



Red Hat CodeReady Workspaces 2.5

Installation Guide

Installing Red Hat CodeReady Workspaces 2.5

Red Hat CodeReady Workspaces 2.5 Installation Guide

Installing Red Hat CodeReady Workspaces 2.5

Robert Kratky
rkratky@redhat.com

Michal Maléř
mmaler@redhat.com

Fabrice Flore-Thébault
ffloreth@redhat.com

Yana Hontyk
yhontyk@redhat.com

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Information for administrators installing Red Hat CodeReady Workspaces.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
CHAPTER 1. SUPPORTED PLATFORMS	5
CHAPTER 2. CONFIGURING THE CODEREADY WORKSPACES INSTALLATION	6
2.1. UNDERSTANDING THE CHECLUSTER CUSTOM RESOURCE	6
2.2. CHECLUSTER CUSTOM RESOURCE FIELDS REFERENCE	6
CHAPTER 3. INSTALLING CODEREADY WORKSPACES	16
3.1. INSTALLING CODEREADY WORKSPACES ON OPENSIFT 4 USING OPERATORHUB	16
3.1.1. Creating a project in OpenShift Web Console	16
3.1.2. Installing the Red Hat CodeReady Workspaces Operator	16
3.1.3. Creating an instance of the Red Hat CodeReady Workspaces Operator	17
3.2. INSTALLING CODEREADY WORKSPACES ON OPENSIFT CONTAINER PLATFORM 3.11	18
3.2.1. Installing the crwctl CLI management tool	18
3.2.2. Installing CodeReady Workspaces on OpenShift 3 using the Operator	19
3.3. INSTALLING CODEREADY WORKSPACES IN A RESTRICTED ENVIRONMENT	21
3.3.1. Installing CodeReady Workspaces in a restricted environment using OperatorHub	21
3.3.2. Installing CodeReady Workspaces in a restricted environment using CLI management tool	22
3.3.2.1. Preparing an private registry	22
3.3.2.2. Preparing CodeReady Workspaces Custom Resource for restricted environment	29
3.3.2.2.1. Downloading the default CheCluster Custom Resource	29
3.3.2.2.2. Customizing the CheCluster Custom Resource for restricted environment	29
3.3.2.3. Starting CodeReady Workspaces installation in a restricted environment using CodeReady Workspaces CLI management tool	30
3.3.3. Preparing CodeReady Workspaces Custom Resource for installing behind a proxy	30
CHAPTER 4. CONFIGURING CODEREADY WORKSPACES	32
4.1. ADVANCED CONFIGURATION OPTIONS FOR THE CODEREADY WORKSPACES SERVER COMPONENT	32
4.1.1. Understanding CodeReady Workspaces server advanced configuration using the Operator	32
4.1.2. CodeReady Workspaces server component system properties reference	33
4.2. CONFIGURING PROJECT STRATEGIES	68
4.2.1. One project per workspace strategy	69
4.2.2. One project for all workspaces strategy	69
4.2.3. One project per user strategy	69
4.2.4. Allowing user-defined workspace projects	70
4.2.5. Handling incompatible usernames or user IDs	70
4.2.6. Pre-creating projects for users	70
4.3. RUNNING MORE THAN ONE WORKSPACE AT A TIME	71
4.4. CONFIGURING WORKSPACE EXPOSURE STRATEGIES	72
4.4.1. Workspace exposure strategies	72
4.4.1.1. Multi-host strategy	72
4.4.1.2. Single-host strategy	73
4.4.1.2.1. devfile endpoints: single-host	73
4.4.1.2.2. devfile endpoints: multi-host	73
4.4.2. Security considerations	74
4.4.2.1. JSON web token (JWT) proxy	74
4.4.2.2. Secured plug-ins and editors	74
4.4.2.3. Secured container-image components	74
4.4.2.4. Cross-site request forgery attacks	75
4.4.2.5. Phishing attacks	75

4.5. CONFIGURING WORKSPACES NODESELECTOR	75
4.6. CONFIGURING RED HAT CODEREADY WORKSPACES SERVER HOSTNAME	75
4.7. CONFIGURING LABELS FOR OPENSIFT INGRESS	77
4.8. CONFIGURING LABELS FOR OPENSIFT ROUTE	77
4.9. DEPLOYING CODEREADY WORKSPACES WITH SUPPORT FOR GIT REPOSITORIES WITH SELF-SIGNED CERTIFICATES	78
4.10. INSTALLING CODEREADY WORKSPACES USING STORAGE CLASSES	79
4.11. CONFIGURING STORAGE TYPES	83
4.11.1. Persistent storage	83
4.11.2. Ephemeral storage	83
4.11.3. Asynchronous storage	84
4.11.4. Configuring storage type defaults for CodeReady Workspaces dashboard	85
4.11.5. Idling asynchronous storage Pods	87
4.12. IMPORTING UNTRUSTED TLS CERTIFICATES TO CODEREADY WORKSPACES	87
4.12.1. At CodeReady Workspaces installation time	87
4.12.2. On already-running CodeReady Workspaces installations	88
4.12.3. Verification at the CodeReady Workspaces installation level	90
4.12.4. Verification at the workspace level	90
4.13. SWITCHING BETWEEN EXTERNAL AND INTERNAL WAYS IN INTER-COMPONENT COMMUNICATION	92
4.14. SETTING UP THE RH-SSO CODEREADY-WORKSPACES-USERNAME-READONLY THEME FOR THE RED HAT CODEREADY WORKSPACES LOGIN PAGE	92
CHAPTER 5. UPGRADING CODEREADY WORKSPACES	94
5.1. UPGRADING CODEREADY WORKSPACES USING OPERATORHUB	94
5.2. UPGRADING CODEREADY WORKSPACES USING THE CLI MANAGEMENT TOOL	94
5.3. UPGRADING CODEREADY WORKSPACES USING THE CLI MANAGEMENT TOOL IN RESTRICTED ENVIRONMENT	95
5.3.1. Understanding network connectivity in restricted environments	95
5.3.2. Preparing an private registry	96
5.3.3. Upgrading CodeReady Workspaces using the CLI management tool in restricted environment	102
CHAPTER 6. UNINSTALLING CODEREADY WORKSPACES	104
6.1. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION USING THE OPENSIFT WEB CONSOLE	104
6.2. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION USING OPENSIFT CLI	105
6.3. UNINSTALLING CODEREADY WORKSPACES AFTER CRWCTL INSTALLATION	106

MAKING OPEN SOURCE MORE INCLUSIVE

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

CHAPTER 1. SUPPORTED PLATFORMS

This section describes the availability and the supported installation methods of CodeReady Workspaces 2.5 on OpenShift Container Platform.

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 2.5 on OpenShift Container Platform

Platform	Architecture	Deployment method
OpenShift Container Platform 3.11	AMD64 and Intel 64 (x86_64)	crwctl
OpenShift Container Platform 4.5	AMD64 and Intel 64 (x86_64)	OperatorHub
OpenShift Container Platform 4.5	IBM Z (s390x)	OperatorHub
OpenShift Container Platform 4.5	IBM Power Systems (ppc64le)	OperatorHub
OpenShift Container Platform 4.6	AMD64 and Intel 64 (x86_64)	OperatorHub
OpenShift Container Platform 4.6	IBM Power Systems (ppc64le)	OperatorHub



NOTE

- On OpenShift Container Platform 4.5 and 4.6, 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 <code> </code> 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 Resourceauth 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
identityProvider ClientId	omit	Name of an Identity provider (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 .
identityProvider ImagePullPolicy	Always for nightly or latest images, and IfNotPresent in other cases	Overrides the image pull policy used in the Identity Provider (RH-SSO) deployment.
identityProvider Image	omit	Overrides the container image used in the Identity Provider (RH-SSO) deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator.
identityProvider Password	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.
identityProvider PostgresPassword	the Operator sets the value automatically	Password for The Identity Provider (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).
identityProvider Realm	omit	Name of an Identity provider (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.
identityProvider URL	the Operator sets the value automatically	Instructs the Operator to deploy a dedicated Identity Provider (RH-SSO instance). Public URL of the Identity Provider server (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 OpenShifttoAuth 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
openShifttoAuth	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: <code>common</code> (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,omite mpty	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
openShifttoAuthProvisioned	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 OPENSIFT 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: 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 → **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**.

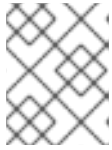
3.1.3. Creating an instance of the Red Hat CodeReady Workspaces Operator

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/2.5/html-single/end-user_guide/index#navigating-codeready-workspaces-using-the-dashboard_crw.
- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.5/html-single/administration_guide/index#viewing-the-state-of-the-codeready-workspaces-cluster-deployment-using-openshift-4-cli-tools_crw.
- It is possible to use the **crwctl** utility script for deploying CodeReady Workspaces on OpenShift Container Platform and OpenShift Dedicated versions 4.6. 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 OPENSIFT 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

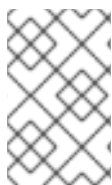
1. Navigate to <https://developers.redhat.com/products/codeready-workspaces/download>.
2. Download the CodeReady Workspaces CLI management tool archive for version 2.5.
3. Extract the archive to a folder, such as **\$(HOME)/crwctl** or **/opt/crwctl**.
4. Run the **crwctl** executable from the extracted 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.



NOTE

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:deploy -n <namespace> -p openshift
```

- In the default project called `workspaces`:

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

Verification steps

1. The output of the previous command ends with:

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

2. Navigate to the CodeReady Workspaces cluster instance: **`https://codeready-
<openshift_deployment_name>.<domain_name>`**.

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": "  
<workspaces>", "name": "codeready-operator"} }]
```


3.3. INSTALLING CODEREADY WORKSPACES IN A RESTRICTED ENVIRONMENT

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 an 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>
```



WARNING

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>
```

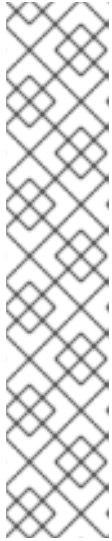
**NOTE**

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 -openj9-
Workspaces	stacks- , plugin-
IBM Z and IBM Power Systems	-openj9-

Table 3.3. Images to copy in the private registry

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac	configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55	devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7	jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7

<source-image>	<target-image>
<p>registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd</p>	<p>machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>	<p>plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>	<p>plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>	<p>plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99</p>	<p>plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>	<p>plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>	<p>plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>	<p>plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763b431744790a89acd5c6d197316b694693b58c19770	plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763b431744790a89acd5c6d197316b694693b58c19770
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f990e9fab9baa47f659ef9450b7881072e3cb40399	plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f990e9fab9baa47f659ef9450b7881072e3cb40399
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a950cc3cbe21ed7301c65f84bb509fa2b6e71fa81d	plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a950cc3cbe21ed7301c65f84bb509fa2b6e71fa81d
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6ccdb8c303f32d046f7eccf9f452eb30e615f2a0bf0e	pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6ccdb8c303f32d046f7eccf9f452eb30e615f2a0bf0e
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505a0fa6f96969cf8cc4aa8c4db95b40626f95854a525	pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505a0fa6f96969cf8cc4aa8c4db95b40626f95854a525
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:c9f48f247cff27280587aeff54cea5d8a27e0eb55c99a73726cd7d575db7fbcc	pluginregistry-rhel8@sha256:c9f48f247cff27280587aeff54cea5d8a27e0eb55c99a73726cd7d575db7fbcc
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57	server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb	stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb

<source-image>	<target-image>
<p>registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2</p>	<p>stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2</p>
<p>registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0</p>	<p>stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0</p>
<p>registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db</p>	<p>stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db</p>
<p>registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3</p>	<p>theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3</p>
<p>registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0</p>	<p>theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0</p>
<p>registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed</p>	<p>traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed</p>
<p>registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce</p>	<p>eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce</p>
<p>registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce</p>	<p>eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce</p>

<source-image>	<target-image>
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c 1992663258c0016c80247f507fd1c39cf9a73 da833	eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c 1992663258c0016c80247f507fd1c39cf9a73 da833
registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b4 6ade182d0f981362f36b7e6fb9c7d8531ac6 39fe0	eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b4 6ade182d0f981362f36b7e6fb9c7d8531ac6 39fe0
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f	sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f	sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:c0045cd676e06eb17083a4 4c4b90b29b11ddb40e1fb6a7b651384cf09 60f5158	sso74-openshift-rhel8@sha256:c0045cd676e06eb17083a4 4c4b90b29b11ddb40e1fb6a7b651384cf09 60f5158
registry.redhat.io/rhel8/postgresql-96@sha256:5b5bf623d89deda89250f422d 352b122bce9533b902b5474f9c63a9facc7a 6f1	postgresql-96@sha256:5b5bf623d89deda89250f422d 352b122bce9533b902b5474f9c63a9facc7a 6f1
registry.redhat.io/rhsc1/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d 401b720600fd9059a3d8eefea6f3b2ffa721c 0dc73	mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d 401b720600fd9059a3d8eefea6f3b2ffa721c 0dc73

<source-image>	<target-image>
registry.redhat.io/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6	ubi8ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6

Verification steps

- Verify the images have the same digests:

```
$ skopeo inspect docker://<source-image>
$ skopeo inspect 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 an 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: '<target-registry>'
airGapContainerRegistryOrganization: '<target-organization>'
# [...]
```

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

This section 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 ${OPENSIFT_API_URL} --username ${OPENSIFT_USERNAME} \
--password ${OPENSIFT_PASSWORD}
```

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

```
$ crwctl server:start \
--che-operator-image=<target-registry>/<target-organization>/crw-2-rhel8-operator:2.5 \
--che-operator-cr-yaml=org_v1_che_cr.yaml
```



NOTE

For slow systems or internet connections, add the **--k8spodwaittimeout=1800000** flag 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>'
# [...]
```

- 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, “Configuring labels for OpenShift Ingress”](#)
- [Section 4.8, “Configuring labels for OpenShift Route”](#)
- [Section 4.9, “Deploying CodeReady Workspaces with support for Git repositories with self-signed certificates”](#)
- [Section 4.10, “Installing CodeReady Workspaces using storage classes”](#)
- [Section 4.11, “Configuring storage types”](#)
- [Section 4.12, “Importing untrusted TLS certificates to CodeReady Workspaces”](#)
- [Section 4.13, “Switching between external and internal ways in inter-component communication”](#)
- [Section 4.14, “Setting up the RH-SSO codeready-workspaces-username-readonly theme for the Red Hat CodeReady Workspaces login page”](#)

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
# [...]
spec:
  server:
    # [...]
    customCheProperties:
      CHE_WORKSPACE_DEFAULT_MEMORY_LIMIT_MB: "2048"
    # [...]
```



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 stores 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 interactions and 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 interactions and messaging.
CHE_WORKSPACE_STORAGE	/\${che.home}/workspaces	Projects are synchronized from the CodeReady Workspaces server into the machine running each workspace. This is the directory in the workspace runtime where the 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 OpenShift-type components in a devfile request project PVC creation (Applied in case of 'unique' and 'per workspace' PVC strategy. In case of the 'common' PVC strategy, it is rewritten with the value of the 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. Provide this value into the machine, for example, as an environment variable. This is to ensure that agent developers can use this directory to back up agent logs.
CHE_WORKSPACE_HTTP_PROXY		Configures proxies used by runtimes powering workspaces.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_HTTPS_PROXY		Configures proxies used by runtimes powering workspaces.
CHE_WORKSPACE_NO_PROXY		Configures proxies used by runtimes powering workspaces.
CHE_TRUSTED_CA_BUNDLES_CONFIGMAP	NULL	When cluster-wide proxy is configured, che-operator creates a special configmap and allows the OpenShift Network Operator to inject the ca-bundle into it. In addition, it adds the CHE_TRUSTED_CA_BUNDLES_CONFIGMAP key with the name of this configmap in the CodeReady Workspaces server configmap (and corresponding environment variable). So, its presence can be used to detect if proxy mode is enabled or not. Do not set this property manually unless required for this purpose.
CHE_WORKSPACE_AUTO_START	true	By default, when users access a workspace with its URL, the workspace automatically starts (if currently stopped). Set this to false to disable this behavior.
CHE_WORKSPACE_POOL_TYPE	fixed	Workspace threads pool configuration. This pool is used for workspace-related operations that require asynchronous execution, for example, starting and stopping. Possible values are fixed and cached .
CHE_WORKSPACE_POOL_EXACT_SIZE	30	This property is ignored when pool type is different from fixed . It configures the exact size of the pool. When set, the multiplier property is ignored. If this property is not set (0 , <0 , NULL), then the pool size equals the number of cores. See also che.workspace.pool.cores_multiplier .

Environment Variable Name	Default value	Description
CHE_WORKSPACE_POOL_CORES__MULTIPLIER	2	This property is ignored when pool type is not set to fixed , che.workspace.pool.exact_size is set. When set, the pool size is N_CORES * multiplier .
CHE_WORKSPACE_PROBE__POOL__SIZE	10	This property specifies how many threads to use for workspace server 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 added to JVMs running in 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 JVMs running agents in 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	Java command-line options added to the JVM running the 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 its environment. Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_DEFAULT_MEMORY_REQUEST_MB	200	RAM request for each container that has no explicit RAM settings in its environment. This amount is allocated when the workspace container is created. This property may not be supported by all infrastructure implementations. Currently it is supported by OpenShift. A memory request exceeding the memory limit is ignored, and only the limit size is used. Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_DEFAULT_CPU_LIMIT_CORES	-1	CPU limit for each container that has no CPU settings in its environment. Specify either in floating point cores number, for example, 0.125 , or using the OpenShift format, integer millicores, for example, 125m . Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_DEFAULT_CPU_REQUEST_CORES	-1	CPU request for each container that has no CPU settings in environment. A CPU request exceeding the CPU limit is ignored, and only limit number is used. Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_SIDECAR_DEFAULT_MEMORY_LIMIT_MB	128	RAM limit and request for each sidecar that has no RAM settings in the CodeReady Workspaces plug-in configuration. Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_SIDECAR_DEFAULT_MEMORY_REQUEST_MB	64	RAM limit and request for each sidecar that has no RAM settings in the CodeReady Workspaces plug-in configuration. Value less or equal to 0 is interpreted as disabling the limit.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_SIDECAR_DEFAULT_CPU_LIMIT_CORES	-1	CPU limit and request default for each sidecar that has no CPU settings in the CodeReady Workspaces plug-in configuration. Specify either in floating point cores number, for example, 0.125 , or using the OpenShift format, integer millicores, for example, 125m . Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_SIDECAR_DEFAULT_CPU_REQUEST_CORES	-1	CPU limit and request default for each sidecar that has no CPU settings in the CodeReady Workspaces plug-in configuration. Specify either in floating point cores number, for example, 0.125 , or using the OpenShift format, integer millicores, for example, 125m . Value less or equal to 0 is interpreted as disabling the limit.
CHE_WORKSPACE_SIDECAR_IMAGE_PULL_POLICY	Always	Defines image-pulling strategy for sidecars. Possible values are: Always, Never, IfNotPresent . For any other value, Always is assumed for images with the :latest tag, or IfNotPresent for all other cases.
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.
CHE_WORKSPACE_ACTIVITY_CLEANUP_SCHEDULER_INITIAL_DELAY_S	60	The delay after server startup to start the first activity clean up job.

Environment Variable Name	Default value	Description
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	Period of stopped temporary workspaces cleanup job execution.
CHE_WORKSPACE_SERVER_PING_SUCCESS_THRES_HOLD	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,intellij	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	<code>\${che.home}/templates</code>	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	<code>/error-oauth</code>	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	You can setup GitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_GITHUB_AUTH_URI	<code>https://github.com/login/oauth/authorize</code>	You can setup GitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_GITHUB_TOKEN_URI	<code>https://github.com/login/oauth/access_token</code>	You can setup GitHub OAuth to automate authentication to remote repositories. You need to first register this application with GitHub OAuth.
CHE_OAUTH_GITHUB_REDIRECTURIS	<code>http://localhost:\${CHE_PORT}/api/oauth/callback</code>	You can setup GitHub 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_OPENSIFT_CLIENTID	NULL	Configuration of OpenShift OAuth client. Used to obtain OpenShift OAuth token.
CHE_OAUTH_OPENSIFT_CLIENTSECRET	NULL	Configuration of OpenShift OAuth client. Used to obtain OpenShift OAuth token.
CHE_OAUTH_OPENSIFT_OAUTH_ENDPOINT	NULL	Configuration of OpenShift OAuth client. Used to obtain OpenShift OAuth token.
CHE_OAUTH_OPENSIFT_VERIFY_TOKEN_URL	NULL	Configuration of OpenShift OAuth 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_ASYNCCHRONOUS	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_ASYNCCHRONOUS_POOL_SIZE	20	Quantity of asynchronous requests which may be processed at the same time
ORG_EVERREST_ASYNCCHRONOUS_QUEUE_SIZE	500	Size of queue. If asynchronous request can't be processed after consuming it will be added in queue.

Environment Variable Name	Default value	Description
ORG_EVERREST_ASYNC HRONOUS_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_ASYNC HRONOUS_CACHE_SIZE	1024	Size of cache for waiting, running and ended request.
ORG_EVERREST_ASYNC HRONOUS_SERVICE_PATH	/async/	Path to asynchronous service
DB_SCHEMA_FLYWAY_BAS ELINE_ENABLED	true	DB initialization and migration configuration
DB_SCHEMA_FLYWAY_BAS ELINE_VERSION	5.0.0.8.1	DB initialization and migration configuration
DB_SCHEMA_FLYWAY_SCRIP TS_PREFIX		DB initialization and migration configuration
DB_SCHEMA_FLYWAY_SCRIP TS_SUFFIX	.sql	DB initialization and migration configuration
DB_SCHEMA_FLYWAY_SCRIP TS_VERSION_SEPARATO R	---	DB initialization and migration configuration
DB_SCHEMA_FLYWAY_SCRIP TS_LOCATIONS	classpath:che-schema	DB initialization and migration configuration

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		Configuration of OpenShift client that Infra will use

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_SERVER_STRATEGY	multi-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_SINGLEHOST_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. - 'gateway': Exposes servers using reverse-proxy gateway.
CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE_ENDPOINT_EXPOSURE	multi-host	Defines the way how to expose devfile endpoints, thus end-user's applications, in single-host server strategy. They can either follow the single-host strategy and be exposed on subpaths, or they can be exposed on subdomains. - 'multi-host': expose on subdomains - 'single-host': expose on subpaths
CHE_INFRA_KUBERNETES_SINGLEHOST_GATEWAY_CONFIGMAP_LABELS	app=che,component=che-gateway-config	Defines labels which will be set to ConfigMaps configuring single-host gateway.
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

Environment Variable Name	Default value	Description
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.
CHE_INFRA_KUBERNETES_NAMESPACE_CREATION_ALLOWED	true	Indicates whether CodeReady Workspaces server is allowed to create namespaces/projects for user workspaces, or they're intended to be created manually by cluster administrator. This property is also used by the OpenShift infra.
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

Environment Variable Name	Default value	Description
CHE_INFRA_KUBERNETES_NAMESPACE_ALLOW_USE_R_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
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

Environment Variable Name	Default value	Description
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 the workspace needs e.g backup projects, logs etc or disable it.
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 <code>che.infra.kubernetes.pvc.strategy</code> 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	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_INGRESS_ANNOTATIONS_JSON	NULL	<p>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 <code>che.infra.kubernetes.ingress.path.rewrite_transform</code> 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:</p> <pre>{'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'}</pre> <p>and the <code>che.infra.kubernetes.ingress.path.rewrite_transform</code> should be set to <code>'%s(.*)'</code> For nginx ingress controller older than 0.22.0, the <code>rewrite-target</code> should be set to merely <code>'/'</code> and the <code>path transform</code> to <code>'%s'</code> (see the <code>che.infra.kubernetes.ingress.path.rewrite_transform</code> property). Please consult the nginx ingress controller documentation for the explanation of how the ingress controller uses the regular</p>

Environment Variable Name	Default value	Description
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_INGRESS_LABELS	NULL	Additional labels to add into every Ingress created by CodeReady Workspaces server to allow clear identification.
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	Defines security context for pods that will be created by OpenShift Infra This is ignored by OpenShift infra
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.

Environment Variable Name	Default value	Description
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	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_CONNECTION_POOL_MAX_IDLE	5	Max number of idle connections in the connection pool of the OpenShift-client shared http client
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

Environment Variable Name	Default value	Description
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
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_OPENSIFT_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_OPENSIFT_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_OPENSIFT_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_OPENSIFT_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_INFRA_OPENSIFT_ROUTE_LABELS	NULL	Additional labels to add into every Route created by CodeReady Workspaces server to allow clear identification.
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:v3.4.0	Docker image of CodeReady Workspaces plugin broker app that resolves workspace tooling configuration and copies plugins dependencies to a workspace Note these images are overridden by the CodeReady Workspaces Operator by default; changing the images here will not have an effect if CodeReady Workspaces is installed via Operator.
CHE_WORKSPACE_PLUGIN__BROKER_ARTIFACTS_IMAGE	quay.io/eclipse/che-plugin-artifacts-broker:v3.4.0	Docker image of CodeReady Workspaces plugin broker app that resolves workspace tooling configuration and copies plugins dependencies to a workspace Note these images are overridden by the CodeReady Workspaces Operator by default; changing the images here will not have an effect if CodeReady Workspaces is installed via Operator.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_PLUGIN__BROKER_DEFAULT_MERGE_PLUGINS	false	Configures the default behavior of the plugin brokers when provisioning plugins into a workspace. If set to true, the plugin brokers will attempt to merge plugins when possible (i.e. they run in the same sidecar image and do not have conflicting settings). This value is the default setting used when the devfile does not specify otherwise, via the 'mergePlugins' attribute.
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
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

Environment Variable Name	Default value	Description
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: <ul style="list-style-type: none"> - '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: <ul style="list-style-type: none"> - 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
CHE_SERVER_SECURE_EXPOSER	jwtproxy	Configures in which way secure servers will be protected with authentication. Suitable values: <ul style="list-style-type: none"> - '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.

Environment Variable Name	Default value	Description
CHE_SERVER_SECURE_EX POSER_JWTPROXY_TOKEN _ISSUER	wsmaster	Jwtproxy issuer string, token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE_EX POSER_JWTPROXY_TOKEN _TTL	8800h	Jwtproxyissuer string, token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE_EX POSER_JWTPROXY_AUTH_ LOADER_PATH	/_app/loader.html	Jwtproxyissuerstring, token lifetime and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE_EX POSER_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_EX POSER_JWTPROXY_MEMOR Y_LIMIT	128mb	Jwtproxyissuerstring,tokenlifetim e and optional auth page path to route unsigned requests to.
CHE_SERVER_SECURE_EX POSER_JWTPROXY_CPU__L IMIT	0.5	Jwtproxyissuerstring,tokenlifetim eand 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_PRO CESSOR__MAX__POOL__SI ZE	50	Maximum size of the JSON RPC processing pool in case if pool size would be exceeded message execution will be rejected
CHE_CORE_JSONRPC_PRO CESSOR__CORE__POOL__S IZE	5	Initial json processing pool. Minimum number of threads that used to process major JSON RPC messages.
CHE_CORE_JSONRPC_PRO CESSOR__QUEUE__CAPACI TY	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_MIN_OR_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_MIN_OR_PROCESSOR_CORE_POOL_SIZE	15	Initial json processing pool. Minimum number of threads that used to process minor JSON RPC messages.
CHE_CORE_JSONRPC_MIN_OR_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_OR_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
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
---------------------------	---------------	-------------

Environment Variable Name	Default value	Description
CHE_FACTORY_DEFAULT__EDITOR	eclipse/che-theia/7.20.1	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/7.20.1	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__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/7.20.1	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.

Environment Variable Name	Default value	Description
CHE_WORKSPACE_DEVFILE_DEFAULT_EDITOR_PLUGINS	eclipse/che-machine-exec-plugin/7.20.1	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
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

Environment Variable Name	Default value	Description
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
---------------------------	---------------	-------------

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_RUNNING_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've been added to a Che Organization	Organization notifications subjects and templates
CHE_ORGANIZATION_EMAIL_MEMBER_ADDED_TEMPLATE	st-html-templates/user_added_to_organization	Organization notifications subjects and templates
CHE_ORGANIZATION_EMAIL_MEMBER_REMOVED_SUBJECT	You've been 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_TEMP_LATE	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_END POINT	NULL	URL to access OSO oauth tokens
CHE_KEYCLOAK_GITHUB_E NDPOINT	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__NO NCE	true	Use the OIDC optional nonce feature to increase security.
CHE_KEYCLOAK_JS__ADAP TER__URL	NULL	URL to the Keycloak Javascript adapter we want to use. if set to NULL, then the default used value is <code>#{che.keycloak.auth_server_url}/js/keycloak.js</code> , or <code><che-server>/api/keycloak/OIDCKe ycloak.js</code> if an alternate <code>oidc_provider</code> is used
CHE_KEYCLOAK_OIDC__PR OVIDER	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__FIX ED__REDIRECT__URLS	false	Set to true when using an alternate OIDC provider that only supports fixed redirect Urls This property is ignored when <code>che.keycloak.oidc_provider</code> is NULL
CHE_KEYCLOAK_USERNAM E__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.
CHE_KEYCLOAK_CASCADE_USER_REMOVAL_ENABLED	false	Configuration for enabling removing user from Keycloak server on removing user from CodeReady Workspaces database. By default it's disabled. Can be enabled in some special cases when deleting a user in CodeReady Workspaces database should execute removing related-user from Keycloak. For correct work need to set admin username <code>\${che.keycloak.admin_username}</code> and password <code>\${che.keycloak.admin_password}</code> .
CHE_KEYCLOAK_ADMIN_USERNAME	NULL	Keycloak admin username. Will be used for deleting user from Keycloak on removing user from CodeReady Workspaces database. Make sense only in case <code>\${che.keycloak.cascade_user_removal_enabled}</code> set to 'true'
CHE_KEYCLOAK_ADMIN_PASSWORD	NULL	Keycloak admin password. Will be used for deleting user from Keycloak on removing user from CodeReady Workspaces database. Make sense only in case <code>\${che.keycloak.cascade_user_removal_enabled}</code> set to 'true'

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_OPENSIFT_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
```

4.2.5. Handling incompatible usernames or user IDs

CodeReady Workspaces server automatically checks usernames and IDs for compatibility with OpenShift objects naming convention before creating a project from a template. Incompatible username or IDs are reduced to the nearest valid name by replacing groups of unsuitable symbols with the `-` symbol. To avoid collisions, a random 6-symbol suffix is added and the result is stored in preferences for reuse.

4.2.6. Pre-creating projects for users

To pre-create projects for users, use project labels and annotations.

```
metadata:
  labels:
    app.kubernetes.io/component: workspace
    app.kubernetes.io/part-of: che.eclipse.org
  annotations:
    che.eclipse.org/username: <username> 1
```

- 1 target user's username

To configure the labels, set the **CHE_INFRA_KUBERNETES_NAMESPACE_LABELS** to desired labels. To configure the annotations, set the **CHE_INFRA_KUBERNETES_NAMESPACE_ANNOTATIONS** to desired annotations. See the [Section 4.1.2, "CodeReady Workspaces server component system properties reference"](#) for more details.

IMPORTANT

On OpenShift with OAuth, target user must have **admin** role privileges in target namespace:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: admin
  namespace: <namespace> 1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: <username> 2
```

- 1** pre-created namespace
- 2** target user

On Kubernetes, **che** ServiceAccount must have a cluster-wide **list namespaces** privilege as well as an **admin** role in target namespace.

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, **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 workspaces --type merge \
-p '{ "spec": { "server": {"customCheProperties":
{"CHE_LIMITS_USER_WORKSPACES_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
- **single-host** strategy
 - with the **gateway** subtype

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.

4.4.1.2. Single-host strategy

Single-host strategy have two subtypes with different implementation methods. First subtype is named **native**. This strategy is available and default on Kubernetes, but not on OpenShift, since it uses ingresses for servers exposing. The second subtype named **gateway**, works both on OpenShift, and uses a special pod with reverse-proxy running inside to route requests.

When any of those single-host types is used, all workspaces are deployed to sub-paths of the main CodeReady Workspaces server domain.

This is convenient for TLS-secured CodeReady Workspaces servers because it is sufficient to have a single certificate for the CodeReady Workspaces server, which will cover all the workspace component deployments as well.

There are two ways of exposing the endpoints specified in the devfile. These can be configured using the

CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE__ENDPOINT__EXPOSURE environment variable of the CodeReady Workspaces. This environment variable is only effective with the single-host server strategy and is applicable to all workspaces of all users.

4.4.1.2.1. devfile endpoints: single-host

CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE__ENDPOINT__EXPOSURE: 'single-host'

This single-host configuration exposes the endpoints on subpaths, for example: **https://<che-host>/serverihzmuqqc/go-cli-server-8080**. This limits the exposed components and user applications. Any absolute URL generated on the server side that points back to the server does not work. This is because the server is hidden behind a path-rewriting reverse proxy that hides the unique URL path prefix from the component or user application.

For example, when the user accesses the hypothetical [**https://codeready-<openshift_deployment_name>.<domain_name>/component-prefix-djh3d/app/index.php**] URL, the application sees the request coming to **https://internal-host/app/index.php**. If the application used the host in the URL that it generates in its UI, it would not work because the internal host is different from the externally visible host. However, if the application used an absolute path as the URL (for the example above, this would be **/app/index.php**), such URL would still not work. This is because on the outside, such URL does not point to the application, because it is missing the component-specific prefix.

Therefore, only applications that use relative URLs in their UI work with the single-host workspace exposure strategy.

4.4.1.2.2. devfile endpoints: multi-host

CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE__ENDPOINT__EXPOSURE: 'multi-host'

This single-host configuration exposes the endpoints on subdomains, for example: **http://serverihzmuqqc-go-cli-server-8080.<che-host>**. These endpoints are exposed on an unsecured HTTP port. A dedicated Ingress or Route is used for such endpoints, even with **gateway** single-host setup.

This configuration limits the usability of previews shown directly in the editor page when CodeReady Workspaces is configured with TLS. Since **https** pages allow communication only with secured endpoints, users must open their application previews in another browser tab.

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 workspaces
```

2. Create a tls secret:

```
$ oc create secret tls ${secret} \ 1  
--key ${key_file} \ 2  
--cert ${cert_file} \ 3  
-n 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.

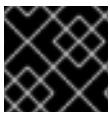
* Valid Redirect URIs ⓘ	https://<hostname>/*	-
	http://<hostname>/*	-
		+
Base URL ⓘ		
Admin URL ⓘ		
Web Origins ⓘ	https://<hostname>	-
	http://<hostname>	-
		+

4.7. CONFIGURING LABELS FOR OPENSIFT INGRESS

This procedure describes how to configure labels for OpenShift Ingress to organize and categorize (scope and select) objects.

Prerequisites

- The **oc** tool is available.
- An instance of CodeReady Workspaces running in OpenShift.



IMPORTANT

Use comma to separate labels: **key1=value1,key2=value2**

Procedure

1. To configure labels for OpenShift Ingress update the Custom Resource with the following commands:

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/server/cheServerIngress/labels", \
"value": "<labels for a codeready-workspaces server ingress>"}'
```

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/auth/identityProviderIngress/labels", \
"value": "<labels for a RH-SSO ingress>"}'
```

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/server/pluginRegistryIngress/labels", \
"value": "<labels for a plugin registry ingress>"}'
```

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/server/devfileRegistryIngress/labels", \
"value": "<labels for a devfile registry ingress>"}'
```

4.8. CONFIGURING LABELS FOR OPENSIFT ROUTE

This procedure describes how to configure labels for OpenShift Route to organize and categorize (scope and select) objects.

Prerequisites

- The **oc** tool is available.
- An instance of CodeReady Workspaces running in OpenShift.



IMPORTANT

Use comma to separate labels: **key1=value1,key2=value2**

Procedure

1. To configure labels for OpenShift Route update the Custom Resource with the following commands:

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/server/cheServerRoute/labels", \
"value": "<labels for a codeready-workspaces server route>"}'
```

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/auth/identityProviderRoute/labels", \
"value": "<labels for a RH-SSO route>"}'
```

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/server/pluginRegistryRoute/labels", \
"value": "<labels for a plugin registry route>"}'
```

```
$ oc patch checluster codeready-workspaces -n workspaces --type=json -p \
'{"op": "replace", "path": "/spec/server/devfileRegistryRoute/labels", \
"value": "<labels for a devfile registry route>"}'
```

4.9. 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 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
```

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



NOTE

CodeReady Workspaces workspace 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:deploy` command**
 - i. Provide the storage class name for the PostgreSQL PVC
Use the **`crwctl server:deploy`** command with the **`--postgres-pvc-storage-class-name`** flag:

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

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

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

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



NOTE

`postgres-pvc-storage-class-name=postgres-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:deploy -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 PostgreSQL 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:deploy -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**.

4.11. CONFIGURING STORAGE TYPES

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

- Persistent
- Ephemeral
- Asynchronous

4.11.1. 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'
```

4.11.2. 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.

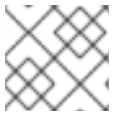
Ephemeral mode provides faster I/O than persistent storage. To enable this storage type, add the following to workspace configuration:

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

Table 4.21. Comparison between I/O of ephemeral (**emptyDir**) and persistent modes on AWS EBS

Command	Ephemeral	Persistent
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

4.11.3. Asynchronous storage



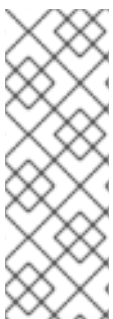
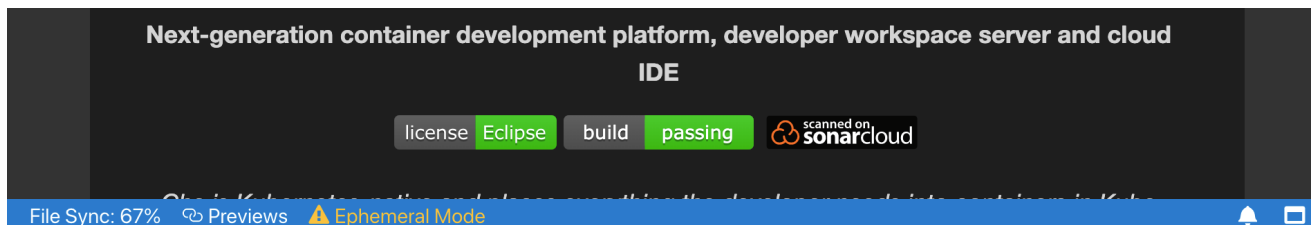
NOTE

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'
```


4.11.4. Configuring storage type defaults for CodeReady Workspaces dashboard

Use the following 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:

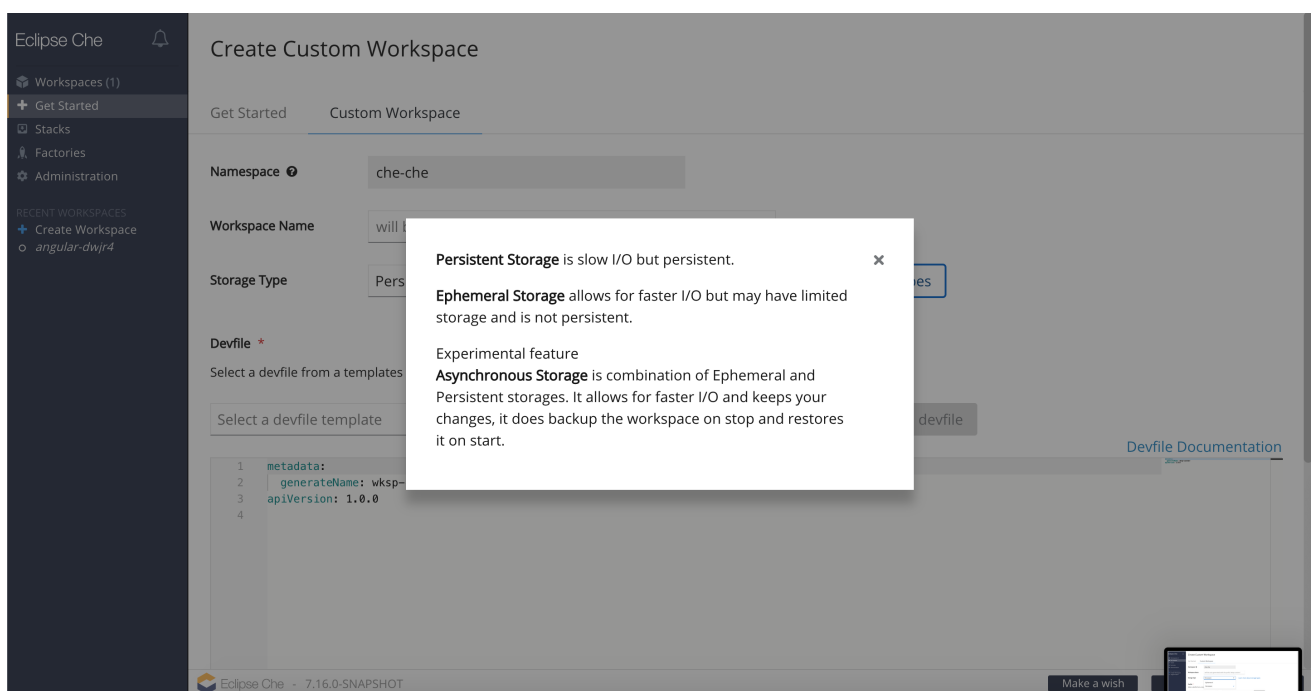
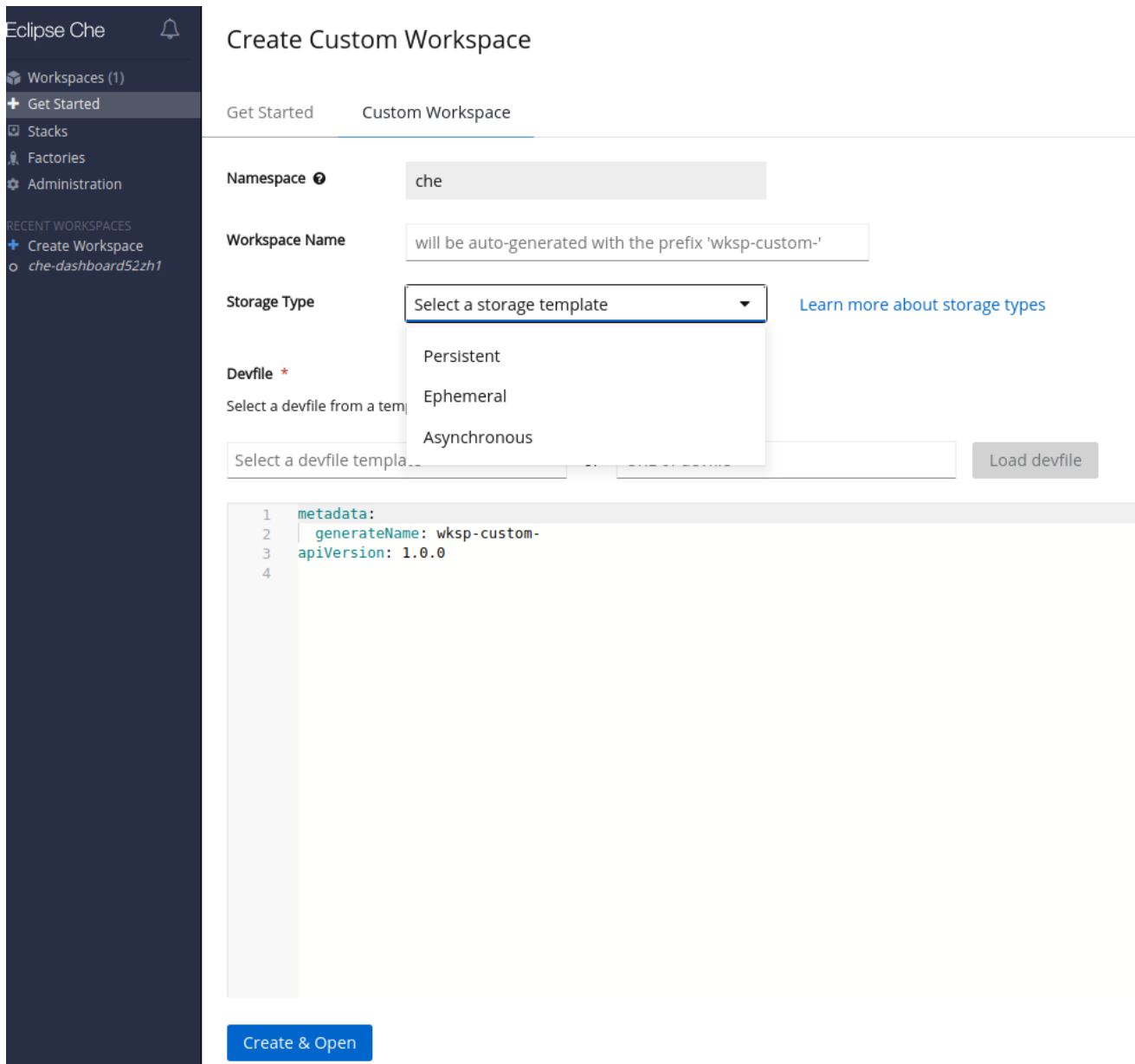
```
che.workspace.storage.available_types=persistent,ephemeral,async
```

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:



4.11.5. 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.

4.12. IMPORTING UNTRUSTED TLS CERTIFICATES TO CODEREADY WORKSPACES

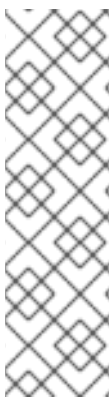
Internal communications between CodeReady Workspaces components are, by default, encrypted with TLS. Communications of CodeReady Workspaces components with external services such as proxies, source code repositories, identity providers may require TLS tool. Those communications require the use of SSL certificates signed by trusted Certificate Authorities.

When the certificates used by CodeReady Workspaces components or by an external service are signed by an untrusted CA it can be necessary to import the CA certificate in the CodeReady Workspaces installation, so that every CodeReady Workspaces component will consider them as signed by a trusted CA.

Typical cases that may require this addition are:

- when the underlying OpenShift cluster uses TLS certificates signed by a CA that is not trusted,
- when CodeReady Workspaces server or workspace components connect to external services such as RH-SSO or a Git server that use TLS certificates signed by an untrusted CA.

To store those certificates, CodeReady Workspaces uses a dedicated ConfigMap. Its default name is **ca-certs** but CodeReady Workspaces allows configuring its name.



NOTE

When the cluster contains cluster-wide trusted CA certificates added through the [cluster-wide-proxy configuration](#), CodeReady Workspaces Operator detects them and automatically injects them into this ConfigMap:

- CodeReady Workspaces automatically labels the ConfigMap with the **config.openshift.io/inject-trusted-cabundle="true"** label.
- Based on this annotation, OpenShift automatically injects the cluster-wide trusted CA certificates inside the **ca-bundle.crt** key of ConfigMap

4.12.1. At CodeReady Workspaces installation time

Prerequisites

- The **oc** tool is available.

- You are ready to create **CheCluster** custom resource.

Procedure

1. Save the certificates you need to import, to a local file system.

CAUTION

- Certificate files are typically stored as Base64 ASCII files, such as **.pem**, **.crt**, **.ca-bundle**. But, they can also be binary-encoded, for example, as **.cer** files. All Secrets that hold certificate files should use the Base64 ASCII certificate rather than the binary-encoded certificate.
 - CodeReady Workspaces already uses some reserved file names to automatically inject certificates into the ConfigMap, so you should avoid using the following reserved file names to save your certificates:
 - **ca-bundle.crt**
 - **ca.crt**
2. Create a new ConfigMap with the required TLS certificates:

```
$ oc create configmap ca-certs --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. During the installation process, when creating the **CheCluster** custom resource, take care of configuring the right name for the created ConfigMap.
For a CodeReady Workspaces [Operator](#) deployment, ensure you add the **spec.server.ServerTrustStoreConfigMapName** field with the name of the ConfigMap, to the **CheCluster** Custom Resource you will create during the installation:

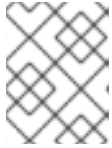
```
spec:
  server:
    ...
    spec.server.ServerTrustStoreConfigMapName: ca-certs
```

4.12.2. On already-running CodeReady Workspaces installations

Prerequisites

- The **oc** tool is available.
- You should first gather the name of the ConfigMap used to import certificates:
On instances of CodeReady Workspaces deployed with the CodeReady Workspaces [Operator](#), retrieve the name of the ConfigMap by reading the **spec.server.ServerTrustStoreConfigMapName** **CheCluster** Custom Resource property:

```
$ oc get checluster codeready-workspaces -n <crw-namespace-name> -o jsonpath={.spec.server.serverTrustStoreConfigMapName}
```

**NOTE**

If the existing installation did not define any name for the ConfigMap, just use **ca-certs**.

Procedure

1. Save the certificates you need to import, to a local file system.

CAUTION

- Certificate files are typically stored as Base64 ASCII files, such as **.pem**, **.crt**, **.ca-bundle**. But, they can also be binary-encoded, for example, as **.cer** files. All Secrets that hold certificate files should use the Base64 ASCII certificate rather than the binary-encoded certificate.
- CodeReady Workspaces already uses some reserved file names to automatically inject certificates into the ConfigMap, so you should avoid using the following reserved file names to save your certificates:
 - **ca-bundle.crt**
 - **ca.crt**

2. Add the required TLS certificates in the ConfigMap:

```
$ oc create configmap <config-map-name> --from-file=<certificate-file-path> -n=<crw-namespace-name> -o yaml --dry-run | oc apply -f -
```

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

3. Configure the CodeReady Workspaces installation to use the ConfigMap:

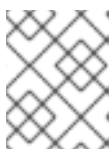
For a CodeReady Workspaces [Operators](#) deployment:

1. Edit the **spec.server.ServerTrustStoreConfigMapName CheCluster** Custom Resource property to match the name of the ConfigMap:

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

4. Restart the CodeReady Workspaces Operator, the CodeReady Workspaces server and RH-SSO to load the new certificates:

```
$ oc rollout restart -n <crw-namespace-name> deployment/codeready-workspaces-operator
$ oc rollout restart -n <crw-namespace-name> deployment/codeready-workspaces/keycloak
$ oc rollout restart -n <crw-namespace-name> deployment/codeready-workspaces
```

**NOTE**

Restarting the CodeReady Workspaces components is not required anymore on CodeReady Workspaces 2.5.0 and higher.

4.12.3. Verification at the CodeReady Workspaces installation level

If you added the certificates without error, the CodeReady Workspaces server starts and obtains RH-SSO configuration over https. Otherwise here is a list of things to verify:

- In case of a CodeReady Workspaces [Operator](#) deployment, the **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 <crw-namespace-name> | 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 pod -o json <codeready-workspaces-pod-name> -n <crw-namespace-name> | jq
.spec.volumes
```

- CodeReady Workspaces mounts certificates 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 <crw-namespace-name> -- ls /public-
certs/
```

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

```
$ oc logs <codeready-workspaces-pod-name> -n <crw-namespace-name>
(...)
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 truststore contains the certificates. The certificates SHA1 fingerprints are among the list of the SHA1 of the certificates included in the truststore returned by the following command:

```
$ oc exec -t <codeready-workspaces-pod-name> -n workspaces -- keytool -list -keystore
/home/jboss/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
```

4.12.4. Verification at the workspace level

- Start a workspace, get the OpenShift namespace in which it has been created, and wait for it to be started

- Get the name of the workspace Pod with the following command:

```
$ oc get pods -o=jsonpath='{.items[0].metadata.name}' -n <workspace namespace> | grep '^workspace.*'
```

- Get the name of the theia IDE container in the workspace POD with the following command:

```
$ oc get -o json pod <workspace pod name> -n <workspace namespace> | \
jq -r '.spec.containers[] | select(.name | startswith("theia-ide")).name'
```

- Look for a **ca-certs** ConfigMap that should have been created inside the workspace namespace:

```
$ oc get cm ca-certs <workspace namespace>
```

- Check that the entries in the **ca-certs** config map contain all the additional entries you added in the certificate ConfigMap at the CodeReady Workspaces installation level, in addition to the the **ca-bundle.crt** entry which is a reserved one:

```
$ oc get cm ca-certs -n <workspace namespace> -o json | jq -r '.data | keys[]'
ca-bundle.crt
manually-added-certificate.crt
```

- Make sure that the **ca-certs** ConfigMap has been added as a volume in the workspace Pod:

```
$ oc get -o json pod <workspace pod name> -n <workspace namespace> | \
jq '.spec.volumes[] | select(.configMap.name == "ca-certs")'
{
  "configMap": {
    "defaultMode": 420,
    "name": "ca-certs"
  },
  "name": "che-self-signed-certs"
}
```

- Confirm that the volume has been mounted into containers, especially in the Theia IDE container:

```
$ oc get -o json pod <workspace pod name> -n <workspace namespace> | \
jq '.spec.containers[] | select(.name == "<theia ide container name>").volumeMounts[] | \
select(.name == "che-self-signed-certs")'
{
  "mountPath": "/public-certs",
  "name": "che-self-signed-certs",
  "readOnly": true
}
```

- Inspect **/public-certs** folder on the Theia IDE container and check that its contents match the list of entries in the **ca-certs** ConfigMap:

```
$ oc exec <workspace pod name> -c <theia ide container name> -n <workspace namespace> -- ls /public-certs
ca-bundle.crt
```

```
manually-added-certificate.crt
```

4.13. SWITCHING BETWEEN EXTERNAL AND INTERNAL WAYS IN INTER-COMPONENT COMMUNICATION

The communication between Red Hat CodeReady Workspaces components uses the internal cluster host name. In following situations, instead, use external OpenShift Route in inter-component communication:

- The OpenShift instance is running multiple CodeReady Workspaces instance.
- The environment is restricting communication between namespaces.

Prerequisites

- The **oc** tool is available.
- An instance of CodeReady Workspaces running in {platform-name}.

Procedure

1. To use external OpenShift Route in inter-component communication:

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

2. To use internal cluster host name in inter-component communication:

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

4.14. SETTING UP THE RH-SSO CODEREADY-WORKSPACES-USERNAME-READONLY THEME FOR THE RED HAT CODEREADY WORKSPACES LOGIN PAGE

The following procedure is relevant for all CodeReady Workspaces instances with the OpenShift OAuth service enabled.

When a user with pre-created namespaces logs in to Red Hat CodeReady Workspaces Dashboard for the first time, a page allowing the user to update account information is displayed. It is possible to change the username, but choosing a username that doesn't match the OpenShift username, prevents the user's workspaces from running. This is caused by CodeReady Workspaces attempts to use a non-existing namespace, the name of which is derived from a user OpenShift username, to create a workspace. To prevent this, modify the RH-SSO settings using the steps below.

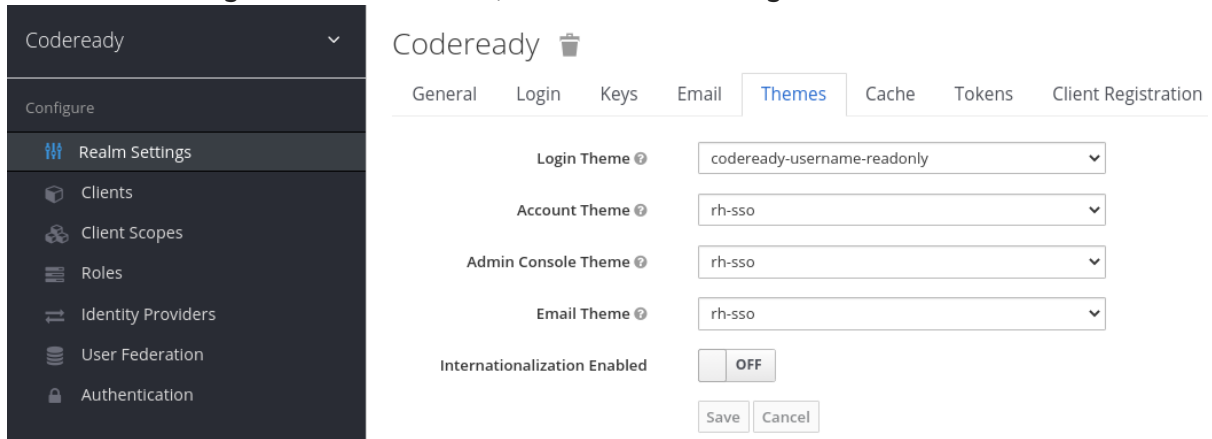
Prerequisites

- An instance of CodeReady Workspaces running in OpenShift.
- A user is logged in to the RH-SSO service.

Procedure

After changing a username, set the **Login Theme** option to **readonly**.

1. In the main **Configure** menu on the left, select **Realm Settings**:



The screenshot shows the Codeready configuration interface. On the left, a dark sidebar menu is open, showing the 'Configure' section with 'Realm Settings' selected. The main content area is titled 'Codeready' and has several tabs: 'General', 'Login', 'Keys', 'Email', 'Themes', 'Cache', 'Tokens', and 'Client Registration'. The 'Themes' tab is active. It contains four theme selection fields: 'Login Theme' (set to 'codeready-username-readonly'), 'Account Theme' (set to 'rh-sso'), 'Admin Console Theme' (set to 'rh-sso'), and 'Email Theme' (set to 'rh-sso'). There is also a toggle for 'Internationalization Enabled' which is currently 'OFF'. At the bottom of the form are 'Save' and 'Cancel' buttons.

2. Navigate to the **Themes** tab.
3. In the **Login Theme** field, select the **codeready-workspaces-username-readonly** option and click the **Save** button to apply the changes.

CHAPTER 5. UPGRADING CODEREADY WORKSPACES

This chapter describes how to upgrade a CodeReady Workspaces instance from version 2.4 to CodeReady Workspaces 2.5.

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, “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 2.5 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

- An 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 **<workspaces>** project.
- An installation of the **crwctl** 2.5 version management tool. See [Section 3.2.1, “Installing the crwctl CLI management tool”](#).

Procedure

1. In all running workspaces in the CodeReady Workspaces 2.4 instance, save and push changes back to the Git repositories.
2. Shut down all workspaces in the CodeReady Workspaces 2.4 instance.
3. Run the following command:

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



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 2.5 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 2.4 to version 2.5.

Prerequisites

- An administrative account on an instance of OpenShift.
- A running instance version 2.4 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 **<workspaces>** project. See [Section 3.3, “Installing CodeReady Workspaces in a restricted environment”](#).
- The **crwctl** 2.5 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 an 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>
```



WARNING

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 openj9-
Workspaces	stacks- , plugin-
IBM Z and IBM Power Systems	-openj9-

Table 5.3. Images to copy in the private registry

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac	configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55	devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7	jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd	machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc

<source-image>	<target-image>
<p>registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>	<p>plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99</p>	<p>plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>	<p>plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>	<p>plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>	<p>plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfbc9da505d715bd75</p>
<p>registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763b431744790a89acd5c6d197316b694693b58c19770</p>	<p>plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763b431744790a89acd5c6d197316b694693b58c19770</p>
<p>registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f990e9fab9baa47f659ef9450b7881072e3cb40399</p>	<p>plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f990e9fab9baa47f659ef9450b7881072e3cb40399</p>
<p>registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a950cc3cbe21ed7301c65f84bb509fa2b6e71fa81d</p>	<p>plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a950cc3cbe21ed7301c65f84bb509fa2b6e71fa81d</p>

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6ccdb8c303f32d046f7eccf9f452eb30e615f2a0bf0e	pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6ccdb8c303f32d046f7eccf9f452eb30e615f2a0bf0e
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505a0fa6f96969cf8cc4aa8c4db95b40626f95854a525	pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505a0fa6f96969cf8cc4aa8c4db95b40626f95854a525
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:c9f48f247cff27280587aef54cea5d8a27e0eb55c99a73726cd7d575db7fbcc	pluginregistry-rhel8@sha256:c9f48f247cff27280587aef54cea5d8a27e0eb55c99a73726cd7d575db7fbcc
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57	server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb	stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2	stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0	stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db	stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db

<source-image>	<target-image>
<p>registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3</p>	<p>theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3</p>
<p>registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0</p>	<p>theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0</p>
<p>registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed</p>	<p>traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed</p>
<p>registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9d3169ce</p>	<p>eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9d3169ce</p>
<p>registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9d3169ce</p>	<p>eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9d3169ce</p>
<p>registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9d3169ce</p>	<p>eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88adb4c4f9f3b5d9d3169ce</p>
<p>registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833</p>	<p>eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833</p>
<p>registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0</p>	<p>eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0</p>

<code><source-image></code>	<code><target-image></code>
<code>registry.redhat.io/rh-ss0-7/sso74-0penj9-0penshift-rhel8@sha256:9297414d1cad8f86871f240c1c0ae324f7d1a3285c22ac7dd878bfcf3c59a75f</code>	<code>sso74-0penj9-0penshift-rhel8@sha256:9297414d1cad8f86871f240c1c0ae324f7d1a3285c22ac7dd878bfcf3c59a75f</code>
<code>registry.redhat.io/rh-ss0-7/sso74-0penj9-0penshift-rhel8@sha256:9297414d1cad8f86871f240c1c0ae324f7d1a3285c22ac7dd878bfcf3c59a75f</code>	<code>sso74-0penj9-0penshift-rhel8@sha256:9297414d1cad8f86871f240c1c0ae324f7d1a3285c22ac7dd878bfcf3c59a75f</code>
<code>registry.redhat.io/rh-ss0-7/sso74-0penshift-rhel8@sha256:c0045cd676e06eb17083a44c4b90b29b11ddb40e1fb6a7b651384cf0960f5158</code>	<code>sso74-0penshift-rhel8@sha256:c0045cd676e06eb17083a44c4b90b29b11ddb40e1fb6a7b651384cf0960f5158</code>
<code>registry.redhat.io/rhel8/postgresql-96@sha256:5b5bf623d89deda89250f422d352b122bce9533b902b5474f9c63a9facc7a6f1</code>	<code>postgresql-96@sha256:5b5bf623d89deda89250f422d352b122bce9533b902b5474f9c63a9facc7a6f1</code>
<code>registry.redhat.io/rhsc1/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73</code>	<code>mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73</code>
<code>registry.redhat.io/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6</code>	<code>ubi8ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6</code>

Verification steps

- Verify the images have the same digests:

```
$ skopeo inspect docker://<source-image>
$ skopeo inspect 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 2.4 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 `<workspaces>` project. See [Section 3.3, “Installing CodeReady Workspaces in a restricted environment”](#).
- 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` 2.5 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 2.4 instance, save and push changes back to the Git repositories.
2. Stop all workspaces in the CodeReady Workspaces 2.4 instance.
3. Run the following command:

```
$ crwctl server:update --che-operator-image=<image-registry>/<organization>/crw-2-rhel8-operator:2.5 -n 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 2.5 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 OPENSIFT 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

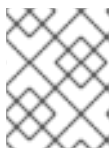
1. Navigate to the OpenShift web console and select the Administrator Perspective.
2. In the **Home** > **Projects** section, navigate to the project containing the CodeReady Workspaces instance.



NOTE

The default project name is `<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.



NOTE

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.

- 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 OPENSIFT 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:

- Sign in to the cluster:

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

- Switch to the project where the CodeReady Workspaces instance is deployed:

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

- 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
```

- Delete the CodeReady Workspaces cluster:

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

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

```
$ oc get csv
NAME                                DISPLAY          VERSION  REPLACES          PHASE
red-hat-codeready-workspaces.v2.5  Red Hat CodeReady Workspaces 2.5      red-hat-codeready-workspaces.v2.4 Succeeded
```

- Delete the CodeReady Workspaces CSV:

```
$ oc delete csv red-hat-codeready-workspaces.v2.5
clusterserviceversion.operators.coreos.com "red-hat-codeready-workspaces.v2.5" 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. Export the name of the CodeReady Workspaces namespace you want to remove:

```
$ export codereadyNamespace=<codeready-namespace-to-remove>
```

3. Export your user access token and Keycloak URLs:

```
$ export KEYCLOAK_BASE_URL="http://${KEYCLOAK_URL}/auth"
```

```
$ export USER_ACCESS_TOKEN=$(curl -X POST
${KEYCLOAK_BASE_URL}/realms/codeready/protocol/openid-connect/token \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "username=admin" \
-d "password=admin" \
-d "grant_type=password" \
-d "client_id=codeready-public" | jq -r .access_token)
```

4. Stop the server using the UAT:

```
$ crwctl/bin/crwctl server:stop -n ${codereadyNamespace} --access-
token=${USER_ACCESS_TOKEN}
```

5. Delete your project and your CodeReady Workspaces deployment:

```
$ oc project ${codereadyNamespace}
```

```
$ oc delete deployment codeready-operator
```

```
$ oc delete checluster codeready-workspaces
```

```
❯ $ oc delete project ${codereadyNamespace}
```

6. Verify that the removal was successful by listing the information about the project:

```
❯ $ oc describe project ${codereadyNamespace}
```

7. Remove a specified **ClusterRoleBinding**:

```
❯ $ oc delete clusterrolebinding codeready-operator
```