Red Hat CodeReady Workspaces 2.4

Installation Guide

Installing Red Hat CodeReady Workspaces 2.4

Last Updated: 2020-12-18

Red Hat CodeReady Workspaces 2.4 Installation Guide

Installing Red Hat CodeReady Workspaces 2.4

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. SUPPORTED PLATFORMS

This section describes the availability and the supported installation methods of CodeReady Workspaces 2.4 on OpenShift Container Platform and OpenShift Dedicated.

The minimal OpenShift Container Platform version supporting Red Hat CodeReady Workspaces is OpenShift Container Platform 3.11.

Table 1.1. Supported deployment environments for CodeReady Workspaces 2.4 on OpenShift Container Platform and OpenShift Dedicated

Platform	Architecture	Deployment method
OpenShift Container Platform 3.11	AMD64 and Intel 64 (x86_64)	**crwctl**
OpenShift Container Platform 4.4	AMD64 and Intel 64 (x86_64)	OperatorHub
OpenShift Container Platform 4.4	IBM Z (s390x)	OperatorHub
OpenShift Container Platform 4.5	AMD64 and Intel 64 (x86_64)	OperatorHub
OpenShift Dedicated 4.3	AMD64 and Intel 64 (x86_64)	Add-On

NOTE

- On OpenShift Container Platform 4.4 and 4.5, when the OperatorHub installation method is not available, consider using **crwctl** as an unofficial backup installation method.

CHAPTER 2. CONFIGURING THE CODEREADY WORKSPACES INSTALLATION

The following section describes configuration options to install Red Hat CodeReady Workspaces using the Operator.

2.1. UNDERSTANDING THE CHECLUSTER CUSTOM RESOURCE

A default deployment of CodeReady Workspaces consist in the application of a parametrized **CheCluster** Custom Resource by the Red Hat CodeReady Workspaces Operator.

CheCluster Custom Resource

- A YAML document describing the configuration of the overall CodeReady Workspaces installation.
- Contains sections to configure each component: **auth**, **database**, **server**, **storage**.

Role of the Red Hat CodeReady Workspaces Operator

- To translate the **CheCluster** Custom Resource into configuration (ConfigMap) usable by each component of the CodeReady Workspaces installation.

Role of the OpenShift platform

- To apply the configuration (ConfigMap) for each component.
- To create the necessary Pods.
- When OpenShift detects a change in the configuration of a component, it restarts the Pods accordingly.

> **Example 2.1. Configuring the main properties of the CodeReady Workspaces server component**
>
> 1. The user applies a **CheCluster** Custom Resource containing some configuration related to the **server**.
> 2. The Operator generates a necessary ConfigMap, called **che**.
> 3. OpenShift detects change in the ConfigMap and triggers a restart of the CodeReady Workspaces Pod.

Additional resources

- Understanding Operators.
- Understanding Custom Resources.
- To learn how to modify the **CheCluster** Custom Resource, see the chosen installation procedure.

2.2. CHECLUSTER CUSTOM RESOURCE FIELDS REFERENCE

This section describes all fields available to customize the **CheCluster** Custom Resource.

- Example 2.2, "A minimal **CheCluster** Custom Resource example."
- Table 2.3, "**CheCluster** Custom Resource **auth** configuration settings related to authentication used by CodeReady Workspaces installation"
- Table 2.2, "**CheCluster** Custom Resource **database** configuration settings related to the database used by CodeReady Workspaces"
- Table 2.1, "**CheCluster** Custom Resource **server** settings, related to the CodeReady Workspaces server component."
- Table 2.4, "**CheCluster** Custom Resource **storage** configuration settings related to persistent storage used by CodeReady Workspaces"
- Table 2.5, "**CheCluster** Custom Resource **k8s** configuration settings specific to CodeReady Workspaces installations on OpenShift"
- Table 2.6, "**CheCluster** Custom Resource **status** defines the observed state of CodeReady Workspaces installation"

Example 2.2. A minimal CheCluster Custom Resource example.

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: codeready-workspaces
spec:
  auth:
    externalIdentityProvider: false
  database:
    externalDb: false
  server:
    selfSignedCert: false
    gitSelfSignedCert: false
    tlsSupport: true
  storage:
    pvcStrategy: 'common'
    pvcClaimSize: '1Gi'
```

Table 2.1. CheCluster Custom Resource server settings, related to the CodeReady Workspaces server component.

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.

Property	Default value	Description
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 CodeReady Workspaces workspaces. Omit or leave empty to use the default roles.
customCheProperties	omit	Map of additional environment variables that will be applied in the generated **codeready-workspaces** ConfigMap 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** ConfigMap from other CR fields, then the value defined in the **customCheProperties** will be used instead.

Property	Default value	Description
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).
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.

Property	Default value	Description
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).
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.
tlsSupport	**true**	Instructs the Operator to deploy CodeReady Workspaces in TLS mode.

Table 2.2. **CheCluster** Custom Resource**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.

Property	Default value	Description
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.

Table 2.3. **CheCluster** Custom Resource**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.

Property	Default value	Description
identityProviderClientId	omit	Name of an Identity provider (RH-SSO / 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 (RH-SSO / RH SSO) deployment.
identityProviderImage	omit	Overrides the container image used in the Identity Provider (RH-SSO / 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 RH-SSO 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 (RH-SSO / 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 (RH-SSO / 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 (RH-SSO or RH SSO instance). Public URL of the Identity Provider server (RH-SSO / RH SSO server). Set it only when using an external Identity Provider (see the **externalIdentityProvider** field).
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.

Property	Default value	Description
openShiftoAuth	**true** on OpenShift	Enables the integration of the identity provider (RH-SSO / RHSSO) with OpenShift OAuth. This allows users to log in with their OpenShift login and have their workspaces created under personal OpenShift projects. 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.

Table 2.4. **CheCluster** Custom Resource**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 CodeReady Workspaces workspaces. Omit or leave empty to use a default storage class.

Table 2.5. **CheCluster** Custom Resource**k8s** configuration settings specific to CodeReady Workspaces installations on OpenShift

Property	Default value	Description
ingressClass	**nginx**	Ingress class that defines which controller manages ingresses.

Property	Default value	Description
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), **single-host** (host is provided, path-based rules) and **default-host.***(no host is provided, path-based rules).
securityContext FsGroup,omitempty	**1724**	FSGroup the CodeReady Workspaces Pod and Workspace Pods containers run in.
securityContext RunAsUser	**1724**	ID of the user the CodeReady Workspaces Pod and Workspace Pods containers 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.

Table 2.6. **CheCluster** Custom Resource**status** 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 (RH-SSO / RH SSO) has been provisioned with realm, client and user.
keycloakURL	Public URL to the Identity Provider server (RH-SSO / RH SSO).
message	A human-readable message with details about why the Pod is in this state.

Property	Description
openShiftoAuthProvisioned	Indicates whether an Identity Provider instance (RH-SSO / 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.

CHAPTER 3. INSTALLING CODEREADY WORKSPACES

This section contains instructions to install Red Hat CodeReady Workspaces. The installation method depends on the target platform and the environment restrictions.

3.1. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT 4 USING OPERATORHUB

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

Operators are a method of packaging, deploying, and managing an 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 independent software vendor (ISV) content.
- A place to encapsulate knowledge from field engineers and spread it to all users.

Prerequisites

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

3.1.1. Creating a project in OpenShift Web Console

A project allows to organize and manage different resources on the cluster in an isolated unit. Create a project first to host the Red Hat CodeReady Workspaces Operator.

Procedure

1. Open the OpenShift web console, in the left panel navigate to the **Home → Projects** section.
2. Click **Create Project**.
3. Specify the project details:
   - Name: **openshift-workspaces**
   - Display Name: **Red Hat CodeReady Workspaces**
   - Description: **Red Hat CodeReady Workspaces**

3.1.2. Installing the Red Hat CodeReady Workspaces Operator

Red Hat CodeReady Workspaces Operator provides all the resources for running CodeReady Workspaces, such as PostgreSQL, RH-SSO, image registries, and the CodeReady Workspaces server, and also configures all these services.

Prerequisites

- Access to the Web Console on the cluster.

Procedure

1. To install the Red Hat CodeReady Workspaces Operator, in the left panel, navigate to the **Operators → OperatorHub** section.
2. In the **Filter by keyword** field, type **Red Hat CodeReady Workspaces** and click the **Red Hat CodeReady Workspaces** tile.
3. In the **Red Hat CodeReady Workspaces** pop-up window, click the **Install** button.
4. On the **Install Operator** screen, specify the following options:
   - **Installation mode**: **A specific project on the cluster**
   - **Installed Namespace**: *Pick an existing project → **openshift-workspaces**

Verification steps

1. To verify the Red Hat CodeReady Workspaces Operator has installed correctly, in the left panel navigate to the **Operators → Installed Operators** section.
2. In the **Installed Operators** screen, click the **Red Hat CodeReady Workspaces** name and navigate to the **Details** tab.
3. In the **ClusterServiceVersion Details** section at the bottom of the page, wait for these messages:
   - Status: **Succeeded**
   - Status Reason: **install strategy completed with no errors**
4. Navigate to the **Events** tab and wait for this message: **install strategy completed with no errors**.

3.1.3. Creating an instance of the Red Hat CodeReady Workspaces Operator

Follow this procedure to install Red Hat CodeReady Workspaces with the default configuration. To modify the configuration, see Chapter 2, *Configuring the CodeReady Workspaces installation* .

Procedure

1. To create an instance of the Red Hat CodeReady Workspaces Operator, in the left panel, navigate to the **Operators → Installed Operators** section.
2. In the **Installed Operators** screen, click the **Red Hat CodeReady Workspaces** name.
3. In the **Operator Details** screen, in the **Details** tab, inside of the **Provided APIs** section, click the **Create Instance** link.
4. The **Create CheCluster** page contains the configuration of the overall CodeReady Workspaces instance to create. It is the **CheCluster** Custom Resource. Keep the default values.
5. To create the **codeready-workspaces** cluster, click the **Create** button in the lower left corner of the window.

6. On the **Operator Details** screen, in the **Red Hat CodeReady Workspaces Cluster** tab, click on the **codeready-workspaces** link.

7. To navigate to the **codeready-workspaces** instance, click the link under **Red Hat CodeReady Workspaces URL**.

NOTE

The installation might take more than 5 minutes. The URL appears after the Red Hat CodeReady Workspaces installation finishes.

Verification steps

1. To verify that the Red Hat CodeReady Workspaces instance has installed correctly, navigate to the **CodeReady Workspaces Cluster** tab. The **CheClusters** screen displays the list of Red Hat CodeReady Workspaces instances and their status.

2. Click **codeready-workspaces CheCluster** in the table and navigate to the **Details** tab.

3. See the content of following fields:

   - **Message**: the field contains error messages, if any. The expected content is **None**.
   - **Red Hat CodeReady Workspaces URL**: displays the URL of the Red Hat CodeReady Workspaces instance, once the deployment is successful.

4. Navigate to the **Resources** tab. The screen displays the list of the resources assigned to the CodeReady Workspaces deployment.

5. To see more details about the state of a resource, click its name and inspect the content of the available tabs.

Additional resources

- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.4/html-single/end-user_guide/index#navigating-codeready-workspaces-using-the-dashboard_crw.
- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.4/html-single/administration_guide/index#viewing-operator-events.adoc.
- It is possible to use the **crwctl** utility script for deploying CodeReady Workspaces on OpenShift Container Platform and OpenShift Dedicated versions 4.5. This method is unofficial and serves as a backup installation method for situations where the installation method using OperatorHub is not available. See the Section 3.2.2, "Installing CodeReady Workspaces on OpenShift 3 using the Operator" section.

3.2. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT CONTAINER PLATFORM 3.11

3.2.1. 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 2.4.
3. Extract the archive to a folder, such as **${HOME}/crwctl** or **/opt/crwctl**.
4. Run the **crwctl** executable from the extracted folder. In this example, **${HOME}/crwctl/bin/crwctl version**.
5. Optionally, add the **bin** folder to your **$PATH**, for example, **PATH=${PATH}:${HOME}/crwctl/bin** to enable running **crwctl** without the full path specification.

Verification step

Running **crwctl version** displays the current version of the tool.

3.2.2. Installing CodeReady Workspaces on OpenShift 3 using the Operator

This section describes how to install CodeReady Workspaces on OpenShift 3 with the **crwctl** CLI management tool. The method of installation is using the Operator and enable TLS (HTTPS).

NOTE

Methods for updating from a previous CodeReady Workspaces installation and enabling multiple instances in the same OpenShift Container Platform 3.11 cluster are provided below the installation procedure.

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 independent software vendor (ISV) content.
- A place to encapsulate knowledge from field engineers and spread it to all users.

TIP

This approach is only supported for use with OpenShift Container Platform and OpenShift Dedicated version 3.11, but also work for newer versions of OpenShift Container Platform and OpenShift Dedicated, and serves as a backup installation method for situations when the installation method using OperatorHub is not available.

Prerequisites

- Administrator rights on a running instance of OpenShift 3.11.
- An installation of the **oc** OpenShift 3.11 CLI management tool. See Installing the OpenShift 3.11 CLI.
- An installation of the **crwctl** management tool. See Section 3.2.1, “Installing the crwctl CLI management tool”.

- To apply settings that the main crwctl command-line parameters cannot set, prepare a configuration file **operator-cr-patch.yaml** that will override the default values in the **CheCluster** Custom Resource used by the Operator. See Chapter 2, *Configuring the CodeReady Workspaces installation*.

- *<namespace>* represents the project of the target installation.

Procedure

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

```
$ oc login
```

2. Run the following command to verify that the version of the **oc** OpenShift CLI management tool is 3.11:

```
$ oc version
oc v3.11.0+0cbc58b
```

3. Run the following command to create the CodeReady Workspaces instance

   - In the user-defined *<namespace>*:

```
$ crwctl server:start -n <namespace> -p openshift
```

   - In the default project called openshift-workspaces:

```
$ crwctl server:start -p openshift
```

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>**.

Upgrading from a previous CodeReady Workspaces installation

- To upgrade from a previous CodeReady Workspaces installation in the same OpenShift Container Platform 3.11 cluster, remove the Custom Resource Definition and the Cluster Roles:

```
$ oc delete customresourcedefinition/checlusters.org.eclipse.che
$ oc patch customresourcedefinition/checlusters.org.eclipse.che \
    --type merge \
    -p '{ "metadata": { "finalizers": null }}'
$ oc delete clusterrole codeready-operator
```

Having multiple CodeReady Workspaces deployments

- To have multiple CodeReady Workspaces deployments in parallel using different versions in the same OpenShift Container Platform 3.11 cluster, create a new service account for the new

deployment. It is, however, strongly recommended that you update all your old CodeReady Workspaces deployments to the latest version instead, as this mix of versions may cause unexpected and unsupported results.

```
$ oc patch clusterrolebinding codeready-operator \
    --type='json' \
    -p '[{"op": "add", "path": "/subjects/0", "value": {"kind":"ServiceAccount", "namespace":
"<openshift-workspaces>", "name": "codeready-operator"} }]'
```

3.3. 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 3.3.3, “Preparing CodeReady Workspaces Custom Resource for installing behind a proxy”.

3.3.1. Installing CodeReady Workspaces in a restricted environment 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 3.1, “Installing CodeReady Workspaces on OpenShift 4 using OperatorHub”.

3.3.2. Installing CodeReady Workspaces in a restricted environment using CLI management tool

NOTE

Use CodeReady Workspaces CLI management tool to install CodeReady Workspaces on restricted networks if installation through OperatorHub is not available. This method is supported for OpenShift Container Platform 3.11.

Prerequisites

- A running OpenShift cluster. See the OpenShift Container Platform 3.11 documentation for instructions on how to install an OpenShift cluster.

3.3.2.1. Preparing a private registry

Prerequisites

- The **oc** tool is available.
- The **skopeo** tool, version 0.1.40 or later, is available.
- The **podman** tool is available.
- An image registry accessible from the OpenShift cluster and supporting the format of the V2 image manifest, schema version 2. Ensure you can push to it from a location having, at least temporarily, access to the internet.

Table 3.1. Placeholders used in examples

<source-image>	Full coordinates of the source image, including registry, organization, and digest.
<target-registry>	Host name and port of the target container-image registry.
<target-organization>	Organization in the target container-image registry
<target-image>	Image name and digest in the target container-image registry.
<target-user>	User name in the target container-image registry.
<target-password>	User password in the target container-image registry.

Procedure

1. Log into the internal image registry:

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

TIP

If you meet an error, such as **x509: certificate signed by unknown authority**, when attempting to push to the internal registry, try one of these workarounds:

- add the OpenShift cluster's certificate to **/etc/containers/certs.d/*<target-registry>***
- add the registry as an insecure registry by adding the following lines to the Podman configuration file located at **/etc/containers/registries.conf**:

```
[registries.insecure]
registries = ['<target-registry>']
```

2. Copy images without changing their digest. Repeat this step for every image in the following table:

```
$ skopeo copy --all docker://<source-image> \
  docker://<target-registry>/<target-organization>/<target-image>
```

NOTE

Table 3.2. Understanding the usage of the container-images from the prefix or keyword they include in their name

Usage	Prefix or keyword
Essential	not **stacks-**, **plugin-**, or **-openj-**
Workspaces	**stacks-**, **plugin-**
Z and Power	**-openj-**

Table 3.3. Images to copy in the private registry

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:89763ddec38a5925a052fa7ea75fc5a0db39124cada1e2d33b6eba3e32e8a7c6	**crw-2-rhel8-operator@sha256:89763ddec38a5925a052fa7ea75fc5a0db39124cada1e2d33b6eba3e32e8a7c6**
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:7702adb0ed28b635e45804e87fe5dd98bdd3aa766fed7845a8ce509b91c22e36	**devfileregistry-rhel8@sha256:7702adb0ed28b635e45804e87fe5dd98bdd3aa766fed7845a8ce509b91c22e36**
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:8afecd5b0edc7734532ee76ff9eac1fc4814d8aaa6c9be440a2a88a20c014e4e	**jwtproxy-rhel8@sha256:8afecd5b0edc7734532ee76ff9eac1fc4814d8aaa6c9be440a2a88a20c014e4e**
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:c9bebc895e5fa5a0bd4ecaedfd5384ab75a45a96b6314ba5d4a6f4c1e8e109f9	**machineexec-rhel8@sha256:c9bebc895e5fa5a0bd4ecaedfd5384ab75a45a96b6314ba5d4a6f4c1e8e109f9**
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:27a71612f9bd3bee77adb4e164c44c61cf5085458d592215b2fe74c55d11abc6	**plugin-java11-openj9-rhel8@sha256:27a71612f9bd3bee77adb4e164c44c61cf5085458d592215b2fe74c55d11abc6**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:e9deebbc320d28a2f425e858ed3dcf87fc67a40f6654d6eb7c2b6feea022b7d6	**plugin-java11-rhel8@sha256:e9deebbc320d28a2f425e858ed3dcf87fc67a40f6654d6eb7c2b6feea022b7d6**
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:14f2774e92b70d85280e506f81e2ea9a89c26490fd53a4421df8a694bd944d2d	**plugin-java8-openj9-rhel8@sha256:14f2774e92b70d85280e506f81e2ea9a89c26490fd53a4421df8a694bd944d2d**
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:d04f70c8340abaee1a282b77158d054f4faf2225bc17c79aafb413396c367782	**plugin-java8-rhel8@sha256:d04f70c8340abaee1a282b77158d054f4faf2225bc17c79aafb413396c367782**
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:d87aed64704369a50d1e54a57815b699f74d4efad1401d1a638808e655a37e48	**plugin-kubernetes-rhel8@sha256:d87aed64704369a50d1e54a57815b699f74d4efad1401d1a638808e655a37e48**
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:9c43a02b0dd0f66744359c5ccdb1f1780ecd92c3dc31b14d73b553ba763af8ab	**plugin-openshift-rhel8@sha256:9c43a02b0dd0f66744359c5ccdb1f1780ecd92c3dc31b14d73b553ba763af8ab**
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:d0eebf2c8b460adb75dc6bc5200aa9fd40d030b7b17c6b1c3b9d3c879f4652ee	**pluginbroker-artifacts-rhel8@sha256:d0eebf2c8b460adb75dc6bc5200aa9fd40d030b7b17c6b1c3b9d3c879f4652ee**
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:cff23432d1d397bbbc7df65be9d6ddf4a97a3ef1801708bb7bb7d2fa72dbcce3	**pluginbroker-metadata-rhel8@sha256:cff23432d1d397bbbc7df65be9d6ddf4a97a3ef1801708bb7bb7d2fa72dbcce3**
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:9f37917122c20fc83e6558a5484efab42650958b513a22920f449f948e50cd51	**pluginregistry-rhel8@sha256:9f37917122c20fc83e6558a5484efab42650958b513a22920f449f948e50cd51**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:63bf304cd04576048012693e7e8544a5a703790f99551554a75798bc799b112b	**server-rhel8@sha256:63bf304cd04576048012693e7e8544a5a703790f99551554a75798bc799b112b**
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:56543cfeeeac030821557ac4937db40f6845e874193c79c30267a680f9b2cbe7	**stacks-cpp-rhel8@sha256:56543cfeeeac030821557ac4937db40f6845e874193c79c30267a680f9b2cbe7**
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:13628110b96de0e516ff2dfa29cdcaee64cd8f8978052c8160c294c332dba9f0	**stacks-dotnet-rhel8@sha256:13628110b96de0e516ff2dfa29cdcaee64cd8f8978052c8160c294c332dba9f0**
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:fef91718ccebc4cd9b89999f6b5df83bf3d60fce657f6f44eda092100549af2c	**stacks-golang-rhel8@sha256:fef91718ccebc4cd9b89999f6b5df83bf3d60fce657f6f44eda092100549af2c**
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:b75f498954fbe858c74f80a89d132ba3560f40c0f697b0cd9550ed5663078ef6	**stacks-php-rhel8@sha256:b75f498954fbe858c74f80a89d132ba3560f40c0f697b0cd9550ed5663078ef6**
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:942e1e6328169508e3fff8fd96c575d7a423339ced17dbf5813d61d1971adaef	**theia-endpoint-rhel8@sha256:942e1e6328169508e3fff8fd96c575d7a423339ced17dbf5813d61d1971adaef**
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:78edc9f75680cbe7f63774d9dfbbc505401486a73c8e420380e1d3078bdf9f2a	**theia-rhel8@sha256:78edc9f75680cbe7f63774d9dfbbc505401486a73c8e420380e1d3078bdf9f2a**
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:d6a7bdbf4726fe0e0e54c0bce9b2257bbd2a165c37cb4ec68e1f994716ffb15c	**eap-xp1-openj9-11-openshift-rhel8@sha256:d6a7bdbf4726fe0e0e54c0bce9b2257bbd2a165c37cb4ec68e1f994716ffb15c**

<source-image>	*<target-image>*
registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833	**eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833**
registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0	**eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0**
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:8e6c7874247053df431c25552c6e2edb050b2627ae21907149f419e0d9909135	**sso74-openj9-openshift-rhel8@sha256:8e6c7874247053df431c25552c6e2edb050b2627ae21907149f419e0d9909135**
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:ec6801343eb1ca085154d8d7481552f2e9debc414125413d25e42216aa5922af	**sso74-openshift-rhel8@sha256:ec6801343eb1ca085154d8d7481552f2e9debc414125413d25e42216aa5922af**
registry.redhat.io/rhel8/postgresql-96@sha256:fdc2398a25530547354714f2538c691d91b700e0cedef5361a3e7d96ddfd4e11	**postgresql-96@sha256:fdc2398a25530547354714f2538c691d91b700e0cedef5361a3e7d96ddfd4e11**
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	**mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73**
registry.redhat.io/ubi8-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6	**ubi8-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6**

3. Verify the images have the same digests:

```
$ skopeo inspect docker://<source-image>
$ skopeo inspect docker://<target-registry>/<target-organization>/<target-image>
```

4. Set the digests explicitly when different:

```
$ skopeo copy --all docker://<source-image> \
  docker://<target-registry>/<target-organization>/<target-image>
```

Additional resources

- To find the sources of the images list, see the values of the **relatedImages** attribute in the CodeReady Workspaces Operator ClusterServiceVersion sources.

3.3.2.2. 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.

3.3.2.2.1. 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.

3.3.2.2.2. 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 3.3.2.1, "Preparing a private registry", 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: '<image-registry>'
    airGapContainerRegistryOrganization: '<organization>'
# [...]
```

3.3.2.3. 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. See Section 3.2.1, "Installing the crwctl CLI management tool".
- 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 a customized Custom Resource to add fields related to the restricted environment:

```
$ crwctl server:start \
  --che-operator-image=<image-registry>/<organization>/crw-2-rhel8-operator:2.4 \
  --che-operator-cr-yaml=org_v1_che_cr.yaml
```

NOTE

For slow systems or internet connections, add the **--k8spodwaittimeout=1800000** option to the **crwctl server:start** command to extend the Pod timeout period to 1800000 ms or longer.

3.3.3. Preparing CodeReady Workspaces Custom Resource for installing behind a proxy

This procedure describes how to provide necessary additional information to the **CheCluster** custom resource when installing CodeReady Workspaces behind 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 4. CONFIGURING CODEREADY WORKSPACES

The following chapter describes configuration methods and options for Red Hat CodeReady Workspaces, with some user stories as example.

- Section 4.1, “Advanced configuration options for the CodeReady Workspaces server component” describes advanced configuration methods to use when the previous method is not applicable.

The next sections describe some specific user stories.

- Section 4.2, “Configuring project strategies”
- Section 4.3, “Running more than one workspace at a time”
- Section 4.5, “Configuring workspaces nodeSelector”
- Section 4.6, “Configuring Red Hat CodeReady Workspaces server hostname”
- Section 4.7, “Deploying CodeReady Workspaces with support for Git repositories with self-signed certificates”
- Section 4.8, “Installing CodeReady Workspaces using storage classes”
- Section 4.9, “Configuring storage types”
- Section 4.10, “Importing TLS certificates to CodeReady Workspaces server Java truststore”

4.1. ADVANCED CONFIGURATION OPTIONS FOR THE CODEREADY WORKSPACES SERVER COMPONENT

The following section describes advanced deployment and configuration methods for the CodeReady Workspaces server component.

4.1.1. Understanding CodeReady Workspaces server advanced configuration using the Operator

The following section describes the CodeReady Workspaces server component advanced configuration method for a deployment using the Operator.

Advanced configuration is necessary to:

- Add environment variables not automatically generated by the Operator from the standard **CheCluster** Custom Resource fields.
- Override the properties automatically generated by the Operator from the standard **CheCluster** Custom Resource fields.

The **customCheProperties** field, part of the **CheCluster** Custom Resource **server** settings, contains a map of additional environment variables to apply to the CodeReady Workspaces server component.

> **Example 4.1. Override 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: codeready-workspaces
  namespace: <openshift-workspaces>
spec:
  server:
    cheImageTag: ''
    devfileRegistryImage: ''
    pluginRegistryImage: ''
    tlsSupport: true
    selfSignedCert: false
    customCheProperties:
      CHE_WORKSPACE_DEFAULT_MEMORY_LIMIT__MB: "2048"
  auth:
# [...]
```

NOTE

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**.

Additional resources

- For the list of all parameters available in the **CheCluster** Custom Resource, see Chapter 2, *Configuring the CodeReady Workspaces installation* .
- For the list of all parameters available to configure **customCheProperties**, see Section 4.1.2, "CodeReady Workspaces server component system properties reference".

4.1.2. CodeReady Workspaces server component system properties reference

The following document describes all possible configuration properties of the CodeReady Workspaces server component.

Table 4.1. Che server

Environment Variable Name	Default value	Description
CHE_DATABASE	**${che.home}/storage**	Folder where CodeReady Workspaces will store internal data objects
CHE_API	**http://${CHE_HOST}:${CHE_PORT}/api**	API service. Browsers initiate REST communications to CodeReady Workspaces server with this URL

Environment Variable Name	Default value	Description
CHE_WEBSOCKET_ENDPOINT	**ws://${CHE_HOST}:${CHE_PORT}/api/websocket**	CodeReady Workspaces websocket major endpoint. Provides basic communication endpoint for major websocket interaction/messaging.
CHE_WEBSOCKET_ENDPOINT__MINOR	**ws://${CHE_HOST}:${CHE_PORT}/api/websocket-minor**	CodeReady Workspaces websocket minor endpoint. Provides basic communication endpoint for minor websocket interaction/messaging.
CHE_WORKSPACE_STORAGE	**${che.home}/workspaces**	Your projects are synchronized from the CodeReady Workspaces server into the machine running each workspace. This is the directory in the ws runtime where your projects are mounted.
CHE_WORKSPACE_PROJECTS_STORAGE	**/projects**	Your projects are synchronized from the CodeReady Workspaces server into the machine running each workspace. This is the directory in the machine where your projects are placed.
CHE_WORKSPACE_PROJECTS_STORAGE_DEFAULT_SIZE	**1Gi**	Used when devfile OpenShift/os type components requests project PVC creation (applied in case of unique and perWorkspace PVC strategy. In case of common PVC strategy, it will be rewritten with value of che.infra.kubernetes.pvc.quantity property)
CHE_WORKSPACE_LOGS_ROOT__DIR	**/workspace_logs**	Defines the directory inside the machine where all the workspace logs are placed. The value of this folder should be provided into machine e.g. like environment variable so agents developers can use this directory for backup agents logs.
CHE_WORKSPACE_HTTP__PROXY		Configures proxies used by runtimes powering workspaces
CHE_WORKSPACE_HTTPS__PROXY		Configuresproxies used by runtimes powering workspaces

Environment Variable Name	Default value	Description
CHE_WORKSPACE_NO__PROXY		Configuresproxiesused by runtimes powering workspaces
CHE_TRUSTED__CA__BUNDLES__CONFIGMAP	**NULL**	When cluster wide proxy is configured, che-operator creates special configmap and allows OpenShift Network operator to inject ca-bundle into it. In addition, it adds the key CHE_TRUSTEDCABUNDLES__CONFIGMAP with name of this configmap into CodeReady Workspaces server configmap (and corresponding ENV variable). So by its presence we can detect if proxy mode is enabled or not. This property is not supposed to be set manually unless that specifically required.
CHE_WORKSPACE_AUTO__START	**true**	By default, when users access to a workspace with its URL the workspace automatically starts if it is stopped. You can set this to false to disable this.
CHE_WORKSPACE_POOL_TYPE	**fixed**	Workspace threads pool configuration, this pool is used for workspace related operations that require asynchronous execution e.g. starting/stopping. Possible values are 'fixed', 'cached'
CHE_WORKSPACE_POOL_EXACT__SIZE	**30**	This property is ignored when pool type is different from 'fixed'. Configures the exact size of the pool, if it's set multiplier property is ignored. If this property is not set(0, < 0, NULL) then pool sized to number of cores, it can be modified within multiplier
CHE_WORKSPACE_POOL_CORES__MULTIPLIER	**2**	This property is ignored when pool type is different from 'fixed' or exact pool size is set. If it's set the pool size will be N_CORES * multiplier

Environment Variable Name	Default value	Description
CHE_WORKSPACE_PROBE__POOL__SIZE	**10**	This property specifies how much threads to use for workspaces servers liveness probes
CHE_WORKSPACE_HTTP__PROXY__JAVA__OPTIONS	**NULL**	Http proxy setting for workspace JVM
CHE_WORKSPACE_JAVA__OPTIONS	**-XX:MaxRAM=150m-XX:MaxRAMFraction=2 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip.disableMemoryMapping=true -Xms20m -Djava.security.egd=file:/dev/./urandom**	Java command line options to be added to JVM's that running within workspaces.
CHE_WORKSPACE_MAVEN__OPTIONS	**-XX:MaxRAM=150m-XX:MaxRAMFraction=2 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip.disableMemoryMapping=true -Xms20m -Djava.security.egd=file:/dev/./urandom**	Maven command line options added to JVM's that run agents within workspaces.
CHE_WORKSPACE_MAVEN__SERVER__JAVA__OPTIONS	**-XX:MaxRAM=128m-XX:MaxRAMFraction=1 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip.disableMemoryMapping=true -Xms20m -Djava.security.egd=file:/dev/./urandom**	Default java command line options to be added to JVM that run maven server.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_DEFAULT__MEMORY__LIMIT__MB	**1024**	RAM limit default for each machine that has no RAM settings in environment. Value less or equal to 0 interpreted as limit disabling.
CHE_WORKSPACE_DEFAULT__MEMORY__REQUEST__MB	**200**	RAM request default for each container that has no explicit RAM settings in environment. this amount will be allocated on workspace container creation this property might not be supported by all infrastructure implementations: currently it is supported by OpenShift and OpenShift Container Platform if default memory request is more than the memory limit, request will be ignored, and only limit will be used. Value less or equal to 0 interpreted as disabling request.
CHE_WORKSPACE_DEFAULT__CPU__LIMIT__CORES	**-1**	CPU limit default for each container that has no CPU settings in environment. Can be specified either in floating point cores number, e.g. 0.125 or in K8S format integer millicores e.g. 125m Value less or equal to 0 interpreted as limit disabling.
CHE_WORKSPACE_DEFAULT__CPU__REQUEST__CORES	**-1**	CPU request default for each container that has no CPU settings in environment. if default CPU request is more than the CPU limit, request will be ignored, and only limit will be used. Value less or equal to 0 interpreted as disabling this request.
CHE_WORKSPACE_SIDECAR_DEFAULT__MEMORY__LIMIT__MB	**128**	RAM limit and request default for each sidecar that has no RAM settings in CodeReady Workspaces plugin configuration. Value less or equal to 0 interpreted as limit disabling.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_SIDECAR_DEFAULT__MEMORY__REQUEST__MB	**64**	RAMlimit and request default for each sidecar that has no RAM settings in **{prod-short}** plugin configuration. Value less or equal to 0 interpreted as limit disabling.
CHE_WORKSPACE_SIDECAR_DEFAULT__CPU__LIMIT__CORES	**-1**	CPU limit and request default for each sidecar that has no CPU settings in CodeReady Workspaces plugin configuration. Can be specified either in floating point cores number, e.g. 0.125 or in K8S format integer millicores e.g. 125m Value less or equal to 0 interpreted as disabling limit.
CHE_WORKSPACE_SIDECAR_DEFAULT__CPU__REQUEST__CORES	**-1**	CPUlimit and request default for each sidecar that has no CPU settings in **{prod-short}** plugin configuration. Can be specified either in floating point cores number, e.g. 0.125 or in K8S format integer millicores e.g. 125m Value less or equal to 0 interpreted as disabling limit.
CHE_WORKSPACE_SIDECAR_IMAGE__PULL__POLICY	**Always**	Define image pulling strategy for sidecars. Possible values are: Always, Never, IfNotPresent. Any other value will be interpreted as unspecified policy (Always if :latest tag is specified, or IfNotPresent otherwise.)
CHE_WORKSPACE_ACTIVITY__CHECK__SCHEDULER__PERIOD__S	**60**	Period of inactive workspaces suspend job execution.
CHE_WORKSPACE_ACTIVITY__CLEANUP__SCHEDULER__PERIOD__S	**3600**	The period of the cleanup of the activity table. The activity table can contain invalid or stale data if some unforeseen errors happen, like a server crash at a peculiar point in time. The default is to run the cleanup job every hour.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_ACTIVITY__CLEANUP__SCHEDULER__INITIAL__DELAY__S	**60**	The delay after server startup to start the first activity clean up job.
CHE_WORKSPACE_ACTIVITY__CHECK__SCHEDULER__DELAY__S	**180**	Delay before first workspace idleness check job started to avoid mass suspend if ws master was unavailable for period close to inactivity timeout.
CHE_WORKSPACE_CLEANUP__TEMPORARY__INITIAL__DELAY__MIN	**5**	Period of stopped temporary workspaces cleanup job execution.
CHE_WORKSPACE_CLEANUP__TEMPORARY__PERIOD__MIN	**180**	Periodof stopped temporary workspaces cleanup job execution.
CHE_WORKSPACE_SERVER_PING__SUCCESS__THRESHOLD	**1**	Number of sequential successful pings to server after which it is treated as available. Note: the property is common for all servers e.g. workspace agent, terminal, exec etc.
CHE_WORKSPACE_SERVER_PING__INTERVAL__MILLISECONDS	**3000**	Interval, in milliseconds, between successive pings to workspace server.
CHE_WORKSPACE_SERVER_LIVENESS__PROBES	**wsagent/http,exec-agent/http,terminal,theia,jupyter,dirigible,cloud-shell**	List of servers names which require liveness probes
CHE_WORKSPACE_STARTUP__DEBUG__LOG__LIMIT__BYTES	**10485760**	Limit size of the logs collected from single container that can be observed by che-server when debugging workspace startup. default 10MB=10485760
CHE_WORKSPACE_STOP_ROLE_ENABLED	**true**	If true, 'stop-workspace' role with the edit privileges will be granted to the 'che' ServiceAccount if OpenShift OAuth is enabled. This configuration is mainly required for workspace idling when the OpenShift OAuth is enabled.

Table 4.2. Templates

Environment Variable Name	Default value	Description
CHE_TEMPLATE_STORAGE	**${che.home}/templates**	Folder that contains JSON files with code templates and samples

Table 4.3. Authentication parameters

Environment Variable Name	Default value	Description
CHE_AUTH_USER__SELF__CREATION	**false**	CodeReady Workspaces has a single identity implementation, so this does not change the user experience. If true, enables user creation at API level
CHE_AUTH_ACCESS__DENIED__ERROR__PAGE	**/error-oauth**	Authentication error page address
CHE_AUTH_RESERVED__USER__NAMES		Reserved user names
CHE_OAUTH_GITHUB_CLIENTID	**NULL**	You can setup GitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_GITHUB_CLIENTSECRET	**NULL**	Youcan setup GitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_GITHUB_AUTHURI	**https://github.com/login/oauth/authorize**	Youcansetup GitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_GITHUB_TOKENURI	**https://github.com/login/oauth/access_token**	YoucansetupGitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.

Environment Variable Name	Default value	Description
CHE_OAUTH_GITHUB_REDIRECTURIS	**http://localhost:${CHE_PORT}/api/oauth/callback**	YoucansetupGitHubOAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_OPENSHIFT_CLIENTID	**NULL**	Configuration of OpenShift OAuth client. Used to obtain OpenShift OAuth token.
CHE_OAUTH_OPENSHIFT_CLIENTSECRET	**NULL**	Configurationof OpenShift OAuth client. Used to obtain OpenShift OAuth token.
CHE_OAUTH_OPENSHIFT_OAUTH__ENDPOINT	**NULL**	ConfigurationofOpenShift OAuth client. Used to obtain OpenShift OAuth token.
CHE_OAUTH_OPENSHIFT_VERIFY__TOKEN__URL	**NULL**	ConfigurationofOpenShiftOAuth client. Used to obtain OpenShift OAuth token.

Table 4.4. Internal

Environment Variable Name	Default value	Description
SCHEDULE_CORE__POOL__SIZE	**10**	CodeReady Workspaces extensions can be scheduled executions on a time basis. This configures the size of the thread pool allocated to extensions that are launched on a recurring schedule.
ORG_EVERREST_ASYNCHRONOUS	**false**	Everrest is a Java Web Services toolkit that manages JAX-RS & web socket communications Users should rarely need to configure this. Disable asynchronous mechanism that is embedded in everrest.
ORG_EVERREST_ASYNCHRONOUS_POOL_SIZE	**20**	Quantity of asynchronous requests which may be processed at the same time

Environment Variable Name	Default value	Description
ORG_EVERREST_ASYNCHRONOUS_QUEUE_SIZE	**500**	Size of queue. If asynchronous request can't be processed after consuming it will be added in queue.
ORG_EVERREST_ASYNCHRONOUS_JOB_TIMEOUT	**10**	Timeout in minutes for request. If after timeout request is not done or client did not come yet to get result of request it may be discarded.
ORG_EVERREST_ASYNCHRONOUS_CACHE_SIZE	**1024**	Size of cache for waiting, running and ended request.
ORG_EVERREST_ASYNCHRONOUS_SERVICE_PATH	**/async/**	Path to asynchronous service
DB_SCHEMA_FLYWAY_BASELINE_ENABLED	**true**	DB initialization and migration configuration
DB_SCHEMA_FLYWAY_BASELINE_VERSION	**5.0.0.8.1**	DBinitialization and migration configuration
DB_SCHEMA_FLYWAY_SCRIPTS_PREFIX		DBinitializationand migration configuration
DB_SCHEMA_FLYWAY_SCRIPTS_SUFFIX	**.sql**	DBinitializationandmigration configuration
DB_SCHEMA_FLYWAY_SCRIPTS_VERSION__SEPARATOR	**__**	DBinitializationandmigrationconfiguration
DB_SCHEMA_FLYWAY_SCRIPTS_LOCATIONS	**classpath:che-schema**	DBinitializationandmigrationconfiguration

Table 4.5. OpenShift Infra parameters

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_MASTER__URL		Configuration of OpenShift client that Infra will use
CHE_INFRA_KUBERNETES_TRUST__CERTS		Configurationof OpenShift client that Infra will use

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_SERVER__STRATEGY	**default-host**	Defines the way how servers are exposed to the world in OpenShift infra. List of strategies implemented in CodeReady Workspaces: default-host, multi-host, single-host
CHE_INFRA_KUBERNETES_SINGLE__HOST_WORKSPACE_EXPOSURE	**native**	Defines the way in which the workspace plugins and editors are exposed in the single-host mode. Supported exposures: - 'native': Exposes servers using OpenShift Ingresses. Works only on OpenShift.
CHE_INFRA_KUBERNETES_INGRESS_DOMAIN		Used to generate domain for a server in a workspace in case property **che.infra.kubernetes.server_strategy** is set to **multi-host**
CHE_INFRA_KUBERNETES_NAMESPACE		DEPRECATED - please do not change the value of this property otherwise the existing workspaces will loose data. Do not set it on new installations. Defines OpenShift namespace in which all workspaces will be created. If not set, every workspace will be created in a new namespace, where namespace = workspace id It's possible to use <username> and <userid> placeholders (e.g.: che-workspace-<username>). In that case, new namespace will be created for each user. Service account with permission to create new namespace must be used. Ignored for OpenShift infra. Use **che.infra.openshift.project** instead If the namespace pointed to by this property exists, it will be used for all workspaces. If it does not exist, the namespace specified by the che.infra.kubernetes.namespace.default will be created and used.

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT	**<username>-che**	Defines OpenShift default namespace in which user's workspaces are created if user does not override it. It's possible to use <username>, <userid> and <workspaceid> placeholders (e.g.: che-workspace-<username>). In that case, new namespace will be created for each user (or workspace). Is used by OpenShift infra as well to specify Project
CHE_INFRA_KUBERNETES_NAMESPACE_ALLOW__USER__DEFINED	**false**	Defines if a user is able to specify OpenShift namespace (or OpenShift project) different from the default. It's NOT RECOMMENDED to configured true without OAuth configured. This property is also used by the OpenShift infra.
CHE_INFRA_KUBERNETES_SERVICE__ACCOUNT__NAME	**NULL**	Defines OpenShift Service Account name which should be specified to be bound to all workspaces pods. Note that OpenShift Infrastructure won't create the service account and it should exist. OpenShift infrastructure will check if project is predefined(if **che.infra.openshift.project** is not empty): - if it is predefined then service account must exist there - if it is 'NULL' or empty string then infrastructure will create new OpenShift project per workspace and prepare workspace service account with needed roles there

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_WORKSPACE__SA__CLUSTER__ROLES	**NULL**	Specifies optional, additional cluster roles to use with the workspace service account. Note that the cluster role names must already exist, and the CodeReady Workspaces service account needs to be able to create a Role Binding to associate these cluster roles with the workspace service account. The names are comma separated. This property deprecates 'che.infra.kubernetes.cluster_role_name'.
CHE_INFRA_KUBERNETES_WORKSPACE__START__TIMEOUT__MIN	**8**	Defines time frame that limits the OpenShift workspace start time
CHE_INFRA_KUBERNETES_INGRESS__START__TIMEOUT__MIN	**5**	Defines the timeout in minutes that limits the period for which OpenShift Ingress become ready
CHE_INFRA_KUBERNETES_WORKSPACE__UNRECOVERABLE__EVENTS	**FailedMount,FailedScheduling,MountVolume.SetUpfailed, Failed to pull image,FailedCreate**	If during workspace startup an unrecoverable event defined in the property occurs, terminate workspace immediately instead of waiting until timeout Note that this SHOULD NOT include a mere 'Failed' reason, because that might catch events that are not unrecoverable. A failed container startup is handled explicitly by CodeReady Workspaces server.
CHE_INFRA_KUBERNETES_PVC_ENABLED	**true**	Defines whether use the Persistent Volume Claim for che workspace needs e.g backup projects, logs etc or disable it.

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_PVC_STRATEGY	**common**	Defined which strategy will be used while choosing PVC for workspaces. Supported strategies: - 'common' All workspaces in the same OpenShift Namespace will reuse the same PVC. Name of PVC may be configured with 'che.infra.kubernetes.pvc.name'. Existing PVC will be used or new one will be created if it doesn't exist. - 'unique' Separate PVC for each workspace's volume will be used. Name of PVC is evaluated as '{che.infra.kubernetes.pvc.name} + '-' + `{generated_8_chars}'. Existing PVC will be used or a new one will be created if it doesn't exist. - 'per-workspace' Separate PVC for each workspace will be used. Name of PVC is evaluated as '{che.infra.kubernetes.pvc.name} + '-' + `{WORKSPACE_ID}'. Existing PVC will be used or a new one will be created if it doesn't exist.

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_PVC_PRECREATE__SUBPATHS	**true**	Defines whether to run a job that creates workspace's subpath directories in persistent volume for the 'common' strategy before launching a workspace. Necessary in some versions of OpenShift/OpenShift as workspace subpath volume mounts are created with root permissions, and thus cannot be modified by workspaces running as a user (presents an error importing projects into a workspace in CodeReady Workspaces). The default is 'true', but should be set to false if the version of Openshift/OpenShift creates subdirectories with user permissions. Relevant issue: https://github.com/kubernetes/kubernetes/issues/41638 Note that this property has effect only if the 'common' PVC strategy used.
CHE_INFRA_KUBERNETES_PVC_NAME	**claim-che-workspace**	Defines the settings of PVC name for che workspaces. Each PVC strategy supplies this value differently. See doc for che.infra.kubernetes.pvc.strategy property
CHE_INFRA_KUBERNETES_PVC_STORAGE__CLASS__NAME		Defines the storage class of Persistent Volume Claim for the workspaces. Empty strings means 'use default'.
CHE_INFRA_KUBERNETES_PVC_QUANTITY	**10Gi**	Defines the size of Persistent Volume Claim of che workspace. Format described here: https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html
CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE	**centos:centos7**	Pod that is launched when performing persistent volume claim maintenance jobs on OpenShift

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE_PULL__POLICY	**IfNotPresent**	Image pull policy of container that used for the maintenance jobs on OpenShift/OpenShift cluster
CHE_INFRA_KUBERNETES_PVC_JOBS_MEMORYLIMIT	**250Mi**	Defines pod memory limit for persistent volume claim maintenance jobs
CHE_INFRA_KUBERNETES_PVC_ACCESS__MODE	**ReadWriteOnce**	Defines Persistent Volume Claim access mode. Note that for common PVC strategy changing of access mode affects the number of simultaneously running workspaces. If OpenShift flavor where che running is using PVs with RWX access mode then a limit of running workspaces at the same time bounded only by che limits configuration like(RAM, CPU etc). Detailed information about access mode is described here: https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html
CHE_INFRA_KUBERNETES_PVC_WAIT__BOUND	**true**	Defines whether CodeReady Workspaces Server should wait workspaces PVCs to become bound after creating. It's used by all PVC strategies. It should be set to **false** in case if **volumeBindingMode** is configured to **WaitForFirstConsumer** otherwise workspace starts will hangs up on phase of waiting PVCs. Default value is true (means that PVCs should be waited to be bound)
CHE_INFRA_KUBERNETES_INSTALLER__SERVER__MIN__PORT	**10000**	Defined range of ports for installers servers By default, installer will use own port, but if it conflicts with another installer servers then OpenShift infrastructure will reconfigure installer to use first available from this range

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_INSTALLER__SERVER__MAX__PORT	**20000**	Definedrange of ports for installers servers By default, installer will use own port, but if it conflicts with another installer servers then OpenShift infrastructure will reconfigure installer to use first available from this range
CHE_INFRA_KUBERNETES_INGRESS_ANNOTATIONS__JSON	**NULL**	Defines annotations for ingresses which are used for servers exposing. Value depends on the kind of ingress controller. OpenShift infrastructure ignores this property because it uses Routes instead of ingresses. Note that for a single-host deployment strategy to work, a controller supporting URL rewriting has to be used (so that URLs can point to different servers while the servers don't need to support changing the app root). The che.infra.kubernetes.ingress.path.rewrite_transform property defines how the path of the ingress should be transformed to support the URL rewriting and this property defines the set of annotations on the ingress itself that instruct the chosen ingress controller to actually do the URL rewriting, potentially building on the path transformation (if required by the chosen ingress controller). For example for nginx ingress controller 0.22.0 and later the following value is recommended: **{'ingress.kubernetes.io/rewrite-target': '/$1','ingress.kubernetes.io/ssl-redirect': 'false',\ 'ingress.kubernetes.io/proxy-connect-timeout': '3600','ingress.kubernetes.io/proxy-read-timeout': '3600'}** and the che.infra.kubernetes.ingress.path.rewrite_transform should be set to '%s(.*)' For nginx ingress controller older than 0.22.0, the rewrite-target should be set to

Environment Variable Name	Default value	Description
		merely '/' and the path transform to '%s' (see the the che.infra.kubernetes.ingress.path.rewrite_transform property). Please consult the nginx ingress controller documentation for the explanation of how the ingress controller uses the regular expression present in the ingress path and how it achieves the URL rewriting.
CHE_INFRA_KUBERNETES_INGRESS_PATH__TRANSFORM	**NULL**	Defines a 'recipe' on how to declare the path of the ingress that should expose a server. The '%s' represents the base public URL of the server and is guaranteed to end with a forward slash. This property must be a valid input to the String.format() method and contain exactly one reference to '%s'. Please see the description of the che.infra.kubernetes.ingress.annotations_json property to see how these two properties interplay when specifying the ingress annotations and path. If not defined, this property defaults to '%s' (without the quotes) which means that the path is not transformed in any way for use with the ingress controller.
CHE_INFRA_KUBERNETES_POD_SECURITY__CONTEXT_RUN__AS__USER	**NULL**	Defines security context for pods that will be created by OpenShift Infra This is ignored by OpenShift infra
CHE_INFRA_KUBERNETES_POD_SECURITY__CONTEXT_FS__GROUP	**NULL**	Definessecurity context for pods that will be created by OpenShift Infra This is ignored by OpenShift infra

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_POD_TERMINATION__GRACE__PERIOD__SEC	**0**	Defines grace termination period for pods that will be created by OpenShift / OpenShift infrastructures Grace termination period of OpenShift / OpenShift workspace's pods defaults '0', which allows to terminate pods almost instantly and significantly decrease the time required for stopping a workspace. Note: if **terminationGracePeriodSeconds** have been explicitly set in OpenShift / OpenShift recipe it will not be overridden.
CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC__REQUESTS_MAX	**1000**	Number of maximum concurrent async web requests (http requests or ongoing web socket calls) supported in the underlying shared http client of the **OpenShiftClient** instances. Default values are 64, and 5 per-host, which doesn't seem correct for multi-user scenarios knowing that CodeReady Workspaces keeps a number of connections opened (e.g. for command or ws-agent logs)
CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC__REQUESTS_MAX__PER__HOST	**1000**	Numberof maximum concurrent async web requests (http requests or ongoing web socket calls) supported in the underlying shared http client of the **OpenShiftClient** instances. Default values are 64, and 5 per-host, which doesn't seem correct for multi-user scenarios knowing that **{prod-short}** keeps a number of connections opened (e.g. for command or ws-agent logs)
CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION__POOL_MAX__IDLE	**5**	Max number of idle connections in the connection pool of the OpenShift-client shared http client

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION__POOL_KEEP__ALIVE__MIN	**5**	Keep-alive timeout of the connection pool of the OpenShift-client shared http client in minutes
CHE_INFRA_KUBERNETES_TLS__ENABLED	**false**	Creates Ingresses with Transport Layer Security (TLS) enabled In OpenShift infrastructure, Routes will be TLS-enabled
CHE_INFRA_KUBERNETES_TLS__SECRET		Name of a secret that should be used when creating workspace ingresses with TLS Ignored by OpenShift infrastructure
CHE_INFRA_KUBERNETES_TLS__KEY	**NULL**	Data for TLS Secret that should be used for workspaces Ingresses cert and key should be encoded with Base64 algorithm These properties are ignored by OpenShift infrastructure
CHE_INFRA_KUBERNETES_TLS__CERT	**NULL**	Datafor TLS Secret that should be used for workspaces Ingresses cert and key should be encoded with Base64 algorithm These properties are ignored by OpenShift infrastructure

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_RUNTIMES__CONSISTENCY__CHECK__PERIOD__MIN	**-1**	Defines the period with which runtimes consistency checks will be performed. If runtime has inconsistent state then runtime will be stopped automatically. Value must be more than 0 or **-1**, where **-1** means that checks won't be performed at all. It is disabled by default because there is possible CodeReady Workspaces Server configuration when CodeReady Workspaces Server doesn't have an ability to interact with OpenShift API when operation is not invoked by user. It DOES work on the following configurations: - workspaces objects are created in the same namespace where CodeReady Workspaces Server is located; - cluster-admin service account token is mount to CodeReady Workspaces Server pod; It DOES NOT work on the following configurations: - CodeReady Workspaces Server communicates with OpenShift API using token from OAuth provider;

Table 4.6. OpenShift Infra parameters

Environment Variable Name	Default value	Description

Environment Variable Name	Default value	Description
CHE_INFRA_OPENSHIFT_PROJECT		DEPRECATED - please do not change the value of this property otherwise the existing workspaces will loose data. Do not set it on new installations. Defines OpenShift namespace in which all workspaces will be created. If not set, every workspace will be created in a new project, where project name = workspace id It's possible to use <username> and <userid> placeholders (e.g.: che-workspace-<username>). In that case, new project will be created for each user. OpenShift oauth or service account with permission to create new projects must be used. If the project pointed to by this property exists, it will be used for all workspaces. If it does not exist, the namespace specified by the che.infra.kubernetes.namespace.default will be created and used.
CHE_INFRA_OPENSHIFT_TRUSTED__CA__BUNDLES__CONFIG__MAP	**ca-certs**	Configures name of the trust-store config map where the CA bundles are stored in Openshift 4. This map is supposed to be initially created by CodeReady Workspaces installer (operator or etc) with basically any name, and CodeReady Workspaces server finds it by specific label (see below) during workspace startup and then creates and mounts same map in the namespace of the workspace. The property defines name of the map in workspace namespace.
CHE_INFRA_OPENSHIFT_TRUSTED__CA__BUNDLES__CONFIG__MAP__LABELS	**config.openshift.io/inject-trusted-cabundle=true**	Label name for config maps which are used for automatic certificate injection in Openshift 4.
CHE_INFRA_OPENSHIFT_TRUSTED__CA__BUNDLES__MOUNT__PATH	**/public-certs**	Configures path on workspace containers where the CA bundles are mount.

Environment Variable Name	Default value	Description
CHE_SINGLEPORT_WILDCARD__DOMAIN_HOST	**NULL**	Single port mode wildcard domain host & port. nip.io is used by default
CHE_SINGLEPORT_WILDCARD__DOMAIN_PORT	**NULL**	Singleport mode wildcard domain host & port. nip.io is used by default
CHE_SINGLEPORT_WILDCARD__DOMAIN_IPLESS	**false**	Enable single port custom DNS without inserting the IP

Table 4.7. Experimental properties

Environment Variable Name	Default value	Description
CHE_WORKSPACE_PLUGIN__BROKER_METADATA_IMAGE	**quay.io/eclipse/che-plugin-metadata-broker:v3.3.0**	Docker image of CodeReady Workspaces plugin broker app that resolves workspace tooling configuration and copies plugins dependencies to a workspace
CHE_WORKSPACE_PLUGIN__BROKER_ARTIFACTS_IMAGE	**quay.io/eclipse/che-plugin-artifacts-broker:v3.3.0**	Dockerimage of **{prod-short}** plugin broker app that resolves workspace tooling configuration and copies plugins dependencies to a workspace
CHE_WORKSPACE_PLUGIN__BROKER_PULL__POLICY	**Always**	Docker image of CodeReady Workspaces plugin broker app that resolves workspace tooling configuration and copies plugins dependencies to a workspace
CHE_WORKSPACE_PLUGIN__BROKER_WAIT__TIMEOUT__MIN	**3**	Defines the timeout in minutes that limits the max period of result waiting for plugin broker.
CHE_WORKSPACE_PLUGIN__REGISTRY__URL	**https://che-plugin-registry.prod-preview.openshift.io/v3**	Workspace tooling plugins registry endpoint. Should be a valid HTTP URL. **Example: http://che-plugin-registry-eclipse-che.192.168.65.2.nip.io** In case CodeReady Workspaces plugins tooling is not needed value 'NULL' should be used

Environment Variable Name	Default value	Description
CHE_WORKSPACE_DEVFILE__REGISTRY__URL	**https://che-devfile-registry.prod-preview.openshift.io/**	Devfile Registry endpoint. Should be a valid HTTP URL. **Example: http://che-devfile-registry-eclipse-che.192.168.65.2.nip.io** In case CodeReady Workspaces plugins tooling is not needed value 'NULL' should be used
CHE_WORKSPACE_STORAGE_AVAILABLE__TYPES	**persistent,ephemeral,async**	The configuration property that defines available values for storage types that clients like Dashboard should propose for users during workspace creation/update. Available values: - 'persistent': Persistent Storage slow I/O but persistent. - 'ephemeral': Ephemeral Storage allows for faster I/O but may have limited storage and is not persistent. - 'async': Experimental feature: Asynchronous storage is combination of Ephemeral and Persistent storage. Allows for faster I/O and keep your changes, will backup on stop and restore on start workspace. Will work only if: - che.infra.kubernetes.pvc.strategy='common' - che.limits.user.workspaces.run.count=1 - che.infra.kubernetes.namespace.allow_user_defined=false - che.infra.kubernetes.namespace.default contains <username> in other cases remove 'async' from the list.
CHE_WORKSPACE_STORAGE_PREFERRED__TYPE	**persistent**	The configuration property that defines a default value for storage type that clients like Dashboard should propose for users during workspace creation/update. The 'async' value not recommended as default type since it's experimental

Environment Variable Name	Default value	Description
CHE_SERVER_SECURE__EXPOSER	**default**	Configures in which way secure servers will be protected with authentication. Suitable values: - 'default': jwtproxy is configured in a pass-through mode. So, servers should authenticate requests themselves. - 'jwtproxy': jwtproxy will authenticate requests. So, servers will receive only authenticated ones.
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_TOKEN_ISSUER	**wsmaster**	Jwtproxy issuer string, token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_TOKEN_TTL	**8800h**	Jwtproxyissuer string, token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_AUTH_LOADER_PATH	**/_app/loader.html**	Jwtproxyissuerstring, token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_IMAGE	**quay.io/eclipse/che-jwtproxy:0.10.0**	Jwtproxyissuerstring,token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_MEMORY__LIMIT	**128mb**	Jwtproxyissuerstring,tokenlifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_CPU__LIMIT	**0.5**	Jwtproxyissuerstring,tokenlifetimeand optional auth page path to route unsigned requests to.

Table 4.8. Configuration of major "/websocket" endpoint

Environment Variable Name	Default value	Description
CHE_CORE_JSONRPC_PROCESSOR__MAX__POOL__SIZE	**50**	Maximum size of the JSON RPC processing pool in case if pool size would be exceeded message execution will be rejected

Environment Variable Name	Default value	Description
CHE_CORE_JSONRPC_PROCESSOR__CORE__POOL__SIZE	**5**	Initial json processing pool. Minimum number of threads that used to process major JSON RPC messages.
CHE_CORE_JSONRPC_PROCESSOR__QUEUE__CAPACITY	**100000**	Configuration of queue used to process Json RPC messages.

Table 4.9. Configuration of major "/websocket-minor" endpoint

Environment Variable Name	Default value	Description
CHE_CORE_JSONRPC_MINOR__PROCESSOR__MAX__POOL__SIZE	**100**	Maximum size of the JSON RPC processing pool in case if pool size would be exceeded message execution will be rejected
CHE_CORE_JSONRPC_MINOR__PROCESSOR__CORE__POOL__SIZE	**15**	Initial json processing pool. Minimum number of threads that used to process minor JSON RPC messages.
CHE_CORE_JSONRPC_MINOR__PROCESSOR__QUEUE__CAPACITY	**10000**	Configuration of queue used to process Json RPC messages.
CHE_METRICS_PORT	**8087**	Port the the http server endpoint that would be exposed with Prometheus metrics

Table 4.10. CORS settings

Environment Variable Name	Default value	Description
CHE_CORS_ALLOWED__ORIGINS	*	CORS filter on WS Master is turned off by default. Use environment variable 'CHE_CORS_ENABLED=true' to turn it on 'cors.allowed.origins' indicates which request origins are allowed

Environment Variable Name	Default value	Description
CHE_CORS_ALLOW__CREDENTIALS	**false**	'cors.support.credentials' indicates if it allows processing of requests with credentials (in cookies, headers, TLS client certificates)

Table 4.11. Factory defaults

Environment Variable Name	Default value	Description
CHE_FACTORY_DEFAULT__EDITOR	**eclipse/che-theia/7.18.2**	Editor and plugin which will be used for factories which are created from remote git repository which doesn't contain any CodeReady Workspaces-specific workspace descriptors (like .devfile of .factory.json) Multiple plugins must be comma-separated, for example: pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion
CHE_FACTORY_DEFAULT__PLUGINS	**eclipse/che-machine-exec-plugin/7.18.2**	Editorand plugin which will be used for factories which are created from remote git repository which doesn't contain any **{prod-short}**-specific workspace descriptors (like .devfile of .factory.json) Multiple plugins must be comma-separated, for example: pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion
CHE_FACTORY_DEFAULT__DEVFILE__FILENAMES	**devfile.yaml,.devfile.yaml**	Devfile filenames to look on repository-based factories (like GitHub etc). Factory will try to locate those files in the order they enumerated in the property.

Table 4.12. Devfile defaults

Environment Variable Name	Default value	Description
CHE_WORKSPACE_DEVFILE_DEFAULT__EDITOR	**eclipse/che-theia/7.18.2**	Default Editor that should be provisioned into Devfile if there is no specified Editor Format is **editorPublisher/editorName/editorVersion** value. **NULL** or absence of value means that default editor should not be provisioned.
CHE_WORKSPACE_DEVFILE_DEFAULT__EDITOR_PLUGINS	**eclipse/che-machine-exec-plugin/7.18.2**	Default Plugins which should be provisioned for Default Editor. All the plugins from this list that are not explicitly mentioned in the user-defined devfile will be provisioned but only when the default editor is used or if the user-defined editor is the same as the default one (even if in different version). Format is comma-separated **pluginPublisher/pluginName/pluginVersion** values, and URLs. For example: eclipse/che-theia-exec-plugin/0.0.1,eclipse/che-theia-terminal-plugin/0.0.1,https://cdn.pluginregistry.com/vi-mode/meta.yaml If the plugin is a URL, the plugin's meta.yaml is retrieved from that URL.
CHE_WORKSPACE_PROVISION_SECRET_LABELS	**app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspace-secret**	Defines comma-separated list of labels for selecting secrets from a user namespace, which will be mount into workspace containers as a files or env variables. Only secrets that match ALL given labels will be selected.
CHE_WORKSPACE_DEVFILE_ASYNC_STORAGE_PLUGIN	**eclipse/che-async-pv-plugin/nightly**	Plugin is added in case async storage feature will be enabled in workspace config and supported by environment
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_IMAGE	**quay.io/eclipse/che-workspace-data-sync-storage:latest**	Docker image for the CodeReady Workspaces async storage

Environment Variable Name	Default value	Description
CHE_WORKSPACE_POD_NODE__SELECTOR	**NULL**	Optionally configures node selector for workspace pod. Format is comma-separated key=value pairs, e.g: disktype=ssd,cpu=xlarge,foo=bar
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN__TIMEOUT__MIN	**120**	The timeout for the Asynchronous Storage Pod shutdown after stopping the last used workspace. Value less or equal to 0 interpreted as disabling shutdown ability.
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN__CHECK__PERIOD__MIN	**30#**	Defines the period with which the Asynchronous Storage Pod stopping ability will be performed (once in 30 minutes by default)

Table 4.13. Che system

Environment Variable Name	Default value	Description
CHE_SYSTEM_SUPER__PRIVILEGED__MODE	**false**	System Super Privileged Mode. Grants users with the manageSystem permission additional permissions for getByKey, getByNameSpace, stopWorkspaces, and getResourcesInformation. These are not given to admins by default and these permissions allow admins gain visibility to any workspace along with naming themselves with admin privileges to those workspaces.
CHE_SYSTEM_ADMIN__NAME	**admin**	Grant system permission for 'che.admin.name' user. If the user already exists it'll happen on component startup, if not - during the first login when user is persisted in the database.

Table 4.14. Workspace limits

Environment Variable Name	Default value	Description
CHE_LIMITS_WORKSPACE_ENV_RAM	**16gb**	Workspaces are the fundamental runtime for users when doing development. You can set parameters that limit how workspaces are created and the resources that are consumed. 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.
CHE_LIMITS_WORKSPACE_RUN_TIMEOUT	**0**	The length of time in milliseconds that a workspace will run, regardless of activity, before the system will suspend it. Set this property if you want to automatically stop workspaces after a period of time. The default is zero, meaning that there is no run timeout.

Table 4.15. Users workspace limits

Environment Variable Name	Default value	Description
CHE_LIMITS_USER_WORKSPACES_RAM	**-1**	The 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.

Environment Variable Name	Default value	Description
CHE_LIMITS_USER_WORKSPACES_COUNT	**-1**	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.

Table 4.16. Organizations workspace limits

Environment Variable Name	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.

Environment Variable Name	Default value	Description
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.
CHE_MAIL_FROM__EMAIL__ADDRESS	**che@noreply.com**	Address that will be used as from email for email notifications

Table 4.17. Organizations notifications settings

Environment Variable Name	Default value	Description
CHE_ORGANIZATION_EMAIL_MEMBER__ADDED__SUBJECT	**You'vebeen added to a Che Organization**	Organization notifications sunjects and templates
CHE_ORGANIZATION_EMAIL_MEMBER__ADDED__TEMPLATE	**st-html-templates/user_added_to_organization**	Organizationnotifications sunjects and templates
CHE_ORGANIZATION_EMAIL_MEMBER__REMOVED__SUBJECT	**You'vebeen removed from a Che Organization**	
CHE_ORGANIZATION_EMAIL_MEMBER__REMOVED__TEMPLATE	**st-html-templates/user_removed_from_organization**	
CHE_ORGANIZATION_EMAIL_ORG__REMOVED__SUBJECT	**CheOrganization deleted**	
CHE_ORGANIZATION_EMAIL_ORG__REMOVED__TEMPLATE	**st-html-templates/organization_deleted**	
CHE_ORGANIZATION_EMAIL_ORG__RENAMED__SUBJECT	**CheOrganization renamed**	

Environment Variable Name	Default value	Description
CHE_ORGANIZATION_EMAIL_ORG__RENAMED__TEMPLATE	**st-html-templates/organization_renamed**	

Table 4.18. Multi-user-specific OpenShift infrastructure configuration

Environment Variable Name	Default value	Description
CHE_INFRA_OPENSHIFT_OAUTH__IDENTITY__PROVIDER	**NULL**	Alias of the Openshift identity provider registered in Keycloak, that should be used to create workspace OpenShift resources in Openshift namespaces owned by the current CodeReady Workspaces user. Should be set to NULL if **che.infra.openshift.project** is set to a non-empty value. For more information see the following documentation: https://www.keycloak.org/docs/latest/server_admin/index.html#openshift-4

Table 4.19. Keycloak configuration

Environment Variable Name	Default value	Description
CHE_KEYCLOAK_AUTH__SERVER__URL	**http://${CHE_HOST}:5050/auth**	Url to keycloak identity provider server Can be set to NULL only if **che.keycloak.oidcProvider** is used
CHE_KEYCLOAK_REALM	**che**	Keycloak realm is used to authenticate users Can be set to NULL only if **che.keycloak.oidcProvider** is used
CHE_KEYCLOAK_CLIENT__ID	**che-public**	Keycloak client id in che.keycloak.realm that is used by dashboard, ide and cli to authenticate users

Table 4.20. RedHat Che specific configuration

Environment Variable Name	Default value	Description
CHE_KEYCLOAK_OSO_ENDPOINT	**NULL**	URL to access OSO oauth tokens
CHE_KEYCLOAK_GITHUB_ENDPOINT	**NULL**	URL to access Github oauth tokens
CHE_KEYCLOAK_ALLOWED__CLOCK__SKEW__SEC	**3**	The number of seconds to tolerate for clock skew when verifying exp or nbf claims.
CHE_KEYCLOAK_USE__NONCE	**true**	Use the OIDC optional **nonce** feature to increase security.
CHE_KEYCLOAK_JS__ADAPTER__URL	**NULL**	URL to the Keycloak Javascript adapter we want to use. if set to NULL, then the default used value is **${che.keycloak.auth_server_url}/js/keycloak.js**, or **<che-server>/api/keycloak/OIDCKeycloak.js** if an alternate **oidc_provider** is used
CHE_KEYCLOAK_OIDC__PROVIDER	**NULL**	Base URL of an alternate OIDC provider that provides a discovery endpoint as detailed in the following specification https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig
CHE_KEYCLOAK_USE__FIXED__REDIRECT__URLS	**false**	Set to true when using an alternate OIDC provider that only supports fixed redirect Urls This property is ignored when **che.keycloak.oidc_provider** is NULL
CHE_KEYCLOAK_USERNAME__CLAIM	**NULL**	Username claim to be used as user display name when parsing JWT token if not defined the fallback value is 'preferred_username'

Environment Variable Name	Default value	Description
CHE_OAUTH_SERVICE__MODE	**delegated**	Configuration of OAuth Authentication Service that can be used in 'embedded' or 'delegated' mode. If set to 'embedded', then the service work as a wrapper to CodeReady Workspaces's OAuthAuthenticator (as in Single User mode). If set to 'delegated', then the service will use Keycloak IdentityProvider mechanism. Runtime Exception wii be thrown, in case if this property is not set properly.

4.2. CONFIGURING PROJECT STRATEGIES

The OpenShift project 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.

WARNING

By default, only one workspace in the same project can be running at one time. See Section 4.3, "Running more than one workspace at a time" .

4.2.1. One project per workspace strategy

The strategy creates a new project 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 4.2. One project per workspace

To assign project names composed of a `*codeready-ws*` prefix and workspace id, set:

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

4.2.2. One project for all workspaces strategy

The strategy uses one predefined project for all workspaces.

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

Example 4.3. One project for all workspaces

To have all workspaces created in `*codeready-ws*` project, set:

```
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT=codeready-ws
```

4.2.3. One project per user strategy

The strategy isolates each user in their own project.

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 4.4. One project per user

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

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

4.2.4. Allowing user-defined workspace projects

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

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

```
allowUserDefinedWorkspaceNamespaces
```

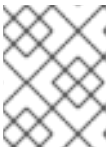
4.3. RUNNING MORE THAN ONE WORKSPACE AT A TIME

This procedure describes how to run more than one workspace simultaneously. This makes it possible for multiple workspace contexts per user to run in parallel.

Prerequisites

- The **'`oc`'** tool is available.

- An instance of CodeReady Workspaces running in OpenShift.



NOTE

The following commands use the default OpenShift project, **openshift-workspaces**, as a user's example for the **-n** option.

Procedure

1. Change the default limit of **1** to **-1** to allow an unlimited number of concurrent workspaces per user:

```
$ oc patch checluster codeready-workspaces -n openshift-workspaces --type merge \
  -p '{ "spec": { "server": {"customCheProperties":
{"CHE_LIMITS_USER_WORKSPACE_RUN_COUNT": "-1"} } }}'
```

1. Set the **per-workspace** or **unique** PVC strategy. See Configuring a CodeReady Workspaces workspace with a persistent volume strategy.

NOTE

When using the *common PVC* strategy, configure the persistent volumes to use the **ReadWriteMany** access mode. That way, any of the user's concurrent workspaces can read from and write to the common PVC.

4.4. CONFIGURING WORKSPACE EXPOSURE STRATEGIES

The following section describes how to configure workspace exposure strategies of a CodeReady Workspaces server and ensure that applications running inside are not vulnerable to outside attacks.

The workspace exposure strategy is configured per CodeReady Workspaces server, using the **che.infra.kubernetes.server_strategy** configuration property or the **CHE_INFRA_KUBERNETES_SERVER__STRATEGY** environment variable.

The supported values for **che.infra.kubernetes.server_strategy** are:

- **multi-host**

For enabling of the **multi-host** strategy:

1. Set the:
   - **che.infra.kubernetes.ingress.domain** configuration property
     or
   - **CHE_INFRA_KUBERNETES_INGRESS_DOMAIN** environment variable
     to match the domain name that will host workspace component subdomains.

4.4.1. Workspace exposure strategies

Specific components of workspaces need to be made accessible outside of the OpenShift cluster. This is typically the user interface of the workspace's IDE, but it can also be the web UI of the application being developed. This enables developers to interact with the application during the development process.

The supported way of making workspace components available to the users is referred to as a *strategy*. This strategy defines whether new subdomains are created for the workspace components and what hosts these components are available on.

CodeReady Workspaces supports: * **multi-host** strategy

4.4.1.1. Multi-host strategy

With this strategy, each workspace component is assigned a new subdomain of the main domain configured for the CodeReady Workspaces server. On OpenShift, this is the only possible strategy, and manual configuration of the workspace exposure strategy is therefore always ignored.

This strategy is the easiest to understand from the perspective of component deployment because any paths present in the URL to the component are received as they are by the component.

On a CodeReady Workspaces server secured using the Transport Layer Security (TLS) protocol, creating new subdomains for each component of each workspace requires a wildcard certificate to be available for all such subdomains for the CodeReady Workspaces deployment to be practical.

4.4.2. Security considerations

This section explains the security impact of using different CodeReady Workspaces workspace exposure strategies.

All the security-related considerations in this section are only applicable to CodeReady Workspaces in multiuser mode. The single user mode does not impose any security restrictions.

4.4.2.1. JSON web token (JWT) proxy

All CodeReady Workspaces plug-ins, editors, and components can require authentication of the user accessing them. This authentication is performed using a JSON web token (JWT) proxy that functions as a reverse proxy of the corresponding component, based on its configuration, and performs the authentication on behalf of the component.

The authentication uses a redirect to a special page on the CodeReady Workspaces server that propagates the workspace and user-specific authentication token (workspace access token) back to the originally requested page.

The JWT proxy accepts the workspace access token from the following places in the incoming requests, in the following order:

1. The token query parameter
2. The Authorization header in the bearer-token format
3. The **access_token** cookie

4.4.2.2. Secured plug-ins and editors

CodeReady Workspaces users do not need to secure workspace plug-ins and workspace editors (such as Che-Theia). This is because the JWT proxy authentication is transparent to the user and is governed by the plug-in or editor definition in their **meta.yaml** descriptors.

4.4.2.3. Secured container-image components

Container-image components can define custom endpoints for which the devfile author can require CodeReady Workspaces-provided authentication, if needed. This authentication is configured using two optional attributes of the endpoint:

- **secure** - A boolean attribute that instructs the CodeReady Workspaces server to put the JWT proxy in front of the endpoint. Such endpoints have to be provided with the workspace access token in one of the several ways explained in Section 4.4.2.1, "JSON web token (JWT) proxy" . The default value of the attribute is **false**.

- **cookiesAuthEnabled** - A boolean attribute that instructs the CodeReady Workspaces server to automatically redirect the unauthenticated requests for current user authentication as described in Section 4.4.2.1, "JSON web token (JWT) proxy" . Setting this attribute to **true** has security consequences because it makes Cross-site request forgery (CSRF) attacks possible. The default value of the attribute is **false**.

4.4.2.4. Cross-site request forgery attacks

Cookie-based authentication can make an application secured by a JWT proxy prone to Cross-site request forgery (CSRF) attacks. See the Cross-site request forgery Wikipedia page and other resources to ensure your application is not vulnerable.

4.4.2.5. Phishing attacks

An attacker who is able to create an Ingress or route inside the cluster with the workspace that shares the host with some services behind a JWT proxy, the attacker may be able to create a service and a specially forged Ingress object. When such a service or Ingress is accessed by a legitimate user that was previously authenticated with a workspace, it can lead to the attacker stealing the workspace access token from the cookies sent by the legitimate user's browser to the forged URL. To eliminate this attack vector, configure OpenShift to disallow setting the host of an Ingress.

4.5. CONFIGURING WORKSPACES NODESELECTOR

This section describes how to configure **nodeSelector** for Pods of CodeReady Workspaces workspaces.

Procedure

CodeReady Workspaces uses the **CHE_WORKSPACE_POD_NODE__SELECTOR** environment variable to configure **nodeSelector**. This variable may contain a set of comma-separated **key=value** pairs to form the nodeSelector rule, or **NULL** to disable it.

```
CHE_WORKSPACE_POD_NODE__SELECTOR=disktype=ssd,cpu=xlarge,[key=value]
```

IMPORTANT

nodeSelector must be configured during CodeReady Workspaces installation. This prevents existing workspaces from failing to run due to volumes affinity conflict caused by existing workspace PVC and Pod being scheduled in different zones.

To avoid Pods and PVCs to be scheduled in different zones on large, multi-zone clusters, create an additional **StorageClass** object (pay attention to the **allowedTopologies** field), which will coordinate the PVC creation process.

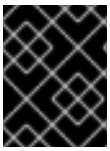
Pass the name of this newly created **StorageClass** to CodeReady Workspaces through the **CHE_INFRA_KUBERNETES_PVC_STORAGE*CLASS*NAME** environment variable. A default empty value of this variable instructs CodeReady Workspaces to use the cluster's default **StorageClass**.

4.6. CONFIGURING RED HAT CODEREADY WORKSPACES SERVER HOSTNAME

This procedure describes how to configure Red Hat CodeReady Workspaces to use custom hostname.

Prerequisites

- The **oc** tool is available.
- The certificate and the private key files are generated.



IMPORTANT

To generate the pair of private key and certificate the same CA must be used as for other Red Hat CodeReady Workspaces hosts.

IMPORTANT

Ask a DNS provider to point the custom hostname to the cluster ingress.

Procedure

1. Pre-create a project for CodeReady Workspaces:

```
$ oc create project openshift-workspaces
```

2. Create a tls secret:

```
$ oc create secret tls ${secret} \ (1)
--key ${key_file} \ (2)
--cert ${cert_file} \ (3)
-n openshift-workspaces
```

(1) The tls secret name

(2) A file with the private key

(3) A file with the certificate

3. Set the following values in the Custom Resource:

```
spec:
  server:
    cheHost: <hostname>          (1)
    cheHostTLSSecret: <secret>   (2)
```

(1) Custom Red Hat CodeReady Workspaces server hostname

(2) The tls secret name

4. If CodeReady Workspaces has been already deployed and CodeReady Workspaces reconfiguring to use a new CodeReady Workspaces hostname is required, log in using RH-SSO and select the **codeready-public** client in the **CodeReady Workspaces** realm and update **Validate Redirect URIs** and **Web Origins** fields with the value of the CodeReady Workspaces hostname.

4.7. 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

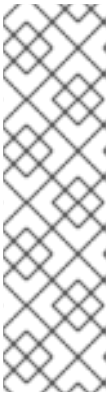
Procedure

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**.
- Certificate files are typically stored as Base64 ASCII files, such as. **.pem**, **.crt**, **.ca-bundle**. Also, they can be encoded as binary data, for example, **.cer**. All **Secrets** that hold certificate files should use the Base64 ASCII certificate rather than the binary data certificate.

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

```
$ oc patch checluster codeready-workspaces -n openshift-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
```

4.8. INSTALLING CODEREADY WORKSPACES USING STORAGE CLASSES

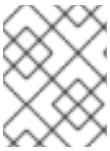
To configure CodeReady Workspaces to use a configured infrastructure storage, install CodeReady Workspaces using storage classes. This is especially useful when a user wants to bind a persistent volume provided by a non-default provisioner. To do so, a user binds this storage for the CodeReady Workspaces data saving and sets the parameters for that storage. These parameters can determine the following:

- A special host path
- A storage capacity
- A volume mod
- Mount options
- A file system
- An access mode
- A storage type
- And many others

CodeReady Workspaces has two components that require persistent volumes to store data:

- A PostgreSQL database.

- A CodeReady Workspaces workspaces. CodeReady Workspaces workspaces store source code using volumes, for example /**projects** volume.



NOTE

CodeReady Workspaces workspaces source code is stored in the persistent volume only if a workspace is not ephemeral.

Persistent volume claims facts:

- CodeReady Workspaces does not create persistent volumes in the infrastructure.
- CodeReady Workspaces uses persistent volume claims (PVC) to mount persistent volumes.
- The CodeReady Workspaces server creates persistent volume claims.
  A user defines a storage class name in the CodeReady Workspaces configuration to use the storage classes feature in the CodeReady Workspaces PVC. With storage classes, a user configures infrastructure storage in a flexible way with additional storage parameters. It is also possible to bind a static provisioned persistent volumes to the CodeReady Workspaces PVC using the class name.

Procedure

Use CheCluster Custom Resource definition to define storage classes:

1. Define storage class names
   To do so, use one of the following methods:

   - **Use arguments for the server:start command**

     i. Provide the storage class name for the PostgreSQL PVC
     Use the **crwctl server:start** command with the **--postgres-pvc-storage-class-name** flag:

     ```
     $ crwctl server:start -m -p minikube -a operator --postgres-pvc-storage-class-name=postgress-storage
     ```

     ii. Provide the storage class name for the CodeReady Workspaces workspaces
     Use the **server:start** command with the **--workspace-pvc-storage-class-name** flag:

     ```
     $ crwctl server:start -m -p minikube -a operator --workspace-pvc-storage-class-name=workspace-storage
     ```

     For CodeReady Workspaces workspaces, the storage class name has different behavior depending on the workspace PVC strategy.

     NOTE

     postgres-pvc-storage-class-name=postgress-storage and **workspace-pvc-storage-class-name** work for the Operator installer and the Helm installer.

   - **Define storage class names using a Custom Resources YAML file:**

i. Create a YAML file with Custom Resources defined for the CodeReady Workspaces installation.

ii. Define fields: **spec#storage#postgresPVCStorageClassName** and **spec#storage#workspacePVCStorageClassName**.

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: codeready-workspaces
spec:
  # ...
  storage:
    # ...
    # keep blank unless you need to use a non default storage class for PostgreSQL
PVC
    postgresPVCStorageClassName: 'postgres-storage'
    # ...
    # keep blank unless you need to use a non default storage class for workspace
PVC(s)
    workspacePVCStorageClassName: 'workspace-storage'
    # ...
```

iii. Start the codeready-workspaces server with your Custom Resources:

```
$ crwctl server:start -m -p minikube -a operator --che-operator-cr-
yaml=/path/to/custom/che/resource/org_v1_che_cr.yaml
```

2. Configure CodeReady Workspaces to store workspaces in one persistent volume and a PostreSQL database in the second one:

   a. Modify your Custom Resources YAML file:

   - Set **pvcStrategy** as **common**.
   - Configure CodeReady Workspaces to start workspaces in a single project.
   - Define storage class names for **postgresPVCStorageClassName** and **workspacePVCStorageClassName**.
   - Example of the YAML file:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: codeready-workspaces
spec:
  server:
    # ...
    workspaceNamespaceDefault: 'che'
    # ...
  storage:
    # ...
    # Defaults to common
    pvcStrategy: 'common'
    # ...
```

```
    # keep blank unless you need to use a non default storage class for PostgreSQL
PVC
    postgresPVCStorageClassName: 'postgres-storage'
    # ...
    # keep blank unless you need to use a non default storage class for workspace
PVC(s)
    workspacePVCStorageClassName: 'workspace-storage'
    # ...
```

b. Start the codeready-workspaces server with your Custom Resources:

```
$ crwctl server:start -m -p minikube -a operator --che-operator-cr-
yaml=/path/to/custom/che/resource/org_v1_che_cr.yaml
```

3. Bind static provisioned volumes using class names:

a. Define the persistent volume for a PostgreSQL database:

```
# che-postgres-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: postgres-pv-volume
  labels:
    type: local
spec:
  storageClassName: postgres-storage
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/che/postgres"
```

b. Define the persistent volume for a CodeReady Workspaces workspace:

```
# che-workspace-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: workspace-pv-volume
  labels:
    type: local
spec:
  storageClassName: workspace-storage
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/che/workspace"
```

c. Bind the two persistent volumes:

```
$ oc apply -f che-workspace-pv.yaml -f che-postgres-pv.yaml
```

NOTE

You must provide valid file permissions for volumes. You can do it using storage class configuration or manually. To manually define permissions, define **storageClass#mountOptions uid** and **gid**. PostgreSQL volume requires **uid=26** and **gid=26**.

4.9. CONFIGURING STORAGE TYPES

Red Hat CodeReady Workspaces supports three types of storage with different capabilities:

- Persistent
- Ephemeral
- Asynchronous

4.9.1. Persistent storage

Persistent storage allows storing user changes directly in the mounted Persistent Volume. User changes are kept safe by the OpenShift infrastructure (storage backend) at the cost of slow I/O, especially with many small files. For example, Node.js projects tend to have many dependencies and the **node_modules**/ directory is filled with thousands of small files.

NOTE

I/O speeds vary depending on the Storage Classes configured in the environment.

Persistent storage is the default mode for new workspaces. To make this setting visible in workspace configuration, add the following to the devfile:

```
attributes:
 persistVolumes: 'true'
```

4.9.2. Ephemeral storage

Ephemeral storage saves files to the **emptyDir** volume. This volume is initially empty. When a Pod is removed from a node, the data in the **emptyDir** volume is deleted forever. This means that all changes are lost on workspace stop or restart.

IMPORTANT

To save the changes, commit and push to the remote before stopping an ephemeral workspace.

Ephemeral mode provides faster I/O than persistent storage. To enable this storage type, add the following to workspace configuration:

```
attributes:
 persistVolumes: 'false'
```

Table 4.21. Comparison between I/O of ephemeral (**emptyDir**) and persitent modes onAWS EBS

Command	Ephemeral	Persitent
Clone Red Hat CodeReady Workspaces	0 m 19 s	1 m 26 s
Generate 1000 random files	1 m 12 s	44 m 53 s

4.9.3. Asynchronous storage

NOTE

Asynchronous storage is an experimental feature.

Asynchronous storage is a combination of persistent and ephemeral modes. The initial workspace container mounts the **emptyDir** volume. Then a backup is performed on workspace stop, and changes are restored on workspace start. Asynchronous storage provides fast I/O (similar to ephemeral mode), and workspace project changes are persisted.

Synchronization is performed by the rsync tool using the SSH protocol. When a workspace is configured with asynchronous storage, the workspace-data-sync plug-in is automatically added to the workspace configuration. The plug-in runs the **rsync** command on workspace start to restore changes. When a workspace is stopped, it sends changes to the permanent storage.

For relatively small projects, the restore procedure is fast, and project source files are immediately available after Che-Theia is initialized. In case **rsync** takes longer, the synchronization process is shown in the Che-Theia status-bar area. (Extension in Che-Theia repository).

NOTE

Asynchronous mode has the following limitations:

- Supports only the *common* PVC strategy
- Supports only the *per-user* project strategy
- Only one workspace can be running at a time

To configure asynchronous storage for a workspace, add the following to workspace configuration:

```
attributes:
 asyncPersist: 'true'
 persistVolumes: 'false'
```

4.9.4. Configuring storage type defaults for CodeReady Workspaces dashboard

Use the fo;llowing two **che.properties** to configure the default client values in CodeReady Workspaces dashboard:

che.workspace.storage.available_types

Defines available values for storage types that clients like the dashboard propose for users during workspace creation or update. Available values: **persistent**, **ephemeral**, and **async**. Separate multiple values by commas. For example:

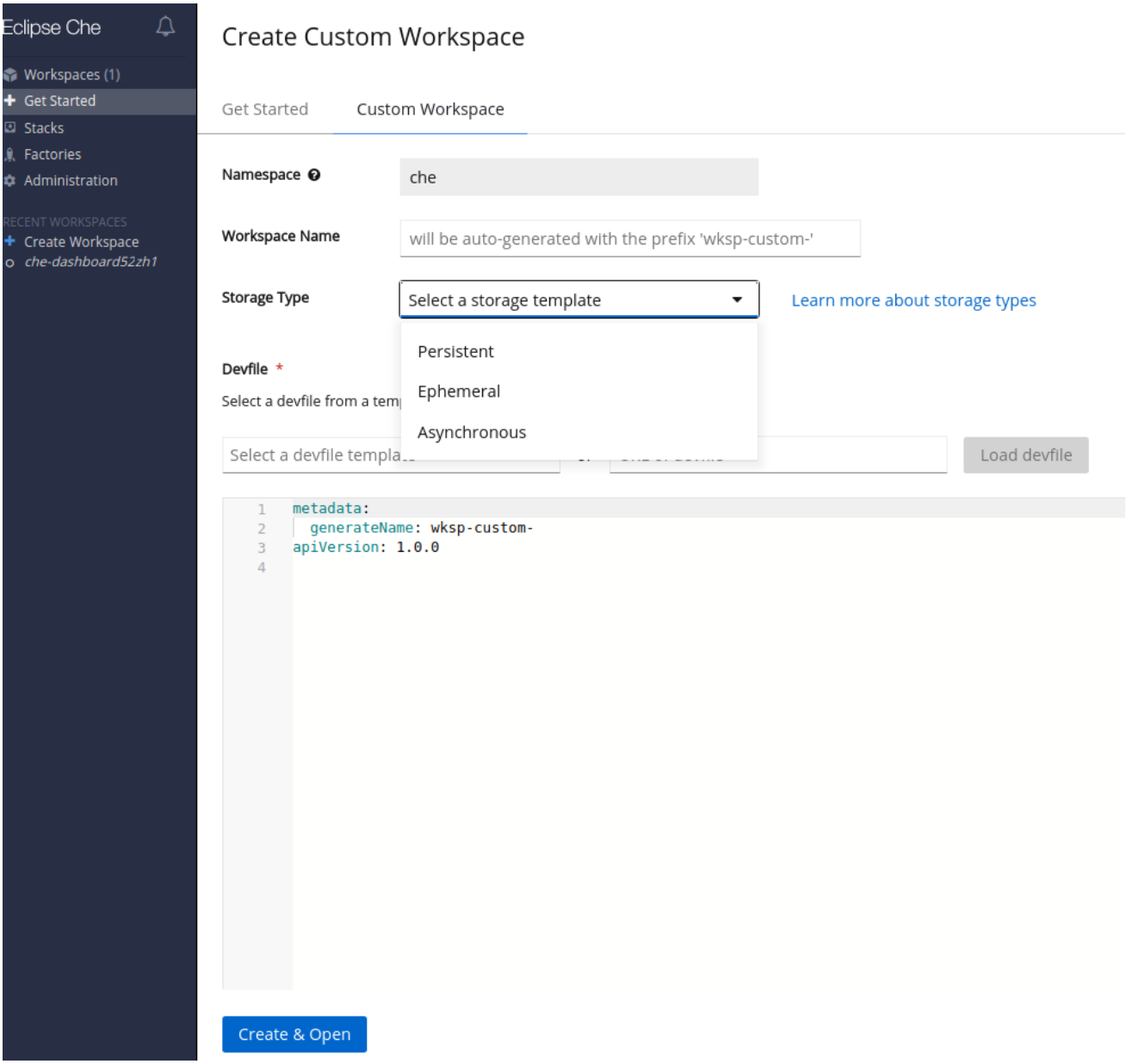
```
che.workspace.storage.available_types=persistent,ephemeral,async
```

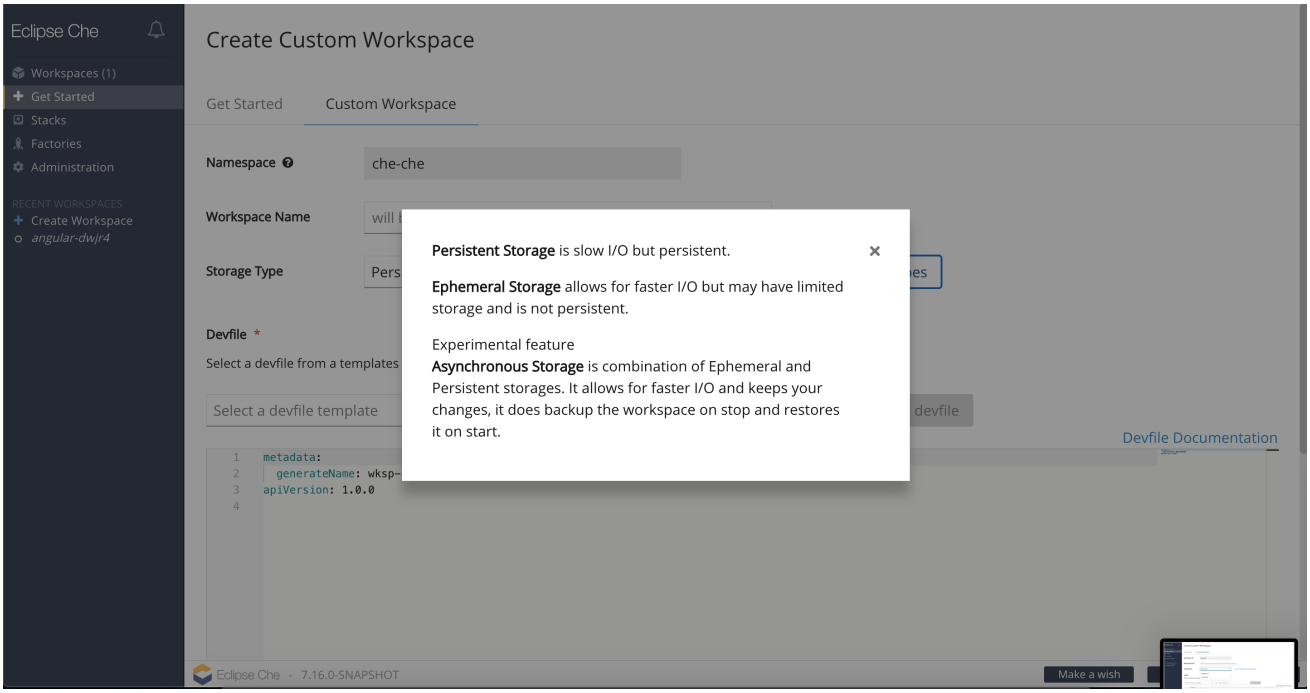
che.workspace.storage.preferred_type

Defines the default value for storage type that clients like the dashboard propose for users during workspace creation or update. The **async** value is not recommended as the default type because it is experimental. For example:

```
che.workspace.storage.preferred_type=persistent
```

The **Storage Type** drop-down menu is available on the **Create Custom Workspace** page of the user dashboard:


Eclipse Che
Workspaces (1)
Get Started
Stacks
Factories
Administration
RECENT WORKSPACES
Create Workspace
che-dashboard52zh1
Create Custom Workspace
Get Started
Custom Workspace
Namespace
che
Workspace Name
will be auto-generated with the prefix 'wksp-custom-'
Storage Type
Select a storage template
Learn more about storage types
Persistent
Ephemeral
Asynchronous
Devfile *
Select a devfile from a tem
Load devfile
metadata:
generateName: wksp-custom-
apiVersion: 1.0.0
Create & Open


Eclipse Che
Create Custom Workspace
Get Started
Custom Workspace
Namespace
che-che
Workspace Name
Storage Type
Devfile *
Select a devfile from a templates
Select a devfile template
devfile
Devfile Documentation
Persistent Storage is slow I/O but persistent.
Ephemeral Storage allows for faster I/O but may have limited storage and is not persistent.
Experimental feature
Asynchronous Storage is combination of Ephemeral and Persistent storages. It allows for faster I/O and keeps your changes, it does backup the workspace on stop and restores it on start.
Eclipse Che - 7.16.0-SNAPSHOT
Make a wish

4.9.5. Idling asynchronous storage Pods

CodeReady Workspaces can shut down the Asynchronous Storage Pod when not used for a configured period of time.

Use these configuration properties to adjust the behavior:

che.infra.kubernetes.async.storage.shutdown_timeout_min

Defines the idle time after which the asynchronous storage Pod is stopped following the stopping of the last active workspace. The default value is 120 minutes.

che.infra.kubernetes.async.storage.shutdown_check_period_min

Defines the frequency with which the asynchronous storage Pod is checked for idleness. The default value is 30 minutes.

4.10. IMPORTING TLS CERTIFICATES TO CODEREADY WORKSPACES SERVER JAVA TRUSTSTORE

When CodeReady Workspaces server attempts to send an HTTPS request to an external service as RH-SSO, a proxy or a git server, the connection fails if CodeReady Workspaces does not trust the TLS certificate used by the external service.

To fix this problem, configure CodeReady Workspaces to authorize HTTPS communication with external services, such as identity and Git servers, by adding information about the TLS certificates to the CodeReady Workspaces configuration.

Prerequisites

- The **oc** tool is available.

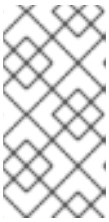
Procedure

1. Save the external services certificates to a local file system.
2. Create a new configMap with the required TLS certificates:

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

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

3. Update the existing CodeReady Workspaces server configuration



NOTE

Use these steps with existing instances of CodeReady Workspaces. To install a new instance of CodeReady Workspaces with self-signed TLS certificates, create a new **CheCluster** 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 CheCluster** Custom Resource

spec.server.serverTrustStoreConfigMapName 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. Otherwise here is a list of things to verify:

- CheCluster attribute **serverTrustStoreConfigMapName** value matches the name of the ConfigMap. Get the value using the following command :

```
$ oc get -o json checluster/codeready-workspaces -n openshift-workspaces | jq .spec.server.serverTrustStoreConfigMapName
```

- CodeReady Workspaces Pod Volumes list contains one Volume that uses the ConfigMap as data-source. To get the list of Volumes of the CodeReady Workspaces Pod:

```
$ oc get po -o json <codeready-workspaces-pod-name> -n openshift-workspaces | jq .spec.volumes
```

- Certificates are mounted in folder /**public-certs**/ of the CodeReady Workspaces server container. This command returns the list of files in that folder:

```
$ oc exec -t <codeready-workspaces-pod-name> -n openshift-workspaces -- ls /public-certs/
```

- In the CodeReady Workspaces server logs there is a line for every certificate added to the Java truststore, including CodeReady Workspaces self signed certificate.

```
$ oc logs <codeready-workspaces-pod-name> -n openshift-workspaces
(...)
Found a custom cert. Adding it to java trust store based on /usr/lib/jvm/java-1.8.0/jre/lib/security/cacerts
(...)
```

- $CodeReady Workspaces server Java trustore contains the certificates. The certificates SHA1 fingerpints are among the list of the SHA1 of the certificates included in the trustore returned by the following command:

```
$ oc exec -t <codeready-workspaces-pod-name> -n openshift-workspaces -- keytool -list -keystore /home/che/cacerts
Your keystore contains 141 entries

(...)
```

To get the SHA1 hash of a certificate on the local filesystem:

```
$ openssl x509 -in <certificate-file-path> -fingerprint -noout
SHA1 Fingerprint=3F:DA:BF:E7:A7:A7:90:62:CA:CF:C7:55:0E:1D:7D:05:16:7D:45:60
```

CHAPTER 5. UPGRADING CODEREADY WORKSPACES

This chapter describes how to upgrade a CodeReady Workspaces instance from version 2.3 to CodeReady Workspaces 2.4.

The method used to install the CodeReady Workspaces instance determines the method to proceed with for the upgrade:

- Section 5.1, “Upgrading CodeReady Workspaces using OperatorHub”
- Section 5.2, “Upgrading CodeReady Workspaces using the CLI management tool”
- Section 5.3.3, “Upgrading CodeReady Workspaces using the CLI management tool in restricted environment”

5.1. UPGRADING CODEREADY WORKSPACES USING OPERATORHUB

This section describes how to upgrade from a previous minor version using the Operator from OperatorHub in the OpenShift web console.

Prerequisites

- An administrator account on an OpenShift instance.
- An instance of a previous minor version of CodeReady Workspaces, installed using the Operator from OperatorHub on the same instance of OpenShift.

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 the installed Operators.
4. Navigate to the **Subscription** tab and enable the following options:
   - **Channel**: **latest**
   - **Approval**: **Automatic**

Verification steps

1. Navigate to the CodeReady Workspaces instance.
2. The 2.4 version number is visible at the bottom of the page.

5.2. UPGRADING CODEREADY WORKSPACES USING THE CLI MANAGEMENT TOOL

This section describes how to upgrade from previous minor version using the CLI management tool.

Prerequisites

- And administrative account on an OpenShift instance.
- A running instance of a previous minor version of Red Hat CodeReady Workspaces, installed using the CLI management tool on the same instance of OpenShift, in the ***<openshift-workspaces>*** project.
- An installation of the **crwctl** 2.4 version management tool. See Section 3.2.1, "Installing the crwctl CLI management tool".

Procedure

1. In all running workspaces in the CodeReady Workspaces 2.3 instance, save and push changes back to the Git repositories.
2. Shut down all workspaces in the CodeReady Workspaces 2.3 instance.
3. Run the following command:

```
$ crwctl -n <openshift-workspaces> server:update
```

NOTE

For slow systems or internet connections, add the **--k8spodwaittimeout=1800000** flag option to the **crwctl server:update** command to extend the Pod timeout period to 1800000 ms or longer.

Verification steps

1. Navigate to the CodeReady Workspaces instance.
2. The 2.4 version number is visible at the bottom of the page.

5.3. UPGRADING CODEREADY WORKSPACES USING THE CLI MANAGEMENT TOOL IN RESTRICTED ENVIRONMENT

This section describes how to upgrade Red Hat CodeReady Workspaces using the CLI management tool in restricted environment. The upgrade path supports minor version update, from CodeReady Workspaces version 2.3 to version 2.4.

Prerequisites

- An administrative account on an instance of OpenShift.
- A running instance version 2.3 of Red Hat CodeReady Workspaces, installed using the CLI management tool on the same instance of OpenShift, with the crwctl **--installer operator** method, in the ***<openshift-workspaces>*** project. See Section 3.3, "Installing CodeReady Workspaces in a restricted enviroment".
- The **crwctl** 2.4 management tool is available. See Section 3.2.1, "Installing the crwctl CLI management tool".

5.3.1. Understanding network connectivity in restricted environments

CodeReady Workspaces requires that each OpenShift Route created for CodeReady Workspaces is accessible from inside the OpenShift cluster. These CodeReady Workspaces components have a OpenShift Route: **codeready-workspaces-server**, **keycloak**, **devfile-registry**, **plugin-registry**.

Consider the network topology of the environment to determine how best to accomplish this.

> **Example 5.1. Network owned by a company or an organization, disconnected from the public Internet**
>
> The network administrators must ensure that it is possible to route traffic bound from the cluster to OpenShift Route host names.

> **Example 5.2. Private subnetwork in a cloud provider**
>
> Create a proxy configuration allowing the traffic to leave the node to reach an external-facing Load Balancer.

5.3.2. Preparing a private registry

Prerequisites

- The **oc** tool is available.
- The **skopeo** tool, version 0.1.40 or later, is available.
- The **podman** tool is available.
- An image registry accessible from the OpenShift cluster and supporting the format of the V2 image manifest, schema version 2. Ensure you can push to it from a location having, at least temporarily, access to the internet.

Table 5.1. Placeholders used in examples

<source-image>	Full coordinates of the source image, including registry, organization, and digest.
<target-registry>	Host name and port of the target container-image registry.
<target-organization>	Organization in the target container-image registry
<target-image>	Image name and digest in the target container-image registry.
<target-user>	User name in the target container-image registry.
<target-password>	User password in the target container-image registry.

Procedure

1. Log into the internal image registry:

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

TIP

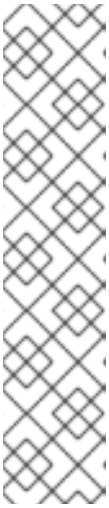
If you meet an error, such as **x509: certificate signed by unknown authority**, when attempting to push to the internal registry, try one of these workarounds:

- add the OpenShift cluster's certificate to **/etc/containers/certs.d/*<target-registry>***
- add the registry as an insecure registry by adding the following lines to the Podman configuration file located at **/etc/containers/registries.conf**:

```
[registries.insecure]
registries = ['<target-registry>']
```

2. Copy images without changing their digest. Repeat this step for every image in the following table:

```
$ skopeo copy --all docker://<source-image> \
  docker://<target-registry>/<target-organization>/<target-image>
```



NOTE

Table 5.2. Understanding the usage of the container-images from the prefix or keyword they include in their name

Usage	Prefix or keyword
Essential	not **stacks-**, **plugin-**, or **-openj-**
Workspaces	**stacks-**, **plugin-**
Z and Power	**-openj-**

Table 5.3. Images to copy in the private registry

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:89763ddec38a5925a052fa7ea75fc5a0db39124cada1e2d33b6eba3e32e8a7c6	**crw-2-rhel8-operator@sha256:89763ddec38a5925a052fa7ea75fc5a0db39124cada1e2d33b6eba3e32e8a7c6**
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:7702adb0ed28b635e45804e87fe5dd98bdd3aa766fed7845a8ce509b91c22e36	**devfileregistry-rhel8@sha256:7702adb0ed28b635e45804e87fe5dd98bdd3aa766fed7845a8ce509b91c22e36**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:8afecd5b0edc7734532ee76ff9eac1fc4814d8aaa6c9be440a2a88a20c014e4e	**jwtproxy-rhel8@sha256:8afecd5b0edc7734532ee76ff9eac1fc4814d8aaa6c9be440a2a88a20c014e4e**
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:c9bebc895e5fa5a0bd4ecaedfd5384ab75a45a96b6314ba5d4a6f4c1e8e109f9	**machineexec-rhel8@sha256:c9bebc895e5fa5a0bd4ecaedfd5384ab75a45a96b6314ba5d4a6f4c1e8e109f9**
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:27a71612f9bd3bee77adb4e164c44c61cf5085458d592215b2fe74c55d11abc6	**plugin-java11-openj9-rhel8@sha256:27a71612f9bd3bee77adb4e164c44c61cf5085458d592215b2fe74c55d11abc6**
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:e9deebbc320d28a2f425e858ed3dcf87fc67a40f6654d6eb7c2b6feea022b7d6	**plugin-java11-rhel8@sha256:e9deebbc320d28a2f425e858ed3dcf87fc67a40f6654d6eb7c2b6feea022b7d6**
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:14f2774e92b70d85280e506f81e2ea9a89c26490fd53a4421df8a694bd944d2d	**plugin-java8-openj9-rhel8@sha256:14f2774e92b70d85280e506f81e2ea9a89c26490fd53a4421df8a694bd944d2d**
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:d04f70c8340abaee1a282b77158d054f4faf2225bc17c79aafb413396c367782	**plugin-java8-rhel8@sha256:d04f70c8340abaee1a282b77158d054f4faf2225bc17c79aafb413396c367782**
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:d87aed64704369a50d1e54a57815b699f74d4efad1401d1a638808e655a37e48	**plugin-kubernetes-rhel8@sha256:d87aed64704369a50d1e54a57815b699f74d4efad1401d1a638808e655a37e48**
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:9c43a02b0dd0f66744359c5ccdb1f1780ecd92c3dc31b14d73b553ba763af8ab	**plugin-openshift-rhel8@sha256:9c43a02b0dd0f66744359c5ccdb1f1780ecd92c3dc31b14d73b553ba763af8ab**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:d0eebf2c8b460adb75dc6bc5200aa9fd40d030b7b17c6b1c3b9d3c879f4652ee	**pluginbroker-artifacts-rhel8@sha256:d0eebf2c8b460adb75dc6bc5200aa9fd40d030b7b17c6b1c3b9d3c879f4652ee**
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:cff23432d1d397bbbc7df65be9d6ddf4a97a3ef1801708bb7bb7d2fa72dbcce3	**pluginbroker-metadata-rhel8@sha256:cff23432d1d397bbbc7df65be9d6ddf4a97a3ef1801708bb7bb7d2fa72dbcce3**
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:9f37917122c20fc83e6558a5484efab42650958b513a22920f449f948e50cd51	**pluginregistry-rhel8@sha256:9f37917122c20fc83e6558a5484efab42650958b513a22920f449f948e50cd51**
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:63bf304cd04576048012693e7e8544a5a703790f99551554a75798bc799b112b	**server-rhel8@sha256:63bf304cd04576048012693e7e8544a5a703790f99551554a75798bc799b112b**
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:56543cfeeeac030821557ac4937db40f6845e874193c79c30267a680f9b2cbe7	**stacks-cpp-rhel8@sha256:56543cfeeeac030821557ac4937db40f6845e874193c79c30267a680f9b2cbe7**
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:13628110b96de0e516ff2dfa29cdcaee64cd8f8978052c8160c294c332dba9f0	**stacks-dotnet-rhel8@sha256:13628110b96de0e516ff2dfa29cdcaee64cd8f8978052c8160c294c332dba9f0**
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:fef91718ccebc4cd9b89999f6b5df83bf3d60fce657f6f44eda092100549af2c	**stacks-golang-rhel8@sha256:fef91718ccebc4cd9b89999f6b5df83bf3d60fce657f6f44eda092100549af2c**
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:b75f498954fbe858c74f80a89d132ba3560f40c0f697b0cd9550ed5663078ef6	**stacks-php-rhel8@sha256:b75f498954fbe858c74f80a89d132ba3560f40c0f697b0cd9550ed5663078ef6**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:942e1e6328169508e3fff8fd96c575d7a423339ced17dbf5813d61d1971adaef	**theia-endpoint-rhel8@sha256:942e1e6328169508e3fff8fd96c575d7a423339ced17dbf5813d61d1971adaef**
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:78edc9f75680cbe7f63774d9dfbbc505401486a73c8e420380e1d3078bdf9f2a	**theia-rhel8@sha256:78edc9f75680cbe7f63774d9dfbbc505401486a73c8e420380e1d3078bdf9f2a**
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:d6a7bdbf4726fe0e0e54c0bce9b2257bbd2a165c37cb4ec68e1f994716ffb15c	**eap-xp1-openj9-11-openshift-rhel8@sha256:d6a7bdbf4726fe0e0e54c0bce9b2257bbd2a165c37cb4ec68e1f994716ffb15c**
registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833	**eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833**
registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0	**eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0**
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:8e6c7874247053df431c25552c6e2edb050b2627ae21907149f419e0d9909135	**sso74-openj9-openshift-rhel8@sha256:8e6c7874247053df431c25552c6e2edb050b2627ae21907149f419e0d9909135**
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:ec6801343eb1ca085154d8d7481552f2e9debc414125413d25e42216aa5922af	**sso74-openshift-rhel8@sha256:ec6801343eb1ca085154d8d7481552f2e9debc414125413d25e42216aa5922af**
registry.redhat.io/rhel8/postgresql-96@sha256:fdc2398a25530547354714f2538c691d91b700e0cedef5361a3e7d96ddfd4e11	**postgresql-96@sha256:fdc2398a25530547354714f2538c691d91b700e0cedef5361a3e7d96ddfd4e11**

<source-image>	*<target-image>*
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	**mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73**
registry.redhat.io/ubi8-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6	**ubi8-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6**

3. Verify the images have the same digests:

```
$ skopeo inspect docker://<source-image>
$ skopeo inspect docker://<target-registry>/<target-organization>/<target-image>
```

4. Set the digests explicitly when different:

```
$ skopeo copy --all docker://<source-image> \
  docker://<target-registry>/<target-organization>/<target-image>
```

Additional resources

- To find the sources of the images list, see the values of the **relatedImages** attribute in the CodeReady Workspaces Operator ClusterServiceVersion sources.

5.3.3. Upgrading CodeReady Workspaces using the CLI management tool in restricted environment

This section describes how to upgrade Red Hat CodeReady Workspaces using the CLI management tool in restricted environment.

Prerequisites

- An administrative account on an OpenShift instance.
- A running instance version 2.3 of Red Hat CodeReady Workspaces, installed using the CLI management tool on the same instance of OpenShift, with the crwctl **--installer operator** method, in the ***<openshift-workspaces>*** project. See Section 3.3, “Installing CodeReady Workspaces in a restricted enviroment”.
- Essential container images are available to the CodeReady Workspaces server running in the cluster. See Section 5.3.2, “Preparing a private registry”.
- The **crwctl** 2.4 management tool is available. See Section 3.2.1, “Installing the crwctl CLI management tool”.

Procedure

1. In all running workspaces in the CodeReady Workspaces 2.3 instance, save and push changes back to the Git repositories.

2. Stop all workspaces in the CodeReady Workspaces 2.3 instance.

3. Run the following command:

```
$ crwctl server:update --che-operator-image=<image-registry>/<organization>/crw-2-rhel8-operator:2.4 -n openshift-workspaces
```

- *<image-registry>*: A host name and a port of the container-image registry accessible in the restricted environment.

- *<organization>*: An organization of the container-image registry. See: Section 5.3.2, “Preparing a private registry”.

Verification steps

1. Navigate to the CodeReady Workspaces instance.

2. The 2.4 version number is visible at the bottom of the page.

NOTE

For slow systems or internet connections, add the **--k8spodwaittimeout=1800000** flag option to the **crwctl server:update** command to extend the Pod timeout period to 1800000 ms or longer.

CHAPTER 6. UNINSTALLING CODEREADY WORKSPACES

This section describes uninstallation procedures for Red Hat CodeReady Workspaces. The uninstallation process leads to a complete removal of CodeReady Workspaces-related user data. The method previously used to install the CodeReady Workspaces instance determines the uninstallation method.

- For CodeReady Workspaces installed using OperatorHub, for the OpenShift Web Console method see Section 6.1, "Uninstalling CodeReady Workspaces after OperatorHub installation using the OpenShift web console".
- For CodeReady Workspaces installed using OperatorHub, for the CLI method see Section 6.2, "Uninstalling CodeReady Workspaces after OperatorHub installation using OpenShift CLI".
- For CodeReady Workspaces installed using crwctl, see Section 6.3, "Uninstalling CodeReady Workspaces after crwctl installation"

6.1. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION 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

1. Navigate to the OpenShift web console and select the Administrator Perspective.
2. In the **Home > Projects** section, navigate to the project containing the CodeReady Workspaces instance.

   TIP

   The default project name is *<openshift-workspaces>*.

3. In the **Operators > Installed Operators** section, click **Red Hat CodeReady Workspaces** in the list of installed operators.
4. In the **Red Hat CodeReady Workspaces Cluster** tab, click the displayed Red Hat CodeReady Workspaces Cluster, and select the **Delete cluster** option in the **Actions** drop-down menu on the top right.

   TIP

   The default Red Hat CodeReady Workspaces Cluster name is *<red-hat-codeready-workspaces>*.

5. In the **Operators > Installed Operators** section, click **Red Hat CodeReady Workspaces** in the list of installed operators and select the **Uninstall Operator** option in the **Actions** drop-down menu on the top right.

6. In the **Home > Projects** section, navigate to the project containing the CodeReady Workspaces instance, and select the **Delete Project** option in the **Actions** drop-down menu on the top right.

6.2. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION USING OPENSHIFT CLI

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.
- The **oc** tool is available.

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.v2.4**:

```
$ oc get csv
NAME                DISPLAY       VERSION   REPLACES              PHASE
red-hat-codeready-workspaces.v2.4   Red Hat CodeReady Workspaces   2.4     red-hat-codeready-workspaces.v2.3   Succeeded
```

6. Delete the CodeReady Workspaces CSV:

```
$ oc delete csv red-hat-codeready-workspaces.v2.4
clusterserviceversion.operators.coreos.com "red-hat-codeready-workspaces.v2.4" deleted
```

6.3. 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.

Prerequisites

- The **crwctl** tool is available.
- The **oc** tool is available.
- The **crwctl** tool installed the CodeReady Workspaces instance on OpenShift.

Procedure

1. Sign in to the OpenShift cluster:

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

2. Obtain the name of the CodeReady Workspaces namespace:

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

3. Remove the CodeReady Workspaces instance from the *<namespace>* project:

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

TIP

When the name of the project containing the CodeReady Workspaces instance is **openshift-workspaces**, the **-n** argument is not necessary.

4. Remove the *<namespace>* project:

```
$ oc delete namespaces <namespace>
```