Red Hat Container Development Kit 3.15

スタートガイド

Red Hat Container Development Kit の使用および開発に関するクイックスタートガイド

Last Updated: 2023-02-24

Red Hat Container Development Kit 3.15 スタートガイド

Red Hat Container Development Kit の使用および開発に関するクイックスタートガイド

Yana Hontyk
yhontyk@redhat.com

Kevin Owen
kowen@redhat.com

Chris Negus

Robert Krátký

法律上の通知

概要

本ガイドでは、Red Hat Container Development Kit を使用して速度を高める方法を説明します。Docker、Kubernetes、および OpenShift Container Platform を使用してコンテナー化アプリケーションを開発する最初のステップについては、お使いのホストワークステーション (Microsoft Windows、macOS、または Red Hat Enterprise Linux)、および Red Hat Container Development Kit が提供する Container Development Environment 内からの手順や例が記載されています。

目次

第1章 CONTAINER DEVELOPMENT KIT の使用

本セクションでは、Container Development Kit の設定、インストール、およびアンインストールを説明します。

1.1. RED HAT CONTAINER DEVELOPMENT KIT の概要

Red Hat Container Development Kit は、コンテナー化されたアプリケーションを開発するプラットフォームを提供します。開発者が Red Hat Enterprise Linux プラットフォームでコンテナーアプリケーションの開発とテストを行うための環境を迅速かつ簡単に設定できるようにするツールセットです。

- Container Development Kit は、使用しているラップトップ、デスクトップ、またはサーバーシステムにインストールすることができる専用のコンテナー開発環境を提供します。Container Development Environment は、Red Hat Enterprise Linux 仮想マシンで提供されます。

- Microsoft Windows、macOS、および Linux オペレーティングシステムでは Container Development Kit が利用できます。そのため、開発者は Red Hat Enterprise Linux エコシステムにデプロイする準備ができているアプリケーションを作成する一方で、開発者は自由にプラットフォームを選択できます。

Container Development Kit は、ローカルおよびクラウドの両方でアプリケーションを作成するために Red Hat ソリューションと製品を利用する開発者向けのツール、リソース、およびサポートを提供する Red Hat Developers プログラムの一部です。詳細情報やプログラムへの登録は、developers.redhat.com を参照してください。

1.1.1. Container Development Kit ドキュメントについて

- Red Hat Container Development Kit 3.15 リリースノートおよび既知の問題 には、製品の現行リリースに関する情報と、ユーザーがこれを使用すると発生する可能性のある既知の問題の一覧が含まれています。

- Container Development Kit スタートガイド には、**OpenShift Container Platform**、**Docker**、**Eclipse**、各種コマンドラインツールなどのツールやサービスを使用して、Red Hat Enterprise Linux ベースのコンテナーを開発するために、Container Development Environment をインストールして使用を開始する方法が記載されています。

- Red Hat Container Development Kit の問題を報告するか、https://issues.jboss.org/projects/CDK で **CDK** プロジェクトを使用して新機能をリクエストします。

- Red Hat Container Development Kit 3.15 リリースノートおよび既知の問題および Container Development Kit スタートガイドの問題は、https://issues.jboss.org/projects/RHDEVDOCS で **RHDEVDOCS** プロジェクトを使用して報告します。

1.2. CDK のインストールの準備

1.2.1. 概要

ここでは、CDK と必要な依存関係をインストールする方法を説明します。

個人システムに CDK を設定する基本的な手順は次のとおりです。

1. 仮想化環境の設定
2. Red Hat Container Development Kit Download ページからオペレーティングシステムの CDK ソフトウェアをダウンロードする
3. CDK をインストールする
4. CDK を設定して起動する
5. 効率的に使用できるように CDK を設定する

設定手順は、仮想マシンの起動権限を持つ通常のユーザーとして実行する必要があります。この手順では、そのパーミッションを割り当てる方法と、ハイパーバイザーおよびコマンドシェルを設定して CDK を起動し、効果的に対話する方法を説明します。

1.2.2. 前提条件

CDK では、OpenShift クラスターがプロビジョニングされる仮想マシンを起動するハイパーバイザーが必要です。CDK を設定する前に、選択したハイパーバイザーがシステムにインストールされ、有効になっていることを確認します。ハイパーバイザーを起動して実行したら、CDK がそのハイパーバイザーと連携するには、追加の設定が必要になります。

ホストオペレーティングシステムによっては、以下の推奨されるネイティブハイパーバイザーを選択できます。

macOS

- hyperkit

Linux

- KVM

Windows

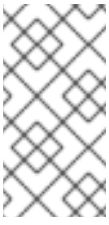
- Hyper-V

注記

Hyper-V で CDK を使用するには、Hyper-V をインストール した後に Hyper-V Manager を使用して 仮想スイッチを追加 して、設定オプション **hyperv-virtual-switch** をこの仮想スイッチに設定するようにしてください。特定の設定手順は、Hyper-V ハイパーバイザーの設定 を参照してください。

すべてのプラットフォーム

- VirtualBox



注記

Error: machine does not exist 問題を回避するために、Windows では VirtualBox 5.1.12 以降を使用することが推奨されます。ハイパーバイザーに関連する問題が発生した場合は、ドライバープラグインのトラブルシューティング を参照してください。

その実行に必要なハードウェアおよびオペレーティングシステムのバージョンを確認するには、各ハイパーバイザーのドキュメント を参照してください。

1.3. 仮想化環境の設定

1.3.1. 概要

特定のオペレーティングシステムのハイパーバイザーを設定するには、適切な手順に従ってください。CDK は、libmachine とそのドライバープラグインアーキテクチャーを使用して、CDK 仮想マシンを管理する一貫した方法を提供します。

ハイパーバイザーによっては、ドライバープラグインを手動でインストールする必要があります。CDK は VirtualBox ドライバープラグインを組み込むため、設定する追加の手順は必要ありません。ただし、**--vm-driver virtualbox** フラグまたは永続設定を使用して、VirtualBox を CDK に識別する必要があります。詳細は、VirtualBox を使用するように CDK を設定 を参照してください。

お使いのハイパーバイザーおよびオペレーティングシステムに適したセクションを参照してください。

- KVM ドライバーの設定 (Red Hat Enterprise Linux の場合)
- hyperkit ドライバーの設定 (macOS の場合)
- Hyper-V ハイパーバイザーの設定 (Windows の場合)
- VirtualBox を使用するように CDK を設定 (VirtualBox の場合)

1.3.2. Red Hat Enterprise Linux

1.3.2.1. KVM ドライバーの設定

CDK は現在 **docker-machine-driver-kvm** バージョン 0.10.0 に対してテストされています。

1. root で KVM バイナリーをインストールし、以下のように実行可能にします。

```
# curl -L https://github.com/dhiltgen/docker-machine-kvm/releases/download/v0.10.0/docker-machine-driver-kvm-centos7 -o /usr/local/bin/docker-machine-driver-kvm
# chmod +x /usr/local/bin/docker-machine-driver-kvm
```

   詳細は、Docker Machine KVM ドライバー の GitHub ドキュメントを参照してください。

2. root で **libvirt** および **qemu-kvm** をシステムにインストールします。

```
# yum install libvirt qemu-kvm
```

3. root で、自身を **libvirt** グループに追加します。

```
# usermod -a -G libvirt username
```

4. 現在のセッションを更新して、グループの変更を適用します。

```
$ newgrp libvirt
```

5. root で **libvirtd** サービスを起動します。

```
# systemctl start libvirtd
# systemctl enable libvirtd
```

1.3.2.2. libvirtd サービスの起動

1. **libvirtd** のステータスを確認します。

```
$ systemctl is-active libvirtd
```

1. **libvirtd** がアクティブでない場合は、root で **libvirtd** サービスを起動します。

```
# systemctl start libvirtd
```

1.3.2.2.1. 次のステップ

ハイパーバイザーをインストールして設定したら、CDK のインストール に進んでください。

1.3.3. macOS

1.3.3.1. hyperkit ドライバーの設定

CDK は現在 **docker-machine-driver-hyperkit** バージョン 1.0.0 に対してテストされています。

hyperkit を使用するには、**hyperkit** と **docker-machine-driver-hyperkit** の両方をインストールする必要があります。

1.3.3.1.1. hyperkit のインストール

- macOS 用の Docker Desktop がインストールされている場合は、**hyperkit** がすでにインストールされています。
- Homebrew を使用する場合は、最新バージョンの **hyperkit** をインストールできます。

```
$ brew install hyperkit
```

1.3.3.1.2. docker-machine-driver-hyperkit のインストール

- Homebrew を使用する場合は、最新バージョンの **docker-machine-driver-hyperkit** をインストールできます。

```
$ brew install docker-machine-driver-hyperkit
```

- ここでは、**docker-machine-driver-hyperkit** バイナリーをダウンロードしてインストールし、**PATH** 環境変数にあるディレクトリーに配置できます。**/usr/local/bin** ディレクトリーは Docker Machine バイナリーのデフォルトインストールディレクトリーであるため、適切な選択肢となります。

以下の手順では、**docker-machine-driver-hyperkit** バイナリーのインストールを **/usr/local/bin/** ディレクトリーに説明します。

1. 以下を使用して **docker-machine-driver-hyperkit** バイナリーをダウンロードします。

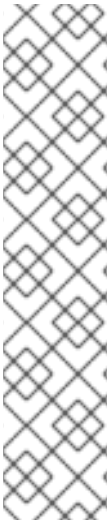
```
$ sudo curl -L  https://github.com/machine-drivers/docker-machine-driver-hyperkit/releases/download/v1.0.0/docker-machine-driver-hyperkit -o /usr/local/bin/docker-machine-driver-hyperkit
```

2. **docker-machine-driver-hyperkit** バイナリーの root アクセスを有効にし、これをデフォルトの **wheel** グループに追加します。

```
$ sudo chown root:wheel /usr/local/bin/docker-machine-driver-hyperkit
```

3. バイナリーの所有者ユーザー ID (SUID) を以下のように設定します。

```
$ sudo chmod u+s,+x /usr/local/bin/docker-machine-driver-hyperkit
```



注記

ダウンロードした **docker-machine-driver-hyperkit** バイナリーは、特定のバージョンの macOS に対してコンパイルされます。macOS バージョンのアップグレード後にドライバーが機能しなくなる可能性があります。この場合は、ソースからドライバーをコンパイルできます。

```
$ go get -u -d github.com/machine-drivers/docker-machine-driver-hyperkit
$ cd $GOPATH/src/github.com/machine-drivers/docker-machine-driver-hyperkit

# Install docker-machine-driver-hyperkit binary into /usr/local/bin
$ make build
```

詳細は、GitHub の hyperkit ドライバー のドキュメントを参照してください。

1.3.3.1.3. 次のステップ

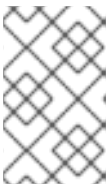
ハイパーバイザーをインストールして設定したら、CDK のインストール に進んでください。

1.3.4. Windows

1.3.4.1. Hyper-V ハイパーバイザーの設定

Hyper-V で CDK を使用するには、以下を行います。

1. Hyper-V をインストールします。
2. ユーザーをローカルの Hyper-V Administrators グループに追加します。



注記

これは、Hyper-V Management API で仮想マシンの作成および削除を可能にするために必要です。詳細は Hyper-V コマンドは管理者として実行する必要がある を参照してください。

3. 外部仮想スイッチ を追加します。
4. 仮想スイッチを、ネットワークに接続しているネットワークカード (有線または無線) でペアになっていることを確認します。

5. 設定オプション **hyperv-virtual-switch** または startup フラグ **--hyperv-virtual-switch** を使用して、CDK に使用する外部仮想スイッチの名前を設定します。
たとえば、PowerShell では、以下を使用します。

```
PS> minishift config set hyperv-virtual-switch "External (Wireless)"
```

または

```
PS> minishift start --hyperv-virtual-switch "External (Wireless)"
```

注記

- 仮想スイッチの名前は、大文字と小文字が区別されます。
- **HYPERV_VIRTUAL_SWITCH** 環境変数の使用が非推奨になりました。代わりに **MINISHIFT_HYPERV_VIRTUAL_SWITCH** を設定オプションとして使用することはできますが、Windows では ASCII 以外の文字に対応していないため、これは推奨されません。

1.3.4.1.1. 次のステップ

ハイパーバイザーをインストールして設定したら、CDK のインストール に進んでください。

1.3.5. VirtualBox を使用するように CDK を設定

組み込みの VirtualBox ドライバーを使用するには、VirtualBox を手動でインストールする必要があります。VirtualBox バージョン 5.1.12 以降が必要です。埋め込みドライバーを使用する前に、必ず VirtualBox をダウンロードし、インストール してください。

--vm-driver virtualbox フラグまたは永続設定オプションを使用して、VirtualBox を CDK に識別する必要があります。

1.3.5.1. VirtualBox の一時的な使用

minishift start コマンドを実行するたびに、**--vm-driver virtualbox** フラグをコマンドラインで指定する必要があります。以下に例を示します。

```
$ minishift start --vm-driver virtualbox
```

1.3.5.2. VirtualBox の永続的な使用

vm-driver オプションを永続設定オプションとして設定すると、毎回 **--vm-driver virtualbox** フラグを明示的に渡さずに、**minishift start** を実行できます。**vm-driver** の永続的な設定オプションを以下のように設定できます。

```
$ minishift config set vm-driver virtualbox
```

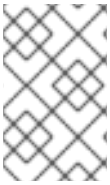
注記

minishift start を実行する前に、**vm-driver** の永続的な設定オプションを指定する必要があります。**minishift start** がすでに実行している場合は、**minishift delete** を実行し、**minishift config set vm-driver virtualbox** で設定してから、VirtualBox ドライバーを使用できるように **minishift start** を実行します。

1.3.5.3. 次のステップ

ハイパーバイザーをインストールして設定したら、CDK のインストール に進んでください。

1.4. CDK のインストール



注記

CDK ソフトウェアをダウンロードする前に、Red Hat Developer Program のサイトに登録するか、Red Hat サブスクリプション認証情報を使用して Red Hat カスタマーポータルにログインする必要があります。

1. Red Hat Container Development Kit Download ページから、お使いのオペレーティングシステムの CDK をダウンロードします。

2. ダウンロードした **minishift** ファイルを **PATH** のディレクトリーにコピーし、実行可能にします。ダウンロードした実行ファイルの名前は、**cdk-3.15.0-1-minishift-darwin-amd64** (macOS の場合)、**cdk-3.15.0-1-minishift-linux-amd64** (Red Hat Enterprise Linux の場合)、または **cdk-3.15.0-1-minishift-windows-amd64.exe** (Windows の場合) になります。実行可能ファイルが **~/Downloads** ディレクトリーにあることを前提とし、オペレーティングシステムの手順に従います。
   Red Hat Enterprise Linux の場合:

```
$ mkdir -p ~/bin
$ cp ~/Downloads/cdk-3.15.0-1-minishift* ~/bin/minishift
$ chmod +x ~/bin/minishift
$ export PATH=$PATH:$HOME/bin
$ echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

   MacOS の場合:

```
$ mkdir -p ~/bin
$ cp ~/Downloads/cdk-3.15.0-2-minishift* ~/bin/minishift
$ chmod +x ~/bin/minishift
$ export PATH=$PATH:$HOME/bin
$ echo export PATH=$PATH:$HOME/bin >> ~/.bash_profile
```

   Windows の場合:

   必要なディレクトリーを作成し、ダウンロードした CDK バイナリーをディレクトリーにコピーし、バイナリーの名前を **minishift.exe** に変更します。Windows **PATH** 変数にディレクトリーパスを追加します。

   PATH で **minishift.exe** を取得できない場合は、現在のディレクトリーから **./minishift.exe** (またはコマンドラインの **.\minishift.exe**) として実行できます。

注記

- Windows オペレーティングシステムでは、#236 の問題により、ローカルの **C:** ドライブから CDK バイナリーを実行する必要があります。ネットワークドライブから CDK を実行できません。

1.5. CDK クイックスタート

1.5.1. 概要

このセクションでは、CDK とプロビジョニングされた OpenShift クラスターの簡単なデモについて説明します。CDK の使用に関する詳細は、基本的な使用方法 セクションを参照してください。

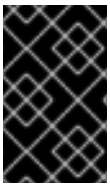
OpenShift との対話は、ホストにコピーされるコマンドラインツールの **oc** を使用しています。CDK がローカル OpenShift インスタンスとの対話および設定を支援する方法の詳細は、OpenShift クライアントバイナリー を参照してください。

OpenShift クラスターアーキテクチャーについての詳細は、OpenShift ドキュメントの アーキテクチャーの概要 を参照してください。

以下の手順では、KVM ハイパーバイザードライバーを使用して、Linux オペレーティングシステムの CDK を使い始める方法を説明します。

1.5.2. CDK の設定

minishift setup-cdk コマンドは、システムで CDK を実行するために必要なコンポーネントを取得して設定します。デフォルトでは、**minishift setup-cdk** は CDK コンテンツを **~/.minishift** ディレクトリーに格納します (Windows では **%USERPROFILE%/.minishift**)。



重要

~/.minishift 以外のディレクトリーを使用するには、環境変数 で説明されているように、**--minishift-home** フラグまたは **MINISHIFT_HOME** 環境変数を設定する必要があります。

以下のコマンドを実行して、Red Hat Enterprise Linux の CDK を設定します。

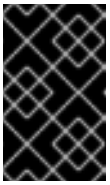
```
$ minishift setup-cdk
Setting up CDK 3 on host using '/home/user/.minishift' as Minishift's home directory
Copying minishift-rhel7.iso to '/home/user/.minishift/cache/iso/minishift-rhel7.iso'
Copying oc to '/home/user/.minishift/cache/oc/v3.10.45/linux/oc'
Creating configuration file '/home/user/.minishift/config/config.json'
Creating marker file '/home/user/.minishift/cdk'
Default add-ons anyuid, admin-user, xpaas, registry-route, che, eap-cd installed
Default add-ons anyuid, admin-user, xpaas enabled
CDK 3 setup complete.
```

Windows または macOS の場合: Windows と macOS で **minishift setup-cdk** コマンドを実行すると、異なるコンポーネントおよびパス名に基づいて、出力が若干異なります。

1.5.3. CDK の起動

1. デフォルトでは、**minishift start** により Red Hat Subscription Manager アカウントのユーザー名とパスワードの入力を求めるプロンプトが表示されます。その情報を入力するか、または以下を選択できます。

   - 登録を省略: **--skip-registration** オプションを追加して **minishift start** が CDK 仮想マシンを登録しないようにします。
   - 登録永続: 登録情報を環境変数にエクスポートし、**minishift** が起動するたびに自動的に取得できるようにします。

   重要

   暗号化されていない登録情報を環境変数に保存することは保護されません。セキュリティーには、**minishift start** プロンプトを使用して認証情報を入力することが推奨されます。

   以下のように登録情報をエクスポートします。

   Red Hat Enterprise Linux の場合:

   ```
   $ export MINISHIFT_USERNAME='<RED_HAT_USERNAME>'
   $ export MINISHIFT_PASSWORD='<RED_HAT_PASSWORD>'
   $ echo export MINISHIFT_USERNAME=$MINISHIFT_USERNAME >> ~/.bashrc
   $ echo export MINISHIFT_PASSWORD=$MINISHIFT_PASSWORD >> ~/.bashrc
   ```

   MacOS の場合:

   ```
   $ export MINISHIFT_USERNAME='<RED_HAT_USERNAME>'
   $ export MINISHIFT_PASSWORD='<RED_HAT_PASSWORD>'
   $ echo export MINISHIFT_USERNAME=$MINISHIFT_USERNAME >> ~/.bash_profile
   $ echo export MINISHIFT_PASSWORD=$MINISHIFT_PASSWORD >> ~/.bash_profile
   ```

   Windows の場合:

   コマンドの使用:

   ```
   C:\> set MINISHIFT_USERNAME='<RED_HAT_USERNAME>'
   C:\> set MINISHIFT_PASSWORD='<RED_HAT_PASSWORD>'
   C:\> setx MINISHIFT_USERNAME %MINISHIFT_USERNAME%
   C:\> setx MINISHIFT_PASSWORD %MINISHIFT_PASSWORD%
   ```

   PowerShell の使用:

   ```
   PS> $env:MINISHIFT_USERNAME = '<RED_HAT_USERNAME>'
   PS> $env:MINISHIFT_PASSWORD = '<RED_HAT_PASSWORD>'
   PS> setx MINISHIFT_USERNAME $env:MINISHIFT_USERNAME
   PS> setx MINISHIFT_PASSWORD $env:MINISHIFT_PASSWORD
   ```

2. **minishift start** コマンドを実行します。

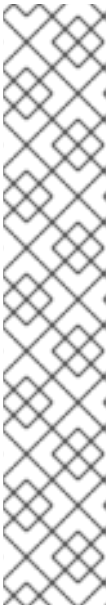
   ```
   $ minishift start
   -- Starting profile 'minishift'
   ...
   -- Minishift VM will be configured with ...
   ```

```
   Memory:    4 GB
   vCPUs :    2
   Disk size: 20 GB
-- Starting Minishift VM ..... OK
-- Registering machine using subscription-manager
   Registration in progress ..... OK [42s]
...
   OpenShift server started.

   The server is accessible via web console at:
      https://192.168.42.60:8443/console

   You are logged in as:
      User:     developer
      Password: <any value>

   To login as administrator:
      oc login -u system:admin
...
```



注記

- IP は OpenShift クラスターごとに動的に生成されます。IP を確認するには、**minishift ip** コマンドを実行します。
- デフォルトでは、CDK はホスト OS に最も適したドライバーを使用します。別のドライバーを使用するには、**minishift start** で **--vm-driver** フラグを設定します。たとえば、Linux オペレーティングシステムで KVM の代わりに VirtualBox を使用するには、**minishift start --vm-driver=virtualbox** を実行します。
- CDK を起動すると、複数のチェックを実行して CDK 仮想マシンと OpenShift クラスターが正常に起動できることを確認します。起動の確認に失敗した場合は、スタートガイドのトラブルシューティング のトピックで、考えられる原因および解決策に関する情報を参照してください。

minishift start オプションの詳細は、**minishift start** を参照してください。

3. **minishift oc-env** を使用して、**oc** バイナリーを **PATH** 環境変数に追加するためにシェルに入力する必要があるコマンドを表示します。**oc-env** の出力は、OS およびシェルタイプによって異なります。

```
$ minishift oc-env
export PATH="/home/user/.minishift/cache/oc/v3.11.346/linux:$PATH"
# Run this command to configure your shell:
# eval $(minishift oc-env)
```

コマンドラインインターフェイスおよび Web コンソールで OpenShift と対話する方法は、OpenShift クライアントバイナリー セクションを参照してください。

1.5.4. サンプルアプリケーションのデプロイ

OpenShift は、テンプレート、ビルダーアプリケーション、クイックスタートなどのさまざまなサンプルアプリケーションを提供します。以下の手順では、コマンドラインから Node.js アプリケーションのサンプルをデプロイする方法を説明します。

1. Node.js サンプルアプリケーションを作成します。

```
$ oc new-app https://github.com/openshift/nodejs-ex -l name=myapp
```

2. アプリケーションがビルドおよびデプロイされるまで、ビルドログを追跡します。

```
$ oc logs -f bc/nodejs-ex
```

3. サービスにルートを公開します。

```
$ oc expose svc/nodejs-ex
```

4. アプリケーションにアクセスします。

```
$ minishift openshift service nodejs-ex --in-browser
```

5. CDK を停止するには、以下のコマンドを使用します。

```
$ minishift stop
Stopping local OpenShift cluster...
Unregistering machine
Cluster stopped.
```

OpenShift でのアプリケーションの作成に関する詳細は、OpenShift ドキュメントの 新規アプリケーションの作成 を参照してください。

1.6. CDK のアンインストール

1.6.1. 概要

このセクションでは、**minishift** バイナリーをアンインストールし、関連するファイルを削除する方法を説明します。

1.6.2. CDK のアンインストール

1. CDK 仮想マシンと仮想マシン固有のファイルを削除します。

```
$ minishift delete
```

このコマンドは、**MINISHIFT_HOME/.minishift/machines/minishift** ディレクトリーにあるすべてのファイルを削除します。キャッシュされた他のデータと 永続設定 は削除されません。

2. CDK を完全にアンインストールするには、**MINISHIFT_HOME** ディレクトリー (デフォルト **~/.minishift**) にあるものをすべて削除します。
Red Hat Enterprise Linux および macOS の場合:

```
$ rm -rf ~/.minishift
```

Windows PowerShell の場合:

<MINISHIFT_HOME> をホームディレクトリーの場所に置き換えます。

```
PS C:\> Remove-Item -Recurse -Force C:\<MINISHIFT_HOME>\.minishift\
```

Windows コマンドプロンプトの場合:

<MINISHIFT_HOME> をホームディレクトリーの場所に置き換えます。(代わりに **del /s** コマンドを使用しないといけない場合があります。)

```
c:\> rm -r c:\<MINISHIFT_HOME>\.minishift
```

3. ハイパーバイザー管理ツールを使用して、CDK 仮想マシンに関連するアーティファクトが残っていないことを確認します。たとえば、KVM を使用する場合は、**virsh** コマンドを実行する必要があります。

第2章 CDK の使用

本セクションでは、**minishift** コマンドラインインターフェイス (CLI) を使用する方法を説明します。これは、基本的なコマンドと、プロファイル、イメージのキャッシュ、アドオン、ホストフォルダーなどの高度な機能を対象としています。

minishift コマンドおよび **oc** コマンドを使用して、ローカルの OpenShift クラスターを管理する方法は、3章*Container Development Kit を使用した OpenShift との対話* を参照してください。

2.1. 基本的な使用方法

2.1.1. 概要

CDK を使用する場合は、以下のコンポーネントと対話します。

- CDK 仮想マシン (VM)
- 仮想マシンで実行している Docker デーモン
- Docker デーモンで実行している OpenShift クラスター

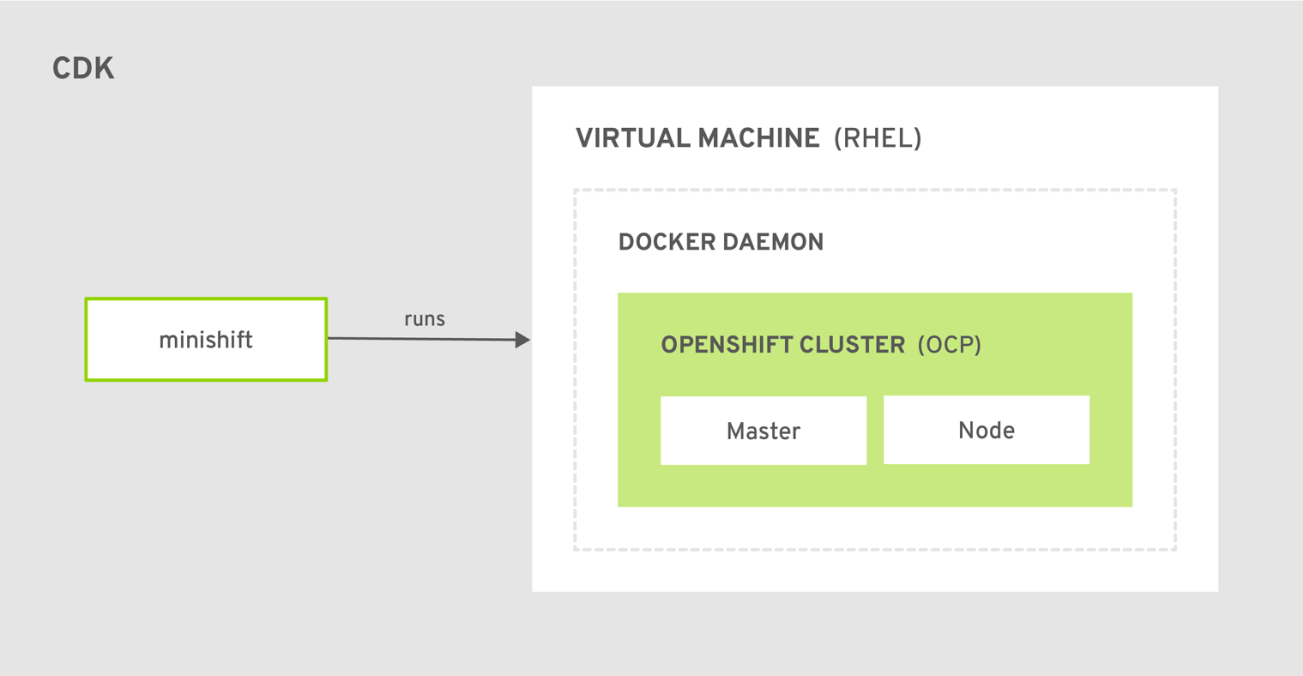
CDK アーキテクチャー の図では、これらのコンポーネントについて説明します。簡単な実行のために **PATH** に配置される **minishift** バイナリーは、CDK 仮想マシンを **起動**、**停止**、および **削除** するために使用されます。仮想マシン自体は、Red Hat Enterprise Linux Live ISO からブートストラップされます。

docker-env などの一部の CDK コマンドは Docker デーモンと対話し、他のものも OpenShift クラスター (**openshift** コマンドなど) と通信します。

OpenShift クラスターが稼働したら、**oc** バイナリーを使用して対話します。(デフォルトの **~/.minishift** ごとに) CDK はこのバイナリーを **$MINISHIFT_HOME** でキャッシュします。**minishift oc-env** は、**oc** バイナリーを **PATH** に追加する簡単な方法です。

CDK を使用したローカル OpenShift クラスターの管理に関する詳細は、3章*Container Development Kit を使用した OpenShift との対話* を参照してください。

図2.1: CDK アーキテクチャー



2.1.2. CDK ライフサイクル

2.1.2.1. minishift setup-cdk コマンド

minishift setup-cdk コマンドは、Red Hat Enterprise Linux ISO イメージを CDK キャッシュにコピーし、デフォルトの設定ファイルを作成し、デフォルトのアドオンをインストールして有効にし、コマンドが実行されていることを確認します。

また、このコマンドは **oc** バイナリーをホストにコピーし、**oc** コマンドラインツールまたは Web コンソールを使用して OpenShift と対話できるようにします。これは、**minishift start** コマンドの出力にある URL を使用してアクセスできます。

2.1.2.2. minishift start コマンド

minishift start コマンドは CDK 仮想マシンを作成し、設定し、CDK 仮想マシン内にローカルで単一ノードの OpenShift クラスターをプロビジョニングします。

2.1.2.3. minishift stop コマンド

minishift stop コマンドは OpenShift クラスターを停止し、CDK 仮想マシンをシャットダウンしますが、OpenShift クラスターの状態を保存します。

CDK を再起動すると、OpenShift クラスターが復元され、最後のセッションから作業を継続できます。ただし、元の start コマンドに使用したのと同じパラメーターを入力する必要があります。

CDK 設定の永続化方法の詳細は、永続設定 セクションを参照してください。

2.1.2.4. minishift delete コマンド

minishift delete コマンドは OpenShift クラスターを削除し、さらに CDK 仮想マシンもシャットダウンします。データや状態は保持されません。

2.1.3. ランタイムオプション

CDK のランタイム動作は、フラグ、環境変数、および永続的な設定オプションで制御できます。

CDK の動作を制御するには、以下の優先順位が適用されます。以下のリストの各アクションは、その下のアクションよりも優先されます。

1. フラグ セクションで指定したコマンドラインフラグを使用します。
2. 環境変数 セクションの説明どおりに、環境変数を設定します。
3. 永続設定 セクションの説明どおりに、永続的な設定オプションを使用します。
4. CDK で定義されるデフォルト値を受け入れます。

2.1.3.1. フラグ

CDK でコマンドラインフラグを使用すると、オプションを指定し、その動作を指示できます。これが最も優先されます。ほとんどのコマンドにはフラグがありますが、コマンドによってはフラグが異なります。**minishift start** コマンドの一般的なコマンドラインフラグの一部は、**cpus**、**memory**、または **vm-driver** です。

2.1.3.2. 環境変数

CDK を使用すると、環境変数で一般的に使用するコマンドラインフラグを指定できます。これを行うには、環境変数として設定するフラグに以下のルールを適用します。

1. **MINISHIFT_** を環境変数として設定するフラグの接頭辞として適用します。たとえば、**minishift start** コマンドの **vm-driver** フラグは **MINISHIFT_vm-driver** になります。
2. フラグに大文字を使用すると、上記の例では **MINISHIFT_vm-driver** は **MINISHIFT_VM-DRIVER** になります。
3. **-** を **_** に置き換えると、**MINISHIFT_VM-DRIVER** は **MINISHIFT_VM_DRIVER** になります。

環境変数を使用して、CDK コマンドのオプションを置き換えることができます。一般的な例は、使用する ISO の URL です。通常、**minishift start** コマンドの **iso-url** フラグで指定します。上記のルールを適用すると、環境変数を **MINISHIFT_ISO_URL** として設定してこの URL を指定することもできます。

注記

また、**MINISHIFT_HOME** 環境変数を使用して CDK 用の別のホームディレクトリーを選択することもできます。デフォルトでは、**minishift** コマンドはすべてのランタイム状態を **~/.minishift** に配置します。この環境変数は現在実験的であり、セマンティクスが今後のリリースで変更する可能性があります。

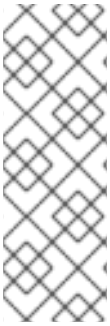
MINISHIFT_HOME を使用するには、最初に CDK を設定する際に、新しいホームディレクトリーを設定する必要があります。たとえば、これにより、Linux システムの **minishift** ホームディレクトリーが **~/.mynewdir** に設定されます。

```
$ minishift setup-cdk --minishift-home ~/.mynewdir
$ export MINISHIFT_HOME=~/.mynewdir
$ echo 'export MINISHIFT_HOME=~/.mynewdir' >> ~/.bashrc
```

2.1.3.3. 永続設定

永続設定を使用すると、環境変数を使用する方法と同様に、実際のコマンドラインフラグを指定せずに CDK の動作を制御できます。**--global** フラグを使用してグローバル設定を定義することもできます。グローバル設定は、すべてのプロファイルに適用できます。

CDK は設定ファイルを **$MINISHIFT_HOME/config/config.json** に保持します。このファイルは、各プロファイルに一般的に使用されるコマンドラインフラグを永続的に設定するために使用できます。グローバル設定ファイルは **MINISHIFT_HOME/config/global.json** で維持されます。



注記

- 永続設定は、**minishift config** サブコマンドの概要に一覧表示される、サポートされる設定オプションのセットにのみ適用できます。これは、任意のコマンドのオプションを置き換えるために使用できる環境変数とは異なります。
- デフォルトでは、**minishift** は、永続設定に起動フラグを永続的に維持します。この動作を無効にするには、**minishift config set save-start-flags false** を使用します。

2.1.3.3.1. 永続設定値の設定

永続的な設定オプションを変更する最も簡単な方法は、**minishift config set** サブコマンドを使用することです。以下に例を示します。

```
# Set default memory 4096 MB
$ minishift config set memory 4096
```

すべてのプロファイルで永続的な設定オプションを設定する最も簡単な方法は、**minishift config set --global** サブコマンドを使用することです。

たとえば、以下のように、プロファイルごとにデフォルトのメモリーを 8192 MB に設定できます。

```
$ minishift config set --global memory 8192
```

docker-env、**insecure-registry** などのコマンド呼び出しごとに複数回使用できるフラグは、**config set** コマンドで使用する場合はコンマ区切りにする必要があります。たとえば、CLI から、以下のように **insecure-registry** を使用できます。

```
$ minishift start --insecure-registry hub.foo.com --insecure-registry hub.bar.com
```

永続設定で同じレジストリーを設定するには、以下のコマンドを実行します。

```
$ minishift config set insecure-registry hub.foo.com,hub.bar.com
```

すべての永続的な設定値を表示するには、**minishift config view** サブコマンドを使用できます。

```
$ minishift config view
- memory: 4096
```

グローバル設定ファイルのすべての永続的な設定値を表示するには、**minishift config view --global** サブコマンドを使用できます。

```
$ minishift config view --global
- memory: 8192
```

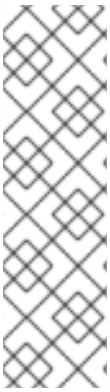
minishift config get サブコマンドを使用すると、値を 1 つ表示できます。

```
$ minishift config get memory
4096
```

2.1.3.3.2. 永続設定値の設定解除

永続的な設定オプションを削除するには、**minishift config unset** サブコマンドを使用できます。以下に例を示します。

```
$ minishift config unset memory
```



注記

ユーザー定義の値の優先度は以下のとおりです。

1. コマンドラインフラグ
2. 環境変数
3. インスタンス固有の設定
4. グローバル設定

2.1.4. 永続ボリューム

OpenShift クラスターのプロビジョニングの一環として、OpenShift クラスター用に 100 個の 永続ボリューム が作成されます。これにより、アプリケーションは 永続ボリューム要求 を作成できます。永続データの場所は、**minishift start** コマンドの **host-pv-dir** フラグで決定し、デフォルトでは CDK 仮想マシンの **/var/lib/minishift/openshift.local.pv** に設定されます。

2.1.5. HTTP/HTTPS プロキシー

HTTP/HTTPS プロキシーの背後にある場合は、Docker および OpenShift が適切に動作するようにプロキシーオプションを指定する必要があります。これには、**minishift start** 時に必要なフラグを渡します。

以下に例を示します。

```
$ minishift start --http-proxy http://YOURPROXY:PORT --https-proxy https://YOURPROXY:PORT
```

認証されたプロキシー環境では、**proxy_user** および **proxy_password** はプロキシー URI の一部である必要があります。

```
$ minishift start --http-proxy http://<proxy_username>:<proxy_password>@YOURPROXY:PORT \
        --https-proxy https://<proxy_username>:<proxy_password>@YOURPROXY:PORT
```

--no-proxy フラグを使用して、プロキシーを設定するべきでないホストをコンマ区切りリストで指定することもできます。

プロキシーオプションを使用すると、Docker デーモンと、指定されたプロキシーを使用するように OpenShift を透過的に設定します。

注記

- **minishift start** は、環境変数 **HTTP_PROXY**、**HTTPS_PROXY**、および **NO_PROXY** に従います。環境変数は小文字または大文字で指定できます。小文字の変数は大文字の変数よりも優先されます。これらの変数が設定されている場合は、対応するコマンドラインフラグで明示的に上書きしない限り、**minishift start** 時に暗黙的に使用されます。
- CDK は、プロキシーの環境変数の特殊文字をエスケープしません。特殊文字は手動でエスケープする必要があります。
- **minishift start --openshift-version** フラグを使用して、OpenShift Container Platform の特定のバージョンを要求します。**minishift openshift version list** コマンドを使用して、CDK と互換性のある OpenShift Container Platform バージョンの一覧を表示できます。

2.1.6. ネットワーク

CDK 仮想マシンは、ホストのみの IP アドレスでホストシステムに公開されます。これは、**minishift ip** コマンドで取得できます。

2.1.7. SSH を使用した CDK 仮想マシンへの接続

minishift ssh コマンドを使用して、CDK 仮想マシンと対話できます。

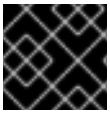
サブコマンドを使用せずに **minishift ssh** を実行して対話式シェルを開き、SSH を使用して任意のリモートマシンでコマンドを実行するのと同じ方法で CDK 仮想マシンでコマンドを実行できます。

サブコマンドで **minishift ssh** を実行して、サブコマンドを CDK 仮想マシンに直接送信し、結果をローカルシェルに戻すこともできます。以下に例を示します。

```
$ minishift ssh -- docker ps
CONTAINER    IMAGE                COMMAND              CREATED       STATUS       NAMES
71fe8ff16548 openshift3/ose:v3.11.346 "/usr/bin/openshift s" 4 minutes ago  Up 4 minutes  origin
```

2.2. CDK プロファイル

2.2.1. 概要



重要

プロファイルを使用する前に **minishift setup-cdk** を実行する必要があります。

プロファイルは、Minishift 仮想マシンのインスタンスと、そのすべての設定と状態です。プロファイル機能により、これらの分離された Minishift インスタンスを作成および管理できます。

各 CDK プロファイルは、独自の設定 (メモリー、CPU、ディスクサイズ、アドオンなど) で作成され、他のプロファイルには依存しません。**cpus**、**memory** などの特定の設定が確実にすべてのプロファイルに適用されるようにするには、環境変数の使用 を参照してください。

active プロファイルは、グローバル **--profile** フラグが使用されていない限り、すべてのコマンドが実行されるプロファイルです。**minishift profile list** コマンドを使用して、アクティブなプロファイルを確認できます。**--profile** フラグを使用すると、アクティブでないプロファイルに対して単一のコマンドを実行できます。たとえば、**minishift --profile profile-demo console** コンソールは、指定された **profile-demo** プロファイルの OpenShift コンソールを開きます。

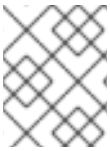
--profile フラグには、アクティブなプロファイルを一覧表示、削除、および設定するためのコマンドがあります。これらのコマンドは、以下のセクションで説明します。

警告

プロファイルは相互に独立していますが、それらは ISO、**oc** バイナリー、およびコンテナーイメージについて同じキャッシュを共有します。**minishift delete --clear-cache** は、この理由によりすべてのプロファイルに影響します。注意して **--clear-cache** を使用することが推奨されます。

2.2.2. プロファイルの作成

新しいプロファイルを作成する方法は 2 つあります。



注記

プロファイル名は英数字のみを使用できます。アンダースコア (_) およびハイフン (-) は区切り文字として使用できます。

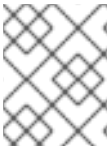
2.2.2.1. --profile フラグの使用

--profile フラグを指定して **minishift start** コマンドを実行すると、プロファイルが存在しない場合に作成されます。以下に例を示します。

```
$ minishift --profile profile-demo start
-- Starting profile 'profile-demo'
-- Check if deprecated options are used ... OK
```

```
-- Checking if https://github.com is reachable ... OK
-- Checking if requested OpenShift version 'v3.11.0' is valid ... OK
-- Checking if requested OpenShift version 'v3.11.0' is supported ... OK
-- Checking if requested hypervisor 'hyperkit' is supported on this platform ... OK
-- Checking if hyperkit driver is installed ...
   Driver is available at /usr/local/bin/docker-machine-driver-hyperkit
   Checking for setuid bit ... OK
-- Checking the ISO URL ... OK
-- Checking if provided oc flags are supported ... OK
-- Starting the OpenShift cluster using 'hyperkit' hypervisor ...
-- Minishift VM will be configured with ...
   Memory:    4 GB
   vCPUs :    2
   Disk size: 20 GB
-- Starting Minishift VM .....
```

プロファイル設定のワークフローの例 も参照してください。



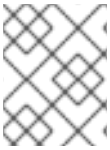
注記

minishift start を介して CDK インスタンスが正常に起動すると、プロファイルは自動的にアクティブなプロファイルになります。

2.2.2.2. profile set コマンドの使用

プロファイルを作成する他のオプションは、**profile set** コマンドを使用します。指定したプロファイルが存在しない場合は、暗黙的に作成されます。

```
$ minishift profile set demo
Profile 'demo' set as active profile
```



注記

デフォルトのプロファイルは **minishift** です。これはデフォルトで存在し、作成する必要はありません。

2.2.3. プロファイルの一覧表示

minishift profile list コマンドを使用して、既存プロファイルの一覧を表示できます。出力にアクティブなプロファイルが強調表示されているようにすることもできます。

```
$ minishift profile list
- minishift     Running      (Active)
- profile-demo  Does Not Exist
```

2.2.4. プロファイルの切り替え

プロファイルを切り替えるには、**minishift profile set** コマンドを使用します。

```
$ minishift profile set profile-demo
Profile 'profile-demo' set as active profile
```

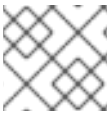
注記

いつでもアクティブにできるプロファイルは1つだけです。

2.2.5. プロファイルの削除

プロファイルを削除するには、以下を実行します。

```
$ minishift profile delete profile-demo
You are deleting the active profile. It will remove the VM and all related artifacts. Do you want to
continue [y/N]?: y
Deleted:  /Users/user/.minishift/profiles/profile-demo
Profile 'profile-demo' deleted successfully
Switching to default profile 'minishift' as the active profile.
```



注記

デフォルトのプロファイル **minishift** は削除できません。

2.2.6. プロファイル設定のワークフローの例

新しいプロファイルを作成し、その 永続的な設定 を行うには、2 つのオプションがあります。最初のオプションは、**profile set** コマンドを使用して、アクティブプロファイルを作成して、新しいプロファイルを暗黙的に作成することです。プロファイルがアクティブになったら、**minishift config** コマンドを実行します。最後に、インスタンスを起動します。

```
$ minishift profile set profile-demo
$ minishift config set memory 8GB
$ minishift config set cpus 4
$ minishift addon enable anyuid
$ minishift start
```

または、**--profile** フラグを使用して、ターゲットプロファイルを明示的に指定する一連のコマンドを実行することです。

```
$ minishift --profile profile-demo config set memory 8GB
$ minishift --profile profile-demo config set cpus 4
$ minishift --profile profile-demo addon enable anyuid
$ minishift --profile profile-demo start
```

2.3. イメージのキャッシュ

2.3.1. 概要

OpenShift クラスターのプロビジョニングを迅速化し、ネットワークトラフィックを最小限に抑えるために、コンテナーイメージをホストにキャッシュすることができます。これらのイメージは、要求に応じて明示的に、または **minishift start** の時に暗黙的に、実行中の Docker デーモンにインポートできます。以下のセクションでは、イメージのキャッシュとその設定について詳しく説明します。

注記

イメージがキャッシュされる形式は、CDK バージョン 3.3.0 で変更されています。3.3.0 より前のバージョンでは、イメージは tar ファイルとして保存されていました。3.3.0 の時点で、イメージは OCI イメージ形式 に保存されます。

CDK 3.3.0 より前のイメージのキャッシュを使用した場合は、キャッシュを再作成する必要があります。廃止された 3.3.0 前のイメージを削除する場合は、以下を実行してキャッシュをクリアできます。

```
$ minishift delete --clear-cache
```

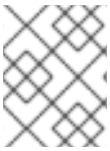
2.3.2. 明示的なイメージキャッシュ

CDK は、イメージキャッシュの動作を制御するために、**image** コマンドとサブコマンドを提供します。CDK 仮想マシンの Docker デーモンからイメージをエクスポートしてインポートするには、**minishift image export** と **minishift image import** を使用します。

2.3.2.1. 単一イメージのインポートおよびエクスポート

CDK 仮想マシンの実行後、イメージは Docker デーモンから明示的にエクスポートできます。

```
$ minishift image export <image-name-0> <image-name-1> ...
Pulling image <image-name-0> .. OK
Exporting <image-name-0>. OK
Pulling image <image-name-1> .. OK
Exporting <image-name-2>. OK
```



注記

Docker デーモンでは利用できないイメージは、ホストにエクスポートされる前にプルされます。

以前キャッシュされたイメージをインポートするには、**minishift image import** コマンドを使用します。

```
$ minishift image import <image-name-0> <image-name-1> ...
Importing <image-name-0> . OK
```

2.3.2.2. キャッシュされたイメージの一覧表示

minishift image list コマンドは、現在キャッシュされたイメージまたは CDK Docker デーモンで利用可能なイメージの一覧を表示します。

ホストで現在キャッシュされているイメージを表示するには、次のコマンドを実行します。

```
$ minishift image list
registry.access.redhat.com/openshift3/ose-docker-registry:v3.11.346
registry.access.redhat.com/openshift3/ose-haproxy-router:v3.11.346
registry.access.redhat.com/openshift3/ose:v3.11.346
```

Docker デーモンで利用可能なイメージを表示するには、以下を実行します。

```
$ minishift image list --vm
registry.access.redhat.com/openshift3/ose-deployer:v3.11.346
registry.access.redhat.com/openshift3/ose-docker-registry:v3.11.346
registry.access.redhat.com/openshift3/ose-haproxy-router:v3.11.346
registry.access.redhat.com/openshift3/ose-pod:v3.11.346
registry.access.redhat.com/openshift3/ose-web-console:v3.11.346
registry.access.redhat.com/openshift3/ose:v3.11.346
```

2.3.2.3. キャッシュされたイメージ名の永続化

image export コマンドまたは **image import** コマンドの一部としてイメージ名を明示的に入力しなくても、インポートおよびエクスポートを行うイメージ名の一覧を永続設定に保存できます。

minishift image cache-config view を使用して、現在設定されているイメージの一覧を表示し、**minishift image cache-config add** を使用してイメージを一覧に追加します。

```
$ minishift image cache-config view
$ minishift image cache-config add alpine:latest busybox:latest
$ minishift image cache-config view
alpine:latest
busybox:latest
```

一覧からイメージを削除するには、**minishift image cache-config remove** を使用します。

```
$ minishift image cache-config remove alpine:latest
$ minishift image cache-config view
busybox:latest
```

イメージ名を永続設定に保存すると、引数なしで **minishift image export** および **minishift image import** を実行できます。

2.3.2.4. すべてのイメージのエクスポートおよびインポート

--all フラグを使用して、すべてのイメージをエクスポートおよびインポートできます。export コマンドの場合は、現在 Docker デーモンで利用可能なすべてのイメージがホストにエクスポートされます。import コマンドの場合は、ローカル CDK キャッシュで利用可能なすべてのイメージが CDK 仮想マシンの Docker デーモンにインポートされることを意味します。

警告

すべてのイメージのエクスポートおよびインポートには時間がかかる可能性があり、ローカルにキャッシュされたイメージにはかなりの量のディスク容量が使用される可能性があります。この機能は注意して使用することが推奨されます。

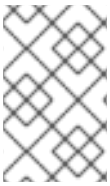
2.3.3. 暗黙的なイメージキャッシュ

CDK では、イメージのキャッシュがデフォルトで有効になっています。これは、**minishift start** コマンドを初めて完了した後にバックグラウンドプロセスで発生します。イメージが **$MINISHIFT_HOME/cache/image** にキャッシュされると、連続した CDK 仮想マシンの作成により、

これらのキャッシュされたイメージが使用されます。

この機能を無効にするには、**minishift config set** コマンドを使用して、永続設定で **image-caching** プロパティーを無効にする必要があります。

```
$ minishift config set image-caching false
```



注記

暗黙的なイメージのキャッシュにより、**cache-images** 設定オプションごとに指定されるように、必要な OpenShift イメージをキャッシュされたイメージの一覧に透過的に追加します。キャッシュされたイメージ名の永続化 を参照してください。

注記

イメージのエクスポートのバックグラウンドプロセスを実行するたびに、ログファイルが **$MINISHIFT_HOME/logs** の下に生成されます。これは、エクスポートの進捗を検証するために使用できます。

image-caching を **true** に設定するか、**minishift config unset** を使用して設定をすべて削除して、OpenShift イメージのキャッシュを再度有効にできます。

```
$ minishift config unset image-caching
```

2.3.4. ローカルキャッシュからのイメージの削除

イメージは **$MINISHIFT_HOME/cache/images** の下にキャッシュされ、SHA256 ブロブとして、イメージ名で SHA を照合する際の詳細情報が含まれるインデックスとして保存されます。

ローカルキャッシュからイメージを削除するには、**minishift image delete** コマンドを使用します。

```
$ minishift image delete <image-name-0> <image-name-1> ...
Deleting <image-name-0> from the local cache OK
Deleting <image-name-1> from the local cache OK
```

2.4. アドオン

2.4.1. 概要

CDK を使用すると、**cluster up** により提供される vanilla OpenShift 設定をアドオンメカニズムで拡張できます。

アドオンは、**.addon** 拡張子が付いたテキストファイルを含むディレクトリーです。このディレクトリーには、JSON テンプレートファイルなどの他のリソースファイルを含めることもできます。ただし、アドオンごとに 1 つの **.addon** ファイルのみが許可されます。

以下の例は、アドオンの名前と説明、追加のメタデータ、および適用する実際のアドオンコマンドなど、アドオンファイルの内容を示しています。

例: anyuid アドオン定義ファイル

```
# Name: anyuid                                                                    (1)
# Description: Allows authenticated users to run images under a non pre-allocated UID   (2)
# Required-Vars: ACME_TOKEN                                                       (3)
# OpenShift-Version: >3.6.0                                                       (4)
# Minishift-Version: >1.22.0                                                      (5)

mytoken := oc sa get-token oauth-client                                           (6)

oc new-app -p OPENSHIFT_OAUTH_CLIENT_SECRET=#{mytoken}                            (7)
oc adm policy add-scc-to-group anyuid system:authenticated                        (8)
```

(1) (必須) アドオンの名前。

(2) (必須) アドオンの説明。

(3) (必要に応じて) 必要な挿入変数のコンマ区切りリスト。変数の挿入 を参照してください。

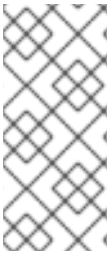
(4) (必要に応じて) 特定のアドオンの実行に必要な OpenShift バージョン。OpenShift バージョンセマンティクス を参照してください。

(5) (必要に応じて) 特定のアドオンの実行に必要な CDK バージョン。最小バージョンセマンティクス を参照してください。

(6) 実際のアドオンコマンドを実行し、その出力を変数に保存します。内部変数 を参照してください。

(7) 実際の追加コマンドです。この場合、コマンドは **mytoken** 変数の値を使用して **oc** バイナリーを実行します。

(8) 実際の追加コマンドです。この場合、コマンドは **oc** バイナリーを実行します。



注記

- # 文字で始まるコメント行は、ファイルのどこにでも挿入できます。
- **!** 文字で始まるコマンドは、実行の失敗を無視します。そのため、アドオンがべき等になります (つまり、アドオンの最終的な動作を変更せずに、コマンドは複数回実行できます)。

有効なアドオンは、初期クラスタープロビジョニングが正常に完了した直後に、**minishift start** 時に適用されます。

2.4.2. OpenShift-Version セマンティクス

アドオンメタデータの一部として、アドオンを適用するために実行中の OpenShift バージョンを指定できます。これを実行するには、任意の **OpenShift-Version** メタデータフィールドを指定できます。セマンティクスは以下のとおりです。

>3.6.0	アドオンを適用するには、3.6.0 を超える OpenShift バージョンを実行する必要があります。

>=3.6.0	アドオンを適用するには、OpenShift バージョン 3.6.0 以降を実行している必要があります。
3.6.0	アドオンを適用するには、OpenShift バージョン 3.6.0 を実行する必要があります。
>=3.5.0, <3.8.0	OpenShift バージョンは 3.5.0 以上 3.8.0 未満にする必要があります。
>=3.5.0, <=3.8.0	OpenShift バージョンは 3.5.0 以上 3.8.0 以下にする必要があります。

注記

- metadata フィールド **OpenShift-Version** がアドオンヘッダーで指定されていない場合、アドオンはすべてのバージョンの OpenShift に対して適用できます。
- **OpenShift-Version** では、バージョンの形式は <major>.<minor>.<patch> のみをサポートします。

2.4.3. Minishift-Version セマンティクス

アドオンメタデータの一部として、アドオンを適用するために実行する必要のある Minishift バージョンを指定できます。これを行うには、オプションの **Minishift-Version** メタデータフィールドを指定できます。セマンティクスは以下のとおりです。

>1.22.0	アドオンを適用するには、1.22.0 より大きい minishift バージョンを実行する必要があります。
>=1.22.0	アドオンを適用するには、1.22.0 以上の minishift バージョンを実行する必要があります。
1.22.0	アドオンを適用するには、minishift バージョン 1.22.0 を実行する必要があります。
>=1.21.0, <1.25.0	minishift バージョンは 1.21.0 以上でなければなりませんが、1.25.0 未満です。
>=1.22.0, <=1.25.0	minishift バージョンは 1.22.0 以上でなければなりませんが、1.25.0 以下である必要があります。

注記

- アドオンヘッダーで metadata フィールド **Minishift-Version** が指定されていない場合、アドオンは CDK のバージョンに対して適用できます。
- **Minishift-Version** では、バージョンの形式は <major>.<minor>.<patch> のみをサポートします。

2.4.4. アドオン依存関係の定義

アドオン依存関係は、**Depends-On** メタデータフィールドを使用して定義できます。複数の依存関係は、アドオン名のコンマ区切りリストを使用して定義できます。

例: アドオン定義ファイルのアドオン依存関係の定義

```
# Name: example
# Description: Shows the use of the Depends-On metadata field
# Depends-On: anyuid, admin-user

echo Depends on the anyuid and admin-user add-ons, requiring them to be installed.
```

2.4.5. アドオンコマンド

本セクションでは、アドオンファイルに追加できるコマンドを説明します。CDK アドオンのドメイン固有の小さな言語を形成します。

ssh

ssh でアドオンコマンドを起動する場合は、CDK 管理の仮想マシン内でコマンドを実行できます。これは、**minishift ssh** の実行と類似しており、仮想マシンで任意のコマンドを実行します。**minishift ssh** コマンドの使用方法は、SSH を使用した CDK 仮想マシンへの接続 を参照してください。

oc

oc でアドオンコマンドを起動する場合は、ホストにキャッシュされた **oc** バイナリーを使用して、指定された **oc** コマンドを実行します。これは、コマンドラインから **oc --as system:admin ...** を実行するのと似ています。

注記

oc コマンドは、system:admin として実行されます。

openshift

openshift でアドオンコマンドを起動する場合は、これが OpenShift コンテナーに存在する **oc** バイナリーを使用してコマンドを実行します。つまり、ファイルパラメーターやその他のシステム固有のパラメーターは、ホストではなくコンテナーの環境と一致する必要があります。

Docker

docker でアドオンコマンドを起動すると、CDK 仮想マシン内の Docker デーモンに対して **docker** コマンドを実行します。これは、単一ノードの OpenShift クラスターを実行しているのと同じデーモンです。これは、ホストで **eval $(minishift docker-env)** を実行し、**docker** コマンドを実行するのと似ています。**minishift docker-env** も参照してください。

echo

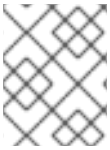
echo でアドオンコマンドを起動すると、**echo** コマンドの後の引数がコンソールに出力されます。これは、アドオンの実行中に追加のフィードバックを提供するために使用できます。

sleep

sleep でアドオンコマンドを起動すると、指定した秒数だけ待機します。これは、特定のリソースをクエリーする前に **oc** などのコマンドに数秒かかる可能性がある場合に便利です。

cat

cat でアドオンコマンドを起動する場合は、**cat** コマンドに続く引数を有効なファイルパスにする必要があります。これにより、有効なファイルの内容をコンソールに出力するか、ファイルパスが無効であればエラーメッセージが表示されます。これは、別のコマンドでファイルの内容を使用するのに便利です。



注記

undefined コマンドの使用を試みると、アドオンが解析される際にエラーが発生します。

2.4.6. 変数の挿入

CDK では、アドオンコマンド内で変数を使用できます。変数の形式は **#{<variable-name>}** です。以下の例は、OpenShift ルーティング接尾辞が **openshift** コマンドに挿入され、OpenShift レジストリーのセキュリティー保護の一環として新規証明書を作成する方法を示しています。使用された変数 **#{routing-suffix}** は組み込みアドオン変数の一部です。

例: routing-suffix 変数の使用

```
$ openshift admin ca create-server-cert \
  --signer-cert=/var/lib/origin/openshift.local.config/master/ca.crt \
  --signer-key=/var/lib/origin/openshift.local.config/master/ca.key \
  --signer-serial=/var/lib/origin/openshift.local.config/master/ca.serial.txt \
  --hostnames='docker-registry-default.#{routing-suffix},docker-
registry.default.svc.cluster.local,172.30.1.1' \
  --cert=/etc/secrets/registry.crt \
  --key=/etc/secrets/registry.key
```

2.4.6.1. 組み込み変数

常に挿入に使用できる組み込み変数が複数存在しています。以下の表は、これらの変数を示しています。

表2.1 サポートされる組み込みアドオン変数

変数	説明
ip	CDK 仮想マシンの IP。
routing-suffix	アプリケーションの OpenShift ルーティング接尾辞。
addon-name	現在のアドオンの名前。
user	SSH を介してコマンドを実行するために CDK が使用するユーザー。

2.4.6.2. 動的変数

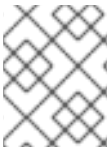
minishift addons apply コマンドおよび **minishift start** コマンドも適用されます。これにより、**--addon-env** フラグも適用され、挿入用の変数を動的に渡すことができます。以下に例を示します。

```
$ minishift addons apply --addon-env PROJECT_USER=user acme
```

--addon-env フラグは、挿入用に複数の変数を定義するために複数回指定できます。

動的変数の指定は、永続的な設定値の設定 と併用することもできます。

```
$ minishift config set addon-env PROJECT_USER=user
$ minishift addons apply acme
```



注記

minishift config set コマンドを使用する場合は、複数の変数をコンマ区切りにする必要があります。

また、アドオンが適用される際に、環境変数の値で変数を動的に挿入する可能性もあります。この場合には、変数値の前に **env** を付ける必要があります。

```
$ minishift config set addon-env PROJECT_USER=env.USER      (1)
$ minishift addons apply acme                               (2)
```

(1) **env** 接頭辞を使用すると、#{PROJECT_USER}を env.USER に置き換える代わりに、環境変数 **USER** の値が使用されます。環境変数が設定されていないと、挿入は行われません。

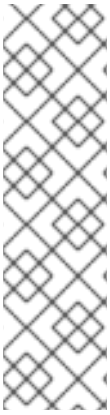
(2) アドオンが適用されると、アドオンコマンドの **#{PROJECT_USER}** が環境変数 **USER** の値に置き換えられます。

アドオンの開発者は、変数名を **Required-Vars** メタデータヘッダーに追加することで、アドオンの適用時に変数の値が提供されるようにすることができます。複数の変数はコンマ区切りにする必要があります。

```
# Name: acme
# Description: ACME add-on
# Required-Vars: PROJECT_USER
```

Var-Defaults メタデータヘッダーを使用して変数のデフォルト値を指定することもできます。**Var-Defaults** は、**<key>=<value>** の形式で指定する必要があります。複数のデフォルトのキーと値のペアはコンマで区切る必要があります。

```
# Name: acme
# Description: ACME add-on
# Required-Vars: PROJECT_USER
# Var-Defaults: PROJECT_USER=user
```



注記

- **=** と **,** はメタ文字で、キーまたは値の一部として使用できません。
- **Var-Defaults** キーの値として NULL、Null、または null が指定されている場合は、空の値が設定されます。以下に例を示します。

  ```
  # Var-Defaults: PROJECT_USER=null
  ```

- **--addon-env** を使用するか、**minishift config set addon-env** を設定してコマンドラインで指定される変数値は、**Var-Defaults** よりも優先されます。

2.4.6.3. 内部変数

アドオンは変数の定義を許可します。これらの変数は、アドオンコマンド の出力を保存するために使用できます。

たとえば、**oc** コマンド出力を **mytoken** 変数に保存し、以下のように後続のアドオンコマンドで使用できます。

```
mytoken := oc sa get-token oauth-client
oc new-app -p OPENSHIFT_OAUTH_CLIENT_SECRET=#{mytoken}
```

2.4.7. デフォルトのアドオン

CDK は、開発を支援する共通の OpenShift カスタマイズを提供する一連のビルトインアドオンを提供します。**minishift setup-cdk** では、Minishift は自動的に **xpaas** アドオン、**anyuid** アドオン、および **admin-user** アドオンをインストールして有効にします。デフォルトのアドオンをインストールするには、以下を実行します。

```
$ minishift addons install --defaults
```

このコマンドは、デフォルトのアドオンインストールディレクトリー **$MINISHIFT_HOME/addons** に展開します。インストールされたアドオンの一覧を表示するには、以下のコマンドを実行します。

```
$ minishift addons list --verbose=true
```

このコマンドは、インストールされているアドオンの一覧を出力します。少なくとも、**anyuid** アドオンが表示されるはずです。これは、事前に割り当てられた UID を使用しないイメージを実行できるようにする重要なアドオンです。デフォルトでは、これは OpenShift では使用できません。

表2.2 デフォルトのアドオン

アドオン名	説明
anyuid	デフォルトのセキュリティーコンテキスト制約を変更し、Pod が任意の UID で実行できるようにします。
admin-user	admin という名前のユーザーを作成し、**cluster-admin** ロールを割り当てます。
che	Eclipse Che 統合開発環境のインスタンスを実行します。
eap-cd	JBoss EAP CD テンプレートをインポートします。
htpasswd-identity-provider	ユーザーは、OpenShift インスタンスのデフォルトのログインユーザー名およびパスワードを変更および追加できます。
registry-route	OpenShift レジストリーのエッジ終端ルートを作成します。
xpaas	xPaaS テンプレートをインポートします。

注記

eap-cd アドオンは、JBoss EAP CD 19 のテンプレートを提供します。最新バージョンの JBoss EAP CD を使用するには、アップストリームの **eap-cd** アドオンを使用します。アップストリームアドオンのインストール方法は、コミュニティーによるアドオン を参照してください。

2.4.7.1. コミュニティーによるアドオン

複数のデフォルトアドオンの他に、CDK 用にコミュニティーで開発したアドオンが多数あります。コミュニティーアドオンは minishift-addons リポジトリーにあります。リポジトリー のアドオンに関する情報をすべて取得できます。インストール手順は、README を参照してください。

2.4.8. アドオンのインストール

アドオンは **minishift addons install** コマンドでインストールされます。

以下の例は、アドオンのインストール方法を示しています。

例: アドオンのインストール

```
$ minishift addons install <path_to_addon_directory>
```

2.4.9. アドオンの有効化および無効化

アドオンは **minishift addons enable** コマンドで有効にされ、**minishift addons disable** コマンドで無効にされます。有効にするアドオンは、**minishift start** 時に自動的に実行されます。

以下の例は、**anyuid** アドオンを有効にして無効にする方法を示しています。

例: anyuid アドオンの有効化

```
$ minishift addons enable anyuid
```

例: anyuid アドオンの無効化

```
$ minishift addons disable anyuid
```

2.4.9.1. アドオンの優先順位

アドオンを有効にすると、優先順位を指定することもできます。これにより、アドオンが適用される順序が決定されます。

以下の例は、優先度が高い **レジストリー** アドオンを有効にする方法を示しています。

例: 優先順位を持つレジストリーアドオンの有効化

```
$ minishift addons enable registry --priority=5
```

アドオンの priority 属性は、アドオンの適用順序を決定します。デフォルトでは、アドオンの優先順位は 0 です。優先順位の値が低いアドオンが最初に適用されます。

以下の例では、**anyuid** アドオン、**registry** アドオン、**eap** アドオンは、それぞれの優先順位 0、5、および 10 で有効にされています。つまり、**anyuid** が最初に適用され、その後に **registry** が続き、最後に **eap** アドオンが適用されます。

例: 明示的な優先順位でのコマンドの出力の一覧表示

```
$ minishift addons list
- anyuid         : enabled    P(0)
- registry       : enabled    P(5)
- eap            : enabled    P(10)
```

注記

2 つのアドオンの優先順位が同じ場合は、適用される順序は判断されません。

2.4.10. アドオンの適用

アドオンは **minishift addons apply** コマンドを使用して明示的に実行できます。有効なアドオンと無効なアドオンの両方に **apply** コマンドを使用できます。1 つのコマンドで複数のアドオンを適用するには、スペースで区切られたアドオン名を指定します。

以下の例は、**anyuid** アドオンおよび **admin-user** アドオンを明示的に適用する方法を示しています。

例: anyuid アドオンおよび admin-user アドオンの適用

```
$ minishift addons apply anyuid admin-user
```

2.4.11. アドオンの削除

アドオンは **minishift addons remove** コマンドを使用して削除できます。これは **minishift addons apply** のミラーコマンドであり、アドオンが有効かどうかに関わらず、同様に使用できます。指定したアドオンがインストールされ、**<addon_name>.addon.remove** ファイルがある場合、**minishift addons remove** は、このファイルで指定されたコマンドを実行します。

1 つのコマンドで複数のアドオンを削除するには、スペースで区切られたアドオン名を指定します。以下の例は、**admin-user** アドオンを明示的に削除する方法を示しています。

例: admin-user アドオンの削除

```
$ minishift addons remove admin-user
-- Removing addon 'admin-user':.
admin user deleted
```

2.4.12. アドオンのアンインストール

アドオンは、**minishift addons uninstall** コマンドを使用してアンインストールできます。これは **minishift addons install** のミラーコマンドであり、アドオンが有効かどうかに関わらず、同様に使用できます。指定したアドオンがインストールされている場合、**minishift addons uninstall** は対応するアドオンディレクトリーを **$MINISHIFT_HOME/addons** から削除します。

以下の例は、**admin-user** アドオンを明示的にアンインストールする方法を示しています。

例: admin-user アドオンのアンインストール

```
$ minishift addons uninstall admin-user
Add-on 'admin-user' uninstalled
```

2.4.13. カスタムアドオンの作成

カスタムアドオンを作成するには、ディレクトリーを作成し、そのディレクトリーに、拡張子 **.addon** で少なくとも 1 つのテキストファイルを作成する必要があります (例: **admin-role.addon**)。

このファイルには、アドオンの一部として実行するコマンドと、**Name** および **Description** メタデータフィールドが含まれている必要があります。

以下の例は、開発者ユーザーに cluster-admin 権限を付与するアドオンの定義を示しています。

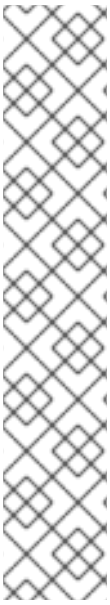
例: admin-role のアドオン定義

```
# Name: admin-role
# Description: Gives the developer user cluster-admin privileges

oc adm policy add-role-to-user cluster-admin developer
```

アドオンを定義したら、以下を実行してインストールできます。

```
$ minishift addons install <ADDON_DIR_PATH>
```



注記

複数の行でメタデータを作成することもできます。

例: 複数行の説明が含まれるアドオン定義

```
# Name: prometheus
# Description: This template creates a Prometheus instance preconfigured to gather
OpenShift and
# Kubernetes platform and node metrics and report them to admins. It is protected by
an
# OAuth proxy that only allows access for users who have view access to the
prometheus
# namespace. You may customize where the images (built from openshift/prometheus
# and openshift/oauth-proxy) are pulled from via template parameters.
# Url: https://prometheus.io/
```

CDK アドオンインストールディレクトリー **$MINISHIFT_HOME/addons** で、アドオンを直接編集することもできます。アドオンにエラーがある場合は、**addons** コマンドの実行時には表示されませんが、これは **minishift start** プロセス中は適用されないことに注意してください。

アドオンの削除手順を提供するには、拡張子 **.addon.remove** でテキストファイルを作成できます (例: **admin-user.addon.remove**)。**.addon** ファイルと同様に、**Name** および **Description** メタデータフィールドが必要です。**.addon.remove** ファイルが存在する場合は、**remove** コマンドで適用できます。

2.5. ホストフォルダー

2.5.1. 概要

ホストフォルダーは、ホストと CDK 仮想マシンとの間で共有されるホストのディレクトリーです。これにより、ホストと仮想マシンの間の 2 方向ファイル同期が可能になります。以下のセクションでは、**minishift hostfolder** コマンドの使用方法を説明します。

2.5.2. minishift hostfolder コマンド

CDK は、ホストフォルダーの一覧表示、追加、マウント、マウント解除、および削除を行う **minishift hostfolder** コマンドを提供します。**hostfolder** コマンドを使用して、複数の共有フォルダーをカスタム指定のマウントポイントにマウントできます。

注記

現在、CIFS および SSHFS ベースのホストフォルダーがサポートされています。

2.5.2.1. 前提条件

2.5.2.1.1. SSHFS

SSHFS は、ホストフォルダーを共有するデフォルトのテクノロジーです。これは前提条件なしで機能します。

2.5.2.1.2. CIFS

ホストフォルダーは、CIFS を使用して共有できます。Windows、macOS、および Linux で CIFS を使用できます。

macOS では、**System Preferences > Sharing** で CIFS ベースの共有を有効にできます。詳細は、How to connect with File Sharing on your Mac を参照してください。

Linux の場合は、ディストリビューション固有の指示に従って Samba をインストールします。Red Hat Enterprise Linux で CIFS の Samba 実装を設定する方法は RHEL の Samba ファイルおよび印刷サーバー を参照してください。

2.5.2.2. ホストフォルダーの表示

minishift hostfolder list コマンドは、定義されたホストフォルダーの概要、その名前、マウントポイント、リモートパス、および現在マウントされているかどうかを示します。

出力例は以下のようになります。

```
$ minishift hostfolder list
Name    Type    Source              Mountpoint      Mounted
test    sshfs   /Users/user/test    /mnt/sda1/test  N
```

この例では、**/Users/user/test** を CDK 仮想マシンの **/mnt/sda1/test** にマウントする、**test** という名前の SSHFS ベースのホストフォルダーがあります。現在、共有はマウントされていません。

2.5.2.3. ホストフォルダーの追加

minishift hostfolder add コマンドを使用すると、新しいホストフォルダーを定義できます。

使用する構文は、ホストフォルダーのタイプによって異なります。タイプに関係なく、非対話型と対話的な設定のいずれかを選択できます。デフォルトは非対話的です。**--interactive** フラグを指定すると、対話形式の設定モードを選択できます。

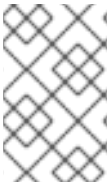
以下のセクションでは、CIFS および SSHFS のホストフォルダーを設定する例を紹介します。

2.5.2.3.1. CIFS

CIFS ベースのホストフォルダーの追加

```
$ minishift hostfolder add -t cifs --source //192.168.99.1/MYSHARE --target /mnt/sda1/myshare --
options username=user,password=mysecret,domain=MYDOMAIN myshare
```

上記のコマンドにより、**myshare** という名前の新しいホストフォルダーが作成されます。ホストフォルダーのソースは、必要な UNC パス **//192.168.99.1/MYSHARE** です。ターゲットまたはマウントポイントは、CDK 仮想マシン内の **/mnt/sda1/myshare** です。**--options** フラグを使用すると、ホストフォルダータイプ固有のオプションを、コンマ区切りのキーと値のペアのリストを使用して指定できます。CIFS ホストフォルダーの場合、必要なオプションは CIFS 共有のユーザー名およびパスワードです。**--options** フラグを使用して、共有のドメインを指定することもできます。多くの場合、これは省略できますが、Windows では、アカウントが Microsoft アカウントにリンクされていると、Microsoft アカウントメールアドレスをユーザー名として使用し、**$env:COMPUTERNAME** がドメインとして表示されるマシン名を使用する必要があります。



注記

Windows ホストでは、**minishift hostfolder add** コマンドは、**users-share** オプションも提供します。このオプションを指定すると、UNC パスを指定せず、**C:\Users** が想定されます。

2.5.2.3.2. SSHFS

SSHFS ベースのホストフォルダーの追加

```
$ minishift hostfolder add -t sshfs --source /Users/user/myshare --target /mnt/sda1/myshare myshare
```

SSHFS ベースのホストフォルダーの場合は、ホストフォルダーのソースとターゲットのみを指定する必要があります。

注記

ホストフォルダーを追加する際に、**--options** フラグを使用してマウント中に使用されるタイプ固有のオプションを指定できます。以下に例を示します。

```
$ minishift hostfolder add -t sshfs --source ~/myshare/ --target /mnt/sda1/myshare --
options "-o compression=no" myshare
```

2.5.2.3.3. インスタンス固有のホストフォルダー

デフォルトでは、ホストフォルダーの定義は永続的であり、他の 永続的な設定 オプションと類似します。つまり、これらのホストフォルダー定義は削除後も CDK 仮想マシンの再作成後も維持されます。

特定の CDK インスタンスに対してのみホストフォルダーを定義しないといけない場合があります。これを実行するには、**minishift hostfolder add** コマンドの **--instance-only** フラグを使用できます。**--**

instance-only フラグで作成されるホストフォルダー定義は、**minishift delete** 時に他のインスタンス固有の状態と共に削除されます。

2.5.2.4. ホストフォルダーのマウント

ホストフォルダーを追加したら、**minishift host folderer mount** コマンドを使用して、その名前でホストフォルダーをマウントします。

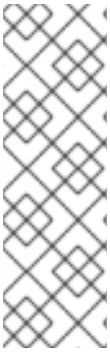
```
$ minishift hostfolder mount myshare
```

以下を実行し、ホストフォルダーがマウントされていることを確認できます。

```
$ minishift hostfolder list
Name       Mountpoint          Remote path              Mounted
myshare    /mnt/sda1/myshare   //192.168.99.1/MYSHARE   Y
```

マウントされたホストフォルダーの実際のコンテンツを一覧表示できます。

```
$ minishift ssh "ls -al /mnt/sda1/myshare"
```



注記

SSHFS ベースのホストフォルダーをマウントすると、SFTP サーバープロセスがホストの 2022 ポートで起動します。ネットワークおよびファイアウォールの設定により、このポートを開くことができることを確認してください。このポートを設定する必要がある場合は、たとえばキー **hostfolders-sftp-port** を使用して CDK の 永続設定 を使用できます。以下に例を示します。

```
$ minishift config set hostfolders-sftp-port 2222
```

2.5.2.4.1. ホストフォルダーの自動マウント

ホストフォルダーは、**minishift start** するたびに自動的にマウントできます。自動マウントを設定するには、CDK 設定ファイルで **hostfolders-automount** オプションを設定する必要があります。

```
$ minishift config set hostfolders-automount true
```

hostfolders-automount オプションを設定すると、CDK は、**minishift start** 時に定義されたすべてのホストフォルダーのマウントを試みます。

2.5.2.5. ホストフォルダーのマウント解除

minishift hostfolder umount コマンドを使用して、ホストフォルダーのマウントを解除します。

```
$ minishift hostfolder umount myshare

$ minishift hostfolder list
Name       Mountpoint          Remote path              Mounted
myshare    /mnt/sda1/myshare   //192.168.99.1/MYSHARE   N
```

2.5.2.6. ホストフォルダーの削除

minishift hostfolder remove コマンドを使用して、ホストフォルダーの定義を削除します。

```
$ minishift hostfolder list
Name        Mountpoint            Remote path              Mounted
myshare     /mnt/sda1/myshare     //192.168.1.82/MYSHARE   N

$ minishift hostfolder remove myshare

$ minishift hostfolder list
No host folders defined
```

2.6. 静的 IP アドレスの割り当て

2.6.1. 概要

ほとんどのハイパーバイザーは、DHCP を使用して IP が割り当てられる際のリース時間の延長に対応していません。これにより、再起動後に仮想マシンに新しい IP が割り当てられる可能性があります。古い IP 用に生成されたセキュリティー証明書と競合するためです。これにより、**minishift delete** に続いて **minishift start** を実行して新規インスタンスが設定されるまで CDK は完全に使用できなくなります。

これを防ぐために、CDK には、静的 IP アドレスを仮想マシンに設定する機能が含まれます。これにより、再起動間で IP アドレスが変更されなくなります。ただし、IP アドレスの解決方法により、現時点ではすべてのドライバープラグインで機能しません。

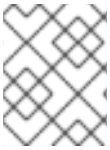
注記

- CDK 仮想マシンへの静的 IP アドレスの割り当ては、Hyper-V でのみ正式にサポートされています。
- KVM ドライバープラグインを使用する際に、CDK 仮想マシンに静的 IP アドレスを割り当てることはできません。

2.6.2. Hyper-V に IP アドレスを割り当て

Internal Virtual Switch for Hyper-V では DHCP が提供するオプションはないため、IP アドレスは別の方法で提供する必要があります。Data Exchange Service for Hyper-V を使用して、起動時に IP アドレスを割り当てる機能が提供されます。

この機能を有効にするには、NAT を使用して仮想スイッチ を作成する必要があります。



注記

WinNAT は、ホストごとに 1 つの NAT ネットワークに制限されます。機能および制限の詳細は、WinNAT 機能および制限に関するブログ を参照してください。

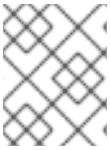
以下のコマンドは、内部仮想スイッチ MyInternal で使用する IP アドレスの割り当てを試みます。

```
PS> minishift.exe config set hyperv-virtual-switch "MyInternal"
PS> minishift.exe start `
  --network-ipaddress 192.168.1.10 `
  --network-gateway 192.168.1.1 `
  --network-nameserver 8.8.8.8
```

–

DockerNAT ネットワークを使用する場合は、正しい NAT ネットワークを設定し、予想される範囲の IP を割り当てる必要があります。

```
PS> New-NetNat -Name SharedNAT -InternalIPInterfaceAddressPrefix 10.0.75.1/24
PS> minishift.exe config set hyperv-virtual-switch "DockerNAT"
PS> minishift.exe start `
  --network-ipaddress 10.0.75.128 `
  --network-gateway 10.0.75.1 `
  --network-nameserver 8.8.8.8
```



注記

有効なゲートウェイおよびネームサーバーを指定してください。実行に失敗すると、接続性に問題が生じます。

2.6.3. 固定 IP アドレスの設定

重要

ドライバーのプラグインは DHCP の使用により IP アドレスを決定するため、この機能は KVM ではサポートされていません。

以下のコマンドは、固定で割り当てられた IP アドレスを設定します。

```
$ minishift ip --set-static
```

動的割り当てを使用する場合は、以下を使用できます。

```
$ minishift ip --set-dhcp
```

注記

minishift config set static-ip false を使用して、サポートされるハイパーバイザーの静的 IP アドレスを自動的に割り当てます。

2.7. CDK DOCKER デーモン

2.7.1. 概要

単一の仮想マシンで OpenShift を実行する場合は、他の Docker のユースケース向けに CDK によって管理される Docker デーモンを再利用できます。CDK と同じ Docker デーモンを使用することで、ローカル開発を迅速化できます。

2.7.2. コンソールの設定

CDK Docker デーモンを再利用するようにコンソールを設定するには、以下の手順に従います。

1. マシンに Docker クライアントバイナリーがインストールされていることを確認します。オペレーティングシステム用の特定のバイナリーインストールに関する詳細は、Docker インストール サイトを参照してください。

2. **minishift start** コマンドを使用して CDK を起動します。

3. **minishift docker-env** コマンドを実行して、Docker クライアントを設定するためにシェルに入力する必要があるコマンドを表示します。コマンド出力は、OS とシェルのタイプによって異なります。

```
$ minishift docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.101:2376"
export DOCKER_CERT_PATH="/Users/user/.minishift/certs"
export DOCKER_API_VERSION="1.24"
# Run this command to configure your shell:
# eval $(minishift docker-env)
```

4. 以下のコマンドを実行して接続をテストします。

```
$ docker ps
```

成功すると、シェルは実行中のコンテナーの一覧を出力します。

2.8. 実験的な機能

2.8.1. 概要

今後の機能および実験的機能に早期にアクセスしたい場合は、環境変数 **MINISHIFT_ENABLE_EXPERIMENTAL** を設定できます。これにより、追加の機能フラグが利用できるようになります。

```
$ export MINISHIFT_ENABLE_EXPERIMENTAL=y
```

重要

実験的機能は公式にサポートされておらず、破損または予期せぬ挙動が発生する可能性があります。これらの機能に関するご意見については、Minishift コミュニティー にご連絡ください。

2.8.2. 実験的な **oc cluster up** フラグの有効化

デフォルトでは、CDK は CDK CLI のすべての **oc cluster up** フラグを公開しません。

MINISHIFT_ENABLE_EXPERIMENTAL 環境変数を設定して、**minishift start** コマンドで以下のオプションを有効にします。

extra-clusterup-flags
CDK CLI で直接公開されない **oc cluster up** にフラグを直接渡すことを有効にします。

たとえば、以下のコマンドは OpenShift の **サービスカタログ** をプロビジョニングします。

```
$ MINISHIFT_ENABLE_EXPERIMENTAL=y minishift start --extra-clusterup-flags "--
enable='*,service-catalog'"
```

注記

サービスカタログを有効にするには、**minishift openshift component add** コマンドを使用します。

2.8.3. ローカルプロキシーサーバー

セキュリティー証明書が組織で使用されるものの、インスタンスと簡単に共有できない状況では、CDK インスタンスが外部リソースにアクセスするために使用できるホストでローカルプロキシーサーバーを実行できます。

プロキシーサーバーの有効化は、以下のコマンドを使用して行います。

```
$ minishift config set local-proxy true
```

これにより、CDK インスタンスに自動的に割り当てられるホストでプロキシーサーバーを起動します。

企業またはアップストリームのプロキシーが分かっている場合は、以下の設定オプションで指定できます。

```
$ minishift config set local-proxy-upstream http(s)://[username:password@]host:port
```

このオプションを使用すると、すべてのトラフィックが CDK 固有の証明書で再暗号化されます。このため、このプロキシーは、CDK の開発および使用のみに使用する必要があります。

重要

ローカルホストへの外部トラフィックを許可するには、ホストのファイアウォールでポート **3128 /tcp** を有効しないといけない場合があります。

2.8.4. ローカル DNS サーバー

CDK は、オフラインの使用のために DNS サーバーを提供し、テスト中に DNS レコードを上書きする可能性があります。これにより、インターネットなしで OpenShift ルートにアクセスできます。

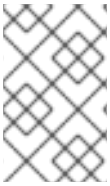
注記

DNS サーバーはプロファイルに固有のものです。

DNS サーバーの起動は、以下のように実行できます。

```
$ minishift dns start
```

DNS サーバーを起動したら、このネームサーバーを使用するようにデバイス設定を設定する必要があります。start コマンドには、オフラインでの使用を入力したときに使用できる一時的なオプションが表示されます。



注記

現在の実装では、サーバーを起動し、ホスト設定に必要な変更を手動で変更する必要があります。DNS 設定は永続的ではなく、デバイスのネットワーク状態が変更するとリセットされる可能性があります。

DNS サーバーの停止は、以下のように実行できます。

```
$ minishift dns stop
```

DNS サーバーの状態を取得するには、次のコマンドを実行します。

```
$ minishift dns status
```

2.8.4.1. macOS のローカル DNS セットアップ

最新バージョンの macOS はオフラインモードで DNS クエリーを送信せず、CDK からローカル DNS サーバーを使用するプロセスは、他のオペレーティングシステムよりも多く関与します。

2.8.4.1.1. tap デバイスの有効化

/dev に **tap** デバイスが存在するかどうかを確認します。

```
$ ls /dev | grep tap
```

tap デバイスがない場合は、**tuntap** パッケージをインストールします。

```
$ brew install tuntap
```

2.8.4.1.2. tap デバイスを使用したネットワークサービスの作成

root で **/Library/Preferences/SystemConfiguration/preferences.plist** ファイルを開き、**<key>NetworkServices</key>** 要素の下に以下の XML を追加します。

```
<key>D16F22CE-6DDE-4E63-837C-E16538EA5CCB</key> 1
<dict>
   <key>DNS</key>
   <dict />
   <key>IPv4</key>
   <dict>
      <key>Addresses</key>
      <array>
         <string>10.10.90.1</string> 2
      </array>
      <key>ConfigMethod</key>
      <string>Manual</string>
      <key>SubnetMasks</key>
      <array>
         <string>255.255.0.0</string>
      </array>
   </dict>
   <key>IPv6</key>
   <dict>
      <key>ConfigMethod</key>
      <string>Automatic</string>
   </dict>
   <key>Interface</key>
   <dict>
      <key>DeviceName</key>
```

```
        <string>tap0</string> 3
        <key>Hardware</key>
        <string>Ethernet</string>
        <key>Type</key>
        <string>Ethernet</string>
        <key>UserDefinedName</key>
        <string>MiniTap</string> 4
    </dict>
    <key>Proxies</key>
    <dict>
        <key>ExceptionsList</key>
        <array>
            <string>*.local</string>
            <string>169.254/16</string>
        </array>
        <key>FTPPassive</key>
        <integer>1</integer>
    </dict>
    <key>SMB</key>
    <dict />
    <key>UserDefinedName</key>
    <string>MiniTap</string> 5
</dict>
```

1 これは、ネットワークサービスの UUID です。この値は、**uuidgen** の出力に置き換えます。

2 ネットワークサービスの IP アドレスです。

3 使用する **/dev/tap** デバイスです。

4 5 ネットワークサービスの名前です (Network Preferences GUI に表示されます)。

2.8.4.1.3. ネットワークサービスを ServiceOrder アレイに追加

/Library/Preferences/SystemConfiguration/preferences.plist ファイルで **<key>ServiceOrder</key>** 要素を見つけます。root で、**MiniTap** ネットワークサービスの UUID をこのアレイに追加します。

```
<key>ServiceOrder</key>
    <array>
        <string>06BFF3C7-13DA-420F-AE9C-B036401184D7</string>
 <string>58231F56-CA25-4D41-930F-46D83CA07BFE</string>
 <string>304203B0-AC87-459F-9761-C2799EEBB2E3</string>
 <string>8655D244-C6E7-4CC0-BF06-BB18F9C3BB85</string>
 <string>3C26FB9D-D918-4B79-9C7B-ADECD8EFE00F</string>
 <string>D16F22CE-6DDE-4E63-837C-E16538EA5CCB</string> 1
    </array>
```

1 **MiniTap** ネットワークサービスの UUID です。

2.8.4.1.4. ネットワークサービスを Service ディクショナリーに追加

/Library/Preferences/SystemConfiguration/preferences.plist ファイルで **<key>Service</key>** 要素を見つけます。root で以下の XML をそのディクショナリーに追加します。

```
<key>Service</key>
  <dict>
    <key>06BFF3C7-13DA-420F-AE9C-B036401184D7</key>
    <dict>
      <key>__LINK__</key>
      <string>/NetworkServices/06BFF3C7-13DA-420F-AE9C-B036401184D7</string>
    </dict>
    <key>304203B0-AC87-459F-9761-C2799EEBB2E3</key>
    <dict>
      <key>__LINK__</key>
      <string>/NetworkServices/304203B0-AC87-459F-9761-C2799EEBB2E3</string>
    </dict>
    <key>3C26FB9D-D918-4B79-9C7B-ADECD8EFE00F</key>
    <dict>
      <key>__LINK__</key>
      <string>/NetworkServices/3C26FB9D-D918-4B79-9C7B-ADECD8EFE00F</string>
    </dict>
    <key>58231F56-CA25-4D41-930F-46D83CA07BFE</key>
    <dict>
      <key>__LINK__</key>
      <string>/NetworkServices/58231F56-CA25-4D41-930F-46D83CA07BFE</string>
    </dict>
    <key>8655D244-C6E7-4CC0-BF06-BB18F9C3BB85</key>
    <dict>
      <key>__LINK__</key>
      <string>/NetworkServices/8655D244-C6E7-4CC0-BF06-BB18F9C3BB85</string>
    </dict>
    <key>D16F22CE-6DDE-4E63-837C-E16538EA5CCB</key> (1)
    <dict>
      <key>__LINK__</key>
      <string>/NetworkServices/D16F22CE-6DDE-4E63-837C-E16538EA5CCB</string> (2)
    </dict>
  </dict>
```

(1) **MiniTap** サービスの UUID です。

(2) この UUID は、**MiniTap** サービスの UUID に置き換えます。

macOS を再起動すると、ネットワーク設定 GUI に **MiniTap** サービスが表示されるはずです。このサービスは切断されます。有効にするには、以下のコマンドを実行します。

```
$ exec 4<>/dev/tap0 (1)
$ ifconfig tap0 10.10.90.1 255.255.0.0 (2) (3)
$ ifconfig tap0 up (4)
```

(1) (2) (4) このデバイスを、**MiniTap** サービスで使用する **/dev/tap** デバイスに置き換えます。

(3) IP アドレスは、**MiniTap** サービス定義の IP アドレスと同じでなければなりません。

2.8.4.1.5. リゾルバー設定の追加

以下の内容で **/etc/resolver/nip.io** ファイルを作成します。

```
nameserver <ip_address_of_the_minishfit_vm>
search_order 1
```

2.8.5. CDK システムトレイ

macOS および Windows で CDK のユーザーが GUI からプロファイルを開始および停止するなどの単純なタスクを実行するのに役立つため、実験的なシステムトレイを含むように、これらのプラットフォームのバイナリーがコンパイルされています。

デフォルトでは、システムトレイは **minishift start** の実行時に自動的に起動します。この動作を無効にするには、次のコマンドを使用します。

```
$ minishift config set auto-start-tray false
```

システムトレイを起動するには、次のコマンドを実行します。

```
$ minishift service start systemtray
```

2.8.6. タイムゾーンの設定

デフォルトとは異なるタイムゾーンを設定する場合は、以下のコマンドを使用します。

```
$ minishift timezone --set <Valid_Timezone>
```

利用可能なタイムゾーンを表示するには、以下のコマンドを使用します。

```
$ minishift timezone --list
```

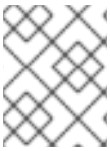
CDK インスタンスの現在のタイムゾーンを確認するには、次のコマンドを実行します。

```
$ minishift timezone
```

2.9. 既存のマシンの実行

2.9.1. 概要

CDK は、**vm-driver** を **汎用的** に使用して、既存のリモートマシンに対して実行できます。



注記

CentOS 7、Red Hat Enterprise Linux 7、および Fedora 27 以降の Fedora は、この機能に推奨される Linux ディストリビューションです。

2.9.2. 既存のリモートマシンの設定

CDK を持つ既存のマシンを使用するには、以下のように設定する必要があります。

1. ホストから既存のリモートマシンへのパスワードなしの SSH を確立します。
   -

```
Host$ ssh-copy-id <user>@<remote_IP_address> # Ensure that the user has sudo access without a password or use root
Host$ ssh <user>@<remote_IP_address>         # Ensure that this login works without a password
```

注記

CentOS 7、Red Hat Enterprise Linux 7、または Fedora (27 以降のバージョン) を使用している場合は、以下の手順を省略します。

1. 既存のリモートマシンを設定します。

```
Remote_Machine$ yum install -y docker net-tools firewalld
Remote_Machine$ systemctl start docker
Remote_Machine$ systemctl enable docker
Remote_Machine$ systemctl start firewalld
Remote_Machine$ systemctl enable firewalld
```

2. TCP ポートの **2376**、**8443**、および **80** が、リモートマシンのファイアウォールを通過するように許可して、ホストから通信できるようにします。

```
Remote_Machine$ firewall-cmd --permanent --add-port 2376/tcp --add-port 8443/tcp --add-port 80/tcp
```

3. Docker ブリッジネットワークのコンテナーサブネットを確認します。

```
Remote_Machine$ docker network inspect -f "{{range .IPAM.Config }}{{ .Subnet }}{{end}}" bridge
```

このコマンドは、サブネット (例: **172.17.0.0/16**) を表示します。

4. Docker ブリッジネットワークのコンテナーサブネットを使用して、サブネットをソースとして使用してファイアウォールの **minishift** ゾーンを作成します。

```
Remote_Machine$ firewall-cmd --permanent --new-zone minishift
Remote_Machine$ firewall-cmd --permanent --zone minishift --add-source <subnet>
```

5. リモートマシンのファイアウォールを通過する UDP ポート **53** および **8053** を許可し、コンテナーが OpenShift マスター API および DNS エンドポイントにアクセスできるようにします。

```
Remote_Machine$ firewall-cmd --permanent --zone minishift --add-port 53/udp --add-port 8053/udp
```

6. リモートマシンでファイアウォールを再読み込みします。

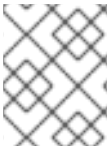
```
Remote_Machine$ firewall-cmd --reload
```

2.9.3. 既存のリモートマシンに対する実行

ホストで次のコマンドを実行し、リモートマシンに対して CDK を実行します。

```
$ minishift start --vm-driver generic --remote-ipaddress <remote_IP_address> --remote-ssh-user
```

```
<username> --remote-ssh-key <private_ssh_key>
$ minishift addon apply htpasswd-identity-provider --addon-env USER_PASSWORD=
<NEW_PASSWORD>
```



注記

--remote-ssh-key フラグの値は、ホストマシンのプライベート SSH キーの場所でなければなりません。

注記

デフォルトのユーザー名とパスワードを変更するには、**minishift start** コマンドの実行後に、必要なユーザー名とパスワードで **htpasswd-identity-provider** アドオンを適用する必要があります。デフォルトでは、CDK は **Allow All** アイデンティティープロバイダーを使用します。**Allow All** アイデンティティープロバイダーは、空でないユーザー名とパスワードがログインできるようにします。

2.10. 既存の DOCKER COMPOSE プロジェクトの変換

2.10.1. Kompose のインストール

Kompose は、Docker Compose を OpenShift や Kubernetes などのコンテナーオーケストレーターに変換するツールです。Kompose は、手動またはパッケージマネージャー (Microsoft Windows の Chocolatey、macOS の Homebrew、または Red Hat Enterprise Linux の Yum) からインストールできます。

2.10.1.1. 手動

1. Kompose GitHub Releases ページから、お使いのオペレーティングシステムのバイナリーをダウンロードします。
2. バイナリーを **PATH** 内の場所に追加します。

2.10.1.2. Chocolatey の使用

Microsoft Windows では、Chocolatey から Kompose をインストールできます。

```
PS> choco.exe install kubernetes-kompose
```

2.10.1.3. Homebrew の使用

macOS では、Homebrew で Kompose をインストールできます。

```
$ brew install kompose
```

2.10.1.4. Red Hat Enterprise Linux の場合

Red Hat Developer Tools リポジトリーおよび Red Hat Software Collections リポジトリーを有効にすることで、コマンドラインから Kompose をインストールできます。

```
$ subscription-manager repos --enable rhel-7-server-devtools-rpms
$ subscription-manager repos --enable rhel-server-rhscl-7-rpms
$ yum install kompose -y
```

2.10.2. Kompose の使用

Kompose を使用して Docker Compose プロジェクトを変換するには、以下の手順に従います。

1. CDK を起動し、OpenShift クラスターと通信できるようにします。

```
$ minishift start
Starting local OpenShift cluster using 'kvm' hypervisor...
-- Checking OpenShift client ... OK
-- Checking Docker client ... OK
-- Checking Docker version ... OK
-- Checking for existing OpenShift container ... OK
...
```

2. サンプル Docker Compose ファイル をダウンロードするか、または独自のファイルを使用します。

```
wget https://raw.githubusercontent.com/kubernetes/kompose/master/examples/docker-compose.yaml
```

3. Docker Compose ファイルを OpenShift に変換します。**docker-compose.yaml** ファイルと同じディレクトリーで **kompose convert --provider=openshift** を実行します。

```
$ kompose convert --provider=openshift
INFO OpenShift file "frontend-service.yaml" created
INFO OpenShift file "redis-master-service.yaml" created
INFO OpenShift file "redis-slave-service.yaml" created
INFO OpenShift file "frontend-deploymentconfig.yaml" created
INFO OpenShift file "frontend-imagestream.yaml" created
INFO OpenShift file "redis-master-deploymentconfig.yaml" created
INFO OpenShift file "redis-master-imagestream.yaml" created
INFO OpenShift file "redis-slave-deploymentconfig.yaml" created
INFO OpenShift file "redis-slave-imagestream.yaml" created
```

ここでは、**kompose up --provider=openshift** を使用して OpenShift に直接変換およびデプロイできます。

```
$ kompose up --provider=openshift
INFO We are going to create OpenShift DeploymentConfigs, Services and
PersistentVolumeClaims for your Dockerized application.
If you need different kind of resources, use the 'kompose convert' and 'oc create -f'
commands instead.

INFO Deploying application in "myproject" namespace
INFO Successfully created Service: frontend
INFO Successfully created Service: redis-master
INFO Successfully created Service: redis-slave
INFO Successfully created DeploymentConfig: frontend
INFO Successfully created ImageStream: frontend
INFO Successfully created DeploymentConfig: redis-master
```

```
INFO Successfully created ImageStream: redis-master
INFO Successfully created DeploymentConfig: redis-slave
INFO Successfully created ImageStream: redis-slave

Your application has been deployed to OpenShift. You can run 'oc get dc,svc,is,pvc' for
details.
```

4. CDK を使用して新たにデプロイされたアプリケーションにアクセスします。
デプロイメント後に、サービスにアクセスするために OpenShift ルートを作成する必要があります。

   oc を使用して **frontend** サービスのルートを作成します。

```
$ oc expose service/frontend
route "frontend" exposed
```

   minishift を使用して **frontend** サービスにアクセスします。

```
$ minishift openshift service frontend --namespace=myproject --in-browser
Opening the service myproject/frontend in the default browser...
```

   デプロイされたコンテナーの概要については、OpenShift の GUI にアクセスすることもできます。

```
$ minishift console
Opening the OpenShift Web console in the default browser...
```

第3章 CONTAINER DEVELOPMENT KIT を使用した OPENSHIFT との対話

CDK は仮想マシンを作成し、この仮想マシンにローカルで単一ノードの OpenShift クラスターをプロビジョニングします。以下のセクションでは、CDK がローカルの OpenShift クラスターの対話および設定を支援する方法を説明します。

CDK 仮想マシンを管理する方法は、基本的な使用方法 セクションを参照してください。

3.1. OPENSHIFT クライアントバイナリー (OC) の使用

3.1.1. 概要

minishift start コマンドは、cluster up 方法を使用して OpenShift クラスターを作成します。このため、**oc** バイナリーをホストにコピーします。

oc バイナリーは、**$MINISHIFT_HOME/cache/oc/v3.11.346** ディレクトリーにあります。これは、CDK のデフォルトの OpenShift バージョンを使用することを前提とします。**minishift oc-env** を使用してこのバイナリーを **PATH** に追加できます。これは、シェルに入力する必要のあるコマンドを表示します。

minishift oc-env の出力は、オペレーティングシステムとシェルタイプによって異なります。

```
$ minishift oc-env
export PATH="/home/user/.minishift/cache/oc/v3.11.346:$PATH"
# Run this command to configure your shell:
# eval $(minishift oc-env)
```

3.1.2. CDK CLI プロファイル

minishift start コマンドの一環として、**minishift** という名前の CLI プロファイル も作成されます。このプロファイルは **コンテキスト** とも呼ばれ、OpenShift クラスターと通信するための設定が含まれます。

CDK は自動的にこのコンテキストをアクティベートしますが、たとえば別の OpenShift インスタンスにログインした後にそのコンテキストに戻す必要がある場合は、以下を実行します。

```
$ oc config use-context minishift
```

oc の使用方法の概要は、OpenShift ドキュメントの CLI の使用方法 トピックを参照してください。

3.1.3. クラスターへのログイン

デフォルトでは、**cluster up** は、AllowAllPasswordIdentityProvider を使用してローカルクラスターに対して認証を行います。つまり、空でないユーザー名およびパスワードを使用してローカルクラスターにログインできます。

推奨されるユーザー名およびパスワードは、それぞれ **developer** と developer です。これは、そのユーザーおよびパスワードがすでにデフォルトのプロジェクト **myproject** に割り当てられており、管理者の権限を借用 できるためです。これにより、**--as system:admin** パラメーターを使用して管理者コマンドを実行できます。

管理者としてログインするには、システムアカウントを使用します。

```
$ oc login -u system:admin
```

この場合は、クライアント証明書 が使用されます。証明書は **~/.kube/config** に保存されます。**cluster up** コマンドは、ブートストラップの一部として適切な証明書をインストールします。

注記

oc login -u system -p admin コマンドを実行すると、ログインはしますが、管理者ではありません。代わりに、特定の権限を持たない非特権ユーザーとしてログインします。

利用可能なログインコンテキストを表示するには、以下を実行します。

```
$ oc config view
```

3.1.4. Web コンソールへのアクセス

OpenShift Web コンソール にアクセスするには、CDK を起動して Web コンソールの URL を取得するために、シェルでこのコマンドを実行します。

```
$ minishift console --url
```

または、CDK の起動後、以下のコマンドを使用してブラウザーでコンソールを直接開くことができます。

```
$ minishift console
```

3.1.5. OpenShift サービスへのアクセス

ルートで公開されるサービスにアクセスするには、シェルで以下のコマンドを実行します。

```
$ minishift openshift service [-n NAMESPACE] [--url] NAME
```

詳細は、サービスの公開 も参照してください。

3.1.6. OpenShift ログの表示

OpenShift ログにアクセスするには、CDK の起動後に以下のコマンドを実行します。

```
$ minishift logs
```

3.1.7. OpenShift 設定の更新

OpenShift の実行中に、クラスターのマスターまたはノード設定を表示および変更できます。

OpenShift マスター設定ファイル **master-config.yaml** を表示するには、以下のコマンドを実行します。

```
$ minishift openshift config view
```

マスター設定の代わりにノードまたは kubeapi-server 設定を表示するには、**target** フラグを指定します。

view サブコマンドの詳細は、**minishift openshift config view** の概要を参照してください。

注記

OpenShift 設定の更新後、OpenShift は透過的に再起動します。

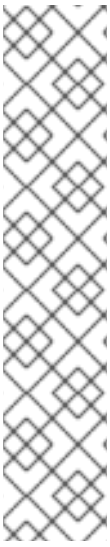
3.1.7.1. 例: クロスオリジンリソース共有の設定

この例では、追加の IP アドレスがリソースを要求するように OpenShift マスター設定を更新して、クロスオリジンリソース共有 (CORS) を設定します。

デフォルトでは、OpenShift はクラスターの IP アドレスまたはローカルホストからのクロスオリジンリソース要求のみを許可します。この設定は、マスター設定 ファイル **master-config.yaml** の **corsAllowedOrigins** プロパティーに保存されます。

プロパティー値を変更し、すべてのドメインからのクロスオリジン要求を許可するには、以下のコマンドを実行します。

```
$  minishift openshift config set --patch '{"corsAllowedOrigins": [".*"]}'
```



注記

上記コマンドを実行すると **The specified patch need to be a valid JSON.**エラーが発生する場合は、オペレーティングシステム、シェル環境とその挿入動作に応じて、上記のコマンドを変更する必要があります。

たとえば、Windows 7 または 10 で PowerShell を使用している場合は、上記のコマンドを以下のように変更します。

```
PS> minishift.exe openshift config set --patch '{\"corsAllowedOrigins\": [\".*\"]}'
```

コマンドプロンプトを使用する場合は、以下を使用します。

```
C:\> minishift.exe openshift config set --patch "{\"corsAllowedOrigins\": [\".*\"]}"
```

3.1.7.2. 例: OpenShift ルーティング接尾辞の変更

この例では、マスター設定で OpenShift ルーティングの接尾辞を変更します。

静的ルーティング接尾辞を使用する場合は、**minishift start** コマンドの一部として **routing-suffix** フラグを設定できます。デフォルトでは、CDK は nip.io をベースとした動的なルーティング接頭辞を使用します。この接頭辞は、仮想マシンの IP アドレスはルーティング接尾辞 (例: **192.168.99.103.nip.io**) に含まれます。

nip.io で問題が発生した場合は、同じ原則を基にした xip.io を使用できます。

ルーティング接尾辞を **xip.io** に設定するには、以下のコマンドを実行します。

```
$ minishift openshift config set --patch '{"routingConfig": {"subdomain": "<IP-ADDRESS>.xip.io"}}'
```

上記の例の **IP-ADDRESS** は、CDK 仮想マシンの IP アドレスに置き換えてください。 **minishift ip** コマンドを実行して、IP アドレスを取得できます。

3.1.8. コンポーネントの OpenShift クラスターへの追加

コンポーネントを実行中の OpenShift クラスターに追加するには、以下を使用します。

```
$ minishift openshift component add <component-name>
```

3.1.8.1. 例: service-catalog コンポーネントの追加

この例では、**service-catalog** コンポーネントを以下のように追加できます。

```
$ minishift openshift component add service-catalog
```

3.1.9. OpenShift クラスターに追加する有効なコンポーネントの一覧表示

実行中の OpenShift クラスターに追加できる有効なコンポーネントを一覧表示するには、以下を実行します。

```
$ minishift openshift component list
```

3.2. サービスの公開

3.2.1. 概要

OpenShift にデプロイした後にサービスを公開する方法は複数あります。以下のセクションでは、さまざまな方法とそれらを使用するタイミングを説明します。

3.2.2. ルート

Web アプリケーションをデプロイする場合、そのアプリケーションを公開する最も一般的な方法は ルート による方法です。ルートはサービスをホスト名として公開します。Web コンソールまたは CLI を使用してルートを作成できます。

```
$ oc expose svc/frontend --hostname=www.example.com
```

アプリケーションを作成し、これをルートで公開する完全な例は、サンプルアプリケーションのデプロイ セクションを参照してください。

3.2.3. NodePort サービス

公開するサービスが HTTP ベースではない場合には、NodePort サービスを作成できます。この場合、各 OpenShift ノードは、そのポートをサービスにプロキシーします。CDK 仮想マシンでこのポートにアクセスするには、**type=LoadBalancer** パラメーターで **oc expose** を使用して Ingress IP を設定する必要があります。

Ingress IP Self-Service の一般的なユースケースは、データベースサービスを公開する機能です。CDK を使用して MariaDB インスタンスを作成および公開する完全なワークフローの例を以下に示します。

```
$ minishift start
```

```
$ eval $(minishift oc-env)
$ oc new-app -e MYSQL_ROOT_PASSWORD=admin
https://raw.githubusercontent.com/openshift/origin/master/examples/db-templates/mariadb-persistent-
template.json
$ oc rollout status -w dc/mariadb
$ oc expose dc mariadb --type=LoadBalancer --name=mariadb-ingress
$ oc export svc mariadb-ingress
 ....
ports:
   - nodePort: 30907
 ....
```

サービスが公開されると、CDK 仮想マシン IP および公開された NodePort サービスを使用して、CLI の **mysql** で MariaDB にアクセスできます。

```
$ mysql --user=root --password=admin --host=$(minishift ip) --port=30907
```

3.2.4. ポート転送

3.2.4.1. oc port-forward の使用

クラスターの特定の Pod のポートにすばやくアクセスする必要がある場合は、OpenShift CLI の **oc port-forward** コマンドを使用することもできます。

```
$ oc port-forward POD [LOCAL_PORT:]REMOTE_PORT
```

3.2.4.2. VirtualBox ツールの使用

VirtualBox ドライバープラグインを使用している場合は、ポート転送に使用する別の方法があります。この方法では、永続的なポート転送だけでなく、複数のポートを同時に転送することができます。この方法では、上記のように **nodePort** を設定する必要があります。

上記の例で説明する場合は、以下を行うことができます。

```
$ VBoxManage controlvm minishift natpf1 "mariadb,tcp,,3306,,30907"
```

これにより、**localhost:3306** の mariadb サービスと通信できます。これは、デフォルトのポートを変更したくない場合に便利な場合があります。また、ポート転送というこの方法の重要な利点は、単一の Pod だけでなくサービスと通信できることです。

3.3. OPENSHIFT DOCKER レジストリーへのアクセス

3.3.1. 概要

OpenShift は、開発に使用できる統合 Docker レジストリーを提供します。レジストリーに存在するイメージはアプリケーションに直接使用でき、ローカル開発ワークフローを高速化できます。

3.3.2. レジストリーへのログイン

1. CDK を起動し、**oc** バイナリーを **PATH** に追加します。詳細な例は、CDK クイックスタートセクションを参照してください。

2. シェルが CDK Docker デーモンを再利用する ように設定されていることを確認します。
3. OpenShift Docker レジストリーにログインします。

```
$ docker login -u developer -p $(oc whoami -t) $(minishift openshift registry)
```

3.3.3. アプリケーションのデプロイ

以下の例は、ローカルにビルドされた Docker イメージから OpenShift アプリケーションを直接デプロイする方法を示しています。この例では、OpenShift プロジェクト **myproject** を使用します。このプロジェクトは、**minishift start** により自動的に作成されます。

1. シェルが CDK Docker デーモンを再利用する ように設定されていることを確認します。
2. 通常どおりに Docker イメージをビルドします。
3. OpenShift レジストリーに対してイメージをタグ付けします。

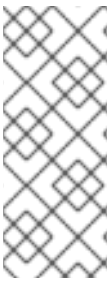
```
$ docker tag my-app $(minishift openshift registry)/myproject/my-app
```

4. イメージをレジストリーにプッシュし、アプリケーションと同じ名前のイメージストリームを作成します。

```
$ docker push $(minishift openshift registry)/myproject/my-app
```

5. イメージストリームからアプリケーションを作成し、サービスを公開します。

```
$ oc new-app --image-stream=my-app --name=my-app
$ oc expose service my-app
```



注記

oc run --image [...] を使用してアプリケーションをデプロイする場合は、公開される内部レジストリールートは機能しません。以下のように、デプロイするプロジェクトとアプリケーションと共に内部レジストリー IP を使用する必要があります。

```
$ oc run myapp --image 172.30.1.1:5000/myproject/myapp
```

第4章 CDK のトラブルシューティング

本セクションでは、CDK の設定および使用中に発生する可能性のある一般的な問題の解決策を説明します。

4.1. トラブルシューティングスターとガイド

4.1.1. 概要

本セクションでは、CDK のインストールおよび設定中に発生する可能性のある一般的な問題の解決策を説明します。

4.1.2. 失敗した CDK の起動チェック

CDK が起動する間、起動チェックが複数実行され、CDK 仮想マシンと OpenShift クラスターが問題なく起動できることを確認します。設定が正しくない、または存在しない場合は、起動チェックに失敗し、CDK が起動しません。

- 以下のコマンドを実行して起動チェックを省略できます。

```
$ minishift config set skip-startup-checks true
```

以下のセクションでは、さまざまな起動チェックを説明します。

4.1.2.1. ドライバープラグインの設定

起動チェックの 1 つは、関連するドライバープラグインが正しく設定されていることを確認します。この起動チェックに失敗する場合は、仮想化環境の設定 のトピックを確認し、適切なドライバーを設定します。

ドライバープラグインチェックに失敗していて CDK を強制的に起動する場合は、CDK にこれらのエラーを警告として扱うように指示することができます。

- Linux 上の KVM/libvirt の場合は、以下のコマンドを実行します。

```
$ minishift config set warn-check-kvm-driver true
```

- macOS の hyperkit の場合は、以下のコマンドを実行します。

```
$ minishift config set warn-check-hyperkit-driver true
```

- Windows 上の Hyper-V の場合は、以下のコマンドを実行します。

```
C:\> minishift.exe config set warn-check-hyperv-driver true
```

4.1.2.2. 永続ストレージボリュームの設定および使用方法

CDK は、永続ストレージボリュームがマウントされ、十分なディスク領域が利用可能かどうかを確認します。たとえば、永続ストレージボリュームが利用可能なディスク領域の 95% を超えると、CDK は起動しません。

データを復元する場合は、このテストを省略し、CDK を起動して永続ボリュームにアクセスすることができます。

```
$ minishift config set skip-check-storage-usage true
```

4.1.2.3. 外部ネットワーク接続

CDK 仮想マシンが起動したら、いくつかのネットワークチェックを実行して、CDK 仮想マシン内から外部接続が可能かどうかを確認します。

デフォルトでは、開発環境の相違点により、ネットワークチェックはエラーを警告として処理するように設定されます。お使いの環境に合わせてネットワークチェックを最適化するように設定できます。

たとえば、ネットワークのいずれかが外部ホストの ping をチェックします。ホストを変更するには、以下のコマンドを実行します。

```
$ minishift config set check-network-ping-host <host-IP-address>
```

<host-IP-address> を、内部 DNS サーバー、プロキシーホスト、またはマシンからアクセスできる外部ホストのアドレスに置き換えます。

プロキシー接続は問題となる可能性があるため、外部 URL の取得を試みるチェックを実行できます。以下を実行して URL を設定できます。

```
$ minishift config set check-network-http-host <URL>
```

4.1.3. OpenShift Web コンソールが古いバージョンの Safari と動作しない

minishift console は、バージョン 10.1.2 (12603.3.8) などの旧バージョンの Safari Web ブラウザーでは機能しません。Web コンソールにアクセスしようとすると、以下のエラーが発生します。

```
Error unable to load details about the server
```

これについては、最新バージョンに更新するか、Firefox または Chrome ブラウザーを使用します。バージョン 11.0.3 (13604.5.6) はテストされ、OpenShift Web コンソールと動作します。**minishift console --url** を使用して Web コンソール URL を取得できます。

4.2. ドライバープラグインのトラブルシューティング

4.2.1. 概要

本セクションでは、CDK のドライバープラグインを設定する際に発生する可能性のある一般的な問題の解決策を説明します。

4.2.2. KVM/libvirt

4.2.2.1. Undefining virsh snapshots fail

KVM/libvirt で **virsh** を使用して開発ワークフローにスナップショットを作成する場合は、**minishift delete** を使用してスナップショットを削除すると、以下のエラーが発生する可能性があります。

```
$ minishift delete
Deleting the Minishift VM...
Error deleting the VM:  [Code-55] [Domain-10] Requested operation is not valid: cannot delete
inactive domain with 4 snapshots
```

原因: スナップショットは **~/.minishift/machines** に保存されますが、定義は **/var/lib/libvirt/qemu/snapshot/minishift** に保存されます。

回避策: スナップショットを削除するには、root で以下の手順を実行します。

1. 定義を削除します。

```
# virsh snapshot-delete --metadata minishift <snapshot-name>
```

2. CDK ドメインの定義を解除します。

```
# virsh undefine minishift
```

minishift delete を実行して仮想マシンを削除して CDK を再起動できるようになりました。

注記

これらの手順で問題が解決しない場合は、以下のコマンドを使用してスナップショットを削除することもできます。

```
$ rm -rf ~/.minishift/machines
```

スナップショットを作成する際にメタデータを使用しないことが推奨されます。これを確認するには、**--no-metadata** フラグを指定します。以下に例を示します。

```
# virsh snapshot-create-as --domain vm1 overlay1 --diskspec vda,file=/export/overlay1.qcow2 --disk-
only --atomic --no-metadata
```

4.2.2.2. Error creating new host: dial tcp: missing address

この問題は、**libvirtd** サービスが実行していない可能性があります。これは、以下のコマンドで確認できます。

```
$ systemctl status libvirtd
```

libvirtd が実行していない場合は、システムの起動時に開始して有効にできるようにします。

```
$ systemctl start libvirtd
$ systemctl enable libvirtd
```

4.2.2.3. Failed to connect socket to '/var/run/libvirt/virtlogd-sock'

この問題は、**virtlogd** サービスが実行されていない可能性があります。これは、以下のコマンドで確認できます。

```
$ systemctl status virtlogd
```

virtlogd が実行していない場合は、システムの起動時に開始して有効にできるようにします。

```
$ systemctl start virtlogd
$ systemctl enable virtlogd
```

4.2.2.4. Domain 'minishift' already exists...

minishift start を試行し、このエラーが表示される場合は、**minishift delete** を使用して、先に作成した仮想マシンを削除するようにしてください。ただし、これに失敗し、CDK を完全にクリーンアップして新規に起動する場合は、以下の手順を実施します。

1. root で、既存の CDK 仮想マシンが実行中かどうかを確認します。

```
# virsh list --all
```

2. CDK 仮想マシンが実行している場合は、停止します。

```
# virsh destroy minishift
```

3. 仮想マシンを削除します。

```
# virsh undefine minishift
```

4. 通常のユーザーとして、**~/.minishift/machines** ディレクトリーを削除します。

```
$ rm -rf ~/.minishift/machines
```

これをすべて失敗した場合は、CDK をアンインストール し、CDK の新規インストールを行う場合があります。

4.2.3. VirtualBox

4.2.3.1. Error machine does not exist

Windows を使用する場合は、**minishift start** コマンドで **--vm-driver virtualbox** フラグを設定してください。また、この問題は、VirtualBox の古いバージョンである可能性があります。

この問題を回避するには、VirtualBox 5.1.12 以降を使用することが推奨されます。

4.2.4. Hyper-V

4.2.4.1. Hyper-V commands must be run as an Administrator

通常のユーザーまたは管理者権限を持つユーザーとして Hyper-V を使用して Windows で CDK を実行すると、以下のエラーが発生する可能性があります。

```
Error starting the VM: Error creating the VM. Error with pre-create check: "Hyper-V commands must be run as an Administrator".
```

回避策: 推奨されている Hyper-V Administrators グループに追加するか、昇格モードでシェルを実行することができます。

PowerShell を使用している場合は、以下のように Hyper-V Administrators グループに追加できます。

1. 管理者として、以下のコマンドを実行します。

```
PS> ([adsi]"WinNT://./Hyper-V
Administrators,group").Add("WinNT://$env:UserDomain/$env:Username,user")
```

2. 変更を有効にするには、ログアウトしてログインします。

また、GUI を使用して、以下のように Hyper-V Administrators グループに追加することもできます。

1. **スタート** ボタンをクリックして、**コンピューターの管理** を選択します。
2. **Computer Management** 画面で、**Local Users And Groups** を選択してから、**Groups** をダブルクリックします。
3. **Hyper-V Administrators** グループをダブルクリックして、**Hyper-V Administrators Properties** ダイアログボックスが表示されます。
4. Hyper-V Administrators グループにアカウントを追加し、ログオフしてからログインして変更を反映します。

これで、Hyper-V コマンドを通常のユーザーとして実行できるようになりました。

Hyper-V のオプションの詳細は、creating Hyper-V administrators local group を参照してください。

4.2.4.2. CDK running with Hyper-V fails when connected to OpenVPN

OpenVPN などの VPN に接続している間に、外部の仮想スイッチを使用して Hyper-V で CDK を使用しようとすると、CDK が仮想マシンのプロビジョニングに失敗する可能性があります。

原因: Hyper-V ネットワークは、VPN に接続する際に、ネットワークトラフィックを双方向で適切にルーティングしない可能性があります。

回避策: VPN から接続を解除し、Hyper-V マネージャーから仮想マシンを停止した後に再試行します。

4.3. その他のトラブルシューティング

4.3.1. 概要

本セクションでは、CDK のさまざまなコンポーネントの使用中に発生する可能性のある一般的な問題の解決策を説明します。

4.3.2. The root filesystem of the CDK VM exceeds overlay size

追加パッケージをインストールしたり、CDK 仮想マシンのルートファイルシステムにコピーしたり、割り当てたオーバーレイサイズを超えたり、CDK 仮想マシンをロックする可能性があります。

原因: CDK 仮想マシンルートファイルシステムには、CDK 仮想マシンおよびコンテナーの実行を最適化するように設定されたコアパッケージが含まれています。root ファイルシステムで利用可能なストレージは、オーバーレイサイズにより決定されます。これは、利用可能なストレージの合計よりも小さくなります。

回避策: CDK 仮想マシンの root ファイルシステムにパッケージをインストールしたり、大きなファイルを保存したりします。その代わりに、**/mnt/sda1/** 永続ストレージボリュームにサブディレクトリーを

作成したり、ホストと CDK 仮想マシンとの間でストレージ領域を共有できる ホストフォルダー を定義してマウントしたりできます。

CDK 仮想マシン内で開発タスクを実行する場合は、永続ストレージボリュームに保存され、CDK Docker デーモンを再利用する コンテナーを使用することが推奨されます。

4.3.3. Special characters cause passwords to fail

オペレーティングシステムおよびシェル環境によっては、特定の特殊文字が変数挿入をトリガーするため、パスワードが失敗します。

回避策: パスワードを作成して入力する場合は、**'<password>'** の形式で文字列を一重引用符で囲みます。

4.3.4. Cannot access web console with Microsoft Edge

Microsoft Edge を使用して OpenShift Web コンソールにアクセスしようとすると、以下のエラーが発生します。

> Site not reachable.

OpenShift Web コンソールは、空のページとしてレンダリングされる可能性があります。

原因: Microsoft Edg が原因で発生したエラーです。

回避策: 代替の Web ブラウザーを使用して OpenShift Web コンソールにアクセスします。Mozilla Firefox と Google Chrome はどちらも期待どおりに機能します。

4.3.5. X.509 certificate is valid for 10.0.2.15, 127.0.0.1, 172.17.0.1, 172.30.0.1, 192.168.99.100, not 192.168.99.101

停止した CDK 仮想マシンを起動すると、以下の X.509 証明書エラーが発生します。

```
$ minishift start
-- Checking if requested hypervisor 'kvm' is supported on this platform ... OK
-- Checking the ISO URL ... OK
-- Starting local OpenShift cluster using 'kvm' hypervisor ...
-- Starting Minishift VM ..... OK
[...]
FAIL
   Error: cannot access master readiness URL https://192.168.99.101:8443/healthz/ready
   Details:
     No log available from "origin" container

   Caused By:
     Error: Get https://192.168.99.101:8443/healthz/ready: x509: certificate is valid for 10.0.2.15,
127.0.0.1, 172.17.0.1, 172.30.0.1, 192.168.99.100, not 192.168.99.101
Error during 'cluster up' execution: Error starting the cluster.
```

上記のエラーの原因は、OpenShift クラスター証明書に CDK 仮想マシンの IP が含まれることです。証明書は、CDK 仮想マシンが新たに起動する場合にのみ生成されます。再起動後、CDK 仮想マシンに新しい IP アドレスが割り当てられる可能性があります。この場合は、証明書が無効になります。

回避策: 既存の CDK 仮想マシンを削除して、再起動します。

```
$ minishift delete --force
$ minishift start
```

4.3.6. Removing the subscription password from an OS-native keychain

4.3.6.1. Windows

Windows では、パスワードは **認証情報マネージャー** を使用して保存されます。コマンドプロンプトで次のコマンドで実行して、保存されたパスワードを削除します。

```
C:\> cmdkey /delete:minishift:<username>
```

4.3.6.2. Red Hat Enterprise Linux

Red Hat Enterprise Linux では、パスワードは、**libsecret** によって提供される **D-Bus Secret Service API** を使用して保存されます。以下のコマンドを実行して、保存されたパスワードを削除します。

```
$ secret-tool clear service minishift username <username>
```

4.3.6.3. macOS

macOS では、このパスワードは **Keychain Access** を使用して保存されます。以下のコマンドを実行して、保存されたパスワードを削除します。

```
$ security delete-generic-password -s minishift
```