Red Hat CodeReady Containers 2.0

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: 2022-04-11

Red Hat CodeReady Containers 2.0 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

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

概要

本ガイドでは、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した最初のステップについての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、ローカルコンピューターに最小限の OpenShift Container Platform 4 クラスターおよび Podman コンテナーランタイムを提供します。これらのランタイムは、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。ヘッドレスや複数開発者の設定など、他の OpenShift Container Platform のユースケースについては、完全な OpenShift インストーラー を使用します。

OpenShift Container Platform の概要については、OpenShift ドキュメント を参照してください。

CodeReady コンテナーには、必要なコンテナーランタイムを使用して CodeReady コンテナーインスタンスと対話するための **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 コンテナーの OpenShift の Preset は、以下の主な違いを使用して通常の OpenShift Container Platform インストールを提供します。

- **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。**
- **CodeReady Containersには、より新しいOpenShiftのバージョンへのアップグレードパスが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。
- コントロールプレーンとワーカーノードの両方として動作する単一のノードを使用します。
- デフォルトではCluster Monitoring Operatorを無効にします。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。
- OpenShift クラスターは、**インスタンス**と呼ばれる仮想マシンで実行します。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 Containers が提供する OpenShift クラスターには、以下のカスタマイズができないクラスター設定も含まれています。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。

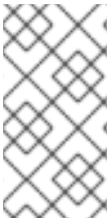
2.1.1. ハードウェア要件

CodeReady Containersは、AMD64およびIntel 64プロセッサアーキテクチャでのみサポートされています。CodeReady Containersは、ARMベースのM1アーキテクチャをサポートしていません。CodeReady Containersは、ネストされた仮想化をサポートしていません。

必要なコンテナーランタイムに応じて、CodeReady コンテナーには以下のシステムリソースが必要です。

2.1.1.1. OpenShift Container Platform の場合

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



注記

OpenShift Container Platform クラスターでは、CodeReady コンテナーインスタンス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 コンテナーインスタンスにより多くのリソースを割り当てるには、「インスタンス の 設定」を参照し てください。

2.1.1.2. Podman コンテナーランタイムの場合

- 2 つの物理 CPU コア
- 2 GB の空きメモリー
- ストレージ領域の 35 GB

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 では、CodeReady Containers には macOS 10.14 Mojave 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

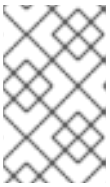
CodeReady コンテナーは、Red Hat Enterprise Linux の移植可能な実行ファイルとして使用できます。Microsoft Windows および macOS では、CodeReady コンテナーは、ガイド付きインストーラーを使用して利用できます。

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 CodeReady コンテナーの最新リリース 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 の場合は、ガイド付きインストーラーを実行し、手順に従います。



注記

Microsoft Windows では、CodeReady Containers をローカル **C:** ドライブにインストールする必要があります。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 の場合は、アーカイブが **~/Downloads** ディレクトリーにあるものとし、以下のステップを実行します。

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. **~/bin** ディレクトリーが存在しない場合は、作成します。また、**crc** 実行可能ファイルをコピーします。

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

c. **~/bin** ディレクトリーを **$ PATH** に追加します。

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

2.4. 使用状況データ収集について

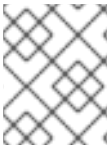
CodeReady Containers は、開発を支援するために、オプションの匿名使用データ収集を使用する前にプロンプトを表示します。個人的識別可能な情報が収集されません。使用状況データ収集の同意は、いつでも付与または取り消すことができます。

関連情報

- 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。
- 使用状況のデータ収集についての合意を許可または取り消すには、「使用 状況データの収集の設定 」を参照してください。

2.5. 使用状況データ収集の設定

使用状況データ収集の同意は、次の設定コマンドを使用していつでも付与または取り消すことができます。



注記

テレメトリー同意を変更しても、実行中のインスタンスは変更されません。変更は、次に **crc start** コマンドを実行したときに有効になります。

手順

- 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
```

関連情報

- 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。

2.6. 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 インスタンスを削除します。

```
$ crc delete
```

> **警告**
>
> **crc delete** コマンドを実行すると、CodeReady Containers インスタンス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、インスタンスに保存されている必要な情報を保存してください。

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. 新しい CodeReady Containers インスタンスを開始します。

```
$ crc start
```

第3章 CODEREADY コンテナーの使用

3.1. 事前設定について

CodeReady コンテナーの Preset は、管理対象のコンテナーランタイムと、その実行に必要なシステムリソースの少ない方を表します。CodeReady コンテナーは OpenShift Container Platform および Podman コンテナーランタイムの事前設定を提供します。

Microsoft Windows および macOS では、CodeReady Containers ガイド付きのインストーラーにより、必要な Preset の入力が求められます。Linux では、OpenShift Container Platform の Preset がデフォルトで選択されます。**crc setup** コマンドを実行する前に、**crc config** コマンドを使用してこの選択を変更できます。選択したプリセットは、Microsoft Windows および macOS のシステムトレイから変更したり、サポートされるすべてのオペレーティングシステムのコマンドラインから変更したりできます。一度にアクティブにできる Preset は 1 つだけです。

関連情報

- 事前設定された各最小システム要件の詳細は、「 最小システム要件 」を参照してください。
- 選択した事前設定の変更に関する詳細は、「選択した事前設定 の変更」を 参照してください。

3.2. CODEREADY コンテナーの設定

crc setup コマンドは操作を実行し、CodeReady Containers インスタンスのホストマシンの環境を設定します。

crc setup コマンドは、**~/.crc** ディレクトリーが存在しない場合は作成します。

警告

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、インスタンスに加えられた変更をすべてキャプチャーします。

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で昇格できることを確認してください。

注記

crcの実行ファイルは、**root**ユーザーまたは管理者として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。

手順

1. （オプション）Linux では、デフォルトで OpenShift Container Platform の Preset が選択されます。Podman コンテナーランタイムの Preset を選択するには、以下を実行します。

```
$ crc config set preset podman
```

2. CodeReady コンテナーのホストマシンを設定します。

```
$ crc setup
```

関連情報

- 利用可能なコンテナーランタイムの Preset の詳細は、「Preset について」を参照して ください。

3.3. インスタンスを開始します

crc start コマンドは、CodeReady Containers インスタンスと構成済みコンテナーランタイムを開始します。

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、「Setting up CodeReady Containers」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ことを確認してください。
- OpenShift プリセットの場合は、有効な OpenShift ユーザープルシークレットがあることを確認してください。Red Hat Hybrid Cloud Console の 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、ダウンロードします。

注記

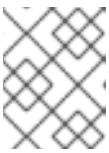
ユーザーのプルシークレットにアクセスするには、Red Hat アカウントが必要です。

手順

1. CodeReady Containers インスタンスを開始します。

```
$ crc start
```

2. OpenShift プリセットの場合は、プロンプトが表示されたら、ユーザーに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 を起動するのに最小 4 分の時間がかかります。

関連情報

- インスタンスに割り当てられたデフォルトのリソースを変更するには、「インスタンス の 設定」を参照し てください。
- **crc start** 中にエラーが表示される場合は、「 CodeReady Containers のトラブルシューティング 」セクションを参照してください。

3.4. OPENSHIFT クラスターへのアクセス

OpenShift Web コンソールまたは OpenShift CLI (**oc**) を使用して、CodeReady Containers インスタンスで実行している OpenShift クラスタにアクセスします。

3.4.1. OpenShift Web コンソールへのアクセス

Webブラウザを使って、OpenShift Webコンソールにアクセスします。

kubeadminまたは**developer**ユーザーのいずれかを使用してクラスタにアクセスします。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。**kubeadmin**ユーザーは、新しいユーザーの作成やロールの設定などの管理作業にのみ使用してください。

前提条件

- 実行中の CodeReady Containers インスタンス。詳細は、「 インスタンスの起動」を参照し てください。

手順

1. デフォルトのWebブラウザでOpenShiftのWebコンソールにアクセスするには、以下のコマンドを実行します。

```
$ crc console
```

2. **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。また、次のコマンドを実行すると、**developer**および**kubeadmin**ユーザーのパスワードを確認できます。

```
$ crc console --credentials
```

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。

3.4.2. OpenShift CLIによるOpenShiftクラスタへのアクセス

OpenShift CLI (**oc**)を使用して、OpenShiftクラスタにアクセスします。

前提条件

- 実行中の CodeReady Containers インスタンス。詳細は、「 インスタンスの起動」を参照し てください。

手順

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 Containersでは、デフォルトで Cluster Monitoring Operator を無効にしています。

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。

3.4.3. 内部 OpenShift レジストリーへのアクセス

CodeReady Containers インスタンスで実行している 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ています。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

- 実行中の CodeReady Containers インスタンス。詳細は、「インスタンスの起動」を参照してください。
- 動作するOpenShift CLI (**oc**)コマンドです。詳細は、「OpenShift CLI 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、**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 **kubeadmin** 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
```

3.5. ODOを使用したサンプルアプリケーションのデプロイ

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 インスタンスで実行されている 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 インスタンスが実行しています。詳細は、「 インスタンスの起動」を参照し てください。

手順

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**を使用した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。

3.6. インスタンスを停止しています

crc stop コマンドは、実行中の CodeReady Containers インスタンスとコンテナーランタイムを停止します。クラスターのシャットダウン中、停止プロセスには数分かかります。

手順

- CodeReady Containers インスタンスとコンテナーランタイムを停止します。

```
$ crc stop
```

3.7. インスタンスの削除

crc delete コマンドは、既存の CodeReady Containers インスタンスを削除します。

手順

- CodeReady Containers インスタンスを削除します。

```
$ crc delete
```

警告

crc delete コマンドを実行すると、CodeReady Containers インスタンス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、インスタンスに保存されている必要な情報を保存してください。

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 インスタンス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構成されている場合に情報を返しません。

4.3. 選択した事前設定の変更

事前設定された事前設定されたを選択して、CodeReady Containers インスタンスに使用するコンテナーランタイムを変更できます。

Microsoft Windows および macOS では、システムトレイまたはコマンドラインインターフェースを使用して、選択した Preset を変更できます。Linux の場合は、コマンドラインインターフェースを使用します。

重要

既存の CodeReady コンテナーインスタンスの事前設定を変更することはできません。事前設定の変更は、CodeReady コンテナーインスタンスの作成時にのみ適用されます。事前設定の変更を有効にするには、既存のインスタンスを削除して、新しいインスタンスを開始する必要があります。

手順

- コマンドラインから選択した Preset を変更します。

```
$ crc config preset <name>
```

有効なプリセット名は、OpenShift Container Platform の **openshift** と Podman コンテナーランタイムの **podman** です。

関連情報

- 事前設定された各最小システム要件の詳細は、「 最小システム要件 」を参照してください。

4.4. インスタンスの設定

cpus および **memory** プロパティーを使用して、CodeReady コンテナーインスタンス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 Containers インスタンスの構成を変更することはできません。構成の変更を有効にするには、実行中のインスタンスを停止して再起動する必要があります。

手順

- インスタンスで使用可能な vCPU の数を設定するには、以下を行います

```
$ 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 でインスタンスを起動するには、以下を実行します。

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

- インスタンスで使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。

- 必要な量のメモリーでインスタンスを起動するには、以下を実行します。

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

第5章 ネットワーキング

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** として 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 コンテナーのインスタンス内で実行される **dnsmasq** DNS コンテナーによって提供されます。

crc setup コマンドは、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 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、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、**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 Containers インスタンスの 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。

```
#!/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 setup** 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、CentOS では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。
- 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。

5.2. 予約された IP サブネット

CodeReady Containers OpenShift クラスターは内部で使用するための IP サブネットを確保しますが、ホストネットワークと共存するべきではありません。以下の IP サブネットが利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は、ホストオペレーティングシステムに応じて別の IP サブネットを確保します。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 には、追加のサブネットが予約されません。Linux 用の追加の予約サブネットは **192.168.130.0/24** です。

5.3. プロキシーの背後にある CODEREADY コンテナーの開始

環境変数や設定可能なプロパティを使って、定義されたプロキシの背後でCodeReady Containersを起動することができます。

注記

SOCKS プロキシーは OpenShift Container Platform ではサポートされません。

前提条件

- ホストマシンで既存の OpenShift CLI (**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、「OpenShift CLI 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>
```

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。

5.4. リモートサーバーでの CODEREADY コンテナーの設定

CodeReady Containers OpenShiftクラスターを実行するリモートサーバーを設定します。

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。この手順のすべてのコマンドをリモートサーバーで実行します。

警告

この手順は、ローカルネットワーク上でのみ実行してください。安全でないサーバーをインターネット上に公開することは、多くのセキュリティ上の問題があります。

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、「 Installing CodeReady Containers 」および「 Setting up CodeReady Containers 」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
```

この手順の間、クラスターが稼働していることを確認してください。

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 enable --now firewalld
$ sudo firewall-cmd --add-service=http --permanent
$ sudo firewall-cmd --add-service=https --permanent
$ sudo firewall-cmd --add-service=kube-apiserver --permanent
$ sudo firewall-cmd --reload
```

4. SELinux の場合、HAProxy が TCP ポート 6443 でリッスンして、このポートで **kube-apiserver** を提供できるようにします。

```
$ 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
    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

dnsmasq を使用して、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。

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。この手順のすべてのコマンドをクライアント上で実行します。

重要

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します。

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ます。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。
- 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 CLI (**oc**)が入っています。

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. 監視を開始します

CodeReady Containers が一般的なノートブックで実行できるように、デフォルトでクラスタ監視を無効にしています。モニタリングは、Red Hat Hybrid Cloud コンソール にクラスターを一覧表示する役割を担っています。以下の手順で、クラスタのモニタリングを有効にします。

前提条件

- CodeReady Containers インスタンスに追加のメモリーを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。詳しくは、「 インスタンスの設定」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. インスタンスを起動します。

```
$ crc start
```

警告

クラスターモニタリングを無効にできません。監視を削除するには、**enable-cluster-monitoring** 設定可能プロパティーを **false** に設定し、既存の CodeReady Containers インスタンスを削除します。

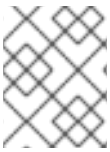
第7章 RED HAT CODEREADY CONTAINERS のトラブルシューティング

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

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。



注記

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。

前提条件

- クラスターへの OpenShift CLI (**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、「OpenShift CLI 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** コマンドを実行して、目的のノードを特定します。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

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。**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。

この追加の起動時間、または証明書の更新プロセスで失敗した場合は、以下の手順を使用します。

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 コマンドを実行すると、CodeReady Containers インスタンス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、インスタンスに保存されている必要な情報を保存してください。

3. 新しいリリースを設定します。

```
$ crc setup
```

4. 新しいインスタンスを開始します。

```
$ 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 インスタンスには、バンドル情報とインスタンスデータが含まれています。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 コマンドを実行すると、CodeReady Containers インスタンス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、インスタンスに保存されている必要な情報を保存してください。

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、インスタンスの停止、削除、**crc setup** コマンドによる変更の取り消し、変更の再適用、およびインスタンスの再起動が含まれます。

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、「 Setting up CodeReady Containers 」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、「 インスタンスの起動」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 Containers インスタンスを停止します。

```
$ crc stop
```

2. CodeReady Containers インスタンスを削除します。

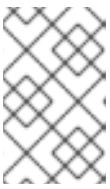
```
$ crc delete
```

警告

crc delete コマンドを実行すると、CodeReady Containers インスタンス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、インスタンスに保存されている必要な情報を保存してください。

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** コンテナーインスタンスを削除し、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 Containers インスタンスを開始します。

```
$ 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 を起動するのに最小 4 分の時間がかかります。

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。