Red Hat CodeReady Containers 1.40

Getting Started Guide

Quick-start guide to using and developing with CodeReady Containers

Last Updated: 2022-02-28

Red Hat CodeReady Containers 1.40 Getting Started Guide

Quick-start guide to using and developing with CodeReady Containers

Kevin Owen
kowen@redhat.com

Legal Notice

Abstract

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

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. INTRODUCING RED HAT CODEREADY CONTAINERS

1.1. ABOUT CODEREADY CONTAINERS

Red Hat CodeReady Containers brings a minimal OpenShift 4 cluster to your local computer. This cluster provides a minimal environment for development and testing purposes. CodeReady Containers is mainly targeted at running on developers' desktops. For other use cases, such as headless or multi-developer setups, use the full OpenShift installer.

See the OpenShift documentation for a full introduction to OpenShift.

CodeReady Containers includes the **crc** command-line interface (CLI) to interact with the CodeReady Containers instance running the OpenShift cluster.

1.2. DIFFERENCES FROM A PRODUCTION OPENSHIFT INSTALLATION

Red Hat CodeReady Containers provides a regular OpenShift installation with the following notable differences:

- **The CodeReady Containers OpenShift cluster is ephemeral and is not intended for production use.**
- **CodeReady Containers does not have a supported upgrade path to newer OpenShift versions.** Upgrading the OpenShift version may cause issues that are difficult to reproduce.
- It uses a single node which behaves as both a control plane and worker node.
- It disables the Cluster Monitoring Operator by default. This disabled Operator causes the corresponding part of the web console to be non-functional.
- The OpenShift cluster runs in a virtual machine known as an *instance*. This may cause other differences, particularly with external networking.

The OpenShift cluster provided by CodeReady Containers also includes the following non-customizable cluster settings. These settings should not be modified:

- Use of the ***.crc.testing** domain.
- The address range used for internal cluster communication.
    - The cluster uses the **172** address range. This can cause issues when, for example, a proxy is run in the same address space.

CHAPTER 2. INSTALLATION

2.1. MINIMUM SYSTEM REQUIREMENTS

CodeReady Containers has the following minimum hardware and operating system requirements.

2.1.1. Hardware requirements

CodeReady Containers requires the following system resources:

- 4 physical CPU cores
- 9 GB of free memory
- 35 GB of storage space

CodeReady Containers is supported only on AMD64 and Intel 64 processor architectures. CodeReady Containers does not support the ARM-based M1 architecture. CodeReady Containers does not support nested virtualization.

NOTE

The OpenShift cluster requires these minimum resources to run in the CodeReady Containers instance. Some workloads may require more resources. To assign more resources to the CodeReady Containers instance, see Configuring the instance.

2.1.2. Operating system requirements

CodeReady Containers requires the following minimum version of a supported operating system:

2.1.2.1. Microsoft Windows

- On Microsoft Windows, CodeReady Containers requires the Windows 10 Fall Creators Update (version 1709) or later. CodeReady Containers does not work on earlier versions of Microsoft Windows. Microsoft Windows 10 Home Edition is not supported.

2.1.2.2. macOS

- On macOS, CodeReady Containers requires macOS 10.14 Mojave or later. CodeReady Containers does not work on earlier versions of macOS.

2.1.2.3. Linux

- On Linux, CodeReady Containers is supported only on Red Hat Enterprise Linux/CentOS 7.5 or later (including 8.x versions) and on the latest two stable Fedora releases.
- When using Red Hat Enterprise Linux, the machine running CodeReady Containers must be registered with the Red Hat Customer Portal .
- Ubuntu 18.04 LTS or later and Debian 10 or later are not supported and may require manual set up of the host machine.
- See Required software packages to install the required packages for your Linux distribution.

2.2. REQUIRED SOFTWARE PACKAGES FOR LINUX

CodeReady Containers requires the **libvirt** and **NetworkManager** packages to run on Linux. Consult the following table to find the command used to install these packages for your Linux distribution:

Table 2.1. Package installation commands by distribution

Linux Distribution	Installation command
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. INSTALLING CODEREADY CONTAINERS

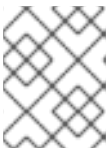
CodeReady Containers is available as a portable executable for Red Hat Enterprise Linux and Microsoft Windows. On macOS, CodeReady Containers is available using a guided installer.

Prerequisites

- Your host machine must meet the minimum system requirements. For more information, see Minimum system requirements.

Procedure

1. Download the latest release of CodeReady Containers for your platform.
2. On Microsoft Windows, extract the contents of the archive.
3. On macOS or Microsoft Windows, run the guided installer and follow the instructions.



NOTE

On Microsoft Windows, you must install CodeReady Containers to your local ***C:***\ drive. You cannot run CodeReady Containers from a network drive.

On Red Hat Enterprise Linux, assuming the archive is in the ***~/Downloads*** directory, follow these steps:

a. Extract the contents of the archive:

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

b. Create the ***~/bin*** directory if it does not exist and copy the **crc** executable to it:

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

c. Add the *~/bin* directory to your **$PATH**:

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

2.4. ABOUT USAGE DATA COLLECTION

CodeReady Containers prompts you before use for optional, anonymous usage data collection to assist with development. No personally identifiable information is collected. Consent for usage data collection can be granted or revoked by you at any time.

Additional resources

- For more information about collected data, see the Red Hat Telemetry data collection notice.
- To grant or revoke consent for usage data collection, see Configuring usage data collection.

2.5. CONFIGURING USAGE DATA COLLECTION

Consent for usage data collection can be granted or revoked by you at any time using the following configuration commands.

NOTE

Changes to telemetry consent do not modify a running instance. The change will take effect next time you run the **crc start** command.

Procedure

- To manually enable telemetry, run the following command:

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

- To manually disable telemetry, run the following command:

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

Additional resources

- For more information about the collected data, see the Red Hat Telemetry data collection notice.

2.6. UPGRADING CODEREADY CONTAINERS

Newer versions of the CodeReady Containers executable require manual set up to prevent potential incompatibilities with earlier versions.

Procedure

1. Download the latest release of CodeReady Containers.
2. Delete the existing CodeReady Containers instance:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers instance. Save any desired information stored in the instance before running this command.

3. Replace the earlier **crc** executable with the executable of the latest release. Verify that the new **crc** executable is in use by checking its version:

```
$ crc version
```

4. Set up the new CodeReady Containers release:

```
$ crc setup
```

5. Start the new CodeReady Containers instance:

```
$ crc start
```

CHAPTER 3. USING CODEREADY CONTAINERS

3.1. SETTING UP CODEREADY CONTAINERS

The **crc setup** command performs operations to set up the environment of your host machine for the CodeReady Containers instance.

This procedure creates the *~/.crc* directory if it does not already exist.

WARNING

If you are setting up a new version, capture any changes made to the instance before setting up a new CodeReady Containers release.

Prerequisites

- On Linux or macOS, ensure that your user account has permission to use the **sudo** command. On Microsoft Windows, ensure that your user account can elevate to Administrator privileges.

NOTE

Do not run the **crc** executable as the **root** user or an administrator. Always run the **crc** executable with your user account.

Procedure

1. Set up your host machine for CodeReady Containers:

```
$ crc setup
```

3.2. STARTING THE INSTANCE

The **crc start** command starts the CodeReady Containers instance and configured container runtime.

Prerequisites

- To avoid networking-related issues, ensure that you are not connected to a VPN and that your network connection is reliable.
- You set up the host machine using the **crc setup** command. For more information, see Setting up CodeReady Containers.
- On Microsoft Windows, ensure that your user account can elevate to Administrator privileges.
- For the OpenShift preset, ensure that you have a valid OpenShift user pull secret. Copy or download the pull secret from the Pull Secret section of the CodeReady Containers page on the Red Hat Hybrid Cloud Console.

NOTE

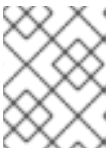
Accessing the user pull secret requires a Red Hat account.

Procedure

1. Start the CodeReady Containers instance:

```
$ crc start
```

2. For the OpenShift preset, supply your user pull secret when prompted.



NOTE

The cluster takes a minimum of four minutes to start the necessary containers and Operators before serving a request.

Additional resources

- To change the default resources allocated to the instance, see Configuring the instance.
- If you see errors during **crc start**, see the Troubleshooting CodeReady Containers section for potential solutions.

3.3. ACCESSING THE OPENSHIFT CLUSTER

Access the OpenShift cluster running in the CodeReady Containers instance by using the OpenShift web console or OpenShift CLI (**oc**).

3.3.1. Accessing the OpenShift web console

Access the OpenShift web console by using your web browser.

Access the cluster by using either the **kubeadmin** or **developer** user. Use the **developer** user for creating projects or OpenShift applications and for application deployment. Use the **kubeadmin** user only for administrative tasks such as creating new users or setting roles.

Prerequisites

- A running CodeReady Containers instance. For more information, see Starting the instance.

Procedure

1. To access the OpenShift web console with your default web browser, run the following command:

```
$ crc console
```

2. Log in as the **developer** user with the password printed in the output of the **crc start** command. You can also view the password for the **developer** and **kubeadmin** users by running the following command:

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

See Troubleshooting CodeReady Containers if you cannot access the CodeReady Containers OpenShift cluster.

Additional resources

- The OpenShift documentation covers the creation of projects and applications.

3.3.2. Accessing the OpenShift cluster with the OpenShift CLI

Access the OpenShift cluster by using the OpenShift CLI (**oc**).

Prerequisites

- A running CodeReady Containers instance. For more information, see Starting the instance.

Procedure

1. Run the **crc oc-env** command to print the command needed to add the cached **oc** executable to your **$PATH**:

```
$ crc oc-env
```

2. Run the printed command.

3. Log in as the **developer** user:

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

NOTE

The **crc start** command prints the password for the **developer** user. You can also view it by running the **crc console --credentials** command.

4. You can now use **oc** to interact with your OpenShift cluster. For example, to verify that the OpenShift cluster Operators are available, log in as the **kubeadmin** user and run the following command:

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

NOTE

CodeReady Containers disables the Cluster Monitoring Operator by default.

See Troubleshooting CodeReady Containers if you cannot access the CodeReady Containers OpenShift cluster.

Additional resources

- The OpenShift documentation covers the creation of projects and applications.

3.3.3. Accessing the internal OpenShift registry

The OpenShift cluster running in the CodeReady Containers instance includes an internal container image registry by default. This internal container image registry can be used as a publication target for locally developed container images.

Prerequisites

- A running CodeReady Containers instance. For more information, see Starting the instance.
- A working OpenShift CLI (**oc**) command. For more information, see Accessing the OpenShift cluster with the OpenShift CLI.
- An installation of **podman** or **docker**.
  - For Docker, add **default-route-openshift-image-registry.apps-crc.testing** as an insecure registry. For more information, see the Docker documentation.

Procedure

1. Check which user is logged in to the cluster:

```
$ oc whoami
```

NOTE

For demonstration purposes, the current user is assumed to be **kubeadmin**.

2. Log in to the registry as that user with its token:

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

3. Create a new project:

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

4. Pull an example container image:

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

5. Tag the image, including namespace details:

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

6. Push the container image to the internal registry:

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

7. Get imagestreams and verify that the pushed image is listed:

```
$ oc get is
```

8. Enable image lookup in the imagestream:

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

This setting allows the imagestream to be the source of images without having to provide the full URL to the internal registry.

9. Create a pod using the recently pushed image:

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

3.4. DEPLOYING A SAMPLE APPLICATION WITH ODO

You can use **odo** to create OpenShift projects and applications from the command line. This procedure deploys a sample application to the OpenShift cluster running in the CodeReady Containers instance.

Prerequisites

- You have installed **odo**. For more information, see Installing **odo** in the **odo** documentation.
- The CodeReady Containers instance is running. For more information, see Starting the instance.

Procedure

1. Log in to the running CodeReady Containers OpenShift cluster as the **developer** user:

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

2. Create a project for your application:

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

3. Create a directory for your components:

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

4. Create a component from a sample application on GitHub:

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

NOTE

Creating a component from a remote Git repository will rebuild the application each time you run the **odo push** command. To create a component from a local Git repository, see Creating a single-component application with **odo** in the **odo** documentation.

5. Create a URL and add an entry to the local configuration file:

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

6. Push the changes:

```
$ odo push
```

Your component is now deployed to the cluster with an accessible URL.

7. List the URLs and check the desired URL for the component:

```
$ odo url list
```

8. View the deployed application using the generated URL.

Additional resources

- For more information about using **odo**, see the **odo** documentation.

3.5. STOPPING THE INSTANCE

The **crc stop** command stops the running CodeReady Containers instance and container runtime. The stopping process takes a few minutes while the cluster shuts down.

Procedure

- Stop the CodeReady Containers instance and container runtime:

```
$ crc stop
```

3.6. DELETING THE INSTANCE

The **crc delete** command deletes an existing CodeReady Containers instance.

Procedure

- Delete the CodeReady Containers instance:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers instance. Save any desired information stored in the instance before running this command.

CHAPTER 4. CONFIGURING CODEREADY CONTAINERS

4.1. ABOUT CODEREADY CONTAINERS CONFIGURATION

Use the **crc config** command to configure both the **crc** executable and the CodeReady Containers instance. The **crc config** command requires a subcommand to act on the configuration. The available subcommands are **get**, **set, unset**, and **view**. The **get**, **set**, and **unset** subcommands operate on named configurable properties. Run the **crc config --help** command to list the available properties.

You can also use the **crc config** command to configure the behavior of the startup checks for the **crc start** and **crc setup** commands. By default, startup checks report an error and stop execution when their conditions are not met. Set the value of a property starting with **skip-check** to **true** to skip the check.

4.2. VIEWING CODEREADY CONTAINERS CONFIGURATION

The CodeReady Containers executable provides commands to view configurable properties and the current CodeReady Containers configuration.

Procedure

- To view the available configurable properties:

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

- To view the values for a configurable property:

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

- To view the complete current configuration:

```
$ crc config view
```

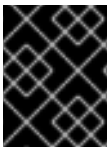
NOTE

The **crc config view** command does not return any information if the configuration consists of default values.

4.3. CONFIGURING THE INSTANCE

Use the **cpus** and **memory** properties to configure the default number of vCPUs and amount of memory available to the CodeReady Containers instance, respectively.

Alternatively, the number of vCPUs and amount of memory can be assigned using the **--cpus** and **--memory** flags to the **crc start** command, respectively.



IMPORTANT

You cannot change the configuration of a running CodeReady Containers instance. To enable configuration changes, you must stop the running instance and start it again.

Procedure

- To configure the number of vCPUs available to the instance:

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

The default value for the **cpus** property is **4**. The number of vCPUs to assign must be greater than or equal to the default.

- To start the instance with the desired number of vCPUs:

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

- To configure the memory available to the instance:

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

NOTE

Values for available memory are set in mebibytes (MiB). One gibibyte (GiB) of memory is equal to 1024 MiB.

The default value for the **memory** property is **9216**. The amount of memory to assign must be greater than or equal to the default.

- To start the instance with the desired amount of memory:

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

CHAPTER 5. NETWORKING

5.1. DNS CONFIGURATION DETAILS

5.1.1. General DNS setup

The OpenShift cluster managed by CodeReady Containers uses 2 DNS domain names, **crc.testing** and **apps-crc.testing**. The **crc.testing** domain is for core OpenShift services. The **apps-crc.testing** domain is for accessing OpenShift applications deployed on the cluster.

For example, the OpenShift API server is exposed as **api.crc.testing** while the OpenShift console is accessed as **console-openshift-console.apps-crc.testing**. These DNS domains are served by a **dnsmasq** DNS container running inside the CodeReady Containers instance.

The **crc setup** command detects and adjusts your system DNS configuration so that it can resolve these domains. Additional checks are done to verify DNS is properly configured when running **crc start**.

5.1.2. Linux

On Linux, depending on your distribution, CodeReady Containers expects the following DNS configuration:

5.1.2.1. NetworkManager + systemd-resolved

This configuration is used by default on Fedora 33 or newer, and on Ubuntu Desktop editions.

- CodeReady Containers expects NetworkManager to manage networking.
- CodeReady Containers configures **systemd-resolved** to forward requests for the **testing** domain to the **192.168.130.11** DNS server. **192.168.130.11** is the IP of the CodeReady Containers instance.
- **systemd-resolved** configuration is done with a NetworkManager dispatcher script in ***/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
```

NOTE

systemd-resolved is also available as an unsupported Technology Preview on Red Hat Enterprise Linux and CentOS 8.3. After configuring the host to use **systemd-resolved**, stop any running clusters and rerun **crc setup**.

5.1.2.2. NetworkManager + dnsmasq

This configuration is used by default on Fedora 32 or older, on Red Hat Enterprise Linux, and on CentOS.

- CodeReady Containers expects NetworkManager to manage networking.
- NetworkManager uses **dnsmasq** with the ***/etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf*** configuration file.
- The configuration file for this **dnsmasq** instance is ***/etc/NetworkManager/dnsmasq.d/crc.conf***:

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

  - The NetworkManager **dnsmasq** instance forwards requests for the **crc.testing** and **apps-crc.testing** domains to the **192.168.130.11** DNS server.

5.2. RESERVED IP SUBNETS

The CodeReady Containers OpenShift cluster reserves IP subnets for internal use which should not collide with your host network. Ensure that the following IP subnets are available for use:

Reserved IP subnets

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

Additionally, the host hypervisor may reserve another IP subnet depending on the host operating system. On Microsoft Windows, the hypervisor reserves a randomly generated IP subnet that cannot be determined ahead-of-time. No additional subnet is reserved on macOS. The additional reserved subnet for Linux is **192.168.130.0/24**.

5.3. STARTING CODEREADY CONTAINERS BEHIND A PROXY

You can start CodeReady Containers behind a defined proxy using environment variables or configurable properties.

NOTE

SOCKS proxies are not supported by OpenShift Container Platform.

Prerequisites

- To use an existing OpenShift CLI (**oc**) executable on your host machine, export the **.testing** domain as part of the **no_proxy** environment variable. The embedded **oc** executable does not require manual settings. For more information about using the embedded **oc** executable, see Accessing the OpenShift cluster with the OpenShift CLI.

Procedure

1. Define a proxy using the **http_proxy** and **https_proxy** environment variables or using the **crc config set** command as follows:

```
$ 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. If the proxy uses a custom CA certificate file, set it as follows:

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

NOTE

Proxy-related values set in the configuration for CodeReady Containers have priority over values set with environment variables.

5.4. SETTING UP CODEREADY CONTAINERS ON A REMOTE SERVER

Configure a remote server to run a CodeReady Containers OpenShift cluster.

This procedure assumes the use of a Red Hat Enterprise Linux, Fedora, or CentOS server. Run every command in this procedure on the remote server.

WARNING

Perform this procedure only on a local network. Exposing an insecure server on the internet has many security implications.

Prerequisites

- CodeReady Containers is installed and set up on the remote server. For more information, see Installing CodeReady Containers and Setting up CodeReady Containers.
- Your user account has **sudo** permissions on the remote server.

Procedure

1. Start the cluster:

```
$ crc start
```

   Ensure that the cluster remains running during this procedure.

2. Install the **haproxy** package and other utilities:

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

3. Modify the firewall to allow communication with the cluster:

```
$ 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. For SELinux, allow HAProxy to listen on TCP port 6443 to serve **kube-apiserver** on this port:

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

5. Create a backup of the default **haproxy** configuration:

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

6. Configure **haproxy** for use with the cluster:

```
$ 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. Start the **haproxy** service:

```
$ sudo systemctl start haproxy
```

5.5. CONNECTING TO A REMOTE CODEREADY CONTAINERS INSTANCE

Use **dnsmasq** to connect a client machine to a remote server running a CodeReady Containers OpenShift cluster.

This procedure assumes the use of a Red Hat Enterprise Linux, Fedora, or CentOS client. Run every command in this procedure on the client.

IMPORTANT

Connect to a server that is only exposed on your local network.

Prerequisites

- A remote server is set up for the client to connect to. For more information, see Setting up CodeReady Containers on a remote server.
- You know the external IP address of the server.
- You have the latest OpenShift CLI (**oc**) in your **$PATH** on the client.

Procedure

1. Install the **dnsmasq** package:

```
$ sudo dnf install dnsmasq
```

2. Enable the use of **dnsmasq** for DNS resolution in NetworkManager:

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

3. Add DNS entries for CodeReady Containers to the **dnsmasq** configuration:

```
$ 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
```

NOTE

Comment out any existing entries in ***/etc/NetworkManager/dnsmasq.d/crc.conf***. These entries are created by running a local instance of CodeReady Containers and will conflict with the entries for the remote cluster.

4. Reload the NetworkManager service:

```
$ sudo systemctl reload NetworkManager
```

5. Log in to the remote cluster as the **developer** user with **oc**:

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

The remote OpenShift Web Console is available at https://console-openshift-console.apps-crc.testing.

CHAPTER 6. ADMINISTRATIVE TASKS

6.1. STARTING MONITORING

CodeReady Containers disables cluster monitoring by default to ensure that CodeReady Containers can run on a typical notebook. Monitoring is responsible for listing your cluster in the Red Hat Hybrid Cloud Console. Follow this procedure to enable monitoring for your cluster.

Prerequisites

- You must assign additional memory to the CodeReady Containers instance. At least 14 GiB of memory, a value of **14336**, is recommended for core functionality. Increased workloads will require more memory. For more information, see Configuring the instance.

Procedure

1. Set the **enable-cluster-monitoring** configurable property to **true**:

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

2. Start the instance:

```
$ crc start
```

WARNING

Cluster monitoring cannot be disabled. To remove monitoring, set the **enable-cluster-monitoring** configurable property to **false** and delete the existing CodeReady Containers instance.

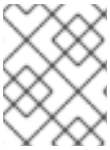
CHAPTER 7. TROUBLESHOOTING RED HAT CODEREADY CONTAINERS

NOTE

The goal of Red Hat CodeReady Containers is to deliver an OpenShift environment for development and testing purposes. Issues occurring during installation or usage of specific OpenShift applications are outside of the scope of CodeReady Containers. Report such issues to the relevant project. For example, OpenShift tracks issues on GitHub.

7.1. GETTING SHELL ACCESS TO THE OPENSHIFT CLUSTER

To access the cluster for troubleshooting or debugging purposes, follow this procedure.



NOTE

Direct access to the OpenShift cluster is not needed for regular use and is strongly discouraged.

Prerequisites

- Enable OpenShift CLI (**oc**) access to the cluster and log in as the **kubeadmin** user. For detailed steps, see Accessing the OpenShift cluster with the OpenShift CLI.

Procedure

1. Run the **oc get nodes** command to identify the desired node. The output will be similar to this:

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

2. Run **oc debug nodes/*<node>*** where ***<node>*** is the name of the node printed in the previous step.

7.2. TROUBLESHOOTING EXPIRED CERTIFICATES

The system bundle in each released **crc** executable expires 30 days after the release. This expiration is due to certificates embedded in the OpenShift cluster. The **crc start** command triggers an automatic certificate renewal process when needed. Certificate renewal can add up to five minutes to the start time of the cluster.

To avoid this additional startup time, or in case of failures in the certificate renewal process, use the following procedure:

Procedure

To resolve expired certificate errors that cannot be automatically renewed:

1. Download the latest CodeReady Containers release and place the **crc** executable in your **$PATH**.

2. Remove the cluster with certificate errors using the **crc delete** command:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers instance. Save any desired information stored in the instance before running this command.

3. Set up the new release:

```
$ crc setup
```

4. Start the new instance:

```
$ crc start
```

7.3. TROUBLESHOOTING BUNDLE VERSION MISMATCH

Created CodeReady Containers instances contain bundle information and instance data. Bundle information and instance data is not updated when setting up a new CodeReady Containers release. This information is not updated due to customization in the earlier instance data. This will lead to errors when running the **crc start** command:

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

Procedure

1. Issue the **crc delete** command before attempting to start the instance:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers instance. Save any desired information stored in the instance before running this command.

7.4. TROUBLESHOOTING UNKNOWN ISSUES

Resolve most issues by restarting CodeReady Containers with a clean state. This involves stopping the instance, deleting it, reverting changes made by the **crc setup** command, reapplying those changes, and restarting the instance.

Prerequisites

- You set up the host machine with the **crc setup** command. For more information, see Setting up CodeReady Containers.
- You started CodeReady Containers with the **crc start** command. For more information, see Starting the instance.
- You are using the latest CodeReady Containers release. Using a version earlier than CodeReady Containers 1.2.0 may result in errors related to expired x509 certificates. For more information, see Troubleshooting expired certificates.

Procedure

To troubleshoot CodeReady Containers, perform the following steps:

1. Stop the CodeReady Containers instance:

```
$ crc stop
```

2. Delete the CodeReady Containers instance:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers instance. Save any desired information stored in the instance before running this command.

3. Clean up remaining changes from the **crc setup** command:

```
$ crc cleanup
```

NOTE

The **crc cleanup** command removes an existing CodeReady Containers instance and reverts changes to DNS entries created by the **crc setup** command. On macOS, the **crc cleanup** command also removes the system tray.

4. Set up your host machine to reapply the changes:

```
$ crc setup
```

5. Start the CodeReady Containers instance:

```
$ crc start
```

NOTE

The cluster takes a minimum of four minutes to start the necessary containers and Operators before serving a request.

If your issue is not resolved by this procedure, perform the following steps:

1. Search open issues for the issue that you are encountering.
2. If no existing issue addresses the encountered issue, create an issue and attach the ***~/.crc/crc.log*** file to the created issue. The ***~/.crc/crc.log*** file has detailed debugging and troubleshooting information which can help diagnose the problem that you are experiencing.