Red Hat Enterprise Linux 8

RHEL システムイメージのカスタマイズの作成

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

Last Updated: 2019-05-07

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

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

法律上の通知

概要

Composer 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。Composer 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。本書では、Composer を設定して、イメージを作成する方法を学びます。

目次

第1章 COMPOSER の説明

1.1. COMPOSER の概要

Composer を使用して、クラウドプラットフォームへのデプロイ用に用意したシステムイメージを含む Red Hat Enterprise Linux のカスタマイズしたシステムイメージを作成できます。Composer は、出力の各タイプに対する設定の詳細を自動的に処理して、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から Composer 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。

Red Hat Enterprise Linux 8 では、Composer は、Application Stream で **lorax-composer** パッケージのテクノロジープレビューとして利用できます。

Composer は、システムサービスの **lorax-composer** として動作します。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** (推奨される方法)
- RHEL 8 Web コンソールの GUI プラグイン

重要

Composer はテクノロジープレビューとして利用できます。詳細は 「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

1.2. COMPOSER 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を一覧表示して、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、バージョン管理されています。システムイメージから Blueprint を作成すると、イメージは、RHEL 8 Web コンソールの Composer インターフェースにある Blueprint に関連付けられます。
TOML (Tom's Obvious Minimal Language) 形式の平文テキストとして、Blueprint をユーザーに提示します。

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、その作成、入力、メタデータ、およびそのプロセス自体を指します。

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. COMPOSER の出力形式

Composer 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 Composer 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**
tar アーカイブ	**tar**	**.tar**
Amazon Machine Image (AMI) 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**

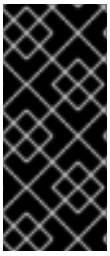
1.4. COMPOSER のシステム要件

Composer 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、潜在的に安全が保護されておらず、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 Composer を実行します。Composer 自身を実行する仮想マシンとは異なり、Red Hat コンテンツ配信ネットワーク (CDN) パッケージリポジトリーのミラーを提供する別のシステムが必要です。

Composer が実行する環境は、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 Composer 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	インターネットおよびリポジトリーのミラーが設定されたシステムに接続できること



重要

Composer はテクノロジープレビューとして利用できます。詳細は 「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

第2章 COMPOSER のインストール

Composer を使用する前に、リポジトリーミラーを設定し、仮想マシンに Composer をインストールする必要があります。

2.1. COMPOSER 用リポジトリーミラーの準備

技術上の理由から、Composer は直接 Red Hat コンテンツ配信ネットワーク (CDN) を使用することができません。以下の手順では、Composer 実行用とは別のシステムに、これらのコンテンツのミラーを準備する方法を説明します。

前提条件

- **/var/www** ディレクトリーが配置されたファイルシステムに、50 GiB 以上の空き容量がある。これを確認するには、以下のコマンドを実行します。

```
$ df -h /var/www/
```

- システムが、Composer を使用するシステムと同じバージョンの Red Hat Enterprise Linux を使用していて、必要なすべてのサブスクリプションがアタッチされている。

手順

1. パッケージとリポジトリー、そして Apache Web サーバーを操作するツールをインストールします。

```
# yum install yum-utils createrepo httpd
```

2. このマシンで有効なリポジトリーを一覧表示し、その識別子をコピーしてあとで再利用します。

```
$ sudo yum repolist
```

3. Composer で使用するリポジトリーのローカルミラーを作成します。各リポジトリーについて、以下のコマンドを実行します。

```
# mkdir -p /var/www/html
# reposync --download-path=/var/www/html --repoid REPO-ID --downloadcomps --
download-metadata
$ cd /var/www/html/REPO-ID
$ createrepo -v /var/www/html/REPO-ID -g comps.xml
```

REPO-ID を、前の手順で書き留めた識別子に置き換えます。

4. **httpd** Apache Web サーバーがリポジトリーのミラーにアクセスできるように、リポジトリーに正しい SELinux コンテキストを設定します。

```
# chcon -vR -t httpd_sys_content_t /var/www/html/
```

5. Web サーバーを設定します。再起動するたびにそれを起動し、システムのファイアウォールを設定して、サーバーへのトラフィックを許可して、初めて起動します。

```
# systemctl enable httpd
```

```
# firewall-cmd --add-service=http --permanent
# firewall-cmd --add-service=http
# systemctl start httpd
```

2.2. 仮想マシンへの COMPOSER のインストール

専用の仮想マシンに Composer をインストールするには、仮想マシンに接続して次の手順を行います。

重要

Composer はテクノロジープレビューとして利用できます。詳細は「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

前提条件

- Composer 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。
- リポジトリーミラーを持つシステムが、ネットワーク上でアクセス可能である。

手順

1. Composer および必要なパッケージをインストールします。

   ```
   # 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. このマシンで有効なリポジトリーの一覧を表示し、その識別 ID を書き留めます。

   ```
   $ sudo yum repolist
   ```

3. **/etc/yum.repos.d** ディレクトリーにリポジトリー設定ファイルを作成し、リポジトリーミラーをポイントします。仮想マシンシステムの IP アドレスまたはホスト名を指定します。各リポジトリーミラーに対して、以下のコマンドを実行します。

   ```
   # cat >> /etc/yum.repos.d/mirror.repo <<EOF
   [mirror-REPO-ID]
   name=NAME
   baseurl=http://IP-ADDR/cdrom/
   enabled=1
   gpgcheck=1
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

   EOF
   ```

   REPO-ID をリポジトリーの識別 ID に、**NAME** をリポジトリー名に、**IP-ADDR** を IP アドレスまたはホスト名に置き換えます。これにより、接頭辞 **mirror-** の付いた識別 ID で、リポジトリーミラーがシステムに提供されます。

4. リポジトリーの設定を確認します。

```
# yum clean all
# yum repolist
```

5. Red Hat コンテンツ配信ネットワークからサブスクライブされた本来のリポジトリーを無効にします。各リポジトリーミラーに対して、以下のコマンドを実行します。

```
# yum-config-manager --disable REPO-ID
```

REPO-ID を、リポジトリーの識別子に置き換えます。

6. 再起動するたびに、Composer 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。

7. 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
```

8. **composer-cli** コマンドのオートコンプリート機能が、再起動しなくても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 COMPOSER 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

Composer は、カスタムのシステムイメージを作成するツールです。Composer を制御してカスタムシステムイメージを作成するには、現在 Composer 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. COMPOSER コマンドラインインターフェース

Composer コマンドラインインターフェースは、現在 Composer 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は次のようにまとめることができます。

1. 平文テキストファイルに Blueprint 定義をエクスポート (**保存**) 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. Composer で Blueprint のテキストファイルをインポート (**プッシュ**) します。
4. compose を実行して、Blueprint からイメージを構築します。
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。

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、Blueprint と Compose の状態を調べるサブコマンドが多数あります。

composer-cli コマンドを実行するには、ユーザーは **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 COMPOSER 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新しい Composer の Blueprint を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対して、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には、**8.*** などの形式を使用します。

3. 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。
4. 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。

5. 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 info
```

6. 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 COMPOSER 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 Composer 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。Composer 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 かどうかを確認します。

4. ファイルを保存してエディターを閉じます。

5. Blueprint を Composer 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. Composer 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. コマンドラインインターフェースの COMPOSER でシステムイメージの作成

この手順では、Composer 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。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。

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。これには数分かかる場合があります。おそらく最大で約 30 分です。
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
```

完成した Compose は、ステータス値 **FINISHED** を示します。リスト内の Compose をその UUID で識別します。

3. Compose が完了したら、結果として得ら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、パス **/var/lib/lorax/composer/results/UUID/** の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. COMPOSER コマンドラインの基本的なコマンド

Composer コマンドラインインターフェースは、以下のサブコマンドを提供します。

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 の内容のファイルへの保存 (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 ファイルの Composer へのプッシュ (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
```

全 Compose、およびそのステータスの表示

```
# 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 に関する詳細情報の表示

```
# 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** コマンドとオプションに関するヘルプを提供します。

  ```
  # composer-cli help
  ```

3.6. COMPOSER の BLUEPRINT 形式

TOML (Tom's Obvious Minimal Language) 形式の平文テキストとして、Composer 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"
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。

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には、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。

結果として得ら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 = "ssh-rsa (...) key-name"
home = "/home/USER-NAME/"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = NUMBER
gid = NUMBER
```

PASSWORD-HASH を、実際の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、たとえば次の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、ユーザー名のみが必要です。

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。

結果として得ら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。

関連情報

- **lorax** Blueprint に関するアップストリームのドキュメント

3.7. 関連情報

- アップストリームのドキュメント - Composer on the Cmdline

第4章 COMPOSER 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

Composer は、カスタムのシステムイメージを作成するためのツールです。Composer を制御してカスタムシステムイメージを作成するには、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、こちらを使用することが推奨されます。

4.1. RHEL 8 WEB での COMPOSER 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、Composer の Blueprint と Compose を管理できるようになります。現在、Composer を制御するための推奨される方法は、コマンドラインインターフェースを使用することです。

重要

Composer はテクノロジープレビューとして利用できます。詳細は 「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

前提条件

- システムへの root アクセス権限がある。

手順

1. Composer 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。Composer にリモートでアクセスする方法は、『RHEL 8 の 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. システムに対して十分な権限を持つ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. Composer 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。
   Composer ビューが表示され、既存 Blueprint の一覧が表示されます。

関連情報

- *3章Composer 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、COMPOSER 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。

重要

Composer はテクノロジープレビューとして利用できます。詳細は 「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

前提条件

- ブラウザーで、RHEL 8 Web コンソールの Composer 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、続いて **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には、同じ要領で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。別の結果の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

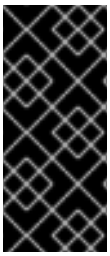
コンポーネントのバージョンを選択しない場合には、コンポーネントリストの右側の **+** ボタンをクリックし、コンポーネントの詳細ウィンドウおよびバージョンの選択をスキップできます。

4. Blueprint を保存するには、左上の **コミット** をクリックします。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。

Composer ビューが表示され、既存 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、COMPOSER 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。



重要

Composer はテクノロジープレビューとして利用できます。詳細は「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

前提条件

- ブラウザーで、RHEL 8 Web コンソールの Composer 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、同じ要領でさらに検索の用語を追加します。

2. Blueprint の左側で、その Blueprint に関する **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. 左側で、**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には、同じ要領で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 **編集** 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、左上の **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。

1. 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には、同じ要領で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。別の結果の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。

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には、コンポーネントリストの右側の **+** ボタンをクリックし、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. 左上の **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。

7. 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。
   Composer ビューが表示され、既存 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、COMPOSER の BLUEPRINT にユーザーおよびグループの追加

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現在できません。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 Composer (**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
```

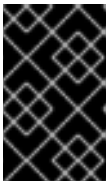
認証情報を求められたら入力してください。端末は、Web コンソールにログインする際に入力した認証情報を再利用しません。

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ための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、実際の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、たとえば次の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、ユーザー名のみが必要です。

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ます。

d. ファイルを保存してエディターを閉じます。

6. Blueprint を Composer にプッシュ (インポート) します。

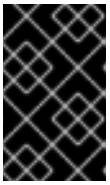
```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

.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。

7. Composer 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、ファイルに入れ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 Composer 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 Composer (**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。

その他の情報

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

4.5. WEB コンソールインターフェースで COMPOSER を使用して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。

重要

Composer はテクノロジープレビューとして利用できます。詳細は「テクノロジプレビュー機能のサポート範囲」を参照してください。

Composer をデプロイしているお客様は、Red Hat にフィードバックをお寄せください。

前提条件

- ブラウザーで、RHEL 8 Web コンソールの Composer 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。

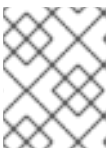
   Blueprint のリストが長すぎる場合には、同じ要領でさらに検索の用語を追加します。

2. Blueprint の左側で、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. イメージのタイプおよびアーキテクチャーを選択し、**作成** を押します。
   左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、**Pending** 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. 関連情報

- アップストリームの Weldr のドキュメント - https://weldr.io/

第5章 COMPOSER を使用したクラウドイメージの作成

Composer 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 Composer 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 account manager でアクセスキー ID を設定している。
- 書き込み可能な S3 bucket を用意しておく。

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 command line tools 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、これまでに S3 にアクセスする権限を付与していない場合は、それを行います。

```
$ 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
```

関連資料

- Weldr upstream documentation on AWS images

5.2. AZURE VHD イメージのアップロードの準備

ここでは、Azure イメージをアップロードするシステムを設定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. Azure CLI ツールをインストールします。

```
# rpm --import https://packages.microsoft.com/keys/microsoft.asc
# 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'
# yum install azure-cli
```

2. Azure CLI にログインします。

```
$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code XXXXXXXXX to authenticate.
...
```

3. 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、**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。

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

```
$ KEY1=value
```

5. 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。

関連資料

- Weldr upstream documentation on Azure images

5.3. AWS で AMI イメージの使用

ここでは、AMI イメージを AWS にアップロードするための手順を説明します。

前提条件

- AWS イメージのアップロードを設定している。
- Composer 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 にプッシュして、EC2 インスタンスを起動します。

```
$ 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 コンパイルへアップロードしたら、スナップショットとして EC2 にインポートします。

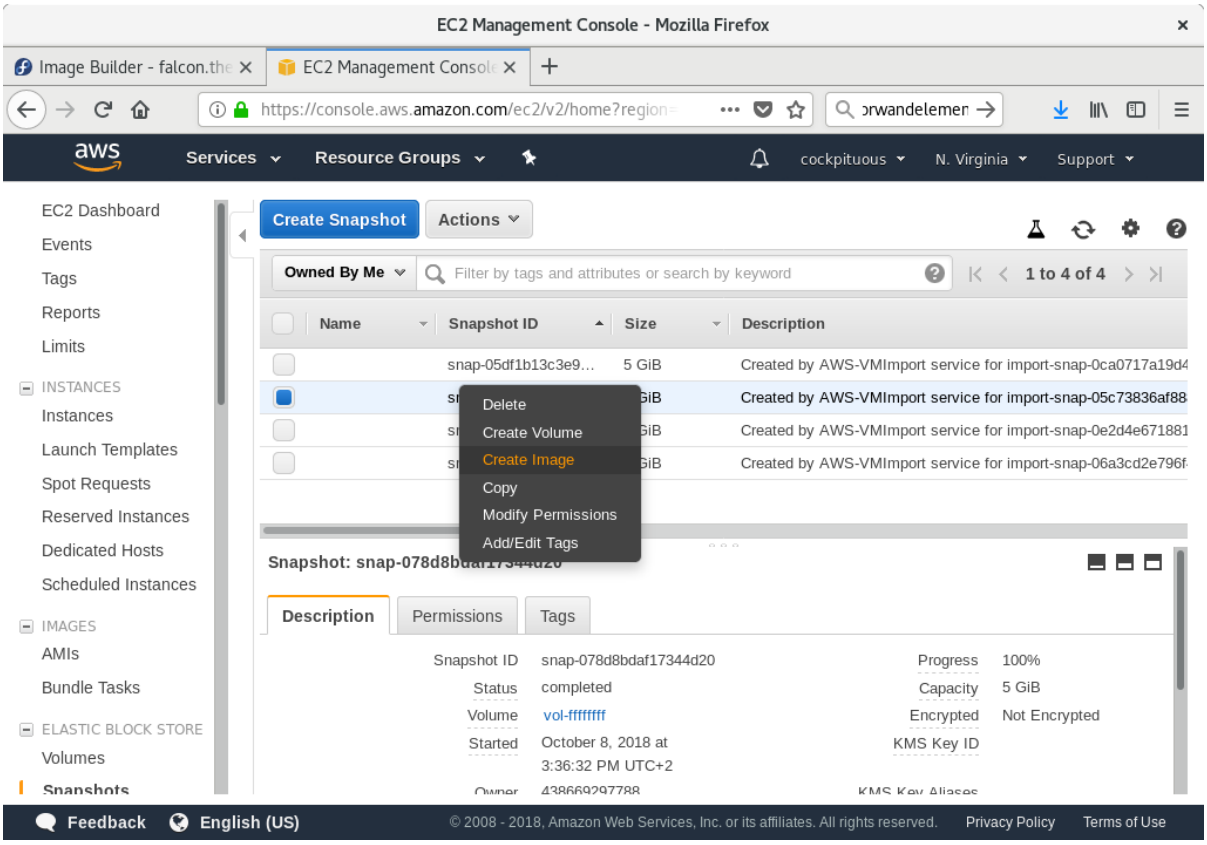
```
$ printf '{ "Description": "my-image", "Format": "raw", "UserBucket": { "S3Bucket": "%s",
"S3Key": "%s" } }' $BUCKET $AMI > containers.json
$ aws ec2 import-snapshot --disk-container file://containers.json
```

my-image を、イメージの名前に置き換えます。

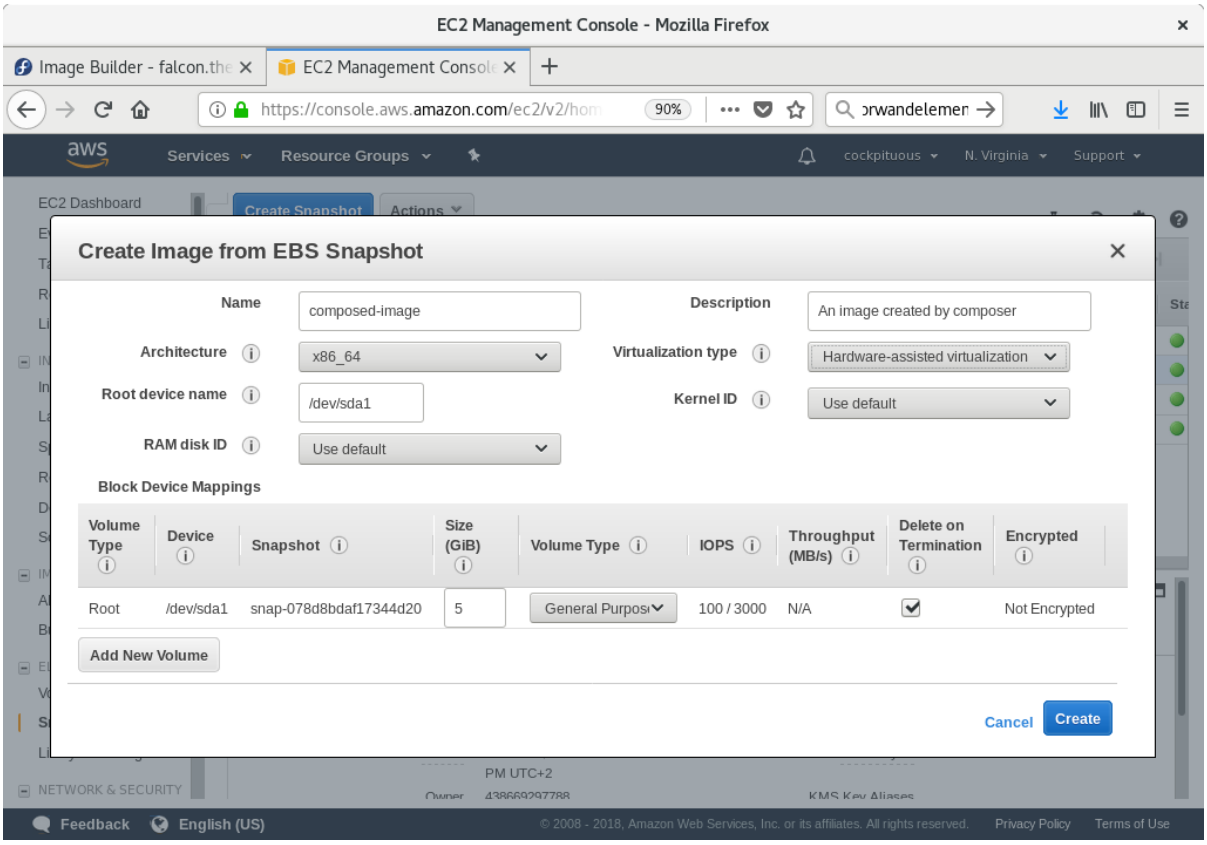
インポートの進行状況を追跡するには、次のコマンドを実行してください。

```
$ 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、**Create Image** 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、秘密鍵を使用します。**ec2-user** としてログインします。

関連資料

- Weldr upstream documentation on AWS images

5.4. AZURE で VHD イメージの使用

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

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 から結果の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。

関連資料

- Weldr upstream documentation on Azure images

5.5. VSPHERE で VMDK イメージの使用

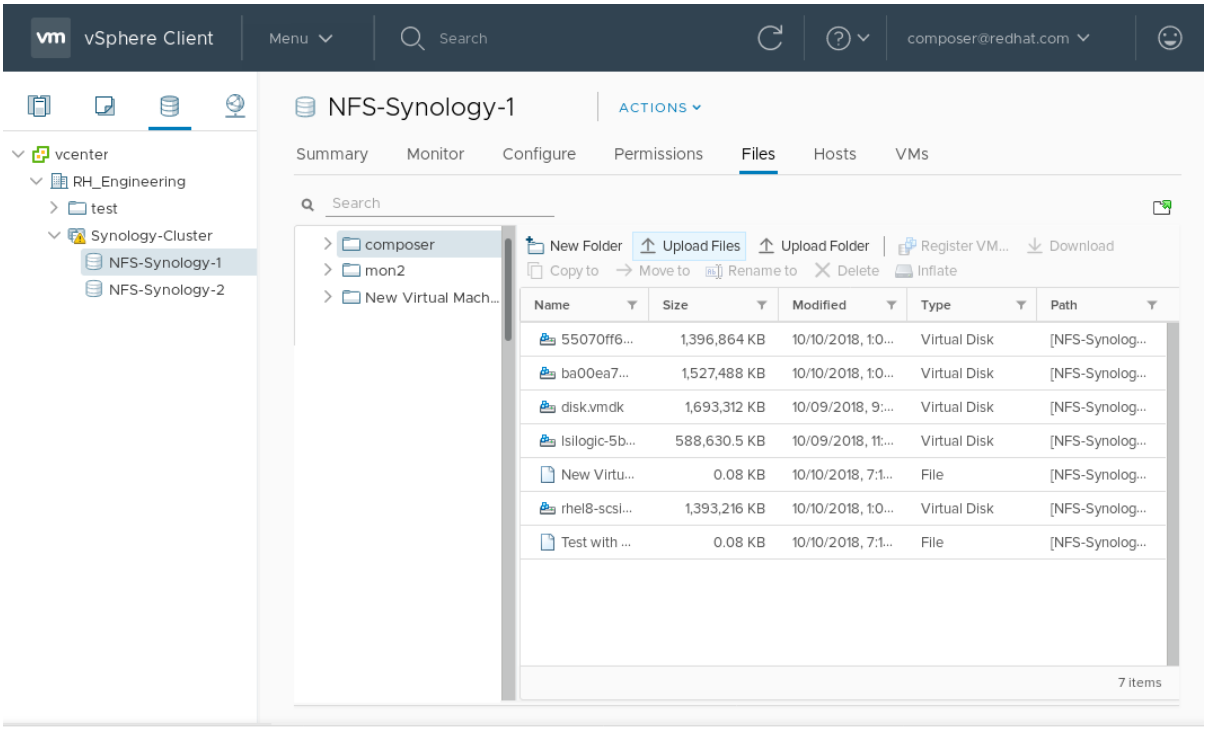
Composer は、VMware ESXi または vSphere システムへのアップロードに適したイメージを生成できます。これは、VMDK イメージを VMware vSphere にアップロードする手順を説明します。

前提条件

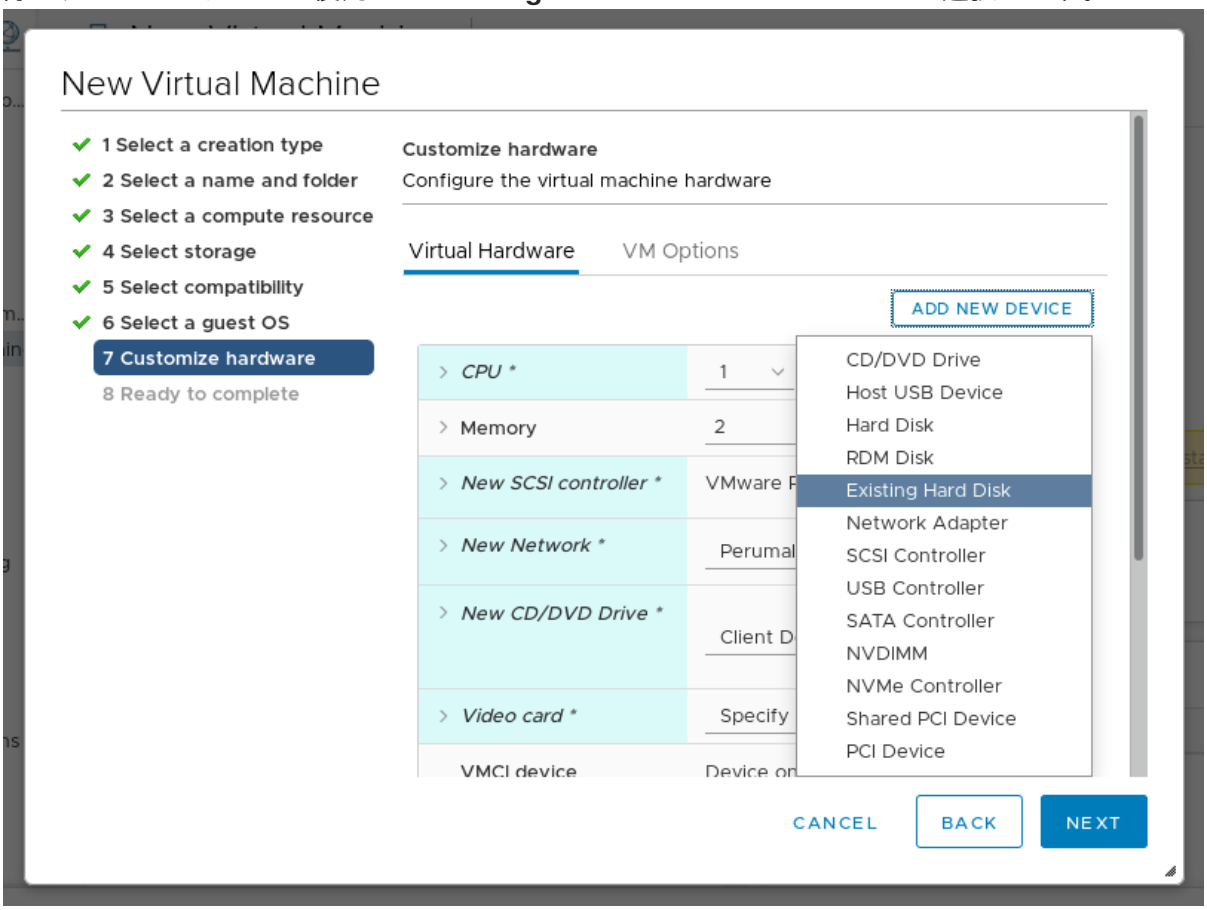
- Composer で VMDK イメージを作成している。イメージの作成時に、CLI で **vmdk** 出力タイプ、または GUI で **VMware Virtual Machine Disk (.vmdk)**を使用します。

手順

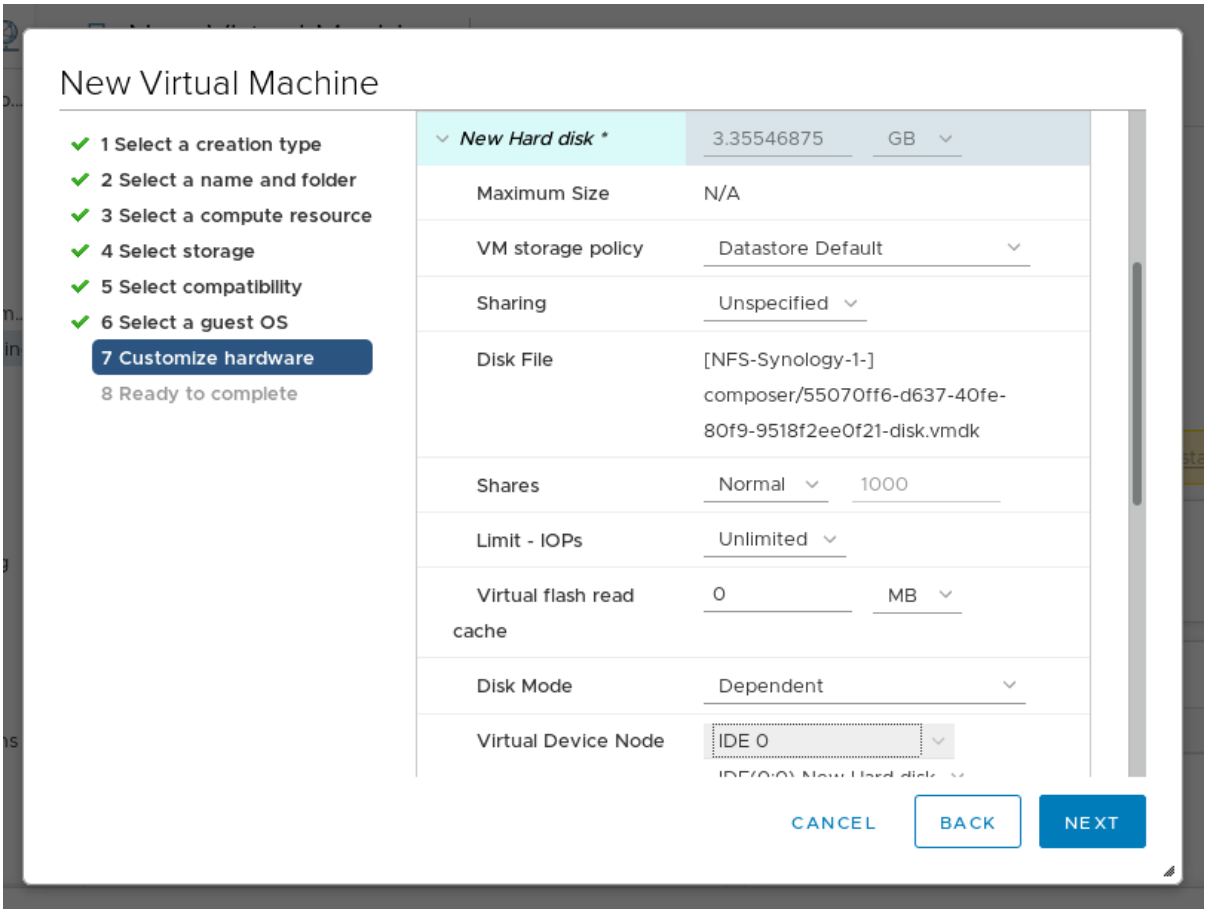
1. HTTP 経由でイメージを vSphere にアップロードします。vCenter で **Upload Files** をクリックします。



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。



関連資料

- Weldr upstream documentation on vSphere and VMware images

5.6. OPENSTACK で QCOW2 イメージの使用

Composer 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

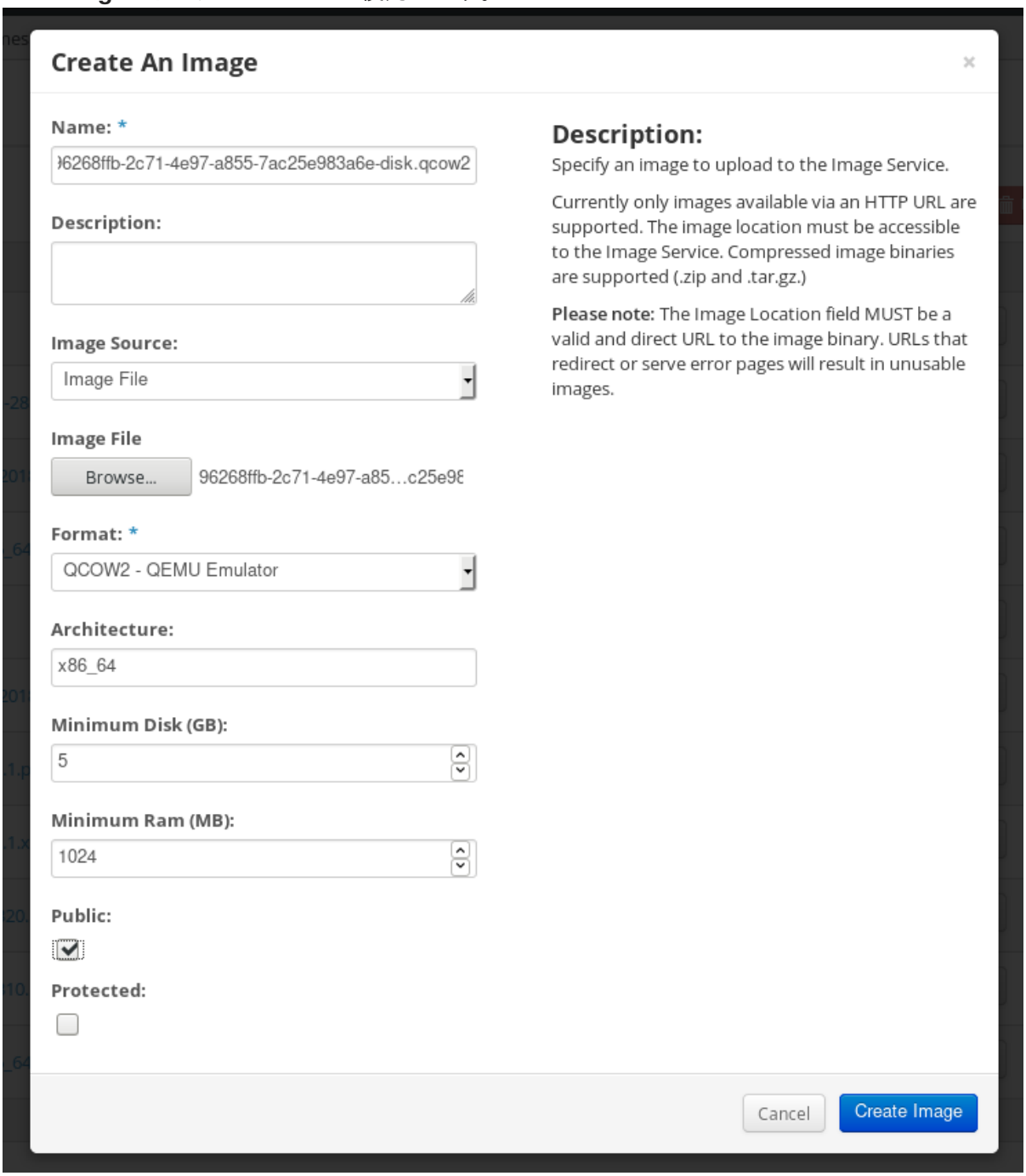
- Composer で OpenStack 固有のイメージを作成している。イメージの作成時に、CLI で **openstack** 出力タイプ、GUI で **OpenStack Image (.qcow2)** を使用します。

警告

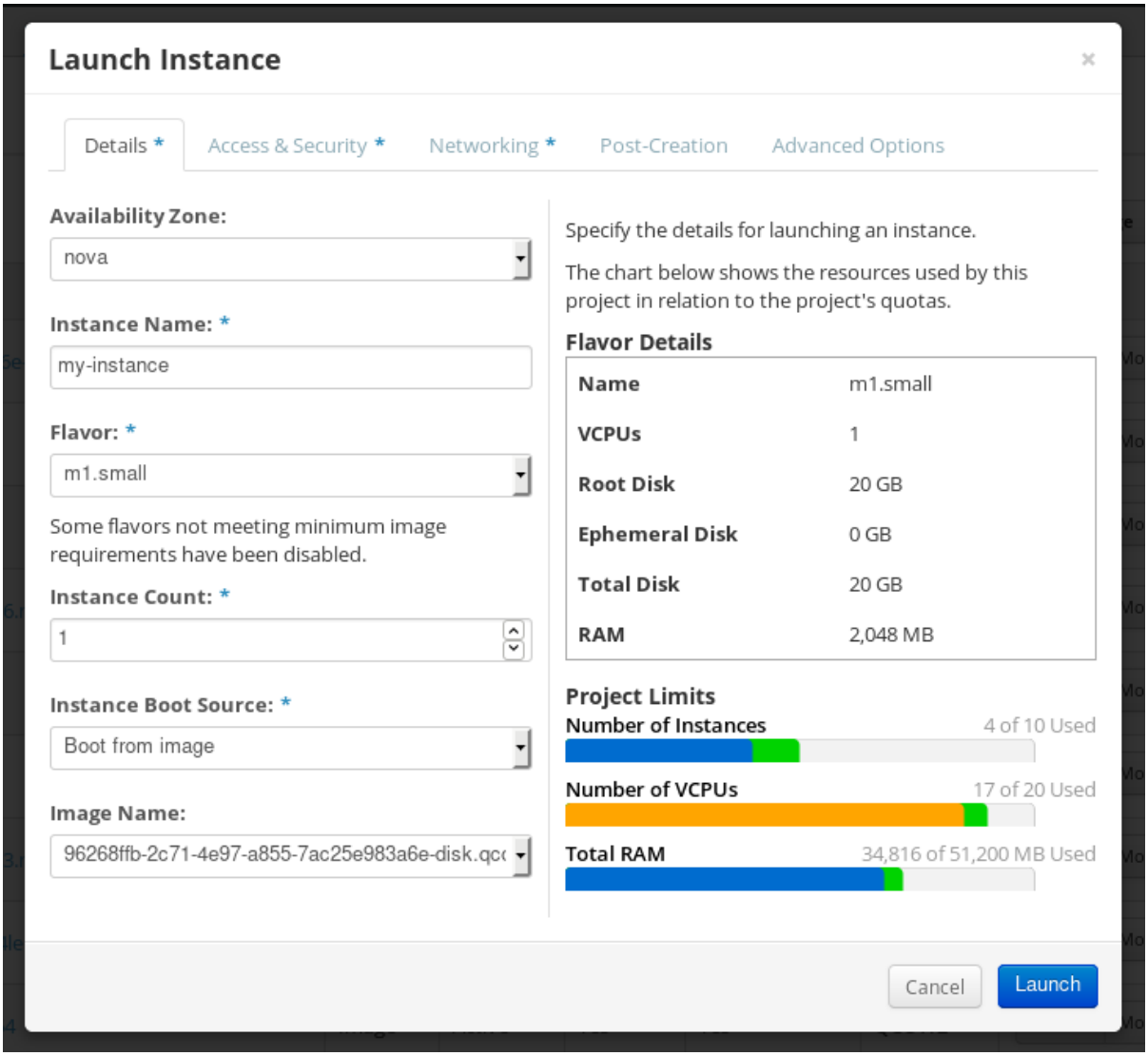
また、Composer は、汎用 QCOW2 イメージタイプの出力を、**qcow2** または **QEMU QCOW2 Image (.qcow2)** として提供します。QCOW2 形式にある OpenStack イメージタイプとは異なります。OpenStack に固有の、その他の変更が含まれています。

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 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、秘密鍵を使用します。**cloud-user** としてログインします。

関連資料

- Weldr upstream documentation on OpenStack images