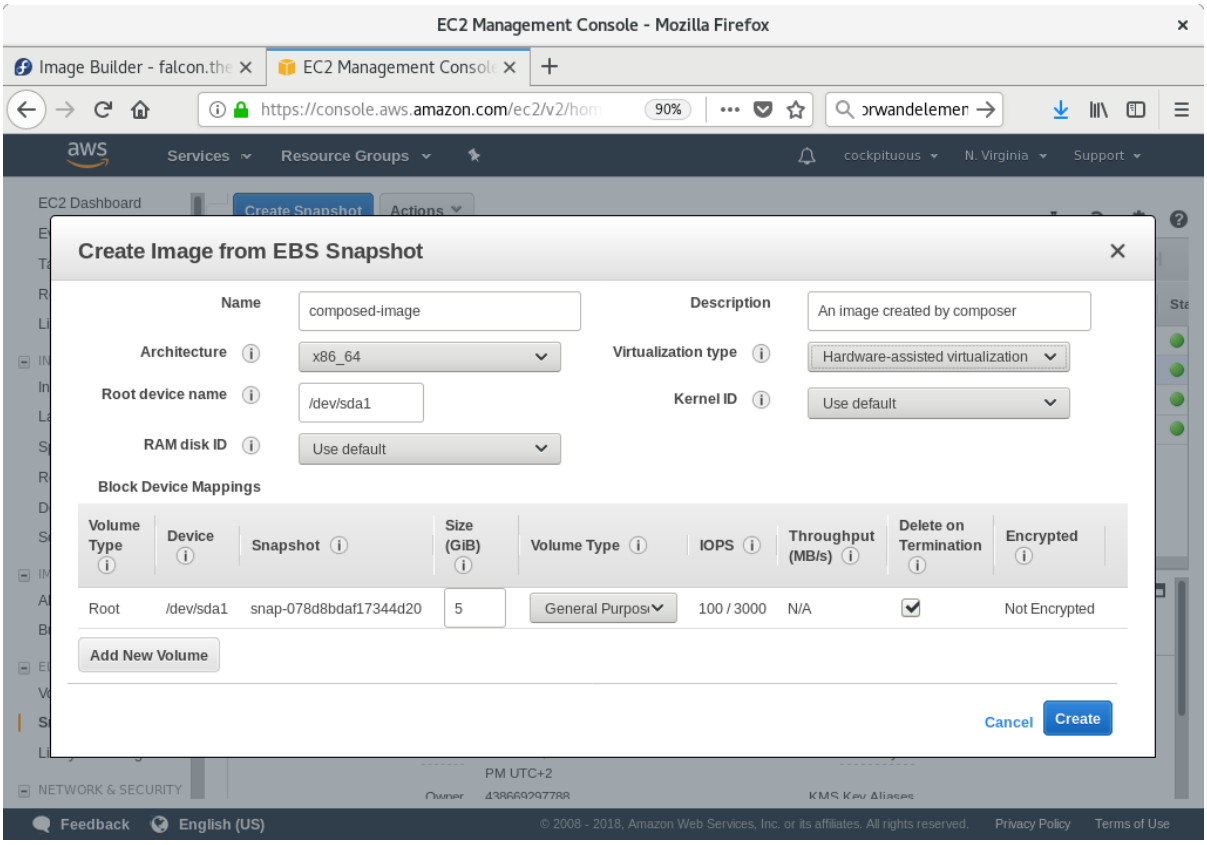
インポートの進行状況を追跡するには、次のコマンドを実行します。

```
$ aws ec2 describe-import-snapshot-tasks --filters Name=task-state,Values=active
```

3. EC2 コンソールでスナップショットを選択して右クリックし、**イメージの作成** を選択して、アップロードしたスナップショットからイメージを作成します。



4. 作成するイメージで、**Hardware-assisted virtualization** の **Virtualization** タイプを選択します。



5. これで、CLI または AWS コンソールを使用して、スナップショットからインスタンスを実行できます。作成した EC2 インスタンスにアクセスするには、SSH で秘密鍵を使用します。**ec2-user** としてログインします。

5.3. AZURE VHD イメージのアップロードの準備

ここでは、VHD イメージを Azure にアップロードする手順を説明します。

前提条件

- 使用可能な Azure リソースグループおよびストレージアカウントがある。

手順

1. python2 をインストールします。

```
# yum install python2
```

注記

AZ CLI は、特に python 2.7 に依存しているため、**python2** パッケージがインストールされている必要があります。

2. 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
```

3. ローカルの azure-cli リポジトリー情報を作成します。

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

4. Azure CLI をインストールします。

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

注記

ダウンロードする Azure CLI パッケージのバージョンは、現在ダウンロードしているバージョンによって異なる可能性があります。

5. Azure CLI を実行します。

```
$ az login
```

端末に「Note, we have launched a browser for you to login.」メッセージが表示されます。デバイスコードに以前使用したものがある場合は、「az login --use-device-code」を使用し、ログインでき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は、そのリンクがブラウザーで開きません。この場合は、表示されたリンクを使用してログインして、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 https://microsoft.com/devicelogin ページを開き、認証するコード **XXXXXXXXX** を入力します。

6. Azure で、ストレージアカウントのキーを一覧表示します。

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

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

注記

以下のコマンドを実行すると、利用可能なリソースを一覧表示できます。

```
$ az resource list
```

7. 上記コマンドの出力の **key1** の値を書き留め、それを環境変数に割り当てます。

```
$ KEY1=value
```

8. ストレージコンテナーを作成します。

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

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

関連情報

- Azure CLI

5.4. VHD イメージの AZURE へのアップロード

ここでは、VHD イメージを Azure にアップロードする手順を説明します。

前提条件

- 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。
- Image Builder で Azure VHD イメージを作成している。イメージの作成時に、CLI で **vhd** 出力タイプ、または GUI で **Azure Disk Image (.vhd)** を使用している。

手順

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

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

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

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

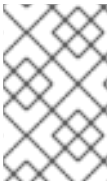
3. Azure ポータル、または以下のようなコマンドを使用して、インスタンスを作成します。

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

4. 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。

5.5. VMDK イメージの VSPHERE へのアップロード

Image Builder は、VMware ESXi システムまたは vSphere システムへのアップロードに適したイメージを生成できます。ここでは、VMDK イメージを VMware vSphere にアップロードする手順を説明します。



注記

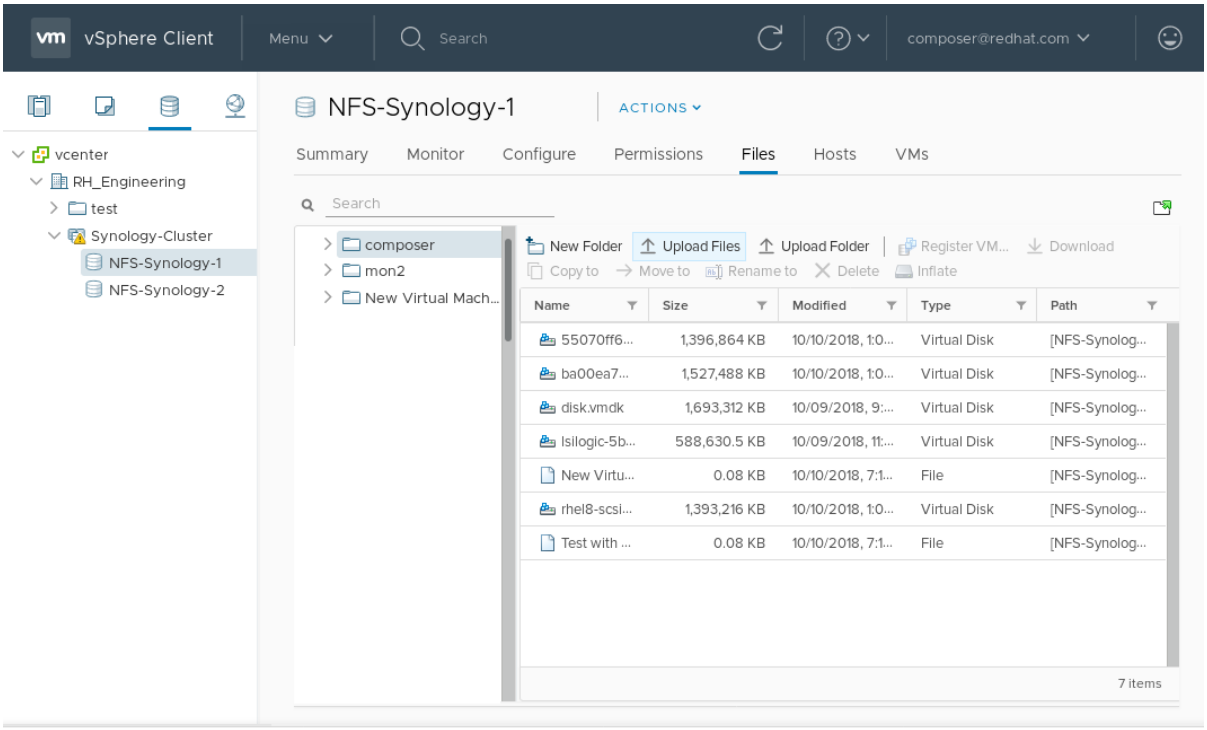
VMWare デプロイメントは、一般的に仮想マシンにユーザーの認証情報を挿入するように cloud-init が構成されていないため、Blueprint でそのタスクを自分で実行する必要があります。

前提条件

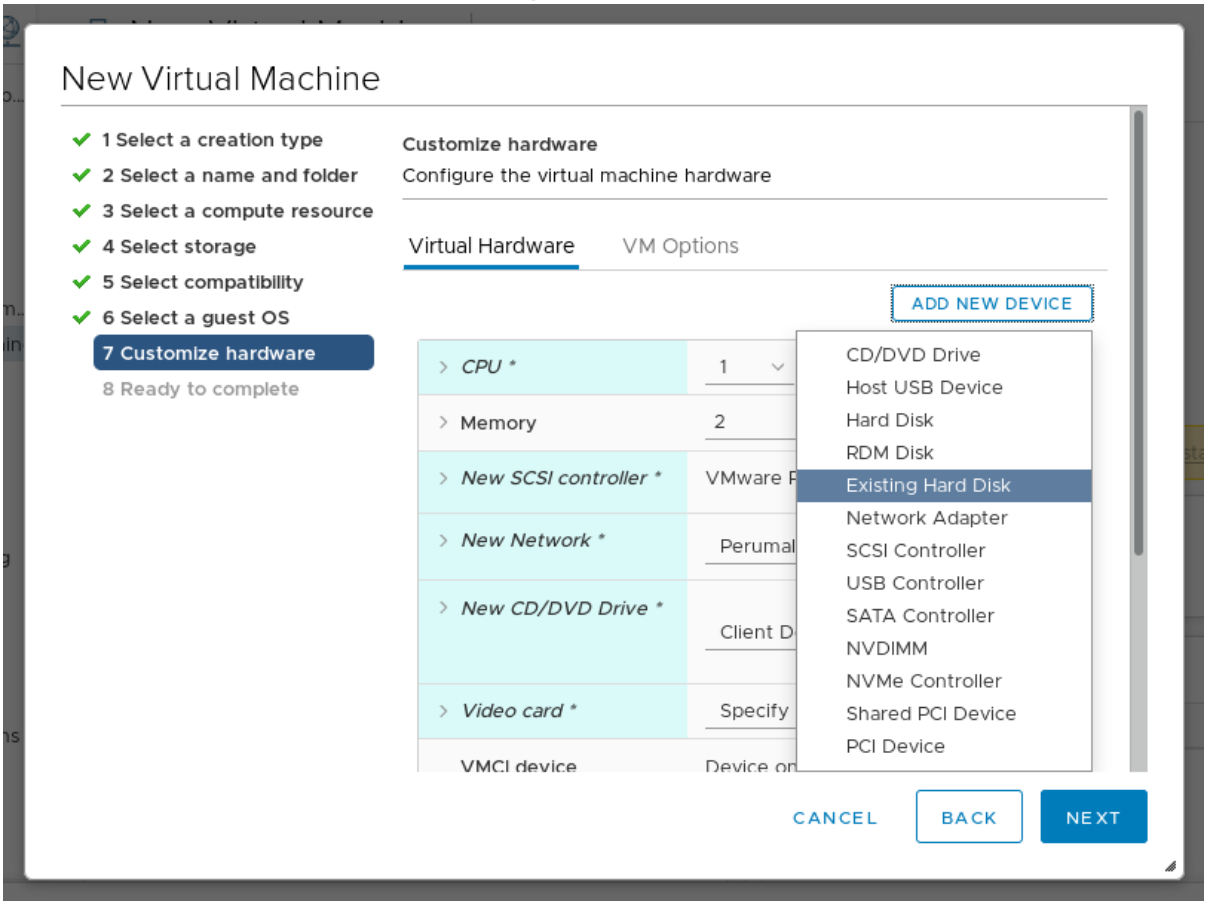
- Image Builder で VMDK イメージを作成している。イメージの作成時に、CLI で **vmdk** 出力タイプ、または GUI で **VMware Virtual Machine Disk (.vmdk)**を使用します。

手順

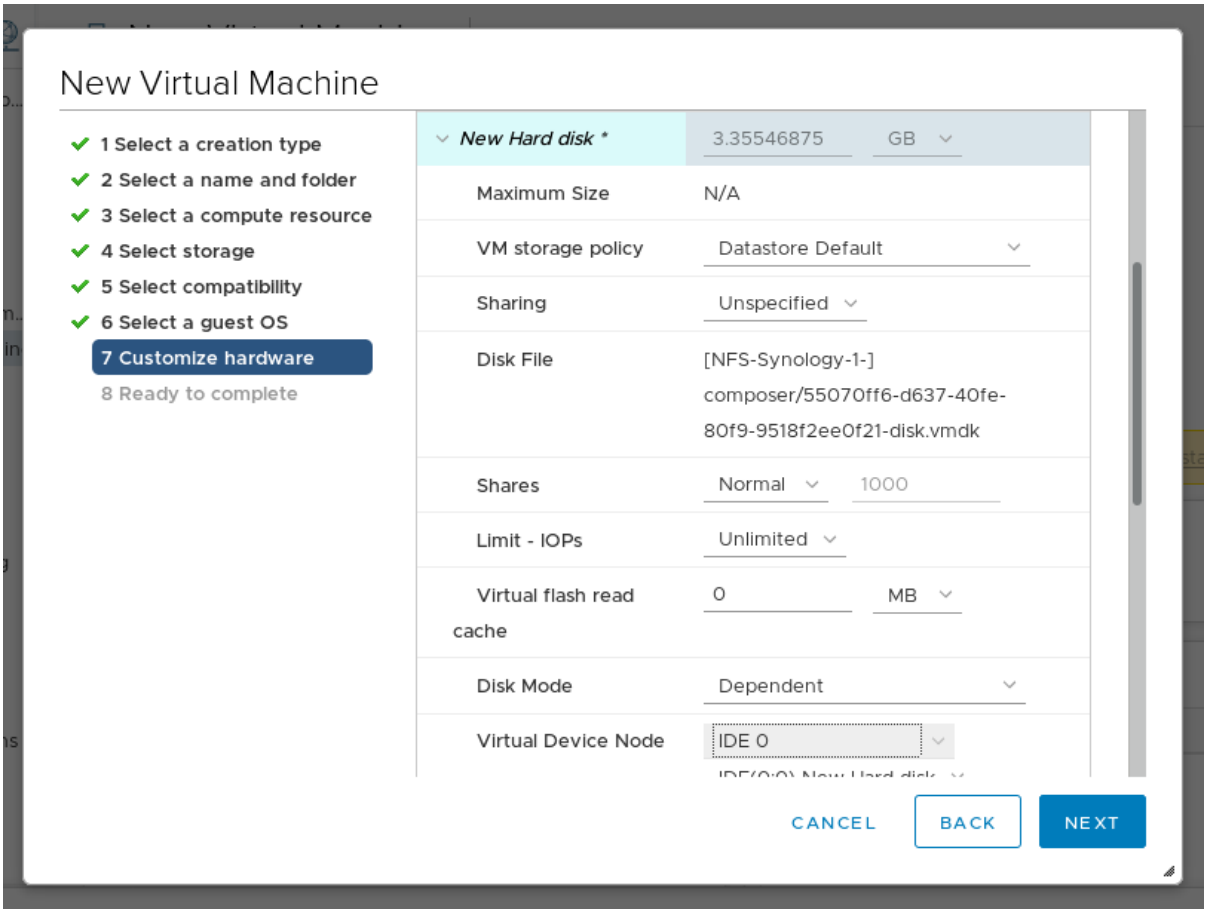
1. HTTP 経由でイメージを vSphere にアップロードします。vCenter で **Upload Files** をクリックします。



2. 仮想マシンを作成する場合は、**Device Configuration** 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。



3. 作成するディスクには、必ず **IDE** デバイスを **Virtual Device Node** として使用してください。デフォルト値の **SCSI** にすると、仮想マシンが起動できません。



5.6. QCOW2 イメージの OPENSTACK へのアップロード

Image Builder は、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イメージを生成できます。ここでは、QCOW2 イメージを OpenStack にアップロードする手順を説明します。

前提条件

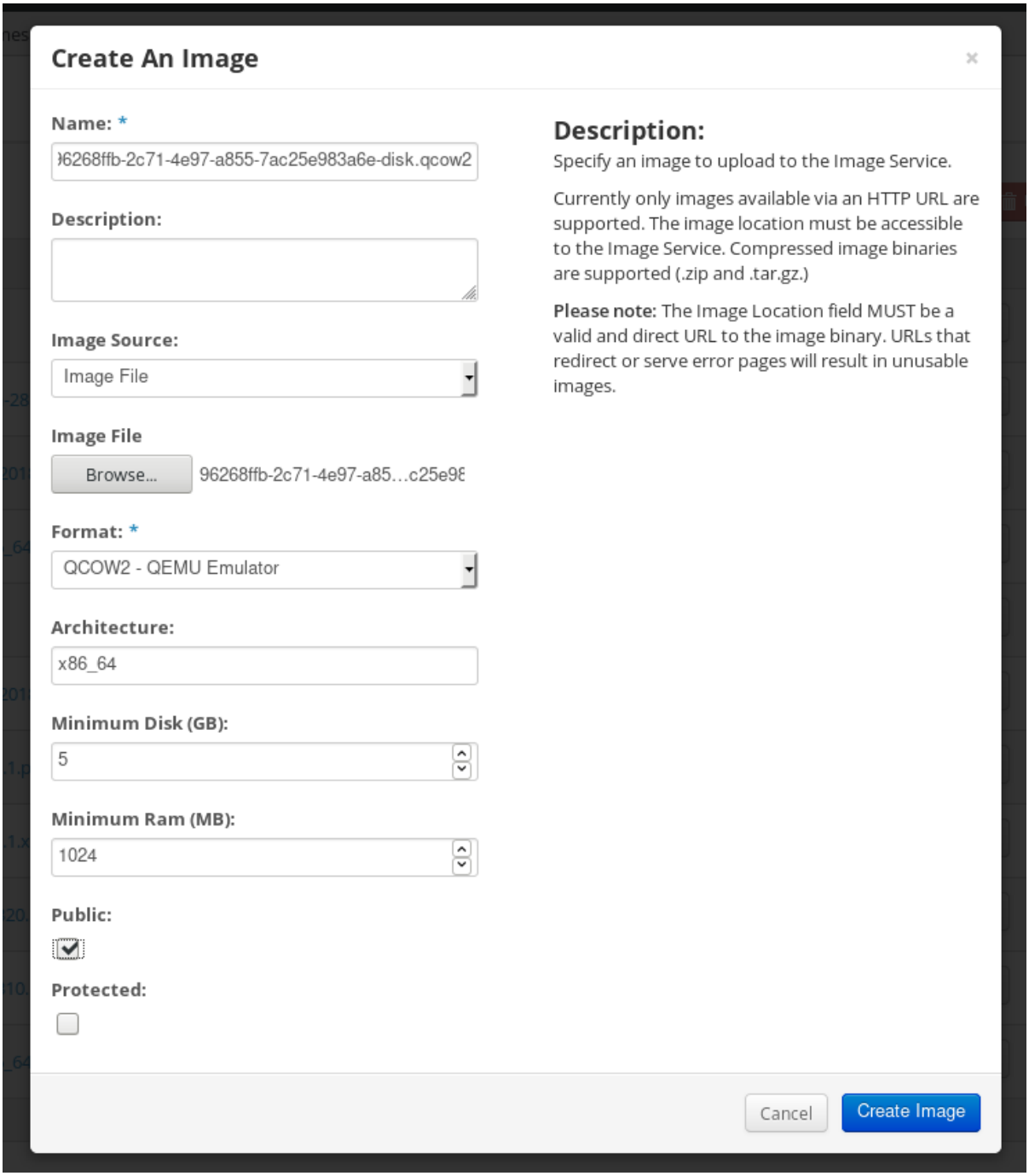
- Image Builder で OpenStack 固有のイメージを作成している。イメージの作成時に、CLI で **openstack** 出力タイプ、GUI で **OpenStack Image (.qcow2)** を使用します。

警告

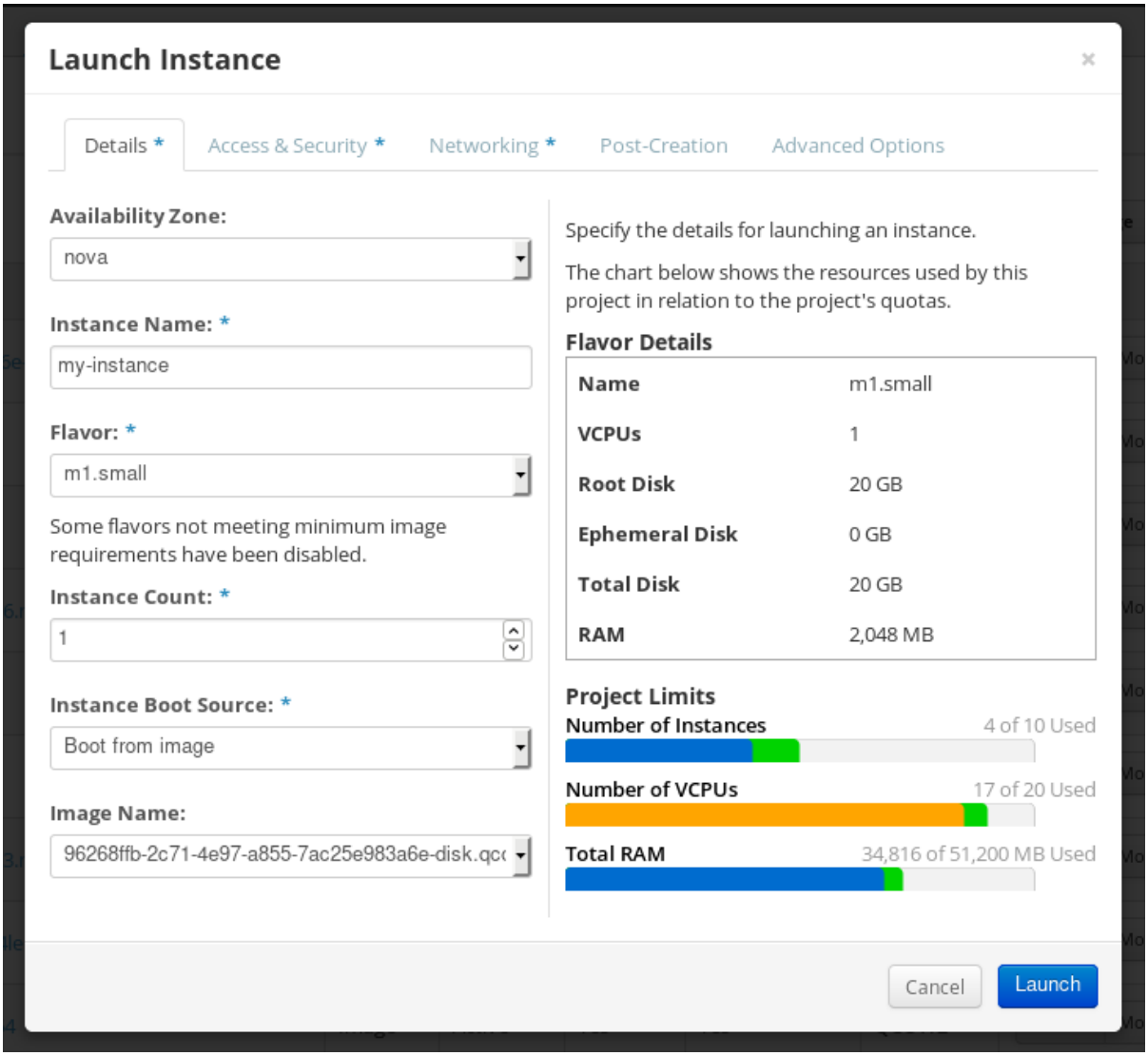
また、Image Builder は、汎用 QCOW2 イメージタイプの出力を、**qcow2** または **QEMU QCOW2 Image (.qcow2)** として提供します。これは、QCOW2 形式にある OpenStack イメージタイプとは異なります。OpenStack に固有の変更が含まれています。

手順

1. イメージを OpenStack にアップロードして、そこからインスタンスを起動します。これを行うには **Images** インターフェースを使用します。

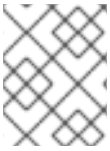


2. そのイメージでインスタンスを起動します。



3. CLI または OpenStack Web UI を使用して、スナップショットから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。

5.7. AZURE VHD イメージのアップロードの準備



注記

カスタムイメージの検証は、必要に応じて行います。Image Builder は、Alibaba の要件に準拠するイメージを生成します。

このセクションでは、Alibaba Cloud でデプロイできるカスタムイメージを検証する手順を説明します。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。このため、Alibaba 社の **image_check tool** を使用することが推奨されます。

前提条件

- Image Builder で Alibaba イメージを作成している。

手順

1. Alibaba 社の **image_check ツール** から、確認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 http://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しているフォルダーにこのレポートを保存します。

5. **検出アイテム** のいずれかが失敗した場合は、指示に従って修正します。詳細は、Detection items のセクション を参照してください。

関連情報

- 詳細は「Image Compliance Tool」 を参照してください。

5.8. ALIBABA へのイメージのアップロード

本セクションでは、Alibaba イメージを OSS (Object Storage Service) にアップロードする方法を説明します。

前提条件

- Alibaba イメージのアップロードを設定している。
- Image Builder で Alibaba イメージを作成している。イメージを作成する際に、RHEL 7 では **ami** 出力タイプ、RHEL 8 では Alibaba を使用します。
- 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 Bucket 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、**ファイル** タブをクリックします。
4. **アップロード** をクリックします。右側のウィンドウダイアログが開きます。以下の情報を選択します。

- **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. **アップロード** をクリックします。
6. アップロードするイメージを選択します。
7. **開く** をクリックします。

これにより、カスタムイメージが OSS コンソールにアップロードされます。

関連情報

- Alibaba Cloud にカスタムイメージをアップロードする方法は、「Upload an object」を参照してください。
- カスタムイメージからインスタンスを作成する方法は、「Creating an instance from custom images」を参照してください。
- Alibaba にカスタムイメージをインポートする方法は、「Importing images」を参照してください。

5.9. イメージの ALIBABA へのインポート

本セクションでは、Alibaba イメージを ECS (Elastic Cloud Console) にインポートする方法を説明します。

前提条件

- イメージを OSS (Object Storage Service) にアップロードしている。

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、**イメージ** をクリックします。
   ii. 右上の **イメージのインポート** をクリックします。ウィンドウダイアログが開きます。
   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. **Operating System**
      d. **System Disk Size**
      e. **System Architecture**
      f. **Platform** - Red Hat

iv. 必要に応じて、以下の情報を指定します。

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、左側のメニューで必要なバケットを選択して、ファイルセクションを選択してから、使用するイメージの右側にある **詳細** 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。

2. **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。

これにより、カスタムイメージが ECS コンソールにインポートされます。カスタムイメージからインスタンスを作成できます。

関連情報

- Alibaba Cloud にカスタムイメージをインポートする方法は、「Notes for importing images」を参照してください。
- カスタムイメージからインスタンスを作成する方法は、「Creating an instance from custom images」を参照してください。
- カスタムイメージからインスタンスを作成する方法は、「Upload an object」を参照してください。

5.10. ALIBABA を使用したカスタム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、カスタム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。

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、**Instances** を選択します。
3. 右上にある **Create Instance** 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入力します。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。

これにより、デプロイメントに有効なインスタンスが準備できました。

関連情報

- インスタンス作成に関する詳細は、「Creating an instance by using a custom image」 を参照してください。
- インスタンスの作成時に詳細を指定する方法は、「Create an instance by using the wizard」 を参照してください。