Red Hat Enterprise Linux 9

RHEL システムイメージのカスタマイズ

Red Hat Enterprise Linux 9 の Image Builder でシステムイメージのカスタマイズ

Last Updated: 2022-08-08

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

Red Hat Enterprise Linux 9 の Image Builder でシステムイメージのカスタマイズ

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

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、各出力タイプに必要な特定の設定が削除されるため、手動の手順と比較すると、このイメージを作成できます。本書では、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

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージをご覧ください。

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。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
    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隅に **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。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
    1. Bugzilla の Web サイトに移動します。
    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 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。

osbuild-composer バックエンドは、Red Hat Enterprise Linux 8.3 で **lorax-composer** に代わります。新しいサービスには、イメージビルド向けの REST API が含まれます。その結果、ユーザーはより信頼性の高いバックエンドと予測可能な出力イメージの利点を活用できます。

Image Builder は、システムサービスの **osbuild-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**。(この手法を推奨)。
- RHEL 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 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 形式のプレーンテキストとしてユーザーに提示されます。

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。サポートされているタイプを確認するには、次のコマンドを実行します。

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

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**

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**	**.raw**
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**
RHEL for Edge のコミット	**edge-commit**	**.tar**
RHEL for Edge コンテナー	**edge-container**	**.tar**
RHEL for Edge インストーラー	**edge-installer**	**.iso**
RHEL for Edge Raw	**edge-raw-image**	**.tar**
RHEL for Edge Simplified Installer	**edge-simplified-installer**	**.iso**
ISO イメージ	**image-installer**	**.iso**

1.4. IMAGE BUILDER のシステム要件

Image Builder を実行する環境 (専用の仮想マシンなど) 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	**/var** ファイルシステムで 20 GiB の空き領域
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

インターネット接続は前提条件ではありません。Red Hat CDNに接続しないように再構成すると、分離されたネットワークでImageBuilderを使用できます。

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 の仮想マシンがインストールされ、RHSM または Red Hat Satellite にサブスクライブされ、実行している必要があります。

手順

1. Image Builder およびその他の必要なパッケージを仮想マシンにインストールします。
   - **osbuild-composer** - RHEL 8.3 以降でサポート
   - **composer-cli**
   - **cockpit-composer**
   - **bash-completion**

   ```
   # dnf install osbuild-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 --now osbuild-composer.socket
   # systemctl enable --now cockpit.socket
   ```

   osbuild-composer 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. 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
   ```

重要

osbuild-composer パッケージは、Red Hat Enterprise Linux 8.3 以降の新機能すべてに焦点を当てた新しいバックエンドエンジンで、デフォルト設定として推奨されています。以前のバックエンドの **lorax-composer** は非推奨となり、Red Hat Enterprise Linux 8 ライフサイクルの残りの期間、一部の修正のみを受信し、今後のメジャーリリースから削除される予定です。osbuild-composer を優先するには、**lorax-composer** のアンインストールを推奨します。

第3章 リポジトリの管理

3.1. IMAGEBUILDERのデフォルトのシステムリポジトリ

osbuild-composerバックエンドは、**/etc/yum.repos.d/**にあるシステムのリポジトリを継承しません。代わりに、**/usr/share/osbuild-composer/repositories**ディレクトリに定義された独自の公式リポジトリのセットがあります。公式リポジトリをオーバーライドするには、**/etc/osbuild-composer/repositories**オーバーライドを定義する必要があります。このディレクトリはユーザー定義のオーバーライド用であり、ここにあるファイルは**/usr**ディレクトリ内のファイルよりも優先されます。

設定ファイルは**/etc/yum.repos.d/**内のファイルから知られている通常のDNFリポジトリ形式ではありません。それらは単純なJSONファイルになります。

3.2. システムリポジトリのオーバーライド

以下の手順により、**/etc/osbuild-composer/repositories** ディレクトリーでリポジトリーの上書きを設定できます。

前提条件

- ホストシステムからアクセスできるカスタムリポジトリがあります

手順

1. 使用するリポジトリオーバーライドを含むディレクトリを作成します。

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

2. たとえば、次の構造のJSONファイルを作成します。

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

metalink、**mirrorlist**、または **baseurl** 属性の中から1つのみを指定してください。残りのフィールドは任意です。

3. RHEL バージョンに対応する名前を使用して JSON ファイルを保存します。以下に例を示します。

```
/etc/osbuild-composer/repositories/rhel-90.json
```

a. または、ディストリビューションの JSON ファイルを **/usr/share/osbuild-composer/** からコピーして、そのコンテンツを変更できます。

i. 作成したディレクトリにリポジトリファイルをコピーします。

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

rhel-version.json を RHEL バージョンに置き換えます（例： rhel-9.json）。

4. テキストエディターを使用して、**rhel-9.json** ファイルの **baseurl** パスを編集します。以下は例になります。

```
$ vi /etc/osbuild-composer/repositories/rhel-9.json
```

その結果、リポジトリは**/etc/yum.repos.d/redhat.repo**ファイルからコピーされた正しいURLを指します。

3.3. サブスクリプションをサポートするシステムリポジトリのオーバーライド

osbuild-composerサービスは**/etc/yum.repos.d/redhat.repo**ファイルで定義されているシステムのサブスクリプションを使用することができます。**osbuild-composer**でシステムサブスクリプションを使用するには、次のようなリポジトリオーバーライドを定義する必要があります。

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

前提条件

- **/etc/yum.repos.d/redhat.repo**で定義されたサブスクリプションを持つシステム
- リポジトリオーバーライドを作成しました。システムリポジトリのオーバーライドを参照してください。

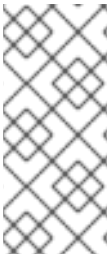
手順

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

```
[AppStream]
name = AppStream mirror example
baseurl = https://mirror.example.com/RHEL-9/9.0/AppStream/x86_64/os/
enabled = 1
gpgcheck = 0
sslverify = 1
sslcacert = /etc/pki/ca1/ca.crt
sslclientkey = /etc/pki/ca1/client.key
sslclientcert = /etc/pki/ca1/client.crt
metadata_expire = 86400
enabled_metadata = 0
```

2. 同じ**baseurl**を使用するようにリポジトリオーバーライドを構成し、**rhsm**をtrueに設定します。

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



注記

osbuild-composerは、自動的に**/etc/yum.repos.d**/で定義されたリポジトリを使用していません。システムリポジトリのオーバーライドとして、**composer-cli**を使用して追加の**source**として手動で指定する必要があります。システムリポジトリのオーバーライドは通常、「BaseOS」および「AppStream」リポジトリに使用されますが、**composer-cli**ソースは他のすべてのリポジトリに使用されます。

関連情報

- ホストが Satellite6 に登録されている場合、Composer イメージビルダーは CDN リポジトリーを使用する

第4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。

4.1. 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。

4.2. 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については、「サポートされているイメージのカスタマイズ」 を参照してください。

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

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

注記

root 以外のユーザーとして **composer-cli** コマンドを使用してイメージを作成できます。これには、ユーザーを **weldr** または **root** グループに追加します。**weldr** グループにユーザーを追加するには、以下の手順を実行します。

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

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

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

4.4. 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 は、前の手順で示した UUID 値に置き換えます。

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。

4.5. 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

利用可能なイメージタイプを一覧表示します。

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

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

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
```

4.6. IMAGE BUILDER の BLUEPRINT 形式

Image Builder の Blueprint は、TOML 形式のプレーンテキストとしてユーザーに提示されます。

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は、**8.*** などの形式を使用してください。

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。

4.7. 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 でカスタマイズを設定して、Image Builder にインポート（プッシュ）する必要があります。

注記

これらのカスタマイズは、現在 Web コンソール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
```

注記

GID は任意で、イメージにすでに存在している必要があり、パッケージにより作成されるか、または Blueprint **[[customizations.group]]** エントリーで作成されます。

重要

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

```
# dnf 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。

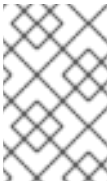
作成されるシステムイメージに対するグループ指定

```
[[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 キーを設定するには、このセクションで **生成されるシステムイメージのカスタマイズのユーザー仕様** を参照してください。

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"
```

デフォルトでは、Image Builder はデフォルトカーネルをイメージにビルドします。ただし、Blueprint で以下の設定でカーネルをカスタマイズできます。

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

イメージで使用するカーネル名を定義します。

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

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]
languages = ["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、一覧を有効化できます。

ファイアウォールサービスのカスタマイズ

利用可能なファイアウォールサービスを確認します。

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

Blueprint のセクションの **customizations.firewall.service** で、カスタマイズするファイアウォールサービスを指定します。

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

firewall.services に一覧表示されるサービスは、**/etc/services** ファイルで利用可能な名前とは異なります。

オプションで、作成する予定のシステムイメージのファイアウォールサービスをカスタマイズできます。

注記

ファイアウォールサービスをカスタマイズしない場合は、Blueprint の **[customizations.firewall]** セクションおよび **[customizations.firewall.services]** セクションを省略します。

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。

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 パッケージの作成後に削除されます。

カスタムファイルシステム設定を指定します。

ブループリントでカスタムファイルシステム構成を指定して、デフォルトのレイアウト構成を使用する代わりに、特定のディスクレイアウトでイメージを作成できます。ブループリントでデフォルト以外のレイアウト構成を使用すると、次の利点が得られます。

- セキュリティベンチマークコンプライアンス
- ディスク外エラーに対する保護
- performance
- 既存の設定との一貫性
  ブループリントのファイルシステム構成をカスタマイズします。

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

  次の**mountpoints**とそのサブディレクトリがサポートされています。

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

- **/app**
- **/data**

注記

マウントポイントのカスタマイズは、CLI を使用した RHEL8.5 および RHEL9.0 ディストリビューション以降でのみサポートされます。初期のディストリビューションでは、**root**パーティションをマウントポイントとして指定し、**size**引数をイメージサイズのエイリアスとして指定することしかできません。

複数のパーティションがある場合は、LVM にカスタマイズしたファイルシステムのパーティションでイメージを作成し、ランタイム時にそのパーティションのサイズを変更できます。そのために、ブループリントでカスタマイズされたファイルシステム設定を指定してから、目的のディスクレイアウトでイメージを作成できます。デフォルトのファイルシステムレイアウトは変更されません。ファイルシステムをカスタマイズせずにプレーンイメージを使用する場合、ルートパーティションは **cloud-init** によってサイズ変更されます。

Blueprint にファイルシステムのカスタマイズを追加すると、ファイルシステムが LVM パーティションに変換されます。

MINIMUM-PARTITION-SIZE を定義する場合、デフォルトのサイズ形式はありません。kB から TB および KiB から TiB までの値と単位がサポートされます。たとえば、マウントポイントサイズをバイト単位で定義できます。

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

ユニットを使用してマウントポイントのサイズを定義することもできます。以下に例を示します。

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

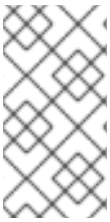
4.8. インストール済みパッケージ

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。デフォルトでは、Image Builder は **core dnf** グループを使用します。

表4.1イメージタイプの作成をサポートするデフォルトパッケージ

イメージタイプ	デフォルトパッケージ

イメージタイプ	デフォルトパッケージ
ami	checkpolicy、chrony、cloud-init、cloud-utils-growpart、@Core、dhcp-client、gdisk、insights-client、kernel、langpacks-en、net-tools、NetworkManager、redhat-release、redhat-release-eula、rng-tools、rsync、selinux-policy-targeted、tar、yum-utils
openstack	@Core、langpacks-en
qcow2	@Core、chrony、dnf、kernel、dnf、nfs-utils、dnf-util、cloud-init、python3-jsonschema、qemu-guest-agent、cloud-utils-growpart、dracut-norescue、tar、tcpdump、rsync、dnf-plugin-spacewalk、rhn-client-tools、rhnlib、rhnsd、rhn-setup、NetworkManager、dhcp-client、cockpit-ws、cockpit-system、subscription-manager-cockpit、redhat-release、redhat-release-eula、rng-tools、insights-client
rhel-edge-commit	glibc、glibc-minimal-langpack、nss-altfiles、kernel、dracut-config-generic、dracut-network、basesystem、bash、platform-python、shadow-utils、chrony、setup、shadow-utils、sudo、systemd、coreutils、util-linux、curl、vim-minimal、rpm、rpm-ostree、polkit、lvm2、cryptsetup、pinentry、e2fsprogs、dosfstools、keyutils、gnupg2、attr、xz、gzip、firewalld、iptables、NetworkManager、NetworkManager-wifi、NetworkManager-wwan、wpa_supplicant、dnsmasq、traceroute、hostname、iproute、iputils、openssh-clients、procps-ng、rootfiles、openssh-server、passwd、policycoreutils、policycoreutils-python-utils、selinux-policy-targeted、setools-console、less、tar、rsync、fwupd、usbguard、bash-completion、tmux、ima-evm-utils、audit、rng-tools、podman、container-selinux、skopeo、criu、slirp4netns、fuse-overlayfs、clevis、clevis-dracut、clevis-luks、greenboot、greenboot-grub2、greenboot-rpm-ostree-grub2、greenboot-reboot、greenboot-status
tar	policycoreutils、selinux-policy-targeted
vhd	@Core、langpacks-en
vmdk	@Core、chrony、firewalld、kernel、langpacks-en、open-vm-tools、selinux-policy-targeted



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

- Image Builder の説明

4.9. 有効なサービス

有効なサービスとは、カスタムイメージの設定時に、**osbuild-composer** を実行している RHEL 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4.2 イメージタイプの作成をサポートするために有効になっているサービス

イメージタイプ	有効なサービス
ami	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
openstack	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
qcow2	cloud-init
rhel-edge-commit	デフォルトでは、追加のサービスは有効になりません。
tar	デフォルトでは、追加のサービスは有効になりません。
vhd	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk	sshd, chronyd, vmtoolsd, cloud-init

備考: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

- サポートされているイメージのカスタマイズ

第5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。

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

RHEL 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**Managing systems using the RHEL web console** を参照してください。

2. システム上で十分な権限を持つユーザーアカウントの認証情報を使用して 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。

関連情報

- 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

5.2. 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 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。

   a. 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。

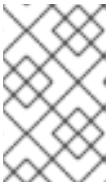
   b. コンポーネントのリストは、1ページずつ表示されます。別の結果ページに移動するには、コンポーネントリストの上にある矢印および入力フィールドを使用します。

   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。

   e. 左上の **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、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。

5.3. 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 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。

5.4. 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。

```
# dnf install editor-name
```

手順

1. blueprint の名前をクリックします。RHEL 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
```

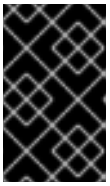
認証情報を求められたら入力してください。端末は、Web コンソールにログインする際に入力した認証情報を再利用しません。

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 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。

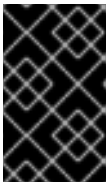
```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

.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限り、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 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

5.5. 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 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。

5.6. 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。

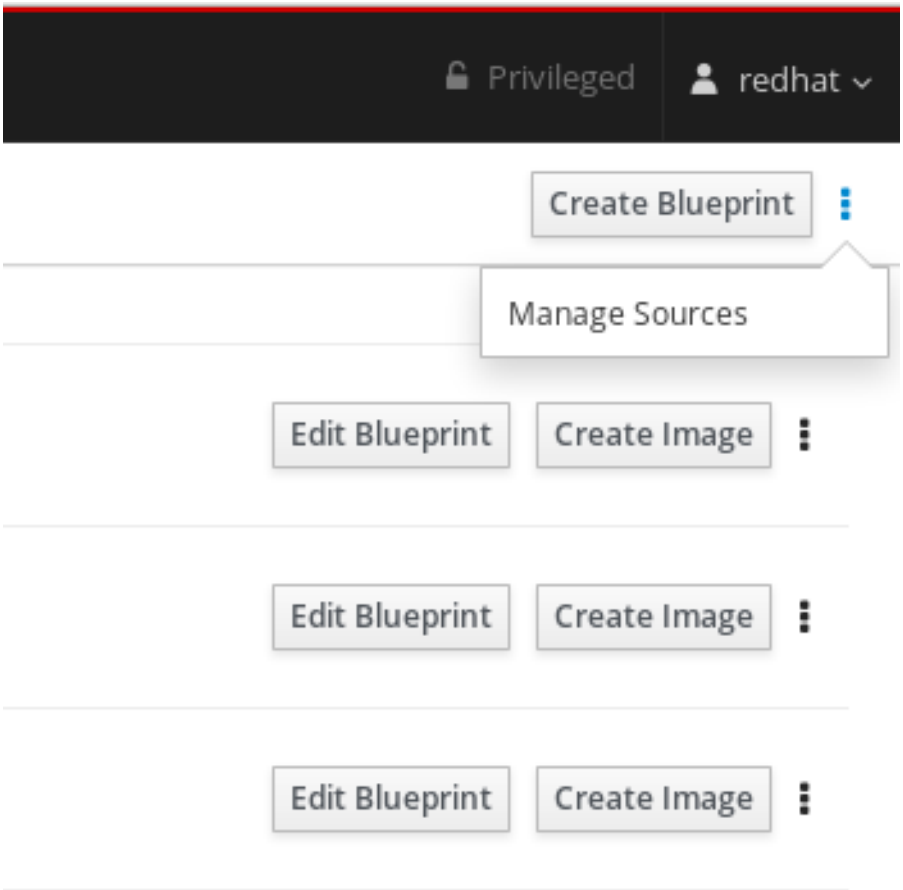
以下の手順では、ローカルシステムにソースを追加する方法を説明します。

前提条件

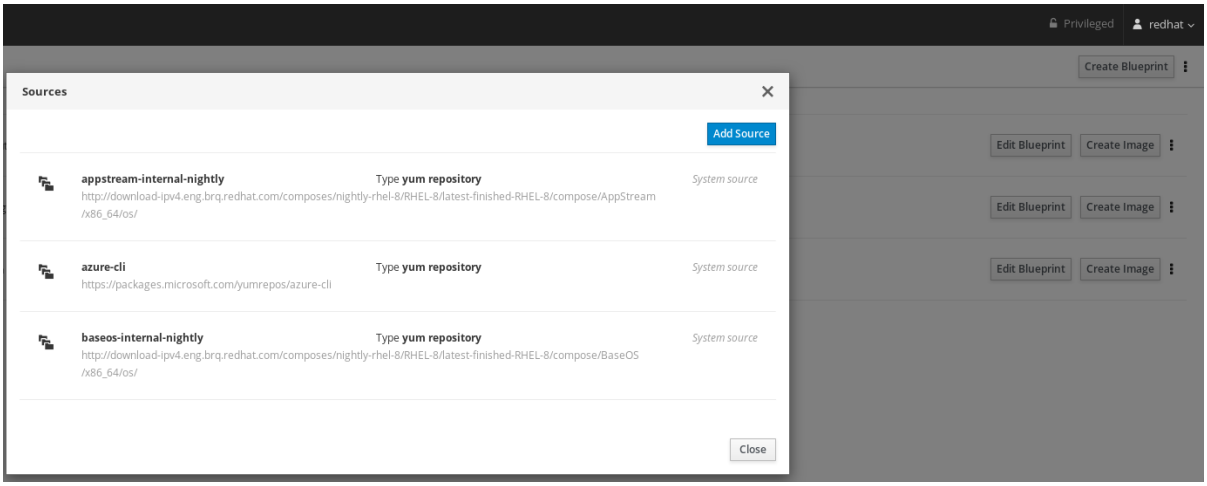
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェースを開いている。

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. **Add Source** 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。

5.7. 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 でパスワードつきのユーザーアカウントを作成でき、このアカウントで Blueprint から作成したイメージにログインできます。

前提条件

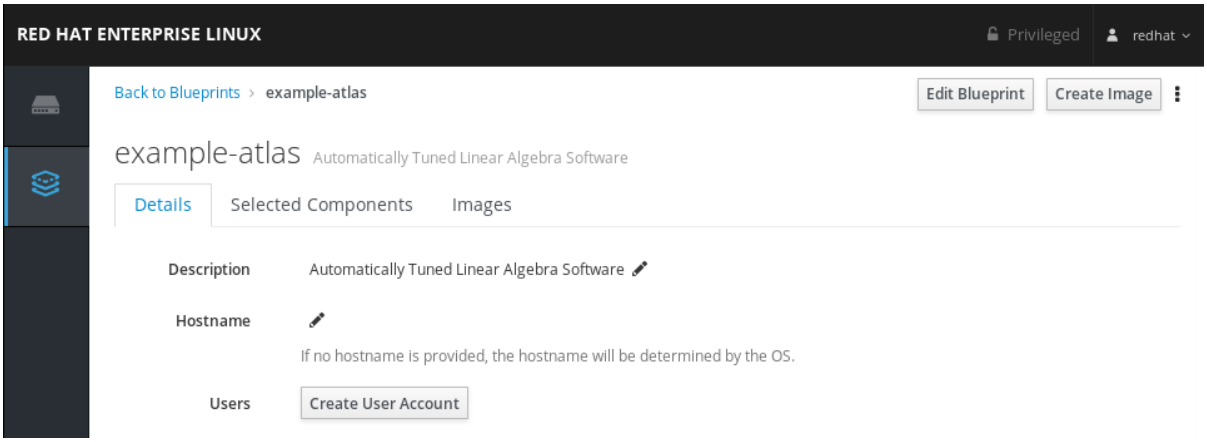
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェースを開いている。
- 既存の Blueprint がある。

手順

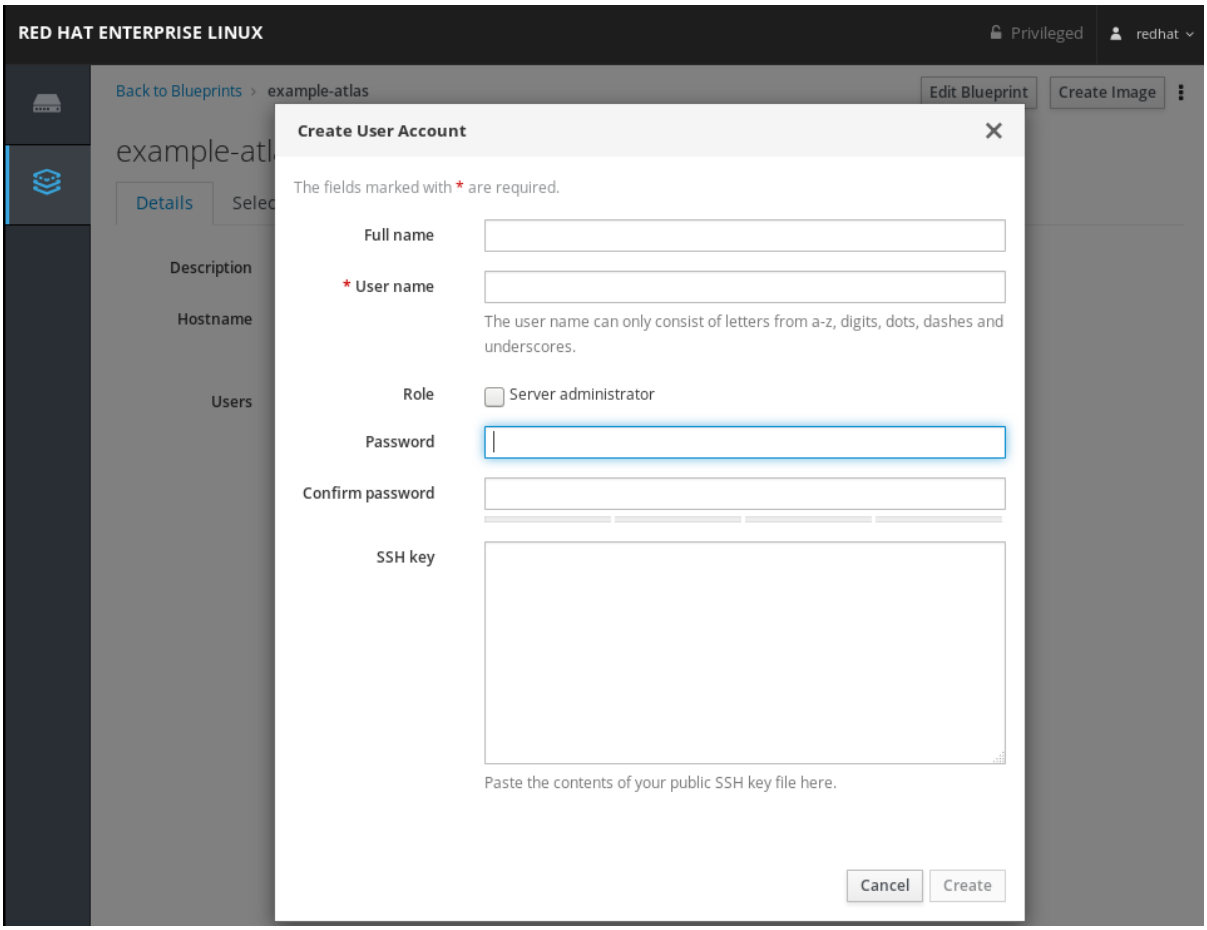
1. 左上の検索ボックスに Blueprint の名前またはその一部を入力し、**Enter** を押して、ユーザーアカウントを作成する Blueprint を検索します。

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。



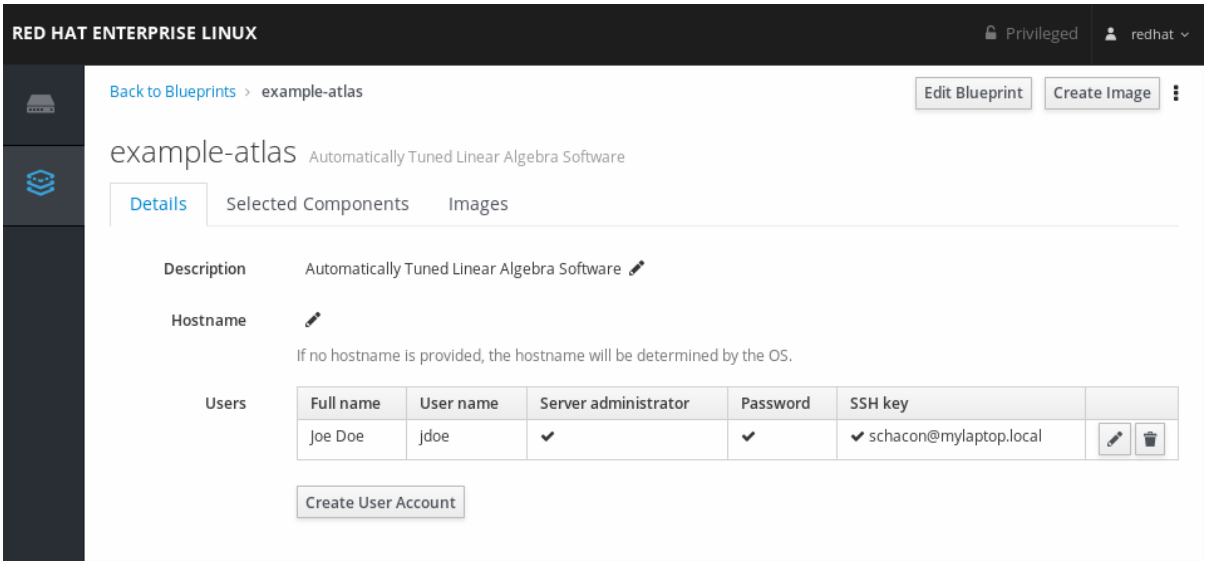
3. **ユーザーアカウントの作成** をクリックします。
これにより、ユーザーアカウントを作成するフィールドを含むウィンドウが開きます。



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、Create **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。

5.8. SSH 鍵を持つユーザーアカウントの作成

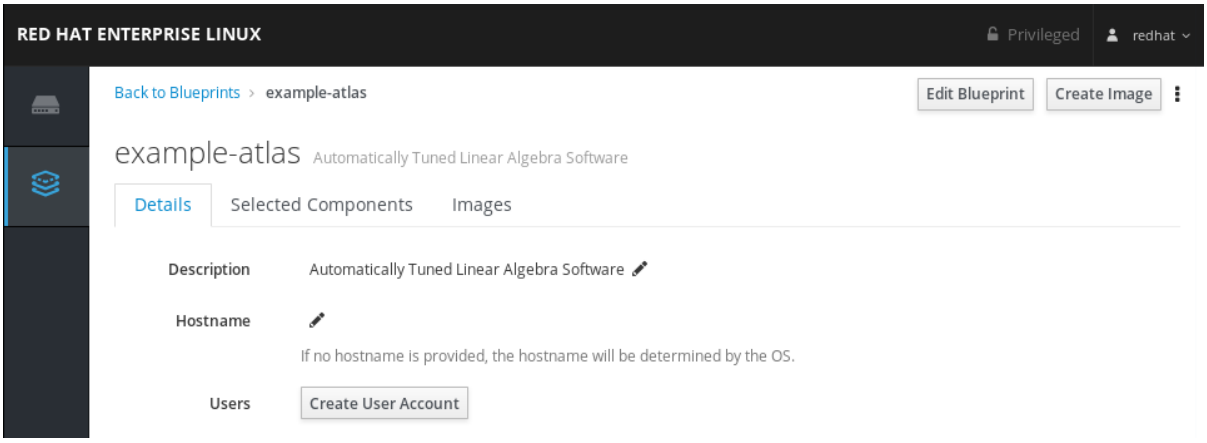
Image Builder で作成したイメージでは root アカウントがロックされ、他のアカウントは含まれていません。このような設定は、デフォルトパスワードを持たないイメージのセキュリティーを確保するために提供されます。Image Builder を使用すると、Blueprint の SSH 鍵でユーザーアカウントを作成し、Blueprint から作成したイメージに対して認証できるようにします。これを実行するには、まず Blueprint を作成します。次に、パスワードと SSH 鍵でユーザーアカウントを作成します。以下の例は、SSH キーを使用してサーバー管理ユーザーを作成する方法を示して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

前提条件

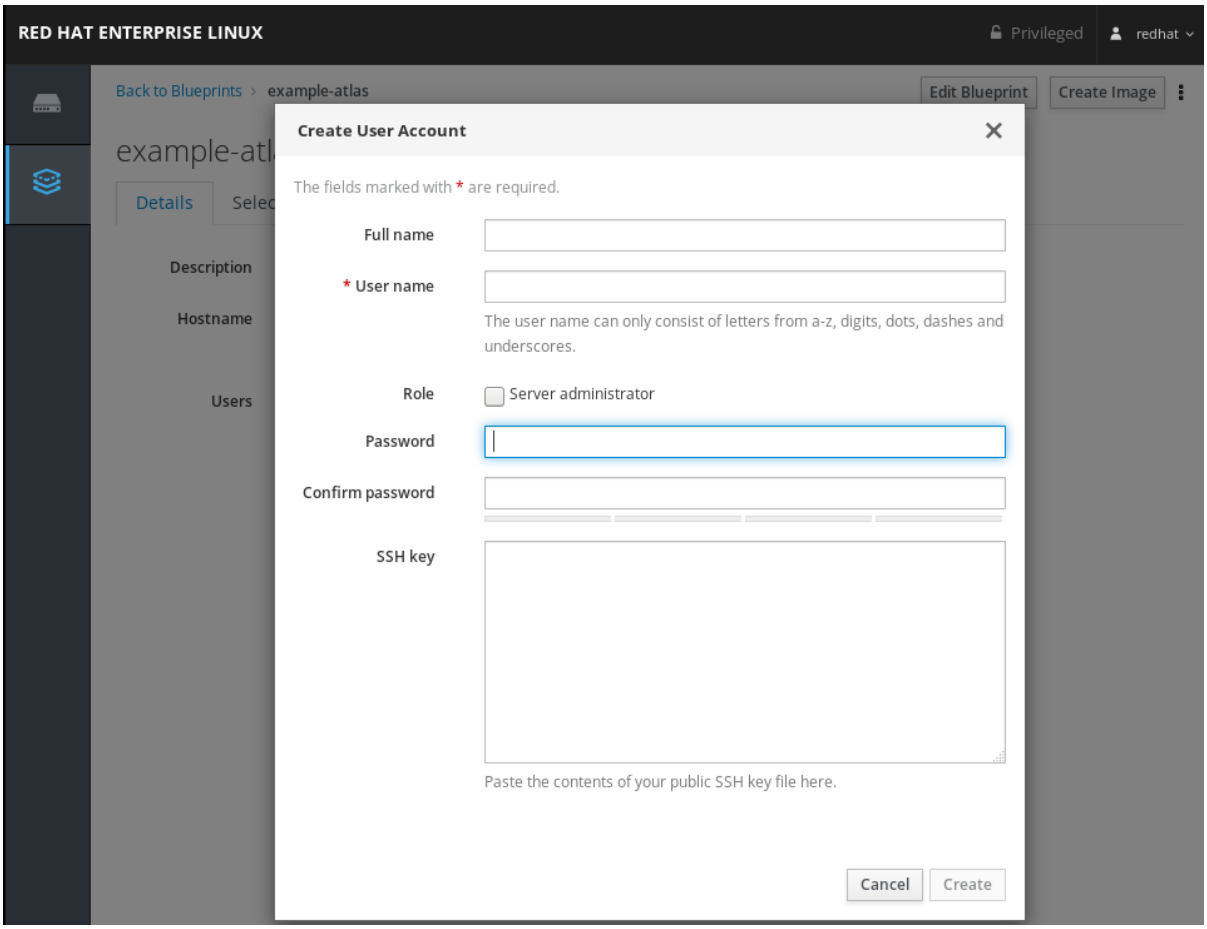
- プロセスを後で作成したユーザーとペアにする SSH キーを作成している。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェースを開いている。
- 既存の Blueprint がある。

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。

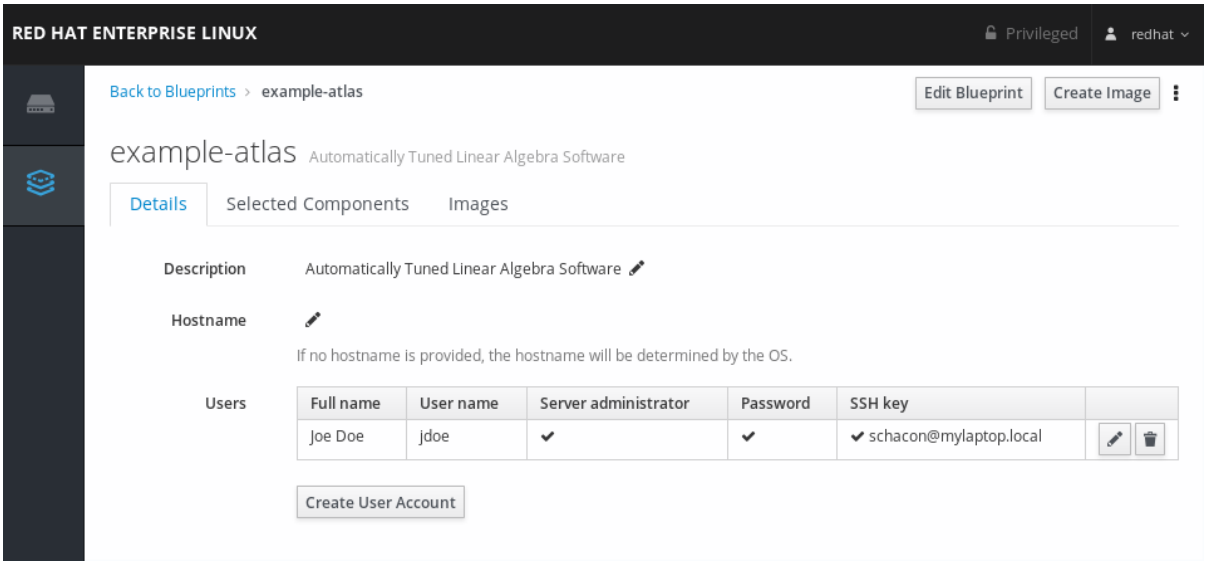


4. 詳細を入力します。名前を入力すると、**ユーザー名** のフィールドが自動補完され、ユーザー名を提案することに注意してください。
作成するユーザーアカウントに管理者権限を付与する場合は、**Role** フィールドを確認します。

   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、Create **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 鍵ペアの生成

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

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

Image Builder は、コミットとルートファイルシステムが含まれるブート ISO を作成するマニフェストを構築します。ISO イメージを作成するには、新しいイメージ種別 **image-installer** を選択します。Image Builder は、以下が含まれる **.tar** ファイルをビルドします。

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

作成されたインストーラー ISO イメージは、ベアメタルサーバーに直接インストールできる事前に設定されたシステムイメージを組み込みます。

6.1. IMAGE BUILDER コマンドラインインターフェースを使用したブート ISO インストーラーイメージの作成

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、カスタムブート ISO インストーラーイメージをビルドする方法を説明します。

前提条件

- ユーザーが含まれるイメージの Blueprint を作成し、Image Builder にプッシュしている。「Blueprint customization for users」を参照してください。

手順

1. ISO イメージを作成します。

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

   - **BLUEPRINT-NAME** は作成した Blueprint の名前です。
   - **image-installer** はイメージタイプです。
     Compose プロセスはバックグラウンドで開始し、Compose の UUID が表示されます。

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。
   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 は、前の手順で示した UUID 値に置き換えます。

これにより、Image Builder は ISO インストーラーイメージが含まれる **.tar** ファイルをビルドします。

検証

1. イメージファイルをダウンロードしたディレクトリーに移動します。
2. ダウンロードした **.tar** イメージを見つけます。
3. **.tar** コンテンツを展開します。

生成された ISO イメージファイルを、ハードドライブで使用したり仮想マシンの起動に使用したりできます(例: HTTP ブートまたは USB インストール)。

関連情報

- 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
- 起動可能なRHEL 用インストールメディアの作成

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

Image Builder GUI を使用して、カスタマイズされたブート ISO インストーラーイメージをビルドできます。

前提条件

- イメージの Blueprint を作成している。

手順

1. ブラウザーで RHEL Web コンソールの Image Builder インターフェースを開きます。
2. イメージをビルドする Blueprint を見つけます。検索ボックスに Blueprint の名前またはその一部を **入力** し、Enter をクリックします。
3. Blueprint の右側で、その Blueprint に関する **イメージの作成** ボタンをクリックします。**Create image** ダイアログウィザードが開きます。
4. **Create image** ダイアログウィザードの **Image Type** リストから以下を行います。
   a. **RHEL インストーラー(.iso)** イメージタイプを選択します。
   b. **Create** をクリックします。

Image Builder は、RHEL **.iso** イメージの Compose をキューに追加します。

注記

イメージのビルドプロセスが完了するまで数分かかります。

プロセスが完了すると、イメージビルドの完全なステータスを確認できます。Image Builder は **.iso** イメージを作成します。

検証

イメージが正常に作成されたら、イメージボタンをダウンロードできます。

1. **Download** をクリックして、RHEL インストーラー(.iso)イメージをシステムに保存します。
2. **RHEL インストーラー(.iso)をダウンロードしたディレクトリーに移動し** ます。
3. ダウンロードした .tar イメージを見つけます。
4. **「RHEL インストーラー(.iso)」** イメージコンテンツを展開します。

生成された ISO イメージファイルを、ハードドライブで使用したり仮想マシンの起動に使用したりできます(例: HTTP ブートまたは USB インストール)。

関連情報

- Web コンソールインターフェースで Image Builder の Blueprint の作成
- 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
- RHEL の起動可能なインストールメディアの作成

6.3. ベアメタルシステムへのISO イメージのインストール

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 を使用して作成した起動可能な ISO イメージをベアメタルシステムにインストールする方法を説明します。

前提条件

- Image Builder を使用して起動可能な ISO イメージを作成している。
- ブート可能な ISO イメージをダウンロードして展開している。
- 8 GB の USB フラッシュドライブがある。

注記

ISO サイズは、Blueprint で選択したパッケージによって大きくなる可能性があります。

手順

1. ブート可能な ISO イメージファイルを USB フラッシュドライブに配置します。
2. USB フラッシュドライブを、起動するコンピューターのポートに接続します。
3. USB フラッシュドライブから ISO イメージを起動します。
4. カスタマイズされた起動可能な ISO イメージをインストールする手順を実行します。
   起動画面に以下のオプションが表示されます。
   - Install Red Hat Enterprise Linux 9

- Test this media & install Red Hat Enterprise Linux 9

関連情報

- インストールの起動

第7章 IMAGE BUILDER を使用した KVM ゲストイメージの準備とデプロイ

お客様は、ISO から手動でイメージを作成することも、Image Builder を使用して専用のイメージを作成することもできます。この手順では、Image Builder を使用して専用のイメージを作成する方法を説明します。これは、Red Hat Virtualization (RHV) に対する **rhel-guest-image** のサポートに限定されます。

カスタマイズした KVM ゲストイメージを作成するには、以下の手順を実施します。

1. Image Builder を使用して KVM ゲストイメージ **.qcow2** イメージを作成します。
2. KVM ゲストイメージから仮想マシンを作成します。

7.1. IMAGE BUILDER を使用したカスタム KVM ゲストイメージの作成

ここでは、Image Builder を使用して **.qcow2** KVM ゲストイメージを作成する方法を説明します。

前提条件

- root または wheel グループでシステムにアクセスできる。
- **cockpit-composer** パッケージがインストールされている。
- RHEL システムで、Cockpit UI の Image Builder ダッシュボードを開いている。

手順

1. **Blueprint の作成** をクリックして Blueprint を作成します。「Web コンソールインターフェースで Image Builder の Blueprint の作成」を参照してください。
2. 作成中の KVM ゲストイメージの一部として必要なコンポーネントとパッケージを選択します。
3. **Commit** をクリックして、Blueprint に加えた変更をコミットします。右側の小さなポップアップに保存の進捗が表示され、続いてコミットした変更の結果が表示されます。
4. 左側のバナーの Blueprint 名リンクをクリックします。
5. **イメージ** タブを選択します。
6. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。ポップアップウィンドウが開きます。
7. **Type** ドロップダウンメニューの一覧から、`QEMU Image(.qcow2)' イメージを選択します。
8. イメージをインスタンス化する場合のサイズを設定し、**作成** をクリックします。
9. ウィンドウの右上にある小さなポップアップに、イメージの作成がキューに追加されたことが表示されます。イメージの作成プロセスが完了したら、イメージ **ビルドの完全なステータス** を確認できます。

検証手順

1. ブレッドクラムアイコンをクリックして、**ダウンロード** オプションを選択します。Image Builder は、KVM ゲストイメージ **.qcow2** ファイルをデフォルトのダウンロード場所にダウンロードします。

関連情報

- Web コンソールインターフェースで Image Builder の Blueprint の作成

7.2. KVM ゲストイメージからの仮想マシンの作成

KVM ゲストイメージを使用して、フットプリントの小さい仮想マシンをホスト上にすばやく作成することができます。この手順では、Image Builder が生成した KVM ゲストイメージを **.qcow2** イメージ形式として使用し、仮想マシン (VM) を作成します。Image Builder を使用して作成した KVM ゲストイメージでは、**cloud-init** がすでにインストールされ、有効になっています。

前提条件

- Image Builder を使用して **.qcow2** イメージを作成している。「Web コンソールインターフェースで Image Builder の Blueprint の作成」を参照してください。
- **qemu-kvm** パッケージがシステムにインストールされている。**/dev/kvm** フォルダーがシステムで利用できることを確認できます。
- システムに **libvirt** がインストールされている。
- システムに **virt-install** がインストールされている。
- **genisoimage** ユーティリティーがシステムにインストールされている。

手順

1. Image Builder を使用して作成した KVM ゲストイメージを **/var/lib/libvirt/images** ディレクトリーに移動し、イメージ名を **rhel-8.4-x86_64-kvm.qcow2** に変更します。
2. ディレクトリー (**cloudinitiso** など) を作成し、新規に作成したそのディレクトリーに移動します。

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

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

```
instance-id: citest
local-hostname: citest-1
```

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

```
#cloud-config
user: admin
password: cilogon
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
  - ssh-rsa AAA...fhHQ== your.email@example.com
```

詳細は以下のようになります。

- **ssh_authorized_keys** は、SSH 公開鍵になります。 **~/.ssh/id_rsa.pub** で SSH 公開鍵を確認できます。

5. **genisoimage** コマンドを使用して、**user-data** ファイルおよび **meta-data** ファイルを含む ISO イメージを作成します。

```
# genisoimage -output ciiso.iso -volid cidata -joliet -rock user-data meta-data

I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 331
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
183 extents written (0 MB)
```

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

```
# virt-install \
    --memory 4096 \
    --vcpus 4 \
    --name mytestcivm \
    --disk /var/lib/libvirt/images/rhel-8.4-x86_64-
kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
    --disk /home/sample/cloudinitiso/ciiso.iso,device=cdrom \
    --os-variant rhel8.4 \
    --virt-type kvm \
    --graphics none \
    --import
```

詳細は以下のようになります。

- --graphics none - ヘッドレス RHEL 8.4 仮想マシンであることを意味します。
- --vcpus 4 - 4 つの仮想 CPU を使用することを意味します。
- --memory 4096 - 4096 MB のメモリーを使用することを意味します。

7. 仮想マシンのインストールが起動します。

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

Red Hat Enterprise Linux 8.4 Beta (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

検証

1. **cloud-user** をユーザー名として使用し、作成した仮想マシンにログインします。パスワードは **cilogon** です。

関連情報

- 「仮想化を有効にする」を参照してください。
- RHEL 9 の「cloud-init の設定および管理」を参照 してください。
- cloud-init の重要なディレクトリーおよびファイルを 参照してください。

第8章 IMAGE BUILDER を使用した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。Red Hat Enterprise Linux 8.3 から、RHEL Web コンソールの **Image Builder** アプリケーション経由でカスタマイズされたイメージクラウドをプッシュする機能は、**AWS** や **Azure** クラウドなどのサポート対象のサービスプロバイダーのサブセットで利用できます。AWS Cloud AMI へのイメージのプッシュ および Azure クラウドへの VHD イメージのプッシュ を参照してください。

8.1. 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。

```
# dnf install python3
# dnf 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. 以下のコマンドを実行してプロファイルを設定します。ターミナルで、認証情報、リージョン、および出力形式を指定するように求められます。

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

4. バケット名を定義し、以下のコマンドを使用してバケットを作成します。

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

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

5. 次に、S3 バケットへのアクセス権限を付与するには、IAM に **vmimport** S3 Role を作成していない場合には作成します。

```
$ 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
```

関連情報

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

8.2. CLI で AWS に AMI イメージをアップロードする

Image Builder を使用して**.ami** イメージを構築し、CLI を使用してそれらを Amazon AWS Cloud サービスプロバイダーに直接プッシュできます。

前提条件

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

手順

1. 選択したテキストエディターを使用して、次の内容の設定ファイルを作成します。

```
provider = "aws"

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

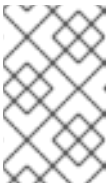
フィールドの値を、**accessKeyID**、**secretAccessKey**、**bucket**、**region** の認証情報に置き換えます。**IMAGE_KEY** 値は、EC2 にアップロードされる VM イメージの名前です。

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

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE IMAGE_KEY CONFIGURATION-FILE.toml
```

以下を置き換えます。

- **BLUEPRINT-NAME** は作成した Blueprint の名前です。
- **IMAGE-TYPE**と**ami**イメージタイプ。
- EC2 にアップロードする VM イメージの名前を含む **IMAGE_KEY**。
- クラウドプロバイダーの設定ファイルの名前を持つ **CONFIGURATION-FILE**.toml。



注記

カスタマイズイメージを送信するバケットの正しい IAM 設定が必要です。イメージをアップロードする前にバケットにポリシーを設定しておく必要があります。

4. イメージビルドのステータスを確認し、AWS にアップロードします。

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

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

検証

イメージのアップロードが成功したことを確認するには、以下を行います。

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

関連情報

- 必要なサービスロール

8.3. イメージの AWS CLOUD AMI へのプッシュ

今回のリリースでは、作成する出力イメージを **AWS Cloud AMI**にプッシュする機能を利用できます。ここでは、Image Builder を使用して作成する **.ami** イメージを Amazon AWS Cloud サービスプロバイダーにプッシュする手順を説明します。

前提条件

- **root** または **wheel** グループでシステムにアクセスできる。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェースを開いている。
- 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. **Blueprint の作成** をクリックして Blueprint を作成します。「Web コンソールインターフェースで Image Builder の Blueprint の作成」を参照してください。

2. 作成するイメージの一部として、コンポーネントおよびパッケージを選択します。

3. **Commit** をクリックして、Blueprint に加えた変更をコミットします。
右側の小さなポップアップに保存の進捗が表示され、続いてコミットした変更の結果が表示されます。

4. 左側のバナーの **Blueprint 名** リンクをクリックします。

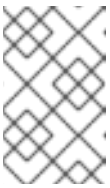
5. **イメージ** タブを選択します。

6. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。
ポップアップウィンドウが開きます。

   a. **Type** ドロップダウンメニューから、「Amazon Machine Image Disk(.ami)」イメージを選択します。

   b. **Upload to AWS** チェックボックスをチェックして、イメージを AWS Cloud にアップロードし、**Next** をクリックします。

   c. AWS へのアクセスを認証するには、対応するフィールドに「AWS アクセスキー ID」および「AWS シークレットアクセスキー」と入力します。**Next** をクリックします。

   注記

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

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

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

   注記

   カスタマイズイメージを送信するバケットの正しい IAM 設定が必要です。IAM のインポートおよびエクスポートを使用するので、イメージをアップロードする前にバケットに **ポリシー** を設定しておく必要があります。詳細は、「IAM ユーザーの必要なパーミッション」 を参照してください。

7. 右側の小さなポップアップに保存の進捗が表示されます。また、イメージの作成、イメージ作成の進捗、およびそれ以降の AWS Cloud にアップロードに関する情報も通知されます。
プロセスが完了すると、**「Image build complete」** というステータスが表示されます。

8. メニューで Service→EC2 をクリックし、AWS コンソールで 正しいリージョン を選択します。
イメージのステータスは、アップロードされていることを示す「Available」でなければなりません。

9. Dashboard でイメージを選択し、**起動** をクリックします。

10. 新しいウィンドウが開きます。イメージを起動する必要のあるリソースに応じて、インスタンスタイプを選択します。**確認と起動** をクリックします。

11. インスタンスの起動の情報を確認します。変更が必要な場合は、各セクションを編集できます。**起動** をクリックします。

12. インスタンスを起動する前に、そのインスタンスにアクセスするための公開鍵を選択する必要があります。
既存のキーペアを使用するか、キーペアーを新規作成します。**Image Builder** を使用して、既存の公開鍵でイメージにユーザーを追加します。詳細は、「SSH 鍵を持つユーザーアカウントの作成」 を参照してください。

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

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

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

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

13. 次に、**Launch Instance** をクリックしてインスタンスを起動します。
インスタンスのステータスが "**Initializing**" と表示されていることが確認できます。

14. インスタンスのステータスが 「**running**」 になると、**Connect** ボタンが利用可能になります。

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

    a. **A standalone SSH client** への任意の接続方法を選択し、ターミナルを開きます。

    b. 秘密鍵の保存先で、SSH が機能するように鍵が公開されているようにします。これには、以下のコマンドを実行します。

    ```
    $ chmod 400 <your-instance-name.pem>_
    ```

    c. パブリック DNS を使用してインスタンスに接続します。

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

    d. 「yes」と入力して、接続の続行を確定します。
    これで、SSH でインスタンスに接続されました。

検証手順

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

関連情報

- Red Hat カスタマーポータルでのケースの作成
- SSH を使用した Linux インスタンスへの接続
- サポートケースの作成

8.4. 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。

```
# dnf 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。

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

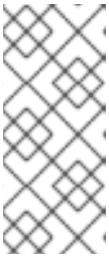
注記

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

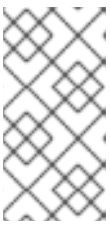
- Azure CLI

8.5. 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。



注記

CLI を使用して **.vhd** イメージを作成すると、Image Builder は一時ファイルを **/var** サブディレクトリーに書き込みます。**.vhd** イメージの作成に失敗するのを回避するには、**/var** サブディレクトリーの容量を少なくとも 15 GB の空き領域に増やして可用性を確保します。

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 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。

8.6. VMDK イメージの VSPHERE へのアップロード

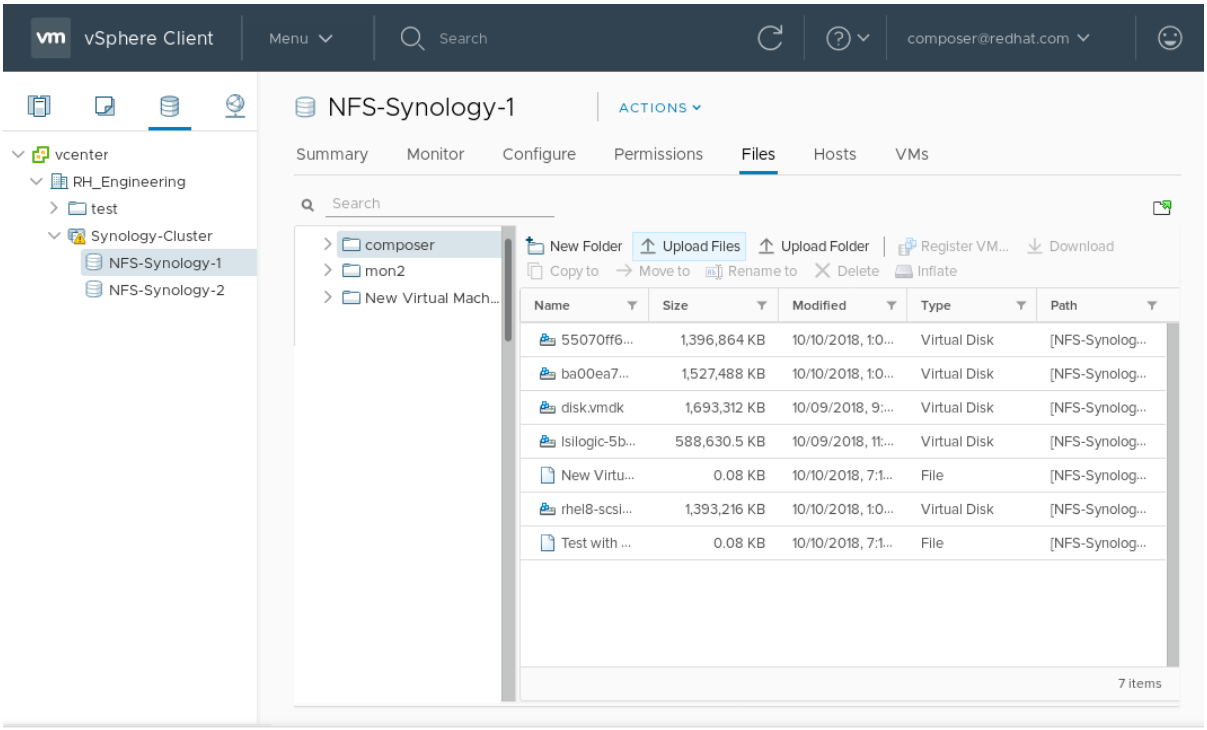
Image Builder は、VMware ESXi システムまたは vSphere システムへのアップロードに適したイメージを生成できます。ここでは、VMDK イメージを VMware vSphere にアップロードする手順を説明します。

前提条件

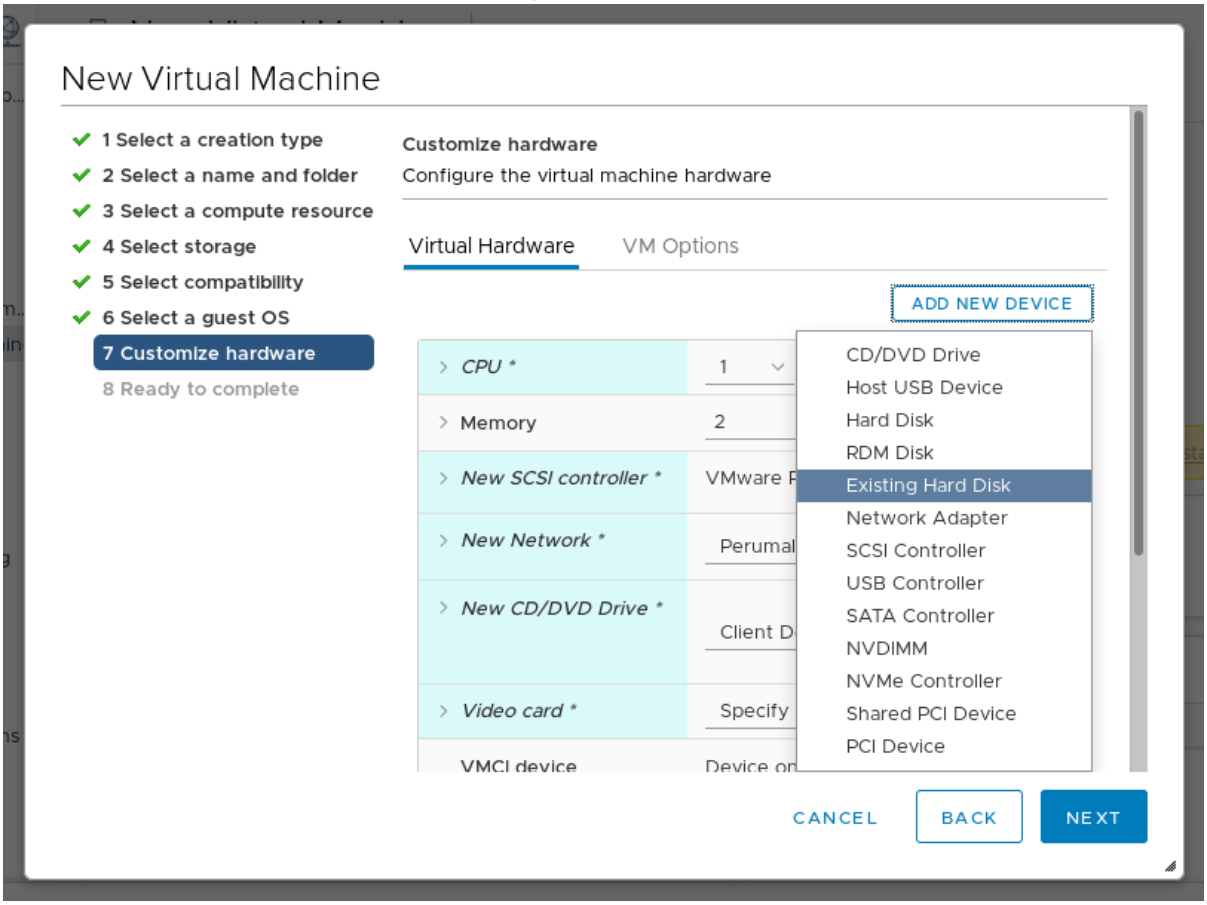
- Image Builder で VMDK イメージを作成している。イメージの作成時に、CLI で **vmdk** 出力タイプ、または GUI で **VMware Virtual Machine Disk (.vmdk)**を使用します。

手順

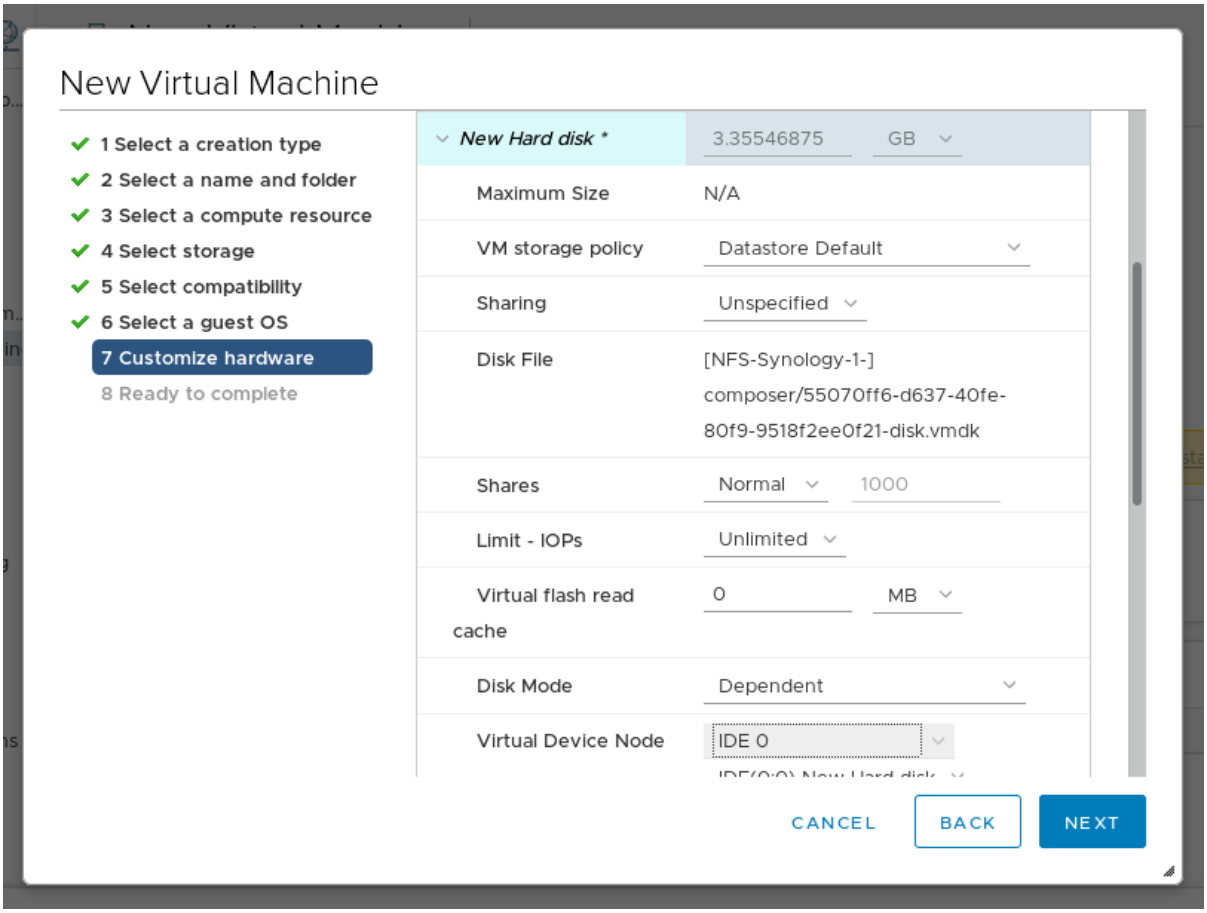
1. HTTP 経由でイメージを vSphere にアップロードします。vCenter で **Upload Files** をクリックします。



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。



8.7. VMWARE イメージの VSPHERE へのプッシュ

VMware イメージをビルドし、それらを vSphere インスタンスに直接プッシュできます。これにより、イメージファイルをダウンロードして手動でプッシュする必要がなくなります。ここでは、Image Builder を使用して作成する **.vmdk** イメージを vSphere インスタンスサービスプロバイダーに直接プッシュする手順を説明します。

前提条件

- **root** または **wheel** グループでシステムにアクセスできる。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェースを開いている。
- vSphere アカウント がある。

手順

1. **Create blueprint** をクリックします。
「Web コンソールインターフェースで Image Builder の Blueprint の作成 」を参照してください。
2. 作成するイメージの一部として、コンポーネントおよびパッケージを選択します。
3. **Commit** をクリックして、Blueprint に加えた変更をコミットします。
右上のポップアップに保存の進捗が表示され、続いてコミットした変更の結果が表示されます。
4. 左側のバナーの **Blueprint 名** リンクをクリックします。

5. **Customizations** タブを選択して、Blueprint のユーザーアカウントを作成します。
「Blueprint のユーザーアカウントの作成」を参照してください。

6. **Images** タブを選択します。

7. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。
イメージタイプウィンドウが開きます。

8. **Image type** ウィンドウで、以下を実行します。

   a. ドロップダウンメニューから、タイプを選択します。VMware VSphere(.vmdk)

   b. **Upload to VMware** チェックボックスをチェックして、イメージを vSphere にアップロードします。

   c. オプション:インスタンス化するイメージのサイズを設定します。最小のデフォルトサイズは 2GB です。

   d. **Next** をクリックします。

9. **Upload to VMware** ウィンドウで、**Authentication** の以下の詳細を入力します。

   a. ユーザー名: vSphere アカウントのユーザー名。

   b. パスワード: vSphere アカウントのパスワード。

10. **Upload to VMware** ウィンドウで **Destination** の以下の詳細を入力します。

    a. **イメージ名**: アップロードするイメージの名前。

    b. **ホスト** :イメージがアップロードされる VMware vSphere の URL。

    c. **クラスター** :イメージがアップロードされるクラスターの名前。

    d. **データセンター** :イメージがアップロードされるデータセンターの名前。

    e. **データストア**: イメージをアップロードするデータストアの名前。

    f. **Next** をクリックします。

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

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

注記

イメージビルドおよびアップロードプロセスの完了には数分かかります。

プロセスが完了すると、**イメージビルドの完全な** ステータスを確認できます。

検証

イメージステータスのアップロードが正常に完了したら、アップロードしたイメージから仮想マシン (VM) を作成し、ログインできます。以下の手順に従ってください。

1. VMware vSphere クライアントにアクセスします。
2. 指定した vSphere インスタンスのクラスターでイメージを検索します。
3. アップロードしたイメージから新しい仮想マシンを作成できます。作成するには、以下を実行します。
   a. アップロードしたイメージを選択します。
   b. 選択したイメージの右側にあるボタンをクリックします。
   c. New Virtual Machine をクリックします。
   New Virtual Machine ウィンドウが開きます。

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

      i. 作成タイプを選択します。新しい仮想マシンの作成を選択できます。
      ii. 名前とフォルダーを選択します。たとえば、仮想マシン名: **vSphere 仮想マシン** および vSphereClient 内で選択した場所。
      iii. コンピューターリソースの選択: この操作の宛先コンピューターリソースを選択します
      iv. ストレージを選択します。たとえば、NFS-Node1 を選択します
      v. 互換性を選択します。イメージは BIOS のみである必要があります。
      vi. ゲスト OS を選択します。たとえば、**Linux** と _Red Hat Fedora (64 ビット) を選択します。
      vii. **ハードウェアのカスタマイズ** :VM を作成するときは、右上の **デバイス設定** ボタンで、デフォルトの新しいハードディスクを削除し、ドロップダウンを使用して既存のハードディスクディスクイメージを選択します。
      viii. 完了する準備ができました:詳細を確認し、**完了** をクリックしてイメージを作成します。

   d. **VMs** タブに移動します。
      i. 一覧から、作成した仮想マシンを選択します。
      ii. パネルから **Start** ボタンをクリックします。仮想マシンイメージを読み込み中であることを示す新しいウィンドウが表示されます。
      iii. Blueprint 用に作成した認証情報を使用してログインします。
      iv. Blueprint に追加したパッケージがインストールされていることを確認できます。以下は例になります。

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

関連情報

- Installing the vSphere Client

8.8. VHD イメージの AZURE クラウドへのプッシュ

Image Builder を使用して **.vhd** イメージを作成できます。次に、**.vhd** イメージを Azure Cloud サービスプロバイダーの Blob ストレージにプッシュできます。

前提条件

- システムへの root アクセス権限がある。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェースを開いている。
- Storage Account を作成している。
- 書き込み可能な Blob ストレージ を準備している。

手順

1. **Blueprint の作成** をクリックして Blueprint を作成します。「Web コンソールインターフェースで Image Builder の Blueprint の作成」を参照してください。
2. 作成するイメージの一部として、コンポーネントおよびパッケージを選択します。
3. **Commit** をクリックして、Blueprint に加えた変更をコミットします。
   右上の小さなポップアップに保存の進捗が表示され、続いてコミットした変更の結果が表示されます。
4. 左側のバナーの **Blueprint 名** リンクをクリックします。
5. **イメージ** タブを選択します。
6. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。
   ポップアップウィンドウが開きます。
   a. **Type** ドロップダウンメニューから、**Azure Disk Image(.vhd)** イメージを選択します。
   b. **Upload to Azure** チェックボックスをチェックして、イメージを Azure Cloud にアップロードし、**Next** をクリックします。
   c. Azure へのアクセスを認証するには、対応するフィールドに「ストレージアカウント」と「ストレージアクセスキー」を入力します。**Next** をクリックします。
   ストレージアカウントの詳細 は、Settings→Access Key メニュー一覧を参照してください。
   d. アップロードするイメージファイルに使用する **イメージ名** と、イメージのプッシュ先のイメージファイルに使用する Blob の「ストレージコンテナー」を入力します。**Next** をクリックします。
   e. 提供された情報を確認し、**Finish** をクリックします。
   必要に応じて、**戻る** をクリックして、誤った情報を変更できます。
7. イメージ作成プロセスが次のメッセージで始まると、右上に小さなポップアップが表示されます。イメージの作成がキューに追加されました。
   イメージプロセスの作成が完了したら、イメージを作成した Blueprint をクリックします。**Images** タブに、作成したイメージの **Image build complete** ステータスが表示されます。
8. **Azure Cloud** にプッシュしたイメージにアクセスするには、Azure Portal にアクセスします。

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

10. **Add** をクリックします。Create an Image ダッシュボードにリダイレクトされます。
以下の情報を追加します。

    a. **名前**:新しいイメージの名前を選択します。

    b. **リソースグループ**:**リソースグループ** を選択します。

    c. **場所**:ストレージアカウントに割り当てられたリージョンと一致する **場所** を選択します。それ以外の場合は、Blob を選択できません。

    d. **OS タイプ**:OS の種別を **Linux** に設定します。

    e. VM Generation:仮想マシンの生成は **Gen 1** に設定したままにします。

    f. Storage Blob:Storage blob input の右側にある **参照** をクリックします。ダイアログを使用して、先ほどアップロードしたイメージを見つけます。
    その他のフィールドはデフォルトのままにしておきます。

11. **作成** をクリックしてイメージを作成します。イメージの作成後、右上隅に **Successfully created image というメッセージが** 表示されます。

12. **Refresh** をクリックし、新しいイメージを表示して新たに作成したイメージを開きます。

13. **+ Create VM** をクリックします。**Create a virtual machine**dashboard にリダイレクトされます。

14. Basic タブの Project Details (**サブスクリプション**) および Resource Group はすでに事前に設定されています。
新しいリソースグループを作成する場合

    a. **Create new** をクリックします。
    ポップアップで、**リソースグループ名** のコンテナーの作成が求められます。

    b. 名前を入力して **OK** をクリックします。
    事前に設定された **リソースグループ** をそのまま使用する場合

15. Instance Details で以下を挿入します。

    a. **Virtual machine name**

    b. **Region**

    c. **イメージ**:作成したイメージがデフォルトで事前に選択されています。

    d. **サイズ**:必要に応じて仮想マシンのサイズを選択します。
    その他のフィールドはデフォルトのままにしておきます。

16. Administrator account に、以下の情報を入力します。

    a. **username**: アカウント管理者の名前。

    b. **SSH Public Key source**: ドロップダウンメニューから、**Generate new key pair** を選択します。

既存のキーペアを使用するか、キーペアーを新規作成します。**Image Builder** を使用して、既存の公開鍵でイメージにユーザーを追加します。詳細は、「SSH 鍵を持つユーザーアカウントの作成」を参照してください。

c. **key pair name**: キーペアの名前を挿入します。

17. **Inbound port rules** で以下を選択します。

a. **Public inbound ports**:**Allow selected ports**.

b. **Select inbound ports**:デフォルト設定 **SSH (22)** を使用します。

18. **Review + Create** をクリックします。**Review + create** タブにリダイレクトされ、検証が正常に終了した旨の確認メッセージが表示されます。

19. 詳細を確認して **Create** をクリックします。
オプションで **Previous** をクリックして、以前に選択したオプションを修正できます。

20. **generates new key pair** のポップアップウィンドウが開きます。**Download private key and create resources** をクリックします。
yourKey.pem として鍵ファイルを保存します。

21. デプロイメントが完了したら、**Go to resource** をクリックします。

22. 実際の仮想マシンの詳細を含む新規ウィンドウに、リダイレクトされます。ページの右上にあるパブリック IP アドレスを選択してクリップボードにコピーします。

次に、仮想マシンとの SSH 接続を作成して、仮想マシンに接続します。

1. 端末プログラムを開きます。

2. プロンプトで、仮想マシンへの SSH 接続を開きます。IP アドレスは、仮想マシンの IP アドレスに、.pem へのパスは、キーファイルのダウンロード先のパスに置き換えます。

```
# ssh -i ./Downloads/yourKey.pem azureuser@10.111.12.123
```

3. 接続を続行するには確定する必要があります。続行するには yes と入力します。

上記の作業の結果、Azure Storage Blob にプッシュした出力イメージをプロビジョニングする準備が整いました。

関連情報

- Azure Storage ドキュメント
- Create an Azure Storage account
- Red Hat カスタマーポータルでのケースの作成
- ヘルプとサポート
- Red Hat に問い合わせる

8.9. 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

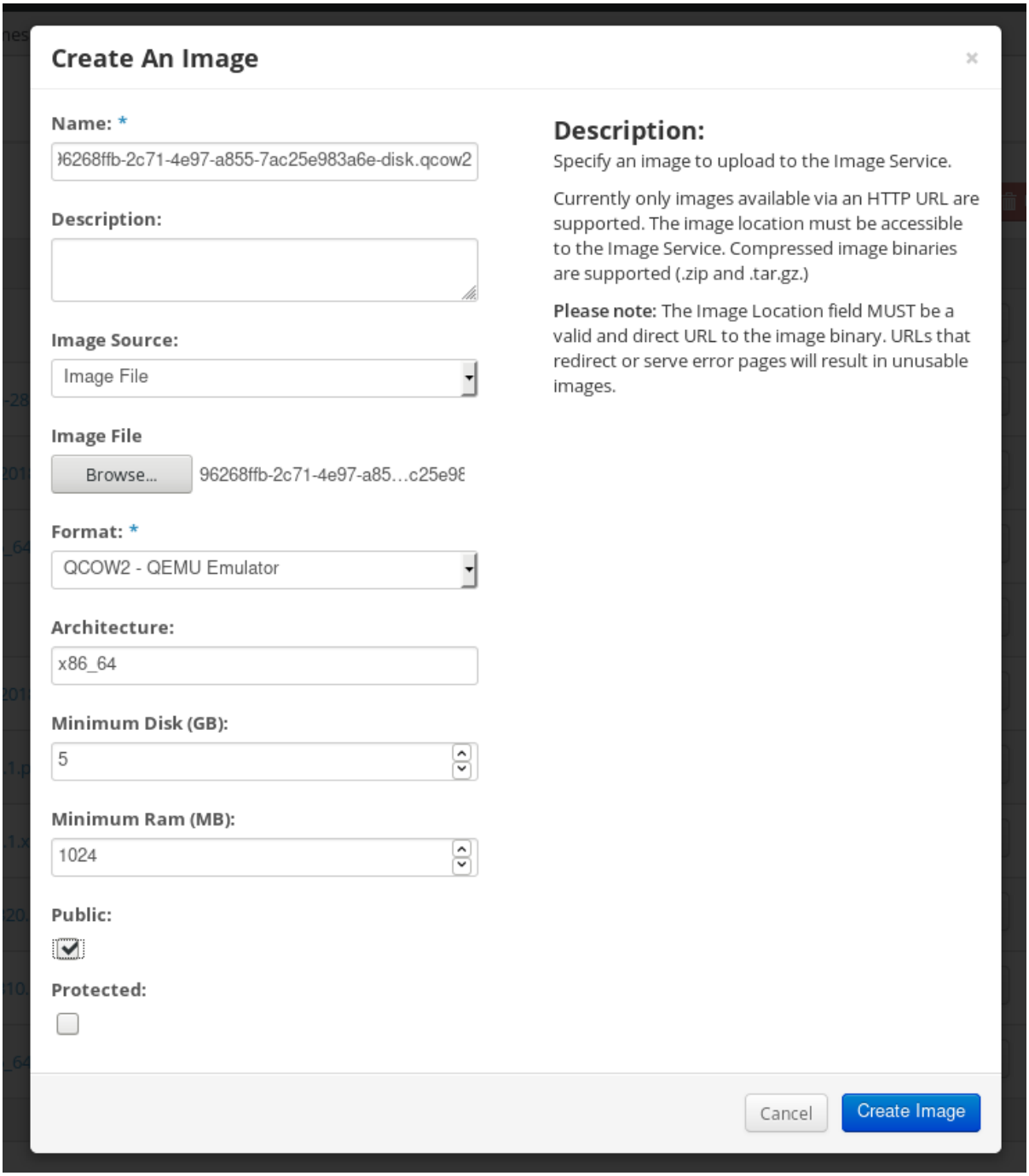
- Image Builder で OpenStack 固有のイメージを作成している。イメージの作成時に、CLI で **openstack** 出力タイプ、GUI で **OpenStack Image (.qcow2)** を使用します。

警告

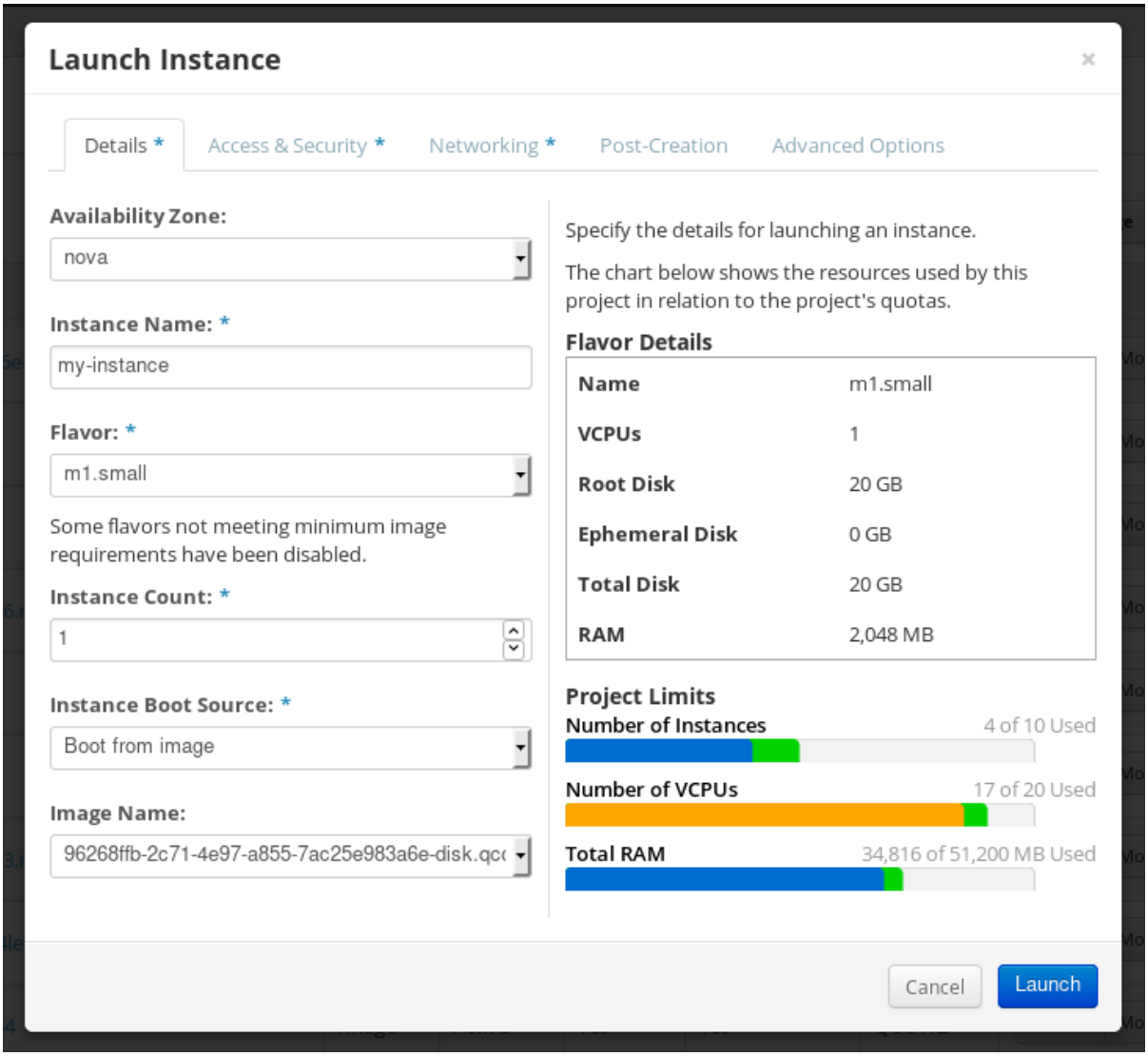
また、Image Builder は、汎用 QCOW2 イメージタイプの出力を、**qcow2** または **QEMU QCOW2 Image (.qcow2)** として提供します。これは、QCOW2 形式にある OpenStack イメージタイプとは異なります。OpenStack に固有の変更が含まれています。

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。

8.10. AIBABA へのイメージのアップロードの準備

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。

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。

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。詳細はリンクを参照してください。Detection items section.

関連情報

- Image Compliance Tool

8.11. 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

- Upload an object
- Creating an instance from custom images
- Importing images

8.12. 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

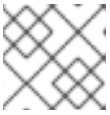
- イメージを OSS (Object Storage Service) にアップロードしている。

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 の取得方法を参照してください。
      b. **Image Name**:
      c. **Operating System**:
      d. **System Disk Size**:
      e. **System Architecture**
      f. **プラットフォーム**: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

- Notes for importing images
- Creating an instance from custom images
- Upload an object

8.13. ALIBABA を使用したカスタムイメージのインスタンスの作成

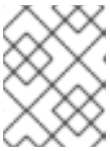
Alibaba ECS コンソールを使用して、カスタムイメージのインスタンスを作成できます。

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

- Creating an instance by using a custom image
- Create an instance by using the wizard