Red Hat CodeReady Containers 1.14

スタートガイド

CodeReady コンテナーの使用および開発のガイド

Last Updated: 2021-03-18

Red Hat CodeReady Containers 1.14 スタートガイド

CodeReady コンテナーの使用および開発のガイド

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 コンテナーの使用をスピードアップする方法を説明します。含まれる手順およびサンプルは、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アプリケーションの開発の手順を説明します。

目次

第1章 RED HAT CODEREADY CONTAINERS の概要

1.1. CODEREADY CONTAINERS

Red Hat CodeReady Containers は、最小限の OpenShift 4 クラスターをローカルコンピューターに提供します。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 Containers は主に開発者のデスクトップ上での実行を対象としています。ヘッドレスやマルチ開発者用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 の完全な概要については、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 Containers 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. 実稼働用の OPENSHIFT インストールとの違い

Red Hat CodeReady Containers は、以下の主な相違点を備えた通常の OpenShift インストールです。

- **CodeReady Containers OpenShift クラスターは一時的なもので、実稼働環境での使用を目的としていません。**
- これは、マスターノード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 **machine-config** および **monitoring** Operator を無効にします。
  - これらの Operator が無効になると、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、特に外部のネットワークでその他の違いが生じる可能性があります。

CodeReady Containers には、以下のカスタマイズ不可のクラスター設定も含まれます。以下の設定は変更しないでください。

- * **.crc.testing** ドメインの使用。
- 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、プロキシーが同じアドレス空間で実行される場合など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

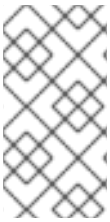
2.1. システムの最小要件

CodeReady Containers には、以下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があります。

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)
- 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 Containers 仮想マシンにより多くのリソースを割り当てる場合は、「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 Containers には、サポートされるオペレーティングシステムの以下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 Pro 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または編集では機能しません。Microsoft Windows 10 Home Edition はサポートされていません。

2.1.2.2. macOS

- macOS では、CodeReady Containers には macOS 10.12 Sierra 以降が必要です。CodeReady Containers は、以前のバージョンの macOS 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 を使用する場合は、CodeReady Containers を実行するマシンを 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設定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 **libvirt** パッケージおよび **NetworkManager** パッケージを実行する必要があります。以下の表を参照して、Linux ディストリビューションにこれらのパッケージをインストールするのに使用するコマンドを確認してください。

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 CONTAINERS のインストール

前提条件

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

手順

1. プラットフォームの 最新リリースの CodeReady Containers をダウンロードし、アーカイブの内容を **PATH** の場所に展開します。

2.4. CODEREADY CONTAINERS のアップグレード

新しいバージョンの CodeReady Containers バイナリーでは、以前のバージョンと非互換性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 コマンドを実行すると、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** バイナリーは、最新リリースのバイナリーに置き換えます。そのバージョンを確認して、新しい **crc** バイナリーが使用されてい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 仮想マシンを起動します。

```
$ crc start
```

第3章 CODEREADY CONTAINERS の使用

3.1. CODEREADY CONTAINERS 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** ディレクトリーが存在しない場合は作成します。

前提条件

- ユーザーアカウントには、**sudo** コマンドを使用するパーミッションがあります。

注記

- **crc** バイナリーを **root**（または Administrator）として実行しないでください。ユーザーアカウントで **crc** バイナリーを常に実行します。
- 新規バージョンを設定する場合は、新しい CodeReady Containers リリースを設定する前に、仮想マシンに加えられた変更を取得します。

手順

1. CodeReady Containers のホストマシンを設定します。

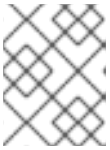
```
$ crc setup
```

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** コマンドを使用してホストマシンを設定します。詳細は、「CodeReady Containers の設定」を 参照してください。
- 有効な OpenShift ユーザープルシークレットがある。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

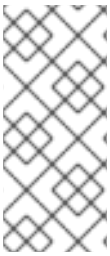
ユーザープルシークレットにアクセスするには、Red Hat アカウントが必要です。

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

- クラスターは要求に対応する前に、必要なコンテナーおよび Operator を起動するのに最低 4 分かかります。
- **crc の起動** 中にエラーが表示される場合は、Troubleshooting CodeReady Containers のセクションにある可能性のある解決策を確認してください。

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バイナリー(**oc**)を使用して CodeReady Container 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

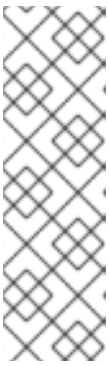
前提条件

- 実行中の CodeReady Containers 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** コマンドの出力で出力されるパスワードを使用して、**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表示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、クラスターに最初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し、アプリケーションのデプロイメントに **開発者** ユーザーを使用します。新規ユーザーの作成、ロールの設定などの管理タスクには **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 Containers のトラブルシューティング 」を参照してください。

その他のリソース

- OpenShift ドキュメント では、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

- 実行中の CodeReady Containers 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** バイナリーを **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. print 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。

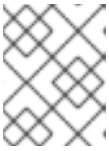
```
$ oc login -u developer https://api.crc.testing:6443
```

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 が利用でき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 はデフォルトで **machine-config** および **monitoring** 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、「CodeReady Containers のトラブルシューティング 」を参照してください。

その他のリソース

- OpenShift ドキュメント では、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. 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 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「 odoのインストール 」を参照してください。**
- CodeReady Containers 仮想マシンが実行している。詳細は、「 仮想マシンの起動」を 参照してください。

手順

odo を使用してアプリケーションのサンプルをデプロイするには、以下の手順に従います。

1. 実行中の CodeReady Containers OpenShift クラスターに **developer** ユーザーとしてログインします。

```
$ 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 上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 --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「 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できる 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。

その他のリソース

- odo の使用についての詳細は、**odo** の **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。停止プロセスには、クラスターがシャットダウンする間数分かかります。

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 の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 コマンドを実行すると、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 CONTAINERS の設定

4.1. CODEREADY CONTAINERS 設定

crc config コマンドを使用して、**crc** バイナリーと CodeReady Containers 仮想マシンの両方を設定します。**crc config** コマンドには、設定で動作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 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、スタートアップチェックでエラーが報告され、条件が満たされていない場合は実行を停止します。**skip-check または warn-check で始まるプロパティーの値を true に設定し** て、チェックを省略するか、エラーではなく警告を報告します。

4.2. CODEREADY CONTAINERS 設定の表示

CodeReady Containers バイナリーは、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る場合は、**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仮想 CPU のデフォルト数とメモリー量を設定します。

重要

既存の CodeReady Containers 仮想マシンの設定は変更できません。設定変更を有効にするには、既存の仮想マシンを削除して、新規仮想マシンを作成する必要があります。

新規仮想マシンを作成するには、最初に既存の CodeReady Containers 仮想マシンを削除してから、設定変更のある新規仮想マシンを起動します。

```
$ crc delete
$ crc start
```

警告

crc delete コマンドを実行すると、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。

手順

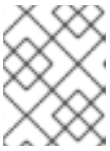
- 仮想マシンで利用可能な vCPU の数を増やすには、以下を実行します。

  ```
  $ 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 数は、デフォルト以上である必要があります。

- 仮想マシンで利用可能なメモリーを増やすには、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 ギビバイト(GiB)は 1024 MiB に等しくなります。

  メモリー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サイズは、デフォルト以上である必要があります。

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 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 OpenShift 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 アプリケーションへのアクセスです。

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する際に 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 コンテナーにより提供されます。

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 DNS 設定を調整します。 **crc の起動** 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チェックが行われます。

5.1.2. linux

Linux では、CodeReady Containers には以下の DNS 設定が必要です。

- CodeReady Containers 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 **dnsmasq** インスタンスは、crc.testing ドメインおよび **apps-crc.testing** ドメインの要求を 192.168.130.11 **DNS** サーバーに転送します。

5.1.3. macOS

macOS では、CodeReady Containers には以下の DNS 設定が必要です。

- CodeReady Containers は、**/etc/resolver/testing** ファイルを作成し、**testing** ドメインに対するすべての DNS 要求を CodeReady Containers 仮想マシンに転送するよう macOS に指示します。
- CodeReady Containers は、**api.crc.testing** エントリーを、仮想マシンの IP アドレスを参照する **/etc/hosts** に追加します。**oc** バイナリーには、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. プロキシーの背後での CODEREADY CONTAINERS の起動

前提条件

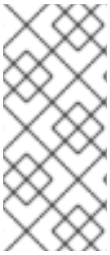
- ホストマシンで既存の **oc** バイナリーを使用するには、**.testing** ドメインを **no_proxy** 環境変数の一部としてエクスポートします。
- 埋め込み **oc** バイナリーには手動の設定は必要ありません。組み込まれた **oc** バイナリーの使用についての詳細は、「 Accessing the OpenShift cluster with **oc** 」を参照してください。

手順

- CodeReady Containers をプロキシーの背後で起動するには、以下を実行します。
  1. **http_proxy および https_proxy 環境変数を使用してプロキシー** を定義するか、**crc config set** コマンドを以下のように使用します。

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

crc バイナリーは、環境変数または CodeReady Containers 設定で設定されたプロキシーを一度使用できるようになります。



注記

- CodeReady Containers の設定で設定された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 ではサポートされていません。

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 Containers を通常のノートパソコンで実行するには、一部のリソース重みサービスがデフォルトで無効にされます。これらのいずれかが Prometheus と、関連するモニタリング、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

- 実行中の CodeReady Containers 仮想マシンおよび作業用の **oc** コマンド。詳細は、「Accessing the OpenShift cluster with **oc** 」を参照してください。
- **oc** で **kubeadmin** ユーザーとしてログインする必要があります。
- 追加のメモリーを CodeReady Containers 仮想マシンに割り当てる必要があります。12 GiB のメモリー（値 **12288**）を推奨します。詳細は、「仮想マシンの設定」を 参照してください。

手順

1. 管理外の Operator を一覧表示し、**cluster-monitoring-operator** の数値インデックスを書き留めます。
   - Linux または macOS の場合：

     ```
     $ oc get clusterversion version -ojsonpath='{range .spec.overrides[*]}{.name}{"\n"}{end}' |
     nl -v 0
     ```

   - PowerShell を使用する Microsoft Windows では、以下を行います。

     ```
     PS> oc get clusterversion version -ojsonpath='{range .spec.overrides[*]}{.name}{"\n"}
     {end}' | % {$nl++;"`t$($nl-1) `t $_"};$nl=0
     ```

2. **cluster-monitoring-operator の識別された数値インデックスを使用して、モニタリング、アラート、および Telemetry サービスを開始します。**

   ```
   $ oc patch clusterversion/version --type='json' -p '[{"op":"remove",
   "path":"/spec/overrides/<unmanaged-operator-index>"}]' -oyaml
   ```

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です。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中に発生した問題は、CodeReady Containers の範囲外です。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 で問題を追跡します。

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ありません。これは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、「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たノードの名前です。

7.2. 期限切れ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、証明書の更新プロセスは意図されたとおりに機能しません。以下の手順に従って、証明書の有効期限による潜在的なエラーが発生しないようにします。

リリースされる各 **crc** バイナリーのシステムバンドルは、リリース後 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によるものです。その結果、古い **crc** バイナリーまたはシステムバンドルを使用すると、証明書の有効期限が切れる可能性があります。

CodeReady Containers 1.2.0 以降では、組み込み証明書は **crc** によって自動的に更新できます。**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。証明書の更新は、クラスターの起動時間に最大 5 分追加できます。

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** バイナリーを **$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 コマンドを実行すると、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に関する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。新しい CodeReady Containers リリースの設定時に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。以前のインスタンスデータのカスタマイズにより、この情報は更新されません。これにより、**crc start** コマンドを実行すると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. インスタンスの起動を試みる前に、**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 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

CodeReady Containers をクリーンな状態で再起動して、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、そのコマンドの削除、**crc setup** コマンドによる変更の取り消し、それらの変更の再適用、仮想マシン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、「CodeReady Containers の設定」を 参照してください。
- **crc start コマンドを使用して CodeReady Containers 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、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、「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 Containers 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 コマンドを実行すると、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します。

4. ホストマシンを設定して、変更を適用します。

```
$ crc setup
```

5. CodeReady Containers 仮想マシンを起動します。

```
$ crc start
```

注記

クラスターは要求に対応する前に、必要なコンテナーおよび Operator を起動するのに最低 4 分かかります。

この手順で問題が解決されない場合は、以下の手順を実施します。

1. 発生した問題について、オープンな問題を検索 します。
2. 既存の問題が解決されない場合は、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情報が記載されており、発生している問題を診断するのに役立ちます。