Red Hat CodeReady Containers 1.22

Getting Started Guide

Quick-start guide to using and developing with CodeReady Containers

Last Updated: 2021-02-08

Red Hat CodeReady Containers 1.22 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.

Refer to the OpenShift documentation for a full introduction to OpenShift.

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

1.2. DIFFERENCES FROM A PRODUCTION OPENSHIFT INSTALLATION

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

- **The CodeReady Containers OpenShift cluster is ephemeral and is not intended for production use.**
- It uses a single node which behaves as both a master and worker node.
- It disables the **machine-config** and **monitoring** Operators by default.
  - These disabled Operators cause the corresponding parts of the web console to be non-functional.
  - For the same reason, there is no upgrade path to newer OpenShift versions.
- The OpenShift instance runs in a virtual machine. This may cause other differences, particularly with external networking.

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

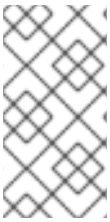
2.1. MINIMUM SYSTEM REQUIREMENTS

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

2.1.1. Hardware requirements

CodeReady Containers requires the following system resources:

- 4 virtual CPUs (vCPUs)
- 9 GB of free memory
- 35 GB of storage space



NOTE

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

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 newer. 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 10.12 Sierra or newer. CodeReady Containers does not work on earlier versions of macOS.

2.1.2.3. Linux

- On Linux, CodeReady Containers is only supported on Red Hat Enterprise Linux/CentOS 7.5 or newer (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 newer and Debian 10 or newer are not officially 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

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 and extract the contents of the archive to a location in your ***PATH***.

2.4. 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 virtual machine:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers virtual machine. Save any desired information stored in the virtual machine 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 virtual machine:

```
$ 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 virtual machine.

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

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 **root** (or Administrator). Always run the **crc** executable with your user account.
- If you are setting up a new version, capture any changes made to the virtual machine before setting up a new CodeReady Containers release.

Procedure

1. Set up your host machine for CodeReady Containers:

```
$ crc setup
```

Consent for telemetry data collection

The **crc setup** command prompts you for optional, anonymous usage data collection to assist with development. No personally identifiable information is collected.

- 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
```

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

3.2. STARTING THE VIRTUAL MACHINE

The **crc start** command starts the CodeReady Containers virtual machine and OpenShift cluster.

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 through 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.
- You have a valid OpenShift user pull secret. Copy or download the pull secret from the Pull Secret section of the Install on Laptop: Red Hat CodeReady Containers page on cloud.redhat.com.

NOTE

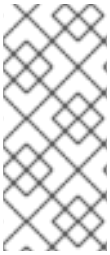
Accessing the user pull secret requires a Red Hat account.

Procedure

1. Start the CodeReady Containers virtual machine:

```
$ crc start
```

2. When prompted, supply your user pull secret.



NOTE

- The cluster takes a minimum of four minutes to start the necessary containers and Operators before serving a request.
- If you see errors during **crc start**, check the Troubleshooting CodeReady Containers section for potential solutions.

Additional resources

- To change the default resources allocated to the virtual machine, see Configuring the virtual machine.

3.3. ACCESSING THE OPENSHIFT CLUSTER

Access the OpenShift cluster running in the CodeReady Containers virtual machine through the OpenShift web console or client executable (**oc**).

3.3.1. Accessing the OpenShift web console

Prerequisites

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

Procedure

To access the OpenShift web console, follow these steps:

1. Run **crc console**. This will open your web browser and direct it to the web console.
2. Choose the **htpasswd_provider** option in the OpenShift web console.

3. Log in as the **developer** user with the password printed in the output of the **crc start** command.

NOTE

- You can also view the password for the **developer** and **kubeadmin** users by running **crc console --credentials**.
- You can initially access the cluster through either the **kubeadmin** or **developer** user. Use the **developer** user for creating projects or OpenShift applications and for application deployment. Only use the **kubeadmin** user for administrative tasks such as creating new users, setting roles, and so on.

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 oc

Prerequisites

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

Procedure

To access the OpenShift cluster through the **oc** command, follow these steps:

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:

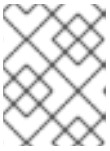
```
$ oc login -u developer https://api.crc.testing:6443
```

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:

```
$ oc get co
```



NOTE

- CodeReady Containers disables the **machine-config** and **monitoring** Operators 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 virtual machine 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. To access the internal OpenShift registry, follow these steps.

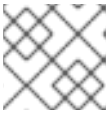
Prerequisites

- A running CodeReady Containers virtual machine. For more information, see Starting the virtual machine.
- A working **oc** command. For more information, see Accessing the OpenShift cluster with **oc**.
- 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 **kube:admin**.

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 for all tags in the imagestream:

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

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 OpenShift Do (**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 virtual machine.

Prerequisites

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

Procedure

To deploy a sample application through **odo**, follow these steps:

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 VIRTUAL MACHINE

The **crc stop** command stops the running CodeReady Containers virtual machine and OpenShift cluster. The stopping process will take a few minutes while the cluster shuts down.

Procedure

- Stop the CodeReady Containers virtual machine and OpenShift cluster:

```
$ crc stop
```

3.6. DELETING THE VIRTUAL MACHINE

The **crc delete** command deletes an existing CodeReady Containers virtual machine.

Procedure

- Delete the CodeReady Containers virtual machine:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers virtual machine. Save any desired information stored in the virtual machine 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 virtual machine. 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 VIRTUAL MACHINE

Use the **cpus** and **memory** properties to configure the default number of vCPUs and amount of memory available to the CodeReady Containers virtual machine, 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 virtual machine. To enable configuration changes, you must stop the running virtual machine and start it again.

Procedure

- To configure the number of vCPUs available to the virtual machine:

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

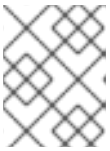
  The default value for the **cpus** property is **4**. The number of vCPUs to assign must be greater than or equal to the default.

- To start the virtual machine with the desired number of vCPUs:

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

- To configure the memory available to the virtual machine:

```
$ 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 virtual machine 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 will be exposed as **api.crc.testing** while the OpenShift console is accessed through **console-openshift-console.apps-crc.testing**. These DNS domains are served by a **dnsmasq** DNS container running inside the CodeReady Containers virtual machine.

Running **crc setup** will adjust 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 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 virtual machine.
- **systemd-resolved** configuration is done through a NetworkManager dispatcher script in **/etc/NetworkManager/dispatcher.d/99-crc.sh**:

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

exit 0
```

5.1.2.2. NetworkManager + dnsmasq

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

- CodeReady Containers expects NetworkManager to manage networking.
- NetworkManager uses **dnsmasq** through 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.

5.1.3. macOS

On macOS, CodeReady Containers expects the following DNS configuration:

- CodeReady Containers creates a **/etc/resolver/testing** file which instructs macOS to forward all DNS requests for the **testing** domain to the CodeReady Containers virtual machine.
- CodeReady Containers also adds an **api.crc.testing** entry to **/etc/hosts** pointing at the VM IP address. The **oc** executable requires this entry. See OpenShift issue #23266 for more information.

5.2. STARTING CODEREADY CONTAINERS BEHIND A PROXY

Prerequisites

- To use an existing **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 **oc**.

Procedure

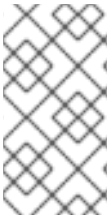
1. Define a proxy using the **http_proxy** and **https_proxy** environment variables or using the **crc config set** command as follows:

```
$ 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>
```

The **crc** executable will be able to use the defined proxy once set through environment variables or CodeReady Containers configuration.



NOTE

- Proxy-related values set in the configuration for CodeReady Containers have priority over values set through environment variables.
- SOCKS proxies are not supported by OpenShift Container Platform.

5.3. SETTING UP CODEREADY CONTAINERS ON A REMOTE SERVER

Follow this procedure to configure a remote server to run a CodeReady Containers OpenShift cluster.

NOTE

- **It is strongly advised to perform this procedure on a local network**Exposing an insecure server on the internet comes with many security implications.
- All of the commands in this procedure must be run on the remote server.
- This procedure assumes the use of a Red Hat Enterprise Linux, Fedora, or CentOS server.

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 throughout this procedure.

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

```
$ sudo dnf install haproxy policycoreutils-python-utils jq
```

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

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --add-port=80/tcp --permanent
$ sudo firewall-cmd --add-port=6443/tcp --permanent
$ sudo firewall-cmd --add-port=443/tcp --permanent
$ sudo systemctl restart firewalld
```

4. For SELinux, allow listening to TCP port 6443:

```
$ 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
    debug

defaults
    log global
```

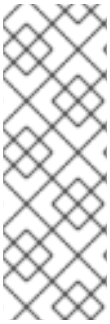
```
    mode http
    timeout connect 5000
    timeout client 500000
    timeout server 500000

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

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

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

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

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

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

7. Start the **haproxy** service:

```
$ sudo systemctl start haproxy
```

5.4. CONNECTING TO A REMOTE CODEREADY CONTAINERS INSTANCE

Follow this procedure to connect a client machine to a remote server running a CodeReady Containers OpenShift cluster.



NOTE

- **It is strongly advised to connect to a server that is only exposed on your local network.**
- All of the commands in this procedure must be run on the client.
- This procedure assumes the use of a Red Hat Enterprise Linux, Fedora, or CentOS client.

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.
- NetworkManager is installed and running.
- You know the external IP address of the server.
- You have the latest OpenShift client executable (**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, ALERTING, AND TELEMETRY

To make sure CodeReady Containers can run on a typical laptop, some resource-heavy services get disabled by default. One of these is Prometheus and the related monitoring, alerting, and telemetry functionality. Telemetry functionality is responsible for listing your cluster in the Red Hat OpenShift Cluster Manager.

Prerequisites

- You must assign additional memory to the CodeReady Containers virtual machine. 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 virtual machine.

Procedure

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

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

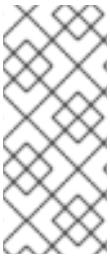
2. Start the virtual machine:

```
$ crc start
```

WARNING

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

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

Direct access to the OpenShift cluster is not needed for regular use and is strongly discouraged. To access the cluster for troubleshooting or debugging purposes, follow this procedure.

Prerequisites

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

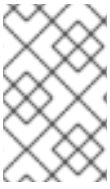
Procedure

1. Run **oc get nodes**. 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



NOTE

As of the CodeReady Containers 1.10.0 release, the certificate renewal process is not working as intended. Follow this procedure to avoid potential errors due to certificate expiration.

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. As a result, using an older **crc** executable or system bundle can result in an expired certificates error.

Starting from CodeReady Containers 1.2.0, the embedded certificates can be automatically renewed by **crc**. The **crc start** command triggers the certificate renewal process when needed. Certificate renewal can add up to five minutes to the start time of the cluster.

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 virtual machine. Save any desired information stored in the virtual machine before running this command.

3. Set up the new release:

```
$ crc setup
```

4. Start the new virtual machine:

```
$ crc start
```

7.3. TROUBLESHOOTING BUNDLE VERSION MISMATCH

Created CodeReady Containers virtual machines 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 virtual machine. Save any desired information stored in the virtual machine before running this command.

7.4. TROUBLESHOOTING UNKNOWN ISSUES

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

Prerequisites

- You set up the host machine through the **crc setup** command. For more information, see Setting up CodeReady Containers.
- You started CodeReady Containers through the **crc start** command. For more information, see Starting the virtual machine.
- 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 virtual machine:

```
$ crc stop
```

2. Delete the CodeReady Containers virtual machine:

```
$ crc delete
```

WARNING

The **crc delete** command results in the loss of data stored in the CodeReady Containers virtual machine. Save any desired information stored in the virtual machine 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 virtual machine 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 virtual machine:

```
$ 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.