Red Hat Enterprise Linux 7

Image Builder ガイド

Image Builder でのカスタムシステムイメージの作成

Last Updated: 2021-01-06

Red Hat Enterprise Linux 7 Image Builder ガイド

Image Builder でのカスタムシステムイメージの作成

Eliane Pereira
Red Hat Customer Content Services
elpereir@redhat.com

Vladimír Slávik
Red Hat Customer Content Services

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

第1章 IMAGE BUILDER の説明 3
1.1. IMAGE BUILDER の概要 3
1.2. IMAGE BUILDER の用語 3
1.3. IMAGE BUILDER の出力形式 3
1.4. IMAGE BUILDER のシステム要件 4

第2章 IMAGE BUILDER のインストール 5
2.1. 仮想マシンへの IMAGE BUILDER のインストール 5

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 6
3.1. IMAGE BUILDER コマンドラインインターフェース 6
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 6
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 7
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 8
3.5. IMAGE BUILDER コマンドラインの基本的なコマンド 9
3.6. IMAGE BUILDER の BLUEPRINT 形式 10
3.7. サポートされているイメージのカスタマイズ 11

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 15
4.1. RHEL 7 WEB コンソールで IMAGE BUILDER GUI へのアクセス 15
4.2. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT の作成 16
4.3. WEB コンソールインターフェースで IMAGE BUILDER の BLUEPRINT の編集 18
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 19
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 22

第5章 IMAGE BUILDER を使用したクラウドイメージのデプロイメント 23
5.1. アップロードする AWS AMI イメージの準備 23
5.2. AWS への AMI イメージのアップロード 24
5.3. VMDK イメージの VSPHERE へのアップロード 26
5.4. QCOW2 イメージの OPENSTACK へのアップロード 28

付録A 改訂履歴 31

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 7 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。詳細は『managing the web console』を参照してください。

Image Builder はシステムサービスの **lorax-composer** として実行されます。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**。この方法が推奨されています。
- RHEL 7 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 7 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、ユーザーアカウント、グループ、カーネル、タイムゾーン、ロケール、ファイアウォール、および ssh キーが含まれます。

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 Image	**qcow2**	**.qcow2**
Ext4 File System Image	**80**	**.qcow2**
Raw Partitioned Disk Image	**partitioned-disk**	**.img**
Live Bootable ISO	**live-iso**	**.iso**

説明	CLI 名	ファイル拡張子
TAR Archive	**tar**	**.tar**
Amazon Machine Image Disk	**ami**	**.ami**
VMware Virtual Machine Disk	**vmdk**	**.vmdk**
Openstack	**openstack**	**.qcow2**

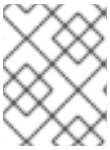
1.4. IMAGE BUILDER のシステム要件

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
System type	専用の仮想マシン
Processor	2 コア
Memory	4 GiB
Disk space	20 GiB
Access privileges	管理者レベル (root)
Network	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でのイメージ作成に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
```

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
```

cockpit および **lorax-composer** サービスは、初回アクセス時に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.**composer-cli** ツールのオートコンプリート機能が、システムを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 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。*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* 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** ツールに、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** ツールには、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 groups** に所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 = "LONGER BLUEPRINT DESCRIPTION TEXT"
version = "0.0.1"
```

BLUEPRINT-NAME および **LONGER BLUEPRINT DESCRIPTION** は、ブループリントの名前と説明に置き換えます。

0.0.1 は、Semantic Versioning スキームに従って、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 は、パッケージ名 (**httpd**、**gdb-doc**、**coreutils** など) に置き換えます。

package-version は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、**7.30** 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、**7.*** 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」セクションを参照してください。

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

4.**BLUEPRINT-NAME.toml** としてファイルを保存し、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.任意のテキストエディターで **BLUEPRINT-NAME.toml** ファイルを編集し、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。

- 次の行がある場合は削除します。

```
packages = []
```

- 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。
- 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。

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。

または、**/var/lib/lorax/composer/results/UUID/** のパス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、**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可能です。

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
```

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** ツールは、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 compose list
  ```

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 を保存します。

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 = "LONGER BLUEPRINT DESCRIPTION"
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。

modules エントリーは、パッケージ名と、イメージにインストールするバージョンと一致する glob を示し、**group** では、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記述します。これらの項目を追加しない場合、Blueprint はそれらを empyt リストとして識別します。

イメージに含まれるパッケージ

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

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

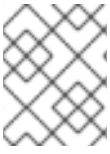
package-version は使用するバージョンに置き換えます。このフィールドは、**dnf** バージョンの指定に対応します。

- 特定のバージョンを指定する場合は、**7.30** 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、**7.*** 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

Blueprint では、多くのイメージのカスタマイズがサポートされています。このオプションを使用するには、最初に Blueprint でオプションを設定し、コマンド push を使用して、変更した Blueprint を Image Builder にインポートする必要があります。



注記

これらのカスタマイズは、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
```

ユーザー名だけが必要です。他の行はそのままにできます。

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())'
```

重要

ハッシュを生成するには、システムに **python3** パッケージが必要です。パッケージをインストールするには、次のコマンドを使用します。

```
# yum install python3
```

PUBLIC-SSH-KEY を、実際の公開鍵に置き換えます。

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 キーの設定

```
[[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。

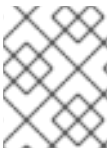
システムの起動時に有効にするサービスの設定

```
[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。

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 を使用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、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。



注記

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 7 WEB コンソールで IMAGE BUILDER GUI へのアクセス

RHEL 7 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。

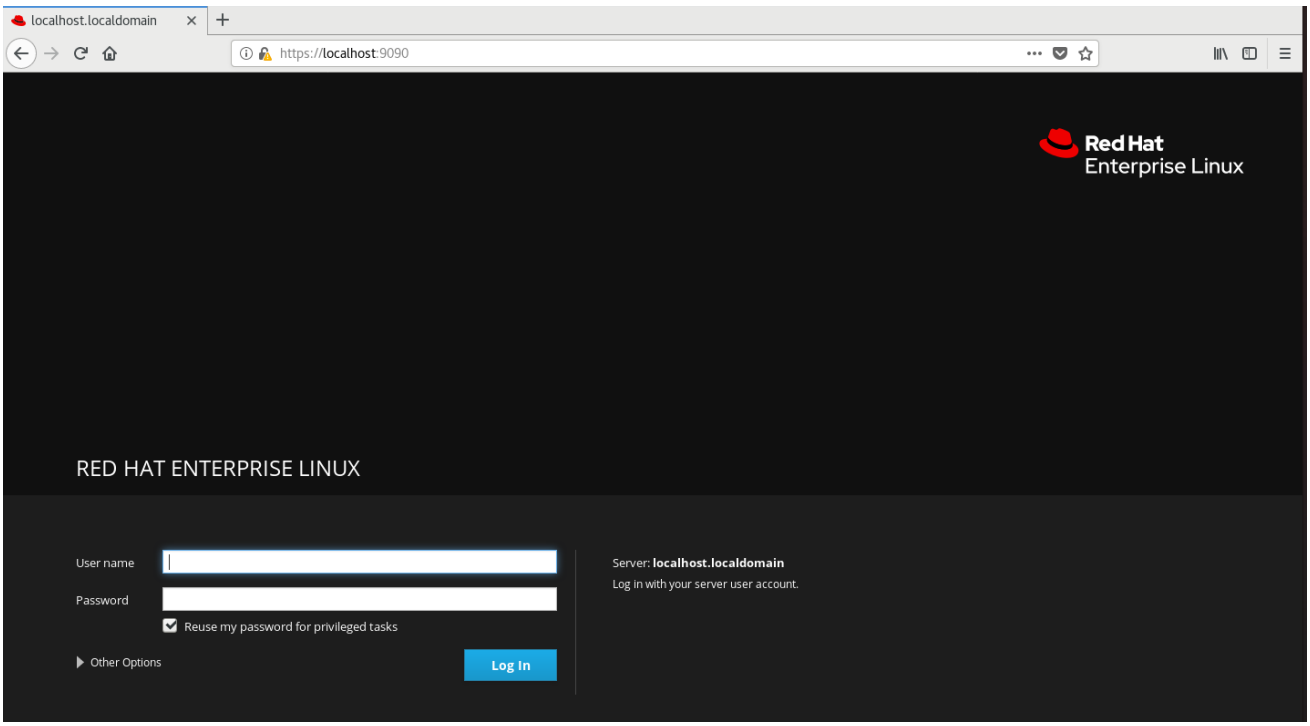
前提条件

- 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システムの Web ブラウザーで **https://localhost:9090/** を開きます。

図4.1 Web コンソールへのログイン



[D]

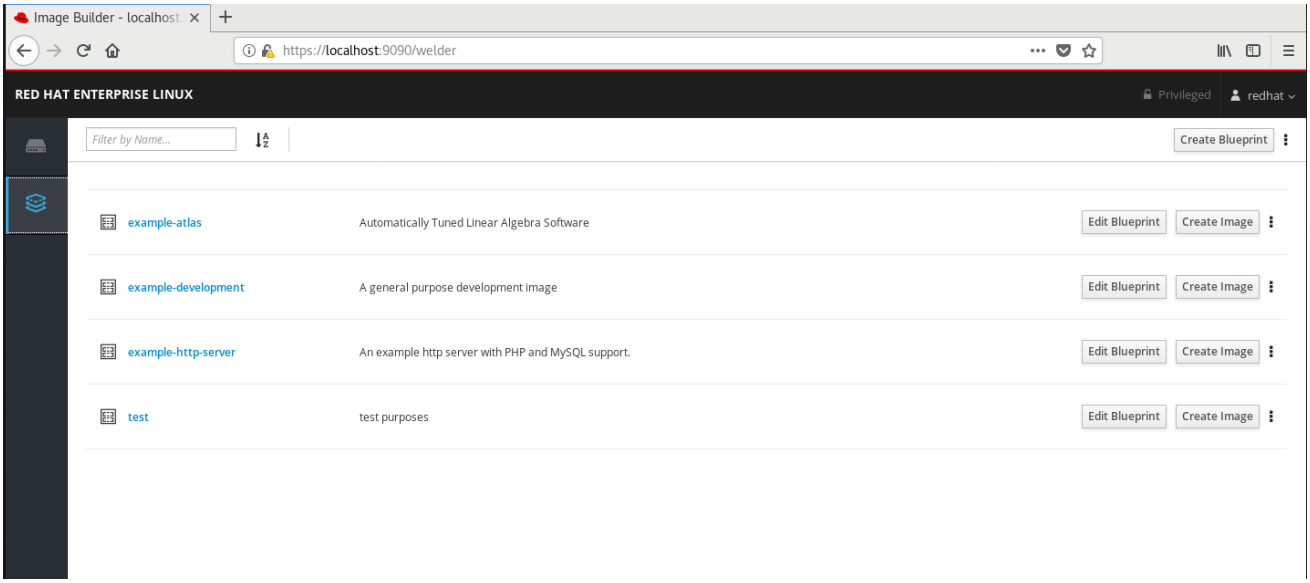
Image Builder にリモートでアクセスする方法は、『managing systems using the RHEL 7 web console』を参照してください。

2.root のユーザー名およびパスワードを使用して 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。

図4.2 Web コンソールの Image Builder



[D]

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*
- Image Builder シナリオ を使用してカスタマイズされた RHEL OS イメージを作成できます。

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。

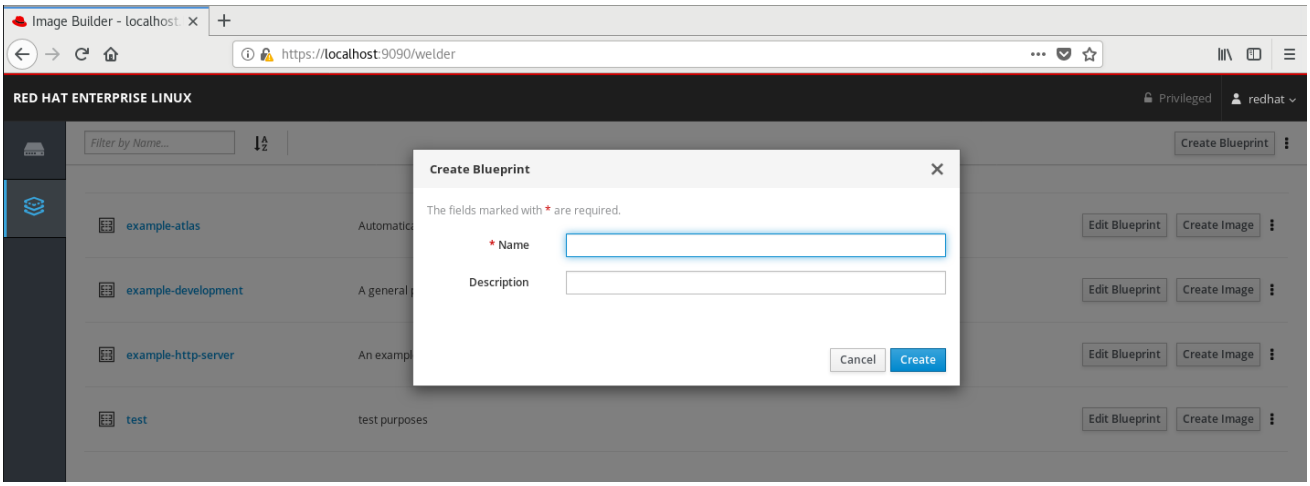
前提条件

- ブラウザーで、RHEL 7 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。

図4.3 Blueprint の作成



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。

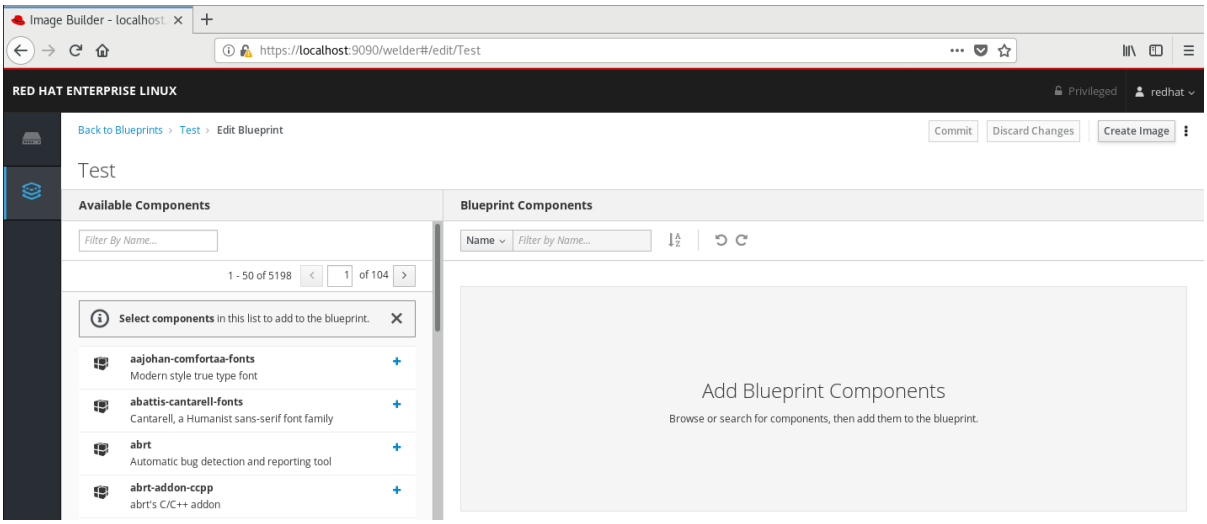
2.Blueprint の名前およびその説明を入力したら、作成をクリックします。

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。

- i.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。

  図4.4 利用可能なコンポーネントの検索

  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。

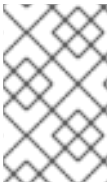
  コンポーネントのリストが長すぎる場合は、さらに検索の用語を追加します。

- ii.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。

- iii.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。

- iv.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。

- v.左上の追加をクリックします。
- vi.誤ってコンポーネントを追加してしまった場合には、右側のペインでエントリーの右端にある ... ボタンをクリックし、メニューで **削除** を選択します。



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、右上の**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。

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 7 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 **Remove** 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。

- i.Blueprint コンポーネント検索フィールドで、**Blueprint Components** 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。

- ii.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 **View** 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。

- iii.**Version Release** 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。

- i.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。

- ii.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。

- iii.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。

- iv.**Version Release** ドロップダウンメニューから **Component Options** ボックスで使用するバージョンを選択します。

- v.右上の **追加** をクリックします。

- vi.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 **Remove** 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。

1.変更を加えた新しいバージョンの Blueprint をコミットします。

- i.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。

- ii.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。

- iii.左上の **Back to Blueprints** をクリックして編集画面を終了します。

  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** または **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 7 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 コンソールで CLI に移動します。左側でシステム管理タブを開いて、左側の一覧にある最後の項目 **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 7 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。

- i.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。

```
[[customization.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、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。

- ii.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
```

iii.バージョン番号を大きくしてください。

iv.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。

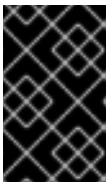
```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

.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 7 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 7 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.イメージのタイプおよびアーキテクチャーを選択し、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。

重要

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。

第5章 IMAGE BUILDER を使用したクラウドイメージのデプロイメント

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. アップロードする AWS AMI イメージの準備

本セクションでは、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 bucket を準備する必要があります。

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 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 に **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
```

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。

```
$ A printf '{ "Description": "my-image", "Format": "raw", "UserBucket": { "S3Bucket": "%s", "S3Key": "%s" } }' $BUCKET $AMI > containers.json
```

```
$ aws ec2 import-snapshot  disk-container file://containers.json
```

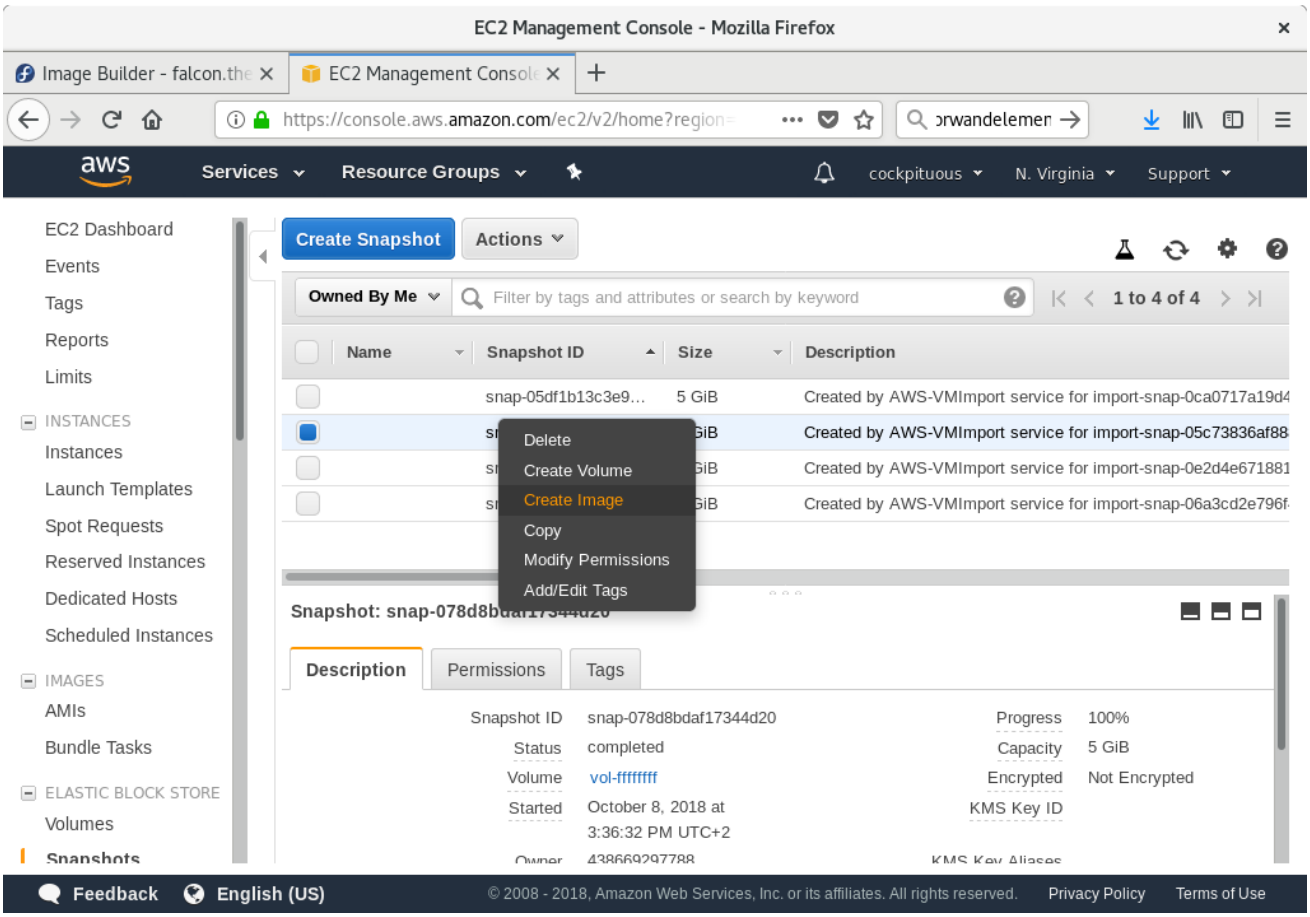
my-image を、イメージ名に置き換えます。

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
```

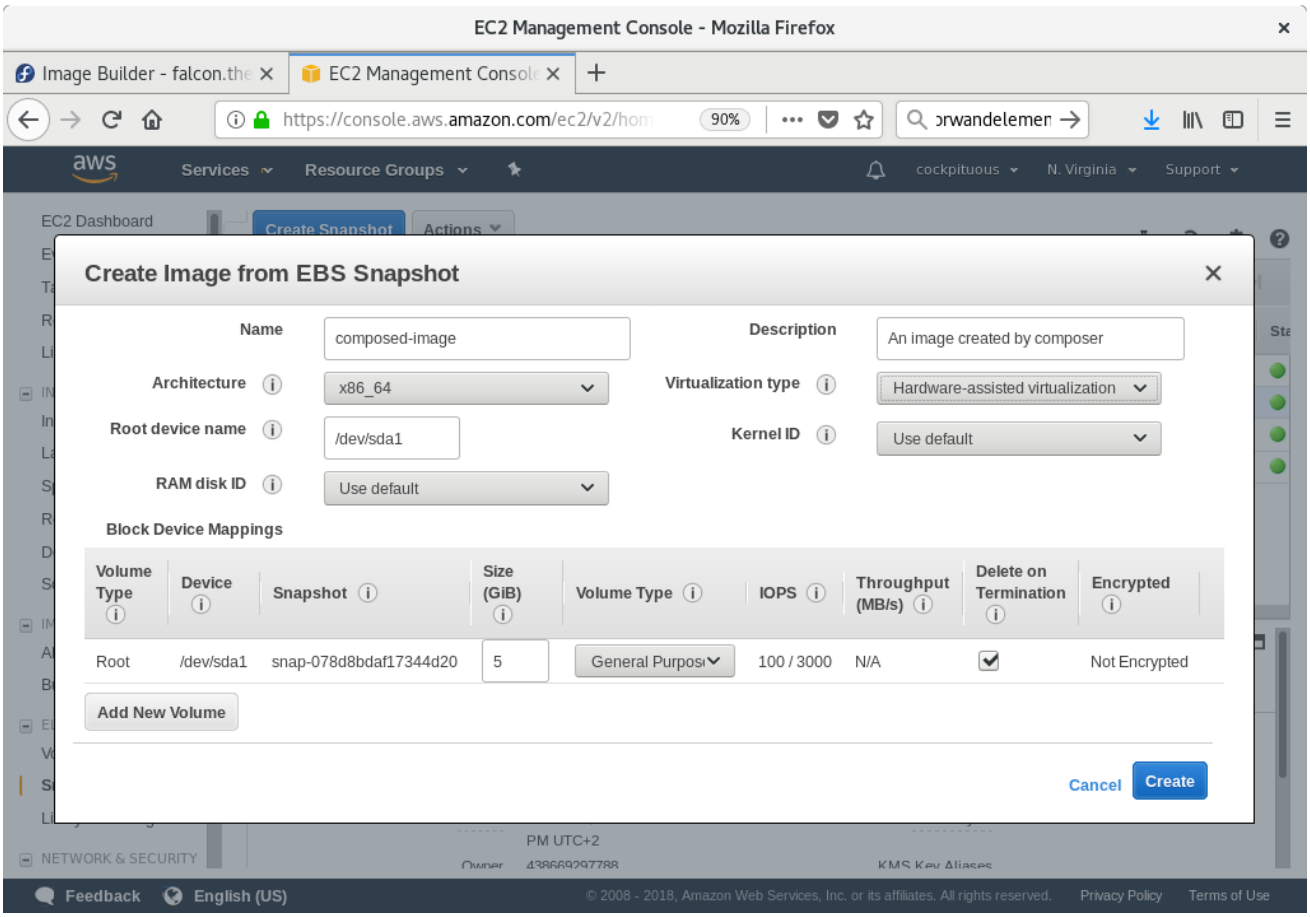
3.EC2 コンソールでスナップショットを選択して右クリックし、イメージの作成を選択して、アップロードしたスナップショットからイメージを作成します。

図5.1 イメージの作成



4.作成するイメージで、ハードウェア仮想化支援機能の仮想化タイプを選択します。

図5.2 仮想化のタイプ



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. 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。本項では、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

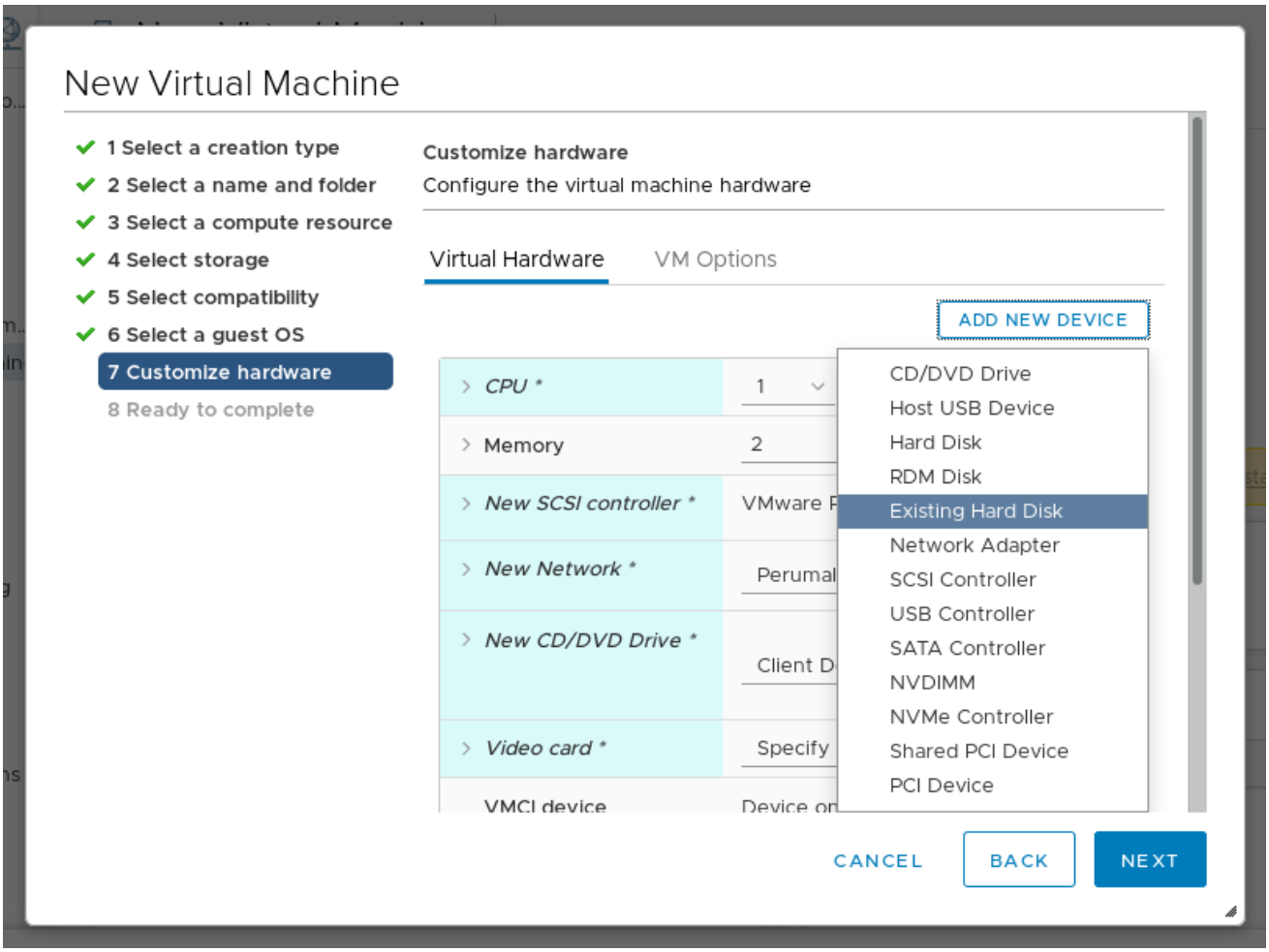
- Image Builder で VMDK イメージを作成している。イメージの作成時に、CLI で vmdk 出力タイプ、または GUI で VMware Virtual Machine Disk (.vmdk) を使用します。

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。

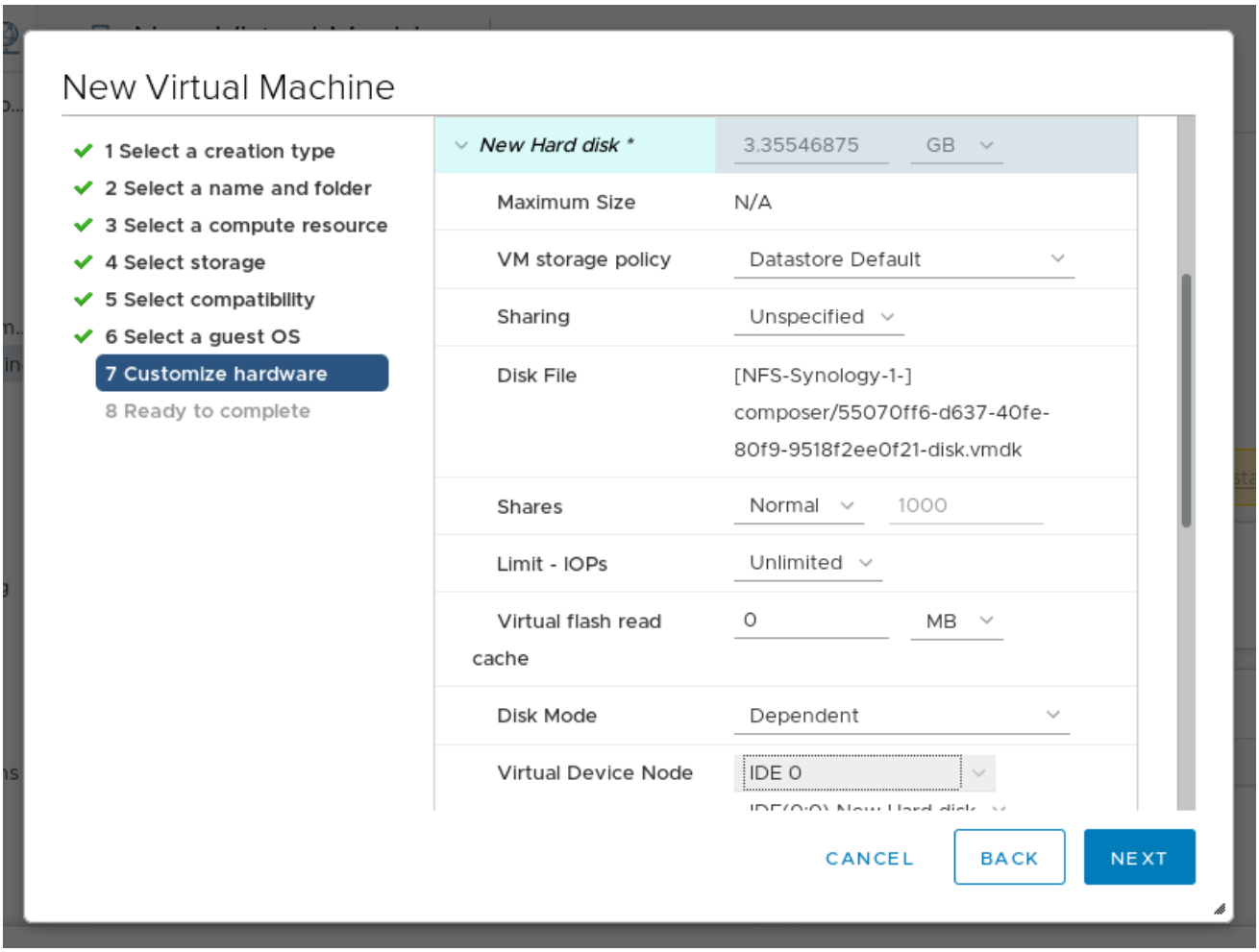
2.仮想マシンを作成する場合は、Device Configuration で、デフォルトの New Hard Disk を削除し、ドロップダウンを使用して Existing Hard Disk ディスクイメージを選択します。

図5.3 仮想化のタイプ



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。

図5.4 仮想化のタイプ



5.4. 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。

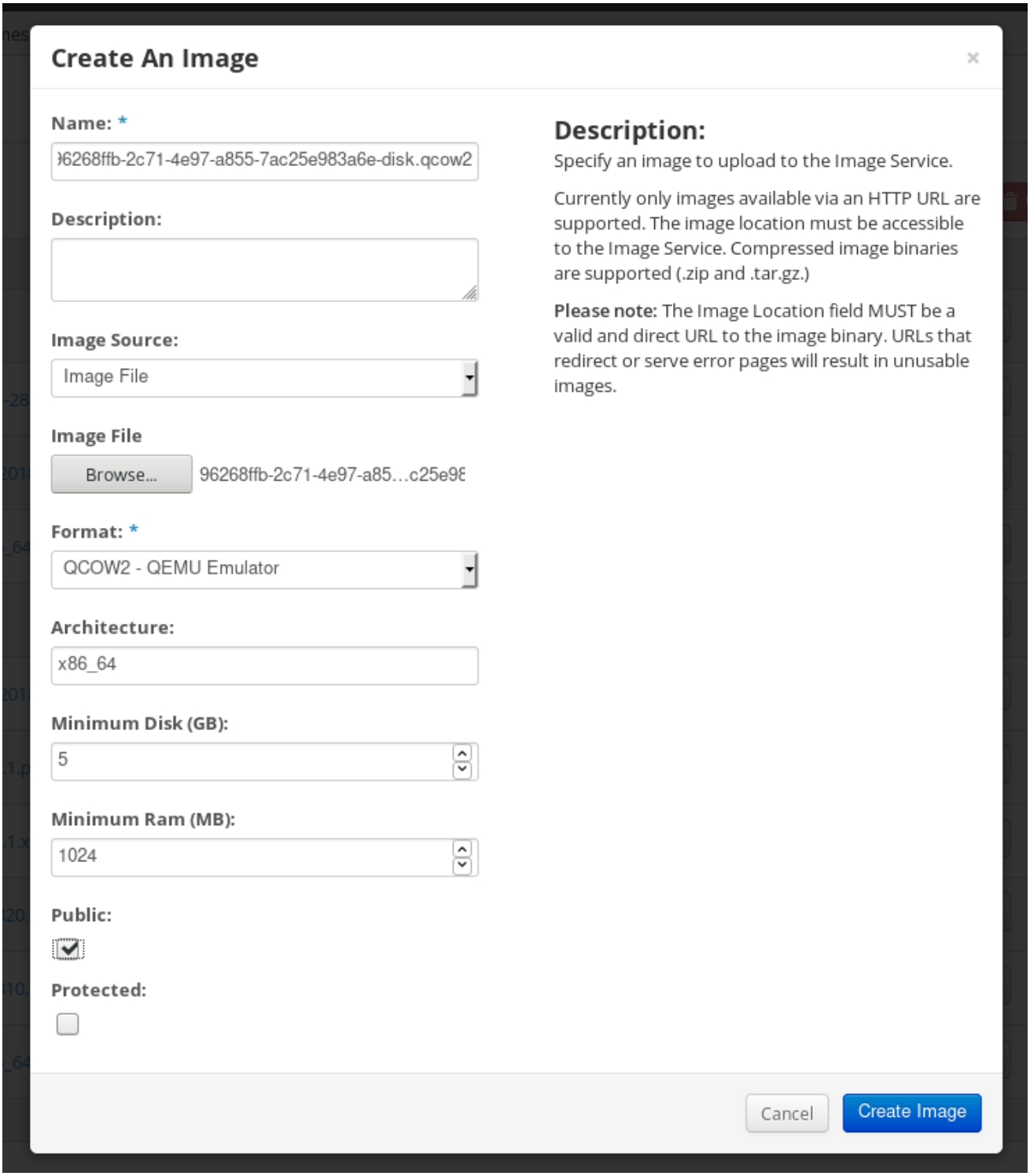
警告

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

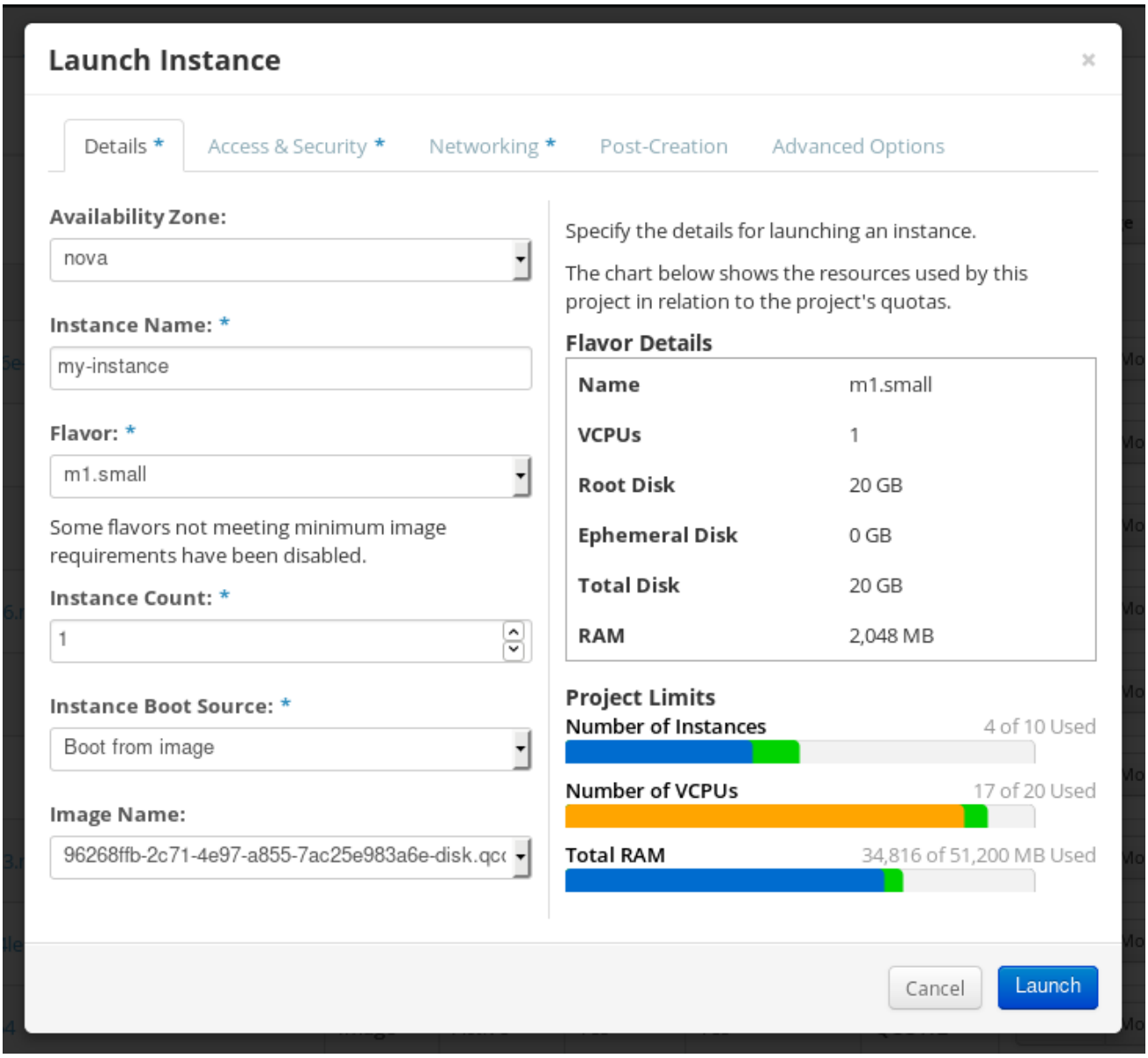
1.イメージを OpenStack にアップロードして、そこからインスタンスを起動します。これを行うには Images インターフェースを使用します。

図5.5 仮想化のタイプ



2.そのイメージでインスタンスを起動します。

図5.6 仮想化のタイプ



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。

付録A 改訂履歴

改訂 1.1.0- **Tue Aug 06 2019** **Eliane Pereira**
改訂 1.0.0- **Thu Jul 11 2019** **Eliane Pereira**
Composer に関する記述を別のガイドに移動

改訂 0.0-0 **Sun Jun 2 2019** **Eliane Pereira**
7.7 GA 公開用ドキュメントの準備