Red Hat Enterprise Linux 9

RHEL for Edge イメージの作成、インストール、および管理

Red Hat Enterprise Linux 9 を使用した Edge システムの作成、デプロイ、管理

Last Updated: 2024-07-02

Red Hat Enterprise Linux 9 RHEL for Edge イメージの作成、インストール、および管理

Red Hat Enterprise Linux 9 を使用した Edge システムの作成、デプロイ、管理

法律上の通知

概要

Image Builder ツールを使用して、Edge 用に最適化されたカスタマイズ済みの RHEL (rpm-ostree) イメージを作成します。次に、リモートでエッジサーバーにイメージをインストールし、デプロイメントをセキュアに管理およびスケーリングできます。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

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

Jira からのフィードバック送信 (アカウントが必要)

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

第1章 RHEL FOR EDGE イメージの概要

RHEL for Edge イメージは、Edge サーバーで RHEL をリモートにインストールするシステムパッケージを含む **rpm-ostree** イメージです。

システムパッケージには以下が含まれます。

- **Base OS** パッケージ
- コンテナーエンジンとしての Podman
- 追加の RPM Package Manager (RPM) コンテンツ

RHEL イメージとは異なり、RHEL for Edge はイミュータブルなオペレーティングシステムです。つまり、次の特性を持つ **read-only** root ディレクトリーが含まれています。

- パッケージは root ディレクトリーから分離されている
- パッケージのインストールにより、以前のバージョンへのロールバックを容易にするレイヤーが作成される
- 切断された環境への効率的な更新
- 複数のオペレーティングシステムのブランチとリポジトリーをサポート
- ハイブリッド **rpm-ostree** パッケージシステムを搭載

Bare Metal、Appliance、および Edge サーバーに RHEL for Edge イメージをデプロイすることができます。

RHEL Image Builder ツールを使用して、カスタマイズした RHEL for Edge イメージを作成できます。Red Hat Hybrid Cloud Console プラットフォームで Edge Management アプリケーションにアクセスし、自動管理を設定して、RHEL for Edge イメージを作成することもできます。

Edge Management アプリケーションは、イメージのプロビジョニングおよび登録方法を簡素化します。Edge Management の詳細は、RHEL for Edge イメージの作成および自動管理に関するドキュメントの設定 を参照してください。

警告

Edge Management アプリケーションでは、**RHEL Image Builder** のオンプレミスバージョンのアーティファクトを使用して作成された RHEL for Edge のカスタマイズイメージの使用はサポートされていません。Edge management supportability を参照してください。

RHEL for Edge イメージを使用すると、以下を実行できます。

1.1. RHEL FOR EDGE 対応アーキテクチャー

現在、AMD と Intel 64 ビットシステムに RHEL for Edge イメージをデプロイすることができます。

注記

RHEL for Edge は、一部のデバイスで ARM システムをサポートします。サポートされているデバイスの詳細は、Red Hat certified hardware を参照してください。

1.2. RHEL FOR EDGE イメージの作成とデプロイ方法

RHEL for Edge イメージの設定とデプロイには、次の 2 つのフェーズが含まれます。

1. RHEL Image Builder ツールを使用した RHEL **rpm-ostree** イメージの作成。**composer-cli** ツールのコマンドラインインターフェイスから RHEL Image Builder にアクセスするか、RHEL Web コンソールのグラフィカルユーザーインターフェイスを使用できます。

2. RHEL インストーラーを使用したイメージのデプロイ

RHEL for Edge イメージの作成中、以下のイメージタイプのいずれかを選択できます。別の RHEL for Edge イメージを設定するには、ネットワークアクセスが必要な場合と必要でない場合があります。次の表を参照してください。

表1.1 RHEL for Edge イメージタイプ

イメージタイプ	説明	ネットワークベースのデプロイメントに適しています	ネットワークベース以外のデプロイメントに適しています	使用方法
RHEL for Edge Commit (.tar)	**edge-commit** イメージは、完全なオペレーティングシステムが含まれていても、直接起動することはできません。**edge-commit** イメージタイプを起動するには、それをデプロイする必要があります。	はい	いいえ	システムにアトミックで安全な更新を提供します。

イメージタイプ	説明	ネットワークベースのデプロイメントに適しています	ネットワークベース以外のデプロイメントに適しています	使用方法
RHEL for Edge Container (**.tar**)	**edge-container** では、**OSTree** のコミットを作成し、Web サーバーのある OCI コンテナーに埋め込みます。**edge-commit** イメージが起動すると、Web サーバーはコミットを OSTree リポジトリーとして提供します。	いいえ	はい	インストーラーイメージから取得される OSTree コミットを提供します。
RHEL for Edge Installer (**.iso**)	**edge-installer** イメージタイプは、実行中のコンテナーからコミットをプルし、埋め込まれた OSTree コミットを使用するように設定された Kickstart ファイルを含むインストール可能なブート ISO を作成します。	いいえ	はい	イメージをフラッシュメディアに書き込み、ISO イメージを必要とする非接続環境で使用します。
RHEL for Edge Raw image (**.raw.xz**)	**edge-raw-image** の圧縮された Raw イメージは、既存のデプロイ済み OSTree コミットを含むパーティションレイアウトを含むファイルから構成されています。	はい	はい	ハードディスクで RHEL Raw イメージをフラッシュするか、仮想マシンで起動して、ベアメタルプラットフォームに使用します。
RHEL for Edge Simplified Installer (**.iso**)	**edge-simplified-installer** イメージは、Ignition ツールを使用して、起動プロセスの初期段階でユーザー設定をイメージに注入します。	はい	はい	デバイスへの無人インストールに使用します。起動後、RHEL for Edge イメージをデバイスにプロビジョニングします。

イメージタイプ	説明	ネットワークベースのデプロイメントに適しています	ネットワークベース以外のデプロイメントに適しています	使用方法
RHEL for Edge AMI (**.ami**)	**edge-ami** イメージは、Ignition ツールを使用して、起動プロセスの初期段階でユーザー設定をイメージに注入します。**.ami** イメージを AWS にアップロードし、AWS で EC2 インスタンスを起動できます。	はい	はい	AWS クラウドで EC2 インスタンスを起動します。
RHEL for Edge VMDK (**.vmdk**)	**edge-vsphere** イメージは、Ignition ツールを使用して、起動プロセスの初期段階でユーザー設定をイメージに注入します。イメージを vSphere にロードし、VM vSphere でイメージを起動できます。	はい	はい	vSphere 仮想マシンで **.vmdk** イメージを起動します。

イメージタイプはその内容によって異なるため、各種デプロイメント環境に適しています。

関連情報

- 標準の RHEL 9 インストールの実行。

1.3. ネットワークベース以外のデプロイメント

RHEL Image Builder を使用して、要件に沿った柔軟な RHEL **rpm-ostree** イメージを作成し、Anaconda を使用してお使いの環境にデプロイします。

composer-cli ツールのコマンドラインインターフェイスから RHEL Image Builder にアクセスするか、RHEL Web コンソールのグラフィカルユーザーインターフェイスを使用できます。

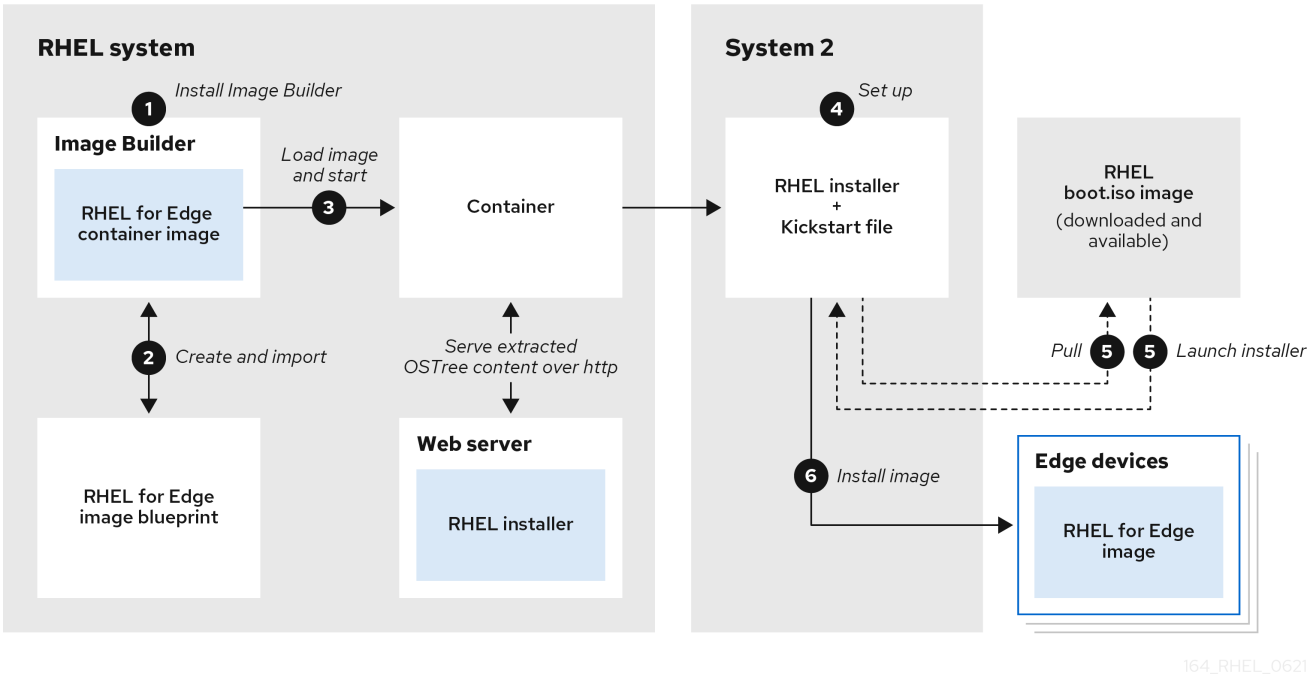
ネットワークベース以外のデプロイメントに RHEL for Edge イメージを作成し、デプロイするには、以下の概要手順を実施します。

1. RHEL システムをインストールおよび登録する
2. RHEL Image Builder をインストールする

3. RHEL Image Builder を使用して RHEL for Edge Container イメージのカスタムブループリントを作成する
4. RHEL Image Builder で RHEL for Edge のブループリントをインポートする
5. OSTreeリポジトリーとしてコミットをデプロイするための Web サーバーを用意した OCI コンテナーに、RHEL for Edge のイメージエンベッドを作成する
6. RHEL for Edge Container イメージファイルをダウンロードする
7. RHEL for Edge Container Commit でリポジトリーを提供するコンテナーをデプロイする
8. RHEL Image Builder を使用して RHEL for Edge インストーラーイメージの別のブループリントを作成する
9. RHEL for Edge Container イメージを組み込んだ実行中のコンテナーからコミットをプルするように設定された RHEL for Edge Installer イメージを作成する
10. RHEL for Edge Installer イメージをダウンロードする
11. インストールを実行する

以下の図は、RHEL for Edge イメージのネットワークを使用しないデプロイメントワークフローを表しています。

図1.1 非ネットワーク環境での RHEL for Edge のデプロイ



1.4. ネットワークベースのデプロイメント

RHEL Image Builder を使用して、要件に沿った柔軟な RHEL **rpm-ostree** イメージを作成し、Anaconda を使用してお使いの環境にデプロイします。RHEL Image Builder は、デプロイメントセットアップの詳細を自動的に識別し、**.tar** ファイル形式の **edge-commit** としてイメージの出力を生成します。

composer-cli ツールのコマンドラインインターフェイスから RHEL Image Builder にアクセスするか、RHEL Web コンソールのグラフィカルユーザーインターフェイスを使用できます。

RHEL for Edge イメージは、以下の概要手順を実行して、作成およびデプロイすることができます。

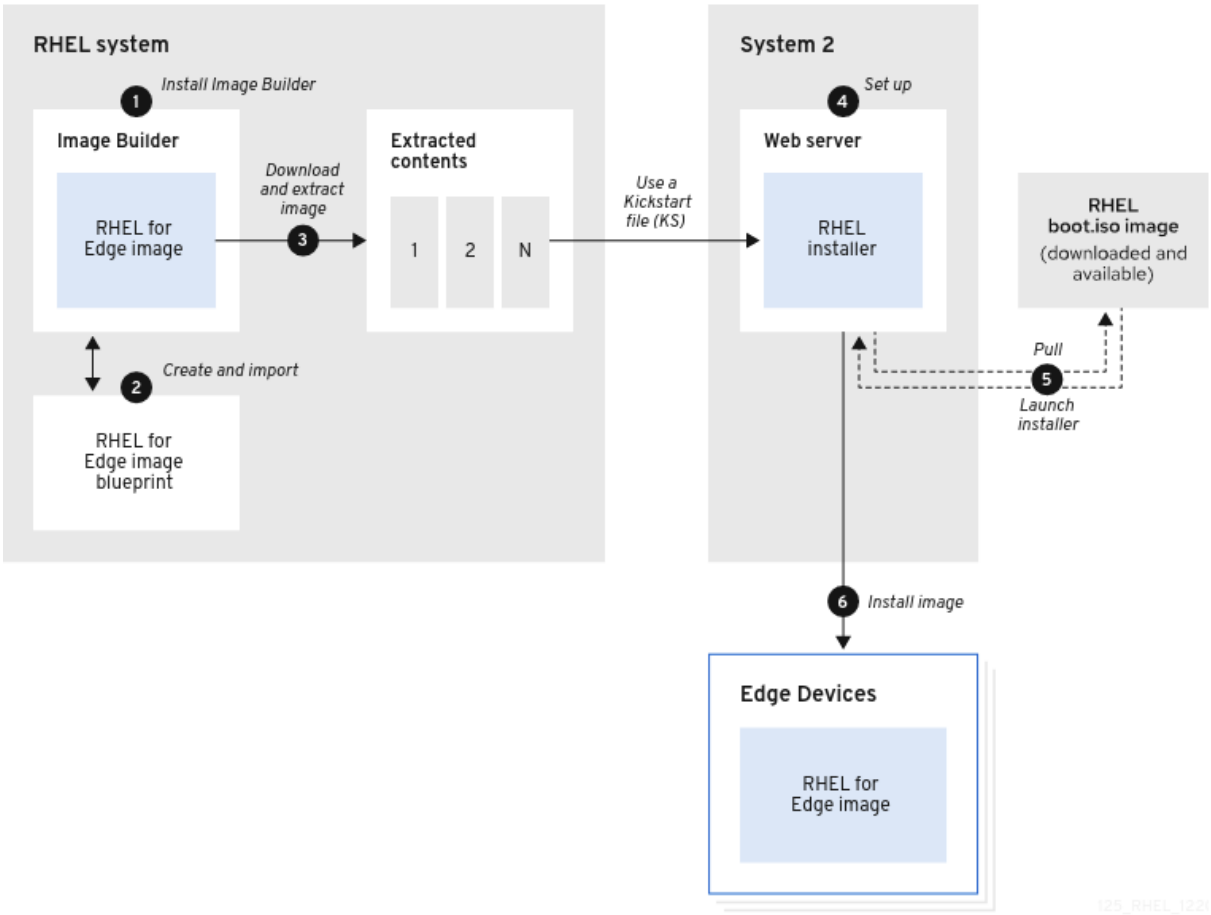
有人インストールの場合

1. RHEL システムをインストールおよび登録する
2. RHEL Image Builder をインストールする
3. RHEL Image Builder を使用して RHEL for Edge イメージのブループリントを作成する
4. RHEL Image Builder で RHEL for Edge のブループリントをインポートする
5. RHEL for Edge Commit (**.tar**) イメージを作成する
6. RHEL for Edge イメージファイをダウンロードする
7. RHEL Image Builder をインストールしたのと同じシステムに、RHEL for Edge Commit コンテンツを提供する Web サーバーをインストールする。手順については、NGINX のセットアップおよび設定 を参照してください。
8. RHEL for Edge Commit (**.tar**) コンテンツを、実行中の Web サーバーに展開する
9. 実行中の Web サーバーから OSTree コンテンツをプルするキックスタートファイルを作成する。キックスタートを変更して OSTree コンテンツをプルする方法の詳細は、RHEL for Edge イメージのコミットの展開 を参照してください。
10. エッジデバイスで RHEL インストーラー ISO を起動し、キックスタートを提供する

無人インストールの場合、RHEL インストール ISO をカスタマイズして、それにキックスタートファイルを埋め込むことができます。

以下の図は、RHEL for Edge ネットワークイメージのデプロイメントワークフローを表しています。

図1.2 ネットワークベースの環境での RHEL for Edge のデプロイ



1.5. RHEL RPM イメージと RHEL FOR EDGE イメージの違い

従来のパッケージベースの RPM 形式で RHEL システムイメージや、RHEL for Edge (**rpm-ostree**) イメージを作成できます。

従来のパッケージベースの RPM を使用して、従来のデータセンターに RHEL をデプロイすることができます。ただし、RHEL for Edge イメージを使用すると、従来のデータセンター以外のサーバーに RHEL をデプロイすることができます。このサーバーには、データが生成されるソース、つまりエッジサーバーに最も近い場所で大量のデータの処理を行うシステムが含まれます。

RHEL for Edge (**rpm-ostree**) イメージはパッケージマネージャーではありません。個々のファイルではなく、完全な起動可能なファイルシステムツリーのみをサポートします。これらのイメージには、これらのファイルがどのように生成されたか、それらの起源に関連するものなど、個々のファイルに関する情報は含まれていません。

rpm-ostree イメージには、追加のアプリケーションを **/var** ディレクトリーにインストールするための別のメカニズムであるパッケージマネージャーが必要です。これにより、**rpm-ostree** イメージは、**/var** ディレクトリーおよび **/etc** ディレクトリーの状態を維持しながら、オペレーティングシステムを変更しないようにします。アトミック更新により、更新のロールバックとバックグラウンドステージングが可能になります。

RHEL for Edge イメージがパッケージベースの RHEL RPM イメージとどのように異なるかを確認するには、以下の表を参照してください。

表1.2 RHEL RPM イメージと RHEL for Edge イメージの違い

主な属性	RHEL RPM イメージ	RHEL for Edge イメージ
OS アセンブリー	パッケージをローカルでアセンブルして、イメージを形成できます。	パッケージは、システムにインストールできる ostree でアセンブルされます。
OS の更新	**dnf update** を使用して、有効なリポジトリーから利用可能な更新を適用することができます。	**/etc/ostree/remotes.d/** の OSTree リモートで新しいコミットが利用可能な場合は、**rpm-ostree upgrade** を使用して更新をステージングできます。更新はシステムの再起動時に有効になります。
リポジトリー	パッケージには DNF リポジトリーが含まれています。	パッケージには OSTree リモートリポジトリーが含まれています。
ユーザーアクセスパーミッション	読み書き	読み取り専用 (**/usr**)
データの永続性	イメージを **tmpfs** 以外のマウントポイントにマウントできます。	**/etc** と **/var** が読み書き可能で、永続的なデータを含んでいます。

第2章 RHEL IMAGE BUILDER の設定

RHEL Image Builder を使用して、カスタマイズした RHEL for Edge イメージを作成します。RHEL システムに RHEL Image Builder をインストールすると、RHEL Image Builder は RHEL Web コンソールでアプリケーションとして利用できます。**composer-cli** ツールのコマンドラインインターフェイスで RHEL Image Builder にアクセスすることもできます。

注記

RHEL Image Builder を仮想マシンにインストールすることが推奨されます。

2.1. IMAGE BUILDER のシステム要件

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

注記

コンテナー内で RHEL Image Builder を実行することは、サポートされていません。

表2.1 Image Builder のシステム要件

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

注記

ホストに RHEL Image Builder をインストールして実行するには、20 GiB のディスク容量要件で十分です。イメージビルドをビルドしてデプロイするには、追加の専用ディスク領域を割り当てる必要があります。

2.2. RHEL IMAGE BUILDER のインストール

RHEL 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

- 仮想マシンが作成され、オンの状態になっている。

- RHEL をインストールし、RHSM または Red Hat Satellite にサブスクライブしている。
- RHEL Image Builder パッケージをインストールできるように、**BaseOS** リポジトリーおよび **AppStream** リポジトリーを有効化している。

手順

1. 仮想マシンに以下のパッケージをインストールします。
   - osbuild-composer
   - composer-cli
   - cockpit-composer
   - bash-completion
   - firewalld

   ```
   # dnf install osbuild-composer composer-cli cockpit-composer bash-completion firewalld
   ```

   RHEL Image Builder が、RHEL Web コンソールでアプリケーションとしてインストールされている。
2. 仮想マシンを再起動します。
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. RHEL Image Builder を有効にします。

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

   osbuild-composer サービスおよび cockpit サービスは、最初のアクセスで自動的に起動します。
5. 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
   ```

関連情報

- リポジトリーの管理

第3章 RHEL IMAGE BUILDER リポジトリーの設定

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

公式リポジトリーのオーバーライド

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

カスタムサードパーティーリポジトリー

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

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

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

前提条件

- カスタムサードパーティーリポジトリーの URL を持っている。

手順

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

```
id = "k8s"
name = "Kubernetes"
type = "yum-baseurl"
url = "https://server.example.com/repos/company_internal_packages/"
check_gpg = false
check_ssl = false
system = false
```

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

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

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

検証

1. 新しいソースが正常に追加されたかどうかを確認します。

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

2. 新しいソースコンテンツを確認します。

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

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

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

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

前提条件

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

手順

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

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

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

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

検証

1. 新しいソースが正常に追加されたかどうかを確認します。

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

2. 新しいソースコンテンツを確認します。

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

3.3. GPG を使用したリポジトリーのメタデータの確認

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

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

前提条件

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

手順

1. リポジトリーを作成するフォルダーにアクセスします。

```
$ cd repo/
```

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

```
$ createrepo_c .
```

3. リポデータがあるディレクトリーにアクセスします。

```
$ cd repodata/
```

4. **repomd.xml** ファイルに署名します。

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

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

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

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

```
check_gpg = true
check_ssl = true
id = "signed local packages"
name = "repository_name"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
check_repogpg = true
gpgkeys=["https://local/keys/repokey.pub"]
```

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

```
check_gpg = true
check_ssl = true
check_repogpg
id = "custom-local"
name = "signed local packages"
```

```
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
gpgkeys=["https://remote/keys/other-repokey.pub",
'''-----BEGIN PGP PUBLIC KEY BLOCK-----
…
-----END PGP PUBLIC KEY BLOCK-----''']
```

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

```
$ GPG verification is enabled, but GPG signature is not available.
This may be an error or the repository does not support GPG verification:
Status code: 404 for http://repo-server/rhel/repodata/repomd.xml.asc (IP:
192.168.1.3)
```

- 署名が無効な場合、GPG ツールは次のようなエラーを表示します。

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

検証

- リポジトリーの署名を手動でテストします。

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

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

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

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

3.5. システムリポジトリーのオーバーライド

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

前提条件

- ホストシステムからアクセスできるカスタムリポジトリーがある。

手順

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

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

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

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

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

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

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

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

4. オプション: JSON ファイルを確認します。

   ```
   $ json_verify  /etc/osbuild-composer/repositories/<file>.json
   ```

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

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

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

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

検証

- リポジトリーが正しい URL を指しているか確認します。

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

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

関連情報

- リポジトリーで利用可能な最新の RPM バージョンは、**osbuild-composer** では表示されません。

3.6. サブスクリプションが必要なシステムリポジトリーのオーバーライド

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

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

注記

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

前提条件

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

手順

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

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

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

```
{
  "x86_64": [
    {
      "name": "AppStream mirror example",
```

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

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

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

関連情報

- RHEL image builder uses CDN repositories when host is registered to Satellite 6

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

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

前提条件

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

手順

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

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

次の出力は例です。

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

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

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

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

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

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

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

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

d. 設定が正しく置き換えられたことを確認します。

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

3. サービスを再起動します。

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

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

関連情報

- RHEL image builder fails when multiple custom repositories are defined on the Satellite

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

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

前提条件

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

手順

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

次のステップ

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

第4章 RHEL WEB コンソールで IMAGE BUILDER を使用して RHEL FOR EDGE イメージを作成する

RHEL Image Builder を使用して、カスタマイズした RHEL for Edge イメージ (OSTree コミット) を作成します。

RHEL Image Builder にアクセスし、カスタマイズした RHEL for Edge イメージを作成するには、RHEL Web コンソールインターフェイスまたはコマンドラインインターフェイスを使用します。

以下の概要手順を実行して、RHEL Web コンソールで RHEL Image Builder を使用し、RHEL for Edge イメージを作成することができます。

1. RHEL Web コンソールで RHEL Image Builder にアクセスします。
2. RHEL for Edge イメージのブループリントの作成します。
3. RHEL for Edge イメージの作成します。次のイメージを作成できます。
   - RHEL for Edge Commit イメージ。
   - RHEL for Edge Container イメージ。
   - RHEL for Edge Installer イメージ。
4. RHEL for Edge イメージをダウンロードする

4.1. RHEL WEB コンソールで RHEL IMAGE BUILDER にアクセスする

RHEL Web コンソールで RHEL Image Builder にアクセスするには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL システムをインストール済みである。
- システムの管理者権限を持っている。
- RHEL システムを Red Hat Subscription Manager (RHSM) または Red Hat Satellite Server にサブスクライブしている。
- システムに電源が入り、ネットワーク経由でアクセスできる。
- システムに RHEL Image Builder がインストール済みである。

手順

1. RHEL システムで、Web ブラウザーで https://localhost:9090/ にアクセスします。
2. RHEL Image Builder にリモートでアクセスする方法の詳細は、RHEL 9 Web コンソールを使用したシステムの管理 を参照してください。
3. 管理ユーザーアカウントを使用して、Web コンソールにログインします。
4. Web コンソールで、左側のメニューの **Apps** をクリックします。
5. **Image Builder** をクリックします。

RHEL Image Builder ダッシュボードが右側のペインに開きます。これで、RHEL for Edge イメージのブループリントの作成に進むことができます。

4.2. WEB コンソールで IMAGE BUILDER を使用して RHEL FOR EDGE イメージのブループリントを作成する

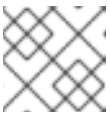
RHEL Web コンソールの RHEL Image Builder を使用して RHEL for Edge イメージのブループリントを作成するには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL システムで、RHEL Image Builder のダッシュボードを開いている。

手順

1. RHEL Image Builder ダッシュボードで、**Create Blueprint** をクリックします。**Create Blueprint** のダイアログボックスが表示されます。
2. **Details** ページで以下を行います。
   a. ブループリントの名前と、必要に応じてその説明を入力します。**Next** をクリックします。
3. オプション: **Packages** ページで以下を行います。
   a. **Available packages** の検索で、パッケージ名を入力し、**>** ボタンをクリックして、パッケージを Chosen packages フィールドに移動します。必要な数のパッケージを検索して含めます。**Next** をクリックします。



注記

特に指定がない限り、これらのカスタマイズはすべてオプションです。

4. **Kernel** ページで、カーネル名とコマンドライン引数を入力します。
5. **File system** ページで、**Use automatic partitioning** を選択します。OSTree イメージには読み取り専用などの独自のマウントルールがあるため、ファイルシステムのカスタマイズは OSTree システムではサポートされません。**Next** をクリックします。
6. **Services** ページで、サービスを有効または無効にします。
   a. 有効または無効にするサービス名をコンマまたはスペースで区切るか、**Enter** キーを押して入力します。**Next** をクリックします。
7. **Firewall** ページで、ファイアウォールを設定します。
   a. **Ports** と、有効または無効にするファイアウォールサービスを入力します。
   b. **Add zone** ボタンをクリックして、各ゾーンのファイアウォールルールを個別に管理します。**Next** をクリックします。
8. **Users** ページで、以下の手順に従ってユーザーを追加します。
   a. **Add user** をクリックします。

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

9. **Groups** ページで、次の手順を実行してグループを追加します。

a. **Add groups** ボタンをクリックします。

i. **Group name** と **Group ID** を入力します。グループをさらに追加できます。**Next** をクリックします。

10. **SSH keys** ページで、キーを追加します。

a. **Add key** ボタンをクリックします。

i. SSH キーを入力します。

ii. **User** を入力します。**Next** をクリックします。

11. **Timezone** ページで、タイムゾーンを設定します。

a. **Timezone** フィールドに、システムイメージに追加するタイムゾーンを入力します。たとえば、次のタイムゾーン形式を追加します。"US/Eastern"。
タイムゾーンを設定しない場合、システムはデフォルトとして協定世界時 (UTC) を使用します。

b. **NTP** サーバーを入力します。**Next** をクリックします。

12. **Locale** ページで、以下の手順を実行します。

a. **Keyboard** 検索フィールドに、システムイメージに追加するパッケージ名を入力します。たとえば、["en_US.UTF-8"] と入力します。

b. **Languages** 検索フィールドに、システムイメージに追加するパッケージ名を入力します。たとえば、"us" と入力します。**Next** をクリックします。

13. **Others** ページで、次の手順を実行します。

a. **Hostname** フィールドに、システムイメージに追加するホスト名を入力します。ホスト名を追加しない場合、オペレーティングシステムによってホスト名が決定されます。

b. Simplifier Installer イメージでのみ必須:**Installation Devices** フィールドに、システムイメージの有効なノードを入力します。たとえば、**dev/sda** です。**Next** をクリックします。

14. FIDO イメージをビルドする場合にのみ必須:**FIDO device onboarding** ページで、次の手順を実行します。

a. **Manufacturing server URL** フィールドに、次の情報を入力します。

i. **DIUN public key insecure** フィールドに、セキュアでない公開鍵を入力します。

ii. **DIUN public key hash** フィールドに、公開鍵ハッシュを入力します。

iii. **DIUN public key root certs** フィールドに、公開鍵ルート証明書を入力します。**Next** をクリックします。

15. **OpenSCAP** ページで、次の手順を実行します。

a. **Datastream** フィールドに、システムイメージに追加する **datastream** 修復手順を入力します。

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

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

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

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

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

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

4.3. WEB コンソールで IMAGE BUILDER を使用して RHEL FOR EDGE COMMIT イメージを作成する

RHEL Web コンソールの RHEL Image Builder を使用して、**"RHEL for Edge Commit"**イメージを作成できます。**RHEL for Edge Commit (.tar)** イメージタイプには完全なオペレーティングシステムが含まれていますが、直接起動することはできません。Commit イメージタイプを起動するには、実行中のコンテナーにデプロイする必要があります。

前提条件

- RHEL システムで、RHEL Image Builder ダッシュボードにアクセス済みである。

手順

1. RHEL Image Builder のダッシュボードで、**Create Image** をクリックします。
2. **Image output** ページで、次の手順を実行します。

a. **Select a blueprint** ドロップダウンメニューから、使用するブループリントを選択します。

b. **Image output type** ドロップダウンリストから、**"RHEL for Edge Commit (.tar)"**を選択します。

c. **Next** をクリックします。

d. **OSTree settings** ページで、以下を入力します。

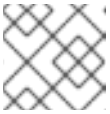
i. **リポジトリー URL**: イメージに埋め込むコミットの OSTree リポジトリーへの URL を指定します。たとえば、http://10.0.2.2:8080/repo/ です。

ii. **親コミット**: 以前のコミットを指定するか、現時点でコミットがない場合は空のままにします。

iii. **Ref** テキストボックスで、コミットを作成する場所の参照パスを指定します。デフォルトでは、Web コンソールは **rhel/9/$ARCH/edge** を指定します。$ARCH の値は、ホストマシンによって決定されます。**Next** をクリックします。

e. **Review** ページでカスタマイズを確認し、**Create** をクリックします。

RHEL Image Builder が、作成したブループリントの RHEL for Edge Commit イメージの作成を開始します。



注記

イメージの作成プロセスは、完了するまでに最大 20 分かかります。

検証

1. RHEL for Edge Commit イメージの作成の進行状況を確認するには、次の手順に従います。
    a. **Images** タブをクリックします。

イメージの作成プロセスが完了したら、結果の **RHEL for Edge Commit (.tar)** イメージをダウンロードできます。

関連情報

- RHEL for Edge イメージのダウンロード

4.4. RHEL WEB コンソールで RHEL IMAGE BUILDER を使用して RHEL FOR EDGE CONTAINER イメージを作成する

RHEL for Edge Container (.tar) を選択すると、RHEL for Edge イメージを作成できます。 **RHEL for Edge Container (.tar)** イメージタイプでは、OSTree のコミットを作成し、Web サーバーのある OCI コンテナーに埋め込みます。コンテナーが起動すると、Web サーバーは OSTree リポジトリーとしてコミットを提供します。

この手順に従って、RHEL Web コンソールの Image Builder を使用して RHEL for Edge Container イメージを作成します。

前提条件

- RHEL システムで、RHEL Image Builder ダッシュボードにアクセス済みである。
- ブループリントを作成している。

手順

1. RHEL Image Builder のダッシュボードで、**Create Image** をクリックします。
2. **Image output** ページで、次の手順を実行します。
3. **Select a blueprint** ドロップダウンメニューから、使用するブループリントを選択します。
    a. **Image output type** ドロップダウンリストから、**"RHEL for Edge Container (.tar)"**を選択します。
    b. **Next** をクリックします。
    c. **OSTree** ページで、次のように入力します。
        i. **リポジトリー URL**: イメージに埋め込むコミットの OSTree リポジトリーへの URL を指定します。たとえば、http://10.0.2.2:8080/repo/ です。デフォルトでは、RHEL for Edge Container イメージのリポジトリーフォルダーは /repo です。

使用する正しい URL を見つけるには、実行中のコンテナーにアクセスし、**nginx.conf** ファイルを確認します。使用する URL を見つけるには、実行中のコンテナーにアクセスし、**nginx.conf** ファイルを確認します。**nginx.conf** ファイル内で、**root** ディレクトリーエントリーを見つけて、**/repo/** フォルダー情報を検索します。RHEL Image Builder を使用して RHEL for Edge Container イメージ **(.tar)** を作成するときにリポジトリー URL を指定しない場合、デフォルトの **/repo/** エントリーが **nginx.conf** ファイルに作成されることに注意してください。

ii. **親コミット**: 以前のコミットを指定するか、現時点でコミットがない場合は空のままにします。

iii. **Ref** テキストボックスで、コミットを作成する場所の参照パスを指定します。デフォルトでは、Web コンソールは **rhel/9/$ARCH/edge** を指定します。$ARCH の値は、ホストマシンによって決定されます。**Next** をクリックします。

d. **Review** ページで、カスタマイズを確認します。**Save blueprint** をクリックします。

4. **Create** をクリックします。
RHEL Image Builder が、作成したブループリントの RHEL for Edge Container イメージの作成を開始します。

注記

イメージの作成プロセスは、完了するまでに最大 20 分かかります。

検証

1. RHEL for Edge Container イメージの作成の進行状況を確認するには、次の手順に従います。

a. **Images** タブをクリックします。

イメージの作成プロセスが完了したら、結果の **RHEL for Edge Container (.tar)** イメージをダウンロードできます。

関連情報

- RHEL for Edge イメージのダウンロード

4.5. RHEL WEB コンソールで IMAGE BUILDER を使用して RHEL FOR EDGE INSTALLER イメージを作成する

RHEL for Edge Installer (.iso) を選択すると、RHEL for Edge インストーラーイメージを作成できます。**RHEL for Edge Installer (.iso)** イメージタイプは、**RHEL for Edge Container (.tar)** によって提供される実行中のコンテナーから OSTree コミットリポジトリーをプルし、組み込み OSTree を使用するように設定されたキックスタートファイルを使用してインストール可能なブート ISO イメージを作成します。

以下の手順に従って、RHEL Web コンソールで Image Builder を使用して RHEL for Edge イメージを作成します。

前提条件

- RHEL システムで、Image Builder ダッシュボードにアクセス済みである。
- ブループリントを作成している。

- RHEL for Edge Container イメージを作成し、実行中のコンテナーにロードした。ネットワークベース以外のデプロイメント用の RHEL for Edge Container イメージの作成 を参照してください。

手順

1. RHEL Image Builder のダッシュボードで、**Create Image** をクリックします。
2. **Image output** ページで、次の手順を実行します。
   a. **Select a blueprint** ドロップダウンメニューから、使用するブループリントを選択します。
   b. **Image output type** ドロップダウンリストから、**RHEL for Edge Installer (.iso)** イメージを選択します。
   c. **Next** をクリックします。
   d. **OSTree settings** ページで、以下を入力します。
      i. **リポジトリー URL**: イメージに埋め込むコミットの OSTree リポジトリーへの URL を指定します。たとえば、http://10.0.2.2:8080/repo/ です。
      ii. **Ref** テキストボックスで、コミットを作成する場所の参照パスを指定します。デフォルトでは、Web コンソールは **rhel/9/$ARCH/edge** を指定します。$ARCH の値は、ホストマシンによって決定されます。**Next** をクリックします。
   e. **Review** ページで、カスタマイズを確認します。**Save blueprint** をクリックします。
3. **Create** をクリックします。
   RHEL Image Builder が、作成したブループリントの RHEL for Edge Installer イメージの作成を開始します。

注記

イメージの作成プロセスは、完了するまでに最大 20 分かかります。

検証

1. RHEL for Edge Installer イメージの作成の進行状況を確認するには:
   a. **Images** タブをクリックします。

イメージの作成プロセスが完了したら、結果の **RHEL for Edge Installer (.iso)** イメージをダウンロードして、ISO イメージをデバイスで起動できます。

関連情報

- RHEL for Edge イメージのダウンロード

4.6. RHEL FOR EDGE イメージのダウンロード

RHEL Image Builder を使用して RHEL for Edge イメージを正常に作成したら、ローカルホストにイメージをダウンロードします。

手順

以下は、イメージをダウンロードするための手順です。

1. **More Options** メニューの **Download** をクリックします。
   RHEL Image Builder ツールは、デフォルトのダウンロード場所にファイルをダウンロードします。

ダウンロードされたファイルは、RHEL for Edge Commit イメージおよび RHEL for Edge Container イメージの OSTree リポジトリーを含む **.tar** ファイル、または RHEL for Edge Installer イメージの **.iso** ファイルと OSTree リポジトリーで構成されます。このリポジトリーには、コミットと、リポジトリーのコンテンツに関する情報のメタデータを含む **json** ファイルが含まれています。

4.7. 関連情報

- Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

第5章 IMAGE BUILDER コマンドラインを使用した RHEL FOR EDGE イメージの作成

Image Builder を使用して、カスタマイズした RHEL for Edge イメージ (OSTree コミット) を作成できます。

Image Builder にアクセスし、カスタマイズした RHEL for Edge イメージを作成するには、RHEL Web コンソールインターフェイスまたはコマンドラインインターフェイスを使用します。

ネットワークベースのデプロイメントの場合、CLI を使用して RHEL for Edge イメージを作成するワークフローでは、以下の概要手順を実施します。

1. RHEL for Edge イメージのブループリントの作成
2. RHEL for Edge Commit イメージの作成
3. RHEL for Edge Commit イメージのダウンロード

ネットワークベース以外のデプロイメントの場合、CLI を使用して RHEL for Edge イメージを作成するワークフローでは、以下の概要手順を実施します。

1. RHEL for Edge イメージのブループリントの作成
2. RHEL for Edge インストーラーイメージのブループリントの作成
3. RHEL for Edge Container イメージの作成
4. RHEL for Edge Installer イメージを作成する
5. RHEL for Edge イメージをダウンロードする

手順を実行するには、**composer-cli** パッケージを使用します。

注記

composer-cli コマンドを root 以外のユーザーとして実行するには、**weldr** グループの一員であるか、システムへの管理者アクセス権を持っている必要があります。

5.1. ネットワークベースのデプロイメントワークフロー

ここでは、**OSTree** コミットを構築する手順を説明します。これらの **OSTree** コミットには完全なオペレーティングシステムが含まれていますが、直接起動することはできません。それらを起動するには、キックスタートファイルを使用してデプロイメントする必要があります。

5.1.1. Image Builder コマンドラインインターフェイスを使用した RHEL for Edge Commit イメージのブループリントの作成

CLI を使用して、RHEL for Edge Commit イメージのブループリントを作成します。

前提条件

- 既存のブループリントがありません。それを確認するには、既存のブループリントをリスト表示します。

```
$ sudo composer-cli blueprints list
```

手順

1. 次の内容のプレーンテキストファイルを TOML 形式で作成します。

```
name = "blueprint-name"
description = "blueprint-text-description"
version = "0.0.1"
modules = [ ]
groups = [ ]
```

詳細は以下のようになります。

- **blueprint-name** はブループリントの名前、blueprint-text-description はブループリントの説明です。
- **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。
- **モジュール** には、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します (例: パッケージ名 = "tmux"、対応するバージョンの glob = "2.9a")。
  現在、パッケージとモジュールには違いがないことに注意してください。
- **グループ** は、イメージにインストールされるパッケージグループです (例: グループパッケージ anaconda-tools)。
  このとき、モジュールおよびグループがわからない場合は、空欄のままにしておきます。

2. 要件に沿うように、必要なパッケージを含め、ブループリントの他の詳細をカスタマイズします。
   ブループリントに含むすべてのパッケージについて、以下の行をファイルに追加します。

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

詳細は以下のようになります。

- package-name は、パッケージ名に置き換えます (例: httpd, gdb-doc、coreutils など)。
- package-version は、使用するパッケージのバージョン番号に置き換えます。
  package-version は、以下の dnf バージョン仕様をサポートしています。
- 特定のバージョンを指定する場合は、バージョン番号を正確に指定してください (例: 9.0 など)。
- 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、9.* などの形式を使用してください。

3. ブループリントを RHEL Image Builder サーバーにプッシュ (インポート) します。

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

4. 既存のブループリントを一覧表示して、作成したブループリントが正常にプッシュされて存在するかどうかを確認します。

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

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

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

関連情報

- サポートされているイメージのカスタマイズ

5.1.2. Image Builder コマンドラインインターフェイスを使用した RHEL for Edge Commit イメージの作成

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge Commit イメージを作成するには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge Commit イメージのブループリントを作成している。

手順

1. RHEL for Edge Commit イメージを作成します。

```
# composer-cli compose start blueprint-name image-type
```

詳細は以下のようになります。

- **blueprint-name** は RHEL for Edge のブループリント名です。
- **image-type** は **network-based deployment** の **edge-commit** です。
composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```

注記

イメージの作成プロセスは、完了するまでに最大 20 分かかります。

イメージ作成プロセスを中断するには、以下を実行します。

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

既存イメージを削除するには、以下を実行します。

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

イメージの準備ができたら、イメージをダウンロードして **ネットワークデプロイメント** で使用できます。

関連情報

- RHEL Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

5.1.3. RHEL Image Builder CLI を使用した ref commit による RHEL for Edge イメージ更新の作成

既存のブループリントに変更を加えた場合、たとえば、新しいパッケージを追加し、既存の RHEL for Edge イメージをこの新しいパッケージで更新する場合、**--parent** 引数を使用して更新された **RHEL for Edge Commit (.tar)** イメージを生成できます。**--parent** 引数には、**URL** 引数で指定されたリポジトリーに存在する **ref** を指定することも、展開された **.tar** イメージファイル内にある **Commit ID** を使用することもできます。**ref** 引数と **Commit ID** 引数はどちらも、ビルドしている新しいコミットの親を取得します。RHEL Image Builder は、ビルド中の新しいコミットの一部に影響する情報を親コミットから読み取ることができます。その結果、RHEL Image Builder は親コミットのユーザーデータベースを読み取り、パッケージで作成されたシステムユーザーとグループの UID と GID を保持します。

前提条件

- RHEL for Edge イメージの既存のブループリントを更新している。
- 既存の RHEL for Edge イメージ (OSTree コミット) がある。RHEL for Edge イメージのコミットの展開 を参照してください。
- ビルド中の **ref** は、URL で指定された **OSTree** リポジトリーで利用できます。

手順

1. RHEL for Edge Commit イメージを作成します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --parent parent-OSTree-REF
--url URL blueprint-name image-type
```

以下に例を示します。

- **parent** に基づいて新しい **ref** を使用して新しい RHEL for Edge Commit を作成するには、次のコマンドを実行します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --parent
rhel/9/x86_64/edge --url http://10.0.2.2:8080/repo rhel_update edge-commit
```

- 同じ **ref** に基づいて新しい RHEL for Edge Commit を作成するには、次のコマンドを実行します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url
http://10.0.2.2:8080/repo rhel_update edge-commit
```

ここでは、以下のようになります。

- **--ref** 引数は、OSTree リポジトリーの構築に使用したものと同じパス値を指定します。
- **--parent** 引数は親コミットを指定します。**rhel/9/x86_64/edge**、または展開された **.tar** ファイルで見つかる **Commit ID** など、解決およびプルする ref を指定できます。
- **blueprint-name** は RHEL for Edge のブループリント名です。
- **--url** 引数は、イメージに埋め込むコミットの OSTree リポジトリーへの URL を指定します (例: http://10.0.2.2:8080/repo)。
- **image-type** は **network-based deployment** の **edge-commit** です。

注記

- **--parent** 引数は、**RHEL for Edge Commit (.tar)** イメージタイプにのみ使用できます。**--url** 引数と **--parent** 引数をあわせて使用すると、**RHEL for Edge Container (.tar)** イメージタイプでエラーが発生します。
- **parent ref** 引数を省略すると、システムは **--ref** 引数で指定された **ref** にフォールバックします。

composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```

注記

イメージ作成プロセスの完了まで数分かかります。

(オプション) イメージ作成プロセスを中断するには、以下を実行します。

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

(オプション) 既存のイメージを削除するには、以下を実行します。

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

イメージの作成が完了した後、既存の OSTree デプロイメントをアップグレードするには、以下が必要です。

- リポジトリーを設定します。Deploying a RHEL for Edge image を参照してください。
- このリポジトリーをリモート、つまり、OSTree コンテンツをホストする http または https のエンドポイントとして追加します。
- 新しい OSTree コミットを既存の稼働中のインスタンスにプルします。RHEL for Edge イメージの更新の手動でのデプロイ を参照してください。

関連情報

- RHEL Image Builder コマンドラインインターフェイスでのシステムイメージの作成
- RHEL Image Builder コマンドラインインターフェイスを使用した RHEL for Edge イメージのダウンロード

5.1.4. Image Builder コマンドラインインターフェイスを使用した RHEL for Edge イメージのダウンロード

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge イメージをダウンロードするには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge イメージを作成済みである。

手順

1. RHEL for Edge イメージの状態を確認します。

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

出力には、以下が表示される必要があります。

```
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
```

2. イメージをダウンロードします。

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

RHEL Image Builder が、**tar** ファイル形式のイメージをカレントディレクトリーにダウンロードします。

UUID 番号とイメージサイズは並んで表示されます。

```
$ <UUID>-commit.tar: size MB
```

イメージには、コミットと、リポジトリーコンテンツに関する情報メタデータを含む **json** ファイルが含まれています。

関連情報

- ネットワークベース環境での RHEL for Edge イメージのデプロイ

5.2. 非ネットワークベースのデプロイメントワークフロー

RHEL for Edge Container イメージと RHEL for Edge Installer イメージを使用して OSTree ベースのシステムをインストールし、後で切断された環境のデバイスにデプロイできるブート ISO イメージを構築するには、以下の手順を実行します。

5.2.1. Image Builder CLI を使用したRHEL for Edge Container のブループリントの作成

RHEL for Edge Container イメージのブループリントを作成するには、以下のステップを実行します。

手順

1. 次の内容のプレーンテキストファイルを TOML 形式で作成します。

```
name = "blueprint-name"
description = "blueprint-text-description"
version = "0.0.1"
modules = [ ]
groups = [ ]
```

詳細は以下のようになります。

- **blueprint-name** はブループリントの名前、blueprint-text-description はブループリントの説明です。
- **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。
- **モジュール** には、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します (例: パッケージ名 = "tmux"、対応するバージョンの glob = "2.9a")。
  現在、パッケージとモジュールには違いがないことに注意してください。
- **グループ** は、イメージにインストールされるパッケージグループです (例: グループパッケージ anaconda-tools)。
  このとき、モジュールおよびグループがわからない場合は、空欄のままにしておきます。

2. 要件に沿うように、必要なパッケージを含め、ブループリントの他の詳細をカスタマイズします。
ブループリントに含むすべてのパッケージについて、以下の行をファイルに追加します。

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

詳細は以下のようになります。

- package-name は、**httpd**、**gdb-doc**、**coreutils** などのパッケージの名前です。
- package-version は、使用するパッケージのバージョン番号に置き換えます。
  package-version は、次の **dnf** バージョン仕様をサポートしています。
- 特定のバージョンを指定する場合は、バージョン番号を正確に指定してください (例: 9.0 など)。

- 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、9.* などの形式を使用してください。

3. ブループリントを RHEL Image Builder サーバーにプッシュ (インポート) します。

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

4. 既存のブループリントを一覧表示して、作成したブループリントが正常にプッシュされて存在するかどうかを確認します。

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

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

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

関連情報

- サポートされているイメージのカスタマイズ

5.2.2. Image Builder CLI を使用した RHEL for Edge Installer のブループリントの作成

ブループリントを作成して **RHEL for Edge Installer (.iso)** イメージを構築し、ユーザーアカウントを指定して、インストール時にシステム上に 1 人以上のユーザーを自動的に作成できます。

警告

customizations.user カスタマイズを使用してブループリントでユーザーを作成すると、ブループリントはユーザーを **/usr/lib/passwd** ディレクトリーの下に、パスワードを **/usr/etc/shadow** ディレクトリーの下に作成します。**OSTree** 更新を使用している実行中のシステムで、さらにバージョンの高いイメージのパスワードを変更することはできない点に注意してください。ブループリントで作成したユーザーは、作成したシステムにアクセスするためだけに使用する必要があります。システムにアクセスしたら、**useradd** コマンドなどを使用してユーザーを作成する必要があります。

RHEL for Edge Installer イメージのブループリントを作成するには、以下の手順を実行します。

手順

1. 次の内容のプレーンテキストファイルを TOML 形式で作成します。

```
name = "blueprint-installer"
description = "blueprint-for-installer-image"
version = "0.0.1"

[[customizations.user]]
```

```
name = "user"
description = "account"
password = "user-password"
key = "user-ssh-key "
home = "path"
groups = ["user-groups"]
```

詳細は以下のようになります。

- **blueprint-name** はブループリントの名前、blueprint-text-description はブループリントの説明です。
- **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. ブループリントを RHEL Image Builder サーバーにプッシュ (インポート) します。

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

3. 既存のブループリントを一覧表示して、作成したブループリントが正常にプッシュされて存在するかどうかを確認します。

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

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

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

関連情報

- サポートされているイメージのカスタマイズ

5.2.3. Image Builder CLI を使用した RHEL for Edge Container イメージの作成

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge Container イメージを作成するには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge Container イメージのブループリントを作成しました。

手順

1. RHEL for Edge Container イメージを作成します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url URL-OSTree-repository blueprint-name image-type
```

詳細は以下のようになります。

- **--ref** は、お客様が ostree リポジトリーの構築に使用した値と同じです。
- **--url** は、イメージに埋め込むコミットの OSTree リポジトリーへの URL ですたとえば、http://10.0.2.2:8080/repo/ です。デフォルトでは、RHEL for Edge Container イメージの

リポジトリーフォルダーは /repo です。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。
使用する正しい URL を見つけるには、実行中のコンテナーにアクセスし、**nginx.conf** ファイルを確認します。使用する URL を見つけるには、実行中のコンテナーにアクセスし、**nginx.conf** ファイルを確認します。**nginx.conf** ファイル内で、**root** ディレクトリーエントリーを見つけて、/**repo**/ フォルダー情報を検索します。RHEL Image Builder を使用して RHEL for Edge Container イメージ **(.tar)** を作成するときにリポジトリー URL を指定しない場合、デフォルトの /**repo**/ エントリーが **nginx.conf** ファイルに作成されることに注意してください。

- **blueprint-name** は RHEL for Edge のブループリント名です。
- **image-type** は、**非ネットワークベースのデプロイメント** 用の **edge-container** です。
composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```

注記

イメージの作成プロセスは、完了するまでに最大 20 分かかります。

イメージ作成プロセスを中断するには、以下を実行します。

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

既存イメージを削除するには、以下を実行します。

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

イメージの準備ができたら、**ネットワーク以外のデプロイメント** に使用できます。ネットワークベース以外のデプロイメント用の RHEL for Edge Container イメージの作成 を参照してください。

関連情報

- RHEL Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

5.2.4. ネットワークベース以外のデプロイメント向けのコマンドラインインターフェイスを使用した RHEL for Edge インストーラーイメージの作成

OSTree コミットを組み込む RHEL for Edge インストーラーイメージを作成するには、RHEL Image Builder コマンドラインインターフェイスを使用して、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge Installer イメージのブループリントを作成した。
- RHEL for Edge Container イメージを作成し、Web サーバーを使用してデプロイした。

手順

1. RHEL for Edge Installer インストーラーイメージの作成を開始します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url URL-OSTree-repository blueprint-name image-type
```

詳細は以下のようになります。

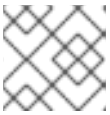
- **ref** は、お客様が ostree リポジトリーの構築に使用した値と同じです。
- **URL-OSTree-repository** は、イメージに埋め込むコミットの OSTree リポジトリーへの URL です。たとえば、http://10.0.2.2:8080/repo です。ネットワークベース以外のデプロイメント用の RHEL for Edge Container イメージの作成 を参照してください。
- **blueprint-name** は、RHEL for Edge Installer のブループリント名です。
- **image-type** は **edge-installer** です。
  composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

コマンド出力には、以下の形式で状態が表示されます。

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



注記

イメージ作成プロセスの完了まで数分かかります。

イメージ作成プロセスを中断するには、以下を実行します。

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

既存イメージを削除するには、以下を実行します。

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

イメージの準備ができたら、**ネットワーク以外のデプロイメント** に使用できます。ネットワークベース以外のデプロイメント向けの RHEL for Edge イメージのインストール を参照してください。

5.2.5. Image Builder CLI を使用した RHEL for Edge Installer イメージのダウンロード

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge Installer イメージをダウンロードするには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge Installer イメージを作成している。

手順

1. RHEL for Edge イメージの状態を確認します。

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

出力には、以下が表示される必要があります。

```
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
```

2. イメージをダウンロードします。

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

RHEL Image Builder が、**.iso** ファイル形式のイメージをカレントディレクトリーにダウンロードします。

UUID 番号とイメージサイズは並んで表示されます。

```
$ <UUID>-boot.iso: size MB
```

結果のイメージは、起動可能な ISO イメージです。

関連情報

- 非ネットワークベース環境での RHEL for Edge イメージのデプロイ

5.3. サポートされているイメージのカスタマイズ

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

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

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

関連情報

- ファイルシステムのカスタマイズサイズを追加した後、ブループリントのインポートが失敗します。

5.3.1. ディストリビューションの選択

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

手順

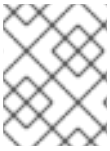
- 指定の RHEL イメージを常にビルドするように、RHEL ディストリビューションを使用してブループリントをカスタマイズします。

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

別のマイナーバージョンをビルドするには、**"different_minor_version"** を置き換えます。たとえば、RHEL 9.4 イメージをビルドする場合は、**distro** = "rhel-94" を使用します。RHEL 9.3 イメージでは、RHEL 9.3、RHEL 8.9、およびそれ以前のリリースなどのマイナーバージョンをビルドできます。

5.3.2. パッケージグループの選択

パッケージとモジュールを使用してブループリントをカスタマイズします。**name** 属性は必須の文字列です。**version** 属性はオプションの文字列で、指定しない場合はリポジトリー内の最新バージョンが使用されます。



注記

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

手順

- パッケージを使用してブループリントをカスタマイズします。

```
[[packages]]
name = "package_group_name"
```

"**package_group_name**" は、パッケージグループの名前に置き換えます。たとえば、"tmux" とします。

```
[[packages]]
name = "tmux"
version = "2.9a"
```

5.3.3. イメージのホスト名の設定

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

手順

- ブループリントをカスタマイズしてホスト名を設定します。

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

5.3.4. 追加ユーザーの指定

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

手順

- ブループリントをカスタマイズして、イメージにユーザーを追加します。

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

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

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

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

その他のプレースホルダーを、適切な値に置き換えます。

name の値を入力し、不要な行は省略します。

追加するすべてのユーザーにこのブロックを繰り返します。

5.3.5. 追加グループの指定

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

手順

- グループを使用してブループリントをカスタマイズします。

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

追加するすべてのグループにこのブロックを繰り返します。

5.3.6. 既存ユーザーの SSH キーの設定

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

手順

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

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

注記

既存ユーザーに対してのみ、**customizations.sshkey** カスタマイズを設定できます。ユーザーの作成と SSH キーの設定は、**システムイメージのカスタマイズに関するユーザー仕様** を参照してください。

5.3.7. カーネル引数の追加

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

手順

- デフォルト設定にカーネルの起動パラメーターオプションを追加します。

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

- イメージで使用するカーネル名を定義

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

5.3.8. タイムゾーンと NTP の設定

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

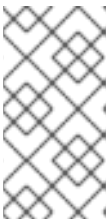
手順

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

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

以下に例を示します。

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



注記

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

5.3.9. ロケール設定のカスタマイズ

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

手順

- ロケール設定を行います。

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

以下に例を示します。

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

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

```
$ localectl list-locales
```

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

```
$ localectl list-keymaps
```

5.3.10. ファイアウォールのカスタマイズ

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

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

注記

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

手順

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

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

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

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

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

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

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

- 利用可能なファイアウォールサービスを確認できます。

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

以下に例を示します。

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

注記

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

5.3.11. サービスの有効化または無効化

システムの起動時に有効にするサービスを制御することができます。一部のイメージタイプでは、イ

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

手順

- 起動時に有効にするサービスをカスタマイズします。

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

以下に例を示します。

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

5.3.12. カスタムファイルシステム設定の指定

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

- セキュリティーベンチマークへの準拠
- ディスク外エラーに対する保護
- パフォーマンスの向上
- 既存のセットアップとの整合性

注記

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

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

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

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

RHEL 8.10 および 9.4 より前のリリースディストリビューションの場合、ブループリントは次の **mountpoints** とそのサブディレクトリーをサポートします。

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

RHEL 9.4 および 8.10 以降のリリースディストリビューションでは、オペレーティングシステム用に予約されている特定のパスを除き、任意のカスタムマウントポイントを指定できます。

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

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

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

注記

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

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

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

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

警告

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

5.3.12.1. カスタマイズされたファイルをブループリントで指定する

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

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

以下のファイルは作成または置き換えることはできません。

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

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

注記

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

警告

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

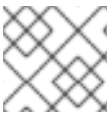
5.3.12.2. カスタマイズされたディレクトリーをブループリントで指定する

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

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

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

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



注記

以下のファイルは作成または置き換えることはできません。

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

以下のカスタマイズが可能です。

- ブループリントのファイルシステム設定をカスタマイズします。

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

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

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

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

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

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

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

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

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

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

ブループリントの各エントリーについて説明します。

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

  - **mode** - オプション - ディレクトリーのアクセスパーミッションを 8 進数形式で設定します。パーミッションを指定しない場合、デフォルトで 0755 に設定されます。先頭のゼロは任意です。

  - **user** - オプション - ユーザーをディレクトリーの所有者として設定します。ユーザーを指定しない場合は、デフォルトで **root** に設定されます。ユーザーは文字列または整数として指定できます。

- **group** - オプション - グループをディレクトリーの所有者として設定します。グループを指定しない場合は、デフォルトで **root** になります。グループは文字列または整数として指定できます。

- **ensure_parents** - オプション - 必要に応じて親ディレクトリーを作成するかどうかを指定します。値を指定しない場合は、デフォルトで **false** に設定されます。

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

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

ブループリントの各エントリーについて説明します。

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

- **mode** - オプション - ファイルのアクセスパーミッションを 8 進数形式で設定します。パーミッションを指定しない場合、デフォルトで 0644 に設定されます。先頭のゼロは任意です。

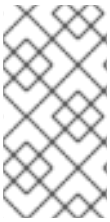
- **user** - オプション - ユーザーをファイルの所有者として設定します。ユーザーを指定しない場合は、デフォルトで **root** に設定されます。ユーザーは文字列または整数として指定できます。

- **group** - オプション - グループをファイルの所有者として設定します。グループを指定しない場合は、デフォルトで **root** になります。グループは文字列または整数として指定できます。

- **data** - オプション - プレーンテキストファイルの内容を指定します。コンテンツを指定しない場合は、空のファイルが作成されます。

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

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



注記

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

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

表5.1 イメージタイプの作成をサポートするデフォルトパッケージ

イメージタイプ	デフォルトパッケージ
ami	checkpolicy、chrony、cloud-init、cloud-utils-growpart、@Core、dhcp-client、gdisk、insights-client、kernel、langpacks-en、net-tools、NetworkManager、redhat-release、redhat-release-eula、rng-tools、rsync、selinux-policy-targeted、tar、yum-utils
openstack	@core、langpacks-en
qcow2	@core、chrony、dnf、kernel、dnf、nfs-utils、dnf-util、cloud-init、python3-jsonschema、qemu-guest-agent、cloud-utils-growpart、dracut-norescue、tar、tcpdump、rsync、dnf-plugin-spacewalk、rhn-client-tools、rhnlib、rhnsd、rhn-setup、NetworkManager、dhcp-client、cockpit-ws、cockpit-system、subscription-manager-cockpit、redhat-release、redhat-release-eula、rng-tools、insights-client
tar	policycoreutils、selinux-policy-targeted
vhd	@core、langpacks-en
vmdk	@core、chrony、cloud-init、firewalld、langpacks-en、open-vm-tools、selinux-policy-targeted

イメージタイプ	デフォルトパッケージ
edge-commit	attr、audit、basesystem、bash、bash-completion、chrony、clevis、clevis-dracut、clevis-luks、container-selinux、coreutils、criu、cryptsetup、curl、dnsmasq、dosfstools、dracut-config-generic、dracut-network、e2fsprogs、firewalld、fuse-overlayfs、fwupd、glibc、glibc-minimal-langpack、gnupg2、greenboot、gzip、hostname、ima-evm-utils、iproute、iptables、iputils、keyutils、less、lvm2、NetworkManager、NetworkManager-wif、NetworkManager-wwan、nss-altfiles、openssh-clients、openssh-server、passwd、pinentry、platform-python、podman、policycoreutils、policycoreutils-python-utils、polkit、procps-ng、redhat-release、rootfiles、rpm、rpm-ostree、rsync、selinux-policy-targeted、setools-console、setup、shadow-utils、shadow-utils、skopeo、slirp4netns、sudo、systemd、tar、tmux、traceroute、usbguard、util-linux、vim-minimal、wpa_supplicant、xz
edge-container	dnf、dosfstools、e2fsprogs、glibc、lorax-templates-generic、lorax-templates-rhel、lvm2、policycoreutils、python36、python3-iniparse、qemu-img、selinux-policy-targeted、systemd、tar、xfsprogs、xz

イメージタイプ	デフォルトパッケージ
edge-installer	**aajohan-comfortaa-fonts、abattis-cantarell-fonts、alsa-firmware、alsa-tools-firmware、anaconda、anaconda-install-env-deps、anaconda-widgets、audit、bind-utils、bitmap-fangsongti-fonts、bzip2、cryptsetup、dbus-x11、dejavu-sans-fonts、dejavu-sans-mono-fonts、device-mapper-persistent-data、dnf、dump、ethtool、fcoe-utils、ftp、gdb-gdbserver、gdisk、gfs2-utils、glibc-all-langpacks、google-noto-sans-cjk-ttc-fonts、gsettings-desktop-schemas、hdparm、hexedit、initscripts、ipmitool、iwl3945-firmware、iwl4965-firmware、iwl6000g2a-firmware、iwl6000g2b-firmware、jomolhari-fonts、kacst-farsi-fonts、kacst-qurn-fonts、kbd、kbd-misc、kdump-anaconda-addon、khmeros-base-fonts、libblockdev-lvm-dbus、libertas-sd8686-firmware、libertas-sd8787-firmware、libertas-usb8388-firmware、libertas-usb8388-olpc-firmware、libibverbs、libreport-plugin-bugzilla、libreport-plugin-reportuploader、libreport-rhel-anaconda-bugzilla、librsvg2、linux-firmware、lklug-fonts、lldpad、lohit-assamese-fonts、lohit-bengali-fonts、lohit-devanagari-fonts、lohit-gujarati-fonts、lohit-gurmukhi-fonts、lohit-kannada-fonts、lohit-odia-fonts、lohit-tamil-fonts、lohit-telugu-fonts、lsof、madan-fonts、metacity、mtr、mt-st、net-tools、nmap-ncat、nm-connection-editor、nss-tools、openssh-server、oscap-anaconda-addon、pciutils、perl-interpreter、pigz、python3-pyatspi、rdma-core、redhat-release-eula、rpm-ostree、rsync、rsyslog、sg3_utils、sil-abyssinica-fonts、sil-padauk-fonts、sil-scheherazade-fonts、smartmontools、smc-meera-fonts、spice-vdagent、strace、system-storage-manager、thai-scalable-waree-fonts、tigervnc-server-minimal、tigervnc-server-module、udisks2、udisks2-iscsi、usbutils、vim-minimal、volume_key、wget、xfsdump、xorg-x11-drivers、xorg-x11-fonts-misc、xorg-x11-server-utils、xorg-x11-server-Xorg、xorg-x11-xauth**

イメージタイプ	デフォルトパッケージ
edge-simplified-installer	attr、basesystem、binutils、bsdtar、clevis-dracut、clevis-luks、cloud-utils-growpart、coreos-installer、coreos-installer-dracut、coreutils、device-mapper-multipath、dnsmasq、dosfstools、dracut-live、e2fsprogs、fcoe-utils、fdo-init、gzip、ima-evm-utils、iproute、iptables、iputils、iscsi-initiator-utils、keyutils、lldpad、lvm2、passwd、policycoreutils、policycoreutils-python-utils、procps-ng、rootfiles、setools-console、sudo、traceroute、util-linux
image-installer	anaconda-dracut、curl、dracut-config-generic、dracut-network、hostname、iwl100-firmware、iwl1000-firmware、iwl105-firmware、iwl135-firmware、iwl2000-firmware、iwl2030-firmware、iwl3160-firmware、iwl5000-firmware、iwl5150-firmware、iwl6000-firmware、iwl6050-firmware、iwl7260-firmware、kernel、less、nfs-utils、openssh-clients、ostree、plymouth、prefixdevname、rng-tools、rpcbind、selinux-policy-targeted、systemd、tar、xfsprogs、xz
edge-raw-image	dnf、dosfstools、e2fsprogs、glibc、lorax-templates-generic、lorax-templates-rhel、lvm2、policycoreutils、python36、python3-iniparse、qemu-img、selinux-policy-targeted、systemd、tar、xfsprogs、xz
gce	@core、langpacks-en、acpid、dhcp-client、dnf-automatic、net-tools、python3、rng-tools、tar、vim

関連情報

- RHEL Image Builder の説明

第6章 簡略化されたインストーラーイメージを構築して、RHEL FOR EDGE IMAGE イメージをプロビジョニングします

デバイスへの無人インストール用に最適化された RHEL for Edge Simplified Installer イメージをビルドし、そのイメージを RHEL for Edge イメージにプロビジョニングできます。

6.1. 簡略化されたインストーラーイメージのビルドおよびデプロイ

edge-simplified-installer イメージタイプを使用して、RHEL for Edge Simplified Installer イメージをビルドします。

RHEL for Edge Simplified Installer イメージをビルドするには、既存の **OSTree** コミットを提供します。作成されるイメージには、OSTree コミットがデプロイされた生のイメージが含まれます。Simplified インストーラーの ISO イメージを起動すると、ハードディスクまたは仮想マシンのブートイメージとして使用できる RHEL for Edge システムがプロビジョニングされます。Simplified Installer イメージの作成に使用したブループリントで指定したユーザー名とパスワードを使用して、デプロイされたシステムにログインできます。

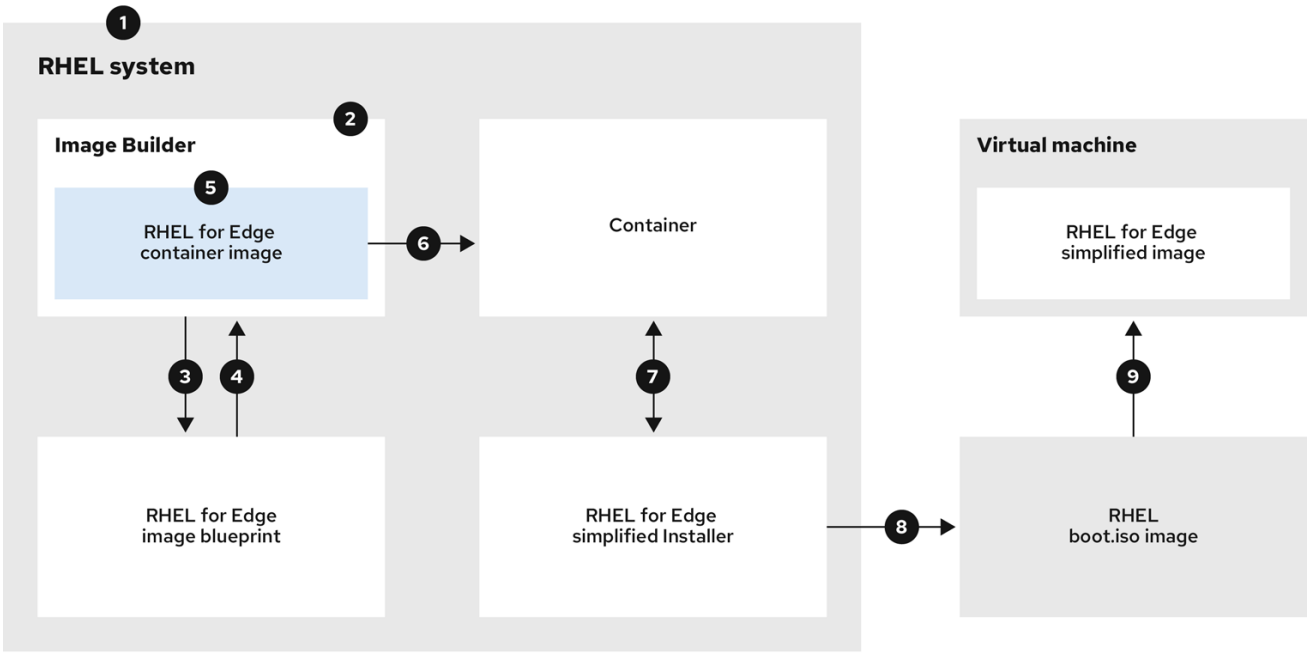
RHEL for Edge Simplified Installer イメージは、デバイスへの無人インストール用に最適化されており、ネットワークベースのデプロイメントと非ネットワークベースのデプロイメントの両方をサポートします。ただし、ネットワークベースのデプロイメントでは、UEFI HTTP ブートのみをサポートします。

簡略化された RHEL for Edge イメージを作成およびデプロイする手順の概要は次のとおりです。

1. RHEL システムをインストールおよび登録する
2. RHEL Image Builder をインストールする
3. RHEL Image Builder を使用して RHEL for Edge Container イメージのカスタムブループリントを作成する
4. RHEL Image Builder で RHEL for Edge のブループリントをインポートする
5. OSTreeリポジトリーとしてコミットをデプロイするための Web サーバーを用意した OCI コンテナーに、RHEL for Edge のイメージエンベッドを作成する
6. **Edge-simplified-installer** イメージのブループリントを作成する
7. 簡略化された RHEL for Edge を構築する
8. RHEL for Edge の簡略化されたイメージのダウンロード
9. **Edge-simplified-installer** virt-install を使用して RAW イメージをインストールする

次の図は、RHEL for Edge Simplified の構築およびプロビジョニングワークフローを表しています。

図6.1 ネットワークベース環境での RHEL for Edge の構築とプロビジョニング



6.2. RHEL IMAGE BUILDER CLI を使用した SIMPLIFIED イメージのブループリントの作成

簡略化された RHEL for Edge イメージのブループリントを作成するには、デバイスへの無人インストールを可能にするための **device file** の場所と、最初のデバイスクレデンシャルエクスチェンジを実行するための **URL** を使用してブループリントをカスタマイズする必要があります。また、ブループリントではユーザーとユーザーグループを指定する必要もあります。そのためには、以下の手順に従います。

手順

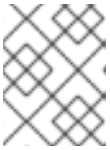
1. TOML (Tom's Obvious, Minimal Language) 形式で、以下のコンテンツのプレーンテキストファイルを作成します。

```
name = "simplified-installer-blueprint"
description = "blueprint for the simplified installer image"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[customizations]
installation_device = "/dev/vda"

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

```
[customizations.fdo]
manufacturing_server_url = "http://10.0.0.2:8080"
diun_pub_key_insecure = "true"
```



注記

ブループリントでの FDO のカスタマイズはオプションであり、エラーなしで RHEL for Edge Simplified Installer イメージをビルドできます。

- **name** はブループリントの名前で、**description** は説明です。
- **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。
- **モジュール** には、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します (例: パッケージ名 = "tmux"、対応するバージョンの glob = "2.9a")。現在、パッケージとモジュールには違いがないことに注意してください。
- **グループ** は、イメージにインストールされるパッケージグループです (例: グループパッケージ **anaconda-tools**)。モジュールとグループがわからない場合は、空のままにします。
- **installation-device** は、デバイスへの無人インストールを有効にするためのカスタマイズです。
- **Manufacturing_server_url** は、最初のデバイスクレデンシャルエクスチェンジを実行するための URL です。
- **name** は、イメージにログインするためのユーザー名です。
- **password** は、任意のパスワードです。
- **groups** は、任意のユーザーグループです ("widget" など)。

2. ブループリントを RHEL Image Builder サーバーにプッシュ (インポート) します。

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

3. 既存のブループリントを一覧表示して、作成したブループリントが正常にプッシュされて存在するかどうかを確認します。

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

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

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

関連情報

- RHEL Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

6.3. IMAGE BUILDER CLI を使用した RHEL FOR EDGE SIMPLIFIED INSTALLER イメージの作成

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge Simplified イメージを作成するには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge Simplified イメージのブループリントを作成している。
- イメージに埋め込むコミットの OSTree リポジトリーを提供している。たとえば、http://10.0.2.2:8080/repo です。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。

手順

1. 起動可能な ISO イメージを作成します。

```
# composer-cli compose start-ostree \
blueprint-name \
edge-simplified-installer \
--ref rhel/9/x86_64/edge \
--url URL-OSTree-repository \
```

詳細は以下のようになります。

- **blueprint-name** は RHEL for Edge のブループリント名です。
- **edge-simplified-installer** は image-type です。
- **--ref** は、コミットが作成される場所の参照です。
- **--url** は、イメージに埋め込むコミットの OSTree リポジトリーへの URL ですたとえば、http://10.0.2.2:8080/repo/ です。RHEL for Edge Container を起動するか、Web サーバーをセットアップすることができます。非ネットワークベースのデプロイメント用の RHEL for Edge Container イメージの作成 および RHEL for Edge イメージをインストールするための Web サーバーのセットアップ を参照してください。
  composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```

注記

イメージの作成プロセスは、完了するまでに最大 10 分かかる場合があります。

イメージ作成プロセスを中断するには、以下を実行します。

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

-

既存イメージを削除するには、以下を実行します。

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

関連情報

- RHEL Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

6.4. IMAGE BUILDER コマンドラインインターフェイスを使用した簡略化された RHEL FOR EDGE イメージのダウンロード

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge イメージをダウンロードするには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge イメージを作成済みである。

手順

1. RHEL for Edge イメージの状態を確認します。

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

出力には、以下が表示される必要があります。

```
$ <UUID> FINISHED date blueprint-name blueprint-version image-type
```

2. イメージをダウンロードします。

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

RHEL Image Builder が、コマンドが実行されたカレントディレクトリーのパスに **.iso** ファイル形式のイメージをダウンロードします。

UUID 番号とイメージサイズは並んで表示されます。

```
$ <UUID>-simplified-installer.iso: size MB
```

その結果、RHEL for Edge Simplified Installer ISO イメージをダウンロードしました。これをブート ISO として直接使用して、RHEL for Edge システムをインストールできます。

6.5. RHEL IMAGE BUILDER GUI を使用した SIMPLIFIED イメージのブループリントの作成

RHEL for Edge Simplified Installer イメージを作成するには、ブループリントを作成し、次の内容でカスタマイズする必要があります。

- デバイスへの無人インストールを可能にするためのデバイスノードの場所。
- 最初のデバイスクレデンシャルエクスチェンジを実行するための URL。

- ユーザーまたはユーザーグループ。

注記

イメージに必要なその他のカスタマイズを追加することもできます。

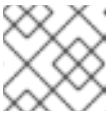
RHEL Image Builder GUI で簡略化された RHEL for Edge イメージのブループリントを作成するには、以下の手順を実行します。

前提条件

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

手順

1. RHEL Image Builder アプリケーションの右上隅にある Create Blueprint をクリックします。ブループリントの名前と説明のフィールドを含むダイアログウィザードが開きます。
2. **Details** ページで以下を行います。
   a. ブループリントの名前と、必要に応じてその説明を入力します。**Next** をクリックします。
3. オプション: Packages ページで、次の手順を実行します。
   a. **Available packages** の検索で、パッケージ名を入力し、**>** ボタンをクリックして、パッケージを Chosen packages フィールドに移動します。必要な数のパッケージを検索して含めます。**Next** をクリックします。



注記

特に指定がない限り、カスタマイズはすべてオプションです。

4. オプション: **Kernel** ページで、カーネル名とコマンドライン引数を入力します。
5. オプション: **File system** ページで、**Use automatic partitioning** を選択します。OSTree イメージには読み取り専用などの独自のマウントルールがあるため、ファイルシステムのカスタマイズは OSTree システムではサポートされません。**Next** をクリックします。
6. オプション: **Services** ページで、サービスを有効または無効にします。
   a. 有効または無効にするサービス名をコンマまたはスペースで区切るか、**Enter** キーを押して入力します。**Next** をクリックします。
7. オプション: **Firewall** ページで、ファイアウォールを設定します。
   a. **Ports** と、有効または無効にするファイアウォールサービスを入力します。
   b. **Add zone** ボタンをクリックして、各ゾーンのファイアウォールルールを個別に管理します。**Next** をクリックします。
8. **Users** ページで、以下の手順に従ってユーザーを追加します。
   a. **Add user** をクリックします。

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

注記

ブループリントのカスタマイズでユーザーを指定し、そのブループリントからイメージを作成すると、ブループリントはインストール時に **/usr/lib/passwd** ディレクトリーにユーザーを作成し、**/usr/etc/shadow** にパスワードを作成します。ブループリント用に作成したユーザー名とパスワードを使用してデバイスにログインできます。システムにアクセスしたら、**useradd** コマンドなどを使用してユーザーを作成する必要があります。

Next をクリックします。

9. オプション: **Groups** ページで、次の手順を実行してグループを追加します。

   a. **Add groups** ボタンをクリックします。

      i. **Group name** と **Group ID** を入力します。グループをさらに追加できます。**Next** をクリックします。

10. オプション: **SSH keys** ページで、キーを追加します。

    a. **Add key** ボタンをクリックします。

       i. SSH キーを入力します。

       ii. **User** を入力します。**Next** をクリックします。

11. オプション: **Timezone** ページで、タイムゾーンを設定します。

    a. **Timezone** フィールドに、システムイメージに追加するタイムゾーンを入力します。たとえば、次のタイムゾーン形式を追加します。"US/Eastern"。
    タイムゾーンを設定しない場合、システムはデフォルトとして協定世界時 (UTC) を使用します。

    b. **NTP** サーバーを入力します。**Next** をクリックします。

12. オプション: **Locale** ページで、以下の手順を実行します。

    a. **Keyboard** 検索フィールドに、システムイメージに追加するパッケージ名を入力します。たとえば、["en_US.UTF-8"] と入力します。

    b. **Languages** 検索フィールドに、システムイメージに追加するパッケージ名を入力します。たとえば、"us" と入力します。**Next** をクリックします。

13. 必須:**Others** ページで、次の手順を実行します。

    a. **Hostname** フィールドに、システムイメージに追加するホスト名を入力します。ホスト名を追加しない場合、オペレーティングシステムによってホスト名が決定されます。

    b. 必須:**Installation Devices** フィールドに、システムイメージの有効なノードを入力して、デバイスへの無人インストールを有効にします。たとえば、**dev/sda1** と入力します。**Next** をクリックします。

14. オプション: **FIDO device onboarding** ページで、次の手順を実行します。

    a. **Manufacturing server URL** フィールドに、最初のデバイスクレデンシャルエクスチェンジ

を実行するための **manufacturing server URL** を入力します (例: "http://10.0.0.2:8080")。ブループリントでの FDO のカスタマイズはオプションであり、エラーなしで RHEL for Edge Simplified Installer イメージをビルドできます。

b. **DIUN public key insecure** フィールドに、最初のデバイスクレデンシャルエクスチェンジを実行するための証明書公開鍵ハッシュを入力します。このフィールドは値として "true" を受け入れます。この値は、この接続が製造サーバーへの非セキュアな接続であることを意味します。たとえば、**manufacturing_server_url="http://${FDO_SERVER}:8080" diun_pub_key_insecure="true"** と入力します。"key insecure"、"key hash"、および "key root certs" の 3 つのオプションのうち 1 つだけを使用する必要があります。

c. **DIUN public key hash** フィールドに、公開鍵のハッシュバージョンを入力します。たとえば、**17BD05952222C421D6F1BB1256E0C925310CED4CE1C4FFD6E5CB968F4B73BF73** と入力します。鍵ハッシュは、製造サーバーの証明書に基づいて生成することで取得できます。鍵ハッシュを生成するには、次のコマンドを実行します。

```
# openssl x509 -fingerprint -sha256 -noout -in /etc/fdo/aio/keys/diun_cert.pem | cut -d"=" -f2 | sed 's/://g'
```

/etc/fdo/aio/keys/diun_cert.pem は、製造サーバーに保存されている証明書です。

d. **DIUN public key root certs** フィールドに、公開鍵ルート証明書を入力します。このフィールドは、製造サーバーに保存されている証明書ファイルの内容を受け入れます。証明書ファイルの内容を取得するには、次のコマンドを実行します。

```
$ cat /etc/fdo/aio/keys/diun_cert.pem.
```

15. **Next** をクリックします。

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

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

6.6. IMAGE BUILDER GUI を使用した RHEL FOR EDGE SIMPLIFIED INSTALLER イメージの RHEL の作成

RHEL Image Builder GUI を使用して RHEL for Edge Simplified イメージを作成するには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- RHEL for Edge Simplified イメージのブループリントを作成している。
- イメージに埋め込むコミットの OSTree リポジトリー (例: **http://10.0.2.2:8080/repo**) を提供した。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。
- FDO 製造サーバーが稼働している。

手順

1. Image Builder ダッシュボードにアクセスします。

2. ブループリントテーブルで、イメージを構築するブループリントを見つけます。
3. **Images** タブに移動し、**Create Image** をクリックします。**Create image** ウィザードが開きます。
4. **Image output** ページで、次の手順を実行します。
    a. **Select a blueprint** リストから、RHEL for Edge Simplified イメージ用に作成したブループリントを選択します。
    b. **Image output type** リストから、**RHEL for Edge Simplified Installer (.iso)** を選択します。
    c. **Image Size** フィールドにイメージサイズを入力します。Simplified Installer イメージに必要な最小イメージサイズは、次のとおりです。
5. **Next** をクリックします。
6. **OSTree settings** ページで、次の手順を実行します。
    a. **Repository URL** フィールドに、親 OSTree コミットのプル元となるリポジトリー URL を入力します。
    b. **Ref** フィールドに **ref** ブランチ名のパスを入力します。**ref** を入力しない場合は、ディストリビューションのデフォルトの **ref** が使用されます。
7. **Review** ページでイメージのカスタマイズを確認し、**Create** をクリックします。

イメージのビルドが開始され、完了するまでに最大 20 分かかります。ビルドを停止するには、**Stop build** をクリックします。

6.7. IMAGE BUILDER GUI を使用した簡略化された RHEL FOR EDGE イメージのダウンロード

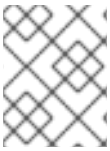
RHEL Image Builder GUI を使用して RHEL for Edge イメージをダウンロードするには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge イメージが正常に作成されている。リンクを参照してください。

手順

1. RHEL Image Builder ダッシュボードにアクセスします。ブループリントリストのダッシュボードが開きます。
2. ブループリントテーブルで、RHEL for Edge Simplified Installer イメージを構築したブループリントを見つけます。
3. **Images** タブに移動します。
4. 次のいずれかを実行します。
    - イメージをダウンロードします。
    - イメージのログをダウンロードして要素を検査し、問題がないかどうかを確認します。



注記

ブート ISO として直接ダウンロードした RHEL for Edge Simplified Installer ISO イメージを使用して、RHEL for Edge システムをインストールできます。

6.8. UEFI HTTP BOOT サーバーのセットアップ

UEFI HTTP Boot サーバーを設定し、この **UEFI HTTP Boot** サーバーに接続してネットワーク経由で RHEL for Edge 仮想マシンのプロビジョニングを開始できるようにするには、以下の手順に従います。

前提条件

- ISO 簡略化インストーラーイメージを作成している。
- ISO コンテンツを提供する http サーバー。

手順

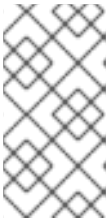
1. 選択したディレクトリーに ISO イメージをマウントします。

```
# mkdir /mnt/rhel9-install/
# mount -o loop,ro -t iso9660 /path_directory/installer.iso /mnt/rhel9-install/
```

/path_directory/installer.iso を RHEL for Edge 起動可能な ISO イメージへのパスに置き換えます。

2. マウントされたイメージから HTTP サーバーのルートにファイルをコピーします。このコマンドにより、イメージに含まれるファイルが保存される **/var/www/html/rhel9-install/** ディレクトリーを作成します。

```
# mkdir /var/www/html/httpboot/
# cp -R /mnt/rhel9-install/* /var/www/html/httpboot/
# chmod -R +r /var/www/html/httpboot/*
```



注記

一部のコピー方法は、有効なインストールソースに必要な **.treeinfo** ファイルを省略できることに注意してください。この手順で示されているように、ディレクトリー全体に対して **cp** コマンドを実行しても、**.treeinfo** が正しくコピーされます。

3. 以下を置き換えて、**/var/www/html/EFI/BOOT/grub.cfg** ファイルを更新します。
   a. **coreos.inst.install_dev=/dev/sda** を **coreos.inst.install_dev=/dev/vda** へ
   b. **linux /images/pxeboot/vmlinuz** を **linuxefi /images/pxeboot/vmlinuz** へ
   c. **initrd /images/pxeboot/initrd.img** を **initrdefi /images/pxeboot/initrd.img** へ
   d. **coreos.inst.image_file=/run/media/iso/disk.img.xz** を **coreos.inst.image_url=http://{IP-ADDRESS}/disk.img.xz** へ
      IP-ADDRESS は、このマシンの IP アドレスで、http ブートサーバーとして機能します。
4. httpd サービスを起動します。

```
# systemctl start httpd.service
```

その結果、**UEFI HTTP Boot** サーバーをセットアップした後、**UEFI HTTP** ブートを使用して RHEL for Edge デバイスをインストールできます。

6.9. 簡略化された ISO イメージを仮想マシンにデプロイ

次のインストールソースを使用して RHEL for Edge の Simplified イメージを作成することにより、生成した RHEL for Edge ISO イメージをデプロイします。

- **UEFI HTTP Boot**
- **virt-install**

この例は、**ネットワークベース** のインストール用に ISO イメージから **virt-install** インストールソースを作成する方法を示しています。

前提条件

- ISO イメージを作成している。
- UEFI HTTP ブートをサポートするようにネットワーク設定をセットアップしている。

手順

1. **UEFI HTTP** ブートをサポートするようにネットワーク設定をセットアップします。libvirt を使用した UEFI HTTP ブートのセットアップ を参照してください。
2. **virt-install** コマンドを使用して、UEFI HTTP Boot から RHEL for Edge Virtual Machine を作成します。

```
# virt-install \
    --name edge-install-image \
    --disk path=”  “, ,format=qcow2
    --ram 3072 \
    --memory 4096 \
    --vcpus 2 \
    --network network=integration,mac=mac_address \
    --os-type linux
    --os-variant rhel9 \
    --cdrom "/var/lib/libvirt/images/”ISO_FILENAME"
    --boot
uefi,loader_ro=yes,loader_type=pflash,nvram_template=/usr/share/edk2/ovmf/OVMF_VARS.fd,
loader_secure=no
    --virt-type kvm \
    --graphics none \
     --wait=-1
     --noreboot
```

コマンドを実行すると、仮想マシンのインストールが開始します。

検証

- 作成した仮想マシンにログインします。

6.10. USB フラッシュドライブからの SIMPLIFIED ISO イメージのデプロイ

USB インストール を使用して RHEL for Edge Simplified イメージを作成して生成した RHEL for Edge ISO イメージをデプロイします。

この例は、ISO イメージから **USB インストール** ソースを作成する方法を示しています。

前提条件

- ISO イメージである簡略化されたインストーラーイメージを作成しました。
- 8 GB の USB フラッシュドライブがある。

手順

1. ISO イメージファイルを USB フラッシュドライブにコピーします。
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。起動メニューには、次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
   ```

4. Red Hat Enterprise Linux 9 のインストールを選択します。これにより、システムのインストールが開始します。

関連情報

- インストールの起動

6.11. FIPS モードでの RHEL FOR EDGE イメージの作成および起動

FIPS 対応の RHEL for Edge イメージを作成して起動できます。結果のイメージで FIPS モードを有効にするかどうかをブループリントで指定できます。デフォルト値は **false** です。

以下のイメージタイプを FIPS モードでビルドできます。

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

重要

FIPS モードを有効にできるのは、イメージのプロビジョニングプロセス中のみです。非 FIPS イメージのビルドを開始した後に FIPS モードに変更することはできません。FIPS 対応イメージをビルドするホストが FIPS が有効になっていない場合は、このホストによって生成されたキーは FIPS に準拠しませんが、作成されるイメージは FIPS に準拠します。

前提条件

- RHEL for Edge Container OSTree コミットを作成してダウンロードしました。
- システムに Podman をインストール済みである。How to install Podman in RHEL を参照してください。

手順

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

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

[ostree]
ref= "example/edge"
url= "http://example.com/repo"

[customizations]
installation_device = "/dev/vda"
fips = true

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

[customizations.fdo]
manufacturing_server_url = "https://fdo.example.com"
diun_pub_key_insecure = true
```

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

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

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

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

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

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

5. イメージに埋め込むコミットの OSTree リポジトリー (例: http://10.0.2.2:8080/repo) を提供した。詳細は、UEFI HTTP Boot サーバーのセットアップ を参照してください。

6. 起動可能な ISO イメージを作成します。

```
# composer-cli compose start-ostree \
blueprint-name \
edge-simplified-installer \
--ref rhel/8/x86_64/edge \
--url URL-OSTree-repository \
```

7. RHEL for Edge イメージの状態を確認します。

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

8. イメージをダウンロードします。

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

RHEL Image Builder が、コマンドが実行されたカレントディレクトリーのパスに **.iso** ファイル形式のイメージをダウンロードします。UUID 番号とイメージサイズは並んで表示されます。

```
$ <UUID>-simplified-installer.iso: size MB
```

9. UEFI HTTP Boot から RHEL for Edge 仮想マシンを作成します。次に例を示します。

```
# virt-install \
    --name edge-device
    --disk path="/var/lib/libvirt/images/edge-device.qcow2",size=5,format=qcow2 \
    --memory 4096 \
    --vcpus 2 \
    --network network=default \
    --os-type linux \
    --os-variant rhel8.9 \
    --cdrom /var/lib/libvirt/images/<UUID>-simplified-installer.iso \
    --boot uefi,loader.secure=false \
    --virt-type kvm \
    --graphics none \
    --wait=-1 \
    --noreboot
```

コマンドを実行すると、仮想マシンのインストールが開始します。

検証

1. ブループリントで設定したユーザー名とパスワードを使用して、作成した仮想マシンにログインします。

2. FIPS 設定が有効になっているかどうかを確認します。

```
$ fips-mode-setup –check
```

```
…
FIPS mode is enabled
…
```

第7章 最小 RAW イメージのビルドとプロビジョニング

minimum-raw イメージは、**xz** 形式で圧縮され、事前にパッケージ化された起動可能な最小限の RPM イメージです。RHEL イメージビルダーを使用して RHEL for Edge Minimal Raw イメージタイプを構築し、その Minimal Raw イメージを **aarch64** および **x86** アーキテクチャーにデプロイできます。

7.1. 最小 RAW イメージのビルドとデプロイ

RHEL for Edge Minimal Raw イメージは、**minimal-raw** イメージタイプを使用してビルドします。イメージをブートするには、イメージを解凍し、SD カードなどのブート可能なデバイスにコピーする必要があります。RHEL for Edge Minimal Raw イメージの作成に使用したブループリントで指定したユーザー名とパスワードを使用して、デプロイされたシステムにログインできます。

RHEL for Edge Minimal Raw イメージを作成およびデプロイする手順の概要は次のとおりです。

1. RHEL システムをインストールおよび登録する
2. RHEL Image Builder をインストールする
3. RHEL Image Builder を使用して RHEL for Edge Minimal Raw イメージのカスタムブループリントを作成する
4. RHEL Image Builder で RHEL for Edge のブループリントをインポートする
5. RHEL for Edge Minimal Raw イメージを作成する
6. RHEL for Edge Minimal Raw イメージを展開する
7. RHEL for Edge Minimal Raw イメージをデプロイする

7.2. RHEL イメージビルダー CLI を使用して MINIMAL RAW イメージのブループリントの作成

RHEL イメージビルダーを使用してブループリントを作成し、ユーザー名とパスワードでカスタマイズします。そうすることで、Minimal Raw イメージを作成し、ブループリントでカスタマイズした認証情報を使用してログインできるようになります。

手順

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

```
name = "minimal-raw-blueprint"
description = "blueprint for the Minimal Raw image"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

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

- name はブループリントの名前、description はブループリントの説明です。
- 0.0.1 は、セマンティックバージョニングスキームに基づくバージョン番号です。
- モジュールには、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します (例: パッケージ名 = "tmux"、対応するバージョンの glob = "2.9a")。現在、パッケージとモジュールには違いがないことに注意してください。
- グループは、イメージにインストールするパッケージグループです (例: anaconda-tools グループパッケージ)。モジュールとグループがわからない場合は、空のままにします。
- name は、イメージにログインするためのユーザー名です。
- パスワードは、任意のパスワードです。
- グループは、"widget"などの任意のユーザーグループです。

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

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

3. ブループリントがシステムで利用できるかどうかを確認します。

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

4. ブループリント内のコンポーネント、バージョン、およびそれらの依存関係の有効性を確認します。

```
# composer-cli blueprints depsolve <blueprint_name>
```

関連情報

- RHEL Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

7.3. RHEL イメージビルダー CLI を使用して MINIMAL RAW イメージの作成

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge Minimal Raw イメージを作成するには、以下の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL for Edge Minimal Raw イメージのブループリントを作成している。

手順

1. 起動可能なイメージを作成します。

```
# composer-cli compose start <_blueprint_name_> minimal-raw
```

ここでは、以下のようになります。

- **<blueprint_name>** は、RHEL for Edge ブループリント名です。

- **minimal-raw-** はイメージタイプです。
  composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
# <UUID> RUNNING date <blueprint_name> blueprint-version minimal-raw
```

関連情報

- Image Builder コマンドラインを使用した RHEL for Edge イメージの作成

7.4. MINIMAL RAW イメージのダウンロードと解凍

RHEL Image Builder コマンドラインインターフェイスを使用して RHEL for Edge Minimal Raw イメージをダウンロードし、イメージを解凍して起動できるようにします。

前提条件

- RHEL for Edge Minimal Raw イメージを作成している。

手順

1. RHEL for Edge Minimal Raw イメージの作成ステータスを確認します。

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

出力に次の詳細が表示される必要があります。

```
$ <UUID>_ FINISHED date <blueprint_name> <blueprint_version> minimal-raw
```

2. イメージをダウンロードします。

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

Image Builder が、**.raw.xz** 形式のイメージを作業ディレクトリーにダウンロードします。UUID 番号とイメージサイズは並んで表示されます。

```
$ <UUID> minimal_raw.img.xz: size MB
```

3. イメージを解凍します。

```
$ xz -d <UUID>_minimal-raw.img.xz
```

イメージを展開した後、起動可能な RHEL for Edge Minimal Raw イメージを使用して RHEL for Edge システムをインストールします。

7.5. USB フラッシュドライブからの MINIMAL RAW イメージのデプロイ

USB インストールソースを作成して生成した RHEL for Edge Minimal Raw イメージをデプロイします。この例では、RHEL for Edge Minimal Raw イメージから USB インストールソースを作成する方法を示します。

前提条件

- RHEL for Edge Minimal Raw イメージを作成している。
- 8 GB の USB フラッシュドライブがある。
- UEFI HTTP サーバーが実行されている。UEFI HTTP ブートサーバーのセットアップ を参照してください。

手順

1. RHEL for Edge Minimal Raw イメージファイルを USB フラッシュドライブにコピーします。
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 RHEL for Edge Minimal Raw イメージを起動します。ブートメニューに次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
```

4. **Install Red Hat Enterprise Linux 9**を選択します。これにより、システムのインストールが開始します。

検証

1. ブループリントで設定したユーザー名とパスワードを使用してイメージを起動します。
   a. リリースを確認します。

```
$ cat /etc/os-release
```

   b. システム内のブロックデバイスをリスト表示します。

```
$ lsblk
```

第8章 RHEL FOR EDGE SIMPLIFIED INSTALLER イメージ用の IGNITION ツールの使用

RHEL for Edge は、Ignition ツールを使用して、ブートプロセスの初期段階でユーザー設定をイメージに注入します。Ignition ツールが注入するユーザー設定には次のものが含まれます。

- ユーザー設定。
- 通常のファイルや **systemd** ユニットなどのファイルの書き込み。

最初の起動時に、Ignition はリモート URL または Simplified Installer ISO に埋め込まれたファイルから設定を読み取ります。その後、Ignition はその設定をイメージに適用します。

8.1. IGNITION 設定ファイルの作成

Butane ツールは、Ignition 設定ファイルを作成するのに推奨されるオプションです。**Butane** は **Butane Config YAML** ファイルを使用して、JSON 形式で **Ignition Config** を生成します。JSON ファイルは、システムの最初の起動時に使用されます。**Ignition Config** は、ユーザーの作成や **systemd** ユニットのインストールなど、イメージ内の設定を適用します。

前提条件

- Butane ツールのバージョン v0.17.0 がインストールされている。

```
$ sudo dnf/yum install -y butane
```

手順

1. **Butane Config** ファイルを作成し、**.bu** 形式で保存します。RHEL for Edge イメージの場合は、**variant** エントリーを **r4e** として指定し、**version** エントリーを **1.0.0** として指定する必要があります。バージョン 1.0.0 の Butane **r4e** バリアントは、Ignition 仕様バージョン **3.3.0** をターゲットとしています。以下は Butane Config YAML ファイルの例です。

```
variant: r4e
version: 1.0.0
ignition:
  config:
    merge:
      - source: http://192.168.122.1:8000/sample.ign
passwd:
  users:
    - name: core
      groups:
        - wheel
      password_hash: password_hash_here
      ssh_authorized_keys:
        - ssh-ed25519 some-ssh-key-here
storage:
  files:
    - path: /etc/NetworkManager/system-connections/enp1s0.nmconnection
      contents:
        inline: |
          [connection]
          id=enp1s0
```

```
          type=ethernet
          interface-name=enp1s0
          [ipv4]
          address1=192.168.122.42/24,192.168.122.1
          dns=8.8.8.8;
          dns-search=
          may-fail=false
          method=manual
      mode: 0600
    - path: /usr/local/bin/startup.sh
      contents:
        inline: |
          #!/bin/bash
          echo "Hello, World!"
      mode: 0755
systemd:
  units:
    - name: hello.service
      contents: |
        [Unit]
        Description=A hello world
        [Install]
        WantedBy=multi-user.target
      enabled: true
    - name: fdo-client-linuxapp.service
      dropins:
        - name: log_trace.conf
          contents: |
            [Service]
            Environment=LOG_LEVEL=trace
```

2. 次のコマンドを実行して、**Butane Config YAML** ファイルを使用し、JSON 形式で Ignition Config を生成します。

```
$ ./path/butane example.bu
{"ignition":{"config":{"merge":[{"source":"http://192.168.122.1:8000/sample.ign"}]},"timeouts":
{"httpTotal":30},"version":"3.3.0"},"passwd":{"users":[{"groups":
["wheel"],"name":"core","passwordHash":"password_hash_here","sshAuthorizedKeys":["ssh-
ed25519 some-ssh-key-here"]}]},"storage":{"files":[{"path":"/etc/NetworkManager/system-
connections/enp1s0.nmconnection","contents":
{"compression":"gzip","source":"data:;base64,H4sIAAAAAAAAC/0yKUcrCMBAG3/csf/ObUKQie5
LShyX5SgPNNiSr0NuLgiDzNMPM8VBFtHzoQjkxtPp+ITsrGLahKYyyGtoqEYNKwfeZc32OC0l
KDb179rfg/HVyPgQ3hv8w/v0WT0k7T+7D/S1Dh7S4MRU5h1XyzqvsHVRg25G4iD5kp1cAAAD
//6Cvq2ihAAAA"},"mode":384},{"path":"/usr/local/bin/startup.sh","contents":
{"source":"data:;base64,IyEvYmluL2Jhc2gKZWNobyAiSGVsbG8sIFdvcmxkISIK"},"mode":493}]
},"systemd":{"units":[{"contents":"[Unit]\nDescription=A hello
world\n[Install]\nWantedBy=multi-user.target","enabled":true,"name":"hello.service"},
{"dropins":[{"contents":"
[Service]\nEnvironment=LOG_LEVEL=trace\n","name":"log_trace.conf"}],"name":"fdo-client-
linuxapp.service"}]}}
```

Butane Config YAML ファイルを実行して **Ignition Config JSON** ファイルをチェックおよび生成すると、サポートされていないフィールド (パーティションなど) を使用している場合に警告が表示されることがあります。これらのフィールドを修正して、チェックを再実行できます。

これで、ブループリントのカスタマイズに使用できる Ignition JSON 設定ファイルが完成しました。

関連情報

- RHEL for Edge Specification v1.0.0。

8.2. GUI での IGNITION をサポートするブループリントの作成

Simplified Installer イメージを構築する際に、ブループリントの Ignition ページに次の詳細を入力してブループリントをカスタマイズできます。

- **Firstboot URL** - 最初の起動時に取得される Ignition 設定を参照する URL を入力する必要があります。これは、RAW イメージと Simplified Installer イメージの両方に使用できます。
- **Embedded Data** - **Base64** でエンコードされた **Ignition Configuration** ファイルを指定する必要があります。Simplified Installer イメージにのみ使用できます。

Ignition ブループリントのカスタマイズを使用して、Ignition 設定をサポートする簡素化された RHEL for Edge イメージのブループリントをカスタマイズするには、次の手順に従います。

前提条件

- ブラウザーの Web コンソールから Image Builder アプリケーションを開いた。RHEL Web コンソールでの Image Builder GUI へのアクセス を参照してください。
- **coreos-installer-dracut** が、現在の OSBuild ファイルの存在に基づいて **-ignition-url|-ignition-file** を定義できる (埋め込みセクションを完全にサポートする場合)。

手順

1. 右上隅にある **Create Blueprint** をクリックします。
ブループリントの名前と説明のフィールドを含むダイアログウィザードが開きます。
   - **Details** ページで以下を行います。
     - ブループリントの名前と、必要に応じてその説明を入力します。**Next** をクリックします。
   - **Ignition** ページで、次の手順を実行します。
     - **Firstboot URL** フィールドに、最初の起動時に取得される Ignition 設定を参照する URL を入力します。
     - **Embedded Data** フィールドに、**base64** でエンコードされた **Ignition 設定** ファイルをドラッグまたはアップロードします。**Next** をクリックします。
   - イメージの詳細を確認し、**Create** をクリックします。

Image Builder ダッシュボードビューが開き、既存のブループリントが一覧表示されます。

次のステップ

- 作成したブループリントを使用して、Simplified Installer イメージを構築できます。Image Builder CLI を使用した RHEL for Edge Simplified Installer イメージの RHEL の作成 を参照してください。

8.3. CLI を使用した IGNITION をサポートするブループリントの作成

Simplified Installer イメージを構築する際に、**customizations.ignition** セクションをブループリントに追加してカスタマイズできます。これにより、ベアメタルプラットフォームに使用できる Simplified Installer イメージまたは RAW イメージを作成できます。ブループリントの **customizations.ignition** のカスタマイズを使用すると、設定ファイルを **edge-simplified-installer** ISO および **edge-raw-image** イメージで使用できるようになります。

- **edge-simplified-installer** ISO イメージの場合、ISO イメージに含める Ignition 設定ファイルを埋め込むようにブループリントをカスタマイズできます。以下に例を示します。

  ```
  [customizations.ignition.embedded]
  config = "eyJ --- BASE64 STRING TRIMMED --- 19fQo="
  ```

  Base64 でエンコードされた Ignition 設定ファイルを指定する必要があります。

- **edge-simplified-installer** ISO イメージと **Edge-raw-image** の両方について、最初の起動時に Ignition 設定を取得するためにフェッチされる URL を定義することで、ブループリントをカスタマイズできます。以下に例を示します。

  ```
  [customizations.ignition.firstboot]
  url = "http://your_server/ignition_configuration.ig"
  ```

  最初の起動時に取得される Ignition 設定を参照する URL を入力する必要があります。

Ignition 設定をサポートする Simplified RHEL for Edge イメージのブループリントをカスタマイズするには、次の手順に従います。

前提条件

- Ignition 設定ファイルが作成されている (**[customizations.ignition.embedded]** のカスタマイズを使用する場合)。
- 最初の起動時に取得される Ignition 設定を参照する URL を持つコンテナーが作成されている (**[customizations.ignition.firstboot]** のカスタマイズを使用する場合)。
- ブループリントのカスタマイズの **[customizations.ignition.embedded]** セクションにより、**coreos-installer-dracut** が、OSBuild のファイルの存在に基づいて **-ignition-url**|**-ignition-file** を定義できる。

手順

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

   ```
   name = "simplified-installer-blueprint"
   description = "Blueprint with Ignition for the simplified installer image"
   version = "0.0.1"
   packages = []
   modules = []
   groups = []
   distro = ""

   [customizations.ignition.embedded]
   config = "eyJ --- BASE64 STRING TRIMMED --- 19fQo="
   ```

ここでは、以下のようになります。

- **name** はブループリントの名前で、**description** は説明です。
- **VERSION** は、セマンティックバージョニングスキームに基づくバージョン番号です。
- **modules** と **packages** には、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します。たとえば、パッケージ **name = "tmux"** と、対応するバージョンの glob **version = "3.3a"** を記述します。現在、パッケージとモジュールには違いがないことに注意してください。
- **groups** は、イメージにインストールするパッケージグループです。たとえば、**groups = "anaconda-tools"** グループパッケージなどです。モジュールとグループがわからない場合は、空のままにします。

警告

Ignition を使用してユーザーを作成する場合、同時に FDO カスタマイズを使用してユーザーを作成することはできません。Ignition を使用してユーザーを作成し、FDO を使用して設定ファイルをコピーできます。しかし、ユーザーを作成する場合は、Ignition または FDO を使用して作成しますが、両方を同時に作成することはできません。

2. ブループリントを Image Builder サーバーにプッシュ (インポート) します。

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

3. 既存のブループリントを一覧表示して、作成したブループリントが正常にプッシュされて存在するかどうかを確認します。

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

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

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

次のステップ

- 作成したブループリントを使用して、Simplified Installer イメージを構築できます。Image Builder CLI を使用した RHEL for Edge Simplified Installer イメージの RHEL の作成 を参照してください。

関連情報

- RHEL for Edge Specification v1.0.0。

第9章 RHEL FOR EDGE の VMDK イメージの作成

RHEL Image Builder を使用して、RHEL for Edge の **.vmdk** イメージを作成できます。Ignition サポートを備えた **edge-vsphere** イメージタイプを作成して、ブートプロセスの初期段階でユーザー設定をイメージに注入できます。その後、イメージを vSphere にロードし、vSphere 仮想マシンでイメージをブートできます。このイメージは、ESXi 7.0 U2、ESXi 8.0 以降と互換性があります。vSphere 仮想マシンは、バージョン 19 および 20 と互換性があります。

9.1. IGNITION 設定を使用したブループリントの作成

.vmdk イメージのブループリントを作成し、そのブループリントを **customizations.ignition** セクションでカスタマイズします。これにより、イメージを作成し、起動時にオペレーティングシステムによってユーザー設定をイメージに注入することができます。

前提条件

- Ignition 設定ファイルを作成している。以下に例を示します。

```
{
  "ignition":{
    "version":"3.3.0"
  },
  "passwd":{
    "users":[
      {
        "groups":[
          "wheel"
        ],
        "name":"core",
        "passwordHash":"$6$jfuNnO9t1Bv7N"
      }
    ]
  }
}
```

手順

1. 以下の内容のブループリントを Tom's Obvious Minimal Language (TOML) 形式で作成します。

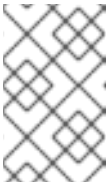
```
name = "vmdk-image"
description = "Blueprint with Ignition for the vmdk image"
version = "0.0.1"
packages = ["open-vm-tools"]
modules = []
groups = []
distro = ""

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

[customizations.ignition.firstboot]
url = http://<IP_address>:8080/config.ig
```

ここでは、以下のようになります。

- **name** はブループリントの名前で、**description** は説明です。
- **VERSION** は、セマンティックバージョニングスキームに基づくバージョン番号です。
- **modules** と **packages** には、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します。たとえば、パッケージ **name = "open-vm-tools"** などです。現在、パッケージとモジュールには違いがないことに注意してください。
- **groups** は、イメージにインストールするパッケージグループです。たとえば、**groups = "anaconda-tools"** グループパッケージなどです。モジュールとグループがわからない場合は、空のままにします。
- **customizations.user** は、仮想マシンにログインするためのユーザー名とパスワードを作成します。
- **customizations.ignition.firstboot** には、Ignition 設定ファイルが提供される URL を含めます。



注記

デフォルトでは、**open-vm-tools** パッケージは **edge-vsphere** イメージに含まれていません。このパッケージが必要な場合は、ブループリントのカスタマイズに含める必要があります。

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

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

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

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

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

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

次のステップ

- 作成したブループリントを使用して **.vmdk** イメージをビルドします。

9.2. RHEL FOR EDGE の VMDK イメージの作成

RHEL for Edge **.vmdk** イメージを作成するには、RHEL Image Builder コマンドラインインターフェイスで edge-vsphere イメージタイプを使用します。

前提条件

- **.vmdk** イメージのブループリントを作成している。

- イメージに埋め込むコミットの OSTree リポジトリーを提供している。たとえば、**http://10.0.2.2:8080/repo** です。詳細は、RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。

手順

1. **.vmdk** イメージの作成を開始します。

```
# composer-cli compose start start-ostree <blueprint-name> edge-vsphere --<url>
```

--<url> はリポジトリーの URL です (例: **http://10.88.0.1:8080/repo**)。

composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
$ <UUID> RUNNING date <blueprint-name>  <blueprint-version> edge-vsphere
```

3. 作成プロセスが完了したら、結果のイメージファイルをダウンロードします。

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

次のステップ

- **.vmdk** イメージを vSphere にアップロードします。

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

govc import.vmdk CLI ツールを使用して **.vmdk** イメージを VMware vSphere にアップロードし、仮想マシンでイメージを起動します。

前提条件

- RHEL Image Builder を使用して **.vmdk** イメージを作成し、それをホストシステムにダウンロードしている。
- CLI ツール **govc import.vmdk** をインストールしている。
- CLI ツールクライアント **govc import.vmdk** を設定している。
  - 環境で次の値を設定する必要があります。

```
GOVC_URL
GOVC_DATACENTER
GOVC_FOLDER
```

```
GOVC_DATASTORE
GOVC_RESOURCE_POOL
GOVC_NETWORK
```

手順

1. **.vmdk** イメージをダウンロードしたディレクトリーに移動します。
2. 次の手順を実行して、vSphere でイメージを起動します。
   a. **.vmdk** イメージを vSphere にインポートします。

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

   b. 電源をオンにせずに vSphere に仮想マシンを作成します。

```
govc vm.create \
-net="VM Network" -net.adapter=vmxnet3 \
-disk.controller=pvscsi -on=false \
-m=4096 -c=2 -g=rhel9_64Guest \
-firmware=efi vm_name govc vm.disk.attach \
-disk=”foldername/composer-api.vmdk” govc vm.power -on\
-vm vm_name -link=false \
 vm_name
```

   c. 仮想マシンの電源をオンにします。

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

   d. 仮想マシンの IP アドレスを取得します。

```
HOST=$(govc vm.ip vmname)
```

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

```
$ ssh admin@HOST
```

第10章 RHEL FOR EDGE AMI イメージの作成

RHEL Image Builder を使用して、カスタマイズした RHEL for Edge **edge-ami** イメージを作成できます。RHEL for Edge **edge-ami** は、ブートプロセスの初期段階でユーザー設定をイメージに注入するための Ignition サポートを備えています。その後、イメージを AWS クラウドにアップロードし、AWS で EC2 インスタンスを起動できます。AMI イメージタイプは、AMD または Intel 64 ビットアーキテクチャーで使用できます。

10.1. EDGE AMI イメージのブループリントの作成

edge-ami イメージのブループリントを作成し、そのブループリントを **customizations.ignition** セクションでカスタマイズします。これにより、イメージを作成し、イメージの起動時にユーザー設定を注入することができます。

前提条件

- Ignition 設定ファイルを作成している。以下に例を示します。

```
{
  "ignition":{
    "version":"3.3.0"
  },
  "passwd":{
    "users":[
      {
        "groups":[
          "wheel"
        ],
        "name":"core",
        "passwordHash":"$6$jfuNnO9t1Bv7N"
      }
    ]
  }
}
```

詳細は、Ignition 設定ファイルの作成 を参照してください。

手順

1. 以下の内容のブループリントを Tom's Obvious Minimal Language (TOML) 形式で作成します。

```
name = "ami-edge-image"
description = "Blueprint for Edge AMI image"
version = "0.0.1"
packages = ["cloud-init"]
modules = []
groups = []
distro = ""

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

```
[customizations.ignition.firstboot]
url = http://<IP_address>:8080/config.ig
```

ここでは、以下のようになります。

- **name** はブループリントの名前で、**description** は説明です。
- **VERSION** は、セマンティックバージョニングスキームに基づくバージョン番号です。
- **modules** と **packages** には、イメージにインストールするパッケージの名前と、それに対応するバージョンの glob を記述します。たとえば、パッケージ **name = "open-vm-tools"** などです。現在、パッケージとモジュールには違いがないことに注意してください。
- **groups** は、イメージにインストールするパッケージグループです。たとえば、**groups = "wheel"** です。モジュールとグループがわからない場合は、空のままにします。
- **customizations.user** は、仮想マシンにログインするためのユーザー名とパスワードを作成します。
- **customizations.ignition.firstboot** には、Ignition 設定ファイルが提供される URL を含めます。

注記

デフォルトでは、**open-vm-tools** パッケージは **edge-vsphere** イメージに含まれていません。このパッケージが必要な場合は、ブループリントのカスタマイズに含める必要があります。

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

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

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

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

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

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

次のステップ

- 作成したブループリントを使用して **edge-ami** イメージをビルドします。

10.2. RHEL FOR EDGE AMI イメージの作成

RHEL Image Builder のコマンドラインインターフェイスで RHEL for Edge **edge-ami** イメージを作成します。

前提条件

- **edge-ami** イメージのブループリントを作成しました。
- イメージに埋め込むコミットの OSTree リポジトリーを提供している。たとえば、**http://10.0.2.2:8080/repo** です。詳細は、RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。

手順

1. **edge-ami** イメージの作成を開始します。

```
# composer-cli compose start start-ostree <blueprint-name> edge-ami --<url>
```

--<url> はリポジトリーの URL です (例: **http://10.88.0.1:8080/repo**)。

composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

出力には、以下の形式で状態が表示されます。

```
$ <UUID> RUNNING date <blueprint-name>  <blueprint-version> edge-ami
```

3. 作成プロセスが完了したら、結果のイメージファイルをダウンロードします。

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

次のステップ

- **edge-ami** イメージの AWS へのアップロード

10.3. RHEL EDGE AMI イメージの AWS へのアップロード

CLI を使用して、**edge-ami** イメージを Amazon AWS クラウドサービスプロバイダーにアップロードします。

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。AWS バケットに 必要なロール を作成している。**aws-cli** ツールをインストールしている。

手順

1. **aws-cli** ツールを設定します。

```
$ aws configure
```

a. プロフィールを設定します。コマンドを実行し、アクセスキー ID クレデンシャル、シークレットアクセスキー、デフォルトのリージョン名、およびデフォルトの出力名を入力します。

```
$ aws configure --profile
```

2. 既存のバケットをリスト表示します。

```
$ aws s3 ls
```

3. イメージを S3 にアップロードします。

```
$ aws s3 cp <path_to_image/image> s3://<your_bucket_name>
```

4. S3 バケット内のイメージをリスト表示します。

```
$ aws s3 ls s3://<your_bucket_name>
```

5. **container-simple.json** ファイルを作成します。"URL" の内容を S3 バケットに置き換えます。たとえば、**s3://rhel-edge-ami-us-west-2/2ba3c125-cc58-4cc0-861a-4cc78e892df6-image.raw** などです。

```
{
  "Description": "RHEL for Edge image",
  "Format": "edge-ami",
  "Url": "s3://rhel-edge-ami-us-west-2/UUID-image.raw"
}
```

6. **edge.ami** イメージを EC2 スナップショットとして S3 バケットにインポートします。

注記

EC2 イメージは、S3 バケットを作成したリージョンと同じリージョンに存在する必要があります。

```
$ aws ec2 import-snapshot --description "RHEL edge" \
--disk-container file://container-simple.json --region us-west-2
```

次の **.json** はコマンド出力の例です。

```
{
  "Description": "RHEL for Edge image",
  "Format": "edge-ami",
  "Url": "s3://rhel-edge-ami-us-west-2/UUID-image.raw"
}
```

7. json の "ImportTaskId" 値をメモします。これはインポートステータスの確認に使用します。この例では、"ImportTaskId" は **import-snap-0f3055c4b7a454c85** です。

8. 前の手順の出力 json ファイルの "ImportTaskId" 値を使用して、スナップショットのインポートステータスを確認します。

```
$ aws ec2 describe-import-snapshot-tasks \
--import-task-ids import-snap-0f3055c4b7a454c85
{
    "ImportSnapshotTasks": [
        {
            "Description": "RHEL edge",
            "ImportTaskId": "import-snap-0f3055c4b7a454c85",
            "SnapshotTaskDetail": {
                "Description": "RHEL edge",
                "DiskImageSize": 10737418240.0,
                "Format": "RAW",
                "SnapshotId": "snap-001b267e752039eab",
                "Status": "completed",
                "Url": "s3://rhel-edge-ami-us-west-2/2ba3c125-cc58-4cc0-861a-4cc78e892df6-
image.raw",
                "UserBucket": {
                    "S3Bucket": "rhel-edge-ami-us-west-2",
                    "S3Key": "2ba3c125-cc58-4cc0-861a-4cc78e892df6-image.raw"
                }
            },
            "Tags": []
        }
    ]
}
```

"Status" が "completed" とマークされるまで、このコマンドを実行します。その後、EC2 にアクセスしてスナップショットから AMI イメージを作成し、起動します。

検証

イメージのアップロードが成功したことを確認するには、以下を行います。

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

2. ダッシュボードでイメージを選択し、Launch をクリックします。
新しいインスタンスを起動するときは、ブートモードとして UEFI を選択し、EC2 イメージ用に少なくとも 4 GB の RAM を選択する必要があります。

3. Ignition 設定で作成したユーザー名とパスワードを使用して、AWS の **edge-ami** にログインできます。

関連情報

- 仮想マシンのインポートに必要なサービスロール

第11章 FDO を使用した EDGE デバイスの RHEL の自動プロビジョニングとオンボーディング

RHEL for Edge Simplified Installer イメージをビルドし、それを RHEL for Edge イメージにプロビジョニングできます。FIDO Device Onboarding (FDO) プロセスは、Edge デバイスを自動的にプロビジョニングしてオンボーディングし、ネットワークに接続されている他のデバイスやシステムとデータを交換します。

重要

Red Hat は、テクノロジープレビュー機能として **FDO** プロセスを提供し、安全なネットワークで実行する必要があります。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

11.1. FIDO DEVICE ONBOARDING (FDO) プロセス

FIDO Device Onboarding (FDO) は、以下を行うプロセスです。

- デバイスをプロビジョニングし、オンボードします。
- このデバイスの認証情報を自動的に設定します。FDO プロセスは、新しいデバイスのインストールによってトリガーされる自動オンボーディングメカニズムです。
- このデバイスがネットワーク上で安全に接続および対話できるようにします。

FIDO Device Onboarding (FDO) を使用すると、IoT アーキテクチャーに新しいデバイスを追加することで、セキュアなデバイスオンボーディングを実行できます。これには、信頼され、実行中の残りのシステムと統合する必要がある特定のデバイス設定が含まれます。FDO プロセスは、新しいデバイスのインストールによってトリガーされる自動オンボーディングメカニズムです。

FDO プロトコルは次のタスクを実行します。

- デバイスの大規模かつセキュアなオンボーディングに必要な自動化を行うとともに、信頼と所有権の連鎖を解決します。
- 製造段階でデバイスの初期化を実行し、デバイスを実際に使用するために遅延バインディングを実行します。これは、デバイスの管理システムへの実際のバインドは、デバイスの手動設定を必要とせずに、デバイスの最初の起動時に行われることを意味します。
- 自動化されたセキュアなデバイスオンボーディング、つまり、エッジロケーションに専門の担当者を必要としないゼロタッチインストールとオンボーディングをサポートします。デバイスがオンボーディングされると、管理プラットフォームはそのデバイスに接続し、パッチの適用、更新、ロールバックを実行できます。

FDO を使用すると、次の利点が得られます。

- FDO は、デバイスを管理プラットフォームに登録するための安全で簡単な方法です。キックスタート設定をイメージに埋め込む代わりに、FDO はデバイスの初回起動時にデバイス認証情報を ISO イメージに直接適用します。

- FDO は、デバイスへのレイトバインディングの問題を解決し、安全な FDO チャネルを介して機密データを共有できるようにします。
- FDO は、設定やその他のシークレットを登録してシステムに渡す前に、システムの ID と所有権を暗号で識別します。これにより、技術者以外のユーザーがシステムの電源を入れることができます。

RHEL for Edge の Simplified Installer イメージをビルドして自動的にオンボードするには、既存の OSTree コミットを提供します。作成されるイメージには、OSTree コミットがデプロイされた生のイメージが含まれます。Simplified インストーラーの ISO イメージを起動すると、ハードディスクまたは仮想マシンのブートイメージとして使用できる RHEL for Edge システムがプロビジョニングされます。

RHEL for Edge Simplified Installer イメージは、デバイスへの無人インストール用に最適化されており、ネットワークベースのデプロイメントと非ネットワークベースのデプロイメントの両方をサポートします。ただし、ネットワークベースのデプロイメントでは、UEFI HTTP ブートのみをサポートします。

FDO プロトコルは、次のサーバーに基づいています。

製造サーバー

1. デバイス認証情報を生成します。
2. プロセスの後半でデバイスの所有権を設定するために使用される、所有権バウチャーを作成します。
3. デバイスを特定の管理プラットフォームにバインドします。

所有者管理システム

1. 製造サーバーから所有権バウチャーを受け取り、関連付けられたデバイスの所有者になります。
2. プロセスの後半では、デバイス認証後にデバイスと所有者オンボーディングサーバー間にセキュアなチャネルを作成します。
3. セキュアなチャネルを使用して、オンボーディングの自動化に必要なファイルやスクリプトなどの情報をデバイスに送信します。

Service-info API サーバー

Service-info API サーバーの設定とクライアントで利用可能なモジュールとに基づいて、SSH キーとファイルのコピー、コマンドの実行、ユーザーの作成、ディスクの暗号化など、ターゲットクライアントデバイスでのオンボーディングの最後の手順を実行します。

ランデブーサーバー

1. 所有者管理システムから所有権バウチャーを取得し、デバイスの UUID を所有者サーバー IP にマッピングします。次に、ランデブーサーバーはデバイスの UUID をターゲットプラットフォームと照合し、このデバイスがどの所有者オンボーディングサーバーエンドポイントを使用する必要があるかをデバイスに通知します。
2. 初回起動時に、Rendezvous サーバーはデバイスのコンタクトポイントになり、デバイスは所有者に送られ、デバイスと所有者が安全なチャンネルを確立できるようにします。

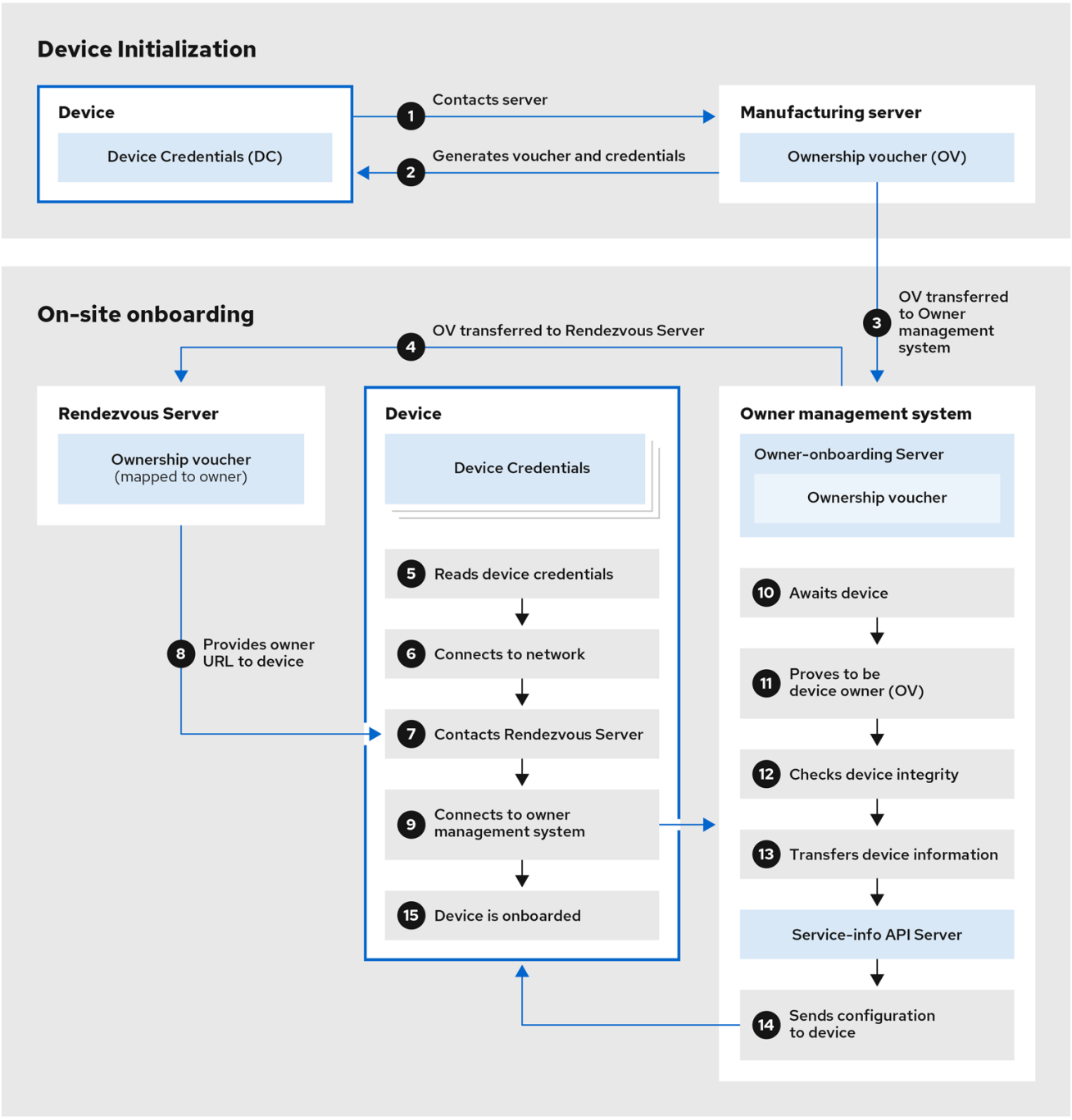
デバイスクライアント

これはデバイスにインストールされています。デバイスクライアントは、以下のアクションを実行します。

1. オンボーディングの自動化が実行される複数のサーバーへのクエリーを開始します。
2. TCP/IP プロトコルを使用してサーバーと通信します。

次の図は、FIDO デバイスのオンボーディングワークフローを表しています。

図11.1 非ネットワーク環境での RHEL for Edge のデプロイ



デバイスの初期化 では、デバイスは製造サーバーに接続して、FDO 認証情報、オペレーティングシステムにインストールされる一連の証明書とキー、およびランデブーサーバーエンドポイント (URL) を取得します。また、所有者の割り当てを変更する必要がある場合に備えて、個別に保持される所有権バウチャーも取得します。

1. デバイスが製造サーバーに接続します。

2. 製造サーバーがデバイスの所有権バウチャーとデバイス認証情報を生成します。

3. 所有権バウチャーが所有者オンボーディングサーバーに転送されます。

オンサイトのオンボーディング では、デバイスはデバイス認証情報からランデブーサーバーエンドポイント (URL) を取得し、ランデブーサーバーエンドポイントに接続してオンボーディングプロセスを開始します。これにより、デバイスは、所有者オンボーディングサーバーと Service Info API サーバーで構成される所有者管理システムにリダイレクトされます。

4. 所有者オンボーディングサーバーは、所有権バウチャーをランデブーサーバーに転送し、ランデブーサーバーは、所有権バウチャーを所有者にマッピングします。

5. デバイスクライアントがデバイス認証情報を読み込みます。

6. デバイスクライアントがネットワークに接続します。

7. ネットワークに接続した後、デバイスクライアントはランデブーサーバーに接続します。

8. ランデブーサーバーは、所有者のエンドポイント URL をデバイスクライアントに送信し、デバイスを登録します。

9. デバイスクライアントは、ランデブーサーバーによって共有された所有者オンボーディングサーバーに接続します。

10. デバイスは、デバイスキーを使用してステートメントに署名することにより、正しいデバイスであることを証明します。

11. 所有者オンボーディングサーバーは、所有者バウチャーの最後のキーを使用してステートメントに署名することで、正しいサーバーであることを証明します。

12. 所有者オンボーディングサーバーは、デバイスの情報を Service Info API サーバーに転送します。

13. Service Info API サーバーは、デバイスの設定を送信します。

14. デバイスがオンボードされます。

11.2. RHEL FOR EDGE デバイス用の自動プロビジョニングとオンボーディング

RHEL for Edge の Simplified Installer イメージをビルドして自動的にオンボードするには、既存の OSTree コミットを提供します。作成されるイメージには、OSTree コミットがデプロイされた生のイメージが含まれます。Simplified インストーラーの ISO イメージを起動すると、ハードディスクまたは仮想マシンのブートイメージとして使用できる RHEL for Edge システムがプロビジョニングされます。

RHEL for Edge Simplified Installer イメージは、デバイスへの無人インストール用に最適化されており、ネットワークベースのデプロイメントと非ネットワークベースのデプロイメントの両方をサポートします。ただし、ネットワークベースのデプロイメントでは、UEFI HTTP ブートのみをサポートします。

RHEL for Edge デバイスを自動的にプロビジョニングしてオンボーディングする手順の概要は次のとおりです。

1. RHEL システムをインストールおよび登録します。

2. RHEL Image Builder をインストールします。

3. RHEL Image Builder を使用して、**rhel-edge-container** イメージタイプ用の RHEL のカスタマイズされたブループリントを作成します。

```
name = "rhel-edge-container"
description = "Minimal RHEL for Edge Container blueprint"
version = "0.0.1"
```

4. RHEL for Edge Container ブループリントを RHEL Image Builder にインポートします。

5. RHEL for Edge Container イメージを作成します。

6. RHEL for Edge Container イメージを使用して、OSTree コミットを提供します。これは、後で RHEL for Edge Simplified Installer イメージタイプを構築する際に使用されます。

7. ストレージデバイスパスのカスタマイズと FDO のカスタマイズを使用して、**edge-simplified-installer** イメージタイプのブループリントを作成します。

```
name = "rhel-edge-simplified-installer-with-fdo"
description = "Minimal RHEL for Edge Simplified Installer with FDO blueprint"
version = "0.0.1"
packages = []
modules = []
groups = []
distro = ""

[customizations]
installation_device = "/dev/vda"

[customizations.fdo]
manufacturing_server_url = "http://10.0.0.2:8080"
diun_pub_key_insecure = "true"
```

8. RHEL for Edge イメージ用の簡略化されたインストーラー RHEL を構築します。

9. RHEL for Edge の簡略化されたインストーラーイメージをダウンロードします。

10. この時点で、FDO サーバーインフラストラクチャーが稼働している必要があり、所有者のインフラストラクチャーの一部である **service-info API** サーバーによって処理される特定のオンボーディングの詳細が設定されます。

11. 簡略化されたインストーラー ISO イメージをデバイスにインストールします。FDO クライアントは Simplified Installer ISO で実行され、UEFI ディレクトリー構造によりイメージが起動可能になります。

12. ネットワーク設定により、デバイスは製造サーバーに接続して、最初のデバイス認証情報の交換を実行できます。

13. システムがエンドポイントに到達すると、デバイスに対してデバイスの認証情報が作成されます。

14. デバイスは、デバイスの認証情報を使用して Rendezvous サーバーに到達します。この場合、Rendezvous サーバーが持つバウチャーに基づいて暗号化認証情報を確認し、Rendezvous サーバーはデバイスを所有者サーバーにリダイレクトします。

15. デバイスは所有者サーバーに接続します。相互の信頼を確立し、Service-info API サーバーの設定に基づいてオンボーディングの最後の手順を行います。たとえば、デバイスに SSH キーをイ

ンストールして、ファイルを転送し、ユーザーの作成、コマンドの実行、ファイルシステムの暗号化などを行います。

関連情報

- FDO 自動オンボーディングテクノロジー

11.3. キーおよび証明書の生成

FIDO Device Onboarding (FDO) インフラストラクチャーを実行するには、キーと証明書を生成する必要があります。FDO はこれらのキーと証明書を生成して、製造サーバーを設定します。サービスをインストールすると、FDO によって証明書と **.yaml** 設定ファイルが自動的に生成されます。再作成はオプションです。サービスをインストールして開始すると、デフォルト設定で実行されます。

重要

Red Hat は、テクノロジープレビュー機能として **fdo-admin-tool** ツールを提供し、安全なネットワークで実行する必要があります。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **fdo-admin-cli** RPM パッケージをインストールしました

手順

1. **/etc/fdo** ディレクトリーに鍵と証明書を生成します。

```
$ for i in "diun" "manufacturer" "device-ca" "owner"; do fdo-admin-tool generate-key-and-cert $i; done
$ ls keys
device_ca_cert.pem device_ca_key.der diun_cert.pem diun_key.der manufacturer_cert.pem manufacturer_key.der owner_cert.pem owner_key.der
```

2. **/etc/fdo/keys** ディレクトリーに作成された鍵と証明書を確認します。

```
$ tree keys
```

次の出力が表示されます。

```
– device_ca_cert.pem
– device_ca_key.der
– diun_cert.pem
– diun_key.dre
– manufacturer_cert.pem
– manufacturer_key.der
– owner_cert.pem
– owner_key.pem
```

関連情報

- **fdo-admin-tool generate-key-and-cert –-help** の man ページを参照してください。

11.4. 製造サーバーのインストールと実行

fdo-manufacturing-server RPM パッケージを使用すると、FDO プロトコルの製造サーバーコンポーネントを実行できます。また、所有者バウチャー、製造者キー、製造セッションに関する情報など、他のコンポーネントも保存されます。デバイスのインストール中に、製造サーバーは、GUID、rendezvous 情報、その他のメタデータを含む、特定のデバイスのデバイス認証情報を生成します。プロセスの後半で、デバイスはこの rendezvous 情報を使用して Rendezvous サーバーに接続します。

重要

Red Hat は **fdo-manufacturing-server** ツールをテクノロジープレビュー機能として提供しています。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) ではサポートされず、機能的に完全ではない可能性があるため、安全なネットワーク上で実行する必要があります。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

manufacturing server の RPM パッケージをインストールするには、次の手順を実行します。

手順

1. **fdo-admin-cli** パッケージをインストールします。

```
# dnf install -y fdo-admin-cli
```

2. **fdo-manufacturing-server** RPM パッケージがインストールされているかどうかを確認します。

```
$ rpm -qa | grep fdo-manufacturing-server --refresh
```

3. ファイルが正しくインストールされたかどうかを確認します。

```
$ ls /usr/share/doc/fdo
```

次の出力が表示されます。

```
Output:
manufacturing-server.yml
owner-onboarding-server.yml
rendezvous-info.yml
rendezvous-server.yml
serviceinfo-api-server.yml
```

4. オプション: 次に、各ファイルの内容を確認します。

```
$ cat /usr/share/doc/fdo/manufacturing-server.yml
```

5. 製造サーバーを設定します。次の情報を提供する必要があります。

   - 製造サーバーの URL
   - Rendezvous サーバーの IP アドレスまたは DNS 名
   - 生成したキーおよび証明書へのパス。キーおよび証明書の生成 を参照してください。
     製造サーバーの設定ファイルの例は、**/usr/share/doc/fdo/manufacturing-server.yml** ディレクトリーにあります。以下は、**/etc/fdo** ディレクトリーに作成および保存される **manufacturing server.yml** の例です。これには、作成したディレクトリー、証明書、キー、ランデブーサーバーの IP アドレス、およびデフォルトのポートへのパスが含まれています。

     ```
     session_store_driver:
       Directory:
         path: /etc/fdo/stores/manufacturing_sessions/
     ownership_voucher_store_driver:
       Directory:
         path: /etc/fdo/stores/owner_vouchers
     public_key_store_driver:
       Directory:
         path: /etc/fdo/stores/manufacturer_keys
     bind: "0.0.0.0:8080"
     protocols:
       plain_di: false
       diun:
         mfg_string_type: SerialNumber
         key_type: SECP384R1
         allowed_key_storage_types:
           - Tpm
           - FileSystem
         key_path: /etc/fdo/keys/diun_key.der
         cert_path: /etc/fdo/keys/diun_cert.pem
     rendezvous_info:
       - deviceport: 8082
         ip_address: 192.168.122.99
         ownerport: 8082
         protocol: http
     manufacturing:
       manufacturer_cert_path: /etc/fdo/keys/manufacturer_cert.pem
       device_cert_ca_private_key: /etc/fdo/keys/device_ca_key.der
       device_cert_ca_chain: /etc/fdo/keys/device_ca_cert.pem
       owner_cert_path: /etc/fdo/keys/owner_cert.pem
       manufacturer_private_key: /etc/fdo/keys/manufacturer_key.der
     ```

6. 製造サーバーを起動します。

   a. systemd ユニットファイルがサーバー内にあるかどうかを確認します。

      ```
      # systemctl list-unit-files | grep fdo | grep manufacturing
      fdo-manufacturing-server.service disabled disabled
      ```

   b. 製造サーバーを有効にして起動します。

      ```
      # systemctl enable --now fdo-manufacturing-server.service
      ```

c. ファイアウォールのデフォルトのポートを開きます。

```
# firewall-cmd --add-port=8080/tcp --permanent
# systemctl restart firewalld
```

d. サービスがポート 8080 でリッスンしていることを確認します。

```
# ss -ltn
```

7. 簡略化されたインストーラーを使用して、RHEL for Edge をシステムにインストールします。簡略化されたインストーラーイメージを構築して、RHEL for Edge image イメージをプロビジョニングする を参照してください。

関連情報

- manufacturing-server.yml の例
- FDO 自動オンボーディング用語

11.5. ランデブーサーバーのインストール、設定、および実行

fdo-rendezvous-server RPM パッケージをインストールして、最初のデバイスの起動時に製造サーバーによって生成されたバウチャーをシステムが受信できるようにします。すると、ランデブーサーバーはデバイスの UUID をターゲットプラットフォームまたはクラウドと照合し、デバイスがどの所有者サーバーエンドポイントを使用する必要があるかをデバイスに通知します。

前提条件

- **manufacturer_cert.pem** 証明書を作成した。キーおよび証明書の生成 を参照してください。
- **maker_cert.pem** 証明書をランデブーサーバーの **/etc/fdo/keys** ディレクトリーにコピーした。

手順

1. **fdo-rendezvous-server** RPM パッケージをインストールします。

```
# dnf install -y fdo-rendezvous-server
```

2. 製造元証明書へのパスを含む、**rendezvous-server.yml** 設定ファイルを作成します。**/usr/share/doc/fdo/rendezvous-server.yml** に例があります。次の例は、**/etc/fdo/rendezvous-server.yml** に保存される設定ファイルを示しています。

```
storage_driver:
  Directory:
    path: /etc/fdo/stores/rendezvous_registered
session_store_driver:
  Directory:
    path: /etc/fdo/stores/rendezvous_sessions
trusted_manufacturer_keys_path: /etc/fdo/keys/manufacturer_cert.pem
max_wait_seconds: ~
bind: "0.0.0.0:8082"
```

3. ランデブーサーバーサービスのステータスを確認します。

-

```
# systemctl list-unit-files | grep fdo | grep rende
fdo-rendezvous-server.service disabled disabled
```

a. サービスが停止して無効になっている場合は、有効にして起動します。

```
# systemctl enable --now fdo-rendezvous-server.service
```

4. サーバーがデフォルト設定のポート 8082 でリッスンしていることを確認します。

```
# ss -ltn
```

5. このサーバーにファイアウォールが設定されている場合は、ポートを開きます。

```
# firewall-cmd --add-port=8082/tcp --permanent
# systemctl restart firewalld
```

11.6. 所有者サーバーのインストール、設定、および実行

fdo-owner-cli および **fdo-owner-onboarding-server** RPM パッケージをインストールして、最初のデバイスの起動時に製造サーバーによって生成されたバウチャーをシステムが受信できるようにします。すると、ランデブーサーバーはデバイスの UUID をターゲットプラットフォームまたはクラウドと照合し、デバイスがどの所有者サーバーエンドポイントを使用する必要があるかをデバイスに通知します。

前提条件

- サーバーがデプロイされるデバイスに、ディスクを暗号化する Trusted Platform Module (TPM) デバイスが搭載されている。搭載されていない場合、RHEL for Edge デバイスを起動するときにエラーが発生します。
- 鍵と証明書を含む **device_ca_cert.pem**、**owner_key.der**、および **owner_cert.pem** を作成し、それらを **/etc/fdo/keys** ディレクトリーにコピーした。

手順

1. このサーバーに必要な RPM をインストールします。

```
# dnf install -y fdo-owner-cli fdo-owner-onboarding-server
```

2. **owner-onboarding-server.yml** 設定ファイルを準備し、**/etc/fdo/** ディレクトリーに保存します。このファイルには、コピー済みの証明書へのパスと、所有者サーバーサービスを公開する場所に関する情報を含めます。
以下は、**/usr/share/doc/fdo/owner-onboarding-server.yml** で利用できる例です。URL や認証トークンなど、Service Info API への参照を見つけることができます。

```
---
ownership_voucher_store_driver:
  Directory:
    path: /etc/fdo/stores/owner_vouchers
session_store_driver:
  Directory:
    path: /etc/fdo/stores/owner_onboarding_sessions
trusted_device_keys_path: /etc/fdo/keys/device_ca_cert.pem
owner_private_key_path: /etc/fdo/keys/owner_key.der
```

```
owner_public_key_path: /etc/fdo/keys/owner_cert.pem
bind: "0.0.0.0:8081"
service_info_api_url: "http://localhost:8083/device_info"
service_info_api_authentication:
  BearerToken:
    token: Kpt5P/5flBkaiNSvDYS3cEdBQXJn2Zv9n1D50431/lo=
owner_addresses:
  - transport: http
    addresses:
      - ip_address: 192.168.122.149
```

3. Service Info API を作成して設定します。

   a. ユーザー作成、コピーまたは作成するファイル、実行するコマンド、暗号化するディスクなどのオンボーディング用の自動化情報を追加します。**/usr/share/doc/fdo/serviceinfo-api-server.yml** にある Service Info API 設定ファイルの例をテンプレートとして使用し、**/etc/fdo/** の下に設定ファイルを作成します。

```
---
service_info:
  initial_user:
    username: admin
    sshkeys:
    - "ssh-rsa AAAA...."
  files:
  - path: /root/resolv.conf
    source_path: /etc/resolv.conf
  commands:
  - command: touch
    args:
    - /root/test
    return_stdout: true
    return_stderr: true
  diskencryption_clevis:
  - disk_label: /dev/vda4
    binding:
      pin: tpm2
      config: "{}"
    reencrypt: true
  additional_serviceinfo: ~
bind: "0.0.0.0:8083"
device_specific_store_driver:
  Directory:
    path: /etc/fdo/stores/serviceinfo_api_devices
service_info_auth_token: Kpt5P/5flBkaiNSvDYS3cEdBQXJn2Zv9n1D50431/lo=
admin_auth_token: zJNoErq7aa0RusJ1w0tkTjdITdMCWYkndzVv7F0V42Q=
```

4. systemd ユニットのステータスを確認します。

```
# systemctl list-unit-files | grep fdo
fdo-owner-onboarding-server.service        disabled        disabled
fdo-serviceinfo-api-server.service         disabled        disabled
```

   a. サービスが停止して無効になっている場合は、有効にして起動します。

```
# systemctl enable --now fdo-owner-onboarding-server.service
# systemctl enable --now fdo-serviceinfo-api-server.service
```

注記

設定ファイルを変更するたびに、**systemd** サービスを再起動する必要があります。

5. サーバーがデフォルト設定のポート 8083 でリッスンしていることを確認します。

```
# ss -ltn
```

6. このサーバーにファイアウォールが設定されている場合は、ポートを開きます。

```
# firewall-cmd --add-port=8081/tcp --permanent
# firewall-cmd --add-port=8083/tcp --permanent
# systemctl restart firewalld
```

11.7. FDO 認証を使用して RHEL FOR EDGE デバイスを自動的にオンボーディング

RHEL for Edge デバイスを自動的にオンボードし、インストールプロセスの一環としてプロビジョニングするようにデバイスを準備するには、次の手順を実行します。

前提条件

- RHEL for Edge の OSTree コミットを構築し、それを使用して **Edge-simplified-installer** アーティファクトを生成した。
- デバイスが組み立てられている。
- **fdo-manufacturing-server** RPM パッケージがインストールされている。製造サーバーパッケージのインストール を参照してください。

手順

1. デバイス上で RHEL for Edge の簡略化されたインストーラーイメージを起動して、インストールプロセスを開始します。たとえば、CD-ROM または USB フラッシュドライブからインストールできます。
2. ターミナルを介して、デバイスが製造サービスに到達し、最初のデバイスクレデンシャルエクスチェンジを実行して、所有権バウチャーを作成したことを確認します。
   所有権バウチャーは、**Manufacturing-sever.yml** ファイルの **ownerning_voucher_store_driver:** パラメーターで設定されている保管場所にあります。

   このディレクトリーには、正しいデバイスクレデンシャルがデバイスに追加されたことを示す、GUID 形式の名前を持つ **owner_voucher** ファイルが含まれているはずです。

   オンボーディングサーバーは、デバイス認証情報を使用して、オンボーディングサーバーに対して認証を行います。次に、設定をデバイスに渡します。デバイスはオンボーディングサーバーから設定を受け取った後、SSH キーを受け取り受信、デバイスにオペレーティングシステムをインストールします。最後に、システムは自動的に再起動し、TPM に保存されている強力なキーで暗号化します。

検証

デバイスが自動的に再起動した後、FDO プロセスの一部として作成した認証情報を使用してデバイスにログインできます。

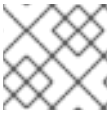
- Service Info API で作成したユーザー名とパスワードを入力して、デバイスにログインします。

関連情報

- USB フラッシュドライブからの Simplified ISO イメージのデプロイ

第12章 FDO を使用してデータベースバックエンドで RHEL FOR EDGE デバイスをオンボードする

FDO サーバー (**manufacturer-server**、**onboarding-server**、および **rendezvous**) を使用すると、ファイルではなく、SQLite や PostgreSQL データベースなどの SQL バックエンドから Owner Voucher を保存および照会できるようになります。この方法では、FDO サーバーオプションで SQL データストアを選択し、認証情報やその他のパラメーターとともに、Owner Voucher を SQL データベースに保存して、RHEL for Edge デバイスをオンボードすることができます。SQL ファイルは RPM にパッケージ化されています。

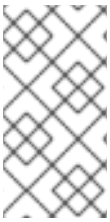


注記

現時点では、SQL バックエンドはすべての FDO 機能をサポートしていません。

12.1. FDO データベースを使用したデバイスのオンボーディング

SQL データベースを使用して Edge デバイスをオンボードします。次の例では **diesel** ツールを使用していますが、SQLite データベースまたは PostgreSQL データベースを使用することもできます。



注記

一部のサーバーでは異なるデータベースストレージを使用し、他のサーバーではファイルシステムストレージを使用できます。たとえば、Manufacturing サーバーではファイルシステムストレージオンボーディングを使用し、Rendezvous サーバーと Owner サーバーでは Postgres を使用します。

前提条件

- FDO を使用してキーと証明書を生成し、製造サーバーを設定している。[キーと証明書の生成] へのリンクを参照してください。
- 製造サーバーをインストールして設定している。製造サーバーのインストールと実行 を参照してください。
- ランデブーサーバーをインストールして設定している。ランデブーサーバーのインストール、設定、および実行 を参照してください。
- 所有者サーバーをインストールして設定している。所有者サーバーのインストール、設定、および実行 を参照してください。
- サーバー設定ファイルが **/etc/fdo** にある。
- RHEL for Edge の OSTree コミットを構築し、それを使用して **Edge-simplified-installer** アーティファクトを生成している。
- デバイスが組み立てられている。
- **diesel** ツールまたは SQL データベースをホストにインストールしている。
- データベースシステムを設定し、テーブルを作成する権限がある。

手順

1. 以下のパッケージをインストールします。

```
$ dnf  install -y sqlite sqlite-devel libpq libpq-devel
```

2. **/usr/share/doc/fdo/migrations/*** ディレクトリーにアクセスします。これには、製造サーバー、ランデブーサーバー、および所有者サーバーの RPM をインストールした後、サーバーとタイプの組み合わせごとにデータベースを作成するのに必要な **.sql** ファイルが含まれています。

3. データベースの内容を初期化します。SQLite や PostgreSQL などの SQL データベース、または **diesel** ツールを使用して SQL を実行することができます。

   - SQL データベースを使用している場合は、ユーザーの作成、アクセス管理などを使用してデータベースサーバーを設定します。データベースサーバーを設定したら、データベースを実行できます。

   - データベースシステムで **.sql** ファイルを実行しない場合は、**diesel** ツールを使用して sql を実行できます。**diesel** ツールを使用している場合は、データベースを設定した後、**diesel migration run** コマンドを使用してデータベースを作成します。

4. DB システムを設定した後、**/usr/share/doc/fdo/migrations/*** にインストールされた .sql ファイルを使用して、各サーバータイプのデータベースを作成できます。
   初期化するサーバータイプとデータベースタイプに一致する **.sql** ファイルを使用する必要があります。たとえば、PostgreSQL データベースで Owner Onboarding Server を初期化する場合は、**/usr/share/doc/fdo/migrations/migrations_owner_onboarding_server_postgres/up.sql** フォルダーを使用する必要があります。移行フォルダー内の **up.sql** ファイルはデータベースを作成し、**down.sql** ファイルはデータベースを破棄します。

5. データベースを作成したら、特定のサーバーの設定ファイルを変更して、データベースが使用されるようにします。
   各サーバーにはストレージ設定セクションがあります。

   - Manufacturer サーバー: **ownership_voucher_store_driver**

   - Owner サーバー: **ownership_voucher_store_driver**

   - Rendezvous サーバー: **storage_driver**

     a. **manufacturing-server.yml** ファイルをエディターで開き、ストアデータベースを変更します。

     ```
     $ sudo editor manufacturing-server.yml
     ```

     b. Directory セクションの **ownership_voucher_store_driver** 設定を変更します。

     ```
     $ /home/rhel/fido-device-onboard-rs/aio-dir/stores/owner_vouchers
     ```

     c. 以下の詳細を指定します。

   - 使用しているデータベースの種類: SQLite または PostgreSQL

   - サーバータイプ: たとえば、PostgreSQL を使用する場合は、次の構成を設定します。

     ```
     ownership_voucher_store_driver:
       Postgres:
         Manufacturer
     ```

a. 手順を繰り返して、Owner サーバーおよび Rendezvous sサーバーを設定します。

6. FDO オンボーディングサービスを実行します。詳細は、FDO を使用した RHEL for Edge デバイスの自動プロビジョニングとオンボーディングを参照してください。製造サーバーを実行して、FDO オンボーディングプロセスデバイスの初期化を開始します。

```
$ sudo LOG-LEVEL=debug SQLITE_MANUFACTURER-DATABASE_URL=./manufacturer-db.sqlite ./usr/libexec/fdo/fdo-manufacturing-server
```

オンボーディングプロセスは 2 つのフェーズで行われます。

- 通常、製造現場で行われるデバイス初期化フェーズ。
- デバイスの最終目的地で行われるデバイスオンボーディングプロセス。
  その結果、Manufacturing Server のデータベースに保存されている Ownership Voucher をエクスポートし、最終的な Owner Database に転送する必要があります。

7. Ownership Voucher をエクスポートするには、製造バウチャーデータベースファイルから Owner Voucher を所有者にコピーして、FDO オンボーディングプロトコルを続行します。

a. **export** フォルダーを作成します。

```
$ mkdir export
```

b. コマンドに必要なすべての変数を指定して、Manufacturing Database にある Owner Voucher をエクスポートします。

```
$ fdo-owner-tool export-manufacturer-vouchers DB_TYPE DB_URL PATH [GUID]
```

DB_TYPE

Owner Voucher を保持する Manufacturing DB のタイプ: sqlite、 postgres

DB_URL

データベース接続 URL またはデータベースファイルへのパス

PATH

Owner Voucher がエクスポートされるディレクトリーへのパス

GUID

エクスポートする所有者バウチャーの GUID。GUID を指定しない場合は、すべての Owner Voucher がエクスポートされます。

8. OV は最終 Owner のデータベースに配信される必要があります。そのためには、**fdo-owner-tool** を使用して Owner Voucher をインポートします。所有者データベースに変更します。次のコマンドを実行して、Ownership Voucher をインポートします。

```
$ fdo-owner-tool import-ownership-vouchers DB_TYPE DB_URL SOURCE_PATH
```

DB_TYPE

OV をインポートする所有者 DB のタイプ。可能な値: sqlite、 postgres

DB_URL

DB 接続 URL または DB ファイルへのパス

SOURCE_PATH

インポートする OV へのパス、またはインポートするすべての OV が配置されているディレクトリーへのパス

このコマンドは、<SOURCE_PATH> で指定された各 OV を 1 つずつ読み取り、データベースにインポートしようとします。コマンドがエラーを検出すると、障害のある OV の GUID と、エラーの原因に関する情報が指定された出力が返されます。障害のない OV がデータベースにインポートされます。デバイスはオンボーディングサーバーから設定を受信します。次に、デバイスは SSH キーを受信し、デバイスにオペレーティングシステムのインストールを開始します。最後に、デバイス内のオペレーティングシステムが自動的に再起動し、TPM に保存されている強力なキーを使用してデバイスを暗号化します。

第13章 ネットワークベース環境での RHEL FOR EDGE イメージのデプロイ

RHEL インストーラーのグラフィカルユーザーインターフェイスまたはキックスタートファイルを使用して、RHEL for Edge イメージをデプロイできます。RHEL for Edge イメージをデプロイする全体的なプロセスは、デプロイメント環境がネットワークベースであるかそうでないかによって異なります。

注記

ベアメタルにイメージをデプロイするには、キックスタートファイルを使用します。

ネットワークベースのデプロイメント

ネットワークベースの環境に RHEL for Edge イメージをデプロイするには、以下の概要手順を実施します。

a. イメージファイルのコンテンツをデプロイメントする。

b. Web サーバーを設定する

c. イメージをインストールする。

13.1. RHEL FOR EDGE イメージのコミットの展開

コミットをダウンロードしたら、**.tar** ファイルを展開して ref 名とコミット ID をメモします。

ダウンロードしたコミットファイルは、**OSTree** リポジトリーが含まれる **.tar** ファイルで構成されています。**OSTree** リポジトリーには、コミットおよび **compose.json** ファイルがあります。

compose.json ファイルには、「Ref」(参照 ID およびコミット ID) などの情報や、コミットに関する情報メタデータが含まれます。コミット ID には RPM パッケージがあります。

パッケージのコンテンツをデプロイメントするには、以下の手順を実行します。

前提条件

- キックスタートファイルを作成するか、既存のファイルを使用する。

手順

1. ダウンロードしたイメージの **.tar** ファイルを展開します。

```
# tar xvf <UUID>-commit.tar
```

2. **.tar** ファイルを展開したディレクトリーに移動します。
これには、**compose.json** ファイルおよび OSTree ディレクトリーがあります。**compose.json** ファイルにはコミット番号があり、**OSTree** ディレクトリーには RPM パッケージがあります。

3. **compose.json** ファイルを開き、コミット ID 番号を書き留めます。Web サーバーの設定を進める際には、この番号が必要になります。
jq JSON プロセッサーがインストールされている場合は、**jq** ツールを使用してコミット ID を取得することもできます。

```
# jq '.["ostree-commit"]' < compose.json
```

4. コミットの RPM パッケージを一覧表示します。

```
# rpm-ostree db list rhel/9/x86_64/edge --repo=repo
```

5. キックスタートファイルを使用して RHEL インストーラーを実行します。必要に応じて、既存のファイルを使用するか、Kickstart Generator ツールを使用して作成できます。
キックスタートファイルに、ファイルシステムのプロビジョニング、ユーザーの作成、RHEL for Edge イメージの取得およびデプロイの方法の詳細を含めるようにしてください。RHEL インストーラーは、インストールプロセス中にこの情報を使用します。

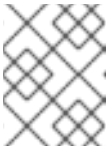
   以下は、キックスタートファイルの例です。

```
lang en_US.UTF-8
keyboard us
timezone Etc/UTC --isUtc
text
zerombr
clearpart --all --initlabel
autopart
reboot
user --name=core --group=wheel
sshkey --username=core "ssh-rsa AAAA3Nza…."
rootpw --lock
network --bootproto=dhcp

ostreesetup --nogpg --osname=rhel --remote=edge --url=https://mirror.example.com/repo/ --
ref=rhel/9/x86_64/edge
```

   OStree ベースのインストールでは、**ostreesetup** コマンドを使用して設定をセットアップします。OSTree のコミットを、以下のフラグを使用して取得します。

   - **--nogpg** - GNU Privacy Guard (GPG) キー検証を無効にします。
   - **--osname** - オペレーティングシステムのインストールの管理ルート。
   - **--remote** - オペレーティングシステムのインストールの管理ルート。
   - **--url** - インストール元となるリポジトリーの URL
   - **--ref** - インストールが使用するリポジトリーからのブランチの名前。
   - **--url=http://mirror.example.com/repo/** - edge-commit を抽出して **nginx** で提供したホストシステムのアドレス。このアドレスを使用して、ゲストコンピューターからホストシステムにアクセスすることができます。
     たとえば、**/var/www/html** ディレクトリーにコミットイメージを抽出し、ホスト名が**www.example.com** のコンピューターで **nginx** を介してコミットを提供する場合、**--url** パラメーターの値は **http://www.example.com/repo** となります。



注記

Apache HTTP Server では https が有効になっていないため、http プロトコルを使用してサービスを開始し、コミットを提供します。

関連情報

- RHEL for Edge イメージのダウンロード
- キックスタートファイルの作成

13.2. RHEL FOR EDGE イメージをインストールするための WEB サーバーの設定

RHEL for Edge イメージのコンテンツを展開したら、HTTP を使用して RHEL インストーラーにイメージのコミット詳細を提供するための Web サーバーを設定します。

以下の例では、コンテナーを使用して Web サーバーを設定する手順を説明します。

前提条件

- システムに Podman をインストール済みである。How do I install Podman in RHEL を参照してください。

手順

1. 以下の手順に従って、**nginx** 設定ファイルを作成します。

```
events {

}

http {
    server{
        listen 8080;
        root /usr/share/nginx/html;
                }
         }

pid /run/nginx.pid;
daemon off;
```

2. 以下の手順で Dockerfile を作成します。

```
FROM registry.access.redhat.com/ubi8/ubi
RUN dnf -y install nginx && dnf clean all
COPY kickstart.ks /usr/share/nginx/html/
COPY repo /usr/share/nginx/html/
COPY nginx /etc/nginx.conf
EXPOSE 8080
CMD ["/usr/sbin/nginx", "-c", "/etc/nginx.conf"]
ARG commit
ADD ${commit} /usr/share/nginx/html/
```

詳細は以下のようになります。

- **kickstart.ks** は、RHEL for Edge イメージのキックスタートファイルの名前です。キックスタートファイルには、ディレクティブの情報が含まれています。後でイメージを管理しやすくするためにも、greenboot チェックのチェックおよび設定を含むことが推奨されま

す。そのためには、以下の設定を含むようにキックスタートファイルを更新します。

```
lang en_US.UTF-8
keyboard us
timezone Etc/UTC --isUtc
text
zerombr
clearpart --all --initlabel
autopart
reboot
user --name=core --group=wheel
sshkey --username=core "ssh-rsa AAAA3Nza…."

ostreesetup --nogpg --osname=rhel --remote=edge
--url=https://mirror.example.com/repo/
--ref=rhel/9/x86_64/edge

%post
cat << EOF > /etc/greenboot/check/required.d/check-dns.sh
#!/bin/bash

DNS_SERVER=$(grep nameserver /etc/resolv.conf | cut -f2 -d" ")
COUNT=0

# check DNS server is available
ping -c1 $DNS_SERVER
while [ $? != '0' ] && [ $COUNT -lt 10 ]; do




      COUNT++
echo "Checking for DNS: Attempt $COUNT ."
sleep 10
ping -c 1 $DNS_SERVER
done
EOF
%end
```

任意の HTTP サービスで OSTree リポジトリーをホストできます。コンテナーを使用する例は、これを行う方法のオプションにすぎません。Dockerfile は次のタスクを実行します。

a. 最新の Universal Base Image (UBI) の使用

b. Web サーバー (nginx) をインストールします。

c. キックスタートファイルのサーバーへの追加

d. RHEL for Edge イメージのコミットのサーバーへの追加

3. Docker コンテナーをビルドします。

```
# podman build -t name-of-container-image --build-arg commit=uuid-commit.tar .
```

4. コンテナーを実行します。

```
# podman run --rm -d -p port:8080 localhost/name-of-container-image
```

これにより、サーバーがセットアップされ、**commit.tar** リポジトリーとキックスタートファイルを使用して、RHEL インストーラーを起動する準備が整いました。

13.3. キックスタートを使用したエッジデバイスへの有人インストールの実行

ネットワークベース環境での有人インストールの場合、RHEL インストーラー ISO、キックスタートファイル、および Web サーバーを使用して、RHEL for Edge イメージをデバイスにインストールできます。Web サーバーは、RHEL for Edge Commit と、RHEL インストーラー ISO イメージを起動するためのキックスタートファイルを提供します。

前提条件

- Web サーバーを実行して、RHEL for Edge Commit を使用できるようにした。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。
- 有人インストールのターゲットとして使用する **.qcow2** ディスクイメージを作成した。qemu-img を使用した仮想ディスクイメージの作成 を参照してください。

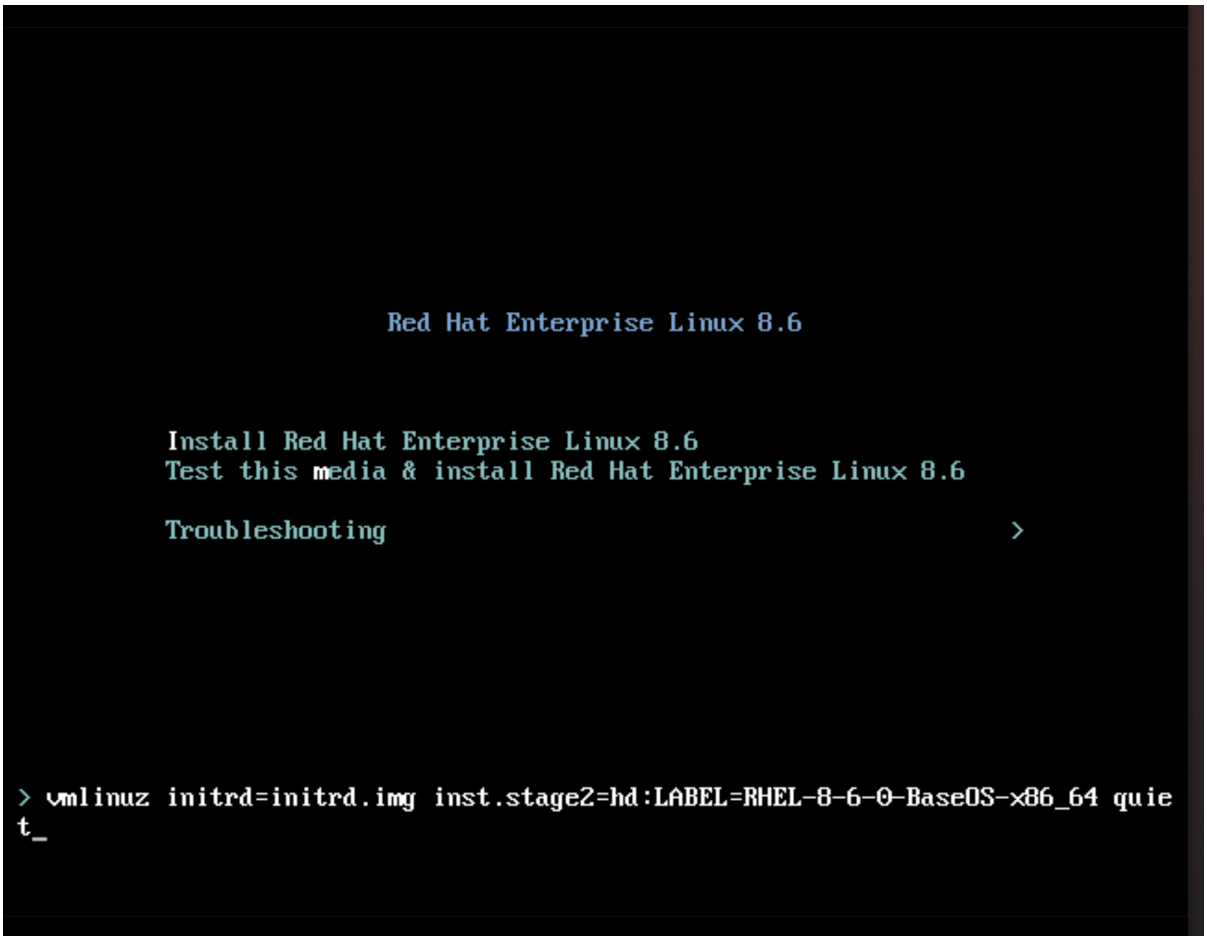
手順

1. **libvirt virt-install** ユーティリティーを使用して RHEL Anaconda インストーラーを実行し、RHEL オペレーティングシステムで仮想マシンを作成します。**.qcow2** ディスクイメージを有人インストールのターゲットディスクとして使用します。

```
virt-install \
--name rhel-edge-test-1 \
--memory 2048 \
--vcpus 2 \
--disk path=prepared_disk_image.qcow2,format=qcow2,size=8 \
--os-variant rhel9 \
--cdrom /home/username/Downloads/rhel-9-x86_64-boot.iso
```

2. インストール画面で:

図13.1 Red Hat Enterprise Linux 起動メニュー



a. **e** キーを押して、追加のカーネルパラメーターを追加します。

```
inst.ks=http://edge_device_ip:port/kickstart.ks
```

カーネルパラメーターは、RHEL インストーラーに含まれる RHEL イメージではなく、キックスタートファイルを使用して RHEL をインストールすることを指定します。

b. カーネルパラメーターを追加したら、**Ctrl**+**X** を押して、キックスタートファイルを使用して RHEL インストールを起動します。
RHEL インストーラーが起動し、サーバー (HTTP) エンドポイントからキックスタートファイルを取得してコマンドを実行します。このコマンドには、HTTP エンドポイントから RHEL for Edge イメージコミットをインストールするコマンドが含まれます。インストールが完了すると、RHEL インストーラーからログインの詳細を求められます。

検証

1. ログイン 画面で、ユーザーアカウントの認証情報を入力し、**Enter** をクリックします。
2. RHEL for Edge イメージが正常にインストールされているかどうかを確認します。

```
$ rpm-ostree status
```

コマンドの出力には、イメージのコミット ID が表示され、正常にインストールされたことがわかります。

以下は出力例です。

```
State: idle
Deployments:
* ostree://edge:rhel/9/x86_64/edge
    Timestamp: 2020-09-18T20:06:54Z
    Commit: 836e637095554e0b634a0a48ea05c75280519dd6576a392635e6fa7d4d5e96
```

関連情報

- キックスタートファイルを ISO イメージに埋め込む方法
- インストールの起動

13.4. キックスタートを使用したエッジデバイスへの無人インストールの実行

ネットワークベース環境での無人インストールの場合、キックスタートファイルと Web サーバーを使用して、RHEL for Edge イメージを Edge デバイスにインストールできます。Web サーバーは、RHEL for Edge Commit とキックスタートファイルを提供し、両方のアーティファクトが RHEL インストーラー ISO イメージの起動に使用されます。

前提条件

- ホストシステムに **qemu-img** ユーティリティーがインストールされている。
- 作成したコミットをインストールするための **.qcow2** ディスクイメージを作成した。CLI で RHEL Image Builder を使用してシステムイメージを作成する を参照してください。
- 実行中の Web サーバーがある。ネットワークベース以外のデプロイメント用の RHEL for Edge Container イメージの作成 を参照してください。

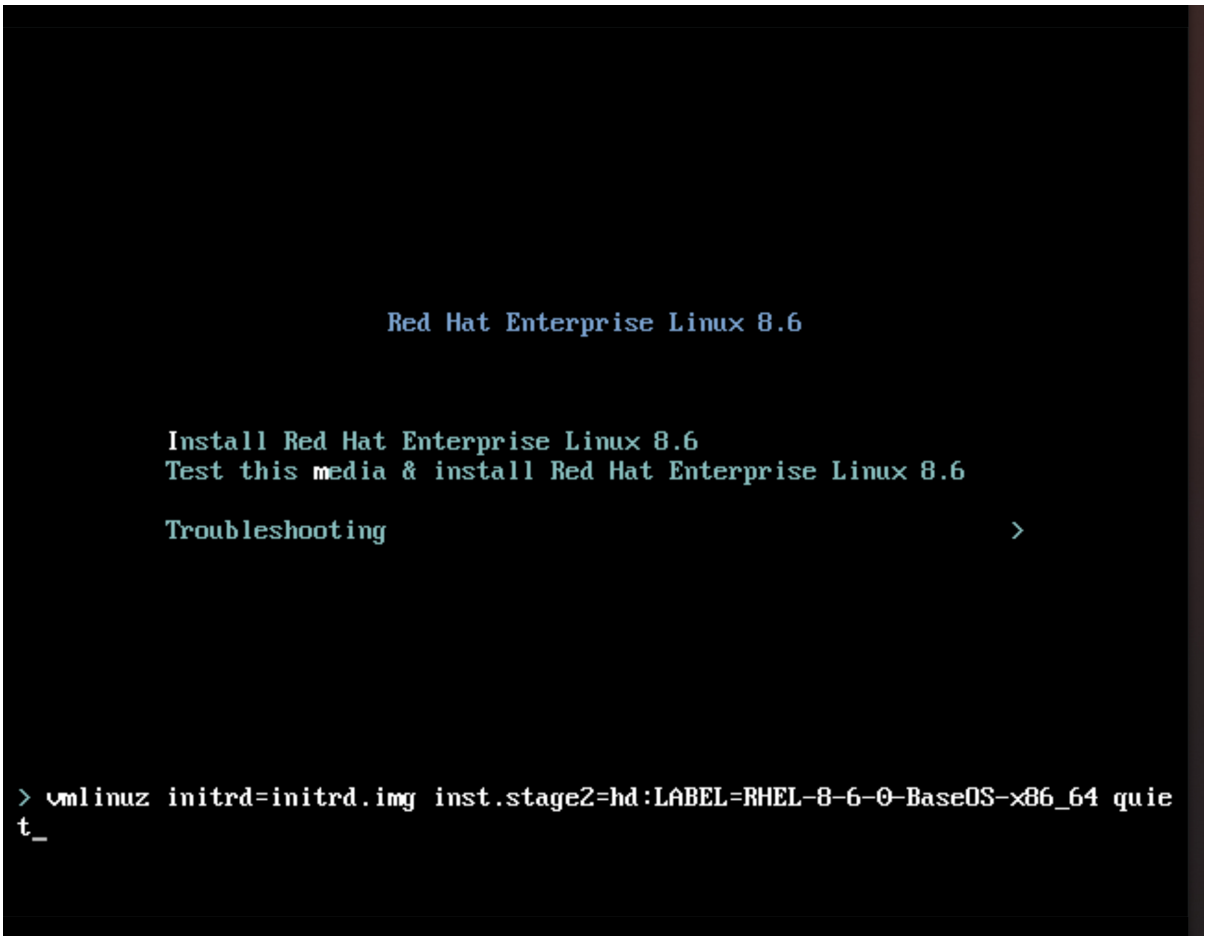
手順

1. RHEL for Edge Container イメージを実行して Web サーバーを起動します。サーバーは、RHEL for Edge Container イメージ内のコミットをフェッチして、利用可能になり実行されます。
2. **libvirt virt-install** を使用して、カスタマイズされた **.qcow2** ディスクイメージを渡して RHEL Anaconda インストーラーを実行します。

```
virt-install \
--name rhel-edge-test-1 \
--memory 2048 \
--vcpus 2 \
--disk path=prepared_disk_image.qcow2,format=qcow2,size=8 \
--os-variant rhel9 \
--cdrom /home/username/Downloads/rhel-9-x86_64-boot.iso
```

3. インストール画面で:

図13.2 Red Hat Enterprise Linux 起動メニュー



a. **TAB** キーを押して、キックスタートカーネル引数を追加します。

```
inst.ks=http://web-server_device_ip:port/kickstart.ks
```

カーネルパラメーターは、RHEL インストーラーに含まれる RHEL イメージではなく、キックスタートファイルを使用して RHEL をインストールすることを指定します。

b. カーネルパラメーターを追加したら、**Ctrl**+**X** を押して、キックスタートファイルを使用して RHEL インストールを起動します。
RHEL インストーラーが起動し、サーバー (HTTP) エンドポイントからキックスタートファイルを取得してコマンドを実行します。このコマンドには、HTTP エンドポイントから RHEL for Edge イメージコミットをインストールするコマンドが含まれます。インストールが完了すると、RHEL インストーラーからログインの詳細を求められます。

検証

1. ログイン 画面で、ユーザーアカウントの認証情報を入力し、**Enter** をクリックします。
2. RHEL for Edge イメージが正常にインストールされているかどうかを確認します。

```
$ rpm-ostree status
```

コマンドの出力には、イメージのコミット ID が表示され、正常にインストールされたことがわかります。

以下は出力例です。

```
State: idle
Deployments:
* ostree://edge:rhel/9/x86_64/edge
    Timestamp: 2020-09-18T20:06:54Z
    Commit: 836e637095554e0b634a0a48ea05c75280519dd6576a392635e6fa7d4d5e96
```

関連情報

- キックスタートファイルを ISO イメージに埋め込む方法
- インストールの起動

第14章 非ネットワークベース環境での RHEL FOR EDGE イメージのデプロイ

RHEL for Edge Container (**.tar**) を RHEL for Edge Installer (**.iso**) イメージタイプと組み合わせると、ISO イメージになります。ISO イメージは、デバイスへのイメージのデプロイメント中に切断された環境で使用できます。ただし、ネットワークアクセスでは、さまざまなアーティファクトを構築するためにネットワークアクセスが必要になる場合があります。

ネットワークベース以外の環境に RHEL for Edge イメージをデプロイするには、以下の概要手順を実施します。

1. RHEL for Edge Container をダウンロードします。RHEL for Edge イメージをダウンロードする方法は、RHEL for Edge イメージのダウンロード を参照してください。
2. RHEL for Edge Container イメージを Podman にロードする。
3. RHEL for Edge Container イメージを Podman で実行する。
4. RHEL for Edge インストーラーブループリントをロードする
5. RHEL for Edge Installer イメージをビルドする。
6. **.qcow2** ディスクを準備する
7. 仮想マシンを起動する。
8. イメージをインストールする。

14.1. ネットワークベース以外のデプロイメント用の RHEL FOR EDGE CONTAINER イメージの作成

ダウンロードした RHEL for Edge Container OSTree コミットを Podman にロードすることで、実行中のコンテナーを構築できます。そのためには、以下の手順に従います。

前提条件

- RHEL for Edge Container OSTree コミットを作成してダウンロードしました。
- システムに **Podman** をインストール済みである。How do I install Podman in RHEL を参照してください。

手順

1. RHEL for Edge Container OSTree コミットをダウンロードしたディレクトリーに移動します。
2. RHEL for Edge Container OSTree コミットを **Podman** にロードします。

```
$ sudo podman load -i UUID-container.tar
```

コマンド出力にはイメージ ID が提供されます (例: **@8e0d51f061ff1a51d157804362bc875b649b27f2ae1e66566a15e7e6530cec63**)。

3. 前のステップで生成したイメージ ID を使用して、新しい RHEL for Edge Container イメージにタグ付けします。

```
$ sudo podman tag image-ID localhost/edge-container
```

podman tag コマンドは、ローカルイメージに別の名前を割り当てます。

4. **edge-container** という名前のコンテナーを実行します。

```
$ sudo podman run -d --name=edge-container -p 8080:8080 localhost/edge-container
```

podman run -d --name=edge-container コマンドは、**localhost/edge-container** イメージに基づいて、コンテナーに名前を割り当てます。

5. コンテナーをリスト表示します。

```
$ sudo podman ps -a
CONTAINER ID  IMAGE                                COMMAND CREATED     STATUS
PORTS   NAMES
2988198c4c4b  …./localhost/edge-container   /bin/bash  3 seconds ago  Up 2 seconds ago
edge-container
```

その結果、**Podman** はコンテナーを実行し、RHEL for Edge Container Commit の OSTree リポジトリーを提供します。

14.2. ネットワークベース以外のデプロイメント用の RHEL FOR EDGE インストーラーイメージの作成

実行中のコンテナーをビルドして、**RHEL for Edge Container** コミットでリポジトリーを提供してから、**RHEL for Edge Installer (.iso)** イメージを作成します。**RHEL for Edge Installer (.iso)** は、**RHEL for Edge Container (.tar)** が提供するコミットをプルします。**RHEL for Edge Container** コミットが Podman に読み込まれると、**OSTree** が URL 形式で公開されます。

CLI で RHEL for Edge Installer イメージを作成するには、以下の手順に従います。

前提条件

- RHEL for Edge イメージのブループリントを作成した。
- RHEL for Edge Container イメージを作成し、Web サーバーを使用してデプロイした。

手順

1. RHEL for Edge Installer インストーラーイメージの作成を開始します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url URL-OSTree-repository
blueprint-name image-type
```

詳細は以下のようになります。

- **ref** は、お客様が ostree リポジトリーの構築に使用した値と同じです。
- **URL-OSTree-repository** は、イメージに埋め込むコミットの OSTree リポジトリーへの URL です。たとえば、http://10.0.2.2:8080/repo/ です。ネットワークベース以外のデプロイメント用の RHEL for Edge Container イメージの作成 を参照してください。
- **blueprint-name** は、RHEL for Edge Installer のブループリント名です。

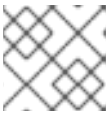
- **image-type** は **edge-installer** です。
  composer プロセスがキューに追加されたことを確認する画面が表示されます。また、作成されたイメージの UUID (Universally Unique Identifier) 番号も表示されます。UUID 番号を使用してビルドを追跡します。また、更なるタスクのために UUID 番号を手元に保管しておきます。

2. イメージの作成状態を確認します。

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

コマンド出力には、以下の形式で状態が表示されます。

```
<UUID> RUNNING date blueprint-name blueprint-version image-type
```



注記

イメージ作成プロセスの完了まで数分かかります。

イメージ作成プロセスを中断するには、以下を実行します。

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

既存イメージを削除するには、以下を実行します。

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

RHEL Image Builder は、イメージのビルド時に、実行中のコンテナーが提供するコミットをプルします。

イメージのビルドが完了したら、作成された ISO イメージをダウンロードできます。

3. イメージをダウンロードします。RHEL for Edge イメージのダウンロード を参照してください。
   イメージの準備ができたら、**ネットワーク以外のデプロイメント** に使用できます。ネットワークベース以外のデプロイメント向けの RHEL for Edge イメージのインストール を参照してください。

関連情報

- ネットワークベース以外のデプロイメント向けのコマンドラインインターフェイスを使用した RHEL for Edge インストーラーイメージの作成

14.3. ネットワークベース以外のデプロイメント向けの RHEL FOR EDGE イメージのインストール

RHEL for Edge イメージをインストールするには、以下の手順に従います。

前提条件

- RHEL for Edge Installer ISO イメージを作成した。
- 実行中のコンテナーを停止している。

- 作成したコミットをインストールするためのディスクイメージ。
- **edk2-ovmf** パッケージをインストールした。
- **virt-viewer** パッケージをインストールした。
- ユーザーアカウントを使用してブループリントをカスタマイズした。RHEL Web コンソールの RHEL Image Builder を使用した RHEL for Edge イメージのブループリントの作成 を参照してください。

警告

ブループリントでユーザーアカウントのカスタマイズを定義しないと、ISO イメージにログインできません。

手順

1. (**.iso**) イメージをインストールするための **qcow** 仮想マシンディスクファイルを作成します。これは、仮想マシン用のハードディスクドライブのイメージです。以下に例を示します。

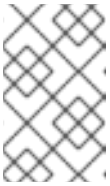
```
$ qemu-img create -f qcow2 diskfile.qcow2 20G
```

2. **virt-install** コマンドを使用して、ディスクをドライブとして使用し、Installer ISO イメージを CD-ROM として使用して仮想マシンを起動します。以下に例を示します。

```
$ virt-install \
--boot uefi \
--name VM_NAME
--memory 2048 \
--vcpus 2 \
--disk path=diskfile.qcow2
--cdrom /var/lib/libvirt/images/UUID-installer.iso \
--os-variant rhel9.0
```

このコマンドは、**virt-install** に次のように指示します。

- BIOS の代わりに UEFI を使用して起動するよう仮想マシンに指示します。
- インストール ISO をマウントします。
- 最初の手順で作成したハードディスクドライブイメージを使用します。Anaconda インストーラーを指定します。RHEL インストーラーが起動し、ISO からキックスタートファイルを読み込み、RHEL for Edge イメージのコミットのインストールなど、コマンドを実行します。インストールが完了すると、RHEL インストーラーはログイン詳細を要求します。



注記

Anaconda は、インストール時にコンテナーコミットを使用するように事前設定されています。ただし、ディスクパーティション、タイムゾーンなどのシステム設定を設定する必要があります。

3. **virt-viewer** を使用して Anaconda GUI に接続し、システム設定をセットアップします。

```
$ virt-viewer --connect qemu:///system --wait VM_NAME
```

4. システムを再起動して、インストールを完了します。
5. ログイン画面で、ユーザーアカウントの認証情報を指定し、**Enter** をクリックします。

検証手順

1. RHEL for Edge イメージが正常にインストールされているかどうかを確認します。

```
$  rpm-ostree status
```

コマンドの出力には、イメージのコミット ID が表示され、正常にインストールされたことがわかります。

第15章 RHEL FOR EDGE イメージの管理

RHEL for Edge イメージを管理するために、以下のいずれかの管理タスクを実行することができます。

- RHEL Web コンソールまたはコマンドラインで Image Builder を使用した RHEL for Edge イメージブループリントの編集
- Image Builder コマンドラインを使用したコミット更新のビルド
- RHEL for Edge イメージの更新
- **rpm-ostree** リモートをノードに設定/ノードポリシーの更新
- RHEL for Edge イメージの手動での復元、または greenboot を使用した自動的な復元

15.1. IMAGE BUILDER を使用した RHEL FOR EDGE イメージのブループリントの編集

RHEL for Edge イメージのブループリントを編集して、以下を実行することができます。

- 必要となる可能性のある追加コンポーネントの追加
- 既存コンポーネントのバージョンの変更
- 既存コンポーネントの削除

15.1.1. RHEL Web コンソールで Image Builder を使用して RHEL for Edge のブループリントにコンポーネントを追加する

RHEL for Edge イメージのブループリントにコンポーネントを追加するには、以下の前提条件を満たしていることを確認してから、対応するブループリントを編集する手順に従ってください。

前提条件

- RHEL システムで、RHEL Image Builder ダッシュボードにアクセス済みである。
- RHEL for Edge イメージのブループリントを作成済みである。

手順

1. RHEL Image Builder のダッシュボードで、編集するブループリントをクリックします。
   特定のブループリントを検索するには、フィルターテキストボックスにブループリント名を入力し、**Enter** をクリックします。
2. ブループリントの右上で、**Edit Packages** をクリックします。
   Edit blueprints ウィザードが開きます。
3. **Details** ページで、ブループリント名を更新し、**Next** をクリックします。
4. **Packages** ページで、次の手順に従います。
   a. **Available Packages** で、フィルターテキストボックスに追加するパッケージ名を入力し、**Enter** をクリックします。
      コンポーネント名のリストが表示されます。

b. **>** をクリックして、コンポーネントをブループリントに追加します。

5. **Review** ページで **Save** をクリックします。
ブループリントが新しいパッケージで更新されます。

15.1.2. Web コンソールで RHEL Image Builder を使用してブループリントからコンポーネントを削除する

RHEL Image Builder を使用して作成したブループリントから 1 つ以上の不要なコンポーネントを削除するには、次の前提条件を満たしていることを確認してから、手順に従ってください。

前提条件

- RHEL システムで、RHEL Image Builder ダッシュボードにアクセス済みである。
- RHEL for Edge イメージのブループリントを作成済みである。
- RHEL for Edge のブループリントに 1 つ以上のコンポーネントを追加済みである。

手順

1. RHEL Image Builder のダッシュボードで、編集するブループリントをクリックします。
特定のブループリントを検索するには、フィルターテキストボックスにブループリント名を入力し、**Enter** をクリックします。
2. ブループリントの右上で、**Edit Packages** をクリックします。
Edit blueprints ウィザードが開きます。
3. **Details** ページで、ブループリント名を更新し、**Next** をクリックします。
4. **Packages** ページで、次の手順に従います。
   a. **Chosen packages** から **<** をクリックして、選択したコンポーネントを削除します。**<<** をクリックしてすべてのパッケージを一度に削除することもできます。
5. **Review** ページで **Save** をクリックします。
ブループリントが更新されます。

15.1.3. コマンドラインインターフェイスを使用した RHEL for Edge イメージのブループリントの編集

RHEL Image Builder のコマンドラインを使用して、RHEL for Edge イメージのブループリントの仕様を変更することができます。そのためには、以下の前提条件を満たしていることを確認してから、手順に従って対応するブループリントを編集してください。

前提条件

- RHEL Image Builder コマンドラインにアクセスできる。
- RHEL for Edge イメージのブループリントを作成済みである。

手順

1. ローカルのテキストファイルにブループリントを保存 (エクスポート) します。

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

2. 選択した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。
   編集を終了する前に、ファイルが有効なブループリントであることを確認します。

3. バージョン番号を大きくしてください。
   Semantic Versioning スキームを使用していることを確認してください。

注記

バージョンを変更しない場合、バージョンのパッチコンポーネントが自動的に増えます。

4. 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。

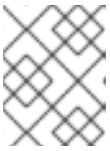
注記

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。

5. ファイルを保存してエディターを閉じます。

6. ブループリントを RHEL Image Builder サーバーにプッシュ (インポート) します。

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



注記

ブループリントを RHEL Image Builder サーバーにプッシュするときは、**.toml** 拡張子を含むファイル名を指定してください。

7. RHEL Image Builder にアップロードされたコンテンツが編集内容と一致していることを確認します。

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

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

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

15.2. RHEL FOR EDGE イメージの更新

15.2.1. RHEL for Edge イメージの更新のデプロイ方法

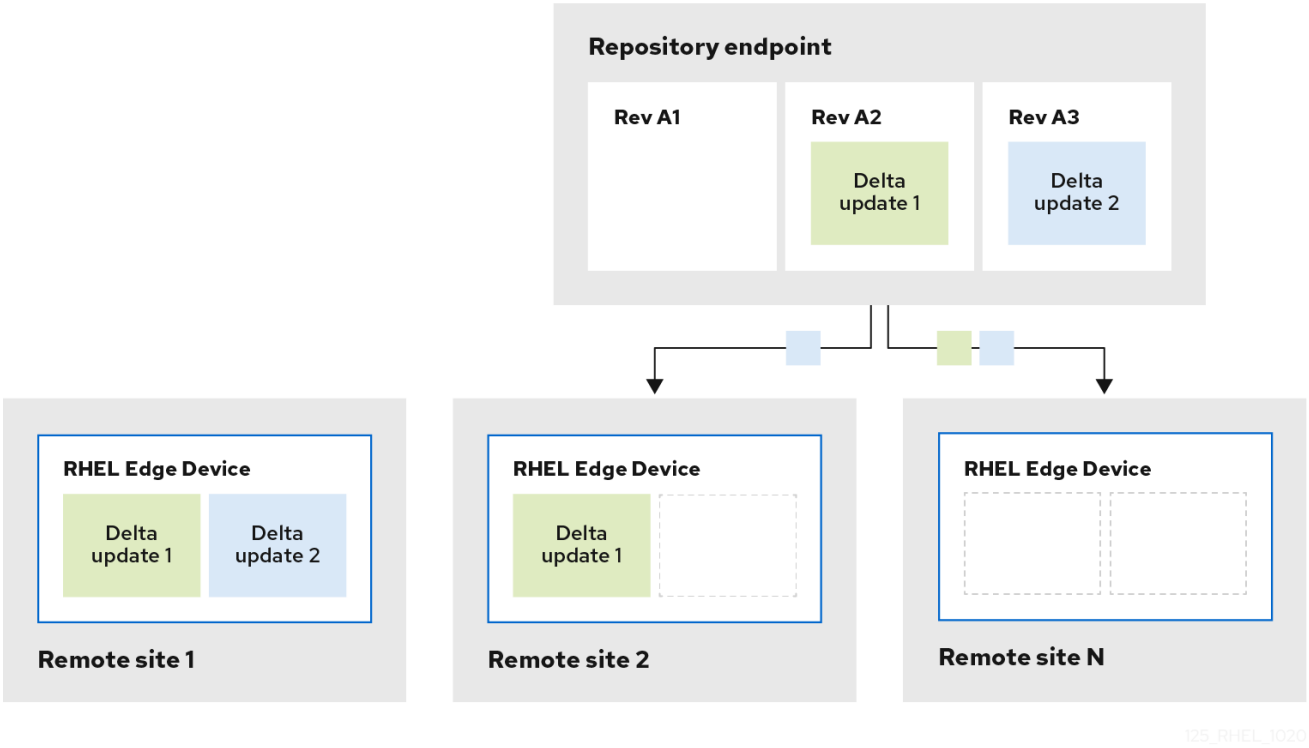
RHEL for Edge イメージでは、更新を手動でデプロイするか、デプロイメントプロセスを自動化することができます。更新はアトミックな方法で適用されます。つまり、各更新の状態が周知され、更新は段階的に行われ、再起動時にのみ適用されます。デバイスを再起動するまで変更を確認できないため、可

能な限り最大限のアップタイムを確保するために再起動をスケジュールすることができます。

イメージの更新中は、更新されたオペレーティングシステムのコンテンツのみがネットワーク経由で転送されます。これにより、イメージ全体を転送するよりもデプロイメントプロセスが効率的になります。**/usr** 内のオペレーティングシステムのバイナリーとライブラリーは **read-only** であり、**read and write** 状態は **/var** および **/etc** ディレクトリーで維持されます。

新しいデプロイメントに移動すると、**/etc** および **/var** ディレクトリーが新しいデプロイメントにコピーされ、**read and write** 権限が付与されます。**/usr** ディレクトリーは、新しいデプロイメントディレクトリーにソフトリンクとしてコピーされ、**read-only** パーミッションが設定されます。

以下の図は、RHEL for Edge イメージの更新デプロイメントプロセスを説明しています。



デフォルトでは、新しいシステムは **chroot** 操作と同様の手順を使用して起動されます。つまり、システムは、基盤となるサーバー環境への公開を制御しながら、ファイルシステムへのアクセス制御を可能にします。新しい **/sysroot** ディレクトリーには、主に次の部分があります。

- **/sysroot/ostree/repo** ディレクトリーにあるリポジトリーデータベース。
- **/sysroot/ostree/deploy/rhel/deploy** ディレクトリーのファイルシステムリビジョン。これは、システム更新の各操作によって作成されます。
- 前出のデプロイメントにリンクしている **/sysroot/ostree/boot** ディレクトリー。**/ostree** は **/sysroot/ostree** へのソフトリンクであることに注意してください。**/sysroot/ostree/boot** ディレクトリーのファイルは複製されません。デプロイメント中に変更されていない場合は、同じファイルが使用されます。ファイルは、**/sysroot/ostree/repo/objects** ディレクトリーに格納されている別のファイルへのハードリンクです。

オペレーティングシステムは、次の方法でデプロイメントを選択します。

1. **dracut** ツールは、**initramfs root** ファイルシステムの **ostree** カーネル引数を解析し、**/usr** ディレクトリーを **read-only** バインドマウントとして設定します。

2. **/sysroot** のデプロイメントディレクトリーを / ディレクトリーにバインドします。

3. **MS_MOVE** マウントフラグを使用して、**dirs** がすでにマウントされているオペレーティングシステムを再マウントします。

何か問題が発生した場合は、**rpm-ostree** cleanup コマンドを使用して古いデプロイメントを削除することで、デプロイのロールバックを実行できます。各クライアントマシンには、**/ostree/repo** に格納されている **OSTree** リポジトリーと、**/ostree/deploy/$STATEROOT/$CHECKSUM** に格納されている一連のデプロイメントが含まれています。

RHEL for Edge イメージのデプロイメントの更新により、複数のデバイス間でのシステムの一貫性が向上し、再現性が容易になり、システム状態の変更前と変更後の分離が改善されます。

15.2.2. コミット更新の構築

ブループリントに次のような変更を加えた後、コミット更新をビルドできます。

- システムに必要な追加パッケージの追加
- 既存のコンポーネントのパッケージバージョンを変更する
- 既存のパッケージを削除します。

前提条件

- RHEL Image Builder を実行しているシステムを更新した。
- ブループリントの更新を作成した。
- 以前に OSTree リポジトリーを作成し、HTTP 経由でそれを提供した。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。

手順

1. **--url**、**--ref**、**blueprint-name**、**edge-commit** の引数を使用して、新しいコミットイメージの作成を開始します。

```
# composer-cli compose start-ostree --ref rhel/9/x86_64/edge --url http://localhost:8080/repo <blueprint-name> edge-commit
```

このコマンドは、作成を開始する前に OStree リポジトリーからメタデータをフェッチするように作成プロセスに指示します。結果として得られる新しい OSTree コミットには、元の OSTree コミットの参照が親イメージとして含まれています。

2. 作成プロセスが完了したら、**.tar** ファイルをフェッチします。

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

3. コミット履歴を OSTree リポジトリーに保存できるように、コミットを一時ディレクトリーに抽出します。

```
$ tar -xf UUID.tar -C /var/tmp
```

4. **tar -xf** コマンドを使用して、結果の OSTree リポジトリーのコミットを検査します。tar ファイルがディスクに抽出されるので、結果の OSTree リポジトリーを検査できます。

```
$ ostree --repo=/var/tmp/repo log rhel/9/x86_64/edge
commit d523ef801e8b1df69ddbf73ce810521b5c44e9127a379a4e3bba5889829546fa
Parent:  f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
ContentChecksum:
f0f6703696331b661fa22d97358db48ba5f8b62711d9db83a00a79b3ae0dfe16
Date:  2023-06-04 20:22:28 /+0000
Version: 9
```

出力例では、リポジトリー内に親コミットを参照する OSTree コミットが 1 つあります。親コミットは、以前に作成した元の OSTree コミットからのチェックサムと同じです。

5. **ostree pull-local** コマンドを使用して、2 つのコミットをマージします。

```
$ sudo ostree --repo=/var/srv/httpd/repo pull-local /var/tmp/repo
20 metadata, 22 content objects imported; 0 bytes content written
```

このコマンドは、新しいメタデータとコンテンツをディスク上の場所 (**/var/tmp** など) から **/var/srv/httpd** の宛先 OSTree リポジトリーにコピーします。

検証

1. ターゲットの OSTree リポジトリーを調べます。

```
$ ostree --repo=/var/srv/httpd/repo log rhel/9/x86_64/edge
commit d523ef801e8b1df69ddbf73ce810521b5c44e9127a379a4e3bba5889829546fa
Parent:  f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
ContentChecksum:
f0f6703696331b661fa22d97358db48ba5f8b62711d9db83a00a79b3ae0dfe16
Date:  2023-06-04 20:22:28 /+0000
Version: 9
(no subject)

commit f47842de7e6859cee07d743d3c67949420874727883fa9dbbaeb5824ad949d91
ContentChecksum:
9054de3fe5f1210e3e52b38955bea0510915f89971e3b1ba121e15559d5f3a63
Date:  2023-06-04 20:01:08 /+0000
Version: 9
(no subject)
```

ターゲットの OSTree リポジトリーには、論理的な順序で 2 つのコミットが含まれていることがわかります。検証が成功したら、RHEL for Edge システムを更新できます。

15.2.3. RHEL for Edge イメージの更新の手動でのデプロイ

RHEL for Edge のブループリントを編集した後、イメージコミットを更新することができます。RHEL Image Builder は、更新された RHEL for Edge イメージの新しいコミットを生成します。この新しいコミットを使用して、最新のパッケージバージョンまたは追加パッケージのイメージをデプロイしてください。

RHEL for Edge イメージの更新をデプロイするには、前提条件を満たしていることを確認し、手順に従ってください。

前提条件

- RHEL システムで、RHEL Image Builder ダッシュボードにアクセス済みである。
- RHEL for Edge イメージのブループリントを作成済みである。
- RHEL for Edge イメージのブループリントを編集済みである。

手順

1. RHEL Image Builder のダッシュボードで、**Create Image** をクリックします。
2. **Create Image** ウィンドウで、以下の手順を実行します。
   a. Image output ページで、以下を実行します。
      i. **Select a blueprint** ドロップダウンリストから、編集したブループリントを選択します。
      ii. **Image output type** ドロップダウンリストから、**RHEL for Edge Commit (.tar)** を選択します。**Next** をクリックします。
   b. **OSTree settings** ページで、以下を入力します。
      i. **Repository URL** フィールドに、イメージに埋め込むコミットの OSTree リポジトリーへの URL を入力します。たとえば、http://10.0.2.2:8080/repo/ です。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。
      ii. **Parent commit** フィールドで、以前に生成された親コミット ID を指定します。RHEL for Edge イメージのコミットの展開 を参照してください。
      iii. **Ref** フィールドでは、コミットの名前を指定するか、空欄のままにしておくことができます。デフォルトでは、Web コンソールは **Ref** を **rhel/9/arch_name/edge** に指定します。**Next** をクリックします。
   c. **Review** ページでカスタマイズを確認し、**Create image** をクリックします。RHEL Image Builder が、更新されたブループリントの RHEL for Edge イメージの作成を開始します。イメージ作成プロセスの完了まで数分かかります。
   RHEL for Edge イメージの作成の進捗状況を確認するには、ブレッドクラムからブループリント名をクリックし、次に **Images** タブをクリックします。

   作成したイメージには、追加した最新のパッケージがある場合は含まれ、親としてオリジナルの **commit ID** を持っています。
3. 作成した RHEL for Edge Commit (**.tar**) イメージをダウンロードします。
   a. **Images** タブで **Download** をクリックして、RHEL for Edge Commit (**.tar**) イメージをシステムに保存します。
4. OSTree コミット (**.tar**) ファイルを展開します。

```
# tar -xf UUID-commit.tar -C UPGRADE_FOLDER
```

5. OSTree リポジトリーをアップグレードします。

```
# ostree --repo=/usr/share/nginx/html/repo pull-local UPGRADE_FOLDER
# ostree --repo=/usr/share/nginx/html/repo summary -u
```

6. プロビジョニングされた RHEL システム上で、オリジナルの Edge イメージから、現在の状態を確認します。

```
$ rpm-ostree status
```

新しいコミット ID がない場合は、以下のコマンドを実行して、利用可能なアップグレードの有無を確認します。

```
$ rpm-ostree upgrade --check
```

コマンドの出力は、現在アクティブな OSTree コミット ID を提供します。

7. OSTree を更新して、新しい OSTree のコミット ID を利用できるようにします。

```
$ rpm-ostree upgrade
```

OSTree は、リポジトリーに更新があるかどうかを確認します。ある場合は、更新を取得し、この新しいコミット更新のデプロイメントを有効化できるように、システムの再起動を要求します。

8. 再度、現在の状態を確認します。

```
$ rpm-ostree status
```

これで、コミットが 2 つあることが分かります。

- アクティブな親コミット。
- アクティブではなく、差異が 1 つ追加された新しいこミット。

9. 新しいデプロイメントを有効にし、新しいコミットを有効にするには、システムを再起動します。

```
# systemctl reboot
```

Anaconda インストーラーは、新しいデプロイメントで再起動します。ログイン画面では、起動に利用可能な新しいデプロイメントが表示されます。

10. 最新のデプロイメント (コミット) で起動すると、**rpm-ostree** upgrade コマンドは、新しいデプロイメントが一覧内で最初に表示されるようにブートエントリーを自動的に順序を指定します。必要に応じて、キーボードの矢印キーを使用して GRUB メニューエントリーを選択し、**Enter** を押します。

11. ログインユーザーのアカウント認証情報を提供してください。

12. OSTree の状態を確認します。

```
$ rpm-ostree status
```

コマンド出力は、アクティブなコミット ID を提供します。

13. 変更したパッケージが存在する場合は、親コミットと新しいコミットの間で diff を実行します。

```
$ rpm-ostree db diff parent_commit new_commit
```

更新により、インストールしたパッケージが利用可能になり、すぐに使用できる状態になっていることがわかります。

15.2.4. コマンドラインを使用した RHEL for Edge イメージの更新の手動によるデプロイ

RHEL for Edge のブループリントを編集した後、イメージコミットを更新することができます。RHEL Image Builder は、更新された RHEL for Edge イメージの新しいコミットを生成します。新しいコミットを使用して、最新のパッケージバージョンでイメージをデプロイするか、CLI を使用して追加のパッケージをデプロイします。

CLI を使用して RHEL for Edge イメージの更新をデプロイするには、前提条件を満たしていることを確認し、手順に従います。

前提条件

- RHEL for Edge イメージブループリントを作成している。
- RHEL for Edge イメージブループリントを編集している。コマンドラインインターフェイスを使用した RHEL for Edge イメージのブループリントの編集 を参照してください。

手順

1. 以下の引数を使用して、RHEL for Edge Commit (**.tar**) イメージを作成します。

```
# composer-cli compose start-ostree --ref ostree_ref --url URL-OSTree-repository -
blueprint_name_ image-type
```

ここでは、以下のようになります。

- **ref** は、RHEL for Edge Container Commit の作成時に指定したリファレンスです。たとえば、**rhel/9/x86_64/edge** です。
- **URL-OSTree-repository** は、イメージに埋め込むコミットの OSTree リポジトリーへの URL です。たとえば、http://10.0.2.2:8080/repo/ です。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。
- **image-type** は **edge-commit** です。
  RHEL Image Builder が、更新されたブループリントの RHEL for Edge イメージを作成します。

2. RHEL for Edge イメージの作成進捗を確認します。

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

注記

イメージの作成プロセスは、最大 10 分から 30 分かかることがあります。

作成したイメージには、最新パッケージがあれば、そのパッケージが追加され、オリジナルの **commit ID** を親とします。

3. 作成した RHEL for Edge イメージをダウンロードします。詳細は、RHEL Image Builder コマンドラインインターフェイスを使用した RHEL for Edge イメージのダウンロード を参照してください。

4. OSTree コミットを展開します。

```
# tar -xf UUID-commit.tar -C upgrade_folder
```

5. httpd を使用して OSTree コミットを提供します。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。

6. OSTree リポジトリーをアップグレードします。

```
# ostree --repo=/var/www/html/repo pull-local /tmp/ostree-commit/repo
# ostree --repo=/var/www/html/repo summary -u
```

7. プロビジョニングされた RHEL システム上で、オリジナルの Edge イメージから、現在の状態を確認します。

```
$ rpm-ostree status
```

新しいコミット ID がない場合は、以下のコマンドを実行して、利用可能なアップグレードの有無を確認します。

```
$ rpm-ostree upgrade --check
```

コマンドの出力は、現在アクティブな OSTree コミット ID を提供します。

8. OSTree を更新して、新しい OSTree のコミット ID を利用できるようにします。

```
$ rpm-ostree upgrade
```

OSTree は、リポジトリーに更新があるかどうかを確認します。ある場合は、更新を取得し、この新しいコミット更新のデプロイメントを有効化できるように、システムの再起動を要求します。

9. 再度、現在の状態を確認します。

```
$ rpm-ostree status
```

利用可能なコミットが 2 つあることが分かります。

- アクティブな親コミット
- アクティブではなく、1 つの違いを含む新しいコミット

10. 新しいデプロイメントを有効にし、新しいコミットを有効にするには、システムを再起動します。

```
# systemctl reboot
```

Anaconda インストーラーは、新しいデプロイメントで再起動します。ログイン画面では、起動に利用可能な新しいデプロイメントが表示されます。

11. 最新のデプロイメントで起動すると、**rpm-ostree upgrade** コマンドは、新しいデプロイメントがリスト内で最初に表示されるようにブートエントリーを自動的に順序を指定します。必要に応じて、キーボードの矢印キーを使用して GRUB メニューエントリーを選択し、**Enter** を押します。

12. アカウントの認証情報を使用してログインします。

13. OSTree の状態を確認します。

```
$ rpm-ostree status
```

コマンド出力は、アクティブなコミット ID を提供します。

14. 変更したパッケージが存在する場合は、親コミットと新しいコミットの間で diff を実行します。

```
$ rpm-ostree db diff parent_commit new_commit
```

更新により、インストールしたパッケージが利用可能になり、すぐに使用できる状態になっていることがわかります。

15.2.5. ネットワーク以外のデプロイメント向けの RHEL for Edge イメージ更新の手動デプロイ

RHEL for Edge ブループリントを編集した後、それらの更新を使用して RHEL for Edge Commit イメージを更新できます。たとえば、RHEL イメージビルダーを使用して新しいコミットを生成し、VM にすでにデプロイされている RHEL for Edge イメージを更新します。この新しいコミットを使用して、最新のパッケージバージョンまたは追加パッケージのイメージをデプロイしてください。

RHEL for Edge イメージの更新をデプロイするには、前提条件を満たしていることを確認し、手順に従ってください。

前提条件

- ホストで、ブラウザーの Web コンソールから RHEL Image Builder アプリケーションを開いている。
- 稼働中の RHEL for Edge システムがプロビジョニングされています。
- HTTP 経由で提供される OSTree リポジトリーがあります。
- 以前に作成した RHEL for Edge イメージブループリントを編集しました。

手順

1. システムホストの RHEL イメージビルダーダッシュボードで、**Create Image** をクリックします。

2. **Create Image** ウィンドウで、以下の手順を実行します。

   a. **Image output** ページで、以下を実行します。

      i. **Select a blueprint** ドロップダウンリストから、編集したブループリントを選択します。

ii. **Image output type** ドロップダウンリストから、**RHEL for Edge Container (.tar)** を選択します。

iii. **Next** をクリックします。

b. **OSTree settings** ページで、以下を入力します。

i. **Repository URL** フィールドに、イメージに埋め込むコミットの OSTree リポジトリーへの URL を入力します。たとえば、http://10.0.2.2:8080/repo/ です。RHEL for Edge イメージをインストールするための Web サーバーの設定 を参照してください。

ii. **Parent commit** フィールドで、以前に生成された親コミット ID を指定します。RHEL for Edge イメージのコミットの展開 を参照してください。

iii. **Ref** フィールドでは、コミットの名前を指定するか、空欄のままにしておくことができます。デフォルトでは、Web コンソールは **Ref** を **rhel/9/arch_name/edge** に指定します。

iv. **Next** をクリックします。

c. **Review** ページでカスタマイズを確認し、**Create** をクリックします。
RHEL Image Builder が、更新されたブループリントの RHEL for Edge イメージを作成します。

d. **Images** タブをクリックして、RHEL for Edge イメージの作成の進行状況を表示します。

注記

イメージ作成プロセスの完了まで数分かかります。

作成したイメージには、最新パッケージがあれば、そのパッケージが追加され、オリジナルの **commit ID** を親とします。

3. 結果として得られる RHEL for Edge イメージをホストにダウンロードします。

a. **Images** タブで **Download** をクリックして、RHEL for Edge Container (**.tar**) イメージをホストシステムに保存します。

4. 元のエッジイメージからプロビジョニングされた RHEL システムで、次の手順を実行します。

a. 今回は子コミット ID を提供して Podman に RHEL for Edge Container イメージを読み込みます。

```
$ cat ./child-commit_ID-container.tar | sudo podman load
```

b. **Podman** を実行します。

```
#  sudo podman run -p 8080:8080 localhost/edge-test
```

c. OSTree リポジトリーをアップグレードします。

```
# ostree --repo=/var/www/html/repo pull-local /tmp/ostree-commit/repo
# ostree --repo=/var/www/html/repo summary -u
```

d. プロビジョニングされた RHEL システム上で、オリジナルの Edge イメージから、現在の状態を確認します。

```
$ rpm-ostree status
```

新しいコミット ID がない場合は、以下のコマンドを実行して、利用可能なアップグレードの有無を確認します。

```
$ rpm-ostree upgrade --check
```

利用可能な更新がある場合は、コマンドの出力は、現在アクティブな OSTree コミット ID など、OSTree リポジトリーで利用可能な更新に関する情報を提供します。その他に、利用可能な更新がないことをメッセージに通知するよう求められます。

e. OSTree を更新して、新しい OSTree のコミット ID を利用できるようにします。

```
$ rpm-ostree upgrade
```

OSTree は、リポジトリーに更新があるかどうかを確認します。ある場合は、更新を取得し、この新しいコミット更新のデプロイメントを有効化できるように、システムの再起動を要求します。

f. 現在のシステムステータスを確認します。

```
$ rpm-ostree status
```

これで、コミットが 2 つあることが分かります。

- アクティブな親コミット。
- アクティブではなく、差異が 1 つ追加された新しいこミット。

g. 新しいデプロイメントを有効にし、新しいコミットを有効にするには、システムを再起動します。

```
# systemctl reboot
```

Anaconda インストーラーは、新しいデプロイメントで再起動します。ログイン画面では、起動に利用可能な新しいデプロイメントが表示されます。

h. 最新のコミットで起動するには、次のコマンドを実行して、新しいデプロイメントがリストの先頭になるようにブートエントリーを自動的に並べ替えます。

```
$ rpm-ostree upgrade
```

必要に応じて、キーボードの矢印キーを使用して GRUB メニューエントリーを選択し、**Enter** を押します。

5. ログインユーザーのアカウント認証情報を提供してください。

6. OSTree の状態を確認します。

```
$ rpm-ostree status
```

コマンド出力は、アクティブなコミット ID を提供します。

7. 変更したパッケージが存在する場合は、親コミットと新しいコミットの間で diff を実行します。

```
$ rpm-ostree db diff parent_commit new_commit
```

更新により、インストールしたパッケージが利用可能になり、すぐに使用できる状態になっていることがわかります。

15.3. RHEL FOR EDGE のアップグレード

15.3.1. RHEL8 システムを RHEL 9 にアップグレード

rpm-ostree rebase コマンドを使用して、RHEL8 システムを RHEL 9 にアップグレードできます。このコマンドは、RHEL 8 の最新の更新から RHEL 9 の最新の更新までの RHEL for Edge アップグレードのデフォルトパッケージセットを完全にサポートします。アップグレードは、バックグラウンドで RHEL 9 イメージをダウンロードしてインストールします。アップグレードが完了したら、システムを再起動して新しい RHEL 9 イメージを使用する必要があります。

警告

アップグレードは、すべての可能な **rpm** パッケージバージョンおよび付属物をサポートしているわけではありません。パッケージの追加をテストして、これらのパッケージが期待どおりに動作することを確認する必要があります。

前提条件

- RHEL for Edge 8 がある
- HTTP による OSTree リポジトリーサーバーがあります
- アップグレードする RHEL for Edge 9 イメージのブループリントを作成しました

手順

1. RHEL Image Builder が実行されているシステムで、RHEL for Edge 9 イメージを作成します。
   a. イメージの作成を開始します。

```
$ sudo composer-cli compose start blueprint-name edge-commit
```

必要に応じて、次のコマンドで、既存の OSTree リポジトリーを使用して新しい RHEL for Edge 9 イメージを作成することもできます。

```
$ sudo composer-cli compose start-ostree --ref rhel/8/x86_64/edge --parent parent-OSTree-REF --url URL blueprint-name edge-commit
```

   b. 作成が終了したら、イメージをダウンロードします。

   c. ダウンロードしたイメージを **/var/www/html/** フォルダーにデプロイメントします。

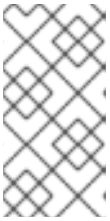
```
$ sudo tar -xf image_file -C /var/www/html
```

d. **httpd** サービスを起動します。

```
$ systemctl start httpd.service
```

2. RHEL for Edge デバイスで、現在のリモートリポジトリーの設定を確認します。

```
$ sudo cat /etc/ostree/remotes.d/edge.conf
```



注記

キックスタートファイルの設定方法によっては、**/etc/ostree/remotes.d** リポジトリーが空になる場合があります。リモートリポジトリーを設定した場合は、その設定を確認できます。**edge-installer**、**raw-image**、および **simple-installer** イメージの場合、デフォルトでリモートが設定されます。

3. 現在の URL リポジトリーを確認します。

```
$ sudo ostree remote show-url edge
```

edge は Ostree リポジトリーです。

4. リモート参照ブランチをリスト表示します。

```
$ ostree remote refs edge
```

次の出力が表示されます。

```
Error: Remote refs not available; server has no summary file
```

5. 新しいリポジトリーを追加するには:

a. リモートリポジトリーを追加するように URL キーを設定します。以下に例を示します。

```
$ sudo ostree remote add \ --no-gpg-verify rhel9 http://192.168.122.1/repo/
```

b. アップグレード用の RHEL 9 コミットを指すように URL キーを設定します。以下に例を示します。

```
$ sudo cat /etc/ostree/remotes.d/edge.conf

[remote "edge"]
url=http://192.168.122.1/ostree/repo/
gpg-verify=false
```

c. URL が新しいリモートリポジトリーに設定されているかどうかを確認します。

```
$ sudo cat /etc/ostree/remotes.d/rhel9.conf

[remote "edge"]
```

```
url=http://192.168.122.1/repo/
gpg-verify=false
```

d. 新しい URL リポジトリーを参照してください。

```
$ sudo ostree remote show-url rhel9 http://192.168.122.1/ostree-rhel9/repo/
```

e. 現在のリモートリストオプションをリスト表示します。

```
$ sudo ostree remote list

output:
edge
rhel9
```

6. システムを RHEL バージョンにリベースし、RHEL 9 バージョンの参照パスを提供します。

```
$ rpm-ostree rebase rhel9:rhel/9/x86_64/edge
```

7. システムを再起動します。

```
$ systemctl reboot
```

8. ユーザー名およびパスワードを入力します。

9. 現在のシステムステータスを確認します。

```
$ rpm-ostree status
```

検証

1. 現在実行中のデプロイメントの現在のステータスを確認します。

```
$ rpm-ostree status
```

2. オプション: カーネルによって管理されているプロセッサーおよびタスクをリアルタイムで一覧表示します。

```
$ top
```

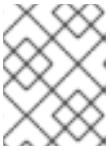
3. アップグレードが要件をサポートしていない場合は、以前の安定したデプロイメント RHEL 8 バージョンに手動でロールバックするオプションがあります。

```
$ sudo rpm-ostree rollback
```

4. システムを再起動します。ユーザー名およびパスワードを入力します。

```
$ systemctl reboot
```

再起動後、システムは RHEL 9 を正常に実行します。



注記

アップグレードが成功し、以前のデプロイメント RHEL 8 バージョンを使用したくない場合は、古いリポジトリーを削除できます。

```
$ sudo ostree remote delete edge
```

関連情報

- rpm-ostree update and rebase fails with failed to find kernel error

15.4. RHEL FOR EDGE の自動イメージ更新のデプロイ

Edge デバイスに RHEL for Edge イメージをインストールした後、利用可能なイメージ更新がある場合は確認し、自動適用することができます。

rpm-ostreed-automatic.service (systemd サービス) と **rpm-ostreed-automatic.timer** (systemd タイマー) は、チェックとアップグレードの頻度を制御します。利用可能な更新がある場合は、ステージングされたデプロイメントとして表示されます。

イメージの自動更新をデプロイするには、以下の概要手順を実施します。

- イメージ更新ポリシーの更新
- 更新の自動ダウンロードとステージングの有効化

15.4.1. RHEL for Edge イメージの更新ポリシーの更新

イメージ更新ポリシーを更新するには、Edge デバイスの **/etc/rpm-ostreed.conf** ロケーションにある **rpm-ostreed.conf** ファイルから **AutomaticUpdatePolicy** および **IdleExitTimeout** の設定を使用します。

AutomaticUpdatePolicy 設定は、自動更新ポリシーを制御し、以下の更新チェックのオプションがあります。

- **none**:自動更新を無効にします。デフォルトでは、**AutomaticUpdatePolicy** 設定は **none** に設定されています。
- **check**:**rpm-ostree** ステータスで利用可能な更新を表示するために十分なメタデータをダウンロードします。
- **stage**:再起動時に適用される更新をダウンロードしてデプロイメントします。

IdleExitTimeout 設定は、デーモンが終了するまでの非活動時間を秒単位で制御するもので、以下のオプションがあります。

- 0:オートイグジットを無効にします。
- 60:デフォルトでは、**IdleExitTimeout**の設定は**60**に設定されています。

自動更新を有効にするには、以下の手順を実行します。

手順

1. **/etc/rpm-ostreed.conf** ファイルで、以下を更新します。

- **AutomaticUpdatePolicy** の値を **check** に変更します。
- 更新チェックを実行するには、**IdleExitTimeout** の値を秒単位で指定します。

2. **rpm-ostreed** サービスを再読み込みし、**systemd** タイマーを有効にします。

```
# systemctl reload rpm-ostreed
# systemctl enable rpm-ostreed-automatic.timer --now
```

3. **rpm-ostree** ステータスを確認し、自動更新ポリシーが設定され、時間がアクティブであることを確認します。

```
# rpm-ostree status
```

コマンド出力は以下を表示します。

```
State: idle; auto updates enabled (check; last run <minutes> ago)
```

さらに、出力には利用可能な更新情報も表示されます。

15.4.2. RHEL for Edge の更新の自動ダウンロードとステージングの有効化

イメージ更新ポリシーを更新してイメージの更新を確認すると、更新がある場合は更新の詳細と一緒に表示されます。更新の適用を決定した場合は、ポリシーを有効にして、更新を自動的にダウンロードしてステージします。続いて、利用可能なイメージの更新がダウンロードされ、デプロイメントのためにステージングされます。更新は、Edge デバイスを再起動すると適用され、有効になります。

更新の自動ダウンロードとステージングのポリシーを有効にするには、以下の更新を実行します。

手順

1. **/etc/rpm-ostreed.conf** ファイルで、AutomaticUpdatePolicy を **stage** に更新します。
2. **rpm-ostreed** サービスを再ロードします。

```
# systemctl enable rpm-ostreed-automatic.timer --now
```

3. **rpm-ostree** のステータスを確認します。

```
# rpm-ostree status
```

コマンド出力は以下を表示します。

```
State: idle
AutomaticUpdates: stage; rpm-ostreed-automatic.timer: last run <time> ago
```

4. 更新を開始するには、タイマーによる更新の開始を待つか、手動でサービスを開始することができます。

```
# systemctl start rpm-ostreed-automatic.service
```

更新が開始すると、**rpm-ostree** のステータスは次のように表示されます。

```
# rpm-ostree status
State: busy
AutomaticUpdates: stage; rpm-ostreed-automatic.service: running
Transaction: automatic (stage)
```

更新が完了すると、デプロイメントのリストに新しいデプロイメントがステージングされ、オリジナルの起動したデプロイメントはそのままになります。新しいデプロイメントを使用してシステムを起動するか、次の更新を待つかを決めることができます。

デプロイメントのリストを表示するには、**rpm-ostree status** コマンドを実行してください。

以下は出力例です。

```
# rpm-ostree status
State: idle
AutomaticUpdates: stage; rpm-ostreed-automatic.timer: last run <time> ago
Deployments:
```

更新されたパッケージの詳細を含むデプロイメントのリストを表示するには、**rpm-ostree status -v** コマンドを実行してください。

15.5. RHEL FOR EDGE イメージのロールバック

RHEL for Edge はオペレーティングシステムにトランザクション更新を適用するため、失敗した更新を最後に確認されている正常な状態に手動または自動でロールバックでき、更新中のシステム障害を防ぐことができます。**greenboot** フレームワークを使用すると、検証とロールバックのプロセスを自動化できます。

greenboot ヘルスチェックフレームワークは、**rpm-ostree** 活用して、システム起動時にカスタムヘルスチェックを実行します。問題が発生した場合、システムは最後の作業状態にロールバックします。**rpm-ostree** 更新をデプロイすると、スクリプトが実行され、重要なサービスが更新後も機能することが確認されます。パッケージの障害などによりシステムが機能しない場合は、システムを以前の安定したバージョンにロールバックできます。このプロセスにより、RHEL for Edge デバイスが確実に動作状態になります。

イメージを更新すると、以前のイメージのデプロイメントを保持しながら、新しいイメージのデプロイメントが作成されます。更新が成功したかどうかを確認できます。パッケージの障害などにより更新が失敗した場合は、システムを以前の安定したバージョンにロールバックできます。

15.5.1. greenboot チェックの概要

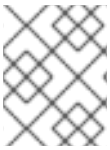
Greenboot は、**rpm-ostree** ベースのシステムで利用できる **systemd** の一般的なヘルスチェックフレームワークです。これには、システムにインストールできる次の RPM パッケージが含まれています。

- **greenboot** - 次の機能を含むパッケージです。
  - 提供されたスクリプトを確認します。
  - チェックが失敗した場合はシステムを再起動します。
  - 再起動しても問題が解決しなかった場合は、以前のデプロイメントにロールバックします。

- **greenboot-default-health-checks** - **greenboot** システムのメンテナーによって提供される、選択された任意のヘルスチェックのセットです。

greenboot は、システム上で実行されるヘルスチェックスクリプトを使用してシステムの正常性を評価し、ソフトウェアに障害が発生した場合には最後の正常な状態へ自動的にロールバックすることにより、RHEL for Edge システムで機能します。これらのヘルスチェックスクリプトは、**/etc/greenboot/check/required.d** ディレクトリーに含まれています。**greenboot** はヘルスチェック用のシェルスクリプトをサポートしています。ヘルスチェックフレームワークは、直接的な保守機能が制限されているか存在しないエッジデバイスでソフトウェアの問題をチェックし、システムロールバックを実行する必要がある場合に特に役立ちます。ヘルスチェックスクリプトをインストールして設定すると、システムが起動するたびにヘルスチェックが実行されます。

独自のヘルスチェックスクリプトを作成して、ワークロードとアプリケーションの正常性を評価できます。これらの追加のヘルスチェックスクリプトは、ソフトウェアの問題チェックと自動システムロールバックの有用なコンポーネントです。



注記

OSTree を使用していないシステムでヘルスチェックが失敗した場合は、ロールバックを使用できません。

15.5.2. greenboot による RHEL for Edge イメージのロールバック

RHEL for Edge イメージでは、トランザクション更新のみがオペレーティングシステムに適用されます。トランザクション更新はアトミックです。つまり、すべての更新が成功した場合にのみ更新が適用されます。また、ロールバックがサポートされます。トランザクション更新では、失敗した更新を最後の既知の良い状態に簡単にロールバックすることができ、更新中のシステム障害を防ぐことができます。

ヘルスチェックの実行は、直接的な保守機能が制限されているか存在しないエッジデバイスでソフトウェアの問題をチェックし、システムロールバックを実行する必要がある場合に特に役立ちます。

注記

OSTree を使用していないシステムで更新が失敗した場合は、ヘルスチェックが実行される可能性がある場合でも、ロールバックを使用できません。

greenboot ヘルスチェックフレームワークによるインテリジェントなロールバックを使用すると、システムが起動するたびにシステムの正常性を自動的に評価できます。事前設定されたヘルスチェックは、**greenboot-default-health-checks** サブパッケージから取得できます。これらのチェックは、**rpm-ostree** システムの **/usr/lib/greenboot/check** 読み取り専用ディレクトリーにあります。

greenboot は **rpm-ostree** を活用し、システム起動時に実行されるカスタムヘルスチェックを実行します。問題が発生した場合、システムは変更をロールバックし、最後の作業状態を保持します。**rpm-ostree** 更新をデプロイすると、スクリプトが実行され、重要なサービスが更新後も機能することが確認されます。システムが機能しない場合、更新はシステムの最新の動作バージョンにロールバックされます。このプロセスにより、RHEL for Edge デバイスが確実に動作状態になります。

事前設定されたヘルスチェックは、**greenboot-default-health-checks` サブパッケージから取得できます。これらのチェックは、rpm-ostree システムの `/usr/lib/greenboot/check** 読み取り専用ディレクトリーにあります。シェルスクリプトを次のタイプのチェックとして設定することもできます。

例15.1 greenboot のディレクトリー構造

```
etc
```

```
└─ greenboot
  ├─ check
  |  └─ required.d
  |  └─ init.py
  └─ green.d
  └─ red.d
```

Required

失敗してはならないヘルスチェックが含まれています。必要なシェルスクリプトを **/etc/greenboot/check/required.d** ディレクトリーに配置します。スクリプトが失敗すると、greenboot はデフォルトでスクリプトを 3 回再試行します。**/etc/greenboot/greenboot.conf** ファイルで再試行回数を設定するには、**GREENBOOT_MAX_BOOTS** パラメーターを目的の再試行回数に設定します。

すべての再試行が失敗すると、ロールバックが利用可能であれば **greenboot** が自動的に開始します。ロールバックが利用できない場合は、手動介入が必要であることをシステムログ出力が示します。

Wanted

失敗してもシステムをロールバックしないヘルスチェックが含まれています。必要なシェルスクリプトを **/etc/greenboot/check/wanted.d** ディレクトリーに配置します。**Greenboot** はスクリプトが失敗したことを通知し、システムの正常性ステータスは影響を受けず、ロールバックも再起動も実行しません。

チェック後に実行するシェルスクリプトを指定することもできます。

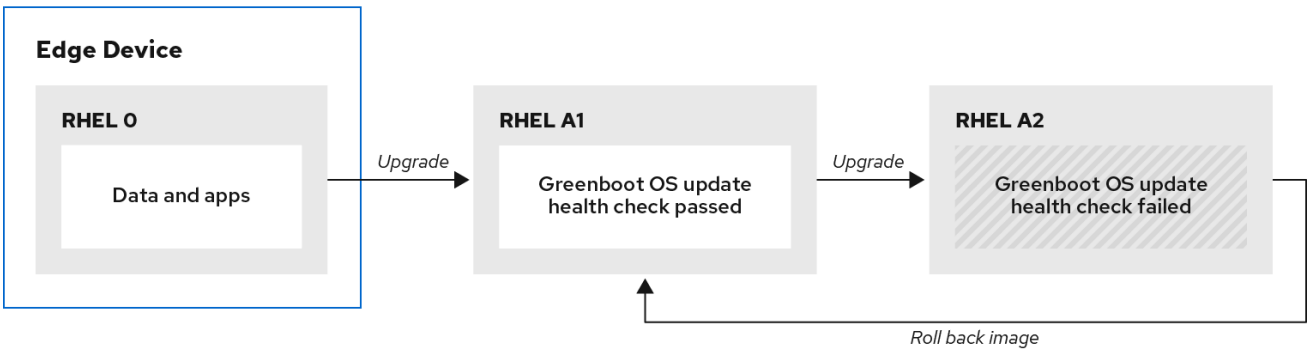
Green

起動が成功した後に実行するスクリプトが含まれています。これらのスクリプトを **/etc/greenboot/green.d** ディレクトリーに配置します。**Greenboot** は起動が成功したことを通知します。

Red

起動に失敗した後に実行するスクリプトが含まれています。この種類のスクリプトは **/etc/greenboot/red.d** ディレクトリーに配置します。システムは起動を 3 回試行し、失敗した場合はスクリプトを実行します。**greenboot** は起動が失敗したことを通知します。

以下の図は、RHEL for Edge イメージのロールバックプロセスを説明しています。



更新されたオペレーティングシステムを起動した後、**greenboot** は **required.d** ディレクトリーと **wanted.d** ディレクトリー内のスクリプトを実行します。**required.d** ディレクトリー内のいずれかのスクリプトが失敗した場合、**greenboot** は **red.d** ディレクトリー内のスクリプトを実行してから、システムを再起動します。

greenboot は、アップグレードしたシステムでの起動をさらに 2 回試行します。3 回目の起動試行中に required.d 内のスクリプトが依然として失敗する場合、**greenboot** は最後に red.d スクリプトを 1 回実行し、red.d ディレクトリー内のスクリプトが問題を修正するための修正アクションを実行しようとして失敗したことを確認します。その後、**greenboot** は、現在の **rpm-ostree** デプロイメントから以前の安定したデプロイメントにシステムをロールバックします。

15.5.3. Greenboot ヘルスチェックのステータス

更新されたシステムをデプロイするときは、greenboot がシステムを以前の状態にロールバックした場合に変更が失われないように、greenboot ヘルスチェックが完了するまで待ってから変更を加えてください。設定を変更したり、アプリケーションをデプロイしたりする場合は、greenboot ヘルスチェックが完了するまで待つ必要があります。これにより、greenboot が **rpm-ostree** システムを以前の状態にロールバックしても、変更が失われなくなります。

greenboot-healthcheck サービスは 1 回実行されて終了します。次のコマンドを使用することで、サービスのステータスをチェックしてサービスが完了したかどうかを確認し、結果を知ることができます。

systemctl is-active greenboot-healthcheck.service

このコマンドは、サービスが終了していれば **active** と報告します。サービスが実行されなかった場合は、**inactive** と表示されます。

systemctl show --property=SubState --value greenboot-healthcheck.service

サービスが完了していれば **exited**、実行中であれば **running** と報告します。

systemctl show --property=Result --value greenboot-healthcheck.service

チェックに合格していれば **success** と報告します。

systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service

サービスの数値終了コードを報告します。0 は成功を、0 以外の値は失敗を意味します。

cat /run/motd.d/boot-status

"Boot Status is GREEN - Health Check SUCCESS" などのメッセージが表示されます。

15.5.4. greenboot ヘルスチェックのステータスの確認

システムに変更を加える前、またはトラブルシューティング中に、greenboot ヘルスチェックのステータスを確認します。次のコマンドのいずれかを使用すると、greenboot スクリプトの実行が完了したことを確認できます。

- ステータスを確認するには、次のオプションのいずれかを使用します。
  - ヘルスチェックステータスのレポートを表示するには、次のように入力します。

    ```
    $ systemctl show --property=SubState --value greenboot-healthcheck.service
    ```

    出力は次のとおりです。

    - **start** は、greenboot チェックがまだ実行中であることを意味します。
    - **exited** は、チェックに合格し、greenboot が終了したことを意味します。Greenboot は、システムが正常な状態の場合、**green.d** ディレクトリー内のスクリプトを実行します。
    - **failed** は、チェックに合格しなかったことを意味します。システムがこの状態にある場合、Greenboot は **red.d** ディレクトリー内のスクリプトを実行し、システムを再起動する可能性があります。

- サービスの数値終了コードを示すレポートを表示するには、次のコマンドを使用します。**0** は成功を、0 以外の値は失敗を意味します。

```
$ systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service
```

- **Boot Status is GREEN - Health Check SUCCESS** など、ブートステータスに関するメッセージを示すレポートを表示するには、次のように入力します。

```
$ cat /run/motd.d/boot-status
```

15.5.5. RHEL for Edge イメージの手動でのロールバック

オペレーティングシステムをアップグレードすると、新しいデプロイメントが作成されます。また、**rpm-ostree** パッケージによって以前のデプロイメントが保持されます。オペレーティングシステムの更新バージョンに問題がある場合は、1 つの **rpm-ostree** コマンドを使用するか、GRUB ブートローダーで以前のデプロイメントを選択することで、以前のデプロイメントに手動でロールバックできます。

以前のバージョンに手動でロールバックするには、次の手順を実行します。

前提条件

1. システムを更新したが、障害が発生した。

手順

1. オプション: 失敗エラーメッセージを確認します。

```
$ journalctl -u greenboot-healthcheck.service.
```

2. **rollback** コマンドを実行します。

```
# rpm-ostree rollback
```

コマンドの出力では、移動中のコミット ID の詳細が確認でき、削除されたパッケージの詳細を含めて、完了したトランザクションが表示されます。

3. システムを再起動します。

```
# systemctl reboot
```

このコマンドは、安定したコンテンツが含まれる 1 つ前のコミットを有効にします。変更が適用され、以前のバージョンが復元されます。

15.5.6. 自動化プロセスを使用した RHEL for Edge イメージのロールバック

greenboot チェックは、起動プロセスに統合されたフレームワークを提供し、ヘルスチェックが失敗した場合は **rpm-ostree** ロールバックをトリガーできます。ヘルスチェックについては、ヘルスチェックの合否を示すカスタムスクリプトを作成できます。結果をもとに、ロールバックをトリガーするタイミングを決めることができます。次の手順では、ヘルスチェックスクリプトの例を作成する方法を示します。

手順

1. 標準の終了コード **0** を返すスクリプトを作成します。
   たとえば、以下のスクリプトにより、設定された DNS サーバーを必ず利用できます。

```
#!/bin/bash

DNS_SERVER=$(grep ^nameserver /etc/resolv.conf | head -n 1 | cut -f2 -d" ")
COUNT=0
# check DNS server is available
ping -c1 $DNS_SERVER
while [ $? != '0' ] && [ $COUNT -lt 10 ]; do
((COUNT++))
echo "Checking for DNS: Attempt $COUNT ."
sleep 10
ping -c 1 $DNS_SERVER
done
```

2. **/etc/greenboot/check/required.d/** でヘルスチェック用の実行ファイルを追加します。

```
chmod +x check-dns.sh
```

   次の再起動時には、システムが **boot-complete.target** ユニットに入る前に、起動プロセスの一環としてスクリプトが実行されます。ヘルスチェックが正常に行われた場合は何もしません。ヘルスチェックに失敗した場合は、システムが数回再起動されてから、更新が失敗したとマークされ、前回の更新にロールバックされます。

検証手順

デフォルトゲートウェイにアクセスできるかどうかを確認するには、以下のヘルスチェックスクリプトを実行します。

1. 標準の終了コード **0** を返すスクリプトを作成します。

```
#!/bin/bash

DEF_GW=$(ip r | awk '/^default/ {print $3}')
SCRIPT=$(basename $0)

count=10
connected=0
ping_timeout=5
interval=5

while [ $count -gt 0 -a $connected -eq 0 ]; do
  echo "$SCRIPT: Pinging default gateway $DEF_GW"
  ping -c 1 -q -W $ping_timeout $DEF_GW > /dev/null 2>&1 && connected=1 || sleep
$interval
  ((--count))
done

if [ $connected -eq 1 ]; then
  echo "$SCRIPT: Default gateway $DEF_GW is reachable."
  exit 0
else
```

```
  echo "$SCRIPT: Failed to ping default gateway $DEF_GW!" 1>&2
  exit 1
fi
```

2. **/etc/greenboot/check/required.d/** ディレクトリーでヘルスチェックの実行ファイルを追加します。

```
chmod +x check-gw.sh
```

第16章 OSTREE イメージの更新の作成と管理

RHEL for Edge システムの OStree イメージの更新を簡単に作成および管理し、RHEL for Edge デバイスですぐに利用可能にすることができます。OSTree では、Image Builder を使用して、RHEL for Edge Commit または RHEL for Edge Container イメージを、OSTree コミットを含む **.tar** ファイルとして作成できます。OSTree 更新バージョン管理システムは、OSTree コミットを保存してバージョン管理する "Git リポジトリー" として機能します。次に、**rpm-ostree** イメージおよびパッケージシステムが、クライアントデバイス上でコミットをアセンブルします。RHEL Image Builder で新しいイメージを作成して更新を実行すると、RHEL Image Builder はこれらのリポジトリーから更新をプルします。

16.1. OSTREE の基本的な概念

イメージの更新時に OSTree と **rpm-ostree** によって使用される基本的な用語を以下に示します。

rpm-ostree

デバイス上で OSTree コミットをアセンブルする方法を処理するエッジデバイス上のテクノロジーです。イメージとパッケージシステムのハイブリッドとして機能します。**rpm-ostree** テクノロジーを使用すると、システムのアトミックなアップグレードとロールバックを行うことができます。

OSTree

OSTree は、コミットの作成と、起動可能なファイルシステムツリーのダウンロードを可能にするテクノロジーです。OSTree を使用してツリーをデプロイし、ブートローダー設定を管理することもできます。

コミット

OSTree コミットには、直接起動できない完全なオペレーティングシステムが含まれています。システムを起動するには、RHEL にインストール可能なイメージなどを使用してコミットをデプロイする必要があります。

参照

ref とも呼ばれます。OSTree ref は名前で、Git ブランチに類似しています。有効な参照名の例を以下に示します。

- **rhel/9/x86_64/edge**
- **ref-name**
- **app/org.gnome.Calculator/x86_64/stable**
- **ref-name-2**

デフォルトでは、Image Builder はパスとして **rhel/9/$ARCH/edge** を指定します。$ARCH の値は、ホストマシンによって決定されます。

Parent

parent 引数は、Image Builder で新しいコミットをビルドするために指定できる OSTree コミットです。**parent** 引数を使用して、ビルドする新しいコミットの親コミットを取得する既存の **ref** を指定できます。解決してプルする ref 値として親コミットを指定する必要があります (例: **rhel/9/x86_64/edge**)。**--parent** コミットは、RHEL for Edge Commit (**.tar**) および RHEL for Edge Container (**.tar**) イメージタイプに使用できます。

Remote

OSTree コンテンツをホストする http または https エンドポイントです。これは yum リポジトリーの baseurl に似ています。

static delta

static delta は、2 つの OSTree コミットの間に生成される更新のコレクションです。これにより、システムクライアントが、サイズの大きな少数のファイルを取得できるようになります。static delta 更新はネットワーク効率を高めます。OStree ベースのホストを更新するときに、システムクライアントがシステム上に存在しない新しい OSTree コミットからのオブジェクトのみを取得するためです。通常、新しい OSTree コミットには多数の小さなファイルが含まれており、複数の TCP 接続が必要です。

summary

summary ファイルは、OSTree リポジトリー内の ref、チェックサム、および利用可能な static delta を列挙する簡潔な方法です。OSTree リポジトリーで、利用可能なすべての ref と static delta の状態を確認できます。ただし、新しい ref、コミット、または static-delta が OSTree リポジトリーに追加されるたびに、summary ファイルを生成する必要があります。

16.2. OSTREE リポジトリーの作成

RHEL for Edge Commit (.tar) または **RHEL for Edge Container (.tar)** イメージタイプを使用して、RHEL Image Builder で OSTree リポジトリーを作成できます。これらのイメージタイプには、単一の OSTree コミットを含む OSTree リポジトリーが含まれています。

- **RHEL for Edge Commit (.tar)** は、Web サーバー上で展開してすぐに提供できます。
- **RHEL for Edge Container (.tar)** は、ローカルのコンテナーイメージストレージにインポートするか、コンテナーレジストリーにプッシュする必要があります。コンテナーを起動すると、統合された **nginx** Web サーバー経由でコミットが提供されます。

Podman を備えた RHEL サーバー上の **RHEL for Edge Container (.tar)** を使用して、OSTree リポジトリーを作成します。

前提条件

- **RHEL for Edge Container (.tar)** イメージを作成している。

手順

1. Image Builder からコンテナーイメージをダウンロードします。

```
$ composer-cli compose image _<UUID>
```

2. コンテナーを Podman にインポートします。

```
$ skopeo copy oci-archive:_<UUID>_-container.tar containers-storage:localhost/ostree
```

3. コンテナーを起動し、ポート **8080** を使用して利用可能にします。

```
$ podman run -rm -p 8080:8080 ostree
```

検証

- コンテナーが実行されていることを確認します。

```
$ podman ps -a
```

16.3. 集中型 OSTREE ミラーの管理

実稼働環境の場合、すべてのコミットを提供する集中型 OSTree ミラーを用意すると、次のようないくつかの利点があります。

- ディスクストレージの重複排除および最小化
- static delta 更新を使用した、クライアントの更新の最適化
- デプロイメント期間中に単一の OSTree ミラーを参照

集中型 OSTree ミラーを管理するには、各コミットを Image Builder から集中型リポジトリーにプルして、そこからコミットをユーザーに提供する必要があります。

注記

osbuild.infra Ansible コレクションを使用して、OSTree ミラーの管理を自動化することもできます。osbuild.infra Ansible を参照してください。

集中型リポジトリーを作成するために、Web サーバー上で次のコマンドを直接実行できます。

手順

1. 空のブループリントを作成し、"rhel-92" をディストリビューションとして使用するようにカスタマイズします。

```
name = "minimal-rhel92"
description = "minimal blueprint for ostree commit"
version = "1.0.0"
modules = []
groups = []
distro = "rhel-92"
```

2. ブループリントをサーバーにプッシュします。

```
# composer-cli blueprints push minimal-rhel92.toml
```

3. 作成したブループリントから RHEL for Edge Commit (**.tar**) イメージをビルドします。

```
# composer-cli compose start-ostree minimal-rhel92 edge-commit
```

4. **.tar** ファイルを取得し、ディスクに解凍します。

```
# composer-cli compose image _<rhel-92-uuid>
$ tar -xf <rhel-92-uuid>.tar -C /usr/share/nginx/html/
```

ディスク上の **/usr/share/nginx/html/repo** が、すべての ref とコミットの一元的な OSTree リポジトリーになります。

5. 別の空のブループリントを作成し、"rhel-87" をディストリビューションとして使用するようにカスタマイズします。

```
name = "minimal-rhel87"
description = "minimal blueprint for ostree commit"
version = "1.0.0"
```

```
modules = []
groups = []
distro = "rhel-87"
```

6. ブループリントをプッシュし、別の RHEL for Edge Commit (**.tar**) イメージを作成します。

```
# composer-cli blueprints push minimal-rhel87.toml
# composer-cli compose start-ostree minimal-rhel87 edge-commit
```

7. **.tar** ファイルを取得し、ディスクに解凍します。

```
# composer-cli compose image <rhel-87-uuid>
$ tar -xf <rhel-87-uuid>.tar
```

8. コミットをローカルリポジトリーにプルします。**ostree pull-local** を使用すると、あるローカルリポジトリーから別のローカルリポジトリーにコミットデータをコピーできます。

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo
```

9. オプション: OSTree リポジトリーのステータスを検査します。出力例を以下に示します。

```
$ ostree --repo=/usr/share/nginx/html/repo refs

rhel/8/x86_64/edge
rhel/9/x86_64/edge

$ ostree --repo=/usr/share/nginx/html/repo show rhel/8/x86_64/edge
commit f7d4d95465fbd875f6358141f39d0c573df6a321627bafde68c73850667e5443
ContentChecksum:
41bf2f8b442a770e9bf03e096a46a286f5836e0a0702b7c3516ef4e0acec2dea
Date:  2023-09-15 16:17:04 +0000
Version: 8.7
(no subject)

$ ostree --repo=/usr/share/nginx/html/repo show rhel/9/x86_64/edge
commit 89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ContentChecksum:
70235bfb9cae82c53f856183750e809becf0b9b076122b19c40fec92fc6d74c1
Date:  2023-09-15 15:30:24 +0000
Version: 9.2
(no subject)
```

10. RHEL 9.2 ブループリントを更新して新しいパッケージを追加し、新しいコミットをビルドします。以下に例を示します。

```
name = "minimal-rhel92"
description = "minimal blueprint for ostree commit"
version = "1.1.0"
modules = []
groups = []
distro = "rhel-92"
```

```
[[packages]]
name = "strace"
version = "*"
```

11. 更新されたブループリントをプッシュし、新しい RHEL for Edge Commit (**.tar**) イメージを作成します。compose が既存の OSTree リポジトリーを参照するようにします。

```
# composer-cli blueprints push minimal-rhel92.toml
# composer-cli compose start-ostree minimal-rhel92 edge-commit --url http://localhost/repo --ref rhel/9/x86_64/edge
```

12. **.tar** ファイルを取得し、ディスクに解凍します。

```
# rm -rf repo
# composer-cli compose image <rhel-92-uuid>
# tar -xf <rhel-92-uuid>.tar
```

13. コミットをリポジトリーにプルします。

```
# ostree --repo=/usr/share/nginx/html/repo pull-local repo
```

14. オプション: OSTree リポジトリーのステータスを再度検査します。

```
$ ostree --repo=/usr/share/nginx/html/repo refs
rhel/8/x86_64/edge
rhel/9/x86_64/edge

$ ostree --repo=/usr/share/nginx/html/repo show rhel/8/x86_64/edge
commit f7d4d95465fbd875f6358141f39d0c573df6a321627bafde68c73850667e5443
ContentChecksum:
41bf2f8b442a770e9bf03e096a46a286f5836e0a0702b7c3516ef4e0acec2dea
Date:  2023-09-15 16:17:04 +0000
Version: 8.7
(no subject)

$ ostree --repo=/usr/share/nginx/html/repo show rhel/9/x86_64/edge
commit a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
Parent:  89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ContentChecksum:
2335930df6551bf7808e49f8b35c45e3aa2a11a6c84d988623fd3f36df42a1f1
Date:  2023-09-15 18:21:31 +0000
Version: 9.2
(no subject)

$ ostree --repo=/usr/share/nginx/html/repo log rhel/9/x86_64/edge
commit a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
Parent:  89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ContentChecksum:
2335930df6551bf7808e49f8b35c45e3aa2a11a6c84d988623fd3f36df42a1f1
Date:  2023-09-15 18:21:31 +0000
Version: 9.2
(no subject)

commit 89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
```

```
ContentChecksum:
70235bfb9cae82c53f856183750e809becf0b9b076122b19c40fec92fc6d74c1
Date:  2023-09-15 15:30:24 +0000
Version: 9.2
(no subject)

$ rpm-ostree db diff --repo=/usr/share/nginx/html/repo
89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
ostree diff commit from:
89290dbfd6f749700c77cbc434c121432defb0c1c367532368eee170d9e53ea9
ostree diff commit to:
a35c3b1a9e731622f32396bb1aa84c73b16bd9b9b423e09d72efaca11b0411c9
Added:
  elfutils-default-yama-scope-0.188-3.el9.noarch
  elfutils-libs-0.188-3.el9.x86_64
  strace-5.18-2.el9.x86_64
```

付録A 用語とコマンド

rpm ostree の用語およびコマンドの詳細をご覧ください。

A.1. OSTREE および RPM-OSTREE の用語

以下は、OSTree および **rpm-ostree** のイメージで使用される便利な用語の一部になります。

表A.1 OSTree および rpm-ostree の用語

用語	定義
OSTree	Linux ベースのオペレーティングシステムのバージョンを管理するために使用するツールです。OSTree のツリービューは Git に似ていて、同様のコンセプトに基づいています。
rpm-ostree	オペレーティングシステムの更新をホストするハイブリッドイメージまたはシステムパッケージ。
Commit	オペレーティングシステムのリリースバージョンまたはイメージバージョン。RHEL Image Builder は、RHEL for Edge イメージの OSTree コミットを生成します。これらのイメージを使用して、Edge サーバーに RHEL をインストールしたり、更新したりすることができます。
Refs	OSTree 内のブランチを表します。Refs は常に最新のコミットに解決します。たとえば、**rhel/9/x86_64/edge** です。
Revision (Rev)	特定のコミットの SHA-256。
Remote	OSTree コンテンツをホストする http または https エンドポイントです。これは、dnf リポジトリーの baseurl と似ています。
static-delta	OSTree イメージの更新は常にデルタ更新です。RHEL for Edge イメージの場合、更新されるファイル数により、TCP のオーバーヘッドが予想以上に大きくなる可能性があります。TCP のオーバーヘッドを回避するために、特定のコミット間で static-delta を生成し、単一の接続で更新を送信することができます。この最適化は、接続性に制約のある大規模なデプロイメントをサポートします。

A.2. OSTREE コマンド

以下の表は、OSTree イメージのインストールまたは管理時に使用できる OSTree コマンドをいくつか示しています。

表A.2 OSTree コマンド

ostree pull	**ostree pull-local --repo [path] src** **ostree pull-local <path> <rev> --repo=<repo-path>** **ostree pull <URL> <rev> --repo=<repo-path>**
ostree summary	**ostree summary -u --repo=<repo-path>**
参照先の表示	**ostree refs --repo ~/Code/src/osbuild-iot/build/repo/ --list**
リポジトリーでのコミットの表示	**ostree log --repo=/home/gicmo/Code/src/osbuild-iot/build/repo/ <REV>**
コミットの検査	**ostree show --repo build/repo <REV>**
リポジトリーのリモートのリスト表示	**ostree remote list --repo <repo-path>**
REV の解決	**ostree rev-parse --repo ~/Code/src/osbuild-iot/build/repo fedora/x86_64/osbuild-demo** **ostree rev-parse --repo ~/Code/src/osbuild-iot/build/repo b3a008eceeddd0cfd**
static-delta の作成	**ostree static-delta generate --repo=[path] --from=REV --to=REV**
GPG 鍵を使用した **既存** の ostree コミットの署名	**ostree gpg-sign --repo=<repo-path> --gpg-homedir <gpg_home> COMMIT KEY-ID…**

A.3. RPM-OSTREE コマンド

以下の表は、OSTree イメージのインストールまたは管理時に使用できる **rpm-ostree** コマンドの一部を示しています。

表A.3 rpm-ostree コマンド

コマンド	説明
rpm-ostree --repo=/home/gicmo/Code/src/osbuild-iot/build/repo/ db list <REV>	このコマンドは、リポジトリーへの <REV> コミットに存在するパッケージをリスト表示します。

コマンド	説明
rpm-ostree rollback	OSTree は、**deployments** と呼ばれるブートローダーエントリーの順序付きリストを管理します。インデックス 0 のエントリーは、デフォルトのブートローダーエントリーです。エントリーごとに個別の **/etc** ディレクトリーがありますが、エントリーはすべて 1 つの **/var** ディレクトリーを共有します。Tab を押して起動を中断すると、ブートローダーを使用してエントリーを選択できます。これにより、以前の状態にロールバックされます。つまり、デフォルトのデプロイメントはデフォルト以外のデプロイメントに変更されます。
rpm-ostree status	このコマンドは、使用中の現在のデプロイメントに関する情報を提供します。起動時にリストで一番上にあるデプロイメントがデフォルトになるなど、利用可能な全デプロイメントの名前と **refspecs** が順番に記載されます。* のマークが付いたデプロイメントは現在の起動されるデプロイメントを、r のマークが付いたものは最新のアップグレードを示します。
rpm-ostree db list	このコマンドを使用して、コミットされたパッケージを確認します。1 つ以上のコミットを指定する必要がありますが、範囲や複数指定したコミットも機能します。
rpm-ostree db diff	このコマンドを使用して、2 つの rev (リビジョン) のツリー間でパッケージがどのように異なるかを表示します。rev を指定しないと、起動したコミットが保留中のコミットと比較されます。rev を 1 つだけ指定すると、起動したコミットはその rev と比較されます。
rpm-ostree upgrade	このコマンドは、現在のツリーの最新バージョンをダウンロードしてデプロイし、現在のツリーを次回の起動のデフォルトとして設定します。これは、実行中のファイルシステムツリーには影響はありません。変更を適用するには、再起動する必要があります。

関連情報

- **rpm-ostree** man ページ

A.4. FDO 自動オンボーディング用語

FDO の用語について

表A.4 FDO の用語

コマンド	説明
FDO	FIDO Device Onboarding
Device	ハードウェア、デバイス、またはコンピューター。
Owner	デバイスの最終的な所有者 - 会社または IT 部門。

コマンド	説明
製造元	デバイスの製造元。
製造元サーバー	そのデバイス向けのデバイス認証情報を作成します。
製造元クライアント	製造サーバーの場所を通知します。
所有者のバウチャー (OV)	個々のデバイスの所有権の記録。 次の情報が含まれています。 * 所有者 (**fdo-owner-onboarding-service**) * ランデブーサーバー - FIDO サーバー (**fdo-rendezvous-server**) * デバイス (少なくとも 1 つの組み合わせ)(**fdo-manufacturing-service**)
デバイス認証情報 (DC)	製造時にデバイスに保存されている主要な認証情報およびランデブー。
鍵	製造サーバーを設定するためのキー * key_path * cert_path * key_type * mfg_string_type: デバイスのシリアル番号 * allowed_key_storage_types:使用しているデバイスの認証に使用されるデータを保護する TPM (Filesystem and Trusted Platform Module)。
ランデブーサーバー	デバイスによって使用され、後でデバイスの所有者を見つけるプロセスで使用されるサーバーへのリンク

関連情報

- FIDO IoT spec

A.5. FDO 自動オンボーディングテクノロジー

以下は、FDO 自動オンボーディングに関連して使用されるテクノロジーです。

表A.5 OSTree および rpm-ostree の用語

Technology	定義
UEFI	Unified Extensible Firmware Interface
RHEL	Red Hat® オペレーティングシステム
rpm-ostree	背景イメージベースのアップグレード。
Greenboot	**rpm-ostree** 上の systemd の **Healthcheck** フレームワーク。
Osbuild	オペレーティングシステムアーティファクト用のパイプラインベースのビルドシステム。
Container	Linux® コンテナーは、システムの他の部分から分離された 1 つ以上のプロセスのセットです。
Coreos-installer	RHEL イメージのインストールを支援し、UEFI でシステムを起動します。
FIDO FDO	設定およびオンボーディングデバイスをプロビジョニングするための仕様プロトコル。