# Red Hat CodeReady Workspaces 2.1

## Installation Guide

Installing Red Hat CodeReady Workspaces 2.1

Last Updated: 2020-07-02

# Red Hat CodeReady Workspaces 2.1 Installation Guide

Installing Red Hat CodeReady Workspaces 2.1

Supriya Takkhi

Robert Kratky
rkratky@redhat.com

Michal Maléř
mmaler@redhat.com

Fabrice Flore-Thébault
ffloreth@redhat.com

Yana Hontyk
yhontyk@redhat.com

## Legal Notice



## Abstract

Information for administrators installing Red Hat CodeReady Workspaces.

# Table of Contents

# CHAPTER 1. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT CONTAINER PLATFORM 4

## 1.1. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT 4 FROM OPERATORHUB

Operators are a method of packaging, deploying, and managing a OpenShift application which also provide the following:

- Repeatability of installation and upgrade.
- Constant health checks of every system component.
- Over-the-air (OTA) updates for OpenShift components and ISV content.
- A place to encapsulate knowledge from field engineers and spread it to all users.

On OpenShift, Red Hat CodeReady Workspaces can be installed using the OperatorHub Catalog present in the OpenShift web console.

**NOTE**

It is possible to use the **crwctl** utility script for deploying CodeReady Workspaces on OpenShift Container Platform and OpenShift Dedicated versions 4.4. This method is considered unofficial and serves as a backup installation method for situations where the installation method using OperatorHub is not available.

For information about how to use the **crwctl** utility script for deploying CodeReady Workspaces on OpenShift, see the Installing CodeReady Workspaces on OpenShift 3 using the Operator section.

Following steps are described:

- Section 1.1.1, “Creating the CodeReady Workspaces project in OpenShift 4 web console” .
- Section 1.1.2, “Installing the CodeReady Workspaces Operator in OpenShift 4 web console” .
- Section 1.1.3, “Installing CodeReady Workspaces using the CodeReady Workspaces Operator in OpenShift 4 web console”.
- Section 1.1.4, “Viewing the state of the CodeReady Workspaces instance deployment in OpenShift 4 web console”.
- Section 1.1.6, “Viewing the state of the CodeReady Workspaces cluster deployment using OpenShift 4 CLI tools”.
- Section 1.1.5, “Finding CodeReady Workspaces instance URL in OpenShift 4 web console” .
- Section 1.1.7, “Finding CodeReady Workspaces cluster URL using the OpenShift 4 CLI” .
- Section 1.1.8, “Enabling SSL on OpenShift 4” .
- Section 1.1.9, “Logging in to CodeReady Workspaces on OpenShift for the first time using OAuth”.

- Section 1.1.10, "Logging in to CodeReady Workspaces on OpenShift for the first time registering as a new user".

### 1.1.1. Creating the CodeReady Workspaces project in OpenShift 4 web console

This section describes how to create the **CodeReady Workspaces** project in OpenShift 4 web console.

**Prerequisites**

- An administrator account on a running instance of OpenShift 4.

**Procedure**

1. Open the OpenShift web console.
2. In the left panel, navigate to **Projects**.
3. Click the **Create Project** button.
4. Enter the project details:
   - In the **Name** field, type **codeready**.
   - In the **Display Name** field, type **Red Hat CodeReady Workspaces**.
5. Click the **Create** button.

### 1.1.2. Installing the CodeReady Workspaces Operator in OpenShift 4 web console

This section describes how to install the CodeReady Workspaces Operator in OpenShift 4 web console.

**Prerequisites**

- An administrator account on a running instance of OpenShift 4.
- Administrative rights on an existing project named **codeready** on this instance of OpenShift 4. See Section 1.1.1, "Creating the CodeReady Workspaces project in OpenShift 4 web console" .
- The Red Hat CodeReady Workspaces 2.0 Operator is not installed.

**Procedure**

1. Open the OpenShift web console.
2. In the left panel, navigate to the **Operators → OperatorHub** section.
3. In the **Search by keyword** field, type **Red Hat CodeReady Workspaces**.
4. Click on the **Red Hat CodeReady Workspaces** tile.
5. Click the **Install** button in the **Red Hat CodeReady Workspaces** pop-up window.
6. In the **A specific namespace on the cluster** field, in the cluster drop-down list, select the namespace into which the previous version of the CodeReady Workspaces Operator was installed.
7. Click the **Subscribe** button.

8. In the left panel, navigate to the **Operators → Installed Operators** section.
9. Red Hat CodeReady Workspaces is displayed as an installed Operator having the **InstallSucceeded** status.
10. Click on the **Red Hat CodeReady Workspaces** name in the list of installed operators.
11. Navigate to the **Overview** tab.
12. In the **Conditions** sections at the bottom of the page, wait for this message: **install strategy completed with no errors**.
13. Navigate to the **Events** tab.
14. Wait for this message: **install strategy completed with no errors**.

### 1.1.3. Installing CodeReady Workspaces using the CodeReady Workspaces Operator in OpenShift 4 web console

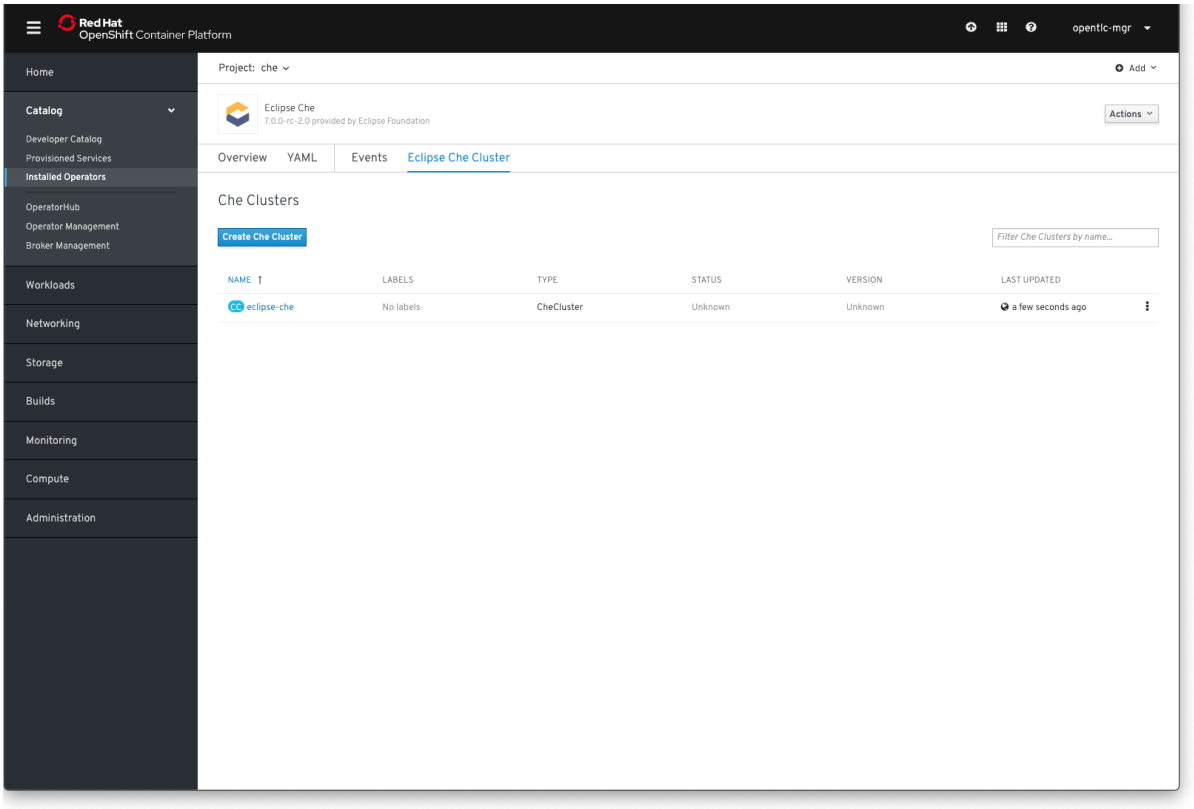
This section describes how to install CodeReady Workspaces using the CodeReady Workspaces Operator in OpenShift 4 web console.

**Prerequisites**

- An administrator account on a running instance of OpenShift 4.
- At least one OAuth user provisioned on this instance of OpenShift 4.
- The CodeReady Workspaces Operator is installed on this instance of OpenShift 4. See Section 1.1.2, "Installing the CodeReady Workspaces Operator in OpenShift 4 web console"

**Procedure**

1. Open the OpenShift web console.
2. Navigate to the **Operators → Installed Operators** section.
3. Click **Red Hat CodeReady Workspaces** in the list of installed operators.
4. Click the **Create Instance** link in **Provided APIs** section.
5. The **Create CodeReady Workspaces Cluster** page is displayed.
6. Leave the default values as they are.
7. Click the **Create** button in the bottom-left corner of the window.
8. The **codeready** cluster is created.




### 1.1.4. Viewing the state of the CodeReady Workspaces instance deployment in OpenShift 4 web console

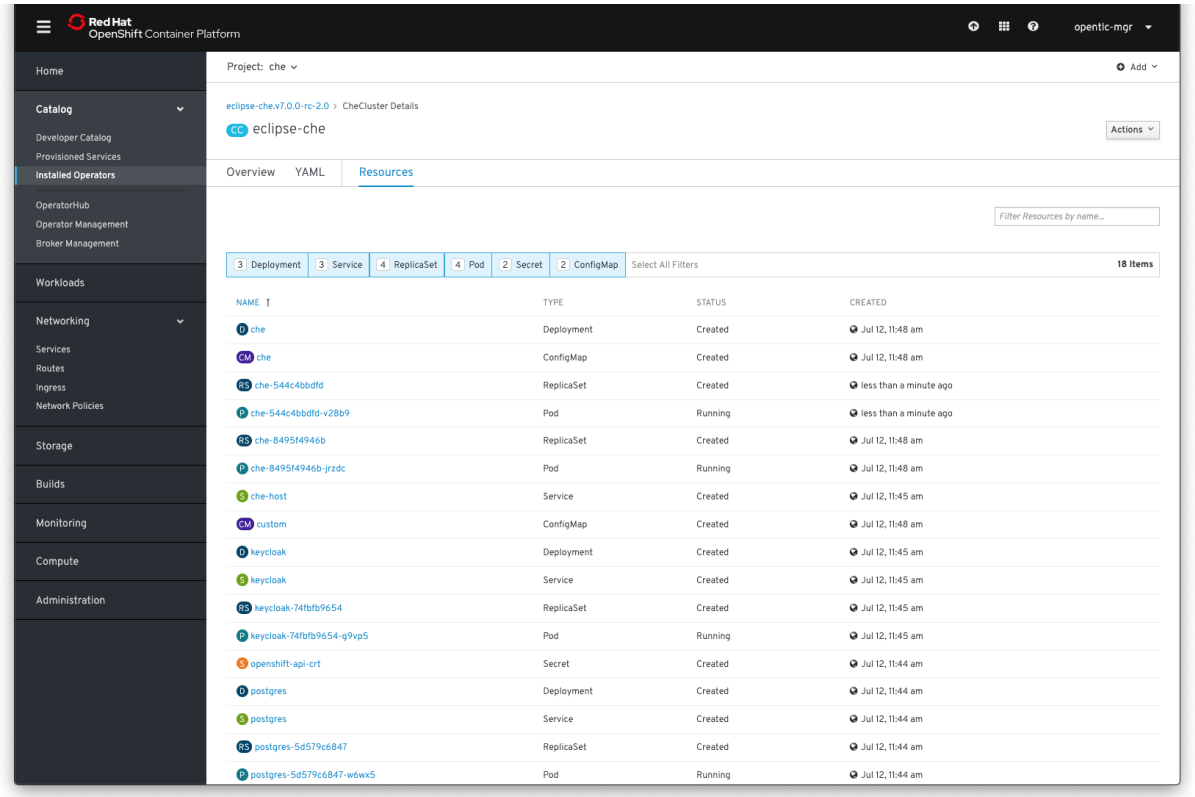
This section describes how to view the state of the CodeReady Workspaces instance deployment in OpenShift 4 web console.

#### Prerequisites

- An administrator account on a running instance of OpenShift 4.
- A CodeReady Workspaces is being deployed on this instance of OpenShift 4.

#### Procedure

1. Open the OpenShift web console.
2. Navigate to the **Operators → Installed Operators** section.
3. Click **Red Hat CodeReady Workspaces**in the list of installed operators.
4. Navigate to the **CodeReady Workspaces Cluster** tab.
5. Click **codeready-workspaces CheCluster** in the table.
   The **Overview** tab is displayed.
6. Watch the content of the **Message** field. The field contain error messages, if any. The expected content is **None**.
7. Navigate to the **Resources** tab.
   The screen displays the state of the resources assigned to the CodeReady Workspaces deployment.




### 1.1.5. Finding CodeReady Workspaces instance URL in OpenShift 4 web console

This section descibes how to find the CodeReady Workspaces instance URL in OpenShift 4 web console.

#### Prerequisites

- A running Red Hat CodeReady Workspaces instance.
  - See Section 1.1.4, “Viewing the state of the CodeReady Workspaces instance deployment in OpenShift 4 web console”.
  - See Section 1.1.6, “Viewing the state of the CodeReady Workspaces cluster deployment using OpenShift 4 CLI tools”.

#### Procedure

1. Open the OpenShift web console.
2. In the left panel, navigate to the **Operators → Installed Operators** section.
3. Click the **Red Hat CodeReady Workspaces** Operator tile.
4. Click **codeready-workspaces CheCluster** in the table.
   The **Overview** tab is displayed.
5. Read the value of the **CodeReady Workspaces URL** field.

### 1.1.6. Viewing the state of the CodeReady Workspaces cluster deployment using OpenShift 4 CLI tools

This section describes how to view the state of the CodeReady Workspaces cluster deployment using OpenShift 4 CLI tools.

**Prerequisites**

- An installation of an Red Hat CodeReady Workspaces cluster. See Section 1.1.3, "Installing CodeReady Workspaces using the CodeReady Workspaces Operator in OpenShift 4 web console".

**Procedure**

1. Run the following commands to select the **crw** project:

```
$ oc project <project_name>
```

2. Run the following commands to get the name and status of the Pods running in the selected project:

```
$ oc get pods
```

3. Check that the status of all the Pods is **Running**.

```
NAME                                  READY     STATUS    RESTARTS   AGE
codeready-8495f4946b-jrzdc            0/1       Running   0          86s
codeready-operator-578765d954-99szc   1/1       Running   0          42m
keycloak-74fbfb9654-g9vp5             1/1       Running   0          4m32s
postgres-5d579c6847-w6wx5             1/1       Running   0          5m14s
```

4. To see the state of the CodeReady Workspaces cluster deployment, run:

```
$ oc logs --tail=10 -f `oc get pods -o name | grep operator`
```

Example output of the command:

```
time="2019-07-12T09:48:29Z" level=info msg="Exec successfully completed"
time="2019-07-12T09:48:29Z" level=info msg="Updating eclipse-che CR with status:
provisioned with OpenShift identity provider: true"
time="2019-07-12T09:48:29Z" level=info msg="Custom resource eclipse-che updated"
time="2019-07-12T09:48:29Z" level=info msg="Creating a new object: ConfigMap, name:
che"
time="2019-07-12T09:48:29Z" level=info msg="Creating a new object: ConfigMap, name:
custom"
time="2019-07-12T09:48:29Z" level=info msg="Creating a new object: Deployment, name:
che"
time="2019-07-12T09:48:30Z" level=info msg="Updating eclipse-che CR with status:
CodeReady Workspaces API: Unavailable"
time="2019-07-12T09:48:30Z" level=info msg="Custom resource eclipse-che updated"
time="2019-07-12T09:48:30Z" level=info msg="Waiting for deployment che. Default timeout:
420 seconds"
```

### 1.1.7. Finding CodeReady Workspaces cluster URL using the OpenShift 4 CLI

This section describes how to obtain the CodeReady Workspaces cluster URL using the OpenShift 4 CLI (command line interface). The URL can be retrieved from the OpenShift logs or from the **checluster** Custom Resource.

**Prerequisites**

- An instance of Red Hat CodeReady Workspaces running on OpenShift.
- User is located in a CodeReady Workspaces installation namespace.

**Procedure**

1. To retrieve the CodeReady Workspaces cluster URL from the **checluster** CR (Custom Resource), run:

```
$ oc get checluster --output jsonpath='{.items[0].status.cheURL}'
```

2. Alternatively, to retrieve the CodeReady Workspaces cluster URL from the OpenShift logs, run:

```
$ oc logs --tail=10 `(oc get pods -o name | grep operator)` | \
  grep "available at" | \
  awk -F'available at: ' '{print $2}' | sed 's/"//'
```

### 1.1.8. Enabling SSL on OpenShift 4

**Prerequisites**

- A running Red Hat CodeReady Workspaces cluster.
  - See Section 1.1.4, "Viewing the state of the CodeReady Workspaces instance deployment in OpenShift 4 web console".
  - See Section 1.1.6, "Viewing the state of the CodeReady Workspaces cluster deployment using OpenShift 4 CLI tools".

**Procedure**

1. Open the OpenShift web console.
2. In the left panel, navigate to the **Operators → Installed Operators** section.
3. Click on the **Red Hat CodeReady Workspaces** Operator tile.
4. Click on **eclipse-che** in the table.
5. Navigate to the **Overview** tab.
6. Toggle the **TLS MODE** switch to **True**.
7. Click **Confirm change**.


8. Navigate to the **Resources** tab.

9. Wait that the Pods are restarted.
10. Navigate to the **Overview** tab.
11. Click the **Red Hat CodeReady Workspaces URL**link.
12. Notice that the link is redirected to HTTPS.
13. The browser displays the Red Hat CodeReady Workspaces **Dashboard** using a valid *Let's Encrypt* certificate.

### 1.1.9. Logging in to CodeReady Workspaces on OpenShift for the first time using OAuth

This section describes how to log in to CodeReady Workspaces on OpenShift for the first time using OAuth.

#### Prerequisites

- Contact the administrator of the OpenShift instance to obtain the **Red Hat CodeReady Workspaces URL**.

#### Procedure

1. Navigate to the **Red Hat CodeReady Workspaces URL**to display the Red Hat CodeReady Workspaces login page.
2. Choose the **OpenShift OAuth** option.
3. The **Authorize Access** page is displayed.
4. Click on the **Allow selected permissions** button.
5. Update the account information: specify the **Username**, **Email**, **First name** and **Last name** fields and click the **Submit** button.

#### Validation steps

- The browser displays the Red Hat CodeReady Workspaces **Dashboard**.

### 1.1.10. Logging in to CodeReady Workspaces on OpenShift for the first time registering as a new user

This section describes how to log in to CodeReady Workspaces on OpenShift for the first time registering as a new user.

#### Prerequisites

- Contact the administrator of the OpenShift instance to obtain the **Red Hat CodeReady Workspaces URL**.

#### Procedure

1. Navigate to the **Red Hat CodeReady Workspaces URL**to display the Red Hat CodeReady Workspaces login page.

2. Choose the **Register as a new user** option.

3. Update the account information: specify the **Username**, **Email**, **First name** and **Last name** field and click the **Submit** button.

**Validation steps**

- The browser displays the Red Hat CodeReady Workspaces **Dashboard**.

## 1.2. INSTALLING CODEREADY WORKSPACES USING CLI MANAGEMENT TOOL

### 1.2.1. Installing the crwctl CLI management tool

This section describes how to install crwctl, the CodeReady Workspaces CLI management tool.

**Procedure**

1. Navigate to https://developers.redhat.com/products/codeready-workspaces/download.

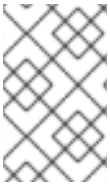
2. Download the CodeReady Workspaces CLI management tool archive for version 2.1.

3. Extract the archive.

4. Place the extracted binary in your **$PATH**.

### 1.2.2. Installing CodeReady Workspaces using CodeReady Workspaces CLI management tool

This sections describes how to install CodeReady Workspaces using the CodeReady Workspaces CLI management tool.



**NOTE**

Use CodeReady Workspaces CLI management tool to install CodeReady Workspaces only if OperatorHub is not available. This method is not oficially supported for OpenShift Container Platform 4.1 or later.

**Prerequisites**

- CodeReady Workspaces CLI management tool is installed.
- OpenShift Container Platform 4 CLI is installed.
- Access to an OpenShift Container Platform instance

#### 1.2.2.1. Installing with default settings

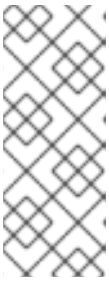
**Procedure**

1. Log in to OpenShift Container Platform 4:

```
$ oc login ${OPENSHIFT_API_URL} -u ${OPENSHIFT_USERNAME} -p
${OPENSHIFT_PASSWORD}
```

2. Run this command to install Red Hat CodeReady Workspaces with defaults settings:

```
$ crwctl server:start
```



NOTE

crwctl default namespace is **workspaces**. If you use a namespace with a different name, run the command with the **--chenamespace=*your namespace*** flag, for example:

```
$ {prod-cli} server:start --chenamespace=codeready-workspaces
```

#### 1.2.2.2. Installing with custom settings

**Procedure**

To override specific settings of the red-hat-codeready-workspaces installation, provide a dedicated custom resource when running the above **crwctl** command:

1. Download the default custom resource YAML file.
2. Name the downloaded custom resource **org_v1_che_cr.yaml**, and copy it into the current directory.
3. Modify the **org_v1_che_cr.yaml** file to override or add any field.
4. Run the installation using the **org_v1_che_cr.yaml** file to override the CodeReady Workspaces CLI management tool defaults:

```
$ crwctl server:start --che-operator-cr-yaml=org_v1_che_cr.yaml
```

NOTE

Some basic installation settings can be overridden in a simpler way by using additional **crwctl** arguments. To display the list of available arguments:

```
$ crwctl server:start --help
```

# CHAPTER 2. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT 3 USING THE OPERATOR

Operators are a method of packaging, deploying, and managing a OpenShift application which also provide the following:

- Repeatability of installation and upgrade.
- Constant health checks of every system component.
- Over-the-air (OTA) updates for OpenShift components and ISV content.
- A place to encapsulate knowledge from field engineers and spread it to all users.

This chapter describes how to install CodeReady Workspaces on OpenShift 3 using the CLI management tool and the Operator method.

## 2.1. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT 3 USING THE OPERATOR

This section describes how to install CodeReady Workspaces on OpenShift 3 with the CLI management tool to install via the Operator with SSL (HTTPS) enabled.

As of 2.1.1, SSL/TLS is enabled by default as it is required by the Che-Theia IDE.

### Prerequisites

- A running instance of OpenShift 3.11.
- Administrator rights on this OpenShift 3 instance.
- The **oc** OpenShift 3.11 CLI management tool is installed and configured. See Installing the OpenShift 3.11 CLI. To check the version of the **oc** tool, use the **oc version** command.
- The **crwctl** CLI management tool is installed. See Installing the **crwctl** CLI management tool.

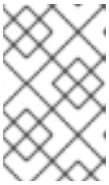
### Procedure

1. Log in to OpenShift. See Basic Setup and Login.

```
$ oc login
```

2. Run the following command to create the CodeReady Workspaces instance:

```
$ crwctl server:start -n <openshift_namespace>
```



NOTE

To create the CodeReady Workspaces instance on OpenShift clusters that have not been configured with a valid certificate for the routes, run the **crwctl** command with the **--self-signed-cert** flag.

### Verification steps

1. The output of the previous command ends with:

```
Command server:start has completed successfully.
```

2. Navigate to the CodeReady Workspaces cluster instance: **https://codeready-<openshift_deployment_name>.<domain_name>**. The domain uses *Let's Encrypt* ACME certificates.

# CHAPTER 3. INSTALLING CODEREADY WORKSPACES IN TLS MODE WITH SELF-SIGNED CERTIFICATES

The following section describes the deployment and configuration of CodeReady Workspaces with self-signed certificates. Self-signed certificates are certificates that are not signed by a commonly trusted certificate authority (CA), but instead signed by a locally created CA. Self-signed certificates are not trusted by default. For example, when a website owner uses a self-signed certificate to provide HTTPS services, users who visit that website see a warning in their browser.

**WARNING**

Self-signed certificates are usually used in development and evaluation environments. Use in production environments is not recommended.

## 3.1. GENERATING SELF-SIGNED TLS CERTIFICATES

This section describes how to prepare self-signed TLS certificates to use with CodeReady Workspaces on different platforms.

**Prerequisites**

- The expected domain name where the CodeReady Workspaces deployment is planned.
- The location of the **openssl.cnf** file on the target machine.

  **Table 3.1. Usual OpenSSL configuration file locations**

| Linux distribution | File location |
|---|---|
| Fedora, Red Hat Enterprise Linux, CentOS | **/etc/pki/tls/openssl.cnf** |
| Debian, Ubuntu, Mint, Arch Linux | **/etc/ssl/openssl.cnf** |

**Procedure**

1. Set the necessary environment variables:

```
$ CA_CN="Local Red Hat CodeReady Workspaces Signer"
$ DOMAIN=*.<expected.domain.com>
$ OPENSSL_CNF=<path_to_openssl.cnf>
```

2. Generate the root Certificate Authority (CA) key. Add the **-des3** parameter to use a passphrase:

```
$ openssl genrsa -out ca.key 4096
```

3. Generate the root CA certificate:

```
$ openssl req -x509 \
  -new -nodes \
  -key ca.key \
  -sha256 \
  -days 1024 \
  -out ca.crt \
  -subj /CN="${CA_CN}" \
  -reqexts SAN \
  -extensions SAN \
  -config <(cat ${OPENSSL_CNF} \
      <(printf '[SAN]\nbasicConstraints=critical, CA:TRUE\nkeyUsage=keyCertSign, cRLSign,
digitalSignature'))
```

4. Generate the domain key:

```
$ openssl genrsa -out domain.key 2048
```

5. Generate the certificate signing request for the domain:

```
$ openssl req -new -sha256 \
    -key domain.key \
    -subj "/O=Local Red Hat CodeReady Workspaces/CN=${DOMAIN}" \
    -reqexts SAN \
    -config <(cat ${OPENSSL_CNF} \
        <(printf "\n[SAN]\nsubjectAltName=DNS:${DOMAIN}\nbasicConstraints=critical,
CA:FALSE\nkeyUsage=digitalSignature, keyEncipherment, keyAgreement,
dataEncipherment\nextendedKeyUsage=serverAuth")) \
    -out domain.csr
```

6. Generate the domain certificate:

```
$ openssl x509 \
    -req \
    -sha256 \
    -extfile <(printf "subjectAltName=DNS:${DOMAIN}\nbasicConstraints=critical,
CA:FALSE\nkeyUsage=digitalSignature, keyEncipherment, keyAgreement,
dataEncipherment\nextendedKeyUsage=serverAuth") \
    -days 365 \
    -in domain.csr \
    -CA ca.crt \
    -CAkey ca.key \
    -CAcreateserial -out domain.crt
```

This procedure allows to use **domain.crt** and **domain.key** for TLS Route and Ingress, and **ca.crt** for importing into browsers.

**Additional resources**

- Section 3.4, “Importing self-signed TLS certificates to browsers”

## 3.2. DEPLOYING CODEREADY WORKSPACES WITH SELF-SIGNED TLS CERTIFICATES ON OPENSHIFT 4

This section describes how to deploy CodeReady Workspaces with self-signed TLS certificates on a local OpenShift 4 cluster.

CodeReady Workspaces uses a default router certificate to secure its endpoints. Therefore, it depends on the OpenShift cluster configuration whether a self-signed certificate is used or not. CodeReady Workspaces automatically detects if the OpenShift default router uses a self-signed certificate by analyzing its certificate chain.

**Prerequisites**

- A running OpenShift 4 instance, version 4.2 or higher.
- All required keys and certificates. See Section 3.1, "Generating self-signed TLS certificates".

**Procedure**

1. Log in to the default OpenShift project:

```
$ oc login -u <username> -p _<password>
```

2. Get the OpenShift 4 self-signed certificate:

```
$ oc get secret router-ca -n openshift-ingress-operator -o jsonpath="{.data.tls\.crt}" | \
  base64 -d > ca.crt
```

3. Pre-create a namespace for CodeReady Workspaces:

```
$ oc create namespace {prod-namespace}
```

4. Create a secret from the CA certificate:

```
$ oc create secret generic self-signed-certificate --from-file=ca.crt -n={prod-namespace}
```

5. Deploy CodeReady Workspaces using **crwctl**:

```
$ crwctl server:start --platform=openshift --installer=operator
```

   When using CodeReady Containers, substitute **openshift** in the above command with **crc**.

**Additional resources**

- Section 3.4, "Importing self-signed TLS certificates to browsers"

## 3.3. DEPLOYING CODEREADY WORKSPACES WITH SELF-SIGNED TLS CERTIFICATES ON OPENSHIFT 3

This section describes how to deploy CodeReady Workspaces with self-signed TLS certificates generated by the user on the OpenShift 3 platform.

**NOTE**

This method involves reconfiguration of OpenShift router to use user-provided TLS certificates.

**Prerequisites**

- A running OpenShift 3 instance, version 3.11 or higher.
- All required keys and certificates. See Section 3.1, "Generating self-signed TLS certificates".

**Procedure**

1. Log in to the default OpenShift project:

```
$ oc login -u system:admin --insecure-skip-tls-verify=true
$ oc project default
```

2. Reconfigure the router with the generated certificate:

```
$ oc delete secret router-certs
$ cat domain.crt domain.key > openshift.crt
$ oc create secret tls router-certs --key=domain.key --cert=openshift.crt
$ oc rollout latest router
```

3. Create a namespace for CodeReady Workspaces:

```
$ oc create namespace workspaces
```

4. Create a secret from the CA certificate:

```
$ oc create secret generic self-signed-certificate --from-file=ca.crt -n=workspaces
```

5. Deploy CodeReady Workspaces using **crwctl**. Red Hat CodeReady Workspaces is installed with TLS mode by default:

```
$ crwctl server:start --platform=openshift --installer=operator
```

**Additional resources**

- Section 3.4, "Importing self-signed TLS certificates to browsers"

## 3.4. IMPORTING SELF-SIGNED TLS CERTIFICATES TO BROWSERS

This section describes how to import a root certificate authority into a web browser to use CodeReady Workspaces with self-signed TLS certificates.

When a TLS certificate is not trusted, the error message **Authorization token is missing. Click here to reload page** blocks the login process. To prevent this, add the public part of the self-signed CA certificate into the browser after installing CodeReady Workspaces.

### 3.4.1. Getting the self-signed CA certificate from CodeReady Workspaces deployment

When **crwctl** is used to deploy CodeReady Workspaces, it exports a self-signed CA certificate into a **cheCA.crt** file to the current user home directory. To get the certificate, use one of the following two methods:

- Exporty the certificate using the crwctl command:

```
$ crwctl cacert:export
```

- Read the **self-signed-certificate** secret from the CodeReady Workspaces namespace:

```
$ oc get secret self-signed-certificate -n workspaces
```

### 3.4.2. Adding certificates to Google Chrome on Linux or Windows

**Procedure**

1. Navigate to URL where CodeReady Workspaces is deployed.
2. Save the certificate:
   a. Click the lock icon on the left of the address bar.
   b. Click **Certificates** and navigate to the **Details** tab.
   c. Select the certificate to use and export it:
      - On Linux, click the **Export** button.
      - On Windows, click the **Save to file** button.
3. Go to Google Chrome Settings, then to the **Authorities** tab
4. In the left panel, select **Advanced** and continue to **Privacy and security**.
5. At the center of the screen, click **Manage certificates** and navigate to **Authorities** tab.
6. Click the **Import** button and open the saved certificate file.
7. Select **Trust this certificate for identifying websites** and click the **OK** button.
8. After adding the CodeReady Workspaces certificate to the browser, the address bar displays the closed lock icon next to the URL, indicating a secure connection.

### 3.4.3. Adding certificates to Google Chrome on macOS

**Procedure**

1. Navigate to URL where CodeReady Workspaces is deployed.
2. Save the certificate:
   a. Click the lock icon on the left of the address bar.

b. Click **Certificates**.

c. Select the certificate to use and drag and drop its displayed large icon to the desktop.

3. Double-click the exported certificate to import it into Google Chrome.

### 3.4.4. Adding certificates to Keychain Access for use with Safari on macOS

**Procedure**

1. Navigate to URL where CodeReady Workspaces is deployed.
2. Save the certificate:
   a. Click the lock icon on the right of the window title bar.
   b. Select the certificate to use and drag and drop its displayed large icon to the desktop.
3. Open the **Keychain Access** application.
4. Select the **System** keychain and drag and drop the saved certificate file to it.
5. Double-click the imported CA, then go to **Trust** and select **When using this certificate**: **Always Trust**.
6. Restart Safari for the added certificated to take effect.

### 3.4.5. Adding certificates to Firefox

**Procedure**

1. Navigate to URL where CodeReady Workspaces is deployed.
2. Save the certificate:
   a. Click the lock icon on the left of the address bar.
   b. Click the **>** button next to the **Connection not secure** warning.
   c. Click the **More information** button.
   d. Click the **View Certificate** button on the **Security** tab.
   e. Click the **PEM (cert)** link and save the certificate.
3. Navigate to about:preferences, search for **certificates**, and click **View Certificates**.
4. Go to the **Authorities** tab, click the **Import** button, and open the saved certificate file.
5. Check **Trust this CA to identify websites** and click **OK**.
6. Restart Firefox for the added certificated to take effect.
7. After adding the CodeReady Workspaces certificate to the browser, the address bar displays the closed lock icon next to the URL, indicating a secure connection.

# CHAPTER 4. INSTALLING CODEREADY WORKSPACES IN A RESTRICTED ENVIROMENT

By default, Red Hat CodeReady Workspaces uses various external resources, mainly container images available in public registries.

To deploy CodeReady Workspaces in an environment where these external resources are not available (for example, on a cluster that is not exposed to the public Internet):

1. Identify the image registry used by the OpenShift cluster, and ensure you can push to it.
2. Push all the images needed for running CodeReady Workspaces to this registry.
3. Configure CodeReady Workspaces to use the images that have been pushed to the registry.
4. Proceed to the CodeReady Workspaces installation.

The procedure for installing CodeReady Workspaces in restricted environments is different based on the installation method you use:

- Installation using OperatorHub on Openshift 4.3 and above
- Installation using the crwctl management tool on both OpenShift 3.11 or 4.x

### Notes on network connectivity in restricted environments

Restricted network environments range from a private subnet in a cloud provider to a separate network owned by a company, disconnected from the public Internet. Regardless of the network configuration, CodeReady Workspaces works **provided that the Routes that are created for CodeReady Workspaces components (codeready-workspaces-server, identity provider, devfile and plugin registries) are accessible from inside the OpenShift cluster.**

Take into account the network topology of the environment to determine how best to accomplish this. For example, on a network owned by a company or an organization, the network administrators must ensure that traffic bound from the cluster can be routed to Route hostnames. In other cases, for example, on AWS, create a proxy configuration allowing the traffic to leave the node to reach an external-facing Load Balancer.

When the restricted network involves a proxy, follow the instructions provided in Section 4.3, “Preparing CodeReady Workspaces Custom Resource for installing behing a proxy”.

## 4.1. INSTALLING CODEREADY WORKSPACES IN A RESTRICTED ENVIROMENT USING OPERATORHUB

### Prerequisites

- A running OpenShift cluster. See the OpenShift Container Platform 4.3 documentation for instructions on how to install an OpenShift cluster on a restricted network.
- Access to the mirror registry used to installed the OpenShift disconnected cluster in restricted network. See the Related OpenShift Container Platform 4.3 documentation about creating a mirror registry for installation in a restricted network.

On disconnected OpenShift 4 clusters running on restricted networks, an Operator can be successfully installed from OperatorHub only if it meets the additional requirements defined in Enabling your Operator for restricted network environments.

The CodeReady Workspaces operator meets these requirements and is therefore compatible with the official documentation about OLM on a restricted network .

### Procedure

To install CodeReady Workspaces from OperatorHub:

1. Build a **redhat-operators** catalog image. See Building an Operator catalog image .
2. Configure OperatorHub to use this catalog image for operator installations. See Configuring OperatorHub for restricted networks.
3. Proceed to the CodeReady Workspaces installation as usual as described in Section 1.1, "Installing CodeReady Workspaces on OpenShift 4 from OperatorHub".

## 4.2. INSTALLING CODEREADY WORKSPACES IN A RESTRICTED ENVIROMENT USING CLI MANAGEMENT TOOL

NOTE

Use CodeReady Workspaces CLI management tool to install CodeReady Workspaces on restricted networks only if installation through OperatorHub is not available. This method is not officially supported for OpenShift Container Platform 4.1 or later.

### Prerequisites

- A running OpenShift cluster. See the OpenShift Container Platform 4.2 documentation for instructions on how to install an OpenShift cluster.

### 4.2.1. Preparing an image registry for installing CodeReady Workspaces in a restricted environment

### Prerequisites

- The **oc** tool is installed.
- An image registry that is accessible from the OpenShift cluster. Ensure you can push to it from a location that has, at least temporarily, access to the Internet.
- The **podman** tool is installed.

NOTE

When pushing images to other registry than the OpenShift internal registry, and the **podman** tool fails to work, use the **docker** tool instead.

The following placeholders are used in this section.

**Table 4.1. Placeholders used in examples**

| | |
|---|---|
| ***<internal-registry>*** | host name and port of the container-image registry accessible in the restricted environment |
| ***<organization>*** | organization of the container-image registry |

NOTE

For the OpenShift internal registry, the placeholder values are typically the following:

**Table 4.2. Placeholders for the internal OpenShift registry**

| | |
|---|---|
| ***<internal-registry>*** | **image-registry.openshift-image-registry.svc:5000** |
| ***<organization>*** | **openshift** |

See OpenShift documentation for more details.

**Procedure**

1. Define the environment variable with the external endpoint of the image registry:
For the OpenShift internal registry, use:

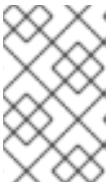
```
$ REGISTRY_ENDPOINT=$(oc get route default-route --namespace openshift-image-registry \
  --template='{{ .spec.host }}')
```

For other registries, use the host name and port of the image registry:

```
$ REGISTRY_ENDPOINT=<internal-registry>
```

2. Log into the internal image registry:

```
$ podman login --username <user> --password <password> <internal-registry>
```



NOTE

When using the OpenShift internal registry, follow the steps described in the related OpenShift documentation to first expose the internal registry through a route, and then log in to it.

3. Download, tag, and push the necessary images. Repeat the step for every image in the following lists:

```
$ podman pull <image_name>:<image_tag>
$ podman tag <image_name>:<image_tag>
${REGISTRY_ENDPOINT}/<organization>/<image_name>:<image_tag>
$ podman push ${REGISTRY_ENDPOINT}/<organization>/<image_name>:<image_tag>
```

**Essential images**

The following infrastructure images are included in every workspace launch:

- registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator:2.1
- registry.redhat.io/codeready-workspaces/server-rhel8:2.1
- registry.redhat.io/codeready-workspaces/pluginregistry-rhel8:2.1
- registry.redhat.io/codeready-workspaces/devfileregistry-rhel8:2.1
- registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8:2.1
- registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8:2.1
- registry.redhat.io/codeready-workspaces/jwtproxy-rhel8:2.1
- registry.redhat.io/codeready-workspaces/machineexec-rhel8:2.1
- registry.redhat.io/codeready-workspaces/theia-rhel8:2.1
- registry.redhat.io/codeready-workspaces/theia-dev-rhel8:2.1
- registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8:2.1
- registry.redhat.io/rhscl/postgresql-96-rhel7:1-47
- registry.redhat.io/redhat-sso-7/sso73-openshift:1.0-15
- registry.redhat.io/ubi8-minimal:8.1-398

#### Workspace-specific images

These are images that are required for running a workspace. A workspace generally uses only a subset of the images below. It is only necessary to include the images related to required technology stacks.

- registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8:2.1
- registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8:2.1
- registry.redhat.io/codeready-workspaces/stacks-golang-rhel8:2.1
- registry.redhat.io/codeready-workspaces/stacks-java-rhel8:2.1
- registry.redhat.io/codeready-workspaces/stacks-node-rhel8:2.1
- registry.redhat.io/codeready-workspaces/stacks-php-rhel8:2.1
- registry.redhat.io/codeready-workspaces/stacks-python-rhel8:2.1
- registry.redhat.io/codeready-workspaces/plugin-java11-rhel8:2.1
- registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8:2.1
- registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8:2.1

### 4.2.2. Preparing CodeReady Workspaces Custom Resource for restricted environment

When installing CodeReady Workspaces in a restricted environment using **crwctl** or OperatorHub, provide a **CheCluster** custom resource with additional information.

#### 4.2.2.1. Downloading the default CheCluster Custom Resource

**Procedure**

1. Download the default custom resource YAML file.
2. Name the downloaded custom resource **org_v1_che_cr.yaml**. Keep it for further modification and usage.

#### 4.2.2.2. Customizing the CheCluster Custom Resource for restricted environment

**Prerequisites**

- All required images available in an image registry that is visible to the OpenShift cluster where CodeReady Workspaces is to be deployed. This is described in Section 4.2.1, "Preparing an image registry for installing CodeReady Workspaces in a restricted environment", where the placeholders used in the following examples are also defined.

**Procedure**

1. In the **CheCluster** Custom Resource, which is managed by the CodeReady Workspaces Operator, add the fields used to facilitate deploying an instance of CodeReady Workspaces in a restricted environment:

```
# [...]
spec:
  server:
    airGapContainerRegistryHostname: '<internal-registry>'
    airGapContainerRegistryOrganization: '<organization>'
# [...]
```

Setting these fields in the Custom Resource uses ***<internal-registry>*** and ***<organization>*** for all images. This means, for example, that the Operator expects the offline plug-in and devfile registries to be available at:

```
<internal-registry>/<organization>/pluginregistry-rhel8:<ver>
<internal-registry>/<organization>/pluginregistry-rhel8:<ver>
```

For example, to use the OpenShift 4 internal registry as the image registry, define the following fields in the **CheCluster** Custom Resource:

```
# [...]
spec:
  server:
    airGapContainerRegistryHostname: 'image-registry.openshift-image-registry.svc:5000'
    airGapContainerRegistryOrganization: 'openshift'
# [...]
```

2. In the downloaded **CheCluster** Custom Resource, add the two fields described above with the proper values according to the container-image registry with all the mirrored container images.

### 4.2.3. Starting CodeReady Workspaces installation in a restricted environment using CodeReady Workspaces CLI management tool

This sections describes how to start the CodeReady Workspaces installation in a restricted environment using the CodeReady Workspaces CLI management tool.

**Prerequisites**

- CodeReady Workspaces CLI management tool is installed.
- The **oc** tool is installed.
- Access to an OpenShift instance.

**Procedure**

1. Log in to OpenShift Container Platform:

```
$ oc login ${OPENSHIFT_API_URL} --username ${OPENSHIFT_USERNAME} \
                        --password ${OPENSHIFT_PASSWORD}
```

2. Install CodeReady Workspaces with the customized Custom Resource to add fields related to restricted environment:

```
$ crwctl server:start \
  --che-operator-image=<image-registry>/<organization>/server-operator-rhel8:2.1 \
  --che-operator-cr-yaml=org_v1_che_cr.yaml
```

## 4.3. PREPARING CODEREADY WORKSPACES CUSTOM RESOURCE FOR INSTALLING BEHING A PROXY

This procedure describes how to provide necessary additional information to the **CheCluster** custom resource when installing CodeReady Workspaces behing a proxy.

**Procedure**

1. In the **CheCluster** Custom Resource, which is managed by the CodeReady Workspaces Operator, add the fields used to facilitate deploying an instance of CodeReady Workspaces in a restricted environment:

```
# [...]
spec:
  server:
    proxyURL: '<URL of the proxy, with the http protocol, and without the port>'
    proxyPort: '<Port of proxy, typically 3128>'
# [...]
```

2. In addition to those basic settings, the proxy configuration usually requires adding the host of the external OpenShift cluster API URL in the list of the hosts to be accessed from CodeReady Workspaces without using the proxy.
To retrieve this cluster API host, run the following command against the OpenShift cluster:

```
$ oc whoami --show-server | sed 's#https://##' | sed 's#:.*$##'
```

The corresponding field of the **CheCluster** Custom Resource is **nonProxyHosts**. If a host already exists in this field, use | as a delimiter to add the cluster API host:

```
# [...]
spec:
  server:
    nonProxyHosts: 'anotherExistingHost|<cluster api host>'
# [...]
```

# CHAPTER 5. UPGRADING CODEREADY WORKSPACES

This chapter describes how to upgrade a CodeReady Workspaces instance to CodeReady Workspaces 2.1.

## 5.1. UPGRADING CODEREADY WORKSPACES USING OPERATORHUB

This section describes how to upgrade from CodeReady Workspaces 2.0 to CodeReady Workspaces 2.1 on OpenShift 4 using the OpenShift web console. This method is using the Operator from OperatorHub.

**Prerequisites**

- An administrator account on an OpenShift 4 instance.
- An instance of CodeReady Workspaces 2.0, running on the same instance of OpenShift 4, installed using an Operator from OperatorHub.

**Procedure**

1. Open the OpenShift web console.
2. Navigate to the **Operators → Installed Operators** section.
3. Click **Red Hat CodeReady Workspaces**in the list of installed operators.
4. Navigate to the **Subscription** tab and enable the following options:
   - Channel: **latest**
   - Approval: **Automatic**

**Verification steps**

1. Log in to the CodeReady Workspaces instance.
2. The 2.1 version number is visible at the bottom of the page.

## 5.2. UPGRADING CODEREADY WORKSPACES USING CLI MANAGEMENT TOOL ON OPENSHIFT 3

This section describes how to upgrade from CodeReady Workspaces 2.0 to CodeReady Workspaces 2.1 on OpenShift 3 using the CLI management tool.

**Prerequisites**

- An administrative account on an OpenShift 3 instance.
- A running instance of Red Hat CodeReady Workspaces running on OpenShift 3, installed using the CLI management tool.
- The **crwctl** management tool installed.

**Procedure**

1. In all running workspaces in the CodeReady Workspaces 2.0 instance, save and push changes to Git repositories.

2. Run the following command:

```
$ crwctl server:update
```

**Verification steps**

1. Log in to the CodeReady Workspaces instance.

2. The 2.1 version number is visible at the bottom of the page.

## 5.3. UPGRADING CODEREADY WORKSPACES FROM PREVIOUS MAJOR VERSION

This sections describes how to perform an upgrade from the previous major version of Red Hat CodeReady Workspaces (1.2).

**Procedure**

- See Upgrading CodeReady Workspaces section in CodeReady Workspaces 2.0 Installation Guide

# CHAPTER 6. ADVANCED CONFIGURATION OPTIONS

The following section describes advanced deployment and configuration methods for Red Hat CodeReady Workspaces.

## 6.1. CODEREADY WORKSPACES CONFIGMAPS AND THEIR BEHAVIOR

The following section describes CodeReady Workspaces **configMaps** and how they behave.

A **configMap** is provided as an editable file that lists options to customize the CodeReady Workspaces environment. Based on the CodeReady Workspaces installation method, **configMaps** can be used to customize the working environment. The type of configMaps available in your CodeReady Workspaces environment varies based on the method used for installing CodeReady Workspaces.

### 6.1.1. CodeReady Workspaces installed using an Operator

Operators are software extensions to OpenShift that use custom resources to manage applications and their components.

CodeReady Workspaces installed using the Operator provides the user with an automatically generated **configMap** called **codeready**.

The **codeready configMap** contains the main properties for the CodeReady Workspaces server, and is in sync with the information stored in the CheCluster Custom Resource file. User modifications of the **codeready configMap** after installing CodeReady Workspaces using the Operator are automatically overwritten by values that the Operator obtains from the **CheCluster** Custom Resource.

To edit the **codeready configMap**, edit the Custom Resource manually. The **configMap** derives values from the **CheCluster** field. User modifications of the **CheCluster** Custom Resource field cause the Operator to change the attributes of the **codeready configMap** accordingly. The **configMap** changes automatically trigger a restart of the CodeReady Workspaces Pod.

To add custom properties to the CodeReady Workspaces server, such as environment variables that are not automatically generated in the **codeready configMap** by the Operator, or to override automatically generated properties, the **CheCluster** Custom Resource has a **customCheProperties** field, which expects a map.

For example, to overrride the default memory limit for workspaces, add the **CHE_WORKSPACE_DEFAULT__MEMORY__LIMIT__MB** property to **customCheProperties**:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: eclipse-che
  namespace: che
spec:
  server:
    cheImageTag: ''
    devfileRegistryImage: ''
    pluginRegistryImage: ''
    tlsSupport: true
    selfSignedCert: false
    customCheProperties:
```

```
    CHE_WORKSPACE_DEFAULT__MEMORY__LIMIT__MB: "2048"
  auth:
...
```

Previous versions of the CodeReady Workspaces Operator had a configMap named **custom** to fulfill this role. If the CodeReady Workspaces Operator finds a **configMap** with the name **custom**, it adds the data it contains into the **customCheProperties** field, redeploys CodeReady Workspaces, and deletes the **custom configMap**.

## 6.2. CONFIGURING NAMESPACE STRATEGIES

NOTE

The term *namespace* (Kubernetes) is used interchangeably with *project* (OpenShift).

The namespace strategies are configured using the **CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** environment variable.

WARNING

**CHE_INFRA_KUBERNETES_NAMESPACE** and **CHE_INFRA_OPENSHIFT_PROJECT** are legacy variables. Keep these variables unset for a new installations. Changing these variables during an update can lead to data loss.

### 6.2.1. One namespace per workspace strategy

The strategy creates a new namespace for each new workspace.

To use the strategy, the **CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** variable value must contain the ***<workspaceID>*** identifier. It can be used alone or combined with other identifiers or any string.

**Example 6.1. One namespace per workspace**

To assign namespace names composed of a **che-ws** prefix and workspace id, set:

```
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT=che-ws-<workspaceID>
```

### 6.2.2. One namespace for all workspaces strategy

The strategy uses one predefined namespace for all workspaces.

To use the strategy, the **CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** variable value must be the name of the desired namespace to use.

**Example 6.2. One namespace for all workspaces**

To have all workspaces created in **che-workspaces** namespace, set:

```
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT=che-workspaces
```

IMPORTANT

To run more than one workspace at a time when using this strategy together with the **common** PVC strategy, configure persistent volumes to use **ReadWriteMany** access mode.

### 6.2.3. One namespace per user strategy

The strategy isolates each user in their own namespace.

To use the strategy, the **CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** variable value must contain one or more user identifiers. Currently supported identifiers are ***<username>*** and ***<userId>***.

**Example 6.3. One namespace per user**

To assign namespace names composed of a **che-ws** prefix and individual usernames (**che-ws-user1**, **che-ws-user2**), set:

```
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT=che-ws-<username>
```

IMPORTANT

To run more than one workspace at a time when using this strategy together with the **common** PVC strategy, configure persistent volumes to use **ReadWriteMany** access mode.

To limit the number of concurrently running workspaces per user to one, set the **CHE_LIMITS_USER_WORKSPACES_RUN_COUNT** environment variable to **1**.

To limit the number of concurrently running workspaces per user to one (1):

- For Operator deployments: set the **spec.server.cheCustomProperties.CHE_LIMITS_USER_WORKSPACE_RUN_COUNT** variable of the CheCluster Custom Resource (CR) to **1**.

### 6.2.4. Allowing user-defined workspace namespaces

CodeReady Workspaces server can be configured to honor the user selection of a namespace when a workspace is created. This feature is disabled by default. To allow user-defined workspace namespaces:

- For Operator deployments, set the following field in the CheCluster Custom Resource:

```
allowUserDefinedWorkspaceNamespaces
```

## 6.3. DEPLOYING CODEREADY WORKSPACES WITH SUPPORT FOR GIT REPOSITORIES WITH SELF-SIGNED CERTIFICATES

This procedure describes how to configure CodeReady Workspaces for deployment with support for Git operations on repositories that use self-signed certificates.

### Prerequisites

- Git version 2 or later

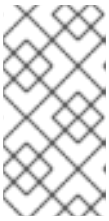
### Procedure

Configuring support for self-signed Git repositories.

1. Create a new **configMap** with details about the Git server:

```
$ oc create configmap che-git-self-signed-cert --from-file=ca.crt \
  --from-literal=githost=<host:port> -n {prod-namespace}
```

In the command, substitute ***<host:port>*** for the host and port of the HTTPS connection on the Git server (optional).



NOTE

- When **githost** is not specified, the given certificate is used for all HTTPS repositories.
- The certificate file must be named **ca.crt**.

2. Configure the workspace exposure strategy:
Update the **gitSelfSignedCert** property. To do that, execute:

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json \
  -p '[{"op": "replace", "path": "/spec/server/gitSelfSignedCert", "value": true}]'
```

3. Create and start a new workspace. Every container used by the workspace mounts a special volume that contains a file with the self-signed certificate. The repository's **.git/config** file contains information about the Git server host (its URL) and the path to the certificate in the **http** section (see Git documentation about git-config). For example:

```
[http "https://10.33.177.118:3000"]
        sslCAInfo = /etc/che/git/cert/ca.crt
```

## 6.4. ADDING SELF-SIGNED SSL CERTIFICATES TO CODEREADY WORKSPACES

When a CodeReady Workspaces user attempts to authenticate with RH-SSO that is using OpenShift OAuth, the authentication fails if the RH-SSO does not know the certificates needed for authorization.

To fix this problem, configure CodeReady Workspaces to authorize HTTPS communication with various components, such as identity and Git servers, by adding information about the self-signed SSL certificates to the CodeReady Workspaces configuration.

**Prerequisites**

- The OpenShift command-line tool, **oc** is installed.

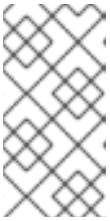
**Procedure**

1. Save the desired self-signed certificates to a local file system.
2. Create a new configMap with the required self-signed SSL certificates:

```
$ oc create configmap <configMap-name> --from-file=<certificate-file-path> -n=<che-namespace-name>
```

   To apply more than one certificate, add another **--from-file=*<certificate-file-path>*** option to the above command.

3. Define a name for the newly created configMap.



**NOTE**

Use these steps with existing instances of CodeReady Workspaces. To install a new instance of CodeReady Workspaces with self-signed SSL certificates, create a new Che Custom Resource or Helm Chart property, based on the installation method selected, instead of updating the existing configuration.

- For a CodeReady Workspaces Operators deployment:
  - Define a name for the newly created configMap by editing the **spec.server.ServerTrustStoreConfigMapName** Che Custom Resource property to match the previously created configMap:

```
$ oc patch checluster codeready-workspaces -n che --type=json -p '[{"op": "replace", "path": "/spec/server/serverTrustStoreConfigMapName", "value": "<config-map-name>"}]'
```

**Verification**

If the certificates have been added correctly, the CodeReady Workspaces server starts and obtains RH-SSO configuration over HTTPS with a self-signed SSL certificate, allowing user to:

- Access the CodeReady Workspaces server.
- Log in using OpenShift OAuth.
- Clone from a Git repository that has a custom self-signed SSL certificate over HTTPS.

## 6.5. CODEREADY WORKSPACES CONFIGMAPS FIELDS REFERENCE

### 6.5.1. server settings related to the CodeReady Workspaces server

| Property | Default value | Description |
|---|---|---|
| **airGapContainerRegistryHostname** | omit | An optional host name or URL to an alternative container registry to pull images from. This value overrides the container registry host name defined in all default container images involved in a CodeReady Workspaces deployment. This is particularly useful to install CodeReady Workspaces in an air-gapped environment. |
| **airGapContainerRegistryOrganization** | omit | Optional repository name of an alternative container registry to pull images from. This value overrides the container registry organization defined in all the default container images involved in a CodeReady Workspaces deployment. This is particularly useful to install CodeReady Workspaces in an air-gapped environment. |
| **cheDebug** | **false** | Enables the debug mode for CodeReady Workspaces server. |
| **cheFlavor** | **codeready-workspaces** | Flavor of the installation. |
| **cheHost** | The Operator automatically sets the value. | A public host name of the installed CodeReady Workspaces server. |
| **cheImagePullPolicy** | **Always** for **nightly** or **latest** images, and **IfNotPresent** in other cases | Overrides the image pull policy used in CodeReady Workspaces deployment. |
| **cheImageTag** | omit | Overrides the tag of the container image used in CodeReady Workspaces deployment. Omit it or leave it empty to use the default image tag provided by the Operator. |
| **cheImage** | omit | Overrides the container image used in CodeReady Workspaces deployment. This does not include the container image tag. Omit it or leave it empty to use the default container image provided by the Operator. |
| **cheLogLevel** | **INFO** | Log level for the CodeReady Workspaces server: **INFO** or **DEBUG**. |
| **cheWorkspaceClusterRole** | omit | Custom cluster role bound to the user for the workspaces. Omit or leave empty to use the default roles. |

| Property | Default value | Description |
|---|---|---|
| **customCheProperties** | omit | Map of additional environment variables that will be applied in the generated **codeready-workspaces** config map to be used by the CodeReady Workspaces server, in addition to the values already generated from other fields of the **CheCluster** custom resource (CR). If **customCheProperties** contains a property that would be normally generated in **codeready-workspaces** config map from other CR fields, then the value defined in the **customCheProperties** will be used instead. |
| **devfileRegistryImage** | omit | Overrides the container image used in the Devfile registry deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator. |
| **devfileRegistryMemoryLimit** | **256Mi** | Overrides the memory limit used in the Devfile registry deployment. |
| **devfileRegistryMemoryRequest** | **16Mi** | Overrides the memory request used in the Devfile registry deployment. |
| **devfileRegistryPullPolicy** | **Always** for **nightly** or **latest** images, and **IfNotPresent** in other cases | Overrides the image pull policy used in the Devfile registry deployment. |
| **devfileRegistryUrl** | The Operator automatically sets the value. | Public URL of the Devfile registry that serves sample, ready-to-use devfiles. Set it if you use an external devfile registry (see the **externalDevfileRegistry** field). |
| **externalDevfileRegistry** | **false** | Instructs the Operator to deploy a dedicated Devfile registry server. By default a dedicated devfile registry server is started. If **externalDevfileRegistry** set to **true**, the Operator does not start a dedicated registry server automatically and you need to set the **devfileRegistryUrl** field manually. |
| **externalPluginRegistry** | **false** | Instructs the Operator to deploy a dedicated Plugin registry server. By default, a dedicated plug-in registry server is started. If **externalPluginRegistry** set to **true**, the Operator does not deploy a dedicated server automatically and you need to set the **pluginRegistryUrl** field manually. |
| **nonProxyHosts** | omit | List of hosts that will not use the configured proxy. Use \|` as delimiter, for example **localhost\|my.host.com\|123.42.12.32** Only use when configuring a proxy is required (see also the **proxyURL** field). |

| Property | Default value | Description |
|---|---|---|
| **pluginRegistryImage** | omit | Overrides the container image used in the Plugin registry deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator. |
| **pluginRegistryMemoryLimit** | **256Mi** | Overrides the memory limit used in the Plugin registry deployment. |
| **pluginRegistryMemoryRequest** | **16Mi** | Overrides the memory request used in the Plugin registry deployment. |
| **pluginRegistryPullPolicy** | **Always** for **nightly** or **latest** images, and **IfNotPresent** in other cases | Overrides the image pull policy used in the Plugin registry deployment. |
| **pluginRegistryUrl** | the Operator sets the value automatically | Public URL of the Plugin registry that serves sample ready-to-use devfiles. Set it only when using an external devfile registry (see the **externalPluginRegistry** field). |
| **proxyPassword** | omit | Password of the proxy server. Only use when proxy configuration is required. |
| **proxyPort** | omit | Port of the proxy server. Only use when configuring a proxy is required (see also the **proxyURL** field). |
| **proxyURL** | omit | URL (protocol+host name) of the proxy server. This drives the appropriate changes in the **JAVA_OPTS** and **https(s)_proxy** variables in the CodeReady Workspaces server and workspaces containers. Only use when configuring a proxy is required. |
| **proxyUser** | omit | User name of the proxy server. Only use when configuring a proxy is required (see also the **proxyURL** field). |
| **selfSignedCert** | **false** | Enables the support of OpenShift clusters with routers that use self-signed certificates. When enabled, the Operator retrieves the default self-signed certificate of OpenShift routes and adds it to the Java trust store of the CodeReady Workspaces server. Required when activating the **tlsSupport** field on demo OpenShift clusters that have not been setup with a valid certificate for the routes. |
| **serverMemoryLimit** | **1Gi** | Overrides the memory limit used in the CodeReady Workspaces server deployment. |
| **serverMemoryRequest** | **512Mi** | Overrides the memory request used in the CodeReady Workspaces server deployment. |

| Property | Default value | Description |
|---|---|---|
| **tlsSupport** | **false** | Instructs the Operator to deploy CodeReady Workspaces in TLS mode. Enabling TLS requires enabling the **selfSignedCert** field. |

### 6.5.2. database configuration settings related to the database used by CodeReady Workspaces

| Property | Default value | Description |
|---|---|---|
| **chePostgresDb** | **dbche** | PostgreSQL database name that the CodeReady Workspaces server uses to connect to the database. |
| **chePostgresHostName** | the Operator sets the value automatically | PostgreSQL Database host name that the CodeReady Workspaces server uses to connect to. Defaults to **postgres**. Override this value only when using an external database. (See the field **externalDb**.) |
| **chePostgresPassword** | auto-generated value | PostgreSQL password that the CodeReady Workspaces server uses to connect to the database. |
| **chePostgresPort** | **5432** | PostgreSQL Database port that the CodeReady Workspaces server uses to connect to. Override this value only when using an external database (see field **externalDb**). |
| **chePostgresUser** | **pgche** | PostgreSQL user that the CodeReady Workspaces server uses to connect to the database. |
| **externalDb** | **false** | Instructs the Operator to deploy a dedicated database. By default, a dedicated PostgreSQL database is deployed as part of the CodeReady Workspaces installation. If set to **true**, the Operator does not deploy a dedicated database automatically, you need to provide connection details to an external database. See all the fields starting with: **chePostgres**. |
| **postgresImagePullPolicy** | Always` for **nightly** or **latest** images, and **IfNotPresent** in other cases | Overrides the image pull policy used in the PostgreSQL database deployment. |
| **postgresImage** | omit | Overrides the container image used in the PostgreSQL database deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator. |

### 6.5.3. auth configuration settings related to authentication used by CodeReady Workspaces installation

| Property | Default value | Description |
|---|---|---|
| **externalIdentityProvider** | **false** | By default, a dedicated Identity Provider server is deployed as part of the CodeReady Workspaces installation. But if **externalIdentityProvider** is **true**, then no dedicated identity provider will be deployed by the Operator and you might need to provide details about the external identity provider you want to use. See also all the other fields starting with: **identityProvider**. |
| **identityProviderAdminUserName** | **admin** | Overrides the name of the Identity Provider admin user. |
| **identityProviderClientId** | omit | Name of an Identity provider (Keycloak / RH SSO) **client-id** that must be used for CodeReady Workspaces. This is useful to override it ONLY if you use an external Identity Provider (see the **externalIdentityProvider** field). If omitted or left blank, it will be set to the value of the **flavor** field suffixed with **-public**. |
| **identityProviderImagePullPolicy** | **Always** for **nightly** or **latest** images, and **IfNotPresent** in other cases | Overrides the image pull policy used in the Identity Provider (Keycloak / RH SSO) deployment. |
| **identityProviderImage** | omit | Overrides the container image used in the Identity Provider (Keycloak / RH SSO) deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator. |
| **identityProviderPassword** | omit | Overrides the password of Keycloak admin user. Override it only when using an external Identity Provider (see the **externalIdentityProvider** field). Omit or leave empty to set an auto-generated password. |
| **identityProviderPostgresPassword** | the Operator sets the value automatically | Password for The Identity Provider (Keycloak / RH SSO) to connect to the database. This is useful to override it ONLY if you use an external Identity Provider (see the **externalIdentityProvider** field). |
| **identityProviderRealm** | omit | Name of an Identity provider (Keycloak / RH SSO) realm. Override it only when using an external Identity Provider (see the **externalIdentityProvider** field). Omit or leave empty blank to set it to the value of the **flavor** field. |
| **identityProviderURL** | the Operator sets the value automatically | Instructs the Operator to deploy a dedicated Identity Provider (Keycloak or RH SSO instance). Public URL of the Identity Provider server (Keycloak / RH SSO server). Set it only when using an external Identity Provider (see the **externalIdentityProvider** field). |

| Property | Default value | Description |
|---|---|---|
| **oAuthClientName** | the Operator sets the value automatically | Name of the OpenShift **OAuthClient** resource used to setup identity federation on the OpenShift side. See also the **OpenShiftoAuth** field. |
| **oAuthSecret** | the Operator sets the value automatically | Name of the secret set in the OpenShift **OAuthClient** resource used to setup identity federation on the OpenShift side. See also the **OAuthClientName** field. |
| **openShiftoAuth** | **true** on OpenShift | Enables the integration of the identity provider (Keycloak / RHSSO) with OpenShift OAuth. This allows users to log in with their OpenShift login and have their workspaces created under personal OpenShift namespaces. The **kubeadmin** user is not supported, and logging through does not allow access to the CodeReady Workspaces Dashboard. |
| **updateAdminPassword** | **false** | Forces the default **admin** CodeReady Workspaces user to update password on first login. |

### 6.5.4. storage configuration settings related to persistent storage used by CodeReady Workspaces

| Property | Default value | Description |
|---|---|---|
| **postgresPVCStorageClassName** | omit | Storage class for the Persistent Volume Claim dedicated to the PostgreSQL database. Omitted or leave empty to use a default storage class. |
| **preCreateSubPaths** | **false** | Instructs the CodeReady Workspaces server to launch a special Pod to pre-create a subpath in the Persistent Volumes. Enable it according to the configuration of your K8S cluster. |
| **pvcClaimSize** | **1Gi** | Size of the persistent volume claim for workspaces. |
| **pvcJobsImage** | omit | Overrides the container image used to create sub-paths in the Persistent Volumes. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator. See also the **preCreateSubPaths** field. |
| **pvcStrategy** | **common** | Available options:`common` (all workspaces PVCs in one volume), **per-workspace** (one PVC per workspace for all declared volumes) and **unique** (one PVC per declared volume). |
| **workspacePVCStorageClassName** | omit | Storage class for the Persistent Volume Claims dedicated to the workspaces. Omit or leave empty to use a default storage class. |

### 6.5.5. k8s configuration settings specific to CodeReady Workspaces installations on OpenShift

| Property | Default value | Description |
|---|---|---|
| **ingressClass** | **nginx** | Ingress class that defines which controller manages ingresses. |
| **ingressDomain** | omit | Global ingress domain for a K8S cluster. This field must be explicitly specified. This drives the **is kubernetes.io/ingress.class** annotation on CodeReady Workspaces-related ingresses. |
| **ingressStrategy** | **multi-host** | Strategy for ingress creation. This can be **multi-host** (host is explicitly provided in ingress) and **default-host**(no host is provided, path-based rules). |
| **securityContext FsGroup,omitempty** | **1724** | FSGroup the CodeReady Workspaces Pod and Workspace Pods containers will run in. |
| **securityContext RunAsUser** | **1724** | ID of the user the CodeReady Workspaces Pod and Workspace Pods containers will run as. |
| **tlsSecretName** | omit | Name of a secret that is used to set ingress TLS termination if TLS is enabled. See also the **tlsSupport** field. |

### 6.5.6. installation defines the observed state of CodeReady Workspaces installation

| Property | Description |
|---|---|
| **cheClusterRunning** | Status of a CodeReady Workspaces installation. Can be **Available**, **Unavailable**, or **Available, Rolling Update in Progress**. |
| **cheURL** | Public URL to the CodeReady Workspaces server. |
| **cheVersion** | Currently installed CodeReady Workspaces version. |
| **dbProvisioned** | Indicates whether a PostgreSQL instance has been correctly provisioned. |
| **devfileRegistryURL** | Public URL to the Devfile registry. |
| **helpLink** | A URL to where to find help related to the current Operator status. |
| **keycloakProvisioned** | Indicates whether an Identity Provider instance (Keycloak / RH SSO) has been provisioned with realm, client and user. |
| **keycloakURL** | Public URL to the Identity Provider server (Keycloak / RH SSO). |

| Property | Description |
|---|---|
| **message** | A human-readable message with details about why the Pod is in this state. |
| **openShiftoAuthProvisioned** | Indicates whether an Identity Provider instance (Keycloak / RH SSO) has been configured to integrate with the OpenShift OAuth. |
| **pluginRegistryURL** | Public URL to the Plugin registry. |
| **reason** | A brief CamelCase message with details about why the Pod is in this state. |

### 6.5.7. Limits for workspaces

| Property | Default value | Description |
|---|---|---|
| **che.limits.workspace.env.ram** | **16gb** | The maximum amount of RAM that a user can allocate to a workspace when they create a new workspace. The RAM slider is adjusted to this maximum value. |
| **che.limits.workspace.idle.timeout** | **1800000** | The length of time that a user is idle with their workspace when the system will suspend the workspace and then stopping it. Idleness is the length of time that the user has not interacted with the workspace, meaning that one of our agents has not received interaction. Leaving a browser window open counts toward idleness. |

### 6.5.8. Limits for the workspaces of an user

| Property | Default value | Description |
|---|---|---|
| **che.limits.user.workspaces.ram** | **16gb** | he total amount of RAM that a single user is allowed to allocate to running workspaces. A user can allocate this RAM to a single workspace or spread it across multiple workspaces. |
| **che.limits.user.workspaces.count** | **1800000** | The maximum number of workspaces that a user is allowed to create. The user will be presented with an error message if they try to create additional workspaces. This applies to the total number of both running and stopped workspaces. |
| **che.limits.user.workspaces.run.count** | **1** | The maximum number of running workspaces that a single user is allowed to have. If the user has reached this threshold and they try to start an additional workspace, they will be prompted with an error message. The user will need to stop a running workspace to activate another. |

### 6.5.9. Limits for for the workspaces of an organization

| Property | Default value | Description |
| --- | --- | --- |
| **che.limits.organization.workspaces.ram** | **-1** | The total amount of RAM that a single organization (team) is allowed to allocate to running workspaces. An organization owner can allocate this RAM however they see fit across the team's workspaces. |
| **che.limits.organization.workspaces.count** | **-1** | The maximum number of workspaces that a organization is allowed to own. The organization will be presented an error message if they try to create additional workspaces. This applies to the total number of both running and stopped workspaces. |
| **che.limits.organization.workspaces.run.count** | **-1** | The maximum number of running workspaces that a single organization is allowed. If the organization has reached this threshold and they try to start an additional workspace, they will be prompted with an error message. The organization will need to stop a running workspace to activate another. |

# CHAPTER 7. UNINSTALLING CODEREADY WORKSPACES

This section describes uninstallation procedures for Red Hat CodeReady Workspaces installed on OpenShift. The uninstallation process leads to a complete removal of CodeReady Workspaces-related user data. The appropriate uninstallation method depends on what method was used to install the CodeReady Workspaces instance.

- For CodeReady Workspaces installed using OperatorHub, see Section 7.1, "Uninstalling CodeReady Workspaces after OperatorHub installation".
- For CodeReady Workspaces installed using crwctl, see Section 7.2, "Uninstalling CodeReady Workspaces after crwctl installation"

## 7.1. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION

Users have two options for uninstalling CodeReady Workspaces from an OpenShift cluster. The following sections describe the following methods:

- Using the OpenShift Administrator Perspective web UI
- Using **oc** commands from the terminal

### 7.1.1. Uninstalling CodeReady Workspaces using the OpenShift web console

This section describes how to uninstall CodeReady Workspaces from a cluster using the OpenShift Administrator Perspective main menu.

**Prerequisites**

- CodeReady Workspaces was installed on an OpenShift cluster using OperatorHub.

**Procedure: deleting the CodeReady Workspaces deployment**

1. Open the OpenShift web console.
2. Navigate to the **Operators > Installed Operators** section.
3. Click **Red Hat CodeReady Workspaces** in the list of installed operators.
4. Navigate to the **Red Hat CodeReady Workspaces Cluster** tab.
5. In the row that displays information about the specific CodeReady Workspaces cluster, delete the CodeReady Workspaces Cluster deployment using the drop-down menu illustrated as three horizontal dots situated on the right side of the screen.
6. Alternatively, delete the CodeReady Workspaces deployment by clicking the displayed Red Hat CodeReady Workspaces Cluster, **red-hat-codeready-workspaces**, and select the **Delete cluster** option in the **Actions** drop-down menu on the top right.

**Procedure: deleting the CodeReady Workspaces Operator**

1. Open the OpenShift web console.
2. Navigate to the **Operators > Installed Operators** section in OpenShift Developer Perspective.

3. In the row that displays information about the specific Red Hat CodeReady Workspaces Operator, uninstall the CodeReady Workspaces Operator using the drop-down menu illustrated as three horizontal dots situated on the right side of the screen.

4. Accept the selected option, **Also completely remove the Operator from the selected namespace**.

5. Alternatively, uninstall the Red Hat CodeReady Workspaces Operator by clicking the displayed Red Hat CodeReady Workspaces Operator, **Red Hat CodeReady Workspaces**, followed by selecting the **Uninstall Operator** option in the **Actions** drop-down menu on the top right.

### 7.1.2. Uninstalling CodeReady Workspaces using `oc` commands

This section provides instructions on how to uninstall a CodeReady Workspaces instance using **oc** commands.

#### Prerequisites

- CodeReady Workspaces was installed on an OpenShift cluster using OperatorHub.
- OpenShift command-line tools (**oc**) are installed on the local workstation.

#### Procedure

The following procedure provides command-line outputs as examples. Note that output in the user terminal may differ.

To uninstall a CodeReady Workspaces instance from a cluster:

1. Sign in to the cluster:

```
$ oc login -u <username> -p <password> <cluster_URL>
```

2. Switch to the project where the CodeReady Workspaces instance is deployed:

```
$ oc project <codeready-workspaces_project>
```

3. Obtain the CodeReady Workspaces cluster name. The following shows a cluster named **red-hat-codeready-workspaces**:

```
$ oc get checluster
NAME          AGE
red-hat-codeready-workspaces   27m
```

4. Delete the CodeReady Workspaces cluster:

```
$ oc delete checluster red-hat-codeready-workspaces
checluster.org.eclipse.che "red-hat-codeready-workspaces" deleted
```

5. Obtain the name of the CodeReady Workspaces cluster service version (CSV) module. The following detects a CSV module named **red-hat-codeready-workspaces.v2.1**:

```
$ oc get csv
NAME              DISPLAY      VERSION   REPLACES          PHASE
```

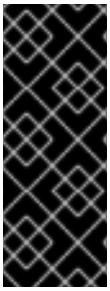
```
red-hat-codeready-workspaces.v2.1   Red Hat CodeReady Workspaces   2.1      red-hat-
codeready-workspaces.v2.0   Succeeded
```

6. Delete the CodeReady Workspaces CSV:

```
$ oc delete csv red-hat-codeready-workspaces.v2.1
clusterserviceversion.operators.coreos.com "red-hat-codeready-workspaces.v2.1" deleted
```

## 7.2. UNINSTALLING CODEREADY WORKSPACES AFTER CRWCTL INSTALLATION

This section describes how to uninstall an instance of Red Hat CodeReady Workspaces that was installed using the **crwctl** tool.



IMPORTANT

- For CodeReady Workspaces installed using the **crwctl server:start** command and the **-n** argument (custom namespace specified), use the **-n** argument also to uninstall the CodeReady Workspaces instance.
- For installations that did not use the **-n** argument, the created namespace is named workspaces by default.

**Prerequisites**

- CodeReady Workspaces was installed on an OpenShift cluster using **crwctl**.
- OpenShift command-line tools (**oc**) and **crwctl** are installed on the local workstation.
- The user is logged in a CodeReady Workspaces cluster using **oc**.

**Procedure**

1. Stop the Red Hat CodeReady Workspaces Server:

```
$ crwctl server:stop
```

2. Obtain the name of the CodeReady Workspaces namespace:

```
$ oc get checluster --all-namespaces -o=jsonpath="{.items[*].metadata.namespace}"
```

3. Remove CodeReady Workspaces from the cluster:

```
$ crwctl server:delete -n <namespace>
```

This removes all CodeReady Workspaces installations from the cluster.