Red Hat CodeReady Containers 1.21

スタートガイド

CodeReady コンテナー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: 2023-05-15

Red Hat CodeReady Containers 1.21 スタートガイド

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

Kevin Owen
kowen@redhat.com

法律上の通知

概要

このガイドでは、CodeReady Containers を使用して速度を高める方法を説明します。ホストワークステーション (Microsoft Windows、macOS、または Red Hat Enterprise Linux) から Red Hat OpenShift Container Platform 4 を使用してコンテナー化されたアプリケーションを開発する最初のステップに関する手順および例が記載されています。

目次

多様性を受け入れるオープンソースの強化

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

第1章 RED HAT CODEREADY CONTAINERS のご紹介

1.1. CODEREADY コンテナーについて

Red Hat CodeReady Containers は、ローカルの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低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。ヘッドレスまたは複数開発者の設定などの他のユースケースの場合は、完全な OpenShift インストーラー を使用します。

OpenShift の詳細な紹介は、OpenShift ドキュメントを参照してください。

CodeReady Containers には、OpenShift クラスターを実行する CodeReady Container 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イス (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、以下の主な変更点を使用した通常の OpenShift インストールです。

- **CodeReady Containers OpenShift クラスターは一時的なクラスターであり、実稼働環境での使用を目的としていません。**
- 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。
- デフォルトでは、**machine-config** と **monitoring** Operator が無効になります。
  - この無効な 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。
  - 同じ理由で、新しい OpenShift バージョンへのアップグレードパスはありません。
- 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。これらの設定は変更しないでください。

- ***.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の範囲。
  - 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てい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. 最小システム要件

CodeReady Containers の最小ハードウェアおよびオペレーティングシステムの要件は以下のとおりです。

2.1.1. ハードウェア要件

CodeReady Containers には以下のシステムリソースが必要です。

- 4 つの仮想 CPU (vCPU)
- 空きメモリー 9 GB
- ストレージ領域の 35 GB

注記

OpenShift クラスターでは、CodeReady Containers 仮想マシンで実行するのに必要な最小リソースが必要です。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 Containers 仮想マシンにより多く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要件

CodeReady Containers には、サポートされるオペレーティングシステムの最小バージョンが必要です。

2.1.2.1. Microsoft Windows

- Microsoft Windows では、CodeReady Containers には Windows 10 Fall Creators Update (バージョン 1709) 以降が必要です。CodeReady Containers は、Microsoft Windows の以前のバージョンでは動作しません。Microsoft Windows 10 Home Edition はサポートされません。

2.1.2.2. macOS

- macOS の CodeReady Containers には macOS 10.12 Sierra 以降が必要です。CodeReady Containers は、macOS の以前のバージョンで動作しません。

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 を使用する場合は、CodeReady Containers を実行するマシンが Red Hat カスタマーポータルに登録されている 必要があります。
- Ubuntu 18.04 LTS 以降および Debian 10 以降は公式にサポートされておらず、ホストマシンの手動設定が必要になる場合があります。
- 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 コンテナーでは、**libvirt** および **NetworkManager** パッケージが Linux 上で実行する必要があります。Linux ディストリビューションで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 ディストリビューション	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

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. ご使用のプラットフォーム用の CodeReady Containers の最新リリース をダウンロードし、アーカイブの内容を **PATH** 内の場所に抽出します。

2.4. CODEREADY コンテナーのアップグレード

CodeReady Containers 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状態を防ぐために手動の設定が必要になります。

手順

1. 最新リリースの CodeReady Containers 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 コマンドは、CodeReady コンテナーの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** 実行可能ファイルが使用中であることを確認します。

```
$ crc version
```

4. 新しい 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 コマンドは操作を実行し、CodeReady Containers 仮想マシンのホストマシンの環境を設定します。

~/**.crc** ディレクトリーが存在しない場合は、この手順を作成します。

前提条件

- Linux または macOS の場合は、ユーザーアカウントに **sudo** コマンドを使用できるようになっている。Microsoft Windows で、ユーザーアカウントが管理者権限で昇格できるようになっている。

注記

- **root**(または管理者) として **crc** 実行可能ファイルを実行しないでください。**crc** 実行ファイルは常にユーザーアカウントで実行します。
- 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、仮想マシンに加えられた変更をすべて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
```

テレメトリーデータ収集の継続

crc setup コマンドは、開発を支援するために、オプション、匿名使用データの収集を要求します。個人的識別可能な情報が収集されません。

- テレメトリー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
```

- テレメトリー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
```

収集されるデータの詳細は、Red Hat テレメトリーデータ収集に関する通知 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 に接続されておらず、ネットワーク接続が信頼できることを確認します。

- **crc setup** コマンドを使用して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で昇格できるようになっている。
- 有効な OpenShift ユーザープルシークレットがある。cloud.redhat.com の Install on Laptop: Red Hat CodeReady Containers ページの 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

ユーザーの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 コンテナーの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、ユーザーの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 を起動するのに少なくとも 4 分かかります。
- **crc start** 時にエラーが表示される場合は、CodeReady コンテナーのトラブルシューティングセクション で潜在的な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の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 コンテナーの仮想マシン。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 console** を実行します。これにより、Web ブラウザーが開き、Web コンソールに転送されます。

2. OpenShift Web コンソールで **htpasswd_provider** オプションを選択します。
3. **crc start** コマンドの出力でパスワードが出力された **developer** ユーザーとしてログインします。

注記

- **crc console --credentials** を実行して **developer** および **kubeadmin** ユーザーのパスワードを確認することもできます。
- **kubeadmin** または **developer** ユーザーのいずれかを使用して、まずクラスターにアクセスできます。プロジェクトまたは OpenShift アプリケーションを作成するために、**developer** ユーザーを使用し、アプリケーションのデプロイメントに使用します。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、CodeReady コンテナーのトラブルシューティング を参照してください。

関連情報

- OpenShift ドキュメント は、プロジェクトと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 コンテナーの仮想マシン。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、以下の手順を実行します。

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** に追加します。

```
$ crc oc-env
```

2. 印刷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、**developer** 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 が使用可能であることを確認するには、以下を実行します。

```
$ oc get co
```

注記

- CodeReady Containers のデフォルトでは、**machine-config** と **monitoring** Operator が無効になります。

CodeReady Containers OpenShift クラスターにアクセスできない場合は、CodeReady コンテナーのトラブルシューティング を参照してください。

関連情報

- OpenShift ドキュメント は、プロジェクトと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 Containers 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は、ローカル開発コンテナーイメージの公開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 コンテナーの仮想マシン。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。詳細は、**oc** を使用した OpenShift クラスターへのアクセス 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. クラスターにログインしているユーザーを確認します。

```
$ oc whoami
```

注記

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

2. トークンでそのユーザーとして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内のすべてのタグのイメージ検索を有効にします。

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

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の完全な URL を指定することなくイメージのソースになります。

9. 最近プッシュされ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

OpenShift Do (**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 Container 仮想マシンで実行されている 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 **odo** のインストール を参照してください。
- CodeReady Containers 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 でサンプルアプリケーションをデプロイするには、以下の手順に従います。

1. **developer** ユーザーとして、実行中の CodeReady Container 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たびにアプリケーションが再ビルドされます。ローカル 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 **odo** で単一コンポーネントアプリケーションの作成 を参照してください。

5. URL を作成し、ローカル設定ファイルにエントリーを追加します。

```
$ 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 でクラスターにデプロイされます。

7. URL を一覧表示し、コンポーネントに必要な 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 コンテナー仮想マシンおよび OpenShift クラスターを停止します。停止プロセスには、クラスターがシャットダウンするまで数分かかります。

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 コンテナーの仮想マシンを削除します。

手順

- CodeReady コンテナーの仮想マシンを削除します。

```
$ crc delete
```

警告

crc delete コマンドは、CodeReady コンテナーの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** 実行可能ファイルと CodeReady Containers 仮想マシンの両方を設定します。**crc config** コマンドには、設定で機能するサブコマンドが必要です。利用可能なサブコマンドは、**get**、**set**、**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** 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、起動はエラーを確認し、条件が満たされない場合に実行を停止します。**skip-check** を **true** に設定して、チェックをスキップします。

4.2. CODEREADY コンテナー設定の表示

CodeReady コンテナーの実行ファイルは、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、以下を実行します。

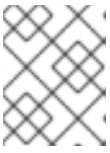
```
$ crc config --help
```

- 設定可能なプロパティーの値を表示するには、以下を実行します。

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

- 現在の設定を完了するには、以下を実行します。

```
$ crc config view
```



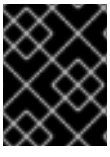
注記

crc config view コマンドは、設定がデフォルト値で設定されている場合に情報を返しません。

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 コンテナーの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、それぞれ **crc start** コマンドに --cpus および --memory フラグを使用して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。設定変更を有効にするには、実行中の仮想マシンを停止してから再起動する必要があります。

手順

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

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

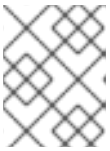
cpus プロパティーのデフォルト値は **4** です。割り当てる vCPU の数は、デフォルト以上である必要があります。

- 必要な数の vCPU 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>
```

- 仮想マシンが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

利用可能なメモリーの値は、メガバイト (MiB) で設定されます。メモリーの 1 つ (GiB) は 1024 MiB と等しくなります。

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 Containers によって管理される OpenShift クラスターは、2 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、OpenShift のコアサービス用です。**apps-crc.testing** ドメインは、クラスターにデプロイされた OpenShift アプリケーションにアクセスするためのものです。

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 コンテナーの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 setup を実行すると、システムの DNS 設定を調整して、これらのドメインを解決できるようにします。**crc start** を起動する際に DNS が適切に設定されていることを確認するには、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 デスクトップエディションで使用されます。

- 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、**testing** 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 スクリプトで行います。

```
resolvectl domain crc ~testing
resolvectl dns crc 192.168.130.11
resolvectl default-route crc false

exit 0
```

5.1.2.2. NetworkManager + dnsmasq

この設定は、Fedora 32 以前、Red Hat Enterprise Linux、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** です。

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

  - NetworkManager の **dnsmasq** インスタンスは、**crc.testing** 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。

5.1.3. macOS

macOS では、CodeReady Containers は次の DNS 設定を想定します。

- CodeReady Containers は、**testing** ドメインに対するすべての DNS リクエストを CodeReady Containers 仮想マシンに転送するように macOS に指示する **/etc/resolver/testing** ファイルを作成します。
- CodeReady Containers は、VM IP アドレスを指す **api.crc.testing** エントリーを **/etc/hosts** に追加します。**oc** 実行可能ファイルにはこのエントリーが必要です。詳細は、OpenShift issue #23266 を参照してください。

5.2. プロキシーの背後にある CODEREADY コンテナーの開始

前提条件

- ホストマシンで既存の **oc** 実行可能ファイルを使用するには、**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。
- 組み込み **oc** 実行可能ファイルには手動設定は必要ありません。埋め込み **oc** 実行可能ファイルの使用に関する詳細は、**oc** を使用した OpenShift クラスターへのアクセス 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 コンテナー設定を介して設定された定義されたプロキシーを一度使用できます。

注記

- CodeReady コンテナーの設定に設定されたプロキシー関連の値は、環境変数を介して設定される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。

5.3. 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

- **この手順は、ローカルネットワークで実行することを強く推奨します。**インターネット上でセキュアでないサーバーを公開すると、多くのセキュリティーに影響が出ます。
- 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 コンテナーが、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 Containers のインストール および CodeReady Containers 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に対する **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
```

   この手順全体で、クラスターが稼働したままであ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 policycoreutils-python-utils jq
```

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 をリッスンしていることを許可します。

```
$ 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ように **haproxy** を設定します。

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

defaults
    log global
    mode http
    timeout connect 5000
    timeout client 500000
    timeout server 500000

frontend apps
    bind 0.0.0.0:80
    option tcplog
    mode tcp
    default_backend apps

frontend apps_ssl
    bind 0.0.0.0:443
    option tcplog
    mode tcp
    default_backend apps_ssl

backend apps
    mode tcp
    balance roundrobin
    server webserver1 $CRC_IP:80 check

backend apps_ssl
    mode tcp
    balance roundrobin
    option ssl-hello-chk
    server webserver1 $CRC_IP:443 check

frontend api
    bind 0.0.0.0:6443
    option tcplog
    mode tcp
    default_backend api

backend api
    mode tcp
    balance roundrobin
    option ssl-hello-chk
    server webserver1 $CRC_IP:6443 check
EOF
```

7. **haproxy** サービスを起動します。

```
$ sudo systemctl start haproxy
```

5.4. 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 Container 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を接続します。

注記

- **ローカルネットワーク上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

- リモートサーバーが、クライアントが接続するように設定されている。詳細は、リモートサーバーでの CodeReady Containers 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。

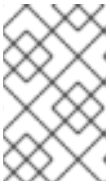
```
$ sudo dnf install dnsmasq
```

2. NetworkManager での DNS 解決に対する **dnsmasq** の使用を有効にします。

```
$ 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し、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** を使用して **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
```

リモートの OpenShift Web コンソールは https://console-openshift-console.apps-crc.testing から入手できます。

第6章 管理タスク

6.1. MONITORING、ALERTING、および TELEMETRY の起動

CodeReady コンテナーの一般的なラップトップで実行されるようにするには、リソース負荷サービスの一部がデフォルトで無効になります。これらのいずれかが Prometheus および関連するモニターリング、アラート、および Telemetry 機能です。Telemetry 機能は、Red Hat OpenShift Cluster Manager で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 14 GiB 以上のメモリー (値は **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です。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
```

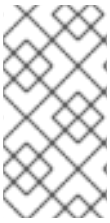
2. 仮想マシンを起動します。

```
$ crc start
```

警告

クラスターモニターリングを無効にできません。モニターリング、アラート、および Telemetry を削除するには、**enable-cluster-monitoring** 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、開発およびテストの目的で OpenShift 環境を提供します。特定の OpenShift アプリケーションのインストール時に生じる問題は、CodeReady コンテナーのスコープ外にあります。該当するプロジェクトに、このような問題を報告します。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ではなく、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。詳細な手順は、**oc** を使用した OpenShift クラスターへのアクセス 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。ここでの **<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

注記

CodeReady Containers 1.10.0 リリースの時点では、証明書の更新プロセスが意図したとおりに機能していません。証明書の有効期限による潜在的なエラーを回避するには、次の手順に従ってください。

リリースされた各 **crc** 実行可能ファイルのシステムバンドルは、リリース後に 30 日後に有効期限が切れます。この有効期限は、OpenShift クラスターに埋め込まれた証明書が原因で行われます。その結果、古い **crc** 実行可能ファイルまたはシステムバンドルを使用すると、証明書の期限切れエラーが発生する可能性があります。

CodeReady Containers 1.2.0 以降、埋め込み証明書は **crc** によって自動的に更新できるようになりました。**crc start** コマンドは、必要に応じて証明書の更新プロセスをトリガーします。証明書の更新では、クラスターの起動時間に最大 5 分後に追加できます。

手順

自動的に更新でき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、証明書エラーでクラスターを削除します。

```
$ crc delete
```

警告

crc delete コマンドは、CodeReady コンテナーの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 Container 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。新規の CodeReady Containers リリースの設定時には、バンドル情報およびインスタンスデータは更新されません。この情報は、以前のインスタンスデータのカスタマイズにより更新されません。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 コマンドは、CodeReady コンテナーの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することで、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、削除して、**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て仮想マシンを再起動する必要があります。

前提条件

- **crc setup** コマンドを使用して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 コンテナーを起動している。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 よりも前のバージョンを使用すると、期限切れの x509 証明書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 コンテナーの仮想マシンを停止します。

```
$ crc stop
```

2. CodeReady コンテナーの仮想マシンを削除します。

```
$ crc delete
```

警告

crc delete コマンドは、CodeReady コンテナーの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** コンテナー仮想マシンを削除し、crc setup コマンドで作成した DNS エントリーへの変更に戻ります。macOS では、**crc cleanup** コマンドはシステムトレイも削除します。

4. 変更を適用するためにホストマシンを設定します。

```
$ crc setup
```

5. CodeReady コンテナーの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 を起動するのに少なくとも 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. 発生した問題のオ ープン問題を検索 します。
2. 既存の問題が問題に対処しない場合は、問題を作成し、**~/.crc/crc.log** ファイルを作成され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とトラブルシューティング情報があり、発生した問題を診断するのに役立ちます。