Red Hat CodeReady Workspaces 2.15

Installation Guide

Installing Red Hat CodeReady Workspaces 2.15

Last Updated: 2022-05-17

Red Hat CodeReady Workspaces 2.15 Installation Guide

Installing Red Hat CodeReady Workspaces 2.15

Robert Kratky
rkratky@redhat.com

Fabrice Flore-Thébault
ffloreth@redhat.com

Jana Vrbkova
jvrbkova@redhat.com

Max Leonov
mleonov@redhat.com

Legal Notice

Abstract

Information for administrators installing Red Hat CodeReady Workspaces.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

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

CHAPTER 1. SUPPORTED PLATFORMS

This section describes the availability and the supported installation methods of CodeReady Workspaces 2.15 on OpenShift Container Platform 4.10 4.8, 3.11, and OpenShift Dedicated.

Table 1.1. Supported deployment environments for CodeReady Workspaces 2.15 on OpenShift Container Platform and OpenShift Dedicated

Platform	Architecture	Deployment method
OpenShift Container Platform 3.11	AMD64 and Intel 64 (x86_64)	**crwctl**
OpenShift Container Platform 4.8 to 4.10	AMD64 and Intel 64 (x86_64)	OperatorHub, **crwctl**
OpenShift Container Platform 4.8 to 4.10	IBM Z (s390x)	OperatorHub, **crwctl**
OpenShift Container Platform 4.8 to 4.10	IBM Power (ppc64le)	OperatorHub, **crwctl**
OpenShift Dedicated 4.10	AMD64 and Intel 64 (x86_64)	Add-on service
Red Hat OpenShift Service on AWS (ROSA)	AMD64 and Intel 64 (x86_64)	Add-on service

CHAPTER 2. CONFIGURING THE CODEREADY WORKSPACES INSTALLATION

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

Additional resources

- Section 2.1, "Understanding the **CheCluster** Custom Resource"
- Section 2.2, "Using the OpenShift web console to configure the **CheCluster** Custom Resource during installation"
- Section 2.3, "Using the OpenShift web console to configure the **CheCluster** Custom Resource"
- Section 2.4, "Using crwctl to configure the **CheCluster** Custom Resource during installation"
- Section 2.5, "Using the CLI to configure the CheCluster Custom Resource"
- Section 2.6, "**CheCluster** Custom Resource fields reference"

2.1. UNDERSTANDING THE CHECLUSTER CUSTOM RESOURCE

A default deployment of CodeReady Workspaces consists of a **CheCluster** Custom Resource parameterized by the Red Hat CodeReady Workspaces Operator.

The **CheCluster** Custom Resource is a Kubernetes object. You can configure it by editing the **CheCluster** Custom Resource YAML file. This file contains sections to configure each component: **auth**, **database**, **server**, **storage**.

The Red Hat CodeReady Workspaces Operator translates the **CheCluster** Custom Resource into a config map usable by each component of the CodeReady Workspaces installation.

The OpenShift platform applies the configuration to each component, and creates the necessary Pods. When OpenShift detects changes 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. Apply the **CheCluster** Custom Resource YAML file with suitable modifications in the **server** component section.
2. The Operator generates the **che ConfigMap**.
3. OpenShift detects changes in the **ConfigMap** and triggers a restart of the CodeReady Workspaces Pod.

Additional resources

- Understanding Operators.
- Understanding Custom Resources.

2.2. USING THE OPENSHIFT WEB CONSOLE TO CONFIGURE THE CHECLUSTER CUSTOM RESOURCE DURING INSTALLATION

To deploy CodeReady Workspaces with a suitable configuration, edit the **CheCluster** Custom Resource YAML file during the installation of CodeReady Workspaces. Otherwise, the CodeReady Workspaces deployment uses the default configuration parameterized by the Operator.

The **CheCluster** Custom Resource YAML file contains sections to configure each component: **auth**, **database**, **server**, **storage**.

Prerequisites

- Access to an administrator account on an instance of OpenShift.

Procedure

1. In the left panel, click **Operators**, then click **Installed Operators**.
2. On the **Installed Operators** page, click the **Red Hat CodeReady Workspaces** name.
3. On the **Operator details** page, in the **Details** tab, click the **Create instance** link in the **Provided APIs** section.
   This navigates you to the **Create CheCluster** page, which contains the configuration needed to create a CodeReady Workspaces instance, stored in the **CheCluster** Custom Resource.
4. On the **Create CheCluster** page, click **YAML view**.
5. In the YAML file, find or add the property to configure. Set the property to a suitable value:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# ...
spec:
  <component>:
    # ...
      <property-to-configure>: <value>
```

6. Create the **codeready-workspaces** cluster by using the **Create** button at the end of the page.
7. On the **Operator details** page, in the **Red Hat CodeReady Workspaces Cluster** tab, click the **codeready-workspaces** link.
8. Navigate to the **codeready-workspaces** instance using the link displayed under the **Red Hat CodeReady Workspaces URL** output.

NOTE

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

Verification

1. In the left panel, click **Workloads**, then click **ConfigMaps**.
2. On the **ConfigMaps** page, click **codeready**.

3. Navigate to the **YAML** tab.
4. Verify that the YAML file contains the configured property and value.

Additional resources

- Chapter 2, *Configuring the CodeReady Workspaces installation* .
- Section 4.1, "Advanced configuration options for the CodeReady Workspaces server component".

2.3. USING THE OPENSHIFT WEB CONSOLE TO CONFIGURE THE CHECLUSTER CUSTOM RESOURCE

To configure a running instance of CodeReady Workspaces, edit the **CheCluster** Custom Resource YAML file.

The **CheCluster** Custom Resource YAML file contains sections to configure each component: **auth**, **database**, **server**, **storage**.

Prerequisites

- An instance of CodeReady Workspaces on OpenShift.
- Access to an administrator account on the instance of OpenShift and to the OpenShift web console.

Procedure

1. In the left panel, click **Operators**, then click **Installed Operators**.
2. On the **Installed Operators** page, click **Red Hat CodeReady Workspaces**
3. Navigate to the **Red Hat CodeReady Workspaces instance Specification**tab and click **codeready-workspaces**.
4. Navigate to the **YAML** tab.
5. In the YAML file, find or add the property to configure. Set the property to a suitable value:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# ...
spec:
  <component>:
    # ...
      <property-to-configure>: <value>
```

6. Click **Save** to apply the changes.

Verification

1. In the left panel, click **Workloads**, then click **ConfigMaps**.

2. On the **ConfigMaps** page, click **codeready**.
3. Navigate to the **YAML** tab.
4. Verify that the YAML file contains the configured property and value.

Additional resources

- Chapter 2, *Configuring the CodeReady Workspaces installation* .
- Section 4.1, "Advanced configuration options for the CodeReady Workspaces server component".

2.4. USING CRWCTL TO CONFIGURE THE CHECLUSTER CUSTOM RESOURCE DURING INSTALLATION

To deploy CodeReady Workspaces with a suitable configuration, edit the **CheCluster** Custom Resource YAML file during the installation of CodeReady Workspaces. Otherwise, the CodeReady Workspaces deployment uses the default configuration parameterized by the Operator.

Prerequisites

- Access to an administrator account on an instance of OpenShift.
- The crwctl tool is available. See Section 3.3.1, "Installing the crwctl CLI management tool" .

Procedure

- Create a **che-operator-cr-patch.yaml** YAML file that contains the subset of the **CheCluster** Custom Resource to configure:

```
spec:
  <component>:
    <property-to-configure>: <value>
```

- Deploy CodeReady Workspaces and apply the changes described in **che-operator-cr-patch.yaml** file:

```
$ {prod-cli} server:deploy \
--che-operator-cr-patch-yaml=che-operator-cr-patch.yaml \
--platform <chosen-platform>
```

Verification

1. Verify the value of the configured property:

```
$ oc get configmap che -o jsonpath='{.data.<configured-property>}' \
-n openshift-workspaces
```

Additional resources

- Chapter 2, *Configuring the CodeReady Workspaces installation* .

- Section 4.1, “Advanced configuration options for the CodeReady Workspaces server component”.

2.5. USING THE CLI TO CONFIGURE THE CHECLUSTER CUSTOM RESOURCE

To configure a running instance of CodeReady Workspaces, edit the **CheCluster** Custom Resource YAML file.

Prerequisites

- An instance of CodeReady Workspaces on OpenShift.
- Access to an administrator account on the instance of OpenShift.
- The **oc** tool is available.

Procedure

1. Edit the CheCluster Custom Resource on the cluster:

```
$ oc edit checluster/eclipse-che -n openshift-workspaces
```

2. Save and close the file to apply the changes.

Verification

1. Verify the value of the configured property:

```
$ oc get configmap che -o jsonpath='{.data.<configured-property>}' \
-n openshift-workspaces
```

Additional resources

- Chapter 2, *Configuring the CodeReady Workspaces installation* .
- Section 4.1, “Advanced configuration options for the CodeReady Workspaces server component”.

2.6. 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 2.1, “**CheCluster** Custom Resource **server** settings, related to the CodeReady Workspaces server component.”
- Table 2.2, “**CheCluster** Custom Resource **database** configuration settings related to the database used by CodeReady Workspaces.”
- Table 2.3, “Custom Resource **auth** configuration settings related to authentication used by CodeReady Workspaces.”

- 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 **metrics** settings, related to the CodeReady Workspaces metrics collection used by CodeReady Workspaces."
- Table 2.7, "**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	Description
airGapContainerRegistryHostname	Optional host name, or URL, to an alternate container registry to pull images from. This value overrides the container registry host name defined in all the default container images involved in a Che deployment. This is particularly useful to install Che in a restricted environment.
airGapContainerRegistryOrganization	Optional repository name of an alternate container registry to pull images from. This value overrides the container registry organization defined in all the default container images involved in a Che deployment. This is particularly useful to install CodeReady Workspaces in a restricted environment.
allowUserDefinedWorkspaceNamespaces	Deprecated. The value of this flag is ignored. Defines that a user is allowed to specify a Kubernetes namespace, or an OpenShift project, which differs from the default. It's NOT RECOMMENDED to set to **true** without OpenShift OAuth configured. The OpenShift infrastructure also uses this property.

Property	Description
cheClusterRoles	A comma-separated list of ClusterRoles that will be assigned to Che ServiceAccount. Each role must have **app.kubernetes.io/part-of=che.eclipse.org** label. Be aware that the Che Operator has to already have all permissions in these ClusterRoles to grant them.
cheDebug	Enables the debug mode for Che server. Defaults to **false**.
cheFlavor	Specifies a variation of the installation. The options are **che** for upstream Che installations, or **codeready** for CodeReady Workspaces installation. Override the default value only on necessary occasions.
cheHost	Public host name of the installed Che server. When value is omitted, the value it will be automatically set by the Operator. See the **cheHostTLSSecret** field.
cheHostTLSSecret	Name of a secret containing certificates to secure ingress or route for the custom host name of the installed Che server. The secret must have **app.kubernetes.io/part-of=che.eclipse.org** label. See the **cheHost** field.
cheImage	Overrides the container image used in Che 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.
cheImagePullPolicy	Overrides the image pull policy used in Che deployment. Default value is **Always** for **nightly**, **next** or **latest** images, and **IfNotPresent** in other cases.
cheImageTag	Overrides the tag of the container image used in Che deployment. Omit it or leave it empty to use the default image tag provided by the Operator.
cheLogLevel	Log level for the Che server: **INFO** or **DEBUG**. Defaults to **INFO**.
cheServerIngress	The Che server ingress custom settings.
cheServerRoute	The Che server route custom settings.
cheWorkspaceClusterRole	Custom cluster role bound to the user for the Che workspaces. The role must have **app.kubernetes.io/part-of=che.eclipse.org** label. The default roles are used when omitted or left blank.
customCheProperties	Map of additional environment variables that will be applied in the generated **che** ConfigMap to be used by the Che server, in addition to the values already generated from other fields of the **CheCluster** custom resource (CR). When **customCheProperties** contains a property that would be normally generated in **che** ConfigMap from other CR fields, the value defined in the **customCheProperties** is used instead.

Property	Description
dashboardCpuLimit	Overrides the CPU limit used in the dashboard deployment. In cores. (500m = .5 cores). Default to 500m.
dashboardCpuRequest	Overrides the CPU request used in the dashboard deployment. In cores. (500m = .5 cores). Default to 100m.
dashboardImage	Overrides the container image used in the dashboard deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator.
dashboardImagePullPolicy	Overrides the image pull policy used in the dashboard deployment. Default value is **Always** for **nightly**, **next** or **latest** images, and **IfNotPresent** in other cases.
dashboardIngress	Dashboard ingress custom settings.
dashboardMemoryLimit	Overrides the memory limit used in the dashboard deployment. Defaults to 256Mi.
dashboardMemoryRequest	Overrides the memory request used in the dashboard deployment. Defaults to 16Mi.
dashboardRoute	Dashboard route custom settings.
devfileRegistryCpuLimit	Overrides the CPU limit used in the devfile registry deployment. In cores. (500m = .5 cores). Default to 500m.
devfileRegistryCpuRequest	Overrides the CPU request used in the devfile registry deployment. In cores. (500m = .5 cores). Default to 100m.
devfileRegistryImage	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.
devfileRegistryIngress	The devfile registry ingress custom settings.
devfileRegistryMemoryLimit	Overrides the memory limit used in the devfile registry deployment. Defaults to 256Mi.
devfileRegistryMemoryRequest	Overrides the memory request used in the devfile registry deployment. Defaults to 16Mi.
devfileRegistryPullPolicy	Overrides the image pull policy used in the devfile registry deployment. Default value is **Always** for **nightly**, **next** or **latest** images, and **IfNotPresent** in other cases.

Property	Description
devfileRegistryRoute	The devfile registry route custom settings.
devfileRegistryUrl	Deprecated in favor of **externalDevfileRegistries** fields.
disableInternalClusterSVCNames	Disable internal cluster SVC names usage to communicate between components to speed up the traffic and avoid proxy issues.
externalDevfileRegistries	External devfile registries, that serves sample, ready-to-use devfiles. Configure this in addition to a dedicated devfile registry (when **externalDevfileRegistry** is **false**) or instead of it (when **externalDevfileRegistry** is **true**)
externalDevfileRegistry	Instructs the Operator on whether to deploy a dedicated devfile registry server. By default, a dedicated devfile registry server is started. When **externalDevfileRegistry** is **true**, no such dedicated server will be started by the Operator and configure at least one devfile registry with **externalDevfileRegistries** field.
externalPluginRegistry	Instructs the Operator on whether to deploy a dedicated plugin registry server. By default, a dedicated plugin registry server is started. When **externalPluginRegistry** is **true**, no such dedicated server will be started by the Operator and you will have to manually set the **pluginRegistryUrl** field.
gitSelfSignedCert	When enabled, the certificate from **che-git-self-signed-cert** ConfigMap will be propagated to the Che components and provide particular configuration for Git. Note, the **che-git-self-signed-cert** ConfigMap must have **app.kubernetes.io/part-of=che.eclipse.org** label.
nonProxyHosts	List of hosts that will be reached directly, bypassing the proxy. Specify wild card domain use the following form **.<DOMAIN>** and **|** as delimiter, for example: **localhost|.my.host.com|123.42.12.32** Only use when configuring a proxy is required. Operator respects OpenShift cluster wide proxy configuration and no additional configuration is required, but defining **nonProxyHosts** in a custom resource leads to merging non proxy hosts lists from the cluster proxy configuration and ones defined in the custom resources. See the doc https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html. See also the **proxyURL** fields.
pluginRegistryCpuLimit	Overrides the CPU limit used in the plugin registry deployment. In cores. (500m = .5 cores). Default to 500m.
pluginRegistryCpuRequest	Overrides the CPU request used in the plugin registry deployment. In cores. (500m = .5 cores). Default to 100m.

Property	Description
pluginRegistryImage	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.
pluginRegistryIngress	Plugin registry ingress custom settings.
pluginRegistryMemoryLimit	Overrides the memory limit used in the plugin registry deployment. Defaults to 256Mi.
pluginRegistryMemoryRequest	Overrides the memory request used in the plugin registry deployment. Defaults to 16Mi.
pluginRegistryPullPolicy	Overrides the image pull policy used in the plugin registry deployment. Default value is **Always** for **nightly**, **next** or **latest** images, and **IfNotPresent** in other cases.
pluginRegistryRoute	Plugin registry route custom settings.
pluginRegistryUrl	Public URL of the plugin registry that serves sample ready-to-use devfiles. Set this ONLY when a use of an external devfile registry is needed. See the **externalPluginRegistry** field. By default, this will be automatically calculated by the Operator.
proxyPassword	Password of the proxy server. Only use when proxy configuration is required. See the **proxyURL**, **proxyUser** and **proxySecret** fields.
proxyPort	Port of the proxy server. Only use when configuring a proxy is required. See also the **proxyURL** and **nonProxyHosts** fields.
proxySecret	The secret that contains **user** and **password** for a proxy server. When the secret is defined, the **proxyUser** and **proxyPassword** are ignored. The secret must have **app.kubernetes.io/part-of=che.eclipse.org** label.
proxyURL	URL (protocol+host name) of the proxy server. This drives the appropriate changes in the **JAVA_OPTS** and **https(s)_proxy** variables in the Che server and workspaces containers. Only use when configuring a proxy is required. Operator respects OpenShift cluster wide proxy configuration and no additional configuration is required, but defining **proxyUrl** in a custom resource leads to overrides the cluster proxy configuration with fields **proxyUrl**, **proxyPort**, **proxyUser** and **proxyPassword** from the custom resource. See the doc https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html. See also the **proxyPort** and **nonProxyHosts** fields.
proxyUser	User name of the proxy server. Only use when configuring a proxy is required. See also the **proxyURL**, **proxyPassword** and **proxySecret** fields.

Property	Description
selfSignedCert	Deprecated. The value of this flag is ignored. The Che Operator will automatically detect whether the router certificate is self-signed and propagate it to other components, such as the Che server.
serverCpuLimit	Overrides the CPU limit used in the Che server deployment In cores. (500m = .5 cores). Default to 1.
serverCpuRequest	Overrides the CPU request used in the Che server deployment In cores. (500m = .5 cores). Default to 100m.
serverExposureStrategy	Sets the server and workspaces exposure type. Possible values are **multi-host**, **single-host**, **default-host**. Defaults to **multi-host**, which creates a separate ingress, or OpenShift routes, for every required endpoint. **single-host** makes Che exposed on a single host name with workspaces exposed on subpaths. Read the docs to learn about the limitations of this approach. Also consult the **singleHostExposureType** property to further configure how the Operator and the Che server make that happen on Kubernetes. **default-host** exposes the Che server on the host of the cluster. Read the docs to learn about the limitations of this approach.
serverMemoryLimit	Overrides the memory limit used in the Che server deployment. Defaults to 1Gi.
serverMemoryRequest	Overrides the memory request used in the Che server deployment. Defaults to 512Mi.
serverTrustStoreConfigMapName	Name of the ConfigMap with public certificates to add to Java trust store of the Che server. This is often required when adding the OpenShift OAuth provider, which has HTTPS endpoint signed with self-signed cert. The Che server must be aware of its CA cert to be able to request it. This is disabled by default. The Config Map must have **app.kubernetes.io/part-of=che.eclipse.org** label.
singleHostGatewayConfigMapLabels	The labels that need to be present in the ConfigMaps representing the gateway configuration.
singleHostGatewayConfigSidecarImage	The image used for the gateway sidecar that provides configuration to the gateway. Omit it or leave it empty to use the default container image provided by the Operator.
singleHostGatewayImage	The image used for the gateway in the single host mode. Omit it or leave it empty to use the default container image provided by the Operator.
tlsSupport	Deprecated. Instructs the Operator to deploy Che in TLS mode. This is enabled by default. Disabling TLS sometimes cause malfunction of some Che components.

Property	Description
useInternalClusterSVCNames	Deprecated in favor of **disableInternalClusterSVCNames**.
workspaceNamespaceDefault	Defines Kubernetes default namespace in which user's workspaces are created for a case when a user does not override it. It's possible to use **<username>**, **<userid>** and **<workspaceid>** placeholders, such as che-workspace-<username>. In that case, a new namespace will be created for each user or workspace.
workspacesDefaultPlugins	Default plug-ins applied to Devworkspaces.

Table 2.2. CheCluster Custom Resourcedatabase configuration settings related to the database used by CodeReady Workspaces.

Property	Description
chePostgresContainerResources	PostgreSQL container custom settings
chePostgresDb	PostgreSQL database name that the Che server uses to connect to the DB. Defaults to **dbche**.
chePostgresHostName	PostgreSQL Database host name that the Che server uses to connect to. Defaults is **postgres**. Override this value ONLY when using an external database. See field **externalDb**. In the default case it will be automatically set by the Operator.
chePostgresPassword	PostgreSQL password that the Che server uses to connect to the DB. When omitted or left blank, it will be set to an automatically generated value.
chePostgresPort	PostgreSQL Database port that the Che server uses to connect to. Defaults to 5432. Override this value ONLY when using an external database. See field **externalDb**. In the default case it will be automatically set by the Operator.
chePostgresSecret	The secret that contains PostgreSQL`user` and **password** that the Che server uses to connect to the DB. When the secret is defined, the **chePostgresUser** and **chePostgresPassword** are ignored. When the value is omitted or left blank, the one of following scenarios applies: 1. **chePostgresUser** and **chePostgresPassword** are defined, then they will be used to connect to the DB. 2. **chePostgresUser** or **chePostgresPassword** are not defined, then a new secret with the name **che-postgres-secret** will be created with default value of**pgche** for **user** and with an auto-generated value for **password**. The secret must have **app.kubernetes.io/part-of=che.eclipse.org** label.
chePostgresUser	PostgreSQL user that the Che server uses to connect to the DB. Defaults to **pgche**.

Property	Description
externalDb	Instructs the Operator on whether to deploy a dedicated database. By default, a dedicated PostgreSQL database is deployed as part of the Che installation. When **externalDb** is **true**, no dedicated database will be deployed by the Operator and you will need to provide connection details to the external DB you are about to use. See also all the fields starting with: **chePostgres**.
postgresImage	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.
postgresImagePullPolicy	Overrides the image pull policy used in the PostgreSQL database deployment. Default value is **Always** for **nightly**, **next** or **latest** images, and **IfNotPresent** in other cases.
postgresVersion	Indicates a PostgreSQL version image to use. Allowed values are: **9.6** and **13.3**. Migrate your PostgreSQL database to switch from one version to another.
pvcClaimSize	Size of the persistent volume claim for database. Defaults to **1Gi**. To update pvc storageclass that provisions it must support resize when CodeReady Workspaces has been already deployed.

Table 2.3. Custom Resource auth configuration settings related to authentication used by CodeReady Workspaces.

Property	Description
debug	Debug internal identity provider.
externalIdentityProvider	Instructs the Operator on whether or not to deploy a dedicated Identity Provider (Keycloak or RH SSO instance). Instructs the Operator on whether to deploy a dedicated Identity Provider (Keycloak or RH-SSO instance). By default, a dedicated Identity Provider server is deployed as part of the Che installation. When **externalIdentityProvider** is **true**, no dedicated identity provider will be deployed by the Operator and you will need to provide details about the external identity provider you are about to use. See also all the other fields starting with: **identityProvider**.
gatewayAuthenticationSidecarImage	Gateway sidecar responsible for authentication when NativeUserMode is enabled. See oauth2-proxy or openshift/oauth-proxy.
gatewayAuthorizationSidecarImage	Gateway sidecar responsible for authorization when NativeUserMode is enabled. See kube-rbac-proxy or openshift/kube-rbac-proxy
gatewayHeaderRewriteSidecarImage	Deprecated. The value of this flag is ignored. Sidecar functionality is now implemented in Traefik plugin.

Property	Description
identityProviderAdminUserName	Overrides the name of the Identity Provider administrator user. Defaults to **admin**.
identityProviderClientId	Name of a Identity provider, Keycloak or RH-SSO, **client-id** that is used for Che. Override this when an external Identity Provider is in use. See the **externalIdentityProvider** field. When omitted or left blank, it is set to the value of the **flavour** field suffixed with **-public**.
identityProviderContainerResources	Identity provider container custom settings.
identityProviderImage	Overrides the container image used in the Identity Provider, Keycloak or RH-SSO, deployment. This includes the image tag. Omit it or leave it empty to use the default container image provided by the Operator.
identityProviderImagePullPolicy	Overrides the image pull policy used in the Identity Provider, Keycloak or RH-SSO, deployment. Default value is **Always** for **nightly**, **next** or **latest** images, and **IfNotPresent** in other cases.
identityProviderIngress	Ingress custom settings.
identityProviderPassword	Overrides the password of Keycloak administrator user. Override this when an external Identity Provider is in use. See the **externalIdentityProvider** field. When omitted or left blank, it is set to an auto-generated password.
identityProviderPostgresPassword	Password for a Identity Provider, Keycloak or RH-SSO, to connect to the database. Override this when an external Identity Provider is in use. See the **externalIdentityProvider** field. When omitted or left blank, it is set to an auto-generated password.
identityProviderPostgresSecret	The secret that contains **password** for the Identity Provider, Keycloak or RH-SSO, to connect to the database. When the secret is defined, the **identityProviderPostgresPassword** is ignored. When the value is omitted or left blank, the one of following scenarios applies: 1. **identityProviderPostgresPassword** is defined, then it will be used to connect to the database. 2. **identityProviderPostgresPassword** is not defined, then a new secret with the name **che-identity-postgres-secret** will be created with an auto-generated value for **password**. The secret must have **app.kubernetes.io/part-of=che.eclipse.org** label.
identityProviderRealm	Name of a Identity provider, Keycloak or RH-SSO, realm that is used for Che. Override this when an external Identity Provider is in use. See the **externalIdentityProvider** field. When omitted or left blank, it is set to the value of the **flavour** field.
identityProviderRoute	Route custom settings.

Property	Description
identityProviderSecret	The secret that contains **user** and **password** for Identity Provider. When the secret is defined, the **identityProviderAdminUserName** and **identityProviderPassword** are ignored. When the value is omitted or left blank, the one of following scenarios applies: 1. **identityProviderAdminUserName** and **identityProviderPassword** are defined, then they will be used. 2. **identityProviderAdminUserName** or **identityProviderPassword** are not defined, then a new secret with the name **che-identity-secret** will be created with default value**admin** for **user** and with an auto-generated value for**password**. The secret must have **app.kubernetes.io/part-of=che.eclipse.org** label.
identityProviderURL	Public URL of the Identity Provider server (Keycloak / RH-SSO server). Set this ONLY when a use of an external Identity Provider is needed. See the **externalIdentityProvider** field. By default, this will be automatically calculated and set by the Operator.
initialOpenShiftOAuthUser	For operating with the OpenShift OAuth authentication, create a new user account since the kubeadmin can not be used. If the value is true, then a new OpenShift OAuth user will be created for the HTPasswd identity provider. If the value is false and the user has already been created, then it will be removed. If value is an empty, then do nothing. The user's credentials are stored in the **openshift-oauth-user-credentials** secret in 'openshift-config' namespace by Operator. Note that this solution is Openshift 4 platform-specific.
nativeUserMode	Enables native user mode. Currently works only on OpenShift and DevWorkspace engine. Native User mode uses OpenShift OAuth directly as identity provider, without Keycloak.
oAuthClientName	Name of the OpenShift **OAuthClient** resource used to setup identity federation on the OpenShift side. Auto-generated when left blank. See also the **OpenShiftoAuth** field.
oAuthSecret	Name of the secret set in the OpenShift **OAuthClient** resource used to setup identity federation on the OpenShift side. Auto-generated when left blank. See also the **OAuthClientName** field.
openShiftoAuth	Enables the integration of the identity provider (Keycloak / RHSSO) with OpenShift OAuth. Empty value on OpenShift by default. This will allow users to directly login with their OpenShift user through the OpenShift login, and have their workspaces created under personal OpenShift namespaces. WARNING: the **kubeadmin** user is NOT supported, and logging through it will NOT allow accessing the Che Dashboard.
updateAdminPassword	Forces the default **admin** Che user to update password on first login. Defaults to **false**.

Table 2.4. CheCluster Custom Resource storage configuration settings related to persistent storage used by CodeReady Workspaces.

Property	Description
postgresPVCStorageClassName	Storage class for the Persistent Volume Claim dedicated to the PostgreSQL database. When omitted or left blank, a default storage class is used.
preCreateSubPaths	Instructs the Che server to start a special Pod to pre-create a sub-path in the Persistent Volumes. Defaults to **false**, however it will need to enable it according to the configuration of your Kubernetes cluster.
pvcClaimSize	Size of the persistent volume claim for workspaces. Defaults to **10Gi**.
pvcJobsImage	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	Persistent volume claim strategy for the Che server. This Can be:`common` (all workspaces PVCs in one volume), **per-workspace** (one PVC per workspace for all declared volumes) and **unique** (one PVC per declared volume). Defaults to **common**.
workspacePVCStorageClassName	Storage class for the Persistent Volume Claims dedicated to the Che workspaces. When omitted or left blank, a default storage class is used.

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

Property	Description
ingressClass	Ingress class that will define the which controller will manage ingresses. Defaults to **nginx**. NB: This drives the **kubernetes.io/ingress.class** annotation on Che-related ingresses.
ingressDomain	Global ingress domain for a Kubernetes cluster. This MUST be explicitly specified: there are no defaults.
ingressStrategy	Strategy for ingress creation. Options are: **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). Defaults to **multi-host** Deprecated in favor of **serverExposureStrategy** in the **server** section, which defines this regardless of the cluster type. When both are defined, the **serverExposureStrategy** option takes precedence.
securityContextFsGroup	The FSGroup in which the Che Pod and workspace Pods containers runs in. Default value is **1724**.
securityContextRunAsUser	ID of the user the Che Pod and workspace Pods containers run as. Default value is **1724**.

Property	Description
singleHostExposureType	When the serverExposureStrategy is set to **single-host**, the way the server, registries and workspaces are exposed is further configured by this property. The possible values are **native**, which means that the server and workspaces are exposed using ingresses on K8s or **gateway** where the server and workspaces are exposed using a custom gateway based on Traefik. All the endpoints whether backed by the ingress or gateway **route** always point to the subpaths on the same domain. Defaults to **native**.
tlsSecretName	Name of a secret that will be used to setup ingress TLS termination when TLS is enabled. When the field is empty string, the default cluster certificate will be used. See also the **tlsSupport** field.

Table 2.6. CheCluster Custom Resource metrics settings, related to the CodeReady Workspaces metrics collection used by CodeReady Workspaces.

Property	Description
enable	Enables **metrics** the Che server endpoint. Default to **true**.

Table 2.7. CheCluster Custom Resource status defines the observed state of CodeReady Workspaces installation

Property	Description
cheClusterRunning	Status of a Che installation. Can be **Available**, **Unavailable**, or **Available, Rolling Update in Progress**.
cheURL	Public URL to the Che server.
cheVersion	Current installed Che version.
dbProvisioned	Indicates that a PostgreSQL instance has been correctly provisioned or not.
devfileRegistryURL	Public URL to the devfile registry.
devworkspaceStatus	The status of the Devworkspace subsystem
gitHubOAuthProvisioned	Indicates whether an Identity Provider instance, Keycloak or RH-SSO, has been configured to integrate with the GitHub OAuth.
helpLink	A URL that points to some URL where to find help related to the current Operator status.
keycloakProvisioned	Indicates whether an Identity Provider instance, Keycloak or RH-SSO, has been provisioned with realm, client and user.

Property	Description
keycloakURL	Public URL to the Identity Provider server, Keycloak or RH-SSO,.
message	A human readable message indicating details about why the Pod is in this condition.
openShiftOAuthUserCredentialsSecret	OpenShift OAuth secret in **openshift-config** namespace that contains user credentials for HTPasswd identity provider.
openShiftoAuthProvisioned	Indicates whether an Identity Provider instance, Keycloak or RH-SSO, has been configured to integrate with the OpenShift OAuth.
pluginRegistryURL	Public URL to the plugin registry.
reason	A brief CamelCase message indicating details about why the Pod is in this state.

CHAPTER 3. INSTALLING CODEREADY WORKSPACES

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

3.1. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT 4 USING OPERATORHUB

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

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

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

Prerequisites

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

3.1.1. 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 it also configures all these services.

Prerequisites

- Access to the OpenShift web console on the cluster.

Procedure

1. In the left panel, navigate to the **Operators → OperatorHub** page.
2. In the **Filter by keyword** field, enter **Red Hat CodeReady Workspaces**.
3. Click the **Red Hat CodeReady Workspaces** tile.
4. In the **Red Hat CodeReady Workspaces** pop-up window, click the **Install** button.
5. On the **Install Operator** page, click the **Install** button.

Verification steps

1. To verify that the Red Hat CodeReady Workspaces Operator has installed correctly, in the left panel, navigate to the **Operators → Installed Operators** page.

2. On the **Installed Operators** page, click the **Red Hat CodeReady Workspaces** name and navigate to the **Details** tab.

3. In the **ClusterServiceVersion details** section, wait for the following messages:

   - Status: **Succeeded**
   - Status reason: **install strategy completed with no errors**

4. Navigate to the **Events** tab and wait for the following message: **install strategy completed with no errors**.

3.1.2. 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. In the left panel, click **Operators**, then click **Installed Operators**.

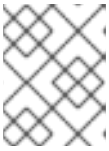
2. On the **Installed Operators** page, click the **Red Hat CodeReady Workspaces** name.

3. On the **Operator details** page, in the **Details** tab, click the **Create instance** link in the **Provided APIs** section.
   This navigates you to the **Create CheCluster** page, which contains the configuration needed to create a CodeReady Workspaces instance, stored in the **CheCluster** Custom Resource.

4. Create the **codeready-workspaces** cluster by using the **Create** button at the end of the page using the default values.

5. On the **Operator details** page, in the **Red Hat CodeReady Workspaces Cluster** tab, click the **codeready-workspaces** link.

6. Navigate to the **codeready-workspaces** instance using the link displayed under the **Red Hat CodeReady Workspaces URL** output.



NOTE

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

Verification

1. To verify the CodeReady Workspaces instance has installed correctly, navigate to the **CodeReady Workspaces Cluster** tab of the **Operator details** page. The **CheClusters** page displays the list of CodeReady Workspaces instances and their status.

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

3. See the content of the following fields:

   - The **Message** field contains error messages. The expected content is **None**.

- The **Red Hat CodeReady Workspaces URL**field contains the URL of the Red Hat CodeReady Workspaces instance. The URL appears when the deployment finishes successfully.

4. Navigate to the **Resources** tab. View the list of resources assigned to the CodeReady Workspaces deployment and their status.

Additional resources

- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.15/html-single/end-user_guide/index#navigating-che-using-the-dashboard.adoc.
- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.15/html-single/administration_guide/index#viewing-operator-events.adoc.

3.2. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT 4 USING THE CLI

This section describes how to install CodeReady Workspaces on OpenShift 4 with the **crwctl** CLI management tool.

Prerequisites

- An OpenShift cluster with an administrator account.
- **oc** is available. See Getting started with the OpenShift CLI. **oc** version must match the OpenShift cluster version.
- You have logged in to OpenShift. See Logging in to the CLI .
- **crwctl** is available. See Section 3.3.1, “Installing the crwctl CLI management tool” .

Procedure

- Run the **server:deploy** command to create the CodeReady Workspaces instance:

```
$ crwctl server:deploy -n openshift-workspaces
```

Verification steps

1. The output of the **server:deploy** command ends with:

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

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

3.3. INSTALLING CODEREADY WORKSPACES ON OPENSHIFT CONTAINER PLATFORM 3.11

3.3.1. 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 2.15.
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.

3.3.2. 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).

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 3.3.1, “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*.

- Use the openshift-workspaces namespace as the default installation project.
- Configure OpenShift to pull images from **registry.redhat.com**. See Red Hat Container Registry Authentication.

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 default project called openshift-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>**.

3.4. 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 3.4.3, “Preparing CodeReady Workspaces Custom Resource for installing behind a proxy”.

3.4.1. 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”.

3.4.2. 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.

3.4.2.1. 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.

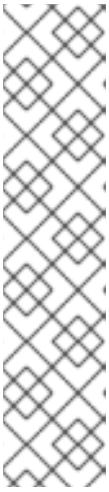
Table 3.1. Placeholders used in examples

<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>
```



NOTE

If you encounter an error, like **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-** or **plugin-**
Workspaces	**stacks-**, **plugin-**

Table 3.3. Images to copy in the private registry

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/backup-rhel8@sha256:6b636d6bba6c509756803c4186960ed69adfa2eae42dde5af48b67c6e0794915	**backup-rhel8@sha256:6b636d6bba6c509756803c4186960ed69adfa2eae42dde5af48b67c6e0794915**
registry.redhat.io/codeready-workspaces/configbump-rhel8@sha256:15574551ec79aa8cd9e0eea3d379fc7a77a4e16cc92f937b4a89c6f9a6f2ce40	**configbump-rhel8@sha256:15574551ec79aa8cd9e0eea3d379fc7a77a4e16cc92f937b4a89c6f9a6f2ce40**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:1c8b06e457ba008f86e5fefc82013acdb4639317cde809e926931d202a194a17	**crw-2-rhel8-operator@sha256:1c8b06e457ba008f86e5fefc82013acdb4639317cde809e926931d202a194a17**
registry.redhat.io/codeready-workspaces/dashboard-rhel8@sha256:e46636f0c66221d9a01506b114e18f3d3afe61da99bf224e71cf4051235e51ac	**dashboard-rhel8@sha256:e46636f0c66221d9a01506b114e18f3d3afe61da99bf224e71cf4051235e51ac**
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:fdcd72766757f08486a6e4dbf3a24bf084aefdb2e86971c440048aec0315d7e8	**devfileregistry-rhel8@sha256:fdcd72766757f08486a6e4dbf3a24bf084aefdb2e86971c440048aec0315d7e8**
registry.redhat.io/codeready-workspaces/idea-rhel8@sha256:eff6db1da4c9743ff77b91acf08378bce6a652826b3b252512e63f767de07785	**idea-rhel8@sha256:eff6db1da4c9743ff77b91acf08378bce6a652826b3b252512e63f767de07785**
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:6176e28c4c02f0a40f8192088ccb505ce5722258bcaab0addff9bafa310c1ca4	**jwtproxy-rhel8@sha256:6176e28c4c02f0a40f8192088ccb505ce5722258bcaab0addff9bafa310c1ca4**
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:dc0e082c9522158cb12345b1d184c3803d8a4a63a7189940e853e51557e43acf	**machineexec-rhel8@sha256:dc0e082c9522158cb12345b1d184c3803d8a4a63a7189940e853e51557e43acf**
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:315273182e1f4dc884365fc3330ada3937b40369f3faf7762847ec433c3ac537	**plugin-java11-rhel8@sha256:315273182e1f4dc884365fc3330ada3937b40369f3faf7762847ec433c3ac537**
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:8cb1e495825051b83cf903bb317e55823a6f57b3bad92e9407dc8fa59c24c0cc	**plugin-java8-rhel8@sha256:8cb1e495825051b83cf903bb317e55823a6f57b3bad92e9407dc8fa59c24c0cc**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:75fe8823dea867489b68169b764dc8b0b03290a456e9bfec5fe0cc413eec7355	**plugin-kubernetes-rhel8@sha256:75fe8823dea867489b68169b764dc8b0b03290a456e9bfec5fe0cc413eec7355**
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:d7603582f7ace76283641809b0c61dbcb78621735e536b789428e5a910d35af3	**plugin-openshift-rhel8@sha256:d7603582f7ace76283641809b0c61dbcb78621735e536b789428e5a910d35af3**
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:6d13003539fcbda201065eae2e66dc67fed007ba3ba41fb3b8ec841650c52bc2	**pluginbroker-artifacts-rhel8@sha256:6d13003539fcbda201065eae2e66dc67fed007ba3ba41fb3b8ec841650c52bc2**
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:de8ede01ce5d3b06ae8b1866bb482bb937f020f7dee5dfb20b041f02c1e63f68	**pluginbroker-metadata-rhel8@sha256:de8ede01ce5d3b06ae8b1866bb482bb937f020f7dee5dfb20b041f02c1e63f68**
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:cbb82d5bcea22d6d65644c2a4c88ce1e3a082e8a696217d6a104b67daa60384e	**pluginregistry-rhel8@sha256:cbb82d5bcea22d6d65644c2a4c88ce1e3a082e8a696217d6a104b67daa60384e**
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:e1694549ca2af22a1d1780cc7d92bb0829a411f74377f825eab3e0fba7c020d9	**server-rhel8@sha256:e1694549ca2af22a1d1780cc7d92bb0829a411f74377f825eab3e0fba7c020d9**
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:c2f38140f52112b2a7688c2a179afcaa930ad6216925eb322cfd9634a71cfc13	**stacks-cpp-rhel8@sha256:c2f38140f52112b2a7688c2a179afcaa930ad6216925eb322cfd9634a71cfc13**
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:f48fe1caa5be1ae91140681bee159ca8b11dc687fa50fbf9dc5644f4852bf5c8	**stacks-dotnet-rhel8@sha256:f48fe1caa5be1ae91140681bee159ca8b11dc687fa50fbf9dc5644f4852bf5c8**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:db76d04752973223e2c0de9401ebf06b84263e1bb6d29f1455daaff0cb39c1b3	**stacks-golang-rhel8@sha256:db76d04752973223e2c0de9401ebf06b84263e1bb6d29f1455daaff0cb39c1b3**
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:d120c41ee8dd80fb960dd4c1657bede536d32f13f3c3ca84e986a830ec2ead3b	**stacks-php-rhel8@sha256:d120c41ee8dd80fb960dd4c1657bede536d32f13f3c3ca84e986a830ec2ead3b**
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:5d26cf000924716d8d03969121a4c636e7fc8ef08aa21148eafa28a2c4aeaff7	**theia-endpoint-rhel8@sha256:5d26cf000924716d8d03969121a4c636e7fc8ef08aa21148eafa28a2c4aeaff7**
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:6000d00ef1029583642c01fec588f92addb95f16d56d0c23991a8f19314b0f06	**theia-rhel8@sha256:6000d00ef1029583642c01fec588f92addb95f16d56d0c23991a8f19314b0f06**
registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:70215465e2ad65a61d1b5401378532a3a10aa60afdda0702fb6061d89b8ba3be	**traefik-rhel8@sha256:70215465e2ad65a61d1b5401378532a3a10aa60afdda0702fb6061d89b8ba3be**
registry.redhat.io/devworkspace/devworkspace-rhel8-operator@sha256:3f96fb70c3f56dea3384ea31b9252a5c6aca8e0f33dc53be590f134912244078	**devworkspacedevworkspace-rhel8-operator@sha256:3f96fb70c3f56dea3384ea31b9252a5c6aca8e0f33dc53be590f134912244078**
registry.redhat.io/jboss-eap-7/eap-xp3-openjdk11-openshift-rhel8@sha256:bb3072afdbf31ddd1071fea37ed5308db3bf8a2478b5aa5aff8373e8042d6aeb	**eap-xp3-openjdk11-openshift-rhel8@sha256:bb3072afdbf31ddd1071fea37ed5308db3bf8a2478b5aa5aff8373e8042d6aeb**
registry.redhat.io/jboss-eap-7/eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d	**eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d**

<source-image>	*<target-image>*
registry.redhat.io/openshift4/ose-kube-rbac-proxy@sha256:1dc542b5ab33368443f698305a90c617385b4e9b101acc4acc0aa7b4bf58a292	**openshift4ose-kube-rbac-proxy@sha256:1dc542b5ab33368443f698305a90c617385b4e9b101acc4acc0aa7b4bf58a292**
registry.redhat.io/openshift4/ose-oauth-proxy@sha256:83988048d5f585ca936442963e23b77520e1e4d8c3d5b8160e43ae834a24b720	**openshift4ose-oauth-proxy@sha256:83988048d5f585ca936442963e23b77520e1e4d8c3d5b8160e43ae834a24b720**
registry.redhat.io/rh-sso-7/sso75-openshift-rhel8@sha256:dd4ea229521fb58dda7e547ea6db993156f4c61aa8a00f2fd1375bb77168b6e6	**sso75-openshift-rhel8@sha256:dd4ea229521fb58dda7e547ea6db993156f4c61aa8a00f2fd1375bb77168b6e6**
registry.redhat.io/rhel8/postgresql-13@sha256:6032adb3eac903ee8aa61f296ca9aaa57f5709e5673504b609222e042823f195	**postgresql-13@sha256:6032adb3eac903ee8aa61f296ca9aaa57f5709e5673504b609222e042823f195**
registry.redhat.io/rhel8/postgresql-96@sha256:314747a4a64ac16c33ead6a34479dccf16b9a07abf440ea7eeef7cda4cd19e32	**postgresql-96@sha256:314747a4a64ac16c33ead6a34479dccf16b9a07abf440ea7eeef7cda4cd19e32**
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	**mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73**
registry.redhat.io/ubi8/ubi-minimal@sha256:2e4bbb2be6e7aff711ddc93f0b07e49c93d41e4c2ffc8ea75f804ad6fe25564e	**ubi8ubi-minimal@sha256:2e4bbb2be6e7aff711ddc93f0b07e49c93d41e4c2ffc8ea75f804ad6fe25564e**

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 link: - CodeReady Workspaces Operator ClusterServiceVersion sources.

3.4.2.2. Preparing CodeReady Workspaces Custom Resource for restricted environment

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

3.4.2.2.1. Downloading the default**CheCluster** Custom Resource

Procedure

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

3.4.2.2.2. Customizing the **CheCluster** Custom Resource for restricted environment

Prerequisites

- All required images available in an image registry that is visible to the OpenShift cluster where CodeReady Workspaces is to be deployed. This is described in Section 3.4.2.1, "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>'
# [...]
```

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

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

Prerequisites

- CodeReady Workspaces CLI management tool is installed. See Section 3.3.1, "Installing the crwctl CLI management tool".
- The **oc** tool is installed.
- Access to an OpenShift instance.

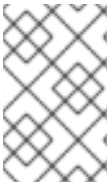
Procedure

1. Log in to OpenShift Container Platform:

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

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

```
$ crwctl server:start \
  --che-operator-image=<target-registry>/<target-organization>/crw-2-rhel8-operator:2.15 \
  --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.

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

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

Procedure

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

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

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

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

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

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

CHAPTER 4. CONFIGURING CODEREADY WORKSPACES

The following chapter describes configuration methods and options for Red Hat CodeReady Workspaces.

- 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.

Specific use-cases:

- Section 4.2, “Configuring workspace target project”
- Section 4.6, “Configuring the number of workspaces that a user can create”
- Section 4.5, “Configuring the number of workspaces that a user can run”
- Section 4.8, “Configuring workspaces nodeSelector”
- Section 4.9, “Configuring Red Hat CodeReady Workspaces server hostname”
- Section 4.10, “Configuring OpenShift Route”
- Section 4.11, “Configuring OpenShift Route to work with Router Sharding”
- Section 4.12, “Deploying CodeReady Workspaces with support for Git repositories with self-signed certificates”
- Section 4.13, “Installing CodeReady Workspaces using storage classes”
- Section 4.4, “Configuring storage types”
- Section 4.14, “Importing untrusted TLS certificates to CodeReady Workspaces”
- Section 4.15, “Configuring communication between CodeReady Workspaces components”
- Section 4.16, “Setting up the RH-SSO codeready-workspaces-username-readonly theme for the Red Hat CodeReady Workspaces login page”
- Section 4.17, “Mounting a Secret or a ConfigMap as a file or an environment variable into a CodeReady Workspaces container”
- Section 4.18, “Enabling Dev Workspace operator”

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.

4.1.2.1. CodeReady Workspaces server

4.1.2.1.1. CHE_API

API service. Browsers initiate REST communications to CodeReady Workspaces server with this URL.

Default

http://${CHE_HOST}:${CHE_PORT}/api

4.1.2.1.2. CHE_API_INTERNAL

API service internal network URL. Back-end services should initiate REST communications to CodeReady Workspaces server with this URL

Default

NULL

4.1.2.1.3. CHE_WEBSOCKET_ENDPOINT

CodeReady Workspaces WebSocket major endpoint. Provides basic communication endpoint for major WebSocket interactions and messaging.

Default

ws://${CHE_HOST}:${CHE_PORT}/api/websocket

4.1.2.1.4. CHE_WEBSOCKET_INTERNAL_ENDPOINT

CodeReady Workspaces WebSocket major internal endpoint. Provides basic communication endpoint for major WebSocket interactions and messaging.

Default

NULL

4.1.2.1.5. CHE_WORKSPACE_PROJECTS_STORAGE

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.

Default

/projects

4.1.2.1.6. CHE_WORKSPACE_PROJECTS_STORAGE_DEFAULT_SIZE

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.)

Default

1Gi

4.1.2.1.7. CHE_WORKSPACE_LOGS_ROOT__DIR

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.

Default

/workspace_logs

4.1.2.1.8. CHE_WORKSPACE_HTTP__PROXY

Configures environment variable HTTP_PROXY to a specified value in containers powering workspaces.

Default

empty

4.1.2.1.9. CHE_WORKSPACE_HTTPS__PROXY

Configures environment variable HTTPS_PROXY to a specified value in containers powering workspaces.

Default

empty

4.1.2.1.10. CHE_WORKSPACE_NO__PROXY

Configures environment variable NO_PROXY to a specified value in containers powering workspaces.

Default

empty

4.1.2.1.11. CHE_WORKSPACE_AUTO__START

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.

Default

true

4.1.2.1.12. CHE_WORKSPACE_POOL_TYPE

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**.

Default

fixed

4.1.2.1.13. CHE_WORKSPACE_POOL_EXACT__SIZE

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**.

Default

30

4.1.2.1.14. CHE_WORKSPACE_POOL_CORES__MULTIPLIER

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**.

Default

2

4.1.2.1.15. CHE_WORKSPACE_PROBE__POOL__SIZE

This property specifies how many threads to use for workspace server liveness probes.

Default

10

4.1.2.1.16. CHE_WORKSPACE_HTTP__PROXY__JAVA__OPTIONS

HTTP proxy setting for workspace JVM.

Default

NULL

4.1.2.1.17. CHE_WORKSPACE_JAVA__OPTIONS

Java command-line options added to JVMs running in workspaces.

Default

-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

4.1.2.1.18. CHE_WORKSPACE_MAVEN__OPTIONS

Maven command-line options added to JVMs running agents in workspaces.

Default

-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

4.1.2.1.19. CHE_WORKSPACE_DEFAULT__MEMORY__LIMIT__MB

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.

Default

1024

4.1.2.1.20. CHE_WORKSPACE_DEFAULT__MEMORY__REQUEST__MB

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.

Default

200

4.1.2.1.21. CHE_WORKSPACE_DEFAULT__CPU__LIMIT__CORES

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.

Default

-1

4.1.2.1.22. CHE_WORKSPACE_DEFAULT__CPU__REQUEST__CORES

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.

Default

-1

4.1.2.1.23. CHE_WORKSPACE_SIDECAR_DEFAULT__MEMORY__LIMIT__MB

RAM limit 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.

Default

128

4.1.2.1.24. CHE_WORKSPACE_SIDECAR_DEFAULT__MEMORY__REQUEST__MB

RAM request for each sidecar that has no RAM settings in the CodeReady Workspaces plug-in configuration.

Default

64

4.1.2.1.25. CHE_WORKSPACE_SIDECAR_DEFAULT__CPU__LIMIT__CORES

CPU limit 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.

Default

-1

4.1.2.1.26. CHE_WORKSPACE_SIDECAR_DEFAULT__CPU__REQUEST__CORES

CPU 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**.

Default

-1

4.1.2.1.27. CHE_WORKSPACE_SIDECAR_IMAGE__PULL__POLICY

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.

Default

Always

4.1.2.1.28. CHE_WORKSPACE_ACTIVITY__CHECK__SCHEDULER__PERIOD__S

Period of inactive workspaces suspend job execution.

Default

60

4.1.2.1.29. CHE_WORKSPACE_ACTIVITY__CLEANUP__SCHEDULER__PERIOD__S

The period of the cleanup of the activity table. The activity table can contain invalid or stale data if some unforeseen errors happen, as a server failure at a peculiar point in time. The default is to run the cleanup job every hour.

Default

3600

4.1.2.1.30. CHE_WORKSPACE_ACTIVITY__CLEANUP__SCHEDULER__INITIAL__DELAY__S

The delay after server startup to start the first activity clean up job.

Default

60

4.1.2.1.31. CHE_WORKSPACE_ACTIVITY__CHECK__SCHEDULER__DELAY__S

Delay before first workspace idleness check job started to avoid mass suspend if CodeReady Workspaces server was unavailable for period close to inactivity timeout.

Default

180

4.1.2.1.32. CHE_WORKSPACE_CLEANUP__TEMPORARY__INITIAL__DELAY__MIN

Time to delay the first execution of temporary workspaces cleanup job.

Default

5

4.1.2.1.33. CHE_WORKSPACE_CLEANUP__TEMPORARY__PERIOD__MIN

Time to delay between the termination of one execution and the commencement of the next execution of temporary workspaces cleanup job

Default

180

4.1.2.1.34. CHE_WORKSPACE_SERVER_PING__SUCCESS__THRESHOLD

Number of sequential successful pings to server after which it is treated as available. the CodeReady Workspaces Operator: the property is common for all servers, for example, workspace agent, terminal, exec.

Default

1

4.1.2.1.35. CHE_WORKSPACE_SERVER_PING__INTERVAL__MILLISECONDS

Interval, in milliseconds, between successive pings to workspace server.

Default

3000

4.1.2.1.36. CHE_WORKSPACE_SERVER_LIVENESS__PROBES

List of servers names which require liveness probes

Default

wsagent/http,exec-agent/http,terminal,theia,jupyter,dirigible,cloud-shell,intellij

4.1.2.1.37. CHE_WORKSPACE_STARTUP__DEBUG__LOG__LIMIT__BYTES

Limit size of the logs collected from single container that can be observed by che-server when debugging workspace startup. default 10MB=10485760

Default

10485760

4.1.2.1.38. CHE_WORKSPACE_STOP_ROLE_ENABLED

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.

Default

true

4.1.2.1.39. CHE_DEVWORKSPACES_ENABLED

Specifies whether CodeReady Workspaces is deployed with DevWorkspaces enabled. This property is set by the CodeReady Workspaces Operator if it also installed the support for DevWorkspaces. This property is used to advertise this fact to the CodeReady Workspaces dashboard. It does not make sense to change the value of this property manually.

Default

false

4.1.2.2. Authentication parameters

4.1.2.2.1. CHE_AUTH_USER__SELF__CREATION

CodeReady Workspaces has a single identity implementation, so this does not change the user experience. If true, enables user creation at API level

Default

false

4.1.2.2.2. CHE_AUTH_ACCESS__DENIED__ERROR__PAGE

Authentication error page address

Default

/error-oauth

4.1.2.2.3. CHE_AUTH_RESERVED__USER__NAMES

Reserved user names

Default

empty

4.1.2.2.4. CHE_OAUTH2_GITHUB_CLIENTID__FILEPATH

Configuration of GitHub OAuth2 client. Used to obtain Personal access tokens. Location of the file with GitHub client id.

Default

NULL

4.1.2.2.5. CHE_OAUTH2_GITHUB_CLIENTSECRET__FILEPATH

Location of the file with GitHub client secret.

Default

NULL

4.1.2.2.6. CHE_OAUTH_GITHUB_AUTHURI

GitHub OAuth authorization URI.

Default

https://github.com/login/oauth/authorize

4.1.2.2.7. CHE_OAUTH_GITHUB_TOKENURI

GitHub OAuth token URI.

Default

https://github.com/login/oauth/access_token

4.1.2.2.8. CHE_OAUTH_GITHUB_REDIRECTURIS

GitHub OAuth redirect URIs. Separate multiple values with comma, for example: URI,URI,URI

Default

http://localhost:${CHE_PORT}/api/oauth/callback

4.1.2.2.9. CHE_OAUTH_OPENSHIFT_CLIENTID

Configuration of OpenShift OAuth client. Used to obtain OpenShift OAuth token. OpenShift OAuth client ID.

Default

NULL

4.1.2.2.10. CHE_OAUTH_OPENSHIFT_CLIENTSECRET

Configurationof OpenShift OAuth client. Used to obtain OpenShift OAuth token. OpenShift OAuth client ID. OpenShift OAuth client secret.

Default

NULL

4.1.2.2.11. CHE_OAUTH_OPENSHIFT_OAUTH__ENDPOINT

ConfigurationofOpenShift OAuth client. Used to obtain OpenShift OAuth token. OpenShift OAuth client ID. OpenShift OAuth client secret. OpenShift OAuth endpoint.

Default

NULL

4.1.2.2.12. CHE_OAUTH_OPENSHIFT_VERIFY__TOKEN__URL

ConfigurationofOpenShiftOAuth client. Used to obtain OpenShift OAuth token. OpenShift OAuth client ID. OpenShift OAuth client secret. OpenShift OAuth endpoint. OpenShift OAuth verification token URL.

Default

NULL

4.1.2.2.13. CHE_OAUTH1_BITBUCKET_CONSUMERKEYPATH

Configuration of Bitbucket Server OAuth1 client. Used to obtain Personal access tokens. Location of the file with Bitbucket Server application consumer key (equivalent to a username).

Default

NULL

4.1.2.2.14. CHE_OAUTH1_BITBUCKET_PRIVATEKEYPATH

Configurationof Bitbucket Server OAuth1 client. Used to obtain Personal access tokens. Location of the file with Bitbucket Server application consumer key (equivalent to a username). Location of the file with Bitbucket Server application private key

Default

NULL

4.1.2.2.15. CHE_OAUTH1_BITBUCKET_ENDPOINT

ConfigurationofBitbucket Server OAuth1 client. Used to obtain Personal access tokens. Location of the file with Bitbucket Server application consumer key (equivalent to a username). Location of the file with Bitbucket Server application private key Bitbucket Server URL. To work correctly with factories the same URL has to be part of **che.integration.bitbucket.server_endpoints** too.

Default

NULL

4.1.2.3. Internal

4.1.2.3.1. SCHEDULE_CORE__POOL__SIZE

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.

Default

10

4.1.2.3.2. DB_SCHEMA_FLYWAY_BASELINE_ENABLED

DB initialization and migration configuration If true, ignore scripts up to the version configured by baseline.version.

Default

true

4.1.2.3.3. DB_SCHEMA_FLYWAY_BASELINE_VERSION

Scripts with version up to this are ignored. Note that scripts with version equal to baseline version are also ignored.

Default

5.0.0.8.1

4.1.2.3.4. DB_SCHEMA_FLYWAY_SCRIPTS_PREFIX

Prefix of migration scripts.

Default

empty

4.1.2.3.5. DB_SCHEMA_FLYWAY_SCRIPTS_SUFFIX

Suffix of migration scripts.

Default

.sql

4.1.2.3.6. DB_SCHEMA_FLYWAY_SCRIPTS_VERSION__SEPARATOR

Separator of version from the other part of script name.

Default

—

4.1.2.3.7. DB_SCHEMA_FLYWAY_SCRIPTS_LOCATIONS

Locations where to search migration scripts.

Default

classpath:che-schema

4.1.2.4. OpenShift Infra parameters

4.1.2.4.1. CHE_INFRA_KUBERNETES_MASTER__URL

Configuration of OpenShift client master URL that Infra will use.

Default

empty

4.1.2.4.2. CHE_INFRA_KUBERNETES_TRUST__CERTS

Boolean to configure OpenShift client to use trusted certificates.

Default

false

4.1.2.4.3. CHE_INFRA_KUBERNETES_CLUSTER__DOMAIN

OpenShift cluster domain. If not set, svc names will not contain information about the cluster domain.

Default

NULL

4.1.2.4.4. CHE_INFRA_KUBERNETES_SERVER__STRATEGY

Defines the way how servers are exposed to the world in Kubernetes infra. List of strategies implemented in CodeReady Workspaces: **default-host**, **multi-host**, **single-host**.

Default

multi-host

4.1.2.4.5. CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_EXPOSURE

Defines the way in which the workspace plugins and editors are exposed in the single-host mode. Supported exposures: **native**: Exposes servers using OpenShift Routees. Works only on Kubernetes. **gateway**: Exposes servers using reverse-proxy gateway.

Default

native

4.1.2.4.6. CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE__ENDPOINT__EXPOSURE

Defines the way how to expose devfile endpoints, as 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

Default

multi-host

4.1.2.4.7. CHE_INFRA_KUBERNETES_SINGLEHOST_GATEWAY_CONFIGMAP__LABELS

Defines labels which will be set to ConfigMaps configuring single-host gateway.

Default

app=che,component=che-gateway-config

4.1.2.4.8. 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**

Default

empty

4.1.2.4.9. CHE_INFRA_KUBERNETES_NAMESPACE_CREATION__ALLOWED

Indicates whether CodeReady Workspaces server is allowed to create project for user workspaces, or they're intended to be created manually by cluster administrator. This property is also used by the OpenShift infra.

Default

true

4.1.2.4.10. CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT

Defines default OpenShift project in which user's workspaces are created if user does not override it. It's possible to use **<username>** and **<userid>** placeholders (for example: **che-workspace-<username>**). In that case, new namespace will be created for each user. Used by OpenShift infra as well to specify a Project. The **<username>** or **<userid>** placeholder is mandatory.

Default

<username>-che

4.1.2.4.11. CHE_INFRA_KUBERNETES_NAMESPACE_LABEL

Defines whether che-server should try to label the workspace namespaces.

Default

true

4.1.2.4.12. CHE_INFRA_KUBERNETES_NAMESPACE_ANNOTATE

Defines whether che-server should try to annotate the workspace namespaces.

Default

true

4.1.2.4.13. CHE_INFRA_KUBERNETES_NAMESPACE_LABELS

List of labels to find project that are used for CodeReady Workspaces Workspaces. They are used to: - find prepared project for users in combination with **che.infra.kubernetes.namespace.annotations**. - actively label project with any workspace.

Default

app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspaces-namespace

4.1.2.4.14. CHE_INFRA_KUBERNETES_NAMESPACE_ANNOTATIONS

List of annotations to find project prepared for CodeReady Workspaces users workspaces. Only project matching the **che.infra.kubernetes.namespace.labels** will be matched against these annotations. project that matches both **che.infra.kubernetes.namespace.labels** and **che.infra.kubernetes.namespace.annotations** will be preferentially used for User's workspaces. It's possible to use **<username>** placeholder to specify the project to concrete user. They are used to: - find prepared project for users in combination with **che.infra.kubernetes.namespace.labels**. - actively annotate project with any workspace.

Default

che.eclipse.org/username=<username>

4.1.2.4.15. CHE_INFRA_KUBERNETES_SERVICE__ACCOUNT__NAME

Defines Kubernetes Service Account name which should be specified to be bound to all workspaces Pods. the CodeReady Workspaces Operator that OpenShift Infrastructure will not 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

Default

NULL

4.1.2.4.16. CHE_INFRA_KUBERNETES_WORKSPACE__SA__CLUSTER__ROLES

Specifies optional, additional cluster roles to use with the workspace service account. the CodeReady Workspaces Operator 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**.

Default

NULL

4.1.2.4.17. CHE_INFRA_KUBERNETES_USER__CLUSTER__ROLES

Cluster roles to assign to user in his namespace

Default

NULL

4.1.2.4.18. CHE_INFRA_KUBERNETES_WORKSPACE__START__TIMEOUT__MIN

Defines wait time that limits the Kubernetes workspace start time.

Default

8

4.1.2.4.19. CHE_INFRA_KUBERNETES_INGRESS__START__TIMEOUT__MIN

Defines the timeout in minutes that limits the period for which OpenShift Route become ready

Default

5

4.1.2.4.20. CHE_INFRA_KUBERNETES_WORKSPACE__UNRECOVERABLE__EVENTS

If during workspace startup an unrecoverable event defined in the property occurs, stop the workspace immediately rather than waiting until timeout. the CodeReady Workspaces Operator 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.

Default

FailedMount,FailedScheduling,MountVolume.SetUpfailed,Failed to pull image,FailedCreate,ReplicaSetCreateError

4.1.2.4.21. CHE_INFRA_KUBERNETES_PVC_ENABLED

Defines whether use the Persistent Volume Claim for CodeReady Workspaces workspace needs, for example: backup projects, logs, or disable it.

Default

true

4.1.2.4.22. CHE_INFRA_KUBERNETES_PVC_STRATEGY

Defined which strategy will be used while choosing PVC for workspaces. Supported strategies: **common**: All workspaces in the same project will reuse the same PVC. Name of PVC may be configured with **che.infra.kubernetes.pvc.name**. Existing PVC will be used or a new one will be created if it does not 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 does not 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.

Default

common

4.1.2.4.23. CHE_INFRA_KUBERNETES_PVC_PRECREATE__SUBPATHS

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 as workspace subpath volume mounts are created with root permissions, and therefore 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 creates

subdirectories with user permissions. See: subPath in volumeMount is not writable for non-root users #41638 the CodeReady Workspaces Operator that this property has effect only if the **common** PVC strategy used.

Default

true

4.1.2.4.24. CHE_INFRA_KUBERNETES_PVC_NAME

Defines the settings of PVC name for CodeReady Workspaces workspaces. Each PVC strategy supplies this value differently. See documentation for **che.infra.kubernetes.pvc.strategy** property

Default

claim-che-workspace

4.1.2.4.25. CHE_INFRA_KUBERNETES_PVC_STORAGE__CLASS__NAME

Defines the storage class of Persistent Volume Claim for the workspaces. Empty strings means "use default".

Default

empty

4.1.2.4.26. CHE_INFRA_KUBERNETES_PVC_QUANTITY

Defines the size of Persistent Volume Claim of CodeReady Workspaces workspace. See: Understanding persistent storage

Default

10Gi

4.1.2.4.27. CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE

Pod that is launched when performing persistent volume claim maintenance jobs on OpenShift

Default

registry.access.redhat.com/ubi8-minimal:8.3-230

4.1.2.4.28. CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE_PULL__POLICY

Image pull policy of container that used for the maintenance jobs on OpenShift cluster

Default

IfNotPresent

4.1.2.4.29. CHE_INFRA_KUBERNETES_PVC_JOBS_MEMORYLIMIT

Defines Pod memory limit for persistent volume claim maintenance jobs

Default

250Mi

4.1.2.4.30. CHE_INFRA_KUBERNETES_PVC_ACCESS__MODE

Defines Persistent Volume Claim access mode. the CodeReady Workspaces Operator that for common PVC strategy changing of access mode affects the number of simultaneously running workspaces. If the OpenShift instance running CodeReady Workspaces is using Persistent Volumes with RWX access mode, then a limit of running workspaces at the same time is bounded only by CodeReady Workspaces limits configuration: RAM, CPU, and so on. See: Understanding persistent storage

Default

ReadWriteOnce

4.1.2.4.31. CHE_INFRA_KUBERNETES_PVC_WAIT__BOUND

Defines if CodeReady Workspaces Server should wait workspaces Persistent Volume Claims to become bound after creating. Default value is **true**. The parameter is used by all Persistent Volume Claim strategies. It should be set to **false** when **volumeBindingMode** is configured to **WaitForFirstConsumer** otherwise workspace starts will hangs up on phase of waiting PVCs.

Default

true

4.1.2.4.32. CHE_INFRA_KUBERNETES_INGRESS_ANNOTATIONS__JSON

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 rather than Ingresses. the CodeReady Workspaces Operator 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 do not need to support changing the app root). The **che.infra.kubernetes.ingress.path.rewrite_transform** property defines how the path of the ingress should be transformed to support the URL rewriting and this property defines the set of annotations on the ingress itself that instruct the chosen ingress controller to actually do the URL rewriting, potentially building on the path transformation (if required by the chosen ingress controller). For example for Nginx ingress controller 0.22.0 and later the following value is recommended: **{"ingress.kubernetes.io/rewrite-target": "/$1","ingress.kubernetes.io/ssl-redirect": "false",\ "ingress.kubernetes.io/proxy-connect-timeout": "3600","ingress.kubernetes.io/proxy-read-timeout": "3600", "nginx.org/websocket-services": "<service-name>"}** and the **che.infra.kubernetes.ingress.path.rewrite_transform** should be set to **"%s(.*)"**. For nginx ingress controller older than 0.22.0, the rewrite-target should be set to merely / and the path transform to **%s** (see the **che.infra.kubernetes.ingress.path.rewrite_transform** property). See the Nginx ingress controller documentation for the explanation of how the ingress controller uses the regular expression available in the ingress path and how it achieves the URL rewriting.

Default

NULL

4.1.2.4.33. CHE_INFRA_KUBERNETES_INGRESS_PATH__TRANSFORM

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**. 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.

Default

NULL

4.1.2.4.34. CHE_INFRA_KUBERNETES_INGRESS_LABELS

Additional labels to add into every Ingress created by CodeReady Workspaces server to allow clear identification.

Default

NULL

4.1.2.4.35. CHE_INFRA_KUBERNETES_POD_SECURITY__CONTEXT_RUN__AS__USER

Defines security context for Pods that will be created by OpenShift Infra This is ignored by OpenShift infra

Default

NULL

4.1.2.4.36. CHE_INFRA_KUBERNETES_POD_SECURITY__CONTEXT_FS__GROUP

Defines security context for Pods that will be created by OpenShift Infra. A special supplemental group that applies to all containers in a Pod. This is ignored by OpenShift infra.

Default

NULL

4.1.2.4.37. CHE_INFRA_KUBERNETES_POD_TERMINATION__GRACE__PERIOD__SEC

Defines grace termination period for Pods that will be created by OpenShift infrastructures. Default value: **0**. It allows to stop Pods quickly and significantly decrease the time required for stopping a workspace. the CodeReady Workspaces Operator: if **terminationGracePeriodSeconds** have been explicitly set in OpenShift recipe it will not be overridden.

Default

0

4.1.2.4.38. CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC__REQUESTS_MAX

Number of maximum concurrent asynchronous web requests (HTTP requests or ongoing WebSocket calls) supported in the underlying shared HTTP client of the **KubernetesClient** instances. Default values: **max=64**, and **max_per_host:5**. Default values are not suitable for multi-user scenarios, as CodeReady Workspaces keeps open connections, for example for command or ws-agent logs.

Default

1000

4.1.2.4.39. CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC__REQUESTS_MAX__PER__HOST

Number of maximum concurrent asynchronous web requests per host.

Default

1000

4.1.2.4.40. CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION__POOL_MAX__IDLE

Max number of idle connections in the connection pool of the Kubernetes-client shared HTTP client.

Default

5

4.1.2.4.41. CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION__POOL_KEEP__ALIVE__MIN

Keep-alive timeout of the connection pool of the Kubernetes-client shared HTTP client in minutes.

Default

5

4.1.2.4.42. CHE_INFRA_KUBERNETES_TLS__ENABLED

Creates Ingresses with Transport Layer Security (TLS) enabled. In OpenShift infrastructure, Routes will be TLS-enabled.

Default

false

4.1.2.4.43. CHE_INFRA_KUBERNETES_TLS__SECRET

Name of a secret that should be used when creating workspace ingresses with TLS. This property is ignored by OpenShift infrastructure.

Default

empty

4.1.2.4.44. CHE_INFRA_KUBERNETES_TLS__KEY

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.

Default

NULL

4.1.2.4.45. CHE_INFRA_KUBERNETES_TLS__CERT

Certificate data for TLS Secret that should be used for workspaces Ingresses. Certificate should be encoded with Base64 algorithm. This property is ignored by OpenShift infrastructure.

Default

NULL

4.1.2.4.46. CHE_INFRA_KUBERNETES_RUNTIMES__CONSISTENCY__CHECK__PERIOD__MIN

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 Kubernetes 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 mounted to CodeReady Workspaces Server Pod. It DOES NOT work on the following configurations: - CodeReady Workspaces Server communicates with Kubernetes API using token from OAuth provider.

Default

-1

4.1.2.4.47. CHE_INFRA_KUBERNETES_TRUSTED__CA_SRC__CONFIGMAP

Name of the ConfigMap in CodeReady Workspaces server namespace with additional CA TLS certificates to be propagated into all user's workspaces. If the property is set on OpenShift 4 infrastructure, and **che.infra.openshift.trusted_ca.dest_configmap_labels** includes the **config.openshift.io/inject-trusted-cabundle=true** label, then cluster CA bundle will be propagated too.

Default

NULL

4.1.2.4.48. CHE_INFRA_KUBERNETES_TRUSTED__CA_DEST__CONFIGMAP

Name of the ConfigMap in a workspace namespace with additional CA TLS certificates. Holds the copy of **che.infra.kubernetes.trusted_ca.src_configmap** but in a workspace namespace. Content of this ConfigMap is mounted into all workspace containers including plugin brokers. Do not change the ConfigMap name unless it conflicts with the already existing ConfigMap. the CodeReady Workspaces Operator that the resulting ConfigMap name can be adjusted eventually to make it unique in project. The original name would be stored in **che.original_name** label.

Default

ca-certs

4.1.2.4.49. CHE_INFRA_KUBERNETES_TRUSTED__CA_MOUNT__PATH

Configures path on workspace containers where the CA bundle should be mounted. Content of ConfigMap specified by **che.infra.kubernetes.trusted_ca.dest_configmap** is mounted.

Default

/public-certs

4.1.2.4.50. CHE_INFRA_KUBERNETES_TRUSTED__CA_DEST__CONFIGMAP__LABELS

Comma separated list of labels to add to the CA certificates ConfigMap in user workspace. See the **che.infra.kubernetes.trusted_ca.dest_configmap** property.

Default

empty

4.1.2.5. OpenShift Infra parameters

4.1.2.5.1. CHE_INFRA_OPENSHIFT_TRUSTED__CA_DEST__CONFIGMAP__LABELS

Comma separated list of labels to add to the CA certificates ConfigMap in user workspace. See **che.infra.kubernetes.trusted_ca.dest_configmap** property. This default value is used for automatic cluster CA bundle injection in OpenShift 4.

Default

config.openshift.io/inject-trusted-cabundle=true

4.1.2.5.2. CHE_INFRA_OPENSHIFT_ROUTE_LABELS

Additional labels to add into every Route created by CodeReady Workspaces server to allow clear identification.

Default

NULL

4.1.2.5.3. CHE_INFRA_OPENSHIFT_ROUTE_HOST_DOMAIN__SUFFIX

The hostname that should be used as a suffix for the workspace routes. For example: Using **domain_suffix=*<codeready-__<openshift_deployment_name>__.__<domain_name>__>***, the route resembles: **routed3qrtk.*<codeready-__<openshift_deployment_name>__.__<domain_name>__>***. It has to be a valid DNS name.

Default

NULL

4.1.2.5.4. CHE_INFRA_OPENSHIFT_PROJECT_INIT__WITH__SERVER__SA

Initialize OpenShift project with CodeReady Workspaces server's service account if OpenShift OAuth is enabled.

Default

true

4.1.2.6. Experimental properties

4.1.2.6.1. CHE_WORKSPACE_PLUGIN__BROKER_METADATA_IMAGE

Docker image of CodeReady Workspaces plugin broker app that resolves workspace tools configuration and copies plugins dependencies to a workspace. The CodeReady Workspaces Operator overrides these images by default. Changing the images here will not have an effect if CodeReady Workspaces is installed using the Operator.

Default

quay.io/eclipse/che-plugin-metadata-broker:v3.4.0

4.1.2.6.2. CHE_WORKSPACE_PLUGIN__BROKER_ARTIFACTS_IMAGE

Docker image of CodeReady Workspaces plugin artifacts broker. This broker runs as an init container on the workspace Pod. Its job is to take in a list of plugin identifiers (either references to a plugin in the registry or a link to a plugin meta.yaml) and ensure that the correct .vsix and .theia extensions are downloaded into the /plugins directory, for each plugin requested for the workspace.

Default

quay.io/eclipse/che-plugin-artifacts-broker:v3.4.0

4.1.2.6.3. CHE_WORKSPACE_PLUGIN__BROKER_DEFAULT__MERGE__PLUGINS

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: 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 the **mergePlugins** attribute.

Default

false

4.1.2.6.4. CHE_WORKSPACE_PLUGIN__BROKER_PULL__POLICY

Docker image of CodeReady Workspaces plugin broker app that resolves workspace tools configuration and copies plugins dependencies to a workspace

Default

Always

4.1.2.6.5. CHE_WORKSPACE_PLUGIN__BROKER_WAIT__TIMEOUT__MIN

Defines the timeout in minutes that limits the max period of result waiting for plugin broker.

Default

3

4.1.2.6.6. CHE_WORKSPACE_PLUGIN__REGISTRY__URL

Workspace plug-ins 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 plug-ins registry is not needed value 'NULL' should be used

Default

https://che-plugin-registry.prod-preview.openshift.io/v3

4.1.2.6.7. CHE_WORKSPACE_PLUGIN__REGISTRY__INTERNAL__URL

Workspace plugins registry internal endpoint. Should be a valid HTTP URL. Example: http://devfile-registry.che.svc.cluster.local:8080 In case CodeReady Workspaces plug-ins registry is not needed value 'NULL' should be used

Default

NULL

4.1.2.6.8. CHE_WORKSPACE_DEVFILE__REGISTRY__URL

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 plug-ins registry is not needed value 'NULL' should be used

Default

https://che-devfile-registry.prod-preview.openshift.io/

4.1.2.6.9. CHE_WORKSPACE_DEVFILE__REGISTRY__INTERNAL__URL

Devfile Registry "internal" endpoint. Should be a valid HTTP URL. Example: http://plugin-registry.che.svc.cluster.local:8080 In case CodeReady Workspaces plug-ins registry is not needed value 'NULL' should be used

Default

NULL

4.1.2.6.10. CHE_WORKSPACE_STORAGE_AVAILABLE__TYPES

The configuration property that defines available values for storage types that clients such as the Dashboard should propose to users during workspace creation and update. Available values: - **persistent**: Persistent Storage slow I/O but persistent. - **ephemeral**: Ephemeral Storage allows for faster I/O but may have limited storage and is not persistent. - **async**: Experimental feature: Asynchronous storage is combination of Ephemeral and Persistent storage. Allows for faster I/O and keep your changes, will backup on stop and restore on start workspace. Will work only if: - **che.infra.kubernetes.pvc.strategy='common'** - **che.limits.user.workspaces.run.count=1** - **che.infra.kubernetes.namespace.default** contains **<username>** in other cases remove **async** from the list.

Default

persistent,ephemeral,async

4.1.2.6.11. CHE_WORKSPACE_STORAGE_PREFERRED__TYPE

The configuration property that defines a default value for storage type that clients such as the Dashboard should propose to users during workspace creation and update. The **async** value is an experimental feature, not recommended as default type.

Default

persistent

4.1.2.6.12. CHE_SERVER_SECURE__EXPOSER

Configures in which way secure servers will be protected with authentication. Suitable values: - **default**: **jwtproxy** is configured in a pass-through mode. Servers should authenticate requests themselves. - **jwtproxy**: **jwtproxy** will authenticate requests. Servers will receive only authenticated requests.

Default

jwtproxy

4.1.2.6.13. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_TOKEN_ISSUER

Jwtproxy issuer string, token lifetime, and optional auth page path to route unsigned requests to.

Default

wsmaster

4.1.2.6.14. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_TOKEN_TTL

JWTProxy issuer token lifetime.

Default

8800h

4.1.2.6.15. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_AUTH_LOADER_PATH

Optional authentication page path to route unsigned requests to.

Default

/_app/loader.html

4.1.2.6.16. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_IMAGE

JWTProxy image.

Default

quay.io/eclipse/che-jwtproxy:0.10.0

4.1.2.6.17. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_MEMORY__REQUEST

JWTProxy memory request.

Default

15mb

4.1.2.6.18. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_MEMORY__LIMIT

JWTProxy memory limit.

Default

128mb

4.1.2.6.19. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_CPU__REQUEST

JWTProxy CPU request.

Default

0.03

4.1.2.6.20. CHE_SERVER_SECURE__EXPOSER_JWTPROXY_CPU__LIMIT

JWTProxy CPU limit.

Default

0.5

4.1.2.7. Configuration of the major WebSocket endpoint

4.1.2.7.1. CHE_CORE_JSONRPC_PROCESSOR__MAX__POOL__SIZE

Maximum size of the JSON RPC processing pool in case if pool size would be exceeded message execution will be rejected

Default

50

4.1.2.7.2. CHE_CORE_JSONRPC_PROCESSOR__CORE__POOL__SIZE

Initial JSON processing pool. Minimum number of threads that used to process major JSON RPC messages.

Default

5

4.1.2.7.3. CHE_CORE_JSONRPC_PROCESSOR__QUEUE__CAPACITY

Configuration of queue used to process JSON RPC messages.

Default

100000

4.1.2.7.4. CHE_METRICS_PORT

Port the HTTP server endpoint that would be exposed with Prometheus metrics.

Default

8087

4.1.2.8. CORS settings

4.1.2.8.1. CHE_CORS_ALLOWED__ORIGINS

Indicates which request origins are allowed. CORS filter on WS Master is turned off by default. Use environment variable "CHE_CORS_ENABLED=true" to turn it on.

Default

*

4.1.2.8.2. CHE_CORS_ALLOW__CREDENTIALS

Indicates if it allows processing of requests with credentials (in cookies, headers, TLS client certificates).

Default

false

4.1.2.9. Factory defaults

4.1.2.9.1. CHE_FACTORY_DEFAULT__PLUGINS

Editor and plugin which will be used for factories that are created from a remote Git repository which does not contain any CodeReady Workspaces-specific workspace descriptor Multiple plugins must be comma-separated, for example:
pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion

Default

redhat/vscode-commons/latest

4.1.2.9.2. CHE_FACTORY_DEFAULT__DEVFILE__FILENAMES

Devfile filenames to look on repository-based factories (for example GitHub). Factory will try to locate those files in the order they enumerated in the property.

Default

devfile.yaml,.devfile.yaml

4.1.2.10. Devfile defaults

4.1.2.10.1. CHE_FACTORY_DEFAULT__EDITOR

Editor that will be used for factories that are created from a remote Git repository which does not contain any CodeReady Workspaces-specific workspace descriptor.

Default

eclipse/che-theia/latest

4.1.2.10.2. CHE_FACTORY_SCM__FILE__FETCHER__LIMIT__BYTES

File size limit for the URL fetcher which fetch files from the SCM repository.

Default

102400

4.1.2.10.3. CHE_FACTORY_DEVFILE2__FILES__RESOLUTION__LIST

Additional files which may be present in repository to complement devfile v2, and should be referenced as links to SCM resolver service in factory to retrieve them.

Default

.che/che-editor.yaml,.che/che-theia-plugins.yaml,.vscode/extensions.json

4.1.2.10.4. CHE_WORKSPACE_DEVFILE_DEFAULT__EDITOR

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.

Default

eclipse/che-theia/latest

4.1.2.10.5. CHE_WORKSPACE_DEVFILE_DEFAULT__EDITOR_PLUGINS

Default Plug-ins 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.

Default

NULL

4.1.2.10.6. CHE_WORKSPACE_PROVISION_SECRET_LABELS

Defines comma-separated list of labels for selecting secrets from a user namespace, which will be mount into workspace containers as a files or environment variables. Only secrets that match ALL given labels will be selected.

Default

app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspace-secret

4.1.2.10.7. CHE_WORKSPACE_DEVFILE_ASYNC_STORAGE_PLUGIN

Plugin is added in case asynchronous storage feature will be enabled in workspace configuration and supported by environment

Default

eclipse/che-async-pv-plugin/latest

4.1.2.10.8. CHE_INFRA_KUBERNETES_ASYNC_STORAGE_IMAGE

Docker image for the CodeReady Workspaces asynchronous storage

Default

quay.io/eclipse/che-workspace-data-sync-storage:0.0.1

4.1.2.10.9. CHE_WORKSPACE_POD_NODE__SELECTOR

Optionally configures node selector for workspace Pod. Format is comma-separated key=value pairs, for example: **disktype=ssd,cpu=xlarge,foo=bar**

Default

NULL

4.1.2.10.10. CHE_WORKSPACE_POD_TOLERATIONS__JSON

Optionally configures tolerations for workspace Pod. Format is a string representing a JSON Array of taint tolerations, or **NULL** to disable it. The objects contained in the array have to follow the toleration v1 core specifications. Example:
[{"effect":"NoExecute","key":"aNodeTaint","operator":"Equal","value":"aValue"}]

Default

NULL

4.1.2.10.11. CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN__TIMEOUT__MIN

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.

Default

120

4.1.2.10.12. CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN__CHECK__PERIOD__MIN

Defines the period with which the Asynchronous Storage Pod stopping ability will be performed (once in 30 minutes by default)

Default

30

4.1.2.10.13. CHE_INTEGRATION_BITBUCKET_SERVER__ENDPOINTS

Bitbucket endpoints used for factory integrations. Comma separated list of Bitbucket server URLs or NULL if no integration expected.

Default

NULL

4.1.2.10.14. CHE_INTEGRATION_GITLAB_SERVER__ENDPOINTS

GitLab endpoints used for factory integrations. Comma separated list of GitLab server URLs or NULL if no integration expected.

Default

NULL

4.1.2.10.15. CHE_INTEGRATION_GITLAB_OAUTH__ENDPOINT

Address of the GitLab server with configured OAuth 2 integration

Default

NULL

4.1.2.10.16. CHE_OAUTH2_GITLAB_CLIENTID__FILEPATH

Configuration of GitLab OAuth2 client. Used to obtain Personal access tokens. Location of the file with GitLab client id.

Default

NULL

4.1.2.10.17. CHE_OAUTH2_GITLAB_CLIENTSECRET__FILEPATH

Location of the file with GitLab client secret.

Default

NULL#

4.1.2.11. Che system

4.1.2.11.1. CHE_SYSTEM_SUPER__PRIVILEGED__MODE

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 administrator privileges to those workspaces.

Default

false

4.1.2.11.2. CHE_SYSTEM_ADMIN__NAME

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.

Default

admin

4.1.2.12. Workspace limits

4.1.2.12.1. CHE_LIMITS_WORKSPACE_ENV_RAM

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.

Default

16gb

4.1.2.12.2. CHE_LIMITS_WORKSPACE_IDLE_TIMEOUT

The length of time in milliseconds 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 the agents has not received interaction. Leaving a browser window open counts toward idleness.

Default

1800000

4.1.2.12.3. CHE_LIMITS_WORKSPACE_RUN_TIMEOUT

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.

Default

0

4.1.2.13. Users workspace limits

4.1.2.13.1. CHE_LIMITS_USER_WORKSPACES_RAM

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.

Default

-1

4.1.2.13.2. CHE_LIMITS_USER_WORKSPACES_COUNT

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.

Default

-1

4.1.2.13.3. CHE_LIMITS_USER_WORKSPACES_RUN_COUNT

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.

Default

1

4.1.2.14. Organizations workspace limits

4.1.2.14.1. CHE_LIMITS_ORGANIZATION_WORKSPACES_RAM

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.

Default

-1

4.1.2.14.2. CHE_LIMITS_ORGANIZATION_WORKSPACES_COUNT

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.

Default

-1

4.1.2.14.3. CHE_LIMITS_ORGANIZATION_WORKSPACES_RUN_COUNT

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.

Default

-1

4.1.2.15. Multi-user-specific OpenShift infrastructure configuration

4.1.2.15.1. CHE_INFRA_OPENSHIFT_OAUTH__IDENTITY__PROVIDER

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. See: OpenShift identity provider

Default

NULL

4.1.2.16. OIDC configuration

4.1.2.16.1. CHE_OIDC_AUTH__SERVER__URL

Url to OIDC identity provider server Can be set to NULL only if **che.oidc.oidcProvider** is used

Default

http://${CHE_HOST}:5050/auth

4.1.2.16.2. CHE_OIDC_AUTH__INTERNAL__SERVER__URL

Internal network service Url to OIDC identity provider server

Default

NULL

4.1.2.16.3. CHE_OIDC_ALLOWED__CLOCK__SKEW__SEC

The number of seconds to tolerate for clock skew when verifying **exp** or **nbf** claims.

Default

3

4.1.2.16.4. CHE_OIDC_USERNAME__CLAIM

Username claim to be used as user display name when parsing JWT token if not defined the fallback value is 'preferred_username' in Keycloak installations and **name** in Dex installations.

Default

NULL

4.1.2.16.5. CHE_OIDC_OIDC__PROVIDER

Base URL of an alternate OIDC provider that provides a discovery endpoint as detailed in the following specification Obtaining OpenID Provider Configuration Information Deprecated, use **che.oidc.auth_server_url** and **che.oidc.auth_internal_server_url** instead.

Default

NULL

4.1.2.17. Keycloak configuration

4.1.2.17.1. CHE_KEYCLOAK_REALM

Keycloak realm is used to authenticate users Can be set to NULL only if **che.keycloak.oidcProvider** is used

Default

che

4.1.2.17.2. CHE_KEYCLOAK_CLIENT__ID

Keycloak client identifier in **che.keycloak.realm** to authenticate users in the dashboard, the IDE, and the CLI.

Default

che-public

4.1.2.17.3. CHE_KEYCLOAK_OSO_ENDPOINT

URL to access OSO OAuth tokens

Default

NULL

4.1.2.17.4. CHE_KEYCLOAK_GITHUB_ENDPOINT

URL to access Github OAuth tokens

Default

NULL

4.1.2.17.5. CHE_KEYCLOAK_USE__NONCE

Use the OIDC optional **nonce** feature to increase security.

Default

true

4.1.2.17.6. CHE_KEYCLOAK_JS__ADAPTER__URL

URL to the Keycloak Javascript adapter to use. if set to NULL, then the default used value is **${che.keycloak.auth_server_url}/js/keycloak.js**, or **<che-server>/api/keycloak/OIDCKeycloak.js** if an alternate **oidc_provider** is used

Default

NULL

4.1.2.17.7. CHE_KEYCLOAK_USE__FIXED__REDIRECT__URLS

Set to true when using an alternate OIDC provider that only supports fixed redirect Urls This property is ignored when **che.keycloak.oidc_provider** is NULL

Default

false

4.1.2.17.8. CHE_OAUTH_SERVICE__MODE

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.

Default

delegated

4.1.2.17.9. CHE_KEYCLOAK_CASCADE__USER__REMOVAL__ENABLED

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 administrator username ${che.keycloak.admin_username} and password ${che.keycloak.admin_password}.

Default

false

4.1.2.17.10. CHE_KEYCLOAK_ADMIN__USERNAME

Keycloak administrator username. Will be used for deleting user from Keycloak on removing user from CodeReady Workspaces database. Make sense only in case ${che.keycloak.cascade_user_removal_enabled} set to 'true'

Default

NULL

4.1.2.17.11. CHE_KEYCLOAK_ADMIN__PASSWORD

Keycloak administrator password. Will be used for deleting user from Keycloak on removing user from CodeReady Workspaces database. Make sense only in case ${che.keycloak.cascade_user_removal_enabled} set to 'true'

Default

NULL

4.1.2.17.12. CHE_KEYCLOAK_USERNAME_REPLACEMENT__PATTERNS

User name adjustment configuration. CodeReady Workspaces needs to use the usernames as part of Kubernetes object names and labels and therefore has stricter requirements on their format than the identity providers usually allow (it needs them to be DNS-compliant). The adjustment is represented by comma-separated key-value pairs. These are sequentially used as arguments to the String.replaceAll function on the original username. The keys are regular expressions, values are replacement strings that replace the characters in the username that match the regular expression. The modified username will only be stored in the CodeReady Workspaces database and will not be advertised back to the identity provider. It is recommended to use DNS-compliant characters as replacement strings (values in the key-value pairs). Example: **=-,@=-at-** changes \ to **-** and @ to **-at-** so the username **org\user@com** becomes **org-user-at-com.**

Default

NULL

Additional resources

- Configuring Che to use an external Keycloak installation

4.2. CONFIGURING WORKSPACE TARGET PROJECT

The OpenShift project where a new workspace is deployed depends on the CodeReady Workspaces server configuration. CodeReady Workspaces deploys each workspace into a user's dedicated project, which hosts all CodeReady Workspaces workspaces created by the user. The name of a OpenShift project must be provided as a CodeReady Workspaces server configuration property or pre-created by CodeReady Workspaces administrator.

OpenShift project strategies are configured using **server.workspaceNamespaceDefault** property.

Operator CheCluster CR patch

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: <che-cluster-name>
spec:
  server:
    workspaceNamespaceDefault: <workspace-namespace> 1
```

1 - CodeReady Workspaces workspace project configuration

NOTE

The underlying environment variable that CodeReady Workspaces server uses is **CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT**.

WARNING

By default, only one workspace in the same project can be running at one time. See Section 4.5, "Configuring the number of workspaces that a user can run" .

WARNING

Kubernetes limits the length of a project name to 63 characters (this includes the evaluated placeholders). Additionally, the names (after placeholder evaluation) must be valid DNS names.

On OpenShift with multihost server exposure strategy, the length is further limited to 49 characters.

Be aware that the ***<userid>*** placeholder is evaluated into a 36 character long UUID string.

WARNING

Use Section 4.2.3, "Pre-creating a project for each user" when:

- **che** ServiceAccount does not have enough permissions when creating new project
- OpenShift OAuth with a **self-provisioner** cluster role is not linked to the **system:authenticated:oauth** group
- CodeReady Workspaces cannot create namespaces

4.2.1. One project per user strategy

The strategy isolates each user in their own project.

To use the strategy, set the *CodeReady Workspaces workspace project configuration* value to contain one or more user identifiers. Currently supported identifiers are ***<username>*** and ***<userid>***.

Example 4.2. One project per user

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

```
...
spec:
  server:
    workspaceNamespaceDefault: codeready-ws-<username>
...
```

4.2.2. 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 usernames or IDs are reduced to the nearest valid name by replacing groups of unsuitable symbols with the **-** symbol. The addition of a random 6-symbol suffix prevents IDs from collisions. The result is stored in preferences for reuse.

4.2.3. Pre-creating a project for each user

To pre-create a project for each user, use OpenShift labels and annotations. Such project is used in preference to **CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** variable.

```
metadata:
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: workspaces-namespace
  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 CodeReady Workspaces server component system properties reference for more details.

WARNING

Do not create multiple namespaces for a single user. It may lead to undefined behavior.

IMPORTANT

On OpenShift with OAuth, the target user must have **admin** role privileges in the 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** and **get namespaces** permissions as well as an **admin** role in target namespace.

4.2.4. Labeling the namespaces

CodeReady Workspaces updates the workspace's project on workspace startup by adding the labels defined in **CHE_INFRA_KUBERNETES_NAMESPACE_LABELS**. To do so, **che** ServiceAccount has to have the following cluster-wide permissions to **update** and **get namespaces**:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <cluster-role-name> 1
rules:
  - apiGroups:
      - ""
    resources:
      - namespaces
    verbs:
      - update
      - get
```

1 name of the cluster role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: <cluster-role-binding-name> 1
subjects:
  - kind: ServiceAccount
    name: <service-account-name> 2
```

```
    namespace: <service-accout-namespace> 3
roleRef:
  kind: ClusterRole
  name: <cluster-role-name> 4
  apiGroup: rbac.authorization.k8s.io
```

1. name of the cluster role binding
2. name of the **che** ServiceAccount
3. CodeReady Workspaces installation namespace
4. name of the cluster role created in previous step

NOTE

A lack of permissions does not prevent a CodeReady Workspaces workspace from starting, it only logs the warning. If you see the warnings in CodeReady Workspaces logs, consider disabling the feature by defining **CHE_INFRA_KUBERNETES_NAMESPACE_LABEL=false**.

4.3. CONFIGURING STORAGE STRATEGIES

This section describes how to configure storage strategies for CodeReady Workspaces workspaces.

4.3.1. Storage strategies for codeready-workspaces workspaces

Storage strategy

A configurable method defining how CodeReady Workspaces workspaces use persistent volume claims (PVCs) and persistent volumes (PVs). This method defines the storage for workspace data, for example, projects, workspace logs, or additional volumes defined by a user.

Table 4.1. Storage strategies comparison

Storage strategy name	common	per-workspace	unique
PV count	One per user	One per workspace	Multiple PVs per workspace
Default	yes	no	no
Limitations	Maximum one running workspace per user when the PV is in the **ReadWriteOnce** access mode	PV count depends on the number of workspaces	Unpredictable PV count

Persistent volumes (PVs) access mode

The nature of the PV determines the available access mode: **ReadWriteMany** or **ReadWriteOnce**. See Kubernetes documentation - access mode . For example, Amazon EBS supports only the **ReadWriteOnce** access mode.

4.3.1.1. The common storage strategy

This is the default storage strategy. For each user, all workspaces use the same PV for the default data storage.

When the user starts a first non-ephemeral workspace, the workspace engine creates a common PV.

When the user starts another non-ephemeral workspace, the workspace engine uses the same common PV. The workspace only binds simultaneously to one node in the OpenShift cluster. The workspace engine ignores user-defined volumes. The workspace engine replaces volumes related to user-defined volumes with a subPath in the common volume. SubPaths have a ***<workspace-ID>*** or ***<original-volume-name>*** prefix. See Section 4.3.1.4, "How subPaths are used in persistent volumes (PVs)" . The CodeReady Workspaces volume name is identical to the name of the user-defined PV. Therefore, if a workspace container uses a CodeReady Workspaces volume with the same name as the user-defined PV, they will use the same shared folder in the common PV.

When the user deletes a workspace, the workspace engine deletes the corresponding subdirectory (**${ws-id}**) in the PV directory.

When the user deletes the last workspace, the workspace engine removes the common PV.

Restrictions on the common storage strategy with the ReadWriteOnce access mode

The **ReadWriteOnce** access mode limits each user to run only one concurrent workspace. See Section 4.5, "Configuring the number of workspaces that a user can run" .

Scalability

The **common** storage strategy is not suitable for multi-node clusters with the 'ReadWriteOnce' access mode for PVs if the number of concurrently running workspaces per user is more than 1.

Persistent volumes (PV) provisioning

Create a large enough PV to accommodate all projects to prevent a situation in which one project depletes the resources of others.

4.3.1.2. The per-workspace storage strategy

Each workspace uses one dedicated PV. All CodeReady Workspaces volumes defined within a single workspace use the same PV.

Persistent volumes (PV) provisioning

Users can run multiple workspaces simultaneously. This action results in more PVs.

4.3.1.3. The unique storage strategy

Each CodeReady Workspaces volume defined in a workspace has its own PV.

When the user starts a workspace, the workspace engine creates the workspace PVs.

The workspace engine generates a unique name for each PV to prevent name conflicts with other PVs in the same project.

To ensure that different storage strategies use the same PV data structure, subPaths of the mounted PVs that reference user-defined PVs are prefixed with ***<workspace-ID>*** or ***<persistent-volume-name>***. See Section 4.3.1.4, "How subPaths are used in persistent volumes (PVs)" .

When the user deletes a workspace, the workspace engine deletes all workspace PVs.

PV provisioning

This is the strategy, which creates the highest volume counts.

4.3.1.4. How subPaths are used in persistent volumes (PVs)

SubPaths illustrate the folder hierarchy in the PV.

```
/<pv0001>
  /<workspaceID1>
  /<workspaceID2>
  /<workspaceIDn>
    /che-logs
    /projects
    /<volume1>
    /<volume2>
    /<user-defined-volume-name-1>
    /<user-defined-volume-name-2>
    /<user-defined-volume-name-3|volume3>
    ...
```

When a user defines volumes for components in the devfile, all components that define the volume of the same name will be backed by the same directory in the PV as ***<persistent-volume-name>***, ***<workspace-ID>***, or ***<original-volume-name>***. Each component can have this location mounted on a different path in its containers.

4.3.2. Configuring a CodeReady Workspaces workspace with a persistent volume strategy

A persistent volume (PV) acts as a virtual storage instance that adds a volume to a cluster.

A persistent volume claim (PVC) is a request to provision persistent storage of a specific type and configuration, available in the following CodeReady Workspaces storage configuration strategies:

- Common
- Per-workspace
- Unique

The mounted PVC is displayed as a folder in a container file system.

4.3.2.1. Configuring a PVC strategy using the Operator

The following section describes how to configure workspace persistent volume claim (PVC) strategies of a CodeReady Workspaces server using the Operator.

WARNING

It is not recommended to reconfigure PVC strategies on an existing CodeReady Workspaces cluster with existing workspaces. Doing so causes data loss.

Operators are software extensions to OpenShift that use Custom Resources to manage applications and their components.

When deploying CodeReady Workspaces using the Operator, configure the intended strategy by modifying the **spec.storage.pvcStrategy** property of the CheCluster Custom Resource object YAML file.

Prerequisites

- The **oc** tool is available.

Procedure

The following procedure steps are available for OpenShift command-line tool, '`oc'.

To do changes to the CheCluster YAML file, choose one of the following:

- Create a new cluster by executing the **oc apply** command. For example:

```
$ oc apply -f <my-cluster.yaml>
```

- Update the YAML file properties of an already running cluster by executing the **oc patch** command. For example:

```
$ oc patch checluster/codeready-workspaces --type=json \
  -p '[{"op": "replace", "path": "/spec/storage/pvcStrategy", "value": "per-workspace"}]'
```

Depending on the strategy used, replace the **per-workspace** option in the above example with **unique** or **common**.

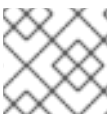
4.4. CONFIGURING STORAGE TYPES

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

- Persistent
- Ephemeral
- Asynchronous

4.4.1. 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'
```

4.4.2. 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 4.2. Comparison between I/O of ephemeral (emptyDir) and persistent modes onAWS 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

4.4.3. Asynchronous storage

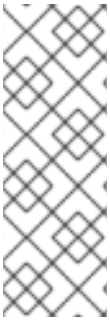
NOTE

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'
```

4.4.4. 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. The **async** value is not recommended as the default type because it is experimental. For example:

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

Then users are able to configure Storage Type on the **Create Custom Workspace** tab on CodeReady Workspaces dashboard during workspace creation. Storage type for existing workspace can be configured in on **Overview** tab of the workspace details.

4.4.5. 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.

To increase the timeout of a CodeReady Workspaces workspace, use the following example, which sets the workspace timeout for 1800000 milliseconds that correspond to the interval of 30 minutes.

+

```
$ oc patch checluster/codeready-workspaces --patch "{\"spec\":{\"server\":{\"customCheProperties\":{\"CHE_LIMITS_WORKSPACE_IDLE_TIMEOUT\": \"1800000\"}}}}" --type=merge -n openshift-workspaces
```

4.5. CONFIGURING THE NUMBER OF WORKSPACES THAT A USER CAN RUN

This procedure describes how to configure CodeReady Workspaces to run more than one workspace simultaneously. By running multiple workspaces, users can use different work environments simultaneously.

Prerequisites

- You have installed an instance of CodeReady Workspaces by using the Operator.
- The combination of storage strategy and access mode matches one of the following cases:

  Table 4.3. Multiple workspaces compatibility list

Storage strategy	**common** (default)	**per-workspace**	**unique**
ReadWriteMany access mode	yes	yes	yes
ReadWriteOnce access mode	no	yes	yes

  - To determine the access modes supported by your storage, see Kubernetes documentation - Access Modes. For example, Amazon EBS only supports the **ReadWriteOnce** access mode.
  - See Section 4.3, “Configuring storage strategies”.
- You have determined the value of the ***<number-of-workspaces>*** placeholder.

NOTE

If the value is **-1**, an unlimited number of workspaces can run per user. If the value is a positive integer, users can run as many workspaces as the value of the integer. The default value is **1**.

Procedure

- In the **CheCluster** Custom Resource **server** settings, configure the number of workspaces that a user can run by adding the **CHE_LIMITS_USER_WORKSPACES_RUN_COUNT** property to **customCheProperties**:

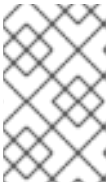
```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# ...
spec:
  server:
    # ...
    customCheProperties:
      CHE_LIMITS_USER_WORKSPACES_RUN_COUNT: "<number-of-workspaces>"
```

4.6. CONFIGURING THE NUMBER OF WORKSPACES THAT A USER CAN CREATE

This procedure describes how to configure the number of workspaces that a user can create. By creating multiple workspaces, users can have access to workspaces with different configurations simultaneously.

Prerequisites

- You have installed an instance of **CodeReady Workspaces** by using the Operator.
- You have determined the value of the ***<number-of-workspaces>*** placeholder.



NOTE

If the value is **-1**, users can create an unlimited number of workspaces. If the value is a positive integer, users can create as many workspaces as the value of the integer. The default value is **-1**.

Procedure

- In the **CheCluster** Custom Resource **server** settings, configure the number of workspaces that a user can create by adding the **CHE_LIMITS_USER_WORKSPACES_COUNT** property to **customCheProperties**:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# ...
spec:
  server:
    # ...
    customCheProperties:
      CHE_LIMITS_USER_WORKSPACES_COUNT: "<number-of-workspaces>"
```

-

4.7. 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.

4.7.1. Configuring workspace exposure strategies using an Operator

Operators are software extensions to OpenShift that use Custom Resources to manage applications and their components.

Prerequisites

- The **oc** tool is available.

Procedure

When deploying CodeReady Workspaces using the Operator, configure the intended strategy by modifying the **spec.server.serverExposureStrategy** property of the CheCluster Custom Resource object YAML file.

The supported values for **spec.server.serverExposureStrategy** are:

- **multi-host**
- **single-host**

See Section 4.7.2, "Workspace exposure strategies" for more detail about individual strategies.

To activate changes done to CheCluster YAML file, do one of the following:

- Create a new cluster by executing the **crwctl** command with applying a patch. For example:

```
$ crwctl server:deploy  --installer=operator --platform=<platform> \
  --che-operator-cr-patch-yaml=patch.yaml
```

NOTE

For a list of available OpenShift deployment platforms, use **crwctl server:deploy --platform --help**.

  - Use the following **patch.yaml** file:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: eclipse-che
spec:
  server:
    serverExposureStrategy: '<exposure-strategy>' 1
```

1 - used workspace exposure strategy

- Update the YAML file properties of an already running cluster by executing the **oc patch** command. For example:

```
$ oc patch checluster/codeready-workspaces --type=json \
  -p '[{"op": "replace",
  "path": "/spec/server/serverExposureStrategy",
  "value": "<exposure-strategy>"}]' \ 1
  -n openshift-workspaces
```

1 - used workspace exposure strategy

4.7.2. 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

4.7.2.1. Multihost strategy

With multihost strategy, each workspace component is assigned a new subdomain of the main domain configured for the CodeReady Workspaces server. This is the default strategy.

This strategy is the easiest to understand from the perspective of component deployment because any paths to the component in the URL 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.

4.7.2.2. Single-host strategy

With single-host strategy, 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.

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.

WARNING

With **gateway** single-host strategy, cluster network policies has to be configured so that workspace's services are reachable from reverse-proxy Pod (typically in CodeReady Workspaces project). These typically lives in different project.

To define how to expose the endpoints specified in the devfile, define the **CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE__ENDPOINT__EXPOSURE** environment variable in the CodeReady Workspaces instance. This environment variable is only effective with the single-host server strategy and is applicable to all workspaces of all users.

4.7.2.2.1. 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.

4.7.2.2.2. 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.

4.7.3. Security considerations

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

4.7.3.1. 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

4.7.3.2. 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 indiscernible to the user and is governed by the plug-in or editor definition in their **meta.yaml** descriptors.

4.7.3.3. 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 4.7.3.1, “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 4.7.3.1, “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**.

4.7.3.4. 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.

4.7.3.5. 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.

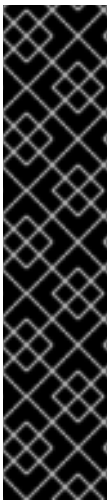
4.8. CONFIGURING WORKSPACES NODESELECTOR

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, multizone clusters, create an additional **StorageClass** object (pay attention to the **allowedTopologies** field), which will coordinate the PVC creation process.

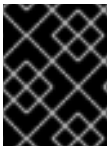
Pass the name of this newly created **StorageClass** to CodeReady Workspaces through the **CHE_INFRA_KUBERNETES_PVC_STORAGE__CLASS__NAME** environment variable. A default empty value of this variable instructs CodeReady Workspaces to use the cluster's default **StorageClass**.

4.9. CONFIGURING RED HAT CODEREADY WORKSPACES SERVER HOSTNAME

This procedure describes how to configure 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 a private key and certificate, the same certification authority (CA) must be used as for other CodeReady Workspaces hosts.

IMPORTANT

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

Procedure

1. Pre-create a project for CodeReady Workspaces:

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

2. Create a TLS secret:

```
$ oc create secret TLS <tls-secret-name> \ 1
--key <key-file> \ 2
--cert <cert-file> \ 3
-n openshift-workspaces
```

1. The TLS secret name
2. A file with the private key
3. A file with the certificate

3. Add the required labels to the secret:

```
$ oc label secret <tls-secret-name> \ 1
app.kubernetes.io/part-of=che.eclipse.org -n openshift-workspaces
```

1. The TLS secret name

4. 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

5. If CodeReady Workspaces has been already deployed, wait until the rollout of all CodeReady Workspaces components finishes.

4.10. CONFIGURING OPENSHIFT ROUTE

By configuring labels and annotations for OpenShift Route you can organize and categorize objects by scoping and selecting.

Prerequisites

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

Procedure

1. To configure labels for OpenShift Route, update the Custom Resource:

IMPORTANT

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

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

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

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

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

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

2. To configure annotations for OpenShift Route, update the Custom Resource with the following commands:

IMPORTANT

Use objects to specify annotations: **{"key1": "value1", "key2" : "value2"}**.

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

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

$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p \
'[{"op": "replace", "path": "/spec/server/pluginRegistryIngress/annotations", '\
'"value": <annotations for a plug-ins registry ingress>}]'

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

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

$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p \
'[{"op": "replace", "path":
"/spec/server/customCheProperties/CHE_INFRA_KUBERNETES_INGRESS_ANNOTATIONS
__JSON", '\
'"value": "<annotations for a workspace ingress in json format>"}]'
```

4.11. CONFIGURING OPENSHIFT ROUTE TO WORK WITH ROUTER SHARDING

This procedure describes how to configure labels, annotations, and domains for OpenShift Route to work with Router Sharding. The chapter then mentions the configuration process for existing instances or those about to be installed.

Prerequisites

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

Procedure

- For a new OperatorHub installation:

  1. Enter the CodeReady Workspaces Cluster using OpenShift Container Platform and create CheCluster Custom Resource (CR). See, Creating an instance of the Red Hat CodeReady Workspaces Operator

  2. Set the following values in codeready-workspaces Custom Resource (CR):

```
spec:
  server:
    devfileRegistryRoute:
      labels: <labels> 1
      domain: <domain> 2
      annotations:  3
        key1: value1
        key2: value2
    pluginRegistryRoute:
      labels: <labels> 4
      domain: <domain> 5
      annotations:  6
        key1: value1
        key2: value2
    dashboardRoute:
      labels: <labels> 7
      domain: <domain> 8
      annotations:  9
        key1: value1
        key2: value2
    cheServerRoute:
      labels: <labels> 10
```

```
    domain: <domain> 11
    annotations:  12
      key1: value1
      key2: value2
  customCheProperties:
    CHE_INFRA_OPENSHIFT_ROUTE_LABELS: <labels> 13
    CHE_INFRA_OPENSHIFT_ROUTE_HOST_DOMAIN__SUFFIX: <domain> 14
 auth:
  identityProviderRoute:
    labels: <labels> 15
    domain: <domain> 16
    annotations:  17
      key1: value1
      key2: value2
```

1 4 7 10 13 15 comma separated list of labels that are used by the target ingress controller to filter the set of Routes to service

2 5 8 11 14 16 DNS name serviced by the target ingress controller

3 6 9 12 17 unstructured key value map stored with a resource

- For a new **crwctl** installation:

  1. Configure the installation using:

     ```
     $ crwctl server:deploy --che-operator-cr-patch-yaml=patch.yaml ...
     ```

     The **patch.yaml** file must contain the following:

     ```
     spec:
       server:
         devfileRegistryRoute:
           labels: <labels> 1
           domain: <domain> 2
           annotations:  3
             key1: value1
             key2: value2
         pluginRegistryRoute:
           labels: <labels> 4
           domain: <domain> 5
           annotations:  6
             key1: value1
             key2: value2
         dashboardRoute:
           labels: <labels> 7
           domain: <domain> 8
           annotations:  9
             key1: value1
             key2: value2
         cheServerRoute:
           labels: <labels> 10
     ```

```
    domain: <domain> 11
    annotations:  12
      key1: value1
      key2: value2
  customCheProperties:
    CHE_INFRA_OPENSHIFT_ROUTE_LABELS: <labels> 13
    CHE_INFRA_OPENSHIFT_ROUTE_HOST_DOMAIN__SUFFIX: <domain> 14
auth:
  identityProviderRoute:
    labels: <labels> 15
    domain: <domain> 16
    annotations:  17
      key1: value1
      key2: value2
```

1 4 7 10 13 15 comma separated list of labels that are used by the target ingress controller to filter the set of Routes to service

2 5 8 11 14 16 DNS name serviced by the target ingress controller

3 6 9 12 17 unstructured key value map stored with a resource

- For already existing CodeReady Workspaces installation:

  1. Update **codeready-workspaces** CR using the **oc** tool:

     a. To configure labels:

IMPORTANT

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

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

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

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

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

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path":
"/spec/server/customCheProperties/CHE_INFRA_OPENSHIFT_ROUTE_LABELS",
'\
'"value": "<labels for a workspace routes>"}]'
```

b. To configure domains:

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path": "/spec/server/cheServerRoute/domain",'\
'"value": "<ingress domain>"}]'
```

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path": "/spec/server/pluginRegistryRoute/domain", '\
'"value": "<ingress domain>"}]'
```

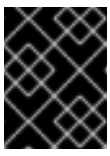
```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path": "/spec/server/devfileRegistryRoute/domain", '\
'"value": "<ingress domain>"}]'
```

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path": "/spec/server/dashboardRoute/domain", '\
'"value": "<ingress domain>"}]'
```

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path": "/spec/auth/identityProviderRoute/domain", '\
'"value": "<ingress domain>"}]'
```

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path":
"/spec/server/customCheProperties/CHE_INFRA_OPENSHIFT_ROUTE_HOST_DO
MAIN__SUFFIX", '\
'"value": "<ingress domain>"}]'
```

c. To configure annotations:



IMPORTANT

Use object to specify annotations: **{"key1": "value1", "key2" : "value2"}**

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

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
\
'[{"op": "replace", "path": "/spec/server/pluginRegistryRoute/annotations", '\
'"value": <annotations for a plug-ins registry ingress>}]'
```

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

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

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

4.12. 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=<path_to_certificate> \ 1
  --from-literal=githost=<host:port> -n openshift-workspaces 2
```

1. Path to self-signed certificate
2. 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.
- 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. Add the required labels to the ConfigMap:

```
$ oc label configmap che-git-self-signed-cert \
app.kubernetes.io/part-of=che.eclipse.org -n openshift-workspaces
```

3. Configure CodeReady Workspaces to use self-signed certificates for Git repositories: Update the **gitSelfSignedCert** property. To do that, execute:

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

4. 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
```

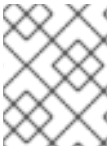
4.13. INSTALLING CODEREADY WORKSPACES USING STORAGE CLASSES

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

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

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

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



NOTE

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

Persistent volume claims facts:

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

Procedure

Use CheCluster Custom Resource definition to define storage classes:

1. Define storage class names
   To do so, use one of the following methods:
   - **Use arguments for the server: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 -p minikube -a operator --postgres-pvc-storage-class-name=postgress-storage
     ```

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

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

     For CodeReady Workspaces workspaces, the storage class name has different behavior depending on the workspace PVC strategy.
   - **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 -p minikube -a operator --che-operator-cr-
yaml=/path/to/custom/che/resource/org_v1_che_cr.yaml
```

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

   a. Modify your Custom Resources YAML file:

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

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: codeready-workspaces
spec:
  server:
    # ...
    workspaceNamespaceDefault: codeready-ws-<username>
    # ...
  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 -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**.

4.14. IMPORTING UNTRUSTED TLS CERTIFICATES TO CODEREADY WORKSPACES

External 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, and "RH-SSO Identity Provider" may require using TLS. Those communications require the use of TLS certificates signed by trusted Certificate Authorities.

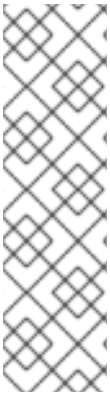
NOTE

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 that 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.

CodeReady Workspaces uses labeled ConfigMaps in project as sources for TLS certificates. The ConfigMaps can have an arbitrary number of keys with a random number of certificates each.



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

IMPORTANT

Some CodeReady Workspaces components require to have a full certificate chain to trust the endpoint. If the cluster is configured with an intermediate certificate, then the whole chain (including self-signed root) should be added to CodeReady Workspaces.

4.14.1. Adding new CA certificates into CodeReady Workspaces

The following procedure is applicable for already installed and running instances and for instances that are to be installed.

NOTE

If you are using CodeReady Workspaces version lower than 2.5.1 see this guide on how to apply additional TLS certificates.

Prerequisites

- The **oc** tool is available.
- Namespace for CodeReady Workspaces exists.

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 custom-certs --from-file=<bundle-file-path> -n=openshift-workspaces
   ```

   Add another **-from-file=*<bundle-file-path>*** flag to apply more than one bundle. Otherwise, create another ConfigMap.

3. Label created ConfigMaps with both **app.kubernetes.io/part-of=che.eclipse.org** and **app.kubernetes.io/component=ca-bundle** labels:

   ```
   $ oc label configmap custom-certs app.kubernetes.io/part-of=che.eclipse.org
   app.kubernetes.io/component=ca-bundle -n <crw-namespace-name>
   ```

4. Deploy CodeReady Workspaces if it has not been deployed before. Otherwise wait until the rollout of CodeReady Workspaces components finishes. If there are running workspaces, they should be restarted for the changes to take effect.

4.14.2. Verification at the CodeReady Workspaces installation level

When something does not work as expected after adding the certificates, here is a list of things to verify:

- In case of a CodeReady Workspaces Operator deployment, the namespace where the **CheCluster** is located contains labeled ConfigMaps with the right content:

```
$ oc get cm --selector=app.kubernetes.io/component=ca-bundle,app.kubernetes.io/part-of=che.eclipse.org -n openshift-workspaces
```

Check the content of ConfigMap by running:

```
$ {orch-cli} get cm __<name>__ -n {prod-namespace} -o yaml
```

- CodeReady Workspaces Pod Volumes list contains a volume that uses **ca-certs-merged** 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 openshift-workspaces | 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 openshift-workspaces -- 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 openshift-workspaces
```

- 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 openshift-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
```

4.14.3. Verification at the workspace level

- Start a workspace, obtain the project name in which it has been created, and wait for the workspace 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 Che-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** ConfigMap contain all the additional entries you added before. In addition, it can contain **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
source-config-map-name.data-key.crt
```

- Confirm 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 is mounted into containers, especially in the Che-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 the /**public-certs** folder in the Che-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
source-config-map-name.data-key.crt
```

4.15. CONFIGURING COMMUNICATION BETWEEN CODEREADY WORKSPACES COMPONENTS

You can select whether CodeReady Workspaces components communicate by using the internal network or external OpenShift Route.

By default, CodeReady Workspaces components communicate by using the internal network. CodeReady Workspaces components use their internal services names, which are exposed in the internal OpenShift network.

As the administrator, disable the use of the internal services names to force the CodeReady Workspaces components to use external OpenShift Route in the following situations:

- To deploy CodeReady Workspaces on a cluster where NetworkPolicies restricts communications between namespaces.
- To deploy CodeReady Workspaces with the multitenant network plug-in.

IMPORTANT

Using the external OpenShift Route might slow the traffic and lead to issues because it uses proxies, certificates, and firewalls.

Prerequisites

- An instance of CodeReady Workspaces running on OpenShift.

Procedure

- In the CheCluster Custom Resource server settings, for the **disableInternalClusterSVCNames** property, set ***<property-value>*** to:

true	To use external OpenShift Route.
false	To use internal OpenShift DNS names.

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# ...
spec:
  server:
    # ...
    disableInternalClusterSVCNames: <property-value>
```

Verification steps

1. Specify CodeReady Workspaces as the default project:

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

2. Inspect the ConfigMap properties to determine which communication method CodeReady Workspaces uses:

```
$ oc get configmap che -o \
jsonpath='{.data.CHE_KEYCLOAK_AUTH__INTERNAL__SERVER__URL}'
$ oc get configmap che -o \
jsonpath='{.data.CHE_WORKSPACE_PLUGIN__REGISTRY__INTERNAL__URL}'
```

   - If CodeReady Workspaces components communicate internally, the output is following:

```
http://keycloak.eclipse-che.svc:8080/auth
http://plugin-registry.eclipse-che.svc:8080/v3
```

- Otherwise, if the components communicate externally, the output is empty.

4.16. 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, log in to RH-SSO and modify the theme settings.

4.16.1. Logging in to RH-SSO

The following procedure describes how to log in to RH-SSO, which acts as a route for OpenShift platforms. To log in to RH-SSO, a user has to obtain the RH-SSO URL and a user's credentials first.

Prerequisites

- The **oc** tool installed.
- Logged in to OpenShift cluster using the **oc** tool.

Procedure

1. Obtain a user RH-SSO login:

```
oc get secret che-identity-secret  -n openshift-workspaces -o json | jq -r '.data.user' | base64 -d
```

2. Obtain a user RH-SSO password:

```
oc get secret che-identity-secret  -n openshift-workspaces -o json | jq -r '.data.password' | base64 -d
```

3. Obtain the RH-SSO URL:

```
oc  get ingress -n openshift-workspaces -l app=che,component=keycloak   -o 'custom-columns=URL:.spec.rules[0].host' --no-headers
```

4. Open the URL in a browser and log in to RH-SSO using the obtained login and password.

4.16.2. Setting up the RH-SSO codeready-workspaces-username-readonly theme

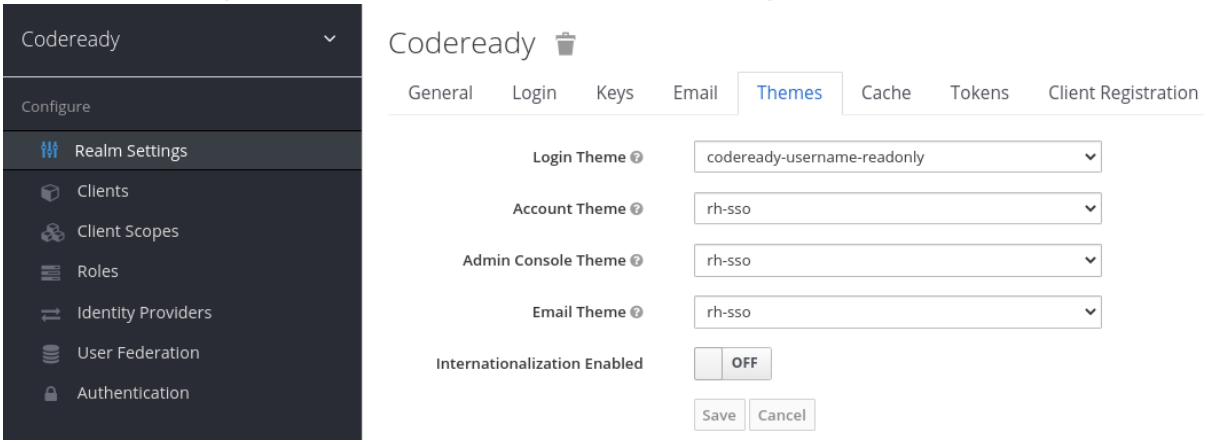
Prerequisites

- 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**:



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.

4.17. MOUNTING A SECRET OR A CONFIGMAP AS A FILE OR AN ENVIRONMENT VARIABLE INTO A CODEREADY WORKSPACES CONTAINER

Secrets are OpenShift objects that store sensitive data such as:

- usernames
- passwords
- authentication tokens

in an encrypted form.

Users can mount a OpenShift Secret that contains sensitive data or a ConfigMap that contains configuration in a CodeReady Workspaces managed containers as:

- a file
- an environment variable

The mounting process uses the standard OpenShift mounting mechanism, but it requires additional annotations and labeling.

4.17.1. Mounting a Secret or a ConfigMap as a file into a CodeReady Workspaces container

Prerequisites

- A running instance of Red Hat CodeReady Workspaces. To install an instance of Red Hat CodeReady Workspaces, see Installing CodeReady Workspaces.

Procedure

1. Create a new OpenShift Secret or a ConfigMap in the OpenShift project where a CodeReady Workspaces is deployed. The labels of the object that is about to be created must match the set of labels:
   - **app.kubernetes.io/part-of: che.eclipse.org**
   - **app.kubernetes.io/component: <DEPLOYMENT_NAME>-<OBJECT_KIND>**
   - The **<DEPLOYMENT_NAME>** corresponds to the one following deployments:
     - **postgres**
     - **keycloak**
     - **devfile-registry**
     - **plugin-registry**
     - **codeready**
       and
   - **<OBJECT_KIND>** is either:
     - **secret**
       or
     - **configmap**

Example 4.3. Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-settings
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: codeready-secret
...
```

or

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-settings
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: codeready-configmap
...
```

Annotations must indicate that the given object is mounted as a file.

1. Configure the annotation values:
   - **che.eclipse.org/mount-as: file** - To indicate that a object is mounted as a file.
   - **che.eclipse.org/mount-path: *<TARGET_PATH>*** - To provide a required mount path.

Example 4.4. Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-data
  annotations:
    che.eclipse.org/mount-as: file
    che.eclipse.org/mount-path: /data
  labels:
...
```

or

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-data
  annotations:
    che.eclipse.org/mount-as: file
    che.eclipse.org/mount-path: /data
  labels:
...
```

The OpenShift object may contain several items whose names must match the desired file name mounted into the container.

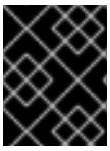
Example 4.5. Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-data
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: codeready-secret
  annotations:
    che.eclipse.org/mount-as: file
    che.eclipse.org/mount-path: /data
data:
  ca.crt: <base64 encoded data content here>
```

or

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-data
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: codeready-configmap
  annotations:
    che.eclipse.org/mount-as: file
    che.eclipse.org/mount-path: /data
data:
  ca.crt: <data content here>
```

This results in a file named **ca.crt** being mounted at the **/data** path of CodeReady Workspaces container.



IMPORTANT

To make the changes in a CodeReady Workspaces container visible, recreate the object entirely.

4.17.2. Mounting a Secret or a ConfigMap as an environment variable into a CodeReady Workspaces container

Prerequisites

- A running instance of Red Hat CodeReady Workspaces. To install an instance of Red Hat CodeReady Workspaces, see Installing CodeReady Workspaces.

Procedure

1. Create a new OpenShift Secret or a ConfigMap in the OpenShift project where a CodeReady Workspaces is deployed. The labels of the object that is about to be created must match the set of labels:
   - **app.kubernetes.io/part-of: che.eclipse.org**
   - **app.kubernetes.io/component: <DEPLOYMENT_NAME>-<OBJECT_KIND>**
   - The **<DEPLOYMENT_NAME>** corresponds to the one following deployments:
     - **postgres**
     - **keycloak**
     - **devfile-registry**
     - **plugin-registry**
     - **codeready**
       and
   - **<OBJECT_KIND>** is either:
     - **secret**

or

- **configmap**

Example 4.6. Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-settings
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: codeready-secret
...
```

or

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-settings
  labels:
    app.kubernetes.io/part-of: che.eclipse.org
    app.kubernetes.io/component: codeready-configmap
...
```

Annotations must indicate that the given object is mounted as a environment variable.

1. Configure the annotation values:
   - **che.eclipse.org/mount-as: env** - to indicate that a object is mounted as an environment variable
   - **che.eclipse.org/env-name: *<FOO_ENV>*** - to provide an environment variable name, which is required to mount a object key value

Example 4.7. Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-settings
  annotations:
    che.eclipse.org/env-name: FOO_ENV
    che.eclipse.org/mount-as: env
  labels:
    ...
data:
  mykey: myvalue
```

or

```
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: custom-settings
  annotations:
    che.eclipse.org/env-name: FOO_ENV
    che.eclipse.org/mount-as: env
  labels:
    ...
data:
  mykey: myvalue
```

This results in two environment variables:

- **FOO_ENV**
- **myvalue**

being provisioned into a CodeReady Workspaces container.

If the object provides more than one data item, the environment variable name must be provided for each of the data keys as follows:

Example 4.8. Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-settings
  annotations:
    che.eclipse.org/mount-as: env
    che.eclipse.org/mykey_env-name: FOO_ENV
    che.eclipse.org/otherkey_env-name: OTHER_ENV
  labels:
    ...
data:
  mykey: __<base64 encoded data content here>__
  otherkey: __<base64 encoded data content here>__
```

or

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-settings
  annotations:
    che.eclipse.org/mount-as: env
    che.eclipse.org/mykey_env-name: FOO_ENV
    che.eclipse.org/otherkey_env-name: OTHER_ENV
  labels:
    ...
data:
  mykey: __<data content here>__
  otherkey: __<data content here>__
```

-

This results in two environment variables:

- **FOO_ENV**
- **OTHER_ENV**

being provisioned into a CodeReady Workspaces container.

NOTE

The maximum length of annotation names in a OpenShift object is 63 characters, where 9 characters are reserved for a prefix that ends with /. This acts as a restriction for the maximum length of the key that can be used for the object.

IMPORTANT

To make the changes in a CodeReady Workspaces container visible, recreate the object entirely.

4.18. ENABLING DEV WORKSPACE OPERATOR

This procedure describes how to enable the Dev Workspace operator to support the Devfile v2 file format and mentions how to do so on existing instances or those about to be installed.

Prerequisites

- The **oc** and **crwctl** tools are available.

Procedure

- For a new OperatorHub installation:
  1. Enter the Red Hat CodeReady Workspaces Cluster using OpenShift Container Platform and create CheCluster Custom Resource (CR). See, Creating an instance of the Red Hat CodeReady Workspaces Operator
  2. Set the following values in codeready-workspaces Custom Resource (CR):

```
spec:
  devWorkspace:
    enable: true
```

- For a new **crwctl** installation:
  1. Configure the the **crwctl** installation using:

```
$ crwctl server:deploy --workspace-engine=dev-workspace ...
```

- For already existing CodeReady Workspaces installation:
  1. Update **codeready-workspaces** CR using the **oc** tool:

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

Additional resources

For information about installation methods mentioned in this chapter, see Chapter 3, *Installing CodeReady Workspaces*.

CHAPTER 5. UPGRADING CODEREADY WORKSPACES

This chapter describes how to upgrade a CodeReady Workspaces instance from version 2.14 to CodeReady Workspaces 2.15.

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”

A CodeReady Workspaces upgrade can be rolled back:

- Section 5.5, “Rolling back a CodeReady Workspaces upgrade”

5.1. UPGRADING CODEREADY WORKSPACES USING OPERATORHUB

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

OperatorHub supports **Automatic** and **Manual** upgrade strategies: **Automatic**:: The upgrade process starts when a new version of the Operator is published. **Manual**:: The update must be manually approved every time the new version of the Operator is published.

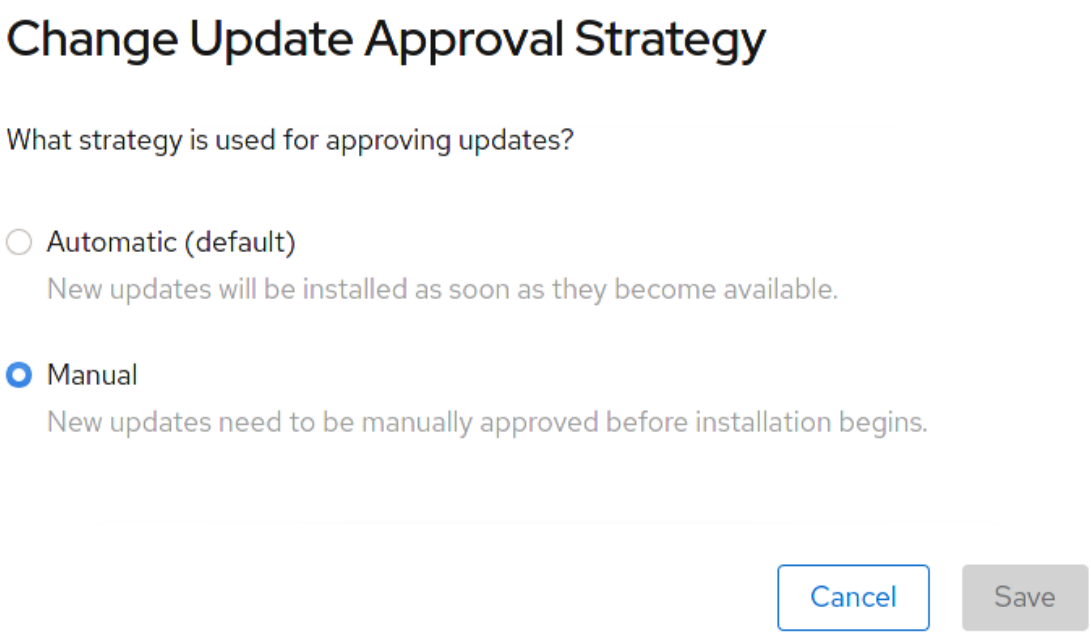
5.1.1. Specifying the approval strategy of CodeReady Workspaces in OperatorHub

Prerequisites

- An administrator account on an instance of OpenShift.
- An instance of CodeReady Workspaces 2.14 or earlier that was installed by using OperatorHub.

Procedure

1. Open the OpenShift web console.
2. Navigate to the **Operators → Installed Operators** page.
3. Click **Red Hat CodeReady Workspaces** in the list of installed Operators.
4. Navigate to the **Subscription** tab.
5. Configure the approval strategy to **Automatic** or **Manual**.



5.1.2. Manually upgrading CodeReady Workspaces in OperatorHub

OperatorHub is an assembly point for sharing Operators. The OperatorHub helps you deploy and update applications. The following section describes the process of upgrading CodeReady Workspaces by using OperatorHub and the **Manual** approval strategy approach. Use the **Manual** approval strategy to prevent automatic updates of the Operator with every release.

Prerequisites

- An administrator account on an instance of OpenShift.
- An instance of CodeReady Workspaces 2.14 or earlier that was installed by using OperatorHub.
- The approval strategy in the subscription is **Manual**.

Procedure

1. Open the OpenShift web console.
2. Navigate to the **Operators → Installed Operators** page.
3. In the list of the installed Operators, click **Red Hat CodeReady Workspaces**
4. Navigate to the **Subscription** tab.
5. Next to the **Upgrade Status**, inspect the upgrades that require approval. The expected message is **1 requires approval**.
6. Click **1 requires approval**.
7. Click **Preview Install Plan**.
8. Review the resources that are available for upgrade and click **Approve**.

Verification steps

1. Navigate to the **Operators → Installed Operators** page.
2. Monitor the upgrade progress. When complete, the status changes to **Succeeded** and **Up to date**. The 2.15 version number is visible at the end of the page.

Additional resources

- Upgrading installed Operators section in the OpenShift documentation.

5.2. UPGRADING CODEREADY WORKSPACES USING THE CLI MANAGEMENT TOOL

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

Prerequisites

- An administrative account on OpenShift.
- A running instance of a previous minor version of Red Hat CodeReady Workspaces, installed using the CLI management tool on the same instance of OpenShift, in the ***<openshift-workspaces>*** project.
- **crwctl** is available and updated. See Section 3.3.1, “Installing the crwctl CLI management tool” .

Procedure

1. Save and push changes back to the Git repositories for all running CodeReady Workspaces 2.14 workspaces.
2. Shut down all workspaces in the CodeReady Workspaces 2.14 instance.
3. Upgrade CodeReady Workspaces:

```
$ crwctl server:update -n openshift-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 2.15 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 2.14 to version 2.15.

Prerequisites

- An administrative account on an instance of OpenShift.
- A running instance version 2.14 of Red Hat CodeReady Workspaces, installed using the CLI management tool on the same instance of OpenShift, with the crwctl **--installer operator** method, in the ***<openshift-workspaces>*** project. See Section 3.4, "Installing CodeReady Workspaces in a restricted environment".
- The **crwctl** 2.15 management tool is available. See Section 3.3.1, "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. Building offline registry images

5.3.2.1. Building an offline devfile registry image

This section describes how to build an offline devfile registry image. Starting workspaces without relying on resources from the outside Internet requires building this image. The image contains all sample projects referenced in devfiles as **zip** files.

Prerequisites:

- A running installation of podman or docker.

Procedure

1. Clone the devfile registry repository and check out the version to deploy:

```
$ git clone git@github.com:redhat-developer/codeready-workspaces.git
$ cd codeready-workspaces
$ git checkout crw-2.15-rhel-8
```

2. Build an offline devfile registry image:

```
$ cd dependencies/che-devfile-registry
$ ./build.sh --organization <my-org> \
        --registry <my-registry> \
        --tag <my-tag> \
        --offline
```

NOTE

To display full options for the **build.sh** script, use the **--help** parameter.

Additional resources

- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.15/html-single/administration_guide/index#customizing-the-registries.adoc.

5.3.2.2. Building an offline plug-in registry image

This section describes how to build an offline plug-in registry image. Starting workspaces without relying on resources from the outside Internet requires building this image. The image contains plug-in metadata and all plug-in or extension artifacts.

Prerequisites

- Node.js 12.x
- A running version of yarn. See Installing Yarn.
- **./node_modules/.bin** is in the **PATH** environment variable.
- A running installation of podman or docker.

Procedure

1. Clone the plug-in registry repository and check out the version to deploy:

```
$ git clone git@github.com:redhat-developer/codeready-workspaces.git
$ cd codeready-workspaces
$ git checkout crw-2.15-rhel-8
```

2. Build offline plug-in registry image:

```
$ cd dependencies/che-plugin-registry
$ ./build.sh --organization <my-org> \
        --registry <my-registry> \
        --tag <my-tag> \
        --offline \
        --skip-digest-generation
```

NOTE

To display full options for the **build.sh** script, use the **--help** parameter.

Additional resources

- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.15/html-single/administration_guide/index#customizing-the-registries.adoc.

5.3.3. 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>
```

NOTE

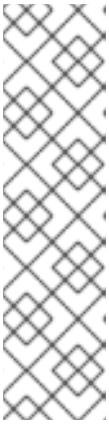
If you encounter an error, like **x509: certificate signed by unknown authority**, when attempting to push to the internal registry, try one of these workarounds:

- 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-** or **plugin-**
Workspaces	**stacks-**, **plugin-**

Table 5.3. Images to copy in the private registry

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/backup-rhel8@sha256:6b636d6bba6c509756803c4186960ed69adfa2eae42dde5af48b67c6e0794915	**backup-rhel8@sha256:6b636d6bba6c509756803c4186960ed69adfa2eae42dde5af48b67c6e0794915**
registry.redhat.io/codeready-workspaces/configbump-rhel8@sha256:15574551ec79aa8cd9e0eea3d379fc7a77a4e16cc92f937b4a89c6f9a6f2ce40	**configbump-rhel8@sha256:15574551ec79aa8cd9e0eea3d379fc7a77a4e16cc92f937b4a89c6f9a6f2ce40**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:1c8b06e457ba008f86e5fefc82013acdb4639317cde809e926931d202a194a17	**crw-2-rhel8-operator@sha256:1c8b06e457ba008f86e5fefc82013acdb4639317cde809e926931d202a194a17**
registry.redhat.io/codeready-workspaces/dashboard-rhel8@sha256:e46636f0c66221d9a01506b114e18f3d3afe61da99bf224e71cf4051235e51ac	**dashboard-rhel8@sha256:e46636f0c66221d9a01506b114e18f3d3afe61da99bf224e71cf4051235e51ac**
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:fdcd72766757f08486a6e4dbf3a24bf084aefdb2e86971c440048aec0315d7e8	**devfileregistry-rhel8@sha256:fdcd72766757f08486a6e4dbf3a24bf084aefdb2e86971c440048aec0315d7e8**
registry.redhat.io/codeready-workspaces/idea-rhel8@sha256:eff6db1da4c9743ff77b91acf08378bce6a652826b3b252512e63f767de07785	**idea-rhel8@sha256:eff6db1da4c9743ff77b91acf08378bce6a652826b3b252512e63f767de07785**
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:6176e28c4c02f0a40f8192088ccb505ce5722258bcaab0addff9bafa310c1ca4	**jwtproxy-rhel8@sha256:6176e28c4c02f0a40f8192088ccb505ce5722258bcaab0addff9bafa310c1ca4**
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:dc0e082c9522158cb12345b1d184c3803d8a4a63a7189940e853e51557e43acf	**machineexec-rhel8@sha256:dc0e082c9522158cb12345b1d184c3803d8a4a63a7189940e853e51557e43acf**
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:315273182e1f4dc884365fc3330ada3937b40369f3faf7762847ec433c3ac537	**plugin-java11-rhel8@sha256:315273182e1f4dc884365fc3330ada3937b40369f3faf7762847ec433c3ac537**
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:8cb1e495825051b83cf903bb317e55823a6f57b3bad92e9407dc8fa59c24c0cc	**plugin-java8-rhel8@sha256:8cb1e495825051b83cf903bb317e55823a6f57b3bad92e9407dc8fa59c24c0cc**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:75fe8823dea867489b68169b764dc8b0b03290a456e9bfec5fe0cc413eec7355	**plugin-kubernetes-rhel8@sha256:75fe8823dea867489b68169b764dc8b0b03290a456e9bfec5fe0cc413eec7355**
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:d7603582f7ace76283641809b0c61dbcb78621735e536b789428e5a910d35af3	**plugin-openshift-rhel8@sha256:d7603582f7ace76283641809b0c61dbcb78621735e536b789428e5a910d35af3**
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:6d13003539fcbda201065eae2e66dc67fed007ba3ba41fb3b8ec841650c52bc2	**pluginbroker-artifacts-rhel8@sha256:6d13003539fcbda201065eae2e66dc67fed007ba3ba41fb3b8ec841650c52bc2**
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:de8ede01ce5d3b06ae8b1866bb482bb937f020f7dee5dfb20b041f02c1e63f68	**pluginbroker-metadata-rhel8@sha256:de8ede01ce5d3b06ae8b1866bb482bb937f020f7dee5dfb20b041f02c1e63f68**
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:cbb82d5bcea22d6d65644c2a4c88ce1e3a082e8a696217d6a104b67daa60384e	**pluginregistry-rhel8@sha256:cbb82d5bcea22d6d65644c2a4c88ce1e3a082e8a696217d6a104b67daa60384e**
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:e1694549ca2af22a1d1780cc7d92bb0829a411f74377f825eab3e0fba7c020d9	**server-rhel8@sha256:e1694549ca2af22a1d1780cc7d92bb0829a411f74377f825eab3e0fba7c020d9**
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:c2f38140f52112b2a7688c2a179afcaa930ad6216925eb322cfd9634a71cfc13	**stacks-cpp-rhel8@sha256:c2f38140f52112b2a7688c2a179afcaa930ad6216925eb322cfd9634a71cfc13**
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:f48fe1caa5be1ae91140681bee159ca8b11dc687fa50fbf9dc5644f4852bf5c8	**stacks-dotnet-rhel8@sha256:f48fe1caa5be1ae91140681bee159ca8b11dc687fa50fbf9dc5644f4852bf5c8**

<source-image>	*<target-image>*
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:db76d04752973223e2c0de9401ebf06b84263e1bb6d29f1455daaff0cb39c1b3	**stacks-golang-rhel8@sha256:db76d04752973223e2c0de9401ebf06b84263e1bb6d29f1455daaff0cb39c1b3**
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:d120c41ee8dd80fb960dd4c1657bede536d32f13f3c3ca84e986a830ec2ead3b	**stacks-php-rhel8@sha256:d120c41ee8dd80fb960dd4c1657bede536d32f13f3c3ca84e986a830ec2ead3b**
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:5d26cf000924716d8d03969121a4c636e7fc8ef08aa21148eafa28a2c4aeaff7	**theia-endpoint-rhel8@sha256:5d26cf000924716d8d03969121a4c636e7fc8ef08aa21148eafa28a2c4aeaff7**
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:6000d00ef1029583642c01fec588f92addb95f16d56d0c23991a8f19314b0f06	**theia-rhel8@sha256:6000d00ef1029583642c01fec588f92addb95f16d56d0c23991a8f19314b0f06**
registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:70215465e2ad65a61d1b5401378532a3a10aa60afdda0702fb6061d89b8ba3be	**traefik-rhel8@sha256:70215465e2ad65a61d1b5401378532a3a10aa60afdda0702fb6061d89b8ba3be**
registry.redhat.io/devworkspace/devworkspace-rhel8-operator@sha256:3f96fb70c3f56dea3384ea31b9252a5c6aca8e0f33dc53be590f134912244078	**devworkspacedevworkspace-rhel8-operator@sha256:3f96fb70c3f56dea3384ea31b9252a5c6aca8e0f33dc53be590f134912244078**
registry.redhat.io/jboss-eap-7/eap-xp3-openjdk11-openshift-rhel8@sha256:bb3072afdbf31ddd1071fea37ed5308db3bf8a2478b5aa5aff8373e8042d6aeb	**eap-xp3-openjdk11-openshift-rhel8@sha256:bb3072afdbf31ddd1071fea37ed5308db3bf8a2478b5aa5aff8373e8042d6aeb**
registry.redhat.io/jboss-eap-7/eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d	**eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d**

<source-image>	*<target-image>*
registry.redhat.io/openshift4/ose-kube-rbac-proxy@sha256:1dc542b5ab33368443f698305a90c617385b4e9b101acc4acc0aa7b4bf58a292	**openshift4ose-kube-rbac-proxy@sha256:1dc542b5ab33368443f698305a90c617385b4e9b101acc4acc0aa7b4bf58a292**
registry.redhat.io/openshift4/ose-oauth-proxy@sha256:83988048d5f585ca936442963e23b77520e1e4d8c3d5b8160e43ae834a24b720	**openshift4ose-oauth-proxy@sha256:83988048d5f585ca936442963e23b77520e1e4d8c3d5b8160e43ae834a24b720**
registry.redhat.io/rh-sso-7/sso75-openshift-rhel8@sha256:dd4ea229521fb58dda7e547ea6db993156f4c61aa8a00f2fd1375bb77168b6e6	**sso75-openshift-rhel8@sha256:dd4ea229521fb58dda7e547ea6db993156f4c61aa8a00f2fd1375bb77168b6e6**
registry.redhat.io/rhel8/postgresql-13@sha256:6032adb3eac903ee8aa61f296ca9aaa57f5709e5673504b609222e042823f195	**postgresql-13@sha256:6032adb3eac903ee8aa61f296ca9aaa57f5709e5673504b609222e042823f195**
registry.redhat.io/rhel8/postgresql-96@sha256:314747a4a64ac16c33ead6a34479dccf16b9a07abf440ea7eeef7cda4cd19e32	**postgresql-96@sha256:314747a4a64ac16c33ead6a34479dccf16b9a07abf440ea7eeef7cda4cd19e32**
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	**mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73**
registry.redhat.io/ubi8/ubi-minimal@sha256:2e4bbb2be6e7aff711ddc93f0b07e49c93d41e4c2ffc8ea75f804ad6fe25564e	**ubi8ubi-minimal@sha256:2e4bbb2be6e7aff711ddc93f0b07e49c93d41e4c2ffc8ea75f804ad6fe25564e**

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 link: - CodeReady Workspaces Operator ClusterServiceVersion sources.

5.3.4. 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 2.14 of Red Hat CodeReady Workspaces, installed using the CLI management tool on the same instance of OpenShift, with the crwctl **--installer operator** method, in the ***<openshift-workspaces>*** project. See Section 3.4, "Installing CodeReady Workspaces in a restricted environment".
- Essential container images are available to the CodeReady Workspaces server running in the cluster. See Section 5.3.3, "Preparing an private registry".
- The **crwctl** 2.15 management tool is available. See Section 3.3.1, "Installing the crwctl CLI management tool".

Procedure

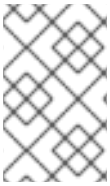
1. In all running workspaces in the CodeReady Workspaces 2.14 instance, save and push changes back to the Git repositories.
2. Stop all workspaces in the CodeReady Workspaces 2.14 instance.
3. Run the following command:

```
$ crwctl server:update --che-operator-image=<image-registry>/<organization>/crw-2-rhel8-operator:2.15 -n openshift-workspaces
```

   - *<image-registry>*: A hostname and a port of the container-image registry accessible in the restricted environment.
   - *<organization>*: An organization of the container-image registry. See: Section 5.3.3, "Preparing an private registry".

Verification steps

1. Navigate to the CodeReady Workspaces instance.
2. The 2.15 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.

5.4. UPGRADING CODEREADY WORKSPACES THAT USES PROJECT STRATEGIES OTHER THAN 'PER USER'

This section describes how to upgrade CodeReady Workspaces that uses project strategies other than 'per user'.

CodeReady Workspaces intends to use Kubernetes secrets as a storage for all sensitive user data. One project per user simplifies the design of the workspaces. This is the reason why project strategies other than **per user** become deprecated. The deprecation process happens in two steps. In the **First Step** project strategies other than **per user** are allowed but not recommended. In the **Second Step** support for project strategies other than **per user** is going to be removed.

NOTE

No automated upgrade support exists between **First Step** and **Second Step** for the installations where project strategies other than **per user** are used without losing data.

Prerequisites

- CodeReady Workspaces configured with the project strategies other than **per user**.
- Intention to use CodeReady Workspaces configured with the **per user** namespace strategies **per user**.

5.4.1. Upgrading CodeReady Workspaces and backing up user data

Procedure

1. Notify all CodeReady Workspaces users about the upcoming data wipe.

NOTE

To back up the data, you can commit workspace configuration to an SCM server and use factories to restore it later.

2. Re-install CodeReady Workspaces with **per user** namespace strategy.

5.4.2. Upgrading CodeReady Workspaces and losing user data

When CodeReady Workspaces is upgraded and user data is not backed up, workspace configuration and user preferences are going to be preserved but all runtime data will be wiped out.

Procedure

1. Notify all CodeReady Workspaces users about the upcoming data wipe.
2. Change project strategy to **per user**.

NOTE

Upgrading without backing up user data has disadvantage. Original PVs with runtime data are going to be preserved and will no longer be used. This may lead to the waste of resources.

Additional resources

- Section 4.2, “Configuring workspace target project”
- Chapter 3, *Installing CodeReady Workspaces*
- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.15/html-single/end-user_guide/index#workspaces-overview.adoc
- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.15/html-single/end-user_guide/index#importing-the-source-code-of-a-project-into-a-workspace.adoc

5.5. ROLLING BACK A CODEREADY WORKSPACES UPGRADE

To restore CodeReady Workspaces to the pre-upgrade version, roll back the CodeReady Workspaces version upgrade as follows:

Prerequisites

- Installed **crwctl**.

Procedure

- Run the following command on a command line:

```
$ crwctl server:restore --rollback
```

NOTE

CodeReady Workspaces Operator automatically creates a backup before every upgrade.

CHAPTER 6. UNINSTALLING CODEREADY WORKSPACES

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

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

6.1. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION USING THE OPENSHIFT WEB CONSOLE

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

Prerequisites

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

Procedure

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

NOTE

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

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

NOTE

The default Red Hat CodeReady Workspaces **checluster** Custom Resource name is *<codeready-workspaces>*.

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

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

6.2. UNINSTALLING CODEREADY WORKSPACES AFTER OPERATORHUB INSTALLATION USING OPENSHIFT CLI

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

Prerequisites

- CodeReady Workspaces was installed on an OpenShift cluster using OperatorHub.
- The **oc** tool is available.

Procedure

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

To uninstall a CodeReady Workspaces instance from a cluster:

1. Sign in to the cluster:

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

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

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

3. Obtain the **checluster** Custom Resource name. The following shows a **checluster** Custom Resource named **codeready-workspaces**:

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

4. Delete the CodeReady Workspaces cluster:

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

5. Obtain the name of the CodeReady Workspaces cluster service version (CSV) module. The following detects a CSV module named **codeready.v2.15**:

```
$ oc get csv
NAME                DISPLAY       VERSION   REPLACES            PHASE
codeready.v2.15   Red Hat CodeReady Workspaces   2.15     codeready.v2.14   Succeeded
```

6. Delete the CodeReady Workspaces CSV:

```
$ oc delete csv codeready.v2.15
clusterserviceversion.operators.coreos.com "codeready.v2.15" 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 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
```