



Red Hat CodeReady Containers 1.35

スタートガイド

CodeReady コンテナの使用および開発に関するクイックスタートガイド

Red Hat CodeReady Containers 1.35 スタートガイド

CodeReady コンテナ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

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	3
第1章 RED HAT CODEREADY CONTAINERS のご紹介	4
1.1. CODEREADY コンテナについて	4
1.2. 実稼働環境の OPENSIFT インストールとの相違点	4
第2章 インストール	5
2.1. 最小システム要件	5
2.1.1. ハードウェア要件	5
2.1.2. オペレーティングシステム要件	5
2.1.2.1. Microsoft Windows	5
2.1.2.2. macOS	5
2.1.2.3. Linux	5
2.2. LINUX に必要なソフトウェアパッケージ	6
2.3. CODEREADY コンテナのインストール	6
2.4. CODEREADY コンテナのアップグレード	7
第3章 CODEREADY コンテナの使用	8
3.1. CODEREADY コンテナの設定	8
3.2. 仮想マシンの起動	8
3.3. OPENSIFT クラスターへのアクセス	9
3.3.1. OpenShift Web コンソールへのアクセス	9
3.3.2. OpenShift CLIによるOpenShiftクラスタへのアクセス	10
3.3.3. 内部 OpenShift レジストリーへのアクセス	11
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	12
3.5. 仮想マシンの停止	13
3.6. 仮想マシンの削除	14
第4章 CODEREADY コンテナの設定	15
4.1. CODEREADY コンテナ設定について	15
4.2. CODEREADY コンテナ設定の表示	15
4.3. 仮想マシンの設定	15
第5章 ネットワーキング	17
5.1. DNS 設定の詳細	17
5.1.1. 一般的な DNS 設定	17
5.1.2. Linux	17
5.1.2.1. NetworkManager + systemd-resolved	17
5.1.2.2. NetworkManager + dnsmasq	17
5.2. 予約された IP サブネット	18
5.3. プロキシの背後にある CODEREADY コンテナの開始	18
5.4. リモートサーバーでの CODEREADY コンテナの設定	19
5.5. リモート CODEREADY コンテナインスタンスへの接続	20
第6章 管理タスク	22
6.1. MONITORING、ALERTING、および TELEMETRY の起動	22
第7章 RED HAT CODEREADY CONTAINERS のトラブルシューティング	23
7.1. OPENSIFT クラスターへのシェルアクセスの取得	23
7.2. 期限切れの証明書のトラブルシューティング	23
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	24
7.4. 不明な問題のトラブルシューティング	25

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 CONTAINERS のご紹介

1.1. CODEREADY コンテナについて

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 **crcc** 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 OPENSIFT 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。**
- **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。
- 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 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他の違いが生じる可能性があります。

CodeReady コンテナには、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。

- 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

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。



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、[Configuring the virtual machine](#)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 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	<code>sudo dnf install NetworkManager</code>
Red Hat Enterprise Linux/CentOS	<code>su -c 'yum install NetworkManager'</code>
Debian/Ubuntu	<code>sudo apt install qemu-kvm libvirt-daemon libvirt-daemon-system network-manager</code>

2.3. CODEREADY コンテナのインストール

CodeReady コンテナは、Red Hat Enterprise Linux および Microsoft Windows のポータブル実行ファイルとして利用できます。macOS では、ガイド付きインストーラーを使用して CodeReady コンテナを利用できます。

要件

- 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 コンテナを実行することはできません。

Red Hat Enterprise Linux の場合は、**~/Downloads** 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. Create the `~/bin` 存在していない場合はディレクトリーを展開し、`crc 実行ファイル` をこれにコピーします。

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

- c. Add the `~/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 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 コンテナの使用

3.1. CODEREADY コンテナの設定

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。

この手順では、`~/crc` ディレクトリが存在しない場合は、これを使用します。



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、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。

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

- CodeReady コンテナのホストマシンを設定します。

```
$ crc setup
```

テレメトリデータ収集の継続

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。

要件

- 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ことを確認してください。
- 有効な OpenShift ユーザープルシークレットがある。[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 コンテナ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および Operator を起動するのに最小 4 分の時間がかかります。

追加リソース

- 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。
- **crc start** 時にエラーが表示される場合は、[CodeReady コンテナ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。

3.3. OPENSIFT 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 仮想マシンで実行している 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

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ーにアクセスします。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

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。

要件

- 稼働中の CodeReady コンテナの仮想マシン。詳細は、[仮想マシンの起動](#) を参照してください。

手順

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 コンテナのトラブルシューティング](#) を参照してください。

追加リソース

- [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. 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。

要件

- 稼働中の CodeReady コンテナの仮想マシン。詳細は、[仮想マシンの起動](#) を参照してください。

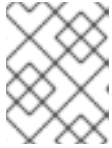
手順

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

oc start コマンドは、**developer** ユーザーのパスワードを出力します。**oc 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 コンテナのトラブルシューティング](#) を参照してください。

追加リソース

- [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イメージレジストリーが含まれます。この内部コンテナイメージレジストリーは、ローカル開発コンテナイメージの公開ターゲットとして使用できます。

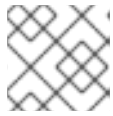
要件

- 稼働中の CodeReady コンテナの仮想マシン。詳細は、[仮想マシンの起動](#) を参照してください。
- 動作する OpenShift CLI (**oc**) コマンドです。詳細は、[Accessing the OpenShift cluster with the OpenShift CLI](#) 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
```

3.4. 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 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。

要件

- **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 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

1. **developer** ユーザーとして、実行中の CodeReady Container OpenShift クラスタにログインします。

```
$ 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、ローカル設定ファイルにエントリを追加します。

```
$ 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。

追加リソース

- **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 コンテナの仮想マシンを削除します。

手順

- CodeReady コンテナ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 コンテナの設定

4.1. CODEREADY コンテナ設定について

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 コンテナ設定の表示

CodeReady コンテナの実行ファイルは、設定可能なプロパティと現在の 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. 仮想マシンの設定

cpus および **memory** プロパティを使用して、CodeReady コンテナ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>
```

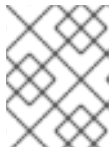
cpus プロパティのデフォルト値は **4** です。割り当てる vCPU の数は、デフォルト以上である必要があります。

- 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 コンテナの仮想マシン内で実行される **dnsmasq** DNS コンテナ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 コンテナの仮想マシンの IP です。
- **systemd-resolved** 設定は、**/etc/NetworkManager/dispatcher.d/99-crc.sh**:

```
#!/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。
- この `dnsmasq` インスタンスの設定ファイルは、`/etc/NetworkManager/dnsmasq.d/crc.conf`:

```
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。

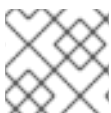
予約された IP サブネット

- **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. プロキシの背後にある CODEREADY コンテナの開始

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

要件

- 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ついての詳細は、[Accessing the OpenShift cluster with the OpenShift CLI](#) 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 コンテナの設定に設定されたプロキシ関連の値は、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。

要件

- CodeReady コンテナが、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 コンテナのインストール](#)および[CodeReady コンテナの設定](#)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 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
```

- 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
```

- 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}
```

- 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
```

- 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。

要件

- 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 コンテナの DNS エントリを **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

の既存のエントリをすべてコメントアウトします。/etc/NetworkManager/dnsmasq.d/crc.conf これらのエントリは、CodeReady コンテナ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 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。Telemetry 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。

要件

- 追加のメモリーを CodeReady コンテナ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** 設定可能なプロパティを **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. OPENSIFT 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。

要件

- 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。詳細な手順については、[Accessing the OpenShift cluster with the OpenShift CLI](#) 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 コマンドは、CodeReady コンテナ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。

要件

- **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 コンテナの仮想マシンを停止します。

```
$ crc stop
```

2. CodeReady コンテナ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. 既存の問題アドレスが見つからない場合は、問題 [を作成し、添付](#) します。 [~/crc/crc.log 作成された問題への](#) ファイル `~/crc/crc.log` ファイルには、異常の診断に役立つ詳細なデバッグとトラブルシューティングの情報が含まれています。