Red Hat CodeReady Containers 1.31

Getting Started ガイド

CodeReady コンテナーで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: 2021-08-23

Red Hat CodeReady Containers 1.31 Getting Started ガイド

CodeReady コンテナーでの使用および開発に関するクイックスタートガイド

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

法律上の通知

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Getting_Started_Guide.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

This guide shows how to get up to speed using CodeReady Containers. Included instructions and examples guide through first steps developing containerized applications using Red Hat OpenShift Container Platform 4 from a host workstation (Microsoft Windows, macOS, or Red Hat Enterprise Linux).

目次

多様性を受け入れるオープンソースの強化

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

第1章 RED HAT CODEREADY コンテナーの概要

1.1. CODEREADY コンテナーについて

Red Hat CodeReady Containers は、ローカルコンピューターに最小限の OpenShift 4 クラスターを提供します。このクラスターは、開発およびテストの目的で最小限の環境を提供します。CodeReady コンテナーは、主に開発者のデスクトップ上での実行を目的としています。ヘッドレス、複数開発者の設定など、他のユースケースでは、完全な OpenShift インストーラーを使用します。

OpenShift を初めて使用する場合は、OpenShift ドキュメントを参照してください。

CodeReady コンテナーには、OpenShift クラスターを実行する CodeReady コンテナーの仮想マシンと対話するための **crc** コマンドラインインターフェース(CLI)が含まれます。

1.2. 実稼働環境の OPENSHIFT インストールとの相違点

Red Hat CodeReady Containers は、以下の重要な違いを持つ通常の OpenShift インストールです。

- **CodeReady Containers OpenShift クラスターは一時的なもので、実稼働環境での使用を目的としていません。**
- **サポートされている OpenShift バージョンへのアップグレードパスはサポートされていません。** OpenShift バージョンをアップグレードすると、再現が困難な問題が発生する可能性があります。
- マスターとワーカーノードの両方として動作する単一ノードを使用します。
- **これはデフォルトでモニタリング** Operator を無効にします。この無効にされた Operator により、Web コンソールの対応する部分が機能しなくなります。
- OpenShift インスタンスは仮想マシンで実行されます。これにより、特に外部ネットワークとの他の違いが生じる可能性があります。

CodeReady コンテナーには、以下のカスタマイズ不可能なクラスター設定も含まれます。これらの設定は修正しないでください。

- ***.crc.testing** ドメインを使用します。
- 内部クラスター通信に使用されるアドレス範囲。
    - クラスターは 172 アドレス範囲を使用します。これにより、たとえばプロキシーが同じアドレス空間で実行される場合に問題が発生する可能性があります。

第2章 インストール

2.1. 最小のシステム要件

CodeReady Containers には、以下の最小限のハードウェアおよびオペレーティングシステムの要件があります。

2.1.1. ハードウェア要件

CodeReady コンテナーには、以下のシステムリソースが必要です。

- 4 物理 CPU コア
- 9 GB の空きメモリー
- 35 GB のストレージ領域

注記

OpenShift クラスターでは、CodeReady Containers 仮想マシンでこれらの最小リソースを実行する必要があります。ワークロードによっては、より多くのリソースが必要になる場合があります。CodeReady コンテナーの仮想マシンに追加のリソースを割り当てるには、「仮想マシンの設定」を参照してください。

2.1.2. オペレーティングシステムの要件

CodeReady コンテナーには、以下の最小バージョンのサポート対象のオペレーティングシステムが必要です。

2.1.2.1. Microsoft Windows

- Microsoft Windows では、CodeReady Containers では、Windows 10 Fall Creators Update(version 1709)以降が必要です。CodeReady コンテナーは、以前のバージョンの Microsoft Windows では機能しません。Microsoft Windows 10 Home Edition はサポートされません。

2.1.2.2. macOS

- macOS では、CodeReady Containers には macOS 10.14 Mojave 以降が必要です。CodeReady コンテナーは、macOS の以前のバージョンでは機能しません。

2.1.2.3. Linux

- Linux では、CodeReady Containers は Red Hat Enterprise Linux/CentOS 7.5 以降（8.x バージョンを含む）および最新の 2 つの安定した Fedora リリースでのみサポートされます。
- Red Hat Enterprise Linux を使用する場合、CodeReady コンテナーを実行するマシンは Red Hat カスタマーポータルに登録する必要があります。
- Ubuntu 18.04 LTS 以降および Debian 10 以降は正式にサポートされておらず、ホストマシンを手動でセットアップする必要がある場合があります。

- Linux ディストリビューションに必要なパッケージをインストールするには、「必要なソフトウェアパッケージ」を参照してください。

2.2. LINUX に必要なソフトウェアパッケージ

CodeReady Containers **では、Linux で実行する libvirt** および **NetworkManager** パッケージが必要です。Linux ディストリビューション向けにこれらのパッケージをインストールするのに使用されるコマンドを確認するには、以下の表を参照してください。

表2.1ディストリビューションによるパッケージのインストールコマンド

Linux Distribution	インストールコマンド
Fedora	**sudo dnf install NetworkManager**
Red Hat Enterprise Linux/CentOS	**su -c 'yum install NetworkManager'**
Debian/Ubuntu	**sudo apt install qemu-kvm libvirt-daemon libvirt-daemon-system network-manager**

2.3. CODEREADY コンテナーのインストール

CodeReady コンテナーは、Red Hat Enterprise Linux および Microsoft Windows の移植可能な実行可能ファイルとして利用できます。macOS では、CodeReady Containers はガイド付きインストーラーを使用して利用できます。

前提条件

- ホストマシンが最小システム要件を満たしている必要があります。詳細は、「最小システム要件」を参照してください。

手順

1. プラットフォーム用の最新の CodeReady Containers リリースをダウンロードします。
2. Microsoft Windows で、アーカイブの内容を展開します。
3. macOS または Microsoft Windows で、ガイド付きインストーラーを実行して、手順に従います。

注記

Microsoft Windows では、CodeReady コンテナーをローカルにインストールする必要があります **C:** ドライブ。ネットワークドライブから CodeReady コンテナーを実行することはできません。

Red Hat Enterprise Linux では、 **~/Downloads** ディレクトリーは、以下の手順に従います。

a. アーカイブの内容を展開します。

```
$ cd ~/Downloads
$ tar xvf crc-linux-amd64.tar.xz
```

b. Create the ~/**bin** ディレクトリーが存在していない場合は、crc **実行可能ファイルをこれにコピーします。**

```
$ mkdir -p ~/bin
$ cp ~/Downloads/crc-linux-*-amd64/crc ~/bin
```

c. 次の追加： ~/**bin PATH** へのディレクトリー：

```
$ export PATH=$PATH:$HOME/bin
$ echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

2.4. CODEREADY コンテナーのアップグレード

CodeReady コンテナーの実行可能ファイルの新しいバージョンでは、以前のバージョンと互換性のないことを防ぐために、手動で設定する必要があります。

手順

1. CodeReady Containers の最新リリースをダウンロードします。
2. 既存の CodeReady Containers 仮想マシンを削除します。

```
$ crc delete
```

> **警告**
>
> **crc delete** コマンドを使用すると、CodeReady Containers 仮想マシンに保存されているデータが失われます。このコマンドを実行する前に、仮想マシンに保存されている必要な情報を保存します。

3. 以前の **crc** 実行可能ファイルを、最新リリースの実行ファイルに置き換えます。バージョンを確認して、新しい **crc** 実行ファイルが使用中であることを確認します。

```
$ crc version
```

4. 新規の CodeReady Containers リリースを設定します。

```
$ crc setup
```

5. 新規の CodeReady Containers 仮想マシンを起動します。

```
$ crc start
```

第3章 CODEREADY コンテナーの使用

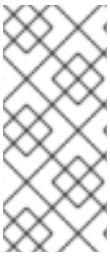
3.1. CODEREADY コンテナーの設定

crc setup コマンドは、CodeReady Containers 仮想マシンのホストマシンの環境を設定する操作を実行します。

この手順では、～/.crc ディレクトリーがない場合は作成します。

前提条件

- Linux または macOS の場合は、ユーザーアカウントに **sudo** コマンドを使用するパーミッションがあることを確認します。Microsoft Windows で、ユーザーアカウントが管理者権限を昇格できることを確認します。



注記

- **crc** 実行可能ファイルを **root**（または Administrator）として実行しないでください。常にユーザーアカウントで **crc** 実行を実行します。
- 新規バージョンを設定する場合は、新しい CodeReady Containers リリースをセットアップする前に、仮想マシンに加えた変更をキャプチャーします。

手順

1. CodeReady コンテナーのホストマシンを設定します。

```
$ crc setup
```

Telemetry データ収集の Consent

crc setup コマンドでは、開発を支援するために、オプションで匿名の使用データ収集の入力を求めるプロンプトが表示されます。個人的識別可能な情報は収集されません。

- Telemetry を手動で有効にするには、以下のコマンドを実行します。

```
$ crc config set consent-telemetry yes
```

- Telemetry を手動で無効にするには、以下のコマンドを実行します。

```
$ crc config set consent-telemetry no
```

収集したデータの詳細は、「Red Hat Telemetry データ収集についての通知」を参照してください。

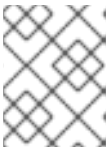
3.2. 仮想マシンの起動

crc start コマンドは、CodeReady Containers 仮想マシンおよび OpenShift クラスターを起動します。

前提条件

- ネットワーク関連の問題を回避するには、VPN に接続していないことと、ネットワーク接続が信頼できることを確認してください。

- **crc setup** コマンドでホストマシンを設定します。詳細は、「CodeReady コンテナーの設定」を参照してください。
- Microsoft Windows で、ユーザーアカウントが管理者権限を昇格できることを確認します。
- 有効な OpenShift ユーザープルシークレットがあること。Install on Laptop: Red Hat CodeReady Containers ページの cloud.redhat.com の「Pull Secret」セクションからプルシークレットをコピーするか、またはダウンロードします。



注記

ユーザープルシークレットにアクセスするには、Red Hat アカウントが必要です。

手順

1. CodeReady Containers 仮想マシンを起動します。

```
$ crc start
```

2. プロンプトが表示されたら、ユーザープルシークレットを指定します。

注記

- クラスターは、要求を提供する前に必要なコンテナーおよび Operator を起動するまでに最低でも 4 分かかります。
- **crc の起動時にエラーが発生する場合は https://access.redhat.com/documentation/en-us/red_hat_codeready_containers/1.31/html-single/getting_started_guide/#troubleshooting-codeready-containers_gsg、「CodeReady Containers のトラブルシューティング」**で解決策を確認してください。

関連情報

- 仮想マシンに割り当てられるデフォルトリソースを変更するには、「仮想マシンの設定」を参照してください。

3.3. OPENSHIFT クラスターへのアクセス

OpenShift Web コンソールまたはクライアント実行可能ファイル(**oc**)を使用して、CodeReady Containers 仮想マシンで実行されている OpenShift クラスターにアクセスします。

3.3.1. OpenShift Web コンソールへのアクセス

前提条件

- 稼働中の CodeReady コンテナー仮想マシン。詳細は、「仮想マシンの起動」を参照してください。

手順

OpenShift Web コンソールにアクセスするには、以下の手順に従います。

1. **crc コンソールを実行します**。これにより、Web ブラウザーが開かれ、Web コンソールに転送されます。
2. **crc start コマンドの出力に出力されるパスワードがある開発者ユーザーとしてログインします**。

注記

- また、crc **console --credentials を実行して、開発者および kubeadmin** ユーザーのパスワードを表示することもできます。
- 最初に **kubeadmin** または **developer** ユーザーのいずれかを使用してクラスターにアクセスできます。**開発者ユーザーを使用して**、プロジェクトまたは OpenShift アプリケーションおよびアプリケーションデプロイメント用に使用します。新規ユーザーの作成、ロールの設定など、管理タスクに **kubeadmin** ユーザーのみを使用します。

CodeReady Containers OpenShift クラスターにアクセスできない場合は、「Troubleshooting CodeReady Containers」を参照してください。

関連情報

- OpenShift ドキュメントは、プロジェクトおよびアプリケーションの作成について説明します。

3.3.2. ocを使用した OpenShift クラスターへのアクセス

前提条件

- 稼働中の CodeReady コンテナー仮想マシン。詳細は、「仮想マシンの起動 」を参照してください。

手順

oc コマンドを使用して OpenShift クラスターにアクセスするには、以下の手順に従います。

1. **crc oc-env** コマンドを実行して、キャッシュされた **oc 実行可能ファイルをPATH** に追加するのに必要なコマンドを出力します。

```
$ crc oc-env
```

2. 出力コマンドを実行します。
3. **developer** ユーザーとしてログインします。

```
$ oc login -u developer https://api.crc.testing:6443
```

注記

crc start コマンドは、開発者ユーザーのパスワードを出力します。crc console --credentials コマンドを実行して表示することもできます。

4. **oc** を使用して OpenShift クラスターと対話できるようになりました。たとえば、OpenShift クラスター Operator **が利用可能であることを確認するには、kubeadmin** ユーザーとしてログインし、以下のコマンドを実行します。

```
$ oc config use-context crc-admin
$ oc whoami
kubeadmin
$ oc get co
```

注記

- CodeReady **コンテナーはデフォルトでモニタリング** Operator を無効にします。

CodeReady Containers OpenShift クラスターにアクセスできない場合は、「Troubleshooting CodeReady Containers」を参照してください。

関連情報

- OpenShift ドキュメントは、プロジェクトおよびアプリケーションの作成について説明します。

3.3.3. 内部 OpenShift レジストリーへのアクセス

CodeReady コンテナーの仮想マシンで実行されている OpenShift クラスターには、デフォルトで内部コンテナーイメージレジストリーが含まれます。この内部コンテナーイメージレジストリーを、ローカルに開発したコンテナーイメージのパブリケーションターゲットとして使用できます。内部 OpenShift レジストリーにアクセスするには、以下の手順に従います。

前提条件

- 稼働中の CodeReady コンテナー仮想マシン。詳細は、「仮想マシンの起動 」を参照してください。
- 稼働中の **oc** コマンドです。詳細は、「Accessing the OpenShift cluster with **oc** 」を参照してください。
- **podman** または **docker** のインストール。
  - Docker **の場合は、default-route-openshift-image-registry.apps-crc.testing** を非セキュアなレジストリーとして追加します。詳細は、Docker ドキュメントを参照してください。

手順

1. どのユーザーがクラスターにログインしているかを確認します。

```
$ oc whoami
```

注記

デモの目的で、現在のユーザーは **kubeadmin** であると想定されます。

2. そのトークンでレジストリーにログインします。

```
$ podman login -u kubeadmin -p $(oc whoami -t) default-route-openshift-image-registry.apps-crc.testing --tls-verify=false
```

3. 新しいプロジェクトを作成します。

```
$ oc new-project demo
```

4. サンプルコンテナーイメージをプルします。

```
$ podman pull quay.io/libpod/alpine
```

5. namespace の詳細を含む、イメージにタグを付けます。

```
$ podman tag alpine:latest default-route-openshift-image-registry.apps-crc.testing/demo/alpine:latest
```

6. コンテナーイメージを内部レジストリーにプッシュします。

```
$ podman push default-route-openshift-image-registry.apps-crc.testing/demo/alpine:latest --tls-verify=false
```

7. イメージストリームを取得し、プッシュされたイメージが表示されていることを確認します。

```
$ oc get is
```

8. イメージストリームでイメージルックアップを有効にします。

```
$ oc set image-lookup alpine
```

この設定により、イメージストリームは内部レジストリーへの完全な URL を指定することなく、イメージのソースにすることができます。

9. 最近プッシュしたイメージを使用して Pod を作成します。

```
$ oc run demo --image=alpine --command -- sleep 600s
```

3.4. ODOを使用したサンプルアプリケーションのデプロイメント

odo を使用して、コマンドラインから OpenShift プロジェクトおよびアプリケーションを作成できます。この手順では、CodeReady Containers 仮想マシンで実行している OpenShift クラスターにサンプルアプリケーションをデプロイします。

前提条件

- **odo** がインストールされている。詳細は、odo ドキュメントの「Installing **odo** 」を参照してください。
- **CodeReady コンテナーの仮想マシンが実行中である。詳細は、「仮想マシンの起動 」を参照してください。**

手順

odo を使用してサンプルアプリケーションをデプロイするには、以下の手順を実行します。

1. **開発者ユーザーとして実行中の CodeReady Containers OpenShift クラスターにログインします。**

```
$ odo login -u developer -p developer
```

2. **アプリケーションのプロジェクトを作成します。**

```
$ odo project create sample-app
```

3. **コンポーネントのディレクトリーを作成します。**

```
$ mkdir sample-app
$ cd sample-app
```

4. **GitHub のサンプルアプリケーションからコンポーネントを作成します。**

```
$ odo create nodejs --s2i --git https://github.com/openshift/nodejs-ex
```

注記

リモート Git リポジトリーからコンポーネントを作成すると、odo push コマンドを実行する際に毎回アプリケーションが再ビルドされます。ローカルの Git リポジトリーからコンポーネントを作成するには、odo ドキュメントの「Creating a single-component application with odo 」を参照してください。

5. **URL を作成し、そのエントリーをローカル設定ファイルに追加します。**

```
$ odo url create --port 8080
```

6. **変更をプッシュします。**

```
$ odo push
```

これで、コンポーネントはアクセス可能な URL を使用してクラスターにデプロイされます。

7. URL を一覧表示し、コンポーネントの必要な URL を確認します。

```
$ odo url list
```

8. 生成された URL を使用してデプロイされたアプリケーションを表示します。

関連情報

- odo の使用についての詳細は、odo ドキュメントを参照してください。

3.5. 仮想マシンの停止

crc stop コマンドは、実行中の CodeReady Containers 仮想マシンおよび OpenShift クラスターを停止します。クラスターのシャットダウン中は、停止プロセスには数分かかります。

手順

- CodeReady Containers 仮想マシンおよび OpenShift クラスターを停止します。

```
$ crc stop
```

3.6. 仮想マシンの削除

crc delete コマンドは、既存の CodeReady Containers 仮想マシンを削除します。

手順

- CodeReady Containers 仮想マシンを削除します。

```
$ crc delete
```

警告

crc delete コマンドを使用すると、CodeReady Containers 仮想マシンに保存されているデータが失われます。このコマンドを実行する前に、仮想マシンに保存されている必要な情報を保存します。

第4章 CODEREADY コンテナーの設定

4.1. CODEREADY コンテナー設定について

crc config コマンドを使用して、crc 実行ファイルと CodeReady Containers 仮想マシンの両方を設定します。crc config コマンドでは、設定で機能するサブコマンドが必要です。使用できるサブコマンドは get、set、un unset、および view です。get、set、および unset サブコマンドは名前付きの設定可能なプロパティーで動作します。crc config --help コマンドを実行して、利用可能なプロパティーを一覧表示します。

crc config コマンドを使用して、crc start コマンドおよび crc setup コマンドの起動チェックの動作を設定することもできます。デフォルトでは、起動を確認すると、エラーが報告され、条件が満たされると実行を停止します。check を省略するには、skip-check で開始するプロパティーの値を true に設定します。

4.2. CODEREADY コンテナー設定の表示

CodeReady Containers 実行可能ファイルは、設定可能なプロパティーと現在の CodeReady Containers 設定を表示するコマンドを提供します。

手順

- 利用可能な設定可能なプロパティーを表示するには、次のコマンドを実行します。

```
$ crc config --help
```

- 設定可能なプロパティーの値を表示するには、次のコマンドを実行します。

```
$ crc config get <property>
```

- 現在の設定をすべて表示するには、以下を実行します。

```
$ crc config view
```

注記

設定がデフォルト値で構成されている場合、crc config view コマンドは情報を返しません。

4.3. 仮想マシンの設定

cpus および memory プロパティーを使用して、CodeReady Containers 仮想マシンで利用可能なデフォルトの仮想 CPU 数およびメモリー容量を設定します。

または、--cpus および -- memory フラグを使用して、vCPU 数とメモリー容量をそれぞれ crc start コマンドに割り当てることができます。

重要

実行中の CodeReady コンテナー仮想マシンの設定を変更することはできません。設定の変更を有効にするには、実行中の仮想マシンを停止して、再び起動する必要があります。

手順

- 仮想マシンで利用可能な vCPU の数を設定するには、以下を実行します。

```
$ crc config set cpus <number>
```

cpus プロパティーのデフォルト値は 4 です。割り当てる vCPU の数は、デフォルト以上である必要があります。

- 必要な数の仮想 CPU で仮想マシンを起動するには、以下を実行します。

```
$ crc start --cpus <number>
```

- 仮想マシンで利用可能なメモリーを設定するには、以下を実行します。

```
$ crc config set memory <number-in-mib>
```

注記

使用可能なメモリーの値は、メビバイト(MiB)で設定されます。メモリー 1 つ(GiB)は 1024 MiB に相当します。

memory プロパティーのデフォルト値は9216 です。割り当てるメモリーの量は、デフォルト以上である必要があります。

- **必要なメモリー量で仮想マシンを起動するには、以下を実行します。**

```
$ crc start --memory <number-in-mib>
```

第5章 ネットワーク

5.1. DNS 設定の詳細

5.1.1. 一般的な DNS 設定

CodeReady コンテナーで管理される OpenShift クラスターは、2 つの DNS ドメイン名 crc.testing および apps-crc.testing を使用します。crc.testing ドメインは、コアの OpenShift サービス用です。apps-crc.testing ドメインは、クラスター上にデプロイされた OpenShift アプリケーションにアクセスするために使用します。

たとえば、OpenShift API サーバーは console-openshift-console.apps-crc.testing を使用して OpenShift コンソールにアクセスしている間に api.crc.testing として公開されます。これらの DNS ドメインは、CodeReady Containers 仮想マシン内で実行される dnsmasq DNS コンテナーによって提供されます。

crc 設定を実行すると、これらのドメインを解決できるように、システムの DNS 設定を検出して調整します。crc 起動時に DNS が適切に設定されていることを確認するために、追加のチェックが行われます。

5.1.2. Linux

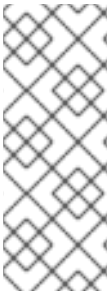
Linux では、ディストリビューションによっては、CodeReady コンテナーは以下の DNS 設定を想定します。

5.1.2.1. NetworkManager + systemd-resolved

この設定は、Fedora 33 以降および Ubuntu Desktop editions でデフォルトで使用されます。

- CodeReady コンテナーは NetworkManager がネットワークを管理することを想定します。
- CodeReady コンテナーは、テストドメインの要求を 192.168.130.11 DNS サーバーに転送するように systemd-resolved を設定します。192.168.130.11 は CodeReady コンテナーの仮想マシンの IP です。
- systemd-resolved 設定は、/etc/NetworkManager/dispatcher.d/99-crc.sh の NetworkManager の dispatcher スクリプトを使用して行います。

```
#!/bin/sh

export LC_ALL=C

systemd-resolve --interface crc --set-dns 192.168.130.11 --set-domain ~testing

exit 0
```



注記

systemd-resolved は、Red Hat Enterprise Linux および CentOS 8.3 では、サポート対象外のテクノロジープレビューとしても利用できます。systemd-resolved を使用するようにホストを設定したら、実行中のクラスターを停止して、crc 設定を再実行します。

5.1.2.2. NetworkManager + dnsmasq

この設定は、Fedora 32 以前、Red Hat Enterprise Linux ではデフォルトで、CentOS で使用されます。

- **CodeReady コンテナーは NetworkManager がネットワークを管理することを想定します。**
- **NetworkManager は、/etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf 設定ファイルを介して dnsmasq を使用します。**
- **この dnsmasq インスタンスの設定ファイルは /etc/NetworkManager/dnsmasq.d/crc.conf になります。**

```
server=/crc.testing/192.168.130.11
server=/apps-crc.testing/192.168.130.11
```

  - **NetworkManager dnsmasq インスタンスは、crc.testing ドメインおよび apps-crc.testing ドメインのリクエストを 192.168.130.11 DNS サーバーに転送します。**

5.2. 予約された IP サブネット

CodeReady Containers OpenShift クラスターは、内部使用の IP サブネットを確保します。これはホストネットワークと競合しないようにしてください。以下の IP サブネットが使用可能であることを

確認します。

予約された IP サブネット

- 10.217.0.0/22
- 10.217.4.0/23
- 192.168.126.0/24

また、ホストハイパーバイザーがホストオペレーティングシステムに応じて別の IP サブネットを確保する場合があります。Microsoft Windows では、ハイパーバイザーは、事前に決定できない無作為に生成される IP サブネットを確保します。macOS では、追加のサブネットが予約されません。Linux 用に追加の予約済みサブネットは 192.168.130.0/24 です。

5.3. プロキシーの背後での CODEREADY コンテナーの起動

前提条件

- ホストマシンで既存の oc executable を使用するには、no_proxy 環境変数の一部として.testing ドメインをエクスポートします。
- 組み込まれた oc 実行可能ファイルには手動での設定は必要ありません。組み込まれた oc 実行可能ファイルの使用についての詳細は、「Accessing the OpenShift cluster with oc 」を参照してください。

手順

1. http_proxy および https_proxy 環境変数を使用するか、以下のように crc config set コマンドを使用してプロキシーを定義します。

```
$ crc config set http-proxy http://proxy.example.com:<port>
$ crc config set https-proxy http://proxy.example.com:<port>
$ crc config set no-proxy <comma-separated-no-proxy-entries>
```

2. プロキシーがカスタム CA 証明書ファイルを使用する場合は、以下のように設定します。

```
$ crc config set proxy-ca-file <path-to-custom-ca-file>
```

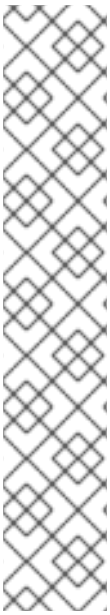
crc 実行可能ファイルは、環境変数または CodeReady コンテナーの設定を介して設定された後に定義されたプロキシーを使用できます。

注記

- CodeReady コンテナーの設定で設定されるプロキシー関連の値は、環境変数で設定された値よりも優先されます。
- SOCKS プロキシーは OpenShift Container Platform ではサポートされません。

5.4. リモートサーバーでの CODEREADY コンテナーの設定

以下の手順に従って、CodeReady Containers OpenShift クラスターを実行するようにリモートサーバーを設定します。



注記

- ローカルネットワーク上でこの手順を実行することが強く推奨されます。インターネット上でセキュアではないサーバーを公開するには、多くのセキュリティー影響があります。
- この手順のコマンドはすべて、リモートサーバーで実行する必要があります。
- この手順では、Red Hat Enterprise Linux、Fedora、または CentOS サーバーを使用することを前提としています。

前提条件

- CodeReady コンテナーはリモートサーバーにインストールされ、設定されている。詳細は、「CodeReady コンテナーのインストール」および「CodeReady コンテナーの設定」を参照してください。

- ユーザーアカウントに、リモートサーバーで sudo パーミッションがある。

手順

1. クラスターを起動します。

```
$ crc start
```

この手順の実行中に、クラスターの実行を継続していることを確認します。

2. haproxy パッケージおよびその他のユーティリティーをインストールします。

```
$ sudo dnf install haproxy /usr/sbin/semanage
```

3. クラスターとの通信を許可するようにファイアウォールを変更します。

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --add-port=80/tcp --permanent
$ sudo firewall-cmd --add-port=6443/tcp --permanent
$ sudo firewall-cmd --add-port=443/tcp --permanent
$ sudo systemctl restart firewalld
```

4. SELinux の場合、TCP ポート 6443 のリッスンを許可します。

```
$ sudo semanage port -a -t http_port_t -p tcp 6443
```

5. デフォルトの haproxy 設定のバックアップを作成します。

```
$ sudo cp /etc/haproxy/haproxy.cfg{,.bak}
```

6. クラスターで使用する haproxy を設定します。

```
$ export CRC_IP=$(crc ip)
$ sudo tee /etc/haproxy/haproxy.cfg &>/dev/null <<EOF
global
    log /dev/log local0

defaults
```

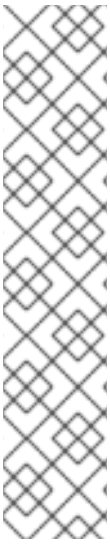
```
    balance roundrobin
    log global
    maxconn 100
    mode tcp
    timeout connect 5s
    timeout client 500s
    timeout server 500s

listen apps
    bind 0.0.0.0:80
    server crcvm $CRC_IP:80 check

listen apps_ssl
    bind 0.0.0.0:443
    server crcvm $CRC_IP:443 check

listen api
    bind 0.0.0.0:6443
    server crcvm $CRC_IP:6443 check
EOF
```

7. **haproxy サービスを起動します。**

```
$ sudo systemctl start haproxy
```

5.5. リモート CODEREADY コンテナーインスタンスへの接続

以下の手順に従って、CodeReady Containers OpenShift クラスターを実行するリモートサーバーにクライアントマシンに接続します。



注記

- **ローカルネットワークでのみ公開されるサーバーに接続することが強く推奨されます。**
- **この手順のコマンドはすべてクライアントで実行する必要があります。**
- **この手順では、Red Hat Enterprise Linux、Fedora、または CentOS クライアントを使用することを前提としています。**

前提条件

- **クライアントが接続するリモートサーバーが設定されている。詳細は、「リモートサーバー**

での CodeReady コンテナーの設定 」を参照してください。

- NetworkManager がインストールされ、実行している。
- サーバーの外部 IP アドレスを把握している。
- クライアントの $PATH に最新の OpenShift クライアント実行ファイル(oc) がある。

手順

1. dnsmasq パッケージをインストールします。

```
$ sudo dnf install dnsmasq
```

2. NetworkManager で DNS 解決に dnsmasq を使用できるようにします。

```
$ sudo tee /etc/NetworkManager/conf.d/use-dnsmasq.conf &>/dev/null <<EOF
[main]
dns=dnsmasq
EOF
```

3. CodeReady コンテナーの DNS エントリーを dnsmasq 設定に追加します。

```
$ sudo tee /etc/NetworkManager/dnsmasq.d/external-crc.conf &>/dev/null <<EOF
address=/apps-crc.testing/SERVER_IP_ADDRESS
address=/api.crc.testing/SERVER_IP_ADDRESS
EOF
```

注記

/etc/NetworkManager/dnsmasq.d/crc.conf の既存のエントリーをコメントアウトします。これらのエントリーは、CodeReady コンテナーのローカルインスタンスを実行して作成され、リモートクラスターのエントリーと競合します。

4. NetworkManager サービスを再読み込みします。

```
$ sudo systemctl reload NetworkManager
```

5. **oc が指定された developer ユーザーとしてリモートクラスターにログインします。**

```
$ oc login -u developer -p developer https://api.crc.testing:6443
```

リモート OpenShift Web コンソールは https://console-openshift-console.apps-crc.testing で利用できます。

第6章 管理タスク

6.1. モニタリング、アラート、および TELEMETRY の起動

CodeReady コンテナーが通常のラップトップで実行できるようにするには、一部のリソースカッビングサービスはデフォルトで無効になります。上記のいずれかが Prometheus および関連するモニタリング、アラート、および Telemetry 機能のいずれかになります。Telemetry 機能は、Red Hat OpenShift Cluster Manager にクラスターを一覧表示します。

前提条件

- 追加のメモリーを CodeReady コンテナーの仮想マシンに割り当てる必要があります。コア機能には、少なくとも 14 GiB のメモリー(14336)が推奨されます。ワークロードを増やすには、より多くのメモリーが必要になります。詳細は、「仮想マシンの設定」を参照してください。

手順

1. enable-cluster-monitoring 設定可能プロパティーを true に設定します。

```
$ crc config set enable-cluster-monitoring true
```

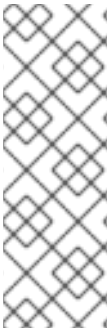
2. 仮想マシンを起動します。

```
$ crc start
```

警告

クラスターモニタリングを無効にできません。モニタリング、アラート、および Telemetry を削除するには、enable-cluster-monitoring の設定可能なプロパティーを false に設定し、既存の CodeReady Containers 仮想マシンを削除します。

第7章 RED HAT CODEREADY CONTAINERS のトラブルシューティング



注記

Red Hat CodeReady Containers の目的は、開発およびテスト目的で OpenShift 環境を提供することを目的としています。特定の OpenShift アプリケーションのインストール時または使用時に発生する問題は、CodeReady Container の範囲外にあります。このような問題を関連するプロジェクトに報告します。たとえば、OpenShift は GitHub の問題を追跡します。

7.1. OPENSHIFT クラスターへのシェルアクセスの取得

OpenShift クラスターへの直接アクセスは、通常の使用には必要なく、強く推奨されません。トラブルシューティングまたはデバッグの目的でクラスターにアクセスするには、以下の手順に従います。

前提条件

- クラスターへの oc アクセスを有効にし、kubeadmin ユーザーとしてログインします。詳細な手順については、「Accessing the OpenShift cluster with oc 」を参照してください。

手順

1. oc get nodes を実行します。出力は以下のようになります。

```
$ oc get nodes
NAME                 STATUS   ROLES          AGE    VERSION
crc-shdl4-master-0   Ready    master,worker   7d7h   v1.14.6+7e13ab9a7
```

2. oc debug nodes/<node> を実行します。ここで、<node> は直前の手順で出力されるノードの名前です。

7.2. 期限切れの証明書のトラブルシューティング

リリースされた各 crc 実行のシステムバンドルは、リリース後の 30 日後に有効期限が切れます。この有効期限は、OpenShift クラスターに組み込まれた証明書が原因です。crc start コマンドは、必要に応じて自動証明書の更新プロセスをトリガーします。証明書の更新は、最大 5 分からクラスターの開始時間に追加できます。

この追加の起動時間を回避したり、証明書の更新プロセスで障害が発生した場合に、以下の手順を使用します。

手順

自動的に更新されない期限切れの証明書エラーを解決するには、以下を実行します。

1. 最新の CodeReady Containers リリースをダウンロードし、crc 実行可能ファイルを $PATH に配置します。

2. crc delete コマンドを使用して、証明書エラーのあるクラスターを削除します。

```
$ crc delete
```

警告

crc delete コマンドを使用すると、CodeReady Containers 仮想マシンに保存されているデータが失われます。このコマンドを実行する前に、仮想マシンに保存されている必要な情報を保存します。

3. 新しいリリースを設定します。

```
$ crc setup
```

4. 新しい仮想マシンを起動します。

```
$ crc start
```

7.3. バンドルバージョンの不一致のトラブルシューティング

作成された CodeReady Containers 仮想マシンには、バンドル情報およびインスタンスデータが含まれます。バンドル情報およびインスタンスデータは、新しい CodeReady Containers リリースの設定時に更新されません。この情報は、以前のインスタンスデータのカスタマイズが原因で更新されません。これにより、crc start コマンドの実行時にエラーが発生します。

```
$ crc start
```

```
...
FATA Bundle 'crc_hyperkit_4.2.8.crcbundle' was requested, but the existing VM is using
'crc_hyperkit_4.2.2.crcbundle'
```

手順

1. **インスタンスを起動する前に crc delete コマンドを実行します。**

```
$ crc delete
```

警告

crc delete コマンドを使用すると、CodeReady Containers 仮想マシンに保存されているデータが失われます。このコマンドを実行する前に、仮想マシンに保存されている必要な情報を保存します。

7.4. 不明な問題のトラブルシューティング

クリーンな状態で CodeReady コンテナーを再起動して、ほとんどの問題を解決します。これには、仮想マシンを停止し、削除し、crc setup コマンドで加えられた変更を元に戻し、それらの変更を再度適用し、仮想マシンを再起動します。

前提条件

- **crc setup コマンドでホストマシンを設定します。詳細は、「CodeReady コンテナーの設定」を参照してください。**
- **crc start コマンドを使用して、CodeReady コンテナーを開始している。詳細は、「仮想マシンの起動」を参照してください。**
- **最新の CodeReady Containers リリースを使用している。CodeReady Containers 1.2.0 より前のバージョンを使用すると、x509 証明書の期限切れに関連するエラーが発生する可能性があります。詳細は、「期限切れの証明書のトラブルシューティング」を参照してください。**

手順

CodeReady コンテナーのトラブルシューティングを行うには、以下の手順を実行します。

1. CodeReady Containers 仮想マシンを停止します。

```
$ crc stop
```

2. CodeReady Containers 仮想マシンを削除します。

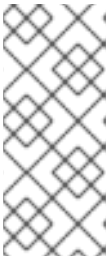
```
$ crc delete
```

警告

crc delete コマンドを使用すると、CodeReady Containers 仮想マシンに保存されているデータが失われます。このコマンドを実行する前に、仮想マシンに保存されている必要な情報を保存します。

3. crc setup コマンドで残りの変更をクリーンアップします。

```
$ crc cleanup
```



注記

crc cleanup コマンドは、既存の CodeReady Containers 仮想マシンを削除し、crc setup コマンドで作成された DNS エントリーへの変更を元に戻します。macOS では、crc cleanup コマンドはシステムトレイも削除されます。

4. ホストマシンを設定し、変更を再適用します。

```
$ crc setup
```

5. CodeReady Containers 仮想マシンを起動します。

```
$ crc start
```

注記

クラスターは、要求を提供する前に必要なコンテナーおよび Operator を起動するまでに最低でも 4 分かかります。

この手順で問題が解決しない場合は、以下の手順を実行します。

1. 発生した問題についての未解決の問題を検索します。

2. 発生した問題に対応する既存の問題がない場合には、問題を作成し 、~/.crc/crc.log ファイルを作成した問題に割り当てます。~/.crc/crc.log ファイルには、詳細なデバッグおよびトラブルシューティングの情報が含まれています。これは、発生している問題の診断に役立ちます。