# Red Hat build of MicroShift 4.15

## インストール

MicroShift クラスターのインストールと設定

Last Updated: 2024-07-24

# Red Hat build of MicroShift 4.15 インストール

MicroShift クラスターのインストールと設定

## 法律上の通知



## 概要

このドキュメントでは、MicroShift のインストールに関する情報と、いくつかの設定プロセスの詳細を説明します。

# 目次

# 第1章 RPM パッケージからのインストール

サポート対象のバージョンの Red Hat Enterprise Linux (RHEL) を搭載したマシンに RPM パッケージから MicroShift をインストールできます。

## 1.1. MICROSHIFT をインストールするためのシステム要件

MicroShift をインストールする前に、次の条件を満たす必要があります。

- RHEL または RHEL for Edge の互換性のあるバージョン。
- AArch64 または x86_64 システムアーキテクチャー。
- 2 CPU コア。
- MicroShift の場合は 2 GB RAM、ネットワークベースの HTTP または FTP インストールの場合は RHEL で必要な 3 GB RAM。
- 10 GB のストレージ。
- Red Hat アカウントに有効な MicroShift サブスクリプションがある。サブスクリプションをお持ちでない場合は、営業担当者にお問い合わせください。
- ワークロードの永続ボリューム (PV) に十分な容量を持つ論理ボリュームマネージャー (LVM) ボリュームグループ (VG) がある。

## 1.2. 互換性に関する表

次の互換性に関する表を参照して、サポート対象のバージョンの RHEL for Edge と使用する予定の MicroShift バージョンの組み合わせを検討してください。

### Red Hat Device Edge リリースの互換性に関する表

Red Hat Device Edge の 2 つの製品は、デバイスエッジコンピューティングのための単一のソリューションとして連携して動作します。製品を正しく組み合わせるには、次の表に示すように、それぞれの検証済みリリースを一緒に使用してください。

| RHEL for Edge Version(s) | MicroShift バージョン | MicroShift のリリースステータス | MicroShift のサポート対象の更新 |
|---|---|---|---|
| 9.2、9.3 | 4.15 | 一般提供 | 4.15.0 → 4.15.z および 4.15→ 今後のマイナーバージョン |
| 9.2、9.3 | 4.14 | 一般提供 | 4.14.0 → 4.14.z および 4.14 → 4.15 |
| 9.2 | 4.13 | テクノロジープレビュー | なし |
| 8.7 | 4.12 | 開発者プレビュー | なし |

## 1.3. RPM パッケージからの MICROSHIFT のインストール

MicroShift をインストールする前に、メモリー設定と FIPS モード用にホストマシンを準備することを推奨します。

### 1.3.1. ボリュームグループの設定

MicroShift は、永続ボリューム (PV) にストレージを提供するために、論理ボリュームマネージャーストレージ (LVMS) Container Storage Interface (CSI) プラグインを使用します。LVMS は、Linux 論理ボリュームマネージャー (LVM) を利用して、PV のバッキング論理ボリューム (LV) を動的に管理します。このため、マシンには、LVMS がワークロードの PV 用の LV を作成できる未使用スペースのある LVM ボリュームグループ (VG) が必要です。

LVMS がワークロードの PV 用の LV を作成できるようにボリュームグループ (VG) を設定するには、RHEL のインストール中にルートボリュームの **希望サイズ** を小さくします。ルートボリュームのサイズを下げると、実行時に LVMS によって作成される追加の LV 用の未割り当て領域がディスク上に確保されます。

### 1.3.2. FIPS モードの準備

ユースケースで MicroShift コンテナーを FIPS モードで実行する必要がある場合は、FIPS を有効にして RHEL をインストールする必要があります。ワーカーマシンが FIPS モードで実行するように設定されると、MicroShift コンテナーも FIPS モードで実行するように自動的に設定されます。

**重要**

クラスターが使用するオペレーティングシステムの初回起動時の前に FIPS を有効にする必要があるため、クラスターのデプロイ後に FIPS を有効にすることはできません。

**関連情報**

- MicroShift での FIPS モードの使用

## 1.4. RPM パッケージから MICROSHIFT をインストールする準備

ワークロードの永続ボリューム (PV) に十分な容量を持つ論理ボリュームマネージャー (LVM) ボリュームグループ (VG) を持つように RHEL マシンを設定します。

**前提条件**

- MicroShift をインストールするためのシステム要件が満たされている。
- マシンへの root ユーザーアクセス権がある。
- ワークロードの PV に必要な容量を使用して LVM VG を設定している。

**手順**

1. **Storage Configuration** サブセクションの **Installation Destination** の下にあるグラフィカルインストーラーで、**Custom → Done** を選択して、パーティションとボリュームを設定するためのダイアログを開きます。手動パーティショニングウィンドウが表示されます。
2. **New Red Hat Enterprise Linux 9.x Installation**で、 **Click here to create them automatically** を選択します。

3. ルートパーティション / を選択し、PV 用に十分な容量が VG に指定されるように **必要な容量** を減らし、**Update Settings** をクリックします。

4. インストールを完了します。

**注記**

パーティション設定のその他のオプションは、手動パーティション設定の追加情報セクションにリンクされているガイドを参照してください。

5. root ユーザーとして、次のコマンドを実行して、システムで使用可能な VG 容量を確認します。

```
$ sudo vgs
```

出力例:

```
VG   #PV #LV #SN Attr   VSize    VFree
rhel   1   2   0 wz--n- <127.00g 54.94g
```

**関連情報**

- Red Hat Hybrid Cloud Console から プルシークレット をダウンロードします。
- MicroShift の設定
- パーティション設定のその他のオプションは、手動パーティショニングの設定 を参照してください。
- 既存の LV のサイズを変更して VG の容量を解放する方法の詳細は、LVM ボリュームグループの管理 を参照してください。
- VG および PV の作成に関する詳細は、論理ボリューム管理の概要 を参照してください。

## 1.5. RPM パッケージからの RED HAT BUILD OF MICROSHIFT のインストール

RPM パッケージから Red Hat build of MicroShift をインストールするには、次の手順を使用します。

**前提条件**

- Red Hat build of MicroShift をインストールするためのシステム要件が満たされている。
- RPM パッケージから Red Hat build of MicroShift をインストールする準備の手順が完了している。

**手順**

1. root ユーザーとして、次のコマンドを実行して Red Hat build of MicroShift リポジトリーを有効にします。

```
$ sudo subscription-manager repos \
    --enable rhocp-4.15-for-rhel-9-$(uname -m)-rpms \
    --enable fast-datapath-for-rhel-9-$(uname -m)-rpms
```

2. 次のコマンドを実行して、Red Hat build of MicroShift をインストールします。

```
$ sudo dnf install -y microshift
```

3. インストールのプルシークレットを Red Hat Hybrid Cloud Console から **$HOME/openshift-pull-secret** などの一時フォルダーにダウンロードします。このプルシークレットを使用すると、Red Hat build of MicroShift で使用されるコンテナーイメージを提供するコンテナーレジストリーで認証できます。

4. 次のコマンドを実行して、プルシークレットを RHEL マシンの **/etc/crio** フォルダーにコピーします。

```
$ sudo cp $HOME/openshift-pull-secret /etc/crio/openshift-pull-secret
```

5. 以下のコマンドを実行して、root ユーザーを **/etc/crio/openshift-pull-secret** ファイルの所有者にします。

```
$ sudo chown root:root /etc/crio/openshift-pull-secret
```

6. 以下のコマンドを実行して、root ユーザーのみが **/etc/crio/openshift-pull-secret** ファイルを読み取りおよび書き込み可能にします。

```
$ sudo chmod 600 /etc/crio/openshift-pull-secret
```

7. RHEL マシンでファイアウォールが有効になっている場合は、必須のファイアウォールルールをいくつか設定する必要があります。**firewalld** の場合は、次のコマンドを実行します。

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=10.42.0.0/16
```

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=169.254.169.1
```

```
$ sudo firewall-cmd --reload
```

Red Hat build of MicroShift 用に準備したボリュームグループ (VG) がデフォルト名 **rhel** を使用した場合は、それ以上の設定が必要ありません。別の名前を使用した場合、またはその他の設定を変更する場合は、Red Hat build of MicroShift の設定セクションを参照してください。

### 1.5.1. RPM パッケージからの Operator Lifecycle Manager (OLM) のインストール

MicroShift をインストールする場合、Operator Lifecycle Manager (OLM) パッケージはデフォルトではインストールされません。RPM パッケージを使用して、MicroShift インスタンスに OLM をインストールできます。

**手順**

1. 次のコマンドを実行して、OLM パッケージをインストールします。

```
$ sudo dnf install microshift-olm
```

2. パッケージからアクティブなクラスターにマニフェストを適用するには、次のコマンドを実行します。

```
$ sudo systemctl restart microshift
```

**関連情報**

- MicroShift をインストールするためのシステム要件
- RPM パッケージから MicroShift をインストールする準備
- MicroShift での Operator Lifecycle Manager の使用

### 1.5.2. RPM パッケージからの GitOps Argo CD マニフェストのインストール

MicroShift で OpenShift GitOps の軽量バージョンを使用すると、アプリケーションの管理に役立ちます。RPM パッケージを使用して、必要な Argo CD マニフェストをインストールします。この RPM パッケージには、コア Argo CD を実行する必要なマニフェストが含まれています。

**重要**

このプロセスは、基本的な GitOps 機能をインストールします。Argo CD CLI は現在 MicroShift では利用できません。

**前提条件**

- MicroShift バージョン 4.14 以降がインストールされている。
- 250 MB の追加の RAM ストレージが推奨される。

**手順**

1. 次のコマンドを実行して、サブスクリプションマネージャーで GitOps リポジトリーを有効にします。

```
$ sudo subscription-manager repos --enable=gitops-1.12-for-rhel-9-$(uname -m)-rpms
```

2. 以下のコマンドを実行して GitOps パッケージをインストールします。

```
$ sudo dnf install -y microshift-gitops
```

3. Argo CD Pod をデプロイするには、次のコマンドを実行して MicroShift サービスを再起動します。

```
$ sudo systemctl restart microshift
```

**検証**

- 以下のコマンドを実行して、Pod が適切に実行されていることを確認できます。

```
$ oc get pods -n openshift-gitops
```

**出力例**

```
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          4m11s
argocd-redis-56844446bc-dzmhf       1/1     Running   0          4m12s
argocd-repo-server-57b4f896cf-7qk8l 1/1     Running   0          4m12s
```

## 1.6. MICROSHIFT サービスの開始

次の手順を使用して、MicroShift サービスを開始します。

**前提条件**

- RPM パッケージから MicroShift をインストールしている。

**手順**

1. root ユーザーとして、次のコマンドを入力して MicroShift サービスを開始します。

```
$ sudo systemctl start microshift
```

2. オプション: マシンの起動時に MicroShift を開始するように RHEL マシンを設定するには、次のコマンドを入力します。

```
$ sudo systemctl enable microshift
```

3. オプション: マシンの起動時に MicroShift が自動的に起動しないようにするには、次のコマンドを入力します。

```
$ sudo systemctl disable microshift
```

**注記**

MicroShift サービスは、初回起動時に MicroShift のコンテナーイメージをダウンロードして初期化します。その結果、サービスの初回デプロイ時には MicroShift が起動されるまでに数分かかる場合があります。それ以降は、MicroShift サービスの起動時間が短縮されます。

## 1.7. MICROSHIFT サービスの停止

次の手順を使用して、MicroShift サービスを停止します。

**前提条件**

- MicroShift サービスが実行されている。

**手順**

1. 次のコマンドを入力して、MicroShift サービスを停止します。

```
$ sudo systemctl stop microshift
```

2. MicroShift にデプロイされたワークロードは、MicroShift サービスが停止した後も引き続き実行されます。次のコマンドを入力して、実行中のワークロードを表示します。

```
$ sudo crictl ps -a
```

3. 次のコマンドを入力して、デプロイされたワークロードを停止します。

```
$ sudo systemctl stop kubepods.slice
```

## 1.8. MICROSHIFT クラスターへのアクセス方法

このセクションの手順を使用して、MicroShift サービスを実行している同じマシンから、またはワークステーションからリモートで、MicroShift クラスターにアクセスします。このアクセス権を使用して、ワークロードを監視および管理できます。これらの手順を使用する場合は、接続するホスト名または IP アドレスが含まれる **kubeconfig** ファイルを選択し、関連するディレクトリーに配置します。各手順にリストされているように、クラスターアクティビティーには OpenShift Container Platform CLI ツール (**oc**) を使用します。

**関連情報**

- OpenShift CLI ツールのインストール

### 1.8.1. MicroShift クラスターへのローカルアクセス

以下の手順に従って、**kubeconfig** ファイルを使用して MicroShift クラスターをローカルでアクセスします。

**前提条件**

- **oc** バイナリーがインストールされている。

**手順**

1. オプション: RHEL マシンに ~/**.kube**/ フォルダーがない場合に作成するには、次のコマンドを実行します。

```
$ mkdir -p ~/.kube/
```

2. 次のコマンドを実行して、生成されたローカルアクセス **kubeconfig** ファイルを ~/**.kube**/ ディレクトリーにコピーします。

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. 次のコマンドを実行して、~/**.kube**/**config** ファイルの権限を更新します。

```
$ chmod go-r ~/.kube/config
```

**検証**

- 次のコマンドを入力して、MicroShift が実行されていることを確認します。

```
$ oc get all -A
```

### 1.8.2. MicroShift クラスターへのリモートアクセス用にファイアウォールを開く

リモートユーザーが MicroShift クラスターにアクセスできるように、次の手順を使用してファイアウォールを開きます。この手順は、ワークステーションユーザーがリモートでクラスターにアクセスする前に完了する必要があります。

この手順では、**user@microshift** は、MicroShift ホストマシン上のユーザーであり、別のワークステーション上のリモートユーザーがアクセスできるようにそのマシンをセットアップする責任があります。

**前提条件**

- **oc** バイナリーがインストールされている。
- クラスター管理者の権限がある。

**手順**

- MicroShift ホストの **user@microshift** として、次のコマンドを実行して、Kubernetes API サーバー (**6443/tcp**) のファイアウォールポートを開きます。

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

**検証**

- **user@microshift** として次のコマンドを実行して、MicroShift が入力されていることを確認します。

```
[user@microshift]$ oc get all -A
```

### 1.8.3. MicroShift クラスターへのリモートアクセス

以下の手順に従って、**kubeconfig** ファイルを使用してリモートワークステーションから MicroShift クラスターにアクセスします。

**user@workstation** ログインは、ホストマシンにリモートからアクセスするのに使用されます。手順の **<user>** 値は、**user@workstation** が MicroShift ホストにログインするユーザーの名前になります。

**前提条件**

- **oc** バイナリーがインストールされている。
- **user@microshift** は、ローカルホストからファイアウォールを開いている。

**手順**

1. RHEL マシンに **~/.kube/** フォルダーがない場合は、**user@workstation** として、次のコマンドを実行してフォルダーを作成します。

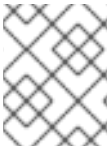
```
[user@workstation]$ mkdir -p ~/.kube/
```

2. **user@workstation** として、次のコマンドを実行して、MicroShift ホストのホスト名の変数を設定します。

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. **user@workstation** として、次のコマンドを実行して、MicroShift を実行している RHEL マシンからローカルマシンに接続するホスト名または IP アドレスを含む生成された **kubeconfig** ファイルをコピーします。

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```



**注記**

この手順で **kubeconfig** ファイルを生成するには、関連資料のセクションの「リモートアクセス用に追加の kubeconfig ファイルの生成」を参照してください。

1. **user@workstation** として、次のコマンドを実行して **~/.kube/config** ファイルのパーミッションを更新します。

```
$ chmod go-r ~/.kube/config
```

**検証**

- **user@workstation** として、次のコマンドを入力して、MicroShift が実行されていることを確認します。

```
[user@workstation]$ oc get all -A
```

# 第2章 MICROSHIFT での FIPS モードの使用

Red Hat Enterprise Linux (RHEL) 9 への MicroShift の RPM ベースのインストールで FIPS モードを使用できます。

- MicroShift コンテナーで FIPS モードを有効にするには、マシンが起動する前に、ワーカーマシンカーネルが FIPS モードで実行できるようにする必要があります。
- Red Hat Enterprise Linux for Edge (RHEL for Edge) イメージでの FIPS の使用はサポートされていません。

## 2.1. RHEL RPM ベースのインストールでの FIPS モード

MicroShift で FIPS を使用するには、Red Hat Enterprise Linux (RHEL) インストールで暗号化モジュールのセルフチェックを有効にする必要があります。ホストオペレーティングシステムが FIPS モジュールで起動するように設定されると、MicroShift コンテナーは FIPS モードで実行できるように自動的に有効になります。

- RHEL が FIPS モードで起動されると、MicroShift コアコンポーネントは、x86_64 アーキテクチャーのみでの FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。
- ワーカーマシンとして使用する予定のマシンに RHEL 9 をインストールする場合は、FIPS モードを有効にする必要があります。

**重要**

クラスターが使用するオペレーティングシステムの初回起動時の前に FIPS を有効にする必要があるため、クラスターのデプロイ後に FIPS を有効にすることはできません。

- MicroShift は、FIPS 互換の Golang コンパイラーを使用します。
- FIPS は CRI-O コンテナーランタイムでサポートされています。

### 2.1.1. 制限事項

- TLS 実装 FIPS サポートは完全ではありません。
- FIPS 実装は、ハッシュ関数を計算し、そのハッシュに基づくキーを検証する単一の機能を提供しません。この制限は、今後の MicroShift リリースでの改善のために引き続き評価されます。

### 2.1.2. FIPS モードでの RHEL のインストール

FIPS を使用して RHEL をインストールするには、RHEL ドキュメントの FIPS モードでのシステムのインストール のガイダンスに従ってください。

## 2.2. 関連情報

- FIPS モードでのシステムのインストール
- コンテナーでの FIPS モードの有効化
- 連邦情報処理標準 140 および FIPS モード

# 第3章 非接続インストール用のコンテナーイメージのミラーリング

MicroShift を非接続ネットワークにデプロイするときに、カスタムコンテナーレジストリーを使用できます。プライベートレジストリー内のミラーリングされたコンテナーイメージのセットからクラスターをインストールすることで、インターネットに直接接続せずに制限されたネットワークでクラスターを実行できます。

## 3.1. コンテナーイメージを既存のレジストリーにミラーリングする

特定のユーザー環境およびワークロード要件では、カスタムのエアギャップコンテナーレジストリー (ミラー) を使用する必要があります。ミラーリングを使用することで、コンテナーイメージと更新をエアギャップ環境に転送し、そこで MicroShift インスタンスにインストールできます。

MicroShift コンテナーのエアギャップミラーレジストリーを作成するには、次の手順を実行する必要があります。

- ミラーリングするコンテナーイメージのリストを取得します。
- ミラーリングの前提条件を設定します。
- インターネットにアクセスできるホスト上のイメージをダウンロードします。
- ダウンロードしたイメージディレクトリーをエアギャップサイトにコピーします。
- エアギャップサイトのミラーレジストリーにイメージをアップロードします。
- ミラーレジストリーを使用するように MicroShift ホストを設定します。

**関連情報**

- Red Hat OpenShift 導入用のミラーレジストリーを使用したミラーレジストリーの作成

## 3.2. ミラーレジストリーコンテナーイメージリストの取得

ミラーレジストリーを使用するには、MicroShift の特定バージョンで使用されているコンテナーイメージ参照を把握しておく必要があります。これらの参照は、**microshift-release-info** RPM パッケージに含まれる **release-<arch>.json** ファイルで提供されます。

**注記**

切断された環境で Operator Lifecycle Manager (OLM) をミラーリングするには、**microshift-olm** RPM に含まれる **release-olm-$ARCH.json** で提供される参照を追加し、同じ手順に従います。Operator カタログと Operator をミラーリングするには **oc-mirror** を使用します。

**前提条件**

- jq がインストールされている。

**手順**

1. 次のいずれかの方法を使用して、コンテナーイメージ参照のリストにアクセスします。

- パッケージが MicroShift ホストにインストールされている場合は、次のコマンドを実行してファイルの場所を取得します。

```
$ rpm -ql microshift-release-info
```

**出力例**

```
/usr/share/microshift/release/release-x86_64.json
```

- パッケージが MicroShift ホストにインストールされていない場合は、次のコマンドを実行して、RPM パッケージをインストールせずにダウンロードして展開します。

```
$ rpm2cpio microshift-release-info*.noarch.rpm | cpio -idmv
```

**出力例**

```
/usr/share/microshift/release/release-x86_64.json
```

2. 次のコマンドを実行して、コンテナーイメージのリストを **microshift-container-refs.txt** ファイルに展開します。

```
$ RELEASE_FILE=/usr/share/microshift/release/release-$(uname -m).json
```

```
$ jq -r '.images | .[]' ${RELEASE_FILE} > microshift-container-refs.txt
```

**注記**

MicroShift コンテナーイメージリストを使用して **microshift-container-refs.txt** ファイルを作成すると、ミラーリング手順を実行する前に、そのファイルに他のユーザー固有のイメージ参照を追加できます。

## 3.3. ミラーリングの前提条件の設定

コンテナーイメージレジストリー認証情報ファイルを作成する必要があります。これにより、インターネットに接続されたミラーホストからエアギャップミラーへのイメージのミラーリングが可能になります。「関連情報」セクションの「イメージのミラーリングを許可する認証情報の設定」に記載された手順を実行します。これは、ミラーにアクセスするためのユーザー認証情報が含まれる **~/.pull-secret-mirror.json** ファイルをミラーレジストリーホスト上に作成するための手順です。

### 3.3.1. ミラーレジストリーのプルシークレットエントリーの例

たとえば次のセクションは、**microshift:microshift** をユーザー名とパスワードとして使用して、**microshift_quay:8443** ミラーレジストリーのプルシークレットファイルに追加されます。

**プルシークレットファイルのミラーレジストリーセクションの例**

```
"<microshift_quay:8443>": { 1
    "auth": "<microshift_auth>", 2
    "email": "<microshift_quay@example.com>" 3
},
```

1. **<registry_host>:<port>** の値の **microshift_quay:8443** を、ミラーレジストリーサーバーのホスト名とポートに置き換えます。
2. **<microshift_auth>** の値をユーザーのパスワードに置き換えます。
3. **</microshift_quay@example.com>** の値をユーザーの電子メールに置き換えます。

**関連情報**

- イメージのミラーリングを可能にする認証情報の設定

## 3.4. コンテナーイメージのダウンロード

コンテナーリストを見つけてミラーリングの前提条件を満たしてから、インターネットアクセスのあるホストにコンテナーイメージをダウンロードします。

**前提条件**

- インターネットにアクセスできるホストにログインしている。
- **.pull-secret-mirror.json** ファイルと **microshift-containers** ディレクトリーの内容がローカルで利用できることを確認する。

**手順**

1. 次のコマンドを実行して、コンテナーイメージのコピーに使用する **skopeo** ツールをインストールします。

```
$ sudo dnf install -y skopeo
```

2. プルシークレットファイルを指す環境変数を設定します。

```
$ PULL_SECRET_FILE=~/.pull-secret-mirror.json
```

3. コンテナーイメージのリストを指す環境変数を設定します。

```
$ IMAGE_LIST_FILE=~/microshift-container-refs.txt
```

4. ダウンロードしたデータを保存する宛先ディレクトリーを指す環境変数を設定します。

```
$ IMAGE_LOCAL_DIR=~/microshift-containers
```

5. 次のスクリプトを実行して、コンテナーイメージを **${IMAGE_LOCAL_DIR}** ディレクトリーにダウンロードします。

```
while read -r src_img ; do
  # Remove the source registry prefix
  dst_img=$(echo "${src_img}" | cut -d '/' -f 2-)

  # Run the image download command
  echo "Downloading '${src_img}' to '${IMAGE_LOCAL_DIR}'"
  mkdir -p "${IMAGE_LOCAL_DIR}/${dst_img}"
```

```
    skopeo copy --all --quiet \
      --preserve-digests \
      --authfile "${PULL_SECRET_FILE}" \
      docker://"${src_img}" dir://"${IMAGE_LOCAL_DIR}/${dst_img}"

done < "${IMAGE_LIST_FILE}"
```

6. イメージセットをターゲット環境 (エアギャップサイトなど) に転送します。その後、イメージセットをミラーレジストリーにアップロードできます。

## 3.5. コンテナーイメージをミラーレジストリーにアップロードする

コンテナーイメージをエアギャップサイトで使用するには、次の手順を実行してミラーレジストリーにコンテナーイメージをアップロードします。

### 前提条件

- **microshift-quay** にアクセスできるホストにログインしている。
- **.pull-secret-mirror.json** ファイルがローカルで使用できる。
- **microshift-containers** ディレクトリーの内容がローカルで使用できる。

### 手順

1. 次のコマンドを実行して、コンテナーイメージのコピーに使用する **skopeo** ツールをインストールします。

```
$ sudo dnf install -y skopeo
```

2. プルシークレットファイルを指す環境変数を設定します。

```
$ IMAGE_PULL_FILE=~/.pull-secret-mirror.json
```

3. ローカルコンテナーイメージディレクトリーを指す環境変数を設定します。

```
$ IMAGE_LOCAL_DIR=~/microshift-containers
```

4. コンテナーイメージをアップロードするためのミラーレジストリー URL を指す環境変数を設定します。

```
$ TARGET_REGISTRY=<registry_host>:<port> 1
```

1 **<registry_host>:<port>** をミラーレジストリーサーバーのホスト名とポートに置き換えます。

5. 次のスクリプトを実行して、コンテナーイメージを **${TARGET_REGISTRY}** ミラーレジストリーにアップロードします。

```
image_tag=mirror-$(date +%y%m%d%H%M%S)
image_cnt=1
  # Uses timestamp and counter as a tag on the target images to avoid
```

```
    # their overwrite by the 'latest' automatic tagging

pushd "${IMAGE_LOCAL_DIR}" >/dev/null
while read -r src_manifest ; do
    # Remove the manifest.json file name
    src_img=$(dirname "${src_manifest}")
    # Add the target registry prefix and remove SHA
    dst_img="${TARGET_REGISTRY}/${src_img}"
    dst_img=$(echo "${dst_img}" | awk -F'@' '{print $1}')

    # Run the image upload command
    echo "Uploading '${src_img}' to '${dst_img}'"
    skopeo copy --all --quiet \
        --preserve-digests \
        --authfile "${IMAGE_PULL_FILE}" \
        dir://"${IMAGE_LOCAL_DIR}/${src_img}" docker://"${dst_img}:${image_tag}-
${image_cnt}"
    # Increment the counter
    (( image_cnt += 1 ))

done < <(find . -type f -name manifest.json -printf '%P\n')
popd >/dev/null
```

## 3.6. ミラーレジストリーアクセス用のホストを設定する

ミラーレジストリーを使用するように MicroShift ホストを設定するには、Red Hat レジストリーのホスト名をミラーにマップする設定ファイルを作成して、MicroShift ホストにレジストリーへのアクセスを許可する必要があります。

**前提条件**

- ミラーホストはインターネットにアクセスできる。
- ミラーホストはミラーレジストリーにアクセスできる。
- 制限されたネットワークで使用するためにミラーレジストリーを設定している。
- プルシークレットをダウンロードし、ミラーリポジトリーへの認証を含めるように変更した。

**手順**

1. MicroShift ホストにログインします。
2. 次の手順を実行して、ミラーレジストリーにアクセスする任意のホストで SSL 証明書の信頼を有効にします。
    a. **rootCA.pem** ファイルを、**<registry_path>/quay-rootCA** などのミラーレジストリーから、**/etc/pki/ca-trust/source/anchors** ディレクトリーにある MicroShift ホストにコピーします。
    b. 次のコマンドを実行して、システム全体のトラストストア設定で証明書を有効にします。

```
$ sudo update-ca-trust
```

3. Red Hat レジストリーのホスト名をミラーレジストリーにマッピングする **/etc/containers/registries.conf.d/999-microshift-mirror.conf** 設定ファイルを作成します。

**ミラー設定ファイルの例**

```
[[registry]]
   prefix = ""
   location = "<registry_host>:<port>" (1)
   mirror-by-digest-only = true
   insecure = false

[[registry]]
   prefix = ""
   location = "quay.io"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false

[[registry]]
   prefix = ""
   location = "registry.redhat.io"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false

[[registry]]
   prefix = ""
   location = "registry.access.redhat.com"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false
```

(1) **<registry_host>:<port>** を、**<microshift-quay:8443>** などのミラーレジストリーサーバーのホスト名とポートに置き換えます。

4. 次のコマンドを実行して、MicroShift サービスを有効にします。

```
$ sudo systemctl enable microshift
```

5. 次のコマンドを実行してホストを再起動します。

```
$ sudo reboot
```

# 第4章 RHEL FOR EDGE イメージへの埋め込み

MicroShift は、Red Hat Enterprise Linux for Edge (RHEL for Edge) イメージに埋め込むことができます。このガイドを使用して、MicroShift を含む RHEL イメージを構築します。

## 4.1. MICROSHIFT をインストールするためのシステム要件

MicroShift をインストールする前に、次の条件を満たす必要があります。

- RHEL または RHEL for Edge の互換性のあるバージョン。
- AArch64 または x86_64 システムアーキテクチャー。
- 2 CPU コア。
- MicroShift の場合は 2 GB RAM、ネットワークベースの HTTP または FTP インストールの場合は RHEL で必要な 3 GB RAM。
- 10 GB のストレージ。
- Red Hat アカウントに有効な MicroShift サブスクリプションがある。サブスクリプションをお持ちでない場合は、営業担当者にお問い合わせください。
- ワークロードの永続ボリューム (PV) に十分な容量を持つ論理ボリュームマネージャー (LVM) ボリュームグループ (VG) がある。

次の表に示すように、サポート対象のバージョンの RHEL を MicroShift のバージョンと組み合わせて使用するように計画してください。

## 4.2. 互換性に関する表

次の互換性に関する表を参照して、サポート対象のバージョンの RHEL for Edge と使用する予定の MicroShift バージョンの組み合わせを検討してください。

### Red Hat Device Edge リリースの互換性に関する表

Red Hat Device Edge の 2 つの製品は、デバイスエッジコンピューティングのための単一のソリューションとして連携して動作します。製品を正しく組み合わせるには、次の表に示すように、それぞれの検証済みリリースを一緒に使用してください。

| RHEL for Edge Version(s) | MicroShift バージョン | MicroShift のリリースステータス | MicroShift のサポート対象の更新 |
|---|---|---|---|
| 9.2、9.3 | 4.15 | 一般提供 | 4.15.0 → 4.15.z および 4.15→ 今後のマイナーバージョン |
| 9.2、9.3 | 4.14 | 一般提供 | 4.14.0 → 4.14.z および 4.14 → 4.15 |
| 9.2 | 4.13 | テクノロジープレビュー | なし |
| 8.7 | 4.12 | 開発者プレビュー | なし |

## 4.3. イメージ構築の準備

RHEL for Edge イメージの作成、インストール、および管理 を参照してください。

特定の CPU アーキテクチャー用の Red Hat Enterprise Linux for Edge (RHEL for Edge) 9.2 イメージをビルドするには、Image Builder のシステム要件 を満たす同じ CPU アーキテクチャーの RHEL 9.2 ビルドホストが必要です。

Image Builder のインストール の手順に従って、Image Builder と **composer-cli** ツールをインストールします。

## 4.4. IMAGE BUILDER への MICROSHIFT リポジトリーの追加

次の手順を使用して、ビルドホスト上の Image Builder に MicroShift リポジトリーを追加します。

**前提条件**

- ビルドホストが Image Builder のシステム要件を満たしている。
- Image Builder と **composer-cli** ツールをインストールしてセットアップしている。
- ビルドホストへの root ユーザーアクセス権がある。

**手順**

1. 次のコマンドを実行して、MicroShift の RPM をプルするために必要な **rhocp-4.15** RPM リポジトリーソースを追加するための Image Builder 設定ファイルを作成します。

```
cat > rhocp-4.15.toml <<EOF
id = "rhocp-4.15"
name = "Red Hat OpenShift Container Platform 4.15 for RHEL 9"
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/rhocp/4.15/os"
check_gpg = true
check_ssl = true
system = false
rhsm = true
EOF
```

2. 次のコマンドを実行して、**fast-datapath** RPM リポジトリーを追加するための Image Builder 設定ファイルを作成します。

```
cat > fast-datapath.toml <<EOF
id = "fast-datapath"
name = "Fast Datapath for RHEL 9"
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/fast-datapath/os"
check_gpg = true
check_ssl = true
system = false
rhsm = true
EOF
```

3. 次のコマンドを実行して、Image Builder にソースを追加します。

```
$ sudo composer-cli sources add rhocp-4.15.toml
```

```
$ sudo composer-cli sources add fast-datapath.toml
```

**検証**

- 以下のコマンドを実行して、ソースが適切に追加されたことを確認します。

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

**出力例**

```
appstream
baseos
fast-datapath
rhocp-4.15
```

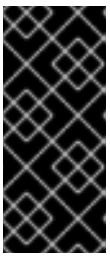
**関連情報**

- Image Builder のシステム要件
- Image Builder のインストール

## 4.5. ブループリントへの MICROSHIFT サービスの追加する

MicroShift RPM パッケージを Image Builder ブループリントに追加すると、MicroShift が埋め込まれた RHEL for Edge イメージをビルドできるようになります。

- 手順 1 から開始して独自の最小限のブループリントファイルを作成すると、MicroShift のインストールが高速化されます。
- 手順 2 から開始して、すべての RPM パッケージとコンテナーイメージを含む、生成されたインストール用のブループリントを使用します。これはインストールプロセスに時間がかかりますが、コンテナー参照がローカルでアクセスされるため、起動は速くなります。



**重要**

- 次の手順の **<microshift_blueprint.toml>** を、使用している TOML ファイルの名前に置き換えます。
- 次の手順の **<microshift_blueprint>** を、ブループリントに使用する名前に置き換えます。

**手順**

1. 次の例を使用して、独自のブループリントファイルを作成します。

**Custom Image Builder ブループリントの例**

```
cat > <microshift_blueprint.toml> <<EOF (1)
name = "<microshift_blueprint>" (2)
```

```
description = ""
version = "0.0.1"
modules = []
groups = []

[[packages]]
name = "microshift"
version = "*"

[customizations.services]
enabled = ["microshift"]
EOF
```

1. **<microshift_blueprint.toml>** は TOML ファイルの名前です。
2. **<microshift_blueprint>** はブループリントの名前です。

**注記**

コマンドのワイルドカード * は、最新の MicroShift RPM を使用します。特定のバージョンが必要な場合は、ワイルドカードを必要なバージョンに置き換えます。たとえば、MicroShift 4.15.0 RPM をダウンロードするには、**4.15.0** を挿入します。

2. オプション: **/usr/share/microshift/blueprint** ディレクトリーにインストールされている、プラットフォームアーキテクチャーに固有のブループリントを使用します。ブループリントセクションの説明については、次のサンプルスニペットを参照してください。

**生成された Image Builder ブループリントのサンプルスニペット**

```
name = "microshift_blueprint"
description = "MicroShift 4.15.1 on x86_64 platform"
version = "0.0.1"
modules = []
groups = []

[[packages]] 1
name = "microshift"
version = "4.15.1"
...
...

[customizations.services] 2
enabled = ["microshift"]

[customizations.firewall]
ports = ["22:tcp", "80:tcp", "443:tcp", "5353:udp", "6443:tcp", "30000-32767:tcp", "30000-
32767:udp"]
...
...

[[containers]] 3
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:f41e79c17e8b41f1b0a5a32c3e2dd7cd15b8274554d3f1ba12b2598a347475f4"
```

```
[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:dbc65f1fba7d92b36cf7514cd130fe83a9bd211005ddb23a8dc479e0eea645fd"
...
…
EOF
```

1. **microshift-release-info** RPM と互換性のある同じバージョンを使用する、オプション以外のすべての MicroShift RPM パッケージの参照。
2. システム起動時に MicroShift を自動的に有効にし、デフォルトのネットワーク設定を適用するための参照。
3. オフラインデプロイメントに必要なオプション以外のすべての MicroShift コンテナーイメージの参照。

3. 次のコマンドを実行して、Image Builder にブループリントを追加します。

```
$ sudo composer-cli blueprints push <microshift_blueprint.toml> 1
```

1. **<microshift_blueprint.toml>** をTOML ファイルの名前に置き換えます。

**検証**

1. 次のコマンドを実行して、MicroShift パッケージのみをリストした Image Builder 設定を確認します。

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> | grep microshift 1
```

1. **<microshift_blueprint>** をブループリントの名前に置き換えます。

**出力例**

```
blueprint: microshift_blueprint v0.0.1
    microshift-greenboot-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-networking-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-release-info-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-selinux-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
```

2. オプション: 次のコマンドを実行して、インストールするすべてのコンポーネントをリストした Image Builder 設定を確認します。

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> 1
```

1. **<microshift_blueprint>** をブループリントの名前に置き換えます。

### 4.5.1. Operator Lifecycle Manager (OLM) サービスのブループリントへの追加

MicroShift をインストールする場合、Operator Lifecycle Manager (OLM) パッケージはデフォルトではインストールされません。ostree ブループリントに **microshift-olm** パッケージを追加して、MicroShift で OLM を有効にすることができます。

1. 次のコマンド例を実行して、ostree ブループリントを編集します。

```
$ vi <microshift_blueprint.toml> 1
```

1 MicroShift サービスを追加するときに使用した、ブループリントファイルの名前を指定します。

2. 次のサンプルテキストを ostree ブループリントに追加します。

```
[[packages]]
name = "microshift-olm"
version = "*"
```

3. パッケージからアクティブなクラスターにマニフェストを適用するには、新しい OSTree システムを構築して、それをマシンにデプロイする必要があります。OSTree システムを更新するには、"OSTree システムへの更新の適用 " の手順に従ってください。

**関連情報**

- MicroShift での Operator Lifecycle Manager の使用
- OSTree システムでの更新の適用

## 4.6. 認証局バンドルの追加

MicroShift は、クライアントがサーバー証明書を評価するときにホスト信頼バンドルを使用します。カスタマイズされたセキュリティー証明書チェーンを使用して、デプロイメント固有のクライアントとエンドポイント証明書の互換性を向上させることもできます。これを行うには、ルート証明書と中間証明書を含む認証局 (CA) バンドルを Red Hat Enterprise Linux for Edge (RHEL for Edge) システム全体の信頼ストアに追加できます。

### 4.6.1. rpm-ostree イメージへの認証局バンドルの追加

イメージの作成に使用するブループリントに別の信頼できる認証局 (CA) を追加することで、Red Hat Enterprise Linux for Edge (RHEL for Edge) **rpm-ostree** イメージに追加の信頼できる認証局 (CA) を含めることができます。次の手順を使用すると、イメージレジストリーからイメージを取得するときにオペレーティングシステムによって信頼される別の CA が設定されます。

**注記**

この手順では、ブループリントで CA バンドルのカスタマイズを設定してから、キックスタートファイルに手順を追加してバンドルを有効にする必要があります。次の手順では、**data** がキーで、**<value>** は PEM エンコードされた証明書を表します。

**前提条件**

- ビルドホストへの root ユーザーアクセス権がある。
- ビルドホストが Image Builder のシステム要件を満たしている。

- Image Builder と **composer-cli** ツールをインストールしてセットアップしている。

**手順**

1. 次のカスタム値をブループリントに追加して、ディレクトリーを追加します。

   a. イメージがビルドされるホスト上のブループリントに指示を追加して、証明書バンドル用のディレクトリー (**/etc/pki/ca-trust/source/anchors/** など) を作成します。

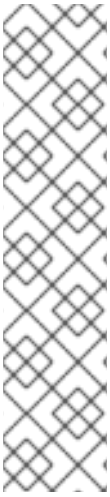
   ```
   [[customizations.directories]]
   path = "/etc/pki/ca-trust/source/anchors"
   ```

   b. イメージが起動したら、証明書バンドル (**/etc/pki/ca-trust/source/anchors/cert1.pem** など) を作成します。

   ```
   [[customizations.files]]
   path = "/etc/pki/ca-trust/source/anchors/cert1.pem"
   data = "<value>"
   ```

2. システム全体のトラストストア設定で証明書バンドルを有効にするには、以下のように、使用しているイメージが起動されているホストで **update-ca-trust** コマンドを使用します。

   ```
   $ sudo update-ca-trust
   ```



**注記**

**update-ca-trust** コマンドは、MicroShift ホストのインストールに使用されるキックスタートファイルの **%post** セクションに含まれている場合があります。この設定により、最初の起動時に必要なすべての証明書の信頼が有効になります。キックスタートファイルに手順を追加してバンドルを有効にする前に、ブループリントで CA バンドルのカスタマイズを設定する必要があります。

```
%post
# Update certificate trust storage in case new certificates were
# installed at /etc/pki/ca-trust/source/anchors directory
update-ca-trust
%end
```

**関連情報**

- RHEL for Edge イメージの作成
- 共有システム証明書の使用 (RHEL 9)
- サポートされているイメージのカスタマイズ (RHEL 9)

## 4.7. RHEL FOR EDGE イメージの作成

ISO を作成するには、以下の手順を使用します。RHEL for Edge Installer イメージは、実行中のコンテナーからコミットをプルし、埋め込みの **rpm-ostree** コミット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持つ、インストール可能なブート ISO を作成します。

**前提条件**

- ビルドホストが Image Builder のシステム要件を満たしている。
- Image Builder と **composer-cli** ツールをインストールしてセットアップしている。
- ビルドホストへの root ユーザーアクセス権がある。
- **podman** ツールがインストールされている。

**手順**

1. 以下のコマンドを実行して **ostree** コンテナーイメージビルドを開始します。

```
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/{op-system-version-major}/$(uname -m)/edge" <microshift_blueprint> edge-container | awk '{print $2}') 1
```

1 <microshift_blueprint> をブループリントの名前に置き換えます。

このコマンドは、監視対象のビルドの ID (ID) も返します。

2. 次のコマンドを実行して、ビルドのステータスを定期的に確認できます。

```
$ sudo composer-cli compose status
```

**実行中のビルドの出力例**

```
ID                                   Status    Time                 Blueprint             Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332   RUNNING   Wed Jun 7 12:26:23 2023
microshift_blueprint      0.0.1     edge-container
```

**完了したビルドの出力例**

```
ID                                   Status    Time                 Blueprint           Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332   FINISHED   Wed Jun 7 12:32:37 2023
microshift_blueprint   0.0.1     edge-container
```

**注記**

起動および停止方法を理解している場合は、**watch** コマンドを使用してビルドを監視できます。

3. ID を使用してコンテナーイメージをダウンロードし、次のコマンドを実行して、使用可能なイメージを取得します。

```
$ sudo composer-cli compose image ${BUILDID}
```

4. 次のコマンドを実行して、ダウンロードしたコンテナーイメージの所有権を現在のユーザーに変更します。

```
$ sudo chown $(whoami). ${BUILDID}-container.tar
```

5. 次のコマンドを実行して、現在のユーザーの読み取り権限をイメージに追加します。

```
$ sudo chmod a+r ${BUILDID}-container.tar
```

6. 以下の手順を実行して、**ostree** コンテナーイメージが ISO ビルドで使用されるようにポート 8085 でサーバーをブートストラップします。

   a. 次のコマンドを実行して、**IMAGEID** 変数の結果を取得します。

```
$ IMAGEID=$(cat < "./${BUILDID}-container.tar" | sudo podman load | grep -o -P '(?
<=sha256[@:])[a-z0-9]*')
```

   b. **IMAGEID** 変数の結果をもとに、以下のコマンドを実行して podman コマンドを実行します。

```
$ sudo podman run -d --name=minimal-microshift-server -p 8085:8080 ${IMAGEID}
```

   このコマンドは、監視用に **IMAGEID** 変数に保存されているコンテナーの ID も返します。

7. 次のコマンドを実行して、インストーラーブループリントファイルを生成します。

```
cat > microshift-installer.toml <<EOF
name = "microshift-installer"

description = ""
version = "0.0.0"
modules = []
groups = []
packages = []
EOF
```

## 4.8. IMAGE BUILDER へのブループリントの追加および ISOの 構築

1. 次のコマンドを実行して、Image Builder にブループリントを追加します。

```
$ sudo composer-cli blueprints push microshift-installer.toml
```

2. 以下のコマンドを実行して **ostree** ISO ビルドを開始します。

```
$ BUILDID=$(sudo composer-cli compose start-ostree --url http://localhost:8085/repo/ --ref
"rhel/9/$(uname -m)/edge" microshift-installer edge-installer | awk '{print $2}')
```

   このコマンドは、監視対象のビルドの ID (ID) も返します。

3. 次のコマンドを実行して、ビルドのステータスを定期的に確認できます。

```
$ sudo composer-cli compose status
```

**実行中のビルドの出力例**

```
ID                                   Status    Time                 Blueprint          Version   Type
Size
c793c24f-ca2c-4c79-b5b7-ba36f5078e8d  RUNNING   Wed Jun 7 13:22:20 2023
```

```
microshift-installer   0.0.0     edge-installer
```

**完了したビルドの出力例**

```
ID                                     Status    Time                  Blueprint           Version   Type
Size
c793c24f-ca2c-4c79-b5b7-ba36f5078e8d   FINISHED  Wed Jun 7 13:34:49 2023
microshift-installer   0.0.0     edge-installer
```

## 4.9. ISO のダウンロードおよび使用準備

1. 以下のコマンドを実行して、ID を使用して ISO をダウンロードします。

```
$ sudo composer-cli compose image ${BUILDID}
```

2. 次のコマンドを実行して、ダウンロードしたコンテナーイメージの所有権を現在のユーザーに変更します。

```
$ sudo chown $(whoami). ${BUILDID}-installer.iso
```

3. 次のコマンドを実行して、現在のユーザーの読み取り権限をイメージに追加します。

```
$ sudo chmod a+r ${BUILDID}-installer.iso
```

**関連情報**

- Image Builder CLI を使用した Edge Container のブループリントの作成
- サポートされているイメージのカスタマイズ
- Building OSTree image
- Blueprint Reference
- Podman のインストール

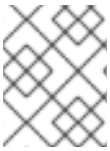
## 4.10. MICROSHIFT 用のマシンのプロビジョニング

RHEL for Edge ドキュメントの手順を使用して、RHEL for Edge イメージでマシンをプロビジョニングします。

MicroShift を使用するには、次の要件を満たすようにシステムをプロビジョニングする必要があります。

- プロビジョニングするマシンは、MicroShift をインストールするためのシステム要件を満たしている必要があります。
- ファイルシステムに、ワークロードの永続ボリューム (PV) に十分な容量を持つ論理ボリュームマネージャー (LVM) ボリュームグループ (VG) がある。
- Red Hat Hybrid Cloud Console からのプルシークレットが **/etc/crio/openshift-pull-secret** として存在し、root ユーザーのみの読み取り/書き込みパーミッションがある。

- ファイアウォールに必要な設定が行われている。



**注記**

RHEL for Edge Installer (ISO) イメージなどのキックスタートを使用している場合は、上記の要件を満たすようにキックスタートファイルを更新できます。

**前提条件**

1. RHEL for Edge Commit を含む RHEL for Edge Installer (ISO) イメージを Red Hat build of MicroShift で作成している。
    a. この要件には、RFE コンテナーイメージの作成、RFE インストーラブループリントの作成、RFE コンテナーの起動、および RFE インストーライメージの作成の手順が含まれます。
2. キックスタートファイルを作成しているか、既存のファイルを使用している。キックスタートファイルには、以下を含める必要があります。
    a. ユーザーの作成方法に関する詳細な手順
    b. RHEL for Edge イメージをフェッチしてデプロイする方法

詳細は、追加リソースを参照してください。

**手順**

1. キックスタートファイルのメインセクションで、システムルートに少なくとも 10GB が指定されている **rhel** という名前の LVM ボリュームグループが含まれるように、ファイルシステムのセットアップを更新します。LVMS CSI ドライバーがワークロードのデータを格納するために使用する空き領域を残します。

**ファイルシステムを設定するためのキックスタートスニペットの例**

```
# Partition disk such that it contains an LVM volume group called `rhel` with a
# 10GB+ system root but leaving free space for the LVMS CSI driver for storing data.
#
# For example, a 20GB disk would be partitioned in the following way:
#
# NAME          MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
# sda             8:0    0  20G  0 disk
# ├─sda1          8:1    0 200M  0 part /boot/efi
# ├─sda1          8:1    0 800M  0 part /boot
# └─sda2          8:2    0  19G  0 part
#   └─rhel-root  253:0    0  10G  0 lvm  /sysroot
#
ostreesetup --nogpg --osname=rhel --remote=edge \
--url=file:///run/install/repo/ostree/repo --ref=rhel/<RHEL VERSION NUMBER>/x86_64/edge
zerombr
clearpart --all --initlabel
part /boot/efi --fstype=efi --size=200
part /boot --fstype=xfs --asprimary --size=800
# Uncomment this line to add a SWAP partition of the recommended size
#part swap --fstype=swap --recommended
part pv.01 --grow
volgroup rhel pv.01
```

```
logvol / --vgname=rhel --fstype=xfs --size=10000 --name=root
# To add users, use a line such as the following
user --name=<YOUR_USER_NAME> \
--password=<YOUR_HASHED_PASSWORD> \
--iscrypted --groups=<YOUR_USER_GROUPS>
```

2. キックスタートファイルの **%post** セクションで、プルシークレットと必須のファイアウォールルールを追加します。

**プルシークレットとファイアウォールルールを追加するためのキックスタートスニペットの例**

```
%post --log=/var/log/anaconda/post-install.log --erroronfail

# Add the pull secret to CRI-O and set root user-only read/write permissions
cat > /etc/crio/openshift-pull-secret << EOF
YOUR_OPENSHIFT_PULL_SECRET_HERE
EOF
chmod 600 /etc/crio/openshift-pull-secret

# Configure the firewall with the mandatory rules for MicroShift
firewall-offline-cmd --zone=trusted --add-source=10.42.0.0/16
firewall-offline-cmd --zone=trusted --add-source=169.254.169.1

%end
```

3. 次のコマンドを実行して、**mkksiso** ツールをインストールします。

```
$ sudo yum install -y lorax
```

4. 次のコマンドを実行して、ISO のキックスタートファイルを新しいキックスタートファイルで更新します。

```
$ sudo mkksiso <your_kickstart>.ks <your_installer>.iso <updated_installer>.iso
```

**関連情報**

- RHEL for Edge のドキュメント
- MicroShift をインストールするためのシステム要件
- Red Hat Hybrid Cloud Console pull secret
- 必要なファイアウォール設定
- キックスタートファイルの作成
- キックスタートファイルを ISO イメージに埋め込む方法

## 4.11. MICROSHIFT クラスターへのアクセス方法

このセクションの手順を使用して、MicroShift サービスを実行している同じマシンから、またはワークステーションからリモートで、MicroShift クラスターにアクセスします。このアクセス権を使用して、ワークロードを監視および管理できます。これらの手順を使用する場合は、接続するホスト名または IP

アドレスが含まれる **kubeconfig** ファイルを選択し、関連するディレクトリーに配置します。各手順にリストされているように、クラスターアクティビティーには OpenShift Container Platform CLI ツール (**oc**) を使用します。

### 4.11.1. MicroShift クラスターへのローカルアクセス

以下の手順に従って、**kubeconfig** ファイルを使用して MicroShift クラスターをローカルでアクセスします。

**前提条件**

- **oc** バイナリーがインストールされている。

**手順**

1. オプション: RHEL マシンに ~/**.kube**/ フォルダーがない場合に作成するには、次のコマンドを実行します。

```
$ mkdir -p ~/.kube/
```

2. 次のコマンドを実行して、生成されたローカルアクセス **kubeconfig** ファイルを ~/**.kube**/ ディレクトリーにコピーします。

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. 次のコマンドを実行して、~/**.kube**/**config** ファイルの権限を更新します。

```
$ chmod go-r ~/.kube/config
```

**検証**

- 次のコマンドを入力して、MicroShift が実行されていることを確認します。

```
$ oc get all -A
```

### 4.11.2. MicroShift クラスターへのリモートアクセス用にファイアウォールを開く

リモートユーザーが MicroShift クラスターにアクセスできるように、次の手順を使用してファイアウォールを開きます。この手順は、ワークステーションユーザーがリモートでクラスターにアクセスする前に完了する必要があります。

この手順では、**user@microshift** は、MicroShift ホストマシン上のユーザーであり、別のワークステーション上のリモートユーザーがアクセスできるようにそのマシンをセットアップする責任があります。

**前提条件**

- **oc** バイナリーがインストールされている。
- クラスター管理者の権限がある。

**手順**

- MicroShift ホストの **user@microshift** として、次のコマンドを実行して、Kubernetes API サーバー (**6443/tcp**) のファイアウォールポートを開きます。

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

**検証**

- **user@microshift** として次のコマンドを実行して、MicroShift が入力されていることを確認します。

```
[user@microshift]$ oc get all -A
```

### 4.11.3. MicroShift クラスターへのリモートアクセス

以下の手順に従って、**kubeconfig** ファイルを使用してリモートワークステーションから MicroShift クラスターにアクセスします。

**user@workstation** ログインは、ホストマシンにリモートからアクセスするのに使用されます。手順の **<user>** 値は、**user@workstation** が MicroShift ホストにログインするユーザーの名前になります。

**前提条件**

- **oc** バイナリーがインストールされている。
- **user@microshift** は、ローカルホストからファイアウォールを開いている。

**手順**

1. RHEL マシンに **~/.kube/** フォルダーがない場合は、**user@workstation** として、次のコマンドを実行してフォルダーを作成します。

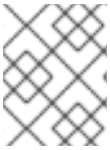
```
[user@workstation]$ mkdir -p ~/.kube/
```

2. **user@workstation** として、次のコマンドを実行して、MicroShift ホストのホスト名の変数を設定します。

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. **user@workstation** として、次のコマンドを実行して、MicroShift を実行している RHEL マシンからローカルマシンに接続するホスト名または IP アドレスを含む生成された **kubeconfig** ファイルをコピーします。

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```



**注記**

この手順で **kubeconfig** ファイルを生成するには、関連資料のセクションの「リモートアクセス用に追加の kubeconfig ファイルの生成」を参照してください。

1. **user@workstation** として、次のコマンドを実行して ~/**.kube**/**config** ファイルのパーミッションを更新します。

```
$ chmod go-r ~/.kube/config
```

**検証**

- **user@workstation** として、次のコマンドを入力して、MicroShift が実行されていることを確認します。

```
[user@workstation]$ oc get all -A
```

**関連情報**

- リモートアクセス用の追加の kubeconfig ファイルの生成

# 第5章 オフラインで使用するための RHEL FOR EDGE イメージへの埋め込み

**rpm-ostree** コミットに MicroShift コンテナーを埋め込むと、エアギャップ環境、非接続環境、またはオフライン環境でクラスターを実行できます。Red Hat Enterprise Linux for Edge (RHEL for Edge) イメージに Red Hat build of MicroShift コンテナーを埋め込むことで、コンテナーエンジンがネットワーク経由でコンテナーレジストリーからイメージをプルする必要がなくなります。ネットワーク接続がなくても、ワークロードをすぐに起動できます。

## 5.1. MICROSHIFT をインストールするためのシステム要件

MicroShift をインストールする前に、次の条件を満たす必要があります。

- RHEL または RHEL for Edge の互換性のあるバージョン。
- AArch64 または x86_64 システムアーキテクチャー。
- 2 CPU コア。
- MicroShift の場合は 2 GB RAM、ネットワークベースの HTTP または FTP インストールの場合は RHEL で必要な 3 GB RAM。
- 10 GB のストレージ。
- Red Hat アカウントに有効な MicroShift サブスクリプションがある。サブスクリプションをお持ちでない場合は、営業担当者にお問い合わせください。
- ワークロードの永続ボリューム (PV) に十分な容量を持つ論理ボリュームマネージャー (LVM) ボリュームグループ (VG) がある。

## 5.2. 互換性に関する表

次の互換性に関する表を参照して、サポート対象のバージョンの RHEL for Edge と使用する予定の MicroShift バージョンの組み合わせを検討してください。

### Red Hat Device Edge リリースの互換性に関する表

Red Hat Device Edge の 2 つの製品は、デバイスエッジコンピューティングのための単一のソリューションとして連携して動作します。製品を正しく組み合わせるには、次の表に示すように、それぞれの検証済みリリースを一緒に使用してください。

| RHEL for Edge Version(s) | MicroShift バージョン | MicroShift のリリースステータス | MicroShift のサポート対象の更新 |
|---|---|---|---|
| 9.2、9.3 | 4.15 | 一般提供 | 4.15.0 → 4.15.z および 4.15→ 今後のマイナーバージョン |
| 9.2、9.3 | 4.14 | 一般提供 | 4.14.0 → 4.14.z および 4.14 → 4.15 |
| 9.2 | 4.13 | テクノロジープレビュー | なし |

| | | | |
|---|---|---|---|
| 8.7 | 4.12 | 開発者プレビュー | なし |

## 5.3. オフラインデプロイ用の MICROSHIFT コンテナーの埋め込み

Image Builder を使用して、MicroShift コンテナーイメージが埋め込まれた **rpm-ostree** システムイメージを作成できます。コンテナーイメージを埋め込むには、Image Builder ブループリントにイメージ参照を追加する必要があります。

**前提条件**

- ビルドホストへの root ユーザーアクセス権がある。
- ビルドホストが Image Builder のシステム要件を満たしている。
- Image Builder と **composer-cli** ツールをインストールしてセットアップしている。
- RHEL for Edge イメージのブループリントを作成済みである。
- jq がインストールされている。

**手順**

1. デプロイする MicroShift バージョンで使用されるコンテナーイメージ参照の正確なリストを取得します。手順 2 に従って **microshift-release-info** RPM パッケージをインストールするか、手順 3 に従って RPM をダウンロードして展開します。
2. **microshift-release-info** RPM パッケージをインストールするには、以下を実行します。
   a. 次のコマンドを実行して、**microshift-release-info** RPM パッケージをインストールします。

      ```
      $ sudo dnf install -y microshift-release-info-<release_version>
      ```

      **<release_version>** は、デプロイするリリースの暗号き換えます。完全なバージョン番号 (**4.15.0** など) を使用してください。

   b. 次のコマンドを実行して、**/usr/share/microshift/release** ディレクトリーの内容をリスト表示し、リリース情報ファイルの存在を確認します。

      ```
      $ ls /usr/share/microshift/release
      ```

      **出力例**

      ```
      release-x86_64.json
      release-aarch64.json
      ```

      **microshift-release-info** RPM をインストールした場合は、手順 4 に進むことができます。
3. 手順 2 を完了していない場合は、**microshift-release-info** RPM をインストールせずにダウンロードして展開します。
   a. 次のコマンドを実行して、RPM パッケージをダウンロードします。

      -

```
$ sudo dnf download microshift-release-info-<release_version>
```

**<release_version>** は、デプロイするリリースの暗号き換えます。完全なバージョン番号 (**4.15.0** など) を使用してください。

**rpm の例**

```
microshift-release-info-4.15.0.*.el9.noarch.rpm 1
```

1 * は日付とコミット ID を表します。両方が出力に含まれているはずです (例: **-202311101230.p0.g7dc6a00.assembly.4.15.0**)。

b. 次のコマンドを実行して、RPM パッケージをインストールせずに展開します。

```
$ rpm2cpio <my_microshift_release_info> | cpio -idmv 1
./usr/share/microshift/release/release-aarch64.json
./usr/share/microshift/release/release-x86_64.json
```

1 **<my_microshift_release_info>** は、前の手順の RPM パッケージの名前に置き換えます。

4. 次のコマンドを実行して、コンテナー参照情報が含まれる JSON ファイルの場所を定義します。

```
$ RELEASE_FILE=</path/to/your/release-$(uname -m).json>
```

**</path/to/your/release-$(uname -m).json>** は、JSON ファイルへのフルパスに置き換えます。必ずアーキテクチャーに必要なファイルを使用してください。

5. 次のコマンドを実行して、イメージをビルドする手順が含まれる TOML ファイルの場所を定義します。

```
$ BLUEPRINT_FILE=</path/to/your/blueprint.toml>
```

**</path/to/your/blueprint.toml>** は、JSON ファイルへのフルパスに置き換えます。

6. 次のコマンドを実行して、コンテナーイメージ参照を生成し、ブループリント TOML ファイルに埋め込みます。

```
$  jq -r '.images | .[] | ("[[containers]]\nsource = \"" + . + "\"\n")' "${RELEASE_FILE}" >>
"${BLUEPRINT_FILE}"
```

**コンテナー参照を示す生成された <my_blueprint.toml> フラグメントの例**

```
[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:82cfef91557f9a70cff5a90accba45841a37524e9b93f98a97b20f6b2b69e5db"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:82cfef91557f9a70cff5a90accba45841a37524e9b93f98a97b20f6b2b69e5db"
```

7. 次の例を使用して、コンテナーイメージを Image Builder ブループリントに追加することで、コンテナーイメージを手動で埋め込むことができます。

   **Image Builder にコンテナーイメージを手動で埋め込むためのセクションの例**

   ```
   [[containers]]
   source = "<my_image_pullspec_with_tag_or_digest>"
   ```

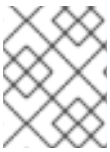
   **<my_image_pullspec_with_tag_or_digest>** は、デプロイする MicroShift バージョンで使用されるコンテナーイメージへの正確な参照に置き換えます。

## 5.4. イメージのビルドに向けた OSBUILDER ワーカー設定の更新

ブループリントを更新したら、osbuilder ワーカー設定を更新して、MicroShift コンテナーを埋め込むイメージのビルドを準備をする必要があります。

**前提条件**

- ビルドホストへの root ユーザーアクセス権がある。
- ビルドホストが Image Builder のシステム要件を満たしている。
- Image Builder と **composer-cli** ツールをインストールしてセットアップしている。



**注記**

**/etc/osbuild-worker/osbuild-worker.toml** ディレクトリーと設定ファイルが存在しない場合は、作成してください。

**手順**

1. **/etc/osbuild-worker/osbuild-worker.toml** osbuilder ワーカー設定ファイルの **[containers]** セクションに **auth_file_path** を設定して、認証用のプルシークレットをレジストリーに追加します。

   ```
   [containers]
   auth_file_path = "/etc/osbuild-worker/pull-secret.json"
   ```

2. **osbuild-worker** を再起動し、ホストを再起動して設定の変更を適用します。ホストを再起動すると、現在実行中のすべての **osbuild-worker** サービスが確実に再起動します。

## 5.5. オフラインデプロイ用の RPM-OSTREE イメージのビルドと使用

Image Builder を使用して、MicroShift コンテナーイメージが埋め込まれた **rpm-ostree** システムイメージを作成できます。コンテナーイメージを埋め込むには、Image Builder ブループリントにイメージ参照を追加する必要があります。必要に応じて、ユースケースに合わせてコミットと ISO を作成できます。

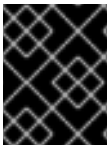
後述する手順の前提条件に加えて、次の前提条件を満たす必要があります。

### 5.5.1. オフラインデプロイの追加の前提条件

- オフラインで使用するための RHEL for Edge イメージブループリントを作成および更新してい

る。次の手順では、コンテナーイメージを使用して作成したブループリントの例を使用します。「オフラインデプロイ用の MicroShift コンテナーの埋め込み」手順で作成した更新済みのブループリントを使用する必要があります。

- **/etc/osbuild-worker/osbuild-worker.toml** 設定ファイルをオフラインで使用するために更新している。



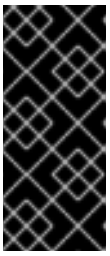
**重要**

次の手順の **minimum-microshift.toml** は、オフラインで使用するために更新した TOML の名前 <my_blueprint_name> に置き換えます。

### 5.5.2. ブループリントへの MicroShift サービスの追加する

MicroShift RPM パッケージを Image Builder ブループリントに追加すると、MicroShift が埋め込まれた RHEL for Edge イメージをビルドできるようになります。

- 手順 1 から開始して独自の最小限のブループリントファイルを作成すると、MicroShift のインストールが高速化されます。
- 手順 2 から開始して、すべての RPM パッケージとコンテナーイメージを含む、生成されたインストール用のブループリントを使用します。これはインストールプロセスに時間がかかりますが、コンテナー参照がローカルでアクセスされるため、起動は速くなります。



**重要**

- 次の手順の **<microshift_blueprint.toml>** を、使用している TOML ファイルの名前に置き換えます。
- 次の手順の **<microshift_blueprint>** を、ブループリントに使用する名前に置き換えます。

**手順**

1. 次の例を使用して、独自のブループリントファイルを作成します。

**Custom Image Builder ブループリントの例**

```
cat > <microshift_blueprint.toml> <<EOF (1)
name = "<microshift_blueprint>" (2)

description = ""
version = "0.0.1"
modules = []
groups = []

[[packages]]
name = "microshift"
version = "*"

[customizations.services]
enabled = ["microshift"]
EOF
```

1. **<microshift_blueprint.toml>** は TOML ファイルの名前です。
2. **<microshift_blueprint>** はブループリントの名前です。

**注記**

コマンドのワイルドカード * は、最新の MicroShift RPM を使用します。特定のバージョンが必要な場合は、ワイルドカードを必要なバージョンに置き換えます。たとえば、MicroShift 4.15.0 RPM をダウンロードするには、**4.15.0** を挿入します。

2. オプション: **/usr/share/microshift/blueprint** ディレクトリーにインストールされている、プラットフォームアーキテクチャーに固有のブループリントを使用します。ブループリントセクションの説明については、次のサンプルスニペットを参照してください。

**生成された Image Builder ブループリントのサンプルスニペット**

```
name = "microshift_blueprint"
description = "MicroShift 4.15.1 on x86_64 platform"
version = "0.0.1"
modules = []
groups = []

[[packages]] 1
name = "microshift"
version = "4.15.1"
...
...

[customizations.services] 2
enabled = ["microshift"]

[customizations.firewall]
ports = ["22:tcp", "80:tcp", "443:tcp", "5353:udp", "6443:tcp", "30000-32767:tcp", "30000-
32767:udp"]
...
...

[[containers]] 3
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:f41e79c17e8b41f1b0a5a32c3e2dd7cd15b8274554d3f1ba12b2598a347475f4"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:dbc65f1fba7d92b36cf7514cd130fe83a9bd211005ddb23a8dc479e0eea645fd"
...
…
EOF
```

1. **microshift-release-info** RPM と互換性のある同じバージョンを使用する、オプション以外のすべての MicroShift RPM パッケージの参照。
2. システム起動時に MicroShift を自動的に有効にし、デフォルトのネットワーク設定を適用するための参照。

3 オフラインデプロイメントに必要なオプション以外のすべての MicroShift コンテナーイメージの参照。

3. 次のコマンドを実行して、Image Builder にブループリントを追加します。

```
$ sudo composer-cli blueprints push <microshift_blueprint.toml> 1
```

1 **<microshift_blueprint.toml>** をTOML ファイルの名前に置き換えます。

**検証**

1. 次のコマンドを実行して、MicroShift パッケージのみをリストした Image Builder 設定を確認します。

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> | grep microshift 1
```

1 **<microshift_blueprint>** をブループリントの名前に置き換えます。

**出力例**

```
blueprint: microshift_blueprint v0.0.1
    microshift-greenboot-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-networking-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-release-info-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-selinux-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
```

2. オプション: 次のコマンドを実行して、インストールするすべてのコンポーネントをリストした Image Builder 設定を確認します。

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> 1
```

1 **<microshift_blueprint>** をブループリントの名前に置き換えます。

### 5.5.3. RHEL for Edge イメージの作成

ISO を作成するには、以下の手順を使用します。RHEL for Edge Installer イメージは、実行中のコンテナーからコミットをプルし、埋め込みの **rpm-ostree** コミット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持つ、インストール可能なブート ISO を作成します。

**前提条件**

- ビルドホストが Image Builder のシステム要件を満たしている。
- Image Builder と **composer-cli** ツールをインストールしてセットアップしている。
- ビルドホストへの root ユーザーアクセス権がある。
- **podman** ツールがインストールされている。

**手順**

1. 以下のコマンドを実行して **ostree** コンテナーイメージビルドを開始します。

```
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/{op-system-version-major}/$(uname -m)/edge" <microshift_blueprint> edge-container | awk '{print $2}') 1
```

1 <microshift_blueprint> をブループリントの名前に置き換えます。

このコマンドは、監視対象のビルドの ID (ID) も返します。

2. 次のコマンドを実行して、ビルドのステータスを定期的に確認できます。

```
$ sudo composer-cli compose status
```

**実行中のビルドの出力例**

```
ID                                   Status    Time                 Blueprint             Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332 RUNNING   Wed Jun 7 12:26:23 2023
microshift_blueprint     0.0.1     edge-container
```

**完了したビルドの出力例**

```
ID                                   Status    Time                 Blueprint             Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332 FINISHED  Wed Jun 7 12:32:37 2023
microshift_blueprint  0.0.1     edge-container
```

**注記**

起動および停止方法を理解している場合は、**watch** コマンドを使用してビルドを監視できます。

3. ID を使用してコンテナーイメージをダウンロードし、次のコマンドを実行して、使用可能なイメージを取得します。

```
$ sudo composer-cli compose image ${BUILDID}
```

4. 次のコマンドを実行して、ダウンロードしたコンテナーイメージの所有権を現在のユーザーに変更します。

```
$ sudo chown $(whoami). ${BUILDID}-container.tar
```

5. 次のコマンドを実行して、現在のユーザーの読み取り権限をイメージに追加します。

```
$ sudo chmod a+r ${BUILDID}-container.tar
```

6. 以下の手順を実行して、**ostree** コンテナーイメージが ISO ビルドで使用されるようにポート 8085 でサーバーをブートストラップします。

   a. 次のコマンドを実行して、**IMAGEID** 変数の結果を取得します。

```
$ IMAGEID=$(cat < "./${BUILDID}-container.tar" | sudo podman load | grep -o -P '(?
<=sha256[@:])[a-z0-9]*')
```

b. **IMAGEID** 変数の結果をもとに、以下のコマンドを実行して podman コマンドを実行します。

```
$ sudo podman run -d --name=minimal-microshift-server -p 8085:8080 ${IMAGEID}
```

このコマンドは、監視用に **IMAGEID** 変数に保存されているコンテナーの ID も返します。

7. 次のコマンドを実行して、インストーラーブループリントファイルを生成します。

```
cat > microshift-installer.toml <<EOF
name = "microshift-installer"

description = ""
version = "0.0.0"
modules = []
groups = []
packages = []
EOF
```

## 5.6. 関連情報

- コンテナーをレジストリーにプッシュしてイメージに埋め込む
- コンテナーレジストリーの認証情報
- 完全に非接続のホストのネットワーク設定
- MicroShift での Operator Lifecycle Manager の使用
- oc-mirror プラグインを使用したカスタムカタログの作成

# 第6章 GREENBOOT ヘルスチェックフレームワーク

Greenboot は、**rpm-ostree** システム (Red Hat Enterprise Linux for Edge (RHEL for Edge) など) の **systemd** サービスの汎用ヘルスチェックフレームワークです。このフレームワークは、**microshift-greenboot** および **greenboot-default-health-checks** RPM パッケージとともに MicroShift インストールに組み込まれています。

Greenboot ヘルスチェックは、システムの健全性を評価し、ソフトウェアトラブルが発生した場合の最後の健全な状態へのロールバックを自動化するために、次のようにさまざまなタイミングで実行されます。

- デフォルトのヘルスチェックスクリプトは、システムが起動するたびに実行されます。
- デフォルトのヘルスチェックに加えて、システムが起動するたびに実行されるようにアプリケーションヘルスチェックスクリプトを作成、インストール、設定することもできます。
- Greenboot を使用すると、更新中にエッジデバイスからロックアウトされるリスクを軽減し、更新が失敗した場合にサービスが大幅に中断されるのを防ぐことができます。
- 障害が検出されると、システムは **rpm-ostree** ロールバック機能を使用して、最後に認識された動作設定で起動します。この機能は、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に特に役立つ自動化機能です。

MicroShift アプリケーションのヘルスチェックスクリプトは、**microshift-greenboot** RPM に含まれています。**greenboot-default-health-checks** RPM には、DNS および **ostree** サービスにアクセスできることを確認するヘルスチェックスクリプトが含まれています。実行しているワークロード用に独自のヘルスチェックスクリプトを作成できます。たとえば、アプリケーションが開始したことを確認するものを作成できます。

**注記**

**rpm-ostree** を使用していないシステムで更新に失敗した場合、ロールバックはできません。これは、ヘルスチェックが実行される場合にも当てはまります。

## 6.1. GREENBOOT がディレクトリーを使用してスクリプトを実行する方法

ヘルスチェックスクリプトは、4 つの **/etc/greenboot** ディレクトリーから実行します。これらのスクリプトはアルファベット順に実行します。ワークロードのスクリプトを設定するときは、このことに留意してください。

システムが起動すると、Greenboot は、**required.d** および **wanted.d** ディレクトリーでスクリプトを実行します。これらのスクリプトの結果に応じて、Greenboot は起動を続行するか、次のようにロールバックを試みます。

1. システムが想定どおりの場合: **required.d** ディレクトリー内のすべてのスクリプトが成功すると、Greenboot は **/etc/greenboot/green.d** ディレクトリーにあるすべてのスクリプトを実行します。
2. システムに問題が発生している場合: **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** ディレクトリー内に存在するプレロールバックスクリプトを実行してから、システムを再起動します。

**注記**

Greenboot は、スクリプトとヘルスチェックの出力をシステムログにリダイレクトします。ログインすると、毎日のメッセージでシステム全体の状態が出力されます。

### 6.1.1. Greenboot ディレクトリーの詳細

スクリプトからゼロ以外の終了コードを返すことは、スクリプトが失敗したことを意味します。Greenboot は、以前のバージョンへのロールバックを試行する前に、システムを数回再起動してスクリプトを再試行します。

- **/etc/greenboot/check/required.d** には、失敗してはならないヘルスチェックが含まれています。
    - 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。
    - MicroShift の **40_microshift_running_check.sh** ヘルスチェックスクリプトは、このディレクトリーにインストールされます。
- **/etc/greenboot/check/wanted.d** には、システムをロールバックさせずに失敗できるヘルススクリプトが含まれています。
    - これらのスクリプトのいずれかが失敗すると、Greenboot は失敗を記録しますが、ロールバックを開始しません。
- **/etc/greenboot/green.d** には、Greenboot が起動の成功を宣言した後に実行されるスクリプトが含まれています。
- **/etc/greenboot/red.d** には、**40_microshift_pre_rollback.sh** プレロールバックスクリプトなど、Greenboot が起動の失敗を宣言した後に実行するスクリプトが含まれています。このスクリプトは、システムロールバックの直前に実行されます。このスクリプトは、MicroShift Pod と OVN-Kubernetes のクリーンアップを実行して、システムが以前のバージョンにロールバックされた後の潜在的な競合を回避します。

## 6.2. MICROSHIFT ヘルスチェックスクリプト

**40_microshift_running_check.sh** ヘルスチェックスクリプトは、コア MicroShift サービスの検証のみを実行します。カスタマイズしたワークロードのヘルスチェックスクリプトを Greenboot ディレクトリーにインストールして、システムの更新後にアプリケーションが正常に動作するようにします。スクリプトはアルファベット順に実行されます。

次の表に、MicroShift ヘルスチェックを示します。

**表6.1 MicroShift の検証ステータスと結果**

| 検証 | Pass | Fail |
|---|---|---|
| スクリプトが **root** 権限で実行されることを確認する | 次の手順 | **exit 0** |

| 検証 | Pass | Fail |
| --- | --- | --- |
| **microshift.service** が有効になっていることを確認する | 次の手順 | **exit 0** |
| **microshift.service** がアクティブになるのを待つ (!failed) | 次の手順 | **exit 1** |
| Kubernetes API ヘルスエンドポイントが機能し、トラフィックを受信するまで待つ | 次の手順 | **exit 1** |
| 任意の Pod が開始するのを待つ | 次の手順 | **exit 1** |
| コア namespace ごとに、イメージがプルされるのを待つ | 次の手順 | **exit 1** |
| コア namespace ごとに、Pod の準備が整うまで待つ | 次の手順 | **exit 1** |
| コア namespace ごとに、Pod が再起動していないかどうかを確認する | **exit 0** | **exit 1** |

### 6.2.1. 検証待機期間

各検証の待機時間は、デフォルトで 5 分です。待機期間の後、検証が成功しなかった場合は、失敗が宣言されます。この待機期間は、検証ループで起動するたびに基本待機期間だけ増加します。

- **/etc/greenboot/greenboot.conf** 設定ファイルで **MICROSHIFT_WAIT_TIMEOUT_SEC** 環境変数を設定することにより、基本時間の待機期間をオーバーライドできます。たとえば、**MICROSHIFT_WAIT_TIMEOUT_SEC=180** のように値を 180 秒にリセットすることで、待機時間を 3 分に変更できます。

## 6.3. SYSTEMD ジャーナルサービスデータの永続性を有効にする

**systemd** ジャーナルサービスのデフォルト設定では、揮発性の **/run/log/journal** ディレクトリーにデータが保存されます。次のシステム起動および再起動までのシステムログを表示するには、ログの永続性を有効にし、最大ジャーナルデータサイズの制限を設定する必要があります。

**手順**

1. 次のコマンドを実行してディレクトリーを作成します。

```
$ sudo mkdir -p /etc/systemd/journald.conf.d
```

2. 次のコマンドを実行して、設定ファイルを作成します。

```
cat <<EOF | sudo tee /etc/systemd/journald.conf.d/microshift.conf &>/dev/null
[Journal]
```

```
Storage=persistent
SystemMaxUse=1G
RuntimeMaxUse=1G
EOF
```

3. サイズ要件に合わせて設定ファイルの値を編集します。

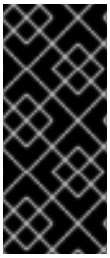
**関連情報**

- マニフェストの自動適用

## 6.4. 更新とサードパーティーのワークロード

ヘルスチェックは、更新後に特に役立ちます。Greenboot ヘルスチェックの出力を調べて、更新が有効であると宣言されたかどうかを判断できます。このヘルスチェックは、システムが正常に動作しているかどうかを判断するのに役立ちます。

更新用のヘルスチェックスクリプトは **/etc/greenboot/check/required.d** ディレクトリーにインストールされ、システムの起動時に自動的に実行します。ゼロ以外のステータスでスクリプトを終了すると、システムの起動が失敗したと宣言されます。



**重要**

サードパーティーのワークロードを開始する前に、更新が有効であると宣言されるまで待ちます。ワークロードの開始後にロールバックを実行すると、データが失われる可能性があります。一部のサードパーティーのワークロードは、更新が完了する前にデバイスでデータを作成または更新します。ロールバックすると、ファイルシステムは更新前の状態に戻ります。

## 6.5. 更新結果の確認

起動に成功すると、Greenboot は GRUB で変数 **boot_success=** を **1** に設定します。次の手順を使用して、更新後のシステムヘルスチェックの全体的なステータスをシステムログで表示できます。

**手順**

- システムヘルスチェックの全体的なステータスにアクセスするには、次のコマンドを実行します。

```
$ sudo grub2-editenv - list | grep ^boot_success
```

**システムが正常に起動した場合の出力例**

```
boot_success=1
```

## 6.6. システムログのヘルスチェック出力へのアクセス

次の手順を使用して、システムログのヘルスチェックの出力に手動でアクセスできます。

**手順**

- ヘルスチェックの結果にアクセスするには、次のコマンドを実行します。

```
$ sudo journalctl -o cat -u greenboot-healthcheck.service
```

**失敗したヘルスチェックの出力例**

```
...
...
Running Required Health Check Scripts...
STARTED
GRUB boot variables:
boot_success=0
boot_indeterminate=0
boot_counter=2
...
...
Waiting 300s for MicroShift service to be active and not failed
FAILURE
...
...
```

## 6.7. システムログのプレロールバックヘルスチェック出力へのアクセス

システムログのヘルスチェックスクリプトの出力にアクセスできます。たとえば、次の手順でプレロールバックスクリプトの結果を確認します。

**手順**

- プレロールバックスクリプトの結果にアクセスするには、次のコマンドを実行します。

```
$ sudo journalctl -o cat -u redboot-task-runner.service
```

**プレロールバックスクリプトの出力例**

```
...
...
Running Red Scripts...
STARTED
GRUB boot variables:
boot_success=0
boot_indeterminate=0
boot_counter=0
The ostree status:
* rhel c0baa75d9b585f3dd989a9cf05f647eb7ca27ee0dbd4b94fe8c93ed3a4b9e4a5.0
    Version: 9.1
    origin: <unknown origin type>
  rhel 6869c1347b0e0ba1bbf0be750cdf32da5138a1fcbc5a4c6325ab9eb647b64663.0 (rollback)
    Version: 9.1
    origin refspec: edge:rhel/9/x86_64/edge
System rollback imminent - preparing MicroShift for a clean start
Stopping MicroShift services
Removing MicroShift pods
Killing conmon, pause and OVN processes
Removing OVN configuration
Finished greenboot Failure Scripts Runner.
```

```
Cleanup succeeded
Script '40_microshift_pre_rollback.sh' SUCCESS
FINISHED
redboot-task-runner.service: Deactivated successfully.
```

## 6.8. ヘルスチェックスクリプトを使用した更新の確認

次の手順を使用して、更新後にシステムログ内の Greenboot ヘルスチェックスクリプトの出力にアクセスします。

**手順**

- 更新チェックの結果にアクセスするには、次のコマンドを実行します。

```
$ sudo grub2-editenv - list | grep ^boot_success
```

**更新が成功した場合の出力例**

```
boot_success=1
```

コマンドが **boot_success=0** を返す場合は、Greenboot ヘルスチェックがまだ実行しているか、更新が失敗しています。

## 6.9. 関連情報

- Greenboot ワークロードヘルスチェックスクリプト

# 第7章 インストールの問題のトラブルシューティング

MicroShift インストールが失敗した場合にトラブルシューティングを行うには、sos report を実行します。**sos report** コマンドを使用して、システム内のさまざまなコンポーネントやアプリケーションからの有効なプラグインとデータをすべて表示する詳細レポートを生成します。

## 7.1. SOS レポートからのデータの収集

**前提条件**

- **sos** パッケージがインストールされている。

**手順**

1. 障害が発生したホストに root ユーザーとしてログインします。
2. 次のコマンドを実行して、デバッグレポートの作成手順を実行します。

```
$ microshift-sos-report
```

**出力例**

```
sosreport (version 4.5.1)

This command will collect diagnostic and configuration information from
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp/sos.o0sznf_8 and may be provided to a Red Hat support
representative.

Any information provided to Red Hat will be treated in accordance with
the published support policies at:

        Distribution Website : https://www.redhat.com/
        Commercial Support   : https://www.access.redhat.com/

The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.

No changes will be made to system configuration.


 Setting up archive ...
 Setting up plugins ...
 Running plugins. Please wait ...

  Starting 1/2   microshift      [Running: microshift]
  Starting 2/2   microshift_ovn  [Running: microshift microshift_ovn]
  Finishing plugins              [Running: microshift]

  Finished running plugins
```

```
Found 1 total reports to obfuscate, processing up to 4 concurrently

sosreport-microshift-rhel9-2023-03-31-axjbyxw :    Beginning obfuscation...
sosreport-microshift-rhel9-2023-03-31-axjbyxw :    Obfuscation completed

Successfully obfuscated 1 report(s)

Creating compressed archive...

A mapping of obfuscated elements is available at
 /var/tmp/sosreport-microshift-rhel9-2023-03-31-axjbyxw-private_map

Your sosreport has been generated and saved in:
 /var/tmp/sosreport-microshift-rhel9-2023-03-31-axjbyxw-obfuscated.tar.xz

 Size 444.14KiB
 Owner root
 sha256 922e5ff2db25014585b7c6c749d2c44c8492756d619df5e9838ce863f83d4269

Please send this file to your support representative.
```

## 7.2. 関連情報

- MicroShift の sos レポートについて
- テクニカルサポート用の SOS レポートの生成