Red Hat CodeReady Workspaces 2.4

インストールガイド

Installing Red Hat CodeReady Workspaces 2.4

Last Updated: 2021-06-16

Red Hat CodeReady Workspaces 2.4 インストールガイド

Installing Red Hat CodeReady Workspaces 2.4

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

管理者による Red Hat CodeReady Workspaces のインストールについての情報

目次

第1章 サポートされるプラットフォーム

このセクションでは、OpenShift Container Platform および OpenShift Dedicated での CodeReady Workspaces 2.4 の可用性およびサポートされるインストール方法を説明します。

Red Hat CodeReady Workspaces をサポートする最小の OpenShift Container Platform バージョンは OpenShift Container Platform 3.11 です。

表1.1 OpenShift Container Platform および OpenShift Dedicated での CodeReady Workspaces 2.4 でサポートされるデプロイメント環境

プラットフォーム	アーキテクチャー	デプロイメント方法
OpenShift Container Platform 3.11	AMD64 および Intel 64 (x86_64)	**crwctl**
OpenShift Container Platform 4.4	AMD64 および Intel 64 (x86_64)	OperatorHub
OpenShift Container Platform 4.4	IBM Z (s390x)	OperatorHub
OpenShift Container Platform 4.5	AMD64 および Intel 64 (x86_64)	OperatorHub
OpenShift Dedicated 4.3	AMD64 および Intel 64 (x86_64)	アドオン

注記

- OpenShift Container Platform 4.4 および 4.5 では、OperatorHub のインストール方法が利用できない場合に、**crwctl** を非公式のバックアップインストールの方法として使用することを検討してください。

第2章 CODEREADY WORKSPACES インストールの設定

以下のセクションでは、Operator を使用して Red Hat CodeReady Workspaces をインストールする設定オプションについて説明します。

2.1. CHECLUSTER カスタムリソースについて

CodeReady Workspaces のデフォルトデプロイメントは、Red Hat CodeReady Workspaces Operator によって準仮想化された **CheCluster** カスタムリソースのアプリケーションで構成されています。

CheCluster カスタムリソース

- CodeReady Workspaces インストール全体の設定を記述する YAML ドキュメント。
- **auth**、**database**、**server**、**storage** などの各コンポーネントを設定するセクションが含まれます。

Red Hat CodeReady Workspaces Operator の役割

- **CheCluster** カスタムリソースを、CodeReady Workspaces インストールの各コンポーネントで使用できる設定 (ConfigMap) に変換します。

OpenShift プラットフォームの役割

- 各コンポーネントの設定 (ConfigMap) を適用するには、以下を実行します。
- 必要な Pod を作成するには、以下を実行します。
- OpenShift がコンポーネントの設定で変更を検知すると、Pod を適宜再起動します。

> **例2.1 CodeReady Workspaces サーバーコンポーネントの主なプロパティーの設定**
>
> 1. ユーザーは、**server** に関連する一部の設定が含まれる **CheCluster** カスタムリソースを適用します。
> 2. Operator は **che** という必要な ConfigMap を生成します。
> 3. OpenShift は ConfigMap の変更を検知し、CodeReady Workspaces Pod の再起動をトリガーします。

関連情報

- Operator について理解し ます。
- カスタムリソースについて
- **CheCluster** カスタムリソースを変更する方法については、選択したインストール手順を参照してください。

2.2. CHECLUSTER カスタムリソースフィールドの参照

本セクションでは、**CheCluster** カスタムリソースのカスタマイズに使用できるすべてのフィールドについて説明します。

- 例2.2「最小の **CheCluster** カスタムリソースの例。」
- 表2.3「**CheCluster** CodeReady Workspaces インストールによって使用される認証に関連するカスタムリソース **auth** 設定」
- 表2.2「**CheCluster** CodeReady Workspaces で使用されるデータベースに関連するカスタムリソース **database** 設定」
- 表2.1「CodeReady Workspaces サーバーコンポーネントに関連する **CheCluster** カスタムリソースの **server** 設定。」
- 表2.4「**CheCluster** CodeReady Workspaces で使用される永続ストレージに関連するカスタムリソース **storage** 設定」
- 表2.5「**CheCluster** OpenShift の CodeReady Workspaces インストールに固有のカスタムリソース **k8s** 設定」
- 表2.6「**CheCluster** カスタムリソース **status** は、CodeReady Workspaces インストールの観察される状態を定義します。」

例2.2 最小の CheCluster カスタムリソースの例。

```
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'
```

表2.1 CodeReady Workspaces サーバーコンポーネントに関連する CheCluster カスタムリソースの server 設定。

プロパティー	デフォルト値	説明
airGapContainerRegistryHostname	省略	イメージのプルに使用する別のコンテナーレジストリーへのオプションのホスト名または URL。この値は、CodeReady Workspaces デプロイメントに関連するすべてのデフォルトコンテナーイメージで定義されるコンテナーレジストリーのホスト名を上書きします。これは、必要な環境に CodeReady Workspaces をインストールする場合にとくに便利です。

プロパティー	デフォルト値	説明
airGapContainerRegistryOrganization	省略	イメージのプルに使用する代替コンテナーレジストリーのオプションのリポジトリー名。この値は、CodeReady Workspaces デプロイメントに関連するすべてのデフォルトコンテナーイメージで定義されるコンテナーレジストリーの組織を上書きします。これは、必要な環境に CodeReady Workspaces をインストールする場合にとくに便利です。
cheDebug	**false**	CodeReady Workspaces サーバーのデバッグモードを有効にします。
cheFlavor	**codeready-workspaces**	インストールのフレーバー。
cheHost	Operator は値を自動的に設定します。	インストールされた CodeReady Workspaces サーバーのパブリックホスト名。
cheImagePullPolicy	**Always** 他の場合は **nightly** または **latest** イメージ、および **IfNotPresent**	CodeReady Workspaces デプロイメントで使用されるイメージプルポリシーを上書きします。
cheImageTag	省略	CodeReady Workspaces デプロイメントで使用されるコンテナーイメージのタグを上書きします。Operator によって提供されるデフォルトのイメージタグを使用するには、これを省略するか、または空のままにします。
cheImage	省略	CodeReady Workspaces デプロイメントで使用されるコンテナーイメージを上書きします。これには、コンテナーイメージタグは含まれません。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。
cheLogLevel	**INFO**	CodeReady Workspaces サーバーのログレベル： **INFO** または **DEBUG**。
cheWorkspaceClusterRole	省略	CodeReady Workspaces ワークスペースのユーザーにバインドされるカスタムロール。デフォルトロールを使用するには、省略するか、または空のままにします。
customCheProperties	省略	**CheCluster** カスタムリソース(CR)の他のフィールドからすでに生成されている値に加えて、CodeReady Workspaces サーバーによって使用される生成された **codeready-workspaces** ConfigMap に適用される追加の環境変数のマップ。**customCheProperties** に他の CR フィールドからの **codeready-workspaces** ConfigMap で生成されるプロパティーが含まれる場合、代わりに **customCheProperties** に定義される値が使用されます。

プロパティー	デフォルト値	説明
devfileRegistryImage	省略	Devfile レジストリーのデプロイメントで使用されるコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。
devfileRegistryMemoryLimit	**256Mi**	Devfile レジストリーのデプロイメントで使用されるメモリー制限を上書きします。
devfileRegistryMemoryRequest	**16Mi**	Devfile レジストリーのデプロイメントで使用されるメモリー要求を上書きします。
devfileRegistryPullPolicy	**Always** 他の場合は **nightly** または **latest** イメージ、および **IfNotPresent**	Devfile レジストリーのデプロイメントで使用されるイメージプルポリシーを上書きします。
devfileRegistryUrl	Operator は値を自動的に設定します。	サンプルのすぐに使用できる devfile を提供する Devfile レジストリーの公開 URL。外部 devfile レジストリーを使用する場合はこれを設定します（**externalDevfileRegistry** フィールドを参照）。
externalDevfileRegistry	**false**	Operator に専用の Devfile レジストリーサーバーをデプロイするよう指示します。デフォルトでは、専用の devfile レジストリーサーバーが起動します。**externalDevfileRegistry** が **true** に設定されている場合、Operator は専用のレジストリーサーバーを自動的に起動せず、**devfileRegistryUrl** フィールドを手動で設定する必要があります。
externalPluginRegistry	**false**	Operator に専用のプラグインレジストリーサーバーをデプロイするよう指示します。デフォルトでは、専用のプラグインレジストリーサーバーが起動します。**externalPluginRegistry** が **true** に設定されている場合、Operator は専用サーバーを自動的にデプロイせず、**pluginRegistryUrl** フィールドを手動で設定する必要があります。
nonProxyHosts	省略	設定済みのプロキシーを使用しないホストの一覧。**|** を区切り文字として使用します。たとえば、**localhost|my.host.com|123.42.12.32** はプロキシーの設定が必要な場合にのみ使用します（**proxyURL** フィールドも参照してください）。
pluginRegistryImage	省略	プラグインレジストリーのデプロイメントで使用されるコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。

プロパティー	デフォルト値	説明
pluginRegistryMemoryLimit	**256Mi**	プラグインレジストリーのデプロイメントで使用されるメモリー制限を上書きします。
pluginRegistryMemoryRequest	**16Mi**	プラグインレジストリーのデプロイメントで使用されるメモリー要求を上書きします。
pluginRegistryPullPolicy	**Always** 他の場合は **nightly** または **latest** イメージ、および **IfNotPresent**	プラグインレジストリーのデプロイメントで使用されるイメージプルポリシーを上書きします。
pluginRegistryUrl	Operator は値を自動的に設定します。	サンプルのすぐに使用できる devfile を提供するプラグインレジストリーの公開 URL。これは、外部の devfile レジストリーを使用している場合にのみ設定します（**externalPluginRegistry** フィールドを参照）。
proxyPassword	省略	プロキシーサーバーのパスワード。プロキシー設定が必要である場合にのみ使用します。
proxyPort	省略	プロキシーサーバーのポート。プロキシーの設定が必要な場合にのみ使用します（**proxyURL** フィールドも参照してください）。
proxyURL	省略	プロキシーサーバーの URL (プロトコル+ホスト名)。これにより、CodeReady Workspaces サーバーおよびワークスペースコンテナーの **JAVA_OPTS** および **https(s)_proxy** 変数の適切な変更が実行されます。プロキシーの設定が必要な場合にのみ使用します。
proxyUser	省略	プロキシーサーバーのユーザー名。プロキシーの設定が必要な場合にのみ使用します（**proxyURL** フィールドも参照してください）。
serverMemoryLimit	**1Gi**	CodeReady Workspaces サーバーのデプロイメントで使用されるメモリー制限を上書きします。
serverMemoryRequest	**512Mi**	CodeReady Workspaces サーバーのデプロイメントで使用されるメモリー要求を上書きします。
tlsSupport	**true**	Operator に対して CodeReady Workspaces を TLS モードでデプロイするように指示します。

表2.2 CheCluster CodeReady Workspaces で使用されるデータベースに関連するカスタムリソース database 設定

プロパティー	デフォルト値	説明
chePostgresDb	**dbche**	CodeReady Workspaces サーバーがデータベースへの接続に使用する PostgreSQL データベース名。
chePostgresHostName	Operator は値を自動的に設定します。	CodeReady Workspaces サーバーが接続に使用する PostgreSQL データベースのホスト名。デフォルトは **postgres** です。この値は、外部データベースを使用する場合のみオーバーライドします。（**externalDb** フィールドを参照）
chePostgresPassword	自動生成される値	CodeReady Workspaces サーバーがデータベースへの接続に使用する PostgreSQL パスワード。
chePostgresPort	**5432**	CodeReady Workspaces サーバーが接続に使用する PostgreSQL データベースのポート。この値は、外部データベースを使用する場合のみオーバーライドします（**externalDb**フィールドを参照）。
chePostgresUser	**pgche**	CodeReady Workspaces サーバーがデータベースへの接続に使用する PostgreSQL ユーザー。
externalDb	**false**	Operator に専用のデータベースをデプロイするよう指示します。デフォルトでは、専用の PostgreSQL データベースは CodeReady Workspaces インストールの一部としてデプロイされます。**true** に設定すると、Operator は専用のデータベースを自動的にデプロイしません。接続の詳細を外部データベースに提供する必要があります。**chePostgres** で始まるすべてのフィールドを参照してください。
postgresImagePullPolicy	常に **nightly** または **latest** イメージの場合は、その他の場合は **IfNotPresent** になります。	PostgreSQL データベースのデプロイメントで使用されるイメージプルポリシーを上書きします。
postgresImage	省略	PostgreSQL データベースのデプロイメントで使用されるコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。

表2.3 CheCluster CodeReady Workspaces インストールによって使用される認証に関連するカスタムリソース auth 設定

プロパティー	デフォルト値	説明

プロパティー	デフォルト値	説明
externalIdentityProvider	**false**	デフォルトでは、専用のアイデンティティープロバイダーサーバーは CodeReady Workspaces インストールの一部としてデプロイされます。ただし、**externalIdentityProvider** が **true** の場合、専用のアイデンティティープロバイダーは Operator によってデプロイされず、使用する外部アイデンティティープロバイダーについての詳細情報を提供する必要がある場合があります。**identityProvider** で始まるその他のフィールドすべても参照してください。
identityProviderAdminUserName	**admin**	アイデンティティープロバイダーの管理ユーザーの名前を上書きします。
identityProviderClientId	省略	CodeReady Workspaces に使用する必要があるアイデンティティープロバイダー(RH-SSO / RH SSO) **client-id** の名前。これは、外部アイデンティティープロバイダーを使用する場合にのみ上書きするのに便利です（**externalIdentityProvider** フィールドを参照）。省略されているか、または空のままの場合、サフィックスとして **-public** が付けられた **flavor** フィールドの値に設定されます。
identityProviderImagePullPolicy	**Always** 他の場合は **nightly** または **latest** イメージ、および **IfNotPresent**	アイデンティティープロバイダー(RH-SSO/RH SSO)デプロイメントで使用するイメージプルポリシーを上書きします。
identityProviderImage	省略	アイデンティティープロバイダー(RH-SSO / RH SSO)デプロイメントで使用するコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。
identityProviderPassword	省略	RH-SSO 管理ユーザーのパスワードを上書きします。外部アイデンティティープロバイダーを使用する場合のみ上書きします（**externalIdentityProvider** フィールドを参照）。自動生成されるパスワードを設定するには、省略するか、または空のままにします。
identityProviderPostgresPassword	Operator は値を自動的に設定します。	データベースに接続するためのアイデンティティープロバイダー(RH-SSO/RH SSO)のパスワードこれは、外部アイデンティティープロバイダーを使用する場合にのみ上書きするのに便利です（**externalIdentityProvider** フィールドを参照）。
identityProviderRealm	省略	アイデンティティープロバイダー(RH-SSO / RH SSO)レルムの名前。外部アイデンティティープロバイダーを使用する場合のみ上書きします（**externalIdentityProvider** フィールドを参照）。省略するか、または空のままに **flavor** フィールドの値に設定します。

プロパティー	デフォルト値	説明
identityProviderURL	Operator は値を自動的に設定します。	Operator に専用のアイデンティティープロバイダー（RH-SSO または RH SSO インスタンス）をデプロイするように指示します。アイデンティティープロバイダーサーバーの公開 URL（RH-SSO / RH SSO サーバー）。外部アイデンティティープロバイダーを使用する場合のみ設定します（**externalIdentityProvider** フィールドを参照）。
oAuthClientName	Operator は値を自動的に設定します。	OpenShift 側でアイデンティティーフェデレーションを設定するために使用される OpenShift **OAuthClient** リソースの名前。**OpenShiftoAuth** フィールドも参照してください。
oAuthSecret	Operator は値を自動的に設定します。	OpenShift 側でアイデンティティーフェデレーションを設定するために使用される OpenShift **OAuthClient** リソースに設定されたシークレットの名前。**OAuthClientName** フィールドも参照してください。
openShiftoAuth	**true** on OpenShift	アイデンティティープロバイダー(RH-SSO / RHSSO)と OpenShift OAuth の統合を有効にします。これにより、ユーザーは OpenShift ログインでログインし、独自のワークスペースを個人の OpenShift プロジェクトに作成することができます。**kubeadmin** ユーザーがサポートされず、ログインしても CodeReady Workspaces ダッシュボードにはアクセスできません。
updateAdminPassword	**false**	デフォルトの **admin** CodeReady Workspaces ユーザーが初回ログイン時にパスワードを強制的に更新します。

表2.4 CheCluster CodeReady Workspaces で使用される永続ストレージに関連するカスタムリソース storage 設定

プロパティー	デフォルト値	説明
postgresPVCStorageClassName	省略	PostgreSQL データベース専用の Persistent Volume Claim（永続ボリューム要求、PVC）のストレージクラス。デフォルトのストレージクラスを使用するには、省略するか、または空のままにします。
preCreateSubPaths	**false**	CodeReady Workspaces サーバーに対し、永続ボリュームでサブパスを事前に作成するために特別な Pod を起動するように指示します。K8S クラスターの設定に応じてこれを有効にします。
pvcClaimSize	**1Gi**	ワークスペースの永続ボリューム要求 (PVC) のサイズ。

プロパティー	デフォルト値	説明
pvcJobsImage	省略	永続ボリュームでサブパスを作成するために使用されるコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。**preCreateSubPaths** フィールドも参照してください。
pvcStrategy	**common**	使用できるオプション：'common'（1 つのボリューム内のすべてのワークスペース PVC）、**per-workspace**（すべての宣言されたボリュームについてワークスペースごとに 1 つの PVC）、および **unique**（宣言されたボリュームごとに 1 つの PVC）
workspacePVCStorageClassName	省略	CodeReady Workspaces ワークスペース専用の Persistent Volume Claim（永続ボリューム要求、PVC）のストレージクラス。デフォルトのストレージクラスを使用するには、省略するか、または空のままにします。

表2.5 CheCluster OpenShift の CodeReady Workspaces インストールに固有のカスタムリソースk8s 設定

プロパティー	デフォルト値	説明
ingressClass	**nginx**	Ingress を管理するコントローラーを定義する Ingress クラス。
ingressDomain	省略	K8S クラスターのグローバル Ingress ドメイン。このフィールドは明示的に指定する必要があります。これにより、CodeReady Workspaces 関連の Ingress で **is kubernetes.io/ingress.class** アノテーションが実行されます。
ingressStrategy	**multi-host**	Ingress 作成のストラテジー。これは、**multi-host**（ホストは Ingress で明示的に指定されます）、**single-host**（ホストは指定される、パスベースのルール）、および **default-host.***（ホストは指定されない、パスベースのルール）。
securityContextFsGroup,omitempty	**1724**	fsGroup は、CodeReady Workspaces Pod および Workspace Pod コンテナーで実行されます。
securityContextRunAsUser	**1724**	CodeReady Workspaces Pod および Workspace Pod コンテナーの ID は以下のようになります。
tlsSecretName	省略	TLS が有効にされている場合、Ingress TLS 終端を設定するために使用されるシークレットの名前。**tlsSupport** フィールドも参照してください。

表2.6 CheCluster カスタムリソース status は、CodeReady Workspaces インストールの観察される状態を定義します。

プロパティー	説明
cheClusterRunning	CodeReady Workspaces インストールのステータス。**Available**、**Unavailable**、または **Available, Rolling Update in Progress** を指定できます。
cheURL	CodeReady Workspaces サーバーへの公開 URL。
cheVersion	現時点でインストールされている CodeReady Workspaces バージョン。
dbProvisioned	PostgreSQL インスタンスが正しくプロビジョニングされているかどうかを示します。
devfileRegistryURL	Devfile レジストリーへの公開 URL。
helpLink	現在の Operator のステータスに関連するヘルプを検索する場所の URL。
keycloakProvisioned	アイデンティティープロバイダーインスタンス(RH-SSO / RH SSO)がレルム、クライアント、およびユーザーと共にプロビジョニングされているかどうかを示します。
keycloakURL	アイデンティティープロバイダーサーバー(RH-SSO / RH SSO)の公開 URL。
message	Pod がこの状態にある理由の詳細と共に、人間が判読できるメッセージです。
openShiftoAuthProvisioned	アイデンティティープロバイダーインスタンス(RH-SSO / RH SSO)が OpenShift OAuth と統合するように設定されているかどうかを示します。
pluginRegistryURL	プラグインレジストリーへの公開 URL。
reason	Pod がこの状態にある理由の詳細が含まれる簡単な CamelCase メッセージ。

第3章 CODEREADY WORKSPACES のインストール

本セクションでは、Red Hat CodeReady Workspaces をインストールする手順を説明します。インストール方法は、ターゲットプラットフォームと環境の制限によって異なります。

3.1. OPERATORHUB を使用した OPENSHIFT 4 への CODEREADY WORKSPACES のインストール

本セクションでは、OpenShift 4 Web コンソールで利用可能な CodeReady Workspaces Operator を使用して CodeReady Workspaces をインストールする方法を説明します。

Operator は、OpenShift アプリケーションをパッケージ化し、デプロイし、管理する方法です。これは、以下も提供します。

- インストールおよびアップグレードの再現性。
- すべてのシステムコンポーネントの定期的なヘルスチェック。
- OpenShift コンポーネントおよび独立ソフトウェアベンダー (ISV) コンテンツの OTA (Over-the-air) 更新。
- フィールドエンジニアの知識をカプセル化し、すべてのユーザーに展開する場所。

前提条件

- OpenShift 4 の実行中のインスタンスの管理者アカウント

3.1.1. OpenShift Web コンソールでのプロジェクトの作成

プロジェクトを使用すると、クラスターの異なるリソースを分離された単位で編成し、管理できます。まずプロジェクトを作成し、Red Hat CodeReady Workspaces Operator をホストします。

手順

1. OpenShift Web コンソールを開きます。左側のパネルで **Home → Projects** セクションに移動します。
2. **Create Project** をクリックします。
3. プロジェクトの詳細を指定します。
   - Name: **openshift-workspaces**
   - Display Name: **Red Hat CodeReady Workspaces**
   - Description: **Red Hat CodeReady Workspaces**

3.1.2. Red Hat CodeReady Workspaces Operator のインストール

Red Hat CodeReady Workspaces Operator は、PostgreSQL、RH-SSO、イメージレジストリー、CodeReady Workspaces サーバーなどの CodeReady Workspaces を実行するためのすべてのリソースを提供し、これらのすべてのサービスも設定します。

前提条件

- クラスターの Web コンソールへのアクセス。

手順

1. Red Hat CodeReady Workspaces Operator をインストールするには、左側のパネルで **Operators → OperatorHub** セクションに移動します。
2. **Filter by keyword** フィールドに **Red Hat CodeReady Workspaces** を入力し、**Red Hat CodeReady Workspaces** タイルをクリックします。
3. **Red Hat CodeReady Workspaces**のポップアップウィンドウで、**Install** ボタンをクリックします。
4. **Install Operator** 画面で、以下のオプションを指定します。
   - **Installation mode**: **A specific project on the cluster**
   - **Installed Namespace**: *既存プロジェクトを選択 → **openshift-workspaces**

検証手順

1. Red Hat CodeReady Workspaces Operator が正しくインストールされたことを確認するには、左側のパネルで **Operators → Installed Operators** セクションに移動します。
2. **Installed Operators** 画面で、**Red Hat CodeReady Workspaces**名をクリックし、**Details** タブに移動します。
3. ページの下部にある **ClusterServiceVersion Details** セクションで、以下のメッセージを待機します。
   - Status: **Succeeded**
   - Status Reason: **install strategy completed with no errors**
4. **Events** タブに移動し、**install strategy completed with no errors** というメッセージが表示されるのを待機します。

3.1.3. Red Hat CodeReady Workspaces Operator のインスタンスの作成

以下の手順に従って、デフォルト設定で Red Hat CodeReady Workspaces をインストールします。設定を変更する場合は、*2章CodeReady Workspaces インストールの設定* を参照してください。

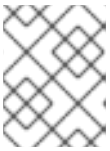
手順

1. Red Hat CodeReady Workspaces Operator のインスタンスを作成するには、左側のパネルで **Operators → Installed Operators** セクションに移動します。
2. **Installed Operators** 画面で、**Red Hat CodeReady Workspaces**名をクリックします。
3. **Operator Details** 画面の **Provided APIs** セクション内の **Details** タブで **Create Instance** リンクをクリックします。
4. **Create CheCluster** ページには、作成する CodeReady Workspaces インスタンス全体の設定が含まれます。これは **CheCluster** カスタムリソースです。デフォルト値を維持します。

5. **codeready-workspaces** クラスターを作成するには、ウィンドウの左下にある **Create** ボタンをクリックします。

6. **Operator Details** 画面の、**Red Hat CodeReady Workspaces Cluster**タブで、**codeready-workspaces** リンクをクリックします。

7. **codeready-workspaces** インスタンスに移動するには、**Red Hat CodeReady Workspaces URL** の下にあるリンクをクリックします。



注記

インストールには 5 分以上かかる場合があります。Red Hat CodeReady Workspaces インストールが完了すると、URL が表示されます。

検証手順

1. Red Hat CodeReady Workspaces インスタンスが正しくインストールされたことを確認するには、**CodeReady Workspaces Cluster** タブに移動します。**CheClusters** 画面には、Red Hat CodeReady Workspaces インスタンスの一覧とそのステータスが表示されます。

2. テーブルの **codeready-workspaces CheCluster** をクリックし、**Details** タブに移動します。

3. 以下のフィールドの内容を参照してください。

   - **Message**: このフィールドにはエラーメッセージが含まれます (ある場合)。予想される内容は **None** です。

   - **Red Hat CodeReady Workspaces URL**: デプロイメントが成功した場合に、Red Hat CodeReady Workspaces インスタンスの URL を表示します。

4. **Resources** タブに移動します。画面には、CodeReady Workspaces デプロイメントに割り当てられたリソースの一覧が表示されます。

5. リソースの状態の詳細を確認するには、その名前をクリックして、利用可能なタブの内容を検査します。

関連情報

- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.4/html-single/end-user_guide/index#navigating-codeready-workspaces-using-the-dashboard_crw.

- https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.4/html-single/administration_guide/index#viewing-operator-events.adoc.

- OpenShift Container Platform および OpenShift Dedicated バージョン 4.5 に CodeReady Workspaces をデプロイするために **crwctl** ユーティリティースクリプトを使用できます。この方法は非公式であり、OperatorHub を使用したインストール方法が利用できない場合にバックアップのインストール方法となります。「Operator を使用した CodeReady Workspaces の OpenShift 3 へのインストール」 セクションを参照してください。

3.2. CODEREADY WORKSPACES の OPENSHIFT CONTAINER PLATFORM 3.11 へのインストール

3.2.1. crwctl CLI 管理ツールのインストール

本セクションでは、CodeReady Workspaces CLI 管理ツールを使用して crwctl をインストールする方法を説明します。

手順

1. https://developers.redhat.com/products/codeready-workspaces/download に移動します。
2. バージョン 2.4 の CodeReady Workspaces CLI 管理ツールアーカイブをダウンロードします。
3. **${HOME}/crwctl** または **/opt/crwctl** などのフォルダーにアーカイブを展開します。
4. 展開したフォルダーから **crwctl** 実行可能ファイルを実行します。この例では、**${HOME}/crwctl/bin/crwctl version** のようになります。
5. オプションで、**bin** フォルダーを **$PATH** に追加し (例: **PATH=${PATH}:${HOME}/crwctl/bin**)、完全パスの指定なしで **crwctl** の実行を有効にします。

検証手順

crwctl version を実行すると、ツールの現行バージョンが表示されます。

3.2.2. Operator を使用した CodeReady Workspaces の OpenShift 3 へのインストール

本セクションでは、**crwctl** CLI 管理ツールを使用して、OpenShift 3 に CodeReady Workspaces をインストールする方法を説明します。このインストールの方法では Operator を使用し、TLS (HTTPS) を有効にします。

注記

直前の CodeReady Workspaces インストールから更新し、同じ OpenShift Container Platform 3.11 クラスターで複数のインスタンスを有効にする方法は、インストール手順で説明されます。

Operator は、OpenShift アプリケーションをパッケージ化し、デプロイし、管理する方法です。これは、以下も提供します。

- インストールおよびアップグレードの再現性。
- すべてのシステムコンポーネントの定期的なヘルスチェック。
- OpenShift コンポーネントおよび独立ソフトウェアベンダー (ISV) コンテンツの OTA (Over-the-air) 更新。
- フィールドエンジニアの知識をカプセル化し、すべてのユーザーに展開する場所。

ヒント

この方法は、OpenShift Container Platform および OpenShift Dedicated バージョン 3.11 でのみサポートされますが、OpenShift Container Platform および OpenShift Dedicated の新しいバージョンでも機能し、OperatorHub を使用したインストール方法が利用できない場合にバックアップのインストール方法として機能します。

前提条件

- OpenShift 3.11 の実行中のインスタンスでの管理者権限
- **oc** OpenShift 3.11 CLI 管理ツールのインストール。「Installing the OpenShift 3.11 CLI」を参照してください。
- **crwctl** 管理ツールのインストール。「crwctl CLI 管理ツールのインストール」 を参照してください。
- 主な crwctl コマンドラインパラメーターが設定できない設定を適用するには、Operator で使用される **CheCluster** カスタムリソースのデフォルト値を上書きする設定ファイル **operator-cr-patch.yaml** を準備します。2章*CodeReady Workspaces インストールの設定* を参照してください。
- **<namespace>** はターゲットインストールのプロジェクトを表します。

手順

1. OpenShift にログインします。「Basic Setup and Login」を参照してください。

```
$ oc login
```

2. 以下のコマンドを実行して、**oc** OpenShift CLI 管理ツールのバージョンが 3.11 であることを確認します。

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

3. 以下のコマンドを実行して、CodeReady Workspaces インスタンスを作成します。
   - ユーザー定義の **<namespace> で以下を行い** ます。

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

   - openshift-workspaces というデフォルトプロジェクトで以下を実行します。

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

検証手順

1. 上記のコマンドの出力は以下で終わります。

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

2. CodeReady Workspaces クラスターインスタンス(**https://codeready-<openshift_deployment_name>.<domain_name>**)に移動します。

以前の CodeReady Workspaces インストールからのアップグレード

- 同じ OpenShift Container Platform 3.11 クラスターで以前の CodeReady Workspaces インストールからアップグレードするには、カスタムリソース定義およびクラスターロールを削除します。

```
$ oc delete customresourcedefinition/checlusters.org.eclipse.che
$ oc patch customresourcedefinition/checlusters.org.eclipse.che \
```

```
    --type merge \
    -p '{ "metadata": { "finalizers": null }}'
$ oc delete clusterrole codeready-operator
```

複数の CodeReady Workspaces デプロイメントがある

- 同じ OpenShift Container Platform 3.11 クラスターで異なるバージョンを使用して複数の CodeReady Workspaces デプロイメントを並行して行うには、新規デプロイメント用に新規サービスアカウントを作成します。ただし、このバージョンの組み合わせにより予期しない結果が発生し、サポートされていない結果があるため、古い CodeReady Workspaces デプロイメントをすべて最新バージョンにアップグレードすることが強く推奨されます。

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

3.3. 制限された環境での CODEREADY WORKSPACES のインストール

デフォルトでは、Red Hat CodeReady Workspaces は、パブリックレジストリーで利用可能なコンテナーイメージを主として、各種の外部リソースを使用します。

これらの外部リソースが利用できない環境に CodeReady Workspaces をデプロイするには (例: 公開インターネットに公開されていないクラスターなど)、以下を実行します。

1. OpenShift クラスターによって使用されるイメージレジストリーを特定し、これにプッシュできることを確認します。
2. CodeReady Workspaces の実行に必要なすべてのイメージをこのレジストリーにプッシュします。
3. レジストリーにプッシュされたイメージを使用するように CodeReady Workspaces を設定します。
4. CodeReady Workspaces インストールに進みます。

制限された環境で CodeReady Workspaces をインストールする手順は、使用するインストール方法によって異なります。

- Openshift 4.3 以降での OperatorHub を使用したインストール
- OpenShift 3.11 または 4.x の両方で crwctl 管理ツールを使用したインストール

制限された環境でのネットワーク接続に関する注

ネットワークが制限された環境は、クラウドプロバイダーのプライベートサブネットから、公開インターネットから切断された会社が所有する別個のネットワークに制限されます。ネットワーク設定に関係なく、CodeReady Workspaces は、**CodeReady Workspaces コンポーネント (codeready-workspaces-server、アイデンティティープロバイダー、devfile、およびプラグインレジストリー) 用に作成されたルートが OpenShift クラスター内からアクセスできる場合** に機能します。

環境のネットワークトポロジーを考慮して、これを実行する最も良い方法を判断します。たとえば、会社または組織が所有するネットワークでは、ネットワーク管理者は、クラスターからのトラフィックがルートホスト名にルーティングできるようにする必要があります。たとえば、AWS で、トラフィックがノードを出て、外部に表示されるロードバランサーに到達できるようにプロキシー設定を作成します。

ネットワークが制限された環境にプロキシーが必要な場合は、「プロキシーの後ろでインストールするための CodeReady Workspaces カスタムリソースの準備」 に記載の手順に従います。

3.3.1. OperatorHub を使用した制限された環境での CodeReady Workspaces のインストール

前提条件

- 実行中の OpenShift クラスター。OpenShift クラスターをネットワークが制限された環境にインストールする方法については、、OpenShift Container Platform 4.3 ドキュメント を参照してください。
- ネットワークが制限された環境でインストールされた OpenShift の非接続クラスターに対して使用されるミラーレジストリーへのアクセス。ネットワークが制限された環境でのインストール用のミラーレジストリーの作成についての関連する OpenShift Container Platform 4.3 ドキュメントを参照してください。

ネットワークが制限された環境で実行される非接続 OpenShift 4 クラスターでは、Operator が ネットワークが制限された環境についての Operator の有効化について定義された追加要件を満たす場合にのみ、Operator を OperatorHub からインストールできます。

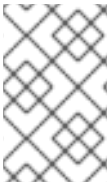
CodeReady Workspaces Operator はこれらの要件を満たしているので、ネットワークが制限された環境での OLM に関する公式ドキュメントに記載の内容と互換性があります。

手順

OperatorHub から CodeReady Workspaces をインストールするには、以下を実行します。

1. **redhat-operators** カタログイメージをビルドします。「Building an Operator catalog image 」を参照してください。
2. OperatorHub を、Operator のインストールにこのカタログイメージを使用するように設定します。「Configuring OperatorHub for restricted networks 」を参照してください。
3. 「OperatorHub を使用した OpenShift 4 への CodeReady Workspaces のインストール」 の説明に従って、通常通りに CodeReady Workspaces のインストールに進みます。

3.3.2. CLI 管理ツールを使用した制限された環境での CodeReady Workspaces のインストール



注記

OperatorHub を使用したインストールが利用できない場合、CodeReady Workspaces CLI 管理ツールを使用して制限されたネットワークに CodeReady Workspaces をインストールします。この方法は OpenShift Container Platform 3.11 でサポートされます。

前提条件

- 実行中の OpenShift クラスター。OpenShift クラスターのインストール方法に関する詳細は、OpenShift Container Platform 3.11 のドキュメント を参照してください。

3.3.2.1. プライベートレジストリーの準備

前提条件

- **oc** ツールが利用可能である。
- **skopeo** ツール (バージョン 0.1.40 以降) が利用できる。
- **podman** ツールが利用できる。
- OpenShift クラスターからアクセスできるイメージ、および V2 イメージマニフェスト (スキーマバージョン 2) フォーマットのサポート。インターネットへのアクセスが一時的に可能な場所から、これにプッシュできることを確認します。

表3.1 サンプルで使用されるプレースホルダー

<source-image>	レジストリー、組織、およびダイジェストなどのソースイメージの詳細な組み合わせ (coordinate)。
<target-registry>	ターゲットコンテナーイメージレジストリーのホスト名およびポート。
<target-organization>	ターゲットのコンテナーイメージレジストリー内の組織
<target-image>	ターゲットのコンテナーイメージレジストリーのイメージ名とダイジェスト。
<target-user>	ターゲットのコンテナーイメージレジストリーのユーザー名。
<target-password>	ターゲットのコンテナーイメージレジストリーのユーザーパスワード。

手順

1. 内部イメージレジストリーにログインします。

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

ヒント

内部レジストリーへのプッシュを試行する際に **x509: certificate signed by unknown authority** などのエラーが発生した場合には、以下のいずれかの回避策を試してください。

- OpenShift クラスターの証明書を **/etc/containers/certs.d/<target-registry>** に追加する。
- **/etc/containers/registries.conf** にある Podman 設定ファイルに以下の行を追加して、レジストリーを非セキュアなレジストリーとして追加する。

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

2. ダイジェストを変更せずにイメージをコピーします。以下の表のすべてのイメージに対して、この手順を繰り返します。

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

注記

表3.2 名前に含まれるプレフィックスまたはキーワードからの container-images の使用について

使用	プレフィックスまたはキーワード
Essential	**stacks-**、**plugin-** ではない。 **-openj-**
Workspaces	**stacks-**、 **plugin-**
z および Power	**-openj-**

表3.3 プライベートレジストリーでコピーするイメージ

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

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

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

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

3. イメージに同じダイジェストがあることを確認します。

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

4. ダイジェストは、異なる場合に明示的に設定します。

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

関連情報

- イメージ一覧のソースを見つけるには、CodeReady Workspaces Operator ClusterServiceVersion ソースの **relatedImages** 属性の値を参照してください。

3.3.2.2. 制限された環境用の CodeReady Workspaces カスタムリソースの準備

crwctl または OperatorHub を使用して制限された環境で CodeReady Workspaces をインストールする場合は、**CheCluster** カスタムリソースに追加の情報を提供します。

3.3.2.2.1. デフォルトの CheCluster カスタムリソースのダウンロード

手順

1. デフォルトのカスタムリソース YAML ファイルをダウンロードし ます。
2. ダウンロードしたカスタムリソース **org_v1_che_cr.yaml** に名前を付けます。追加の変更および使用に備えてこれを保持します。

3.3.2.2.2. 制限された環境用の CheCluster カスタムリソースのカスタマイズ

前提条件

- CodeReady Workspaces がデプロイされる OpenShift クラスターに表示されるイメージレジストリーの利用可能な必要なすべてのイメージ。これについては、「プライベートレジストリーの準備」で説明されています。ここでは、以下の例で使用されているプレースホルダーも定義されています。

手順

1. CodeReady Workspaces Operator によって管理される **CheCluster** カスタムリソースで、制限された環境で CodeReady Workspaces のインスタンスのデプロイを容易にするために使用されるフィールドを追加します。

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

3.3.2.3. CodeReady Workspaces CLI 管理ツールを使用した制限された環境での CodeReady Workspaces インストールの開始

本セクションでは、CodeReady Workspaces CLI 管理ツールを使用して、制限された環境で CodeReady Workspaces インストールを開始する方法を説明します。

前提条件

- CodeReady Workspaces CLI 管理ツールがインストールされている。「crwctl CLI 管理ツールのインストール」 を参照してください。
- **oc** ツールがインストールされている。

- OpenShift インスタンスへのアクセス。

手順

1. OpenShift Container Platform にログインします。

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

2. カスタマイズされたカスタムリソースで CodeReady Workspaces をインストールし、制限された環境に関連するフィールドを追加します。

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

注記

低速なシステムまたはインターネット接続の場合は、**--k8spodwaittimeout=1800000** オプションを **crwctl server:start** コマンドに追加し、Pod タイムアウト期間を 1800000 ms 以上に拡張します。

3.3.3. プロキシーの後ろでインストールするための CodeReady Workspaces カスタムリソースの準備

この手順では、CodeReady Workspaces をプロキシーの後ろでインストールする際に、**CheCluster** カスタムリソースに必要な追加情報を提供する方法を説明します。

手順

1. CodeReady Workspaces Operator によって管理される **CheCluster** カスタムリソースで、制限された環境で CodeReady Workspaces のインスタンスのデプロイを容易にするために使用されるフィールドを追加します。

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

2. これらの基本設定のほかに、プロキシー設定では通常、プロキシーを使用せずに CodeReady Workspaces からアクセスされるホストの一覧に外部 OpenShift クラスター API URL のホストを追加する必要があります。
このクラスター API ホストを取得するには、OpenShift クラスターに対して以下のコマンドを実行します。

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

CheCluster カスタムリソースの対応するフィールドは **nonProxyHosts** です。ホストがこのフィールドにすでに存在する場合は、| を区切り文字として使用し、クラスター API ホストを追加します。

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

第4章 CODEREADY WORKSPACES の設定

本章では、いくつかのユーザーストーリーを例として使用し、Red Hat CodeReady Workspaces の設定方法とオプションについて説明します。

- 「CodeReady Workspaces サーバーコンポーネントの詳細な設定オプション」 では、前述の方法を適用できない場合に使用する詳細な設定方法を説明します。

次のセクションでは、特定のユーザーストーリーを説明します。

- 「プロジェクトストラテジーの設定」
- 「一度に複数のワークスペースの実行」
- 「ワークスペース nodeSelector の設定」
- 「Red Hat CodeReady Workspaces サーバーのホスト名の設定」
- 「自己署名証明書を使用した Git リポジトリーをサポートする CodeReady Workspaces のデプロイ」
- 「ストレージクラスを使用した CodeReady Workspaces のインストール」
- 「ストレージタイプの設定」
- 「TLS 証明書の CodeReady Workspaces サーバーの Java トラストストアへのインポート」

4.1. CODEREADY WORKSPACES サーバーコンポーネントの詳細な設定オプション

以下のセクションでは、CodeReady Workspaces サーバーコンポーネントの詳細なデプロイメントおよび設定方法を説明します。

4.1.1. Operator を使用した CodeReady Workspaces サーバーの詳細設定について

以下のセクションでは、Operator を使用したデプロイメントの CodeReady Workspaces サーバーコンポーネントの詳細な設定方法について説明します。

詳細設定は以下を実行するために必要です。

- 標準の **CheCluster** カスタムリソースフィールドから Operator によって自動的に生成されない環境変数を追加します。
- 標準の **CheCluster** カスタムリソースフィールドから Operator によって自動的に生成されるプロパティーを上書きします。

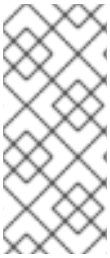
CheCluster カスタムリソースの **server** 設定の一部である **customCheProperties** フィールドには、CodeReady Workspaces サーバーコンポーネントに適用する追加の環境変数のマップが含まれます。

例4.1 ワークスペースのデフォルトのメモリー制限の上書き

- **CHE_WORKSPACE_DEFAULT__MEMORY__LIMIT__MB** プロパティーを **customCheProperties** に追加します。

```
apiVersion: org.eclipse.che/v1
```

```
kind: CheCluster
metadata:
  name: codeready-workspaces
  namespace: <openshift-workspaces>
spec:
  server:
    cheImageTag: ''
    devfileRegistryImage: ''
    pluginRegistryImage: ''
    tlsSupport: true
    selfSignedCert: false
    customCheProperties:
      CHE_WORKSPACE_DEFAULTMEMORYLIMIT__MB: "2048"
  auth:
# [...]
```



注記

以前のバージョンの CodeReady Workspaces Operator には、このロールを実行するために **custom** という名前の configMap が含まれていました。CodeReady Workspaces Operator が **custom** という名前の **configMap** を見つけると、これに含まれるデータを **customCheProperties** フィールドに追加し、CodeReady Workspaces を再デプロイし、**custom configMap** を削除します。

関連情報

- **CheCluster** カスタムリソースで利用可能なすべてのパラメーターの一覧については、2 章*CodeReady Workspaces インストールの設定* を参照してください。
- **customCheProperties** の設定に使用できるすべてのパラメーターの一覧については、「CodeReady Workspaces サーバーコンポーネントのシステムプロパティー参照」 を参照してください。

4.1.2. CodeReady Workspaces サーバーコンポーネントのシステムプロパティー参照

以下のドキュメントでは、CodeReady Workspaces サーバーコンポーネントのすべての使用可能な設定プロパティーについて説明します。

表4.1 Che サーバー

環境変数名	デフォルト値	説明
CHE_DATABASE	**${che.home}/storage**	CodeReady Workspaces が内部データオブジェクトを保存するフォルダー
CHE_API	**http://${CHE_HOST}:${CHE_PORT}/api**	API サービス。ブラウザーがこの URL を使用した CodeReady Workspaces サーバーへの REST 通信の開始

環境変数名	デフォルト値	説明
CHE_WEBSOCKET_ENDPOINT	**ws://${CHE_HOST}:${CHE_PORT}/api/websocket**	CodeReady Workspaces websocket の主なエンドポイント。主な websocket 対話/メッセージング用の基本的な通信エンドポイントを提供します。
CHE_WEBSOCKET_ENDPOINT__MINOR	**ws://${CHE_HOST}:${CHE_PORT}/api/websocket-minor**	CodeReady Workspaces websocket のマイナーエンドポイント。マイナー websocket の対話/メッセージング用の基本的な通信エンドポイントを提供します。
CHE_WORKSPACE_STORAGE	**${che.home}/workspaces**	プロジェクトは、CodeReady Workspaces サーバーから各ワークスペースを実行するマシンに同期されます。これは、プロジェクトがマウントされる ws ランタイム内のディレクトリーです。
CHE_WORKSPACE_PROJECTS_STORAGE	**/projects**	プロジェクトは、CodeReady Workspaces サーバーから各ワークスペースを実行するマシンに同期されます。これは、プロジェクトが配置されているマシンのディレクトリーです。
CHE_WORKSPACE_PROJECTS_STORAGE_DEFAULT_SIZE	**1Gi**	devfile OpenShift/os タイプのコンポーネントがプロジェクト PVC 作成を要求する場合に使用されます（一意および perWorkspace PVC ストラテジーの場合に適用されます）。common PVC ストラテジーの場合、これは che.infra.kubernetes.pvc.quantity プロパティーの値で書き換えられます）。
CHE_WORKSPACE_LOGS_ROOT__DIR	**/workspace_logs**	すべてのワークスペースログが置かれるマシン内のディレクトリーを定義します。エージェント開発者がバックアップエージェントログ用にこのディレクトリーを使用できるように、このフォルダーの値は環境変数などのマシンに指定する必要があります。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_HTTP__PROXY		ワークスペースを駆動するランタイムで使用されるプロキシーを設定します。
CHE_WORKSPACE_HTTPS__PROXY		ワークスペースを駆動するランタイムで使用されるプロキシーの設定
CHE_WORKSPACE_NO__PROXY		ワークスペースを駆動するランタイムで使用されるプロキシーの設定
CHE_TRUSTED__CA__BUNDLES__CONFIGMAP	**NULL**	クラスター全体のプロキシーが設定されると、che-operator は特別な configmap を作成し、OpenShift Network Operator が ca-bundle をこれに挿入することを可能にします。さらに、この configmap という名前のキー CHE_TRUSTED**CA**BUNDLES__CONFIGMAP を CodeReady Workspaces サーバー configmap（および対応する ENV 変数）に追加します。そのため、プロキシーモードが有効かどうかを検出できます。このプロパティーは、特に必要でない限り、手動で設定する必要はありません。
CHE_WORKSPACE_AUTO__START	**true**	デフォルトで、ユーザーが URL を使用してワークスペースにアクセスすると、停止するとワークスペースが自動的に起動します。これを無効にするには、false に設定します。
CHE_WORKSPACE_POOL_TYPE	**fixed**	ワークスペーススレッドプールの設定。このプールは非同期の実行を必要とするワークスペース関連操作に使用されます（例： 起動/停止）。使用できる値は 'fixed'、'cached' です。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_POOL_EXACT__SIZE	**30**	プールタイプが 'fixed' と異なる場合に、このプロパティーは無視されます。乗数プロパティーが設定されている場合は、プールの正確なサイズを設定します。このプロパティーが設定されていない場合 (0, < 0 NULL)、プールサイズがコア数に入っている場合は、乗数内で変更できます。
CHE_WORKSPACE_POOL_CORES__MULTIPLIER	**2**	プールタイプが 'fixed' と異なる場合、または正確なプールサイズが設定されている場合は無視されます。プールサイズが N_CORES * 乗数になる
CHE_WORKSPACE_PROBE__POOL__SIZE	**10**	このプロパティーは、ワークスペースサーバーの liveness プローブに使用するスレッドの数を指定します。
CHE_WORKSPACE_HTTP__PROXY__JAVA__OPTIONS	**NULL**	ワークスペース JVM の HTTP プロキシー設定
CHE_WORKSPACE_JAVA__OPTIONS	**-XX:MaxRAM=150m-XX:MaxRAMFraction=2 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip.disableMemoryMapping=true -Xms20m -Djava.security.egd=file:/dev/./urandom**	ワークスペース内で実行される JVM に追加する Java コマンドラインオプション。
CHE_WORKSPACE_MAVEN__OPTIONS	**-XX:MaxRAM=150m-XX:MaxRAMFraction=2 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip.disableMemoryMapping=true -Xms20m -Djava.security.egd=file:/dev/./urandom**	ワークスペース内でエージェントを実行する JVM に追加される Maven コマンドラインオプション。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_MAVEN__SERVER__JAVA__OPTIONS	**-XX:MaxRAM=128m- XX:MaxRAMFraction=1 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip.disableMemoryMapping=true -Xms20m -Djava.security.egd=file:/dev/./urandom**	Maven サーバーを実行する JVM に追加するデフォルトの java コマンドラインオプション。
CHE_WORKSPACE_DEFAULT__MEMORY__LIMIT__MB	**1024**	環境内に RAM 設定のない各マシンの RAM 制限のデフォルト。0 以下の値は、制限を無効にするものとして解釈されます。
CHE_WORKSPACE_DEFAULT__MEMORY__REQUEST__MB	**200**	環境内に明示的な RAM 設定のない各コンテナーの RAM 要求のデフォルト。この量は、このプロパティーをすべてのインフラストラクチャー実装でサポートされない可能性があります。このプロパティーは OpenShift および OpenShift Container Platform によってサポートされない可能性があります。これは現時点で OpenShift および OpenShift Container Platform によってサポートされない可能性があります。これは、デフォルトのメモリー要求がメモリー制限を超えており、制限のみが使用されます。値が 0 未満または等しい値は無効のリクエストとして解釈されます。
CHE_WORKSPACE_DEFAULT__CPU__LIMIT__CORES	**-1**	環境内に CPU 設定のない各コンテナーの CPU 制限のデフォルト。浮動小数点のコア数（例：0.125）または K8S 形式の整数ミリコア（例： 125m Value less または 0）で指定できます。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_DEFAULT__CPU__REQUEST__CORES	**-1**	環境内に CPU 設定のない各コンテナーの CPU 要求のデフォルト。デフォルトの CPU 要求が CPU 制限を超えた場合、要求は無視され、制限のみが使用されます。0 以下の値は、このリクエストを無効にするものとして解釈されます。
CHE_WORKSPACE_SIDECAR_DEFAULT__MEMORY__LIMIT__MB	**128**	CodeReady Workspaces プラグイン設定に RAM 設定のない各サイドカーの RAM 制限および要求のデフォルト。0 以下の値は、制限を無効にするものとして解釈されます。
CHE_WORKSPACE_SIDECAR_DEFAULT__MEMORY__REQUEST__MB	**64**	**{prod-short}** プラグイン設定に RAM 設定のない各サイドカーの RAM 制限および要求のデフォルト。0 以下の値は、制限を無効にするものとして解釈されます。
CHE_WORKSPACE_SIDECAR_DEFAULT__CPU__LIMIT__CORES	**-1**	CodeReady Workspaces プラグイン設定に CPU 設定のない各サイドカーの CPU 制限および要求のデフォルト。浮動小数点のコア数（例： 0.125）または K8S 形式の整数ミリコア（例： 125m Value less または 0）で指定することが可能です。
CHE_WORKSPACE_SIDECAR_DEFAULT__CPU__REQUEST__CORES	**-1**	**{prod-short}** プラグイン設定に CPU 設定のない各サイドカーの CPU 制限および要求のデフォルト。浮動小数点のコア数（例： 0.125）または K8S 形式の整数ミリコア（例： 125m Value less または 0）で指定することが可能です。
CHE_WORKSPACE_SIDECAR_IMAGE__PULL__POLICY	**Always**	サイドカーのイメージプルストラテジーを定義します。使用できる値は Always、Never、IfNotPresent です。その他の値は未指定のポリシー（:latest タグが指定されている場合は Always）として解釈されます。そうでない場合は IfNotPresent となります。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_ACTIVITY__CHECK__SCHEDULER__PERIOD__S	**60**	非アクティブなワークスペースの一時停止ジョブの実行期間。
CHE_WORKSPACE_ACTIVITY__CLEANUP__SCHEDULER__PERIOD__S	**3600**	アクティビティーテーブルのクリーンアップ期間。アクティビティーテーブルには、サーバーが特定の時点でクラッシュするなどの予想されないエラーが生じる場合に、無効なデータまたは古いデータを含まれることがあります。デフォルトでは、クリーンアップジョブは1時間ごとに実行されます。
CHE_WORKSPACE_ACTIVITY__CLEANUP__SCHEDULER__INITIAL__DELAY__S	**60**	サーバーの起動後から最初のアクティビティーのクリーンアップジョブを開始するまでの遅延。
CHE_WORKSPACE_ACTIVITY__CHECK__SCHEDULER__DELAY__S	**180**	ws マスターが非アクティブのタイムアウトまでの期間利用できない場合の、大規模な一時停止を回避するために最初のワークスペースのアイドルチェックジョブが開始されるまでの遅延。
CHE_WORKSPACE_CLEANUP__TEMPORARY__INITIAL__DELAY__MIN	**5**	停止した一時的なワークスペースのクリーンアップジョブの実行期間。
CHE_WORKSPACE_CLEANUP__TEMPORARY__PERIOD__MIN	**180**	停止した一時的なワークスペースのクリーンアップジョブの実行期間。
CHE_WORKSPACE_SERVER_PING__SUCCESS__THRESHOLD	**1**	サーバーへの正常に順次実行される ping の数。この数を超えると、サーバーは利用可能な状態にあるものとして処理されます。注: このプロパティーはすべてのサーバー (ワークスペースのエージェント、ターミナル、exec など) に共通です。
CHE_WORKSPACE_SERVER_PING__INTERVAL__MILLISECONDS	**3000**	ワークスペースサーバーへの連続する ping の間隔 (ミリ秒単位)。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_SERVER_LIVENESS__PROBES	**wsagent/http,exec-agent/http,terminal,theia,jupyter,dirigible,cloud-shell**	liveness プローブを必要とするサーバー名の一覧
CHE_WORKSPACE_STARTUP__DEBUG__LOG__LIMIT__BYTES	**10485760**	ワークスペースの起動をデバッグする際に che-server で観察される単一コンテナーから収集されるログの制限サイズ。デフォルト値は 10MB=10485760 です。
CHE_WORKSPACE_STOP_ROLE_ENABLED	**true**	true の場合、OpenShift OAuth が有効な場合に、編集権限を持つ「stop-workspace」ロールが「che」 ServiceAccount に付与されます。この設定は、OpenShift OAuth が有効な場合にワークスペースのアイドリングに主に必要になります。

表4.2 templates

環境変数名	デフォルト値	説明
CHE_TEMPLATE_STORAGE	**${che.home}/templates**	コードテンプレートおよびサンプルが含まれる JSON ファイルが含まれるフォルダー

表4.3 認証パラメーター

環境変数名	デフォルト値	説明
CHE_AUTH_USER__SELF__CREATION	**false**	CodeReady Workspaces には単一のアイデンティティー実装があるため、これによるユーザーエクスペリエンスへの変更はありません。true の場合、API レベルでのユーザー作成を有効にします。
CHE_AUTH_ACCESS__DENIED__ERROR__PAGE	**/error-oauth**	認証エラーページアドレス
CHE_AUTH_RESERVED__USER__NAMES		予約済みのユーザー名

環境変数名	デフォルト値	説明
CHE_OAUTH_GITHUB_CLIENTID	**NULL**	GitHub OAuth を設定して、リモートリポジトリーへの認証を自動化できます。最初に、このアプリケーションを GitHub OAuth に登録する必要があります。
CHE_OAUTH_GITHUB_CLIENTSECRET	**NULL**	GitHub OAuth を設定して、リモートリポジトリーへの認証を自動化できます。最初に、このアプリケーションを GitHub OAuth に登録する必要があります。
CHE_OAUTH_GITHUB_AUTHURI	**https://github.com/login/oauth/authorize**	GitHub OAuth を設定して、リモートリポジトリーへの認証を自動化できます。最初に、このアプリケーションを GitHub OAuth に登録する必要があります。
CHE_OAUTH_GITHUB_TOKENURI	**https://github.com/login/oauth/access_token**	GitHub OAuth を設定して、リモートリポジトリーへの認証を自動化できます。最初に、このアプリケーションを GitHub OAuth に登録する必要があります。
CHE_OAUTH_GITHUB_REDIRECTURIS	**http://localhost:${CHE_PORT}/api/oauth/callback**	GitHub OAuth を設定して、リモートリポジトリーへの認証を自動化できます。最初に、このアプリケーションを GitHub OAuth に登録する必要があります。
CHE_OAUTH_OPENSHIFT_CLIENTID	**NULL**	OpenShift OAuth クライアントの設定。OpenShift OAuth トークンの取得に使用されます。
CHE_OAUTH_OPENSHIFT_CLIENTSECRET	**NULL**	OpenShift OAuth クライアントの設定。OpenShift OAuth トークンの取得に使用されます。
CHE_OAUTH_OPENSHIFT_OAUTH__ENDPOINT	**NULL**	Configurationof OpenShift OAuth クライアント。OpenShift OAuth トークンの取得に使用されます。
CHE_OAUTH_OPENSHIFT_VERIFY__TOKEN__URL	**NULL**	ConfigurationofOpenShiftOAuth クライアント。OpenShift OAuth トークンの取得に使用されます。

表4.4 内部

環境変数名	デフォルト値	説明
SCHEDULE_CORE__POOL__SIZE	**10**	CodeReady Workspaces 拡張には、時間ベースでスケジュールされる実行をスケジュールできます。これにより、繰り返されるスケジュールで起動する拡張に割り当てられるスレッドプールのサイズが設定されます。
ORG_EVERREST_ASYNCHRONOUS	**false**	Everrest は、JAX-RS および Web ソケット通信ユーザーを管理する Java Web Services ツールキットで、これを設定する必要はほとんどありません。everrest に組み込まれている非同期メカニズムを無効にします。
ORG_EVERREST_ASYNCHRONOUS_POOL_SIZE	**20**	同時に処理できる非同期リクエストの数
ORG_EVERREST_ASYNCHRONOUS_QUEUE_SIZE	**500**	キューのサイズ。消費後、非同期リクエストを処理できない場合、キューに追加されます。
ORG_EVERREST_ASYNCHRONOUS_JOB_TIMEOUT	**10**	要求のタイムアウト（分単位）。タイムアウトリクエストが完了していないか、クライアントが要求の結果をまだ取得していないと、破棄される可能性があります。
ORG_EVERREST_ASYNCHRONOUS_CACHE_SIZE	**1024**	要求待ち、実行、および終了するためのキャッシュのサイズ。
ORG_EVERREST_ASYNCHRONOUS_SERVICE_PATH	**/async/**	非同期サービスへのパス
DB_SCHEMA_FLYWAY_BASELINE_ENABLED	**true**	DB の初期化および移行設定
DB_SCHEMA_FLYWAY_BASELINE_VERSION	**5.0.0.8.1**	DB の初期化および移行設定
DB_SCHEMA_FLYWAY_SCRIPTS_PREFIX		DB の初期化および移行設定
DB_SCHEMA_FLYWAY_SCRIPTS_SUFFIX	**.sql**	DB の初期化および移行設定

環境変数名	デフォルト値	説明
DB_SCHEMA_FLYWAY_SCRIPTS_VERSION__SEPARATOR	**__**	DB の初期化および移行設定
DB_SCHEMA_FLYWAY_SCRIPTS_LOCATIONS	**classpath:che-schema**	DB の初期化および移行設定

表4.5 OpenShift インフラパラメーター

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_MASTER__URL		インフラが使用する OpenShift クライアントの設定
CHE_INFRA_KUBERNETES_TRUST__CERTS		インフラが使用する OpenShift クライアントの設定
CHE_INFRA_KUBERNETES_SERVER__STRATEGY	**default-host**	サーバーが OpenShift インフラでグローバルに公開される方法を定義します。CodeReady Workspaces に実装されたストラテジーの一覧: default-host、multi-host、single-host
CHE_INFRA_KUBERNETES_SINGLE__HOST_WORKSPACE_EXPOSURE	**native**	ワークスペースのプラグインとエディターを単一ホストモードで公開する方法を定義します。サポートされる公開: - 'native': OpenShift Ingress を使用してサーバーを公開します。OpenShift でのみ動作します。
CHE_INFRA_KUBERNETES_INGRESS_DOMAIN		プロパティー **che.infra.kubernetes.server_strategy** が **multi-host** に設定されている場合に、ワークスペースでサーバーのドメインを生成するために使用されます。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_NAMESPACE		非推奨: このプロパティーの値は変更しないでください。変更すると、既存のワークスペースでデータが失われます。これは新規インストールに設定しないでください。すべてのワークスペースが作成される OpenShift namespace を定義します。これが設定されないと、すべてのワークスペースが新しい namespace に作成されます。ここで、namespace = ワークスペース ID になります。<username> および <userid> プレースホルダー (例: che-workspace-<username>) を使用できます。この場合、ユーザーごとに新規 namespace が作成されます。新規 namespace を作成するパーミッションを持つサービスアカウントを使用する必要があります。OpenShift インフラでは無視されます。**che.infra.openshift.project** を代わりに使用します。このプロパティーで参照される namespace が存在する場合、これはすべてのワークスペースで使用されます。これが存在しない場合、che.infra.kubernetes.namespace.default によって指定される namespace が作成され、使用されます。
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT	**<username>-che**	ユーザーが上書きしない場合に、ユーザーのワークスペースが作成される OpenShift default namespace を定義します。<username>、<userid> および <workspaceid> プレースホルダー (例: che-workspace-<username>）を使用できます。この場合、新規 namespace が各ユーザー (またはワークスペース) について作成されます。OpenShift インフラによってプロジェクトの指定にも使用されます。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_NAMESPACE_ALLOW__USER__DEFINED	**false**	ユーザーがデフォルトとは異なる OpenShift namespace（または OpenShift プロジェクト）を指定できるかどうかを定義します。OAuth が設定されていない場合に true に設定することは推奨されません。このプロパティーは OpenShift infra によっても使用されます。
CHE_INFRA_KUBERNETES_SERVICE__ACCOUNT__NAME	**NULL**	すべてのワークスペース Pod にバインドされるように指定する必要のある OpenShift サービスアカウント名を定義します。OpenShift インフラストラクチャーはサービスアカウントを作成しないため、これは存在するはずであることに注意してください。OpenShift インフラストラクチャーは、プロジェクトが事前に定義されているかどうかをチェックします（**che.infra.openshift.project** が空でない場合)。これが事前に定義されている場合はサービスアカウントが存在するはずです。これが 'NULL' または空の文字列の場合、インフラストラクチャーはワークスペースごとに新しい OpenShift プロジェクトを作成し、必要なロールを持つワークスペーのスサービスアカウントをここに準備します。
CHE_INFRA_KUBERNETES_WORKSPACE__SA__CLUSTER__ROLES	**NULL**	ワークスペースのサービスアカウントで使用するオプションの追加クラスターロールを指定します。クラスターのロール名がすでに存在している必要があり、CodeReady Workspaces サービスアカウントはロールバインディングを作成して、これらのクラスターロールをワークスペースのサービスアカウントに関連付ける必要があることに注意してください。名前はコンマで区切られます。このプロパティーは 'che.infra.kubernetes.cluster_role_name' を非推奨にします。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_WORKSPACE__START__TIMEOUT__MIN	**8**	OpenShift ワークスペースの開始時間を制限する時間枠を定義します。
CHE_INFRA_KUBERNETES_INGRESS__START__TIMEOUT__MIN	**5**	OpenShift Ingress が準備状態になる期間を制限するタイムアウトを分単位で定義します。
CHE_INFRA_KUBERNETES_WORKSPACE__UNRECOVERABLE__EVENTS	**FailedMount,FailedScheduling,MountVolume.SetUpfailed, Failed to pull image,FailedCreate**	ワークスペースの起動中に、プロパティーに定義されるリカバリー不可能なイベントが発生する場合、タイムアウトまで待機するのではなく、ワークスペースをすぐに終了します。これには 'Failed' の理由が含めることができないことに注意してください。これにより、リカバリー不可能なイベントがキャッチされる可能性があるためです。失敗したコンテナーの起動は、CodeReady Workspaces サーバーで明示的に処理されます。
CHE_INFRA_KUBERNETES_PVC_ENABLED	**true**	che ワークスペースに Persistent Volume Claim（永続ボリューム要求、PVC）を使用するかどうか (バックアッププロジェクト、ログなどをが必要かどうか) を定義します。またはこれを無効にします。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_PVC_STRATEGY	**common**	ワークスペース用に PVC を選択する際に使用するストラテジーを定義します。サポートされるストラテジー： - 'common' 同じ OpenShift namespace 内のすべてのワークスペースは同じ PVC を再利用します。PVC の名前は 'che.infra.kubernetes.pvc.name' で設定できます。既存の PVC が使用されるか、または新規 PVC が存在しない場合にはこれが作成されます。- 'unique' 各ワークスペースのボリュームに別個の PVC が使用されます。PVC の名前は '{che.infra.kubernetes.pvc.name} + '-' + '{generated_8_chars}' として評価されます。既存の PVC が使用されるか、または新規 PVC が存在しない場合にはこれが作成されます。- 'per-workspace' 各ワークスペースの別個の PVC が使用されます。PVC の名前は '{che.infra.kubernetes.pvc.name} + '-' + '{WORKSPACE_ID}' として評価されます。既存の PVC が使用されるか、または新規 PVC が存在しない場合にはこれが作成されます。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_PVC_PRECREATE__SUBPATHS	**true**	ワークスペースを起動する前に、「common」ストラテジーの永続ボリュームでワークスペースのサブパスディレクトリーを作成するジョブを実行するかどうかを定義します。ワークスペースサブパスボリュームマウントは root 権限で作成され、ユーザーとして実行するワークスペースで変更できないため（CodeReady Workspaces のワークスペースへのプロジェクトのインポートエラーなど）、OpenShift/OpenShift の一部のバージョンで必要です。デフォルトは「true」ですが、Openshift/OpenShift のバージョンがユーザーパーミッションでサブディレクトリーを作成する場合は false に設定する必要があります。関連する問題：https://github.com/kubernetes/kubernetes/issues/41638 このプロパティーは、「common」 PVC ストラテジーが使用される場合にのみ有効であることに注意してください。
CHE_INFRA_KUBERNETES_PVC_NAME	**claim-che-workspace**	che ワークスペースの PVC 名の設定を定義します。それぞれの PVC ストラテジーは、この値を異なる方法で指定します。che.infra.kubernetes.pvc.strategy プロパティーについてドキュメントを参照してください。
CHE_INFRA_KUBERNETES_PVC_STORAGE__CLASS__NAME		ワークスペースの Persistent Volume Claim（永続ボリューム要求、PVC）のストレージクラスを定義します。空の文字列は「use default」を意味します。
CHE_INFRA_KUBERNETES_PVC_QUANTITY	**10Gi**	che workspace の Persistent Volume Claim（永続ボリューム要求、PVC）のサイズを定義します。形式については以下を参照してください：https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE	**centos:centos7**	OpenShift で永続ボリューム要求 (PVC) のメンテナンスジョブを実行する際に起動する Pod
CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE_PULL__POLICY	**IfNotPresent**	OpenShift/OpenShift クラスターのメンテナンスジョブに使用されるコンテナーのイメージプルポリシー
CHE_INFRA_KUBERNETES_PVC_JOBS_MEMORYLIMIT	**250Mi**	永続ボリューム要求のメンテナンスジョブの Pod メモリー制限を定義します。
CHE_INFRA_KUBERNETES_PVC_ACCESS__MODE	**ReadWriteOnce**	Persistent Volume Claim（永続ボリューム要求、PVC）のアクセスモードを定義します。common PVC ストラテジーの変更の場合、アクセスモードの変更は、同時に実行されるワークスペースの数に影響を与えることに注意してください。実行されている che が RWX アクセスモードので PV を使用している OpenShift フレーバーの場合、同時に実行されるワークスペースの制限は (RAM、CPU などの) che 制限の設定によってのみバインドされます。アクセスモードの詳細は、https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html を参照してください。
CHE_INFRA_KUBERNETES_PVC_WAIT__BOUND	**true**	CodeReady Workspaces Server がワークスペース PVC が作成後にバインドされるまで待機するかどうかを定義します。これは、すべての PVC ストラテジーによって使用されます。**volumeBindingMode** が **WaitForFirstConsumer** に設定されている場合は、**false** に設定する必要があります。それ以外の場合は、ワークスペースの起動が PVC の待機フェーズでハングします。デフォルト値は true (PVC がバインドされるまで待機する必要あることを意味します) です。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_INSTALLER__SERVER__MIN__PORT	**10000**	インストーラーサーバーの定義されたポート範囲。デフォルトで、インストーラーは独自のポートを使用しますが、これが別のインストーラーサーバーと競合する場合は、OpenShift インフラストラクチャーはインストーラーを再設定し、この範囲から最初に利用可能になるものを使用できるようにします。
CHE_INFRA_KUBERNETES_INSTALLER__SERVER__MAX__PORT	**20000**	インストーラーサーバーの定義されたポート範囲。デフォルトで、インストーラーは独自のポートを使用しますが、これが別のインストーラーサーバーと競合する場合は、OpenShift インフラストラクチャーはインストーラーを再設定し、この範囲から最初に利用可能になるものを使用できるようにします。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_INGRESS_ANNOTATIONS__JSON	**NULL**	サーバーを公開するために使用される Ingress のアノテーションを定義します。値は Ingress コントローラーの種類によって異なります。OpenShift インフラストラクチャーは ingress の代わりにルートを使用するため、このプロパティーを無視します。単一ホストのデプロイメントストラテジーが機能するには、URL の書き換えをサポートするコントローラーを使用する必要があります（URL が異なるサーバーをポイントでき、サーバーはアプリケーションのルートの変更をサポートする必要がないようにします)。che.infra.kubernetes.ingress.path.rewrite_transform プロパティーは、Ingress のパスが URL の書き換えをサポートするよう変換する方法を定義します。このプロパティーは、選択した Ingress コントローラーに対して実際に URL の書き換えを実行するように指示する ingress 自体のアノテーションのセットを定義します (選択された Ingress コントローラーで必要な場合)。nginx Ingress コントローラー 0.22.0 以降では、この値は推奨されています：**{'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'}** および che.infra.kubernetes.ingress.path.rewrite_transform を、0.22.0 よりも古い nginx Ingress コントローラーの場合は、rewrite-target を merely '/' および '%s' に変換する必要があります（che.infra.kubernetes.ingress.path.rewrite_form を参照してください）。Ingress コントローラーが Ingress パスにある正規表現を使用する方法と、URL の書き換えを実行する方法についての説明は、nginx Ingress コントローラーのドキュメントを参照してください。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_INGRESS_PATH__TRANSFORM		バーを公開する Ingress のパスを宣言する方法についての「recipe」(レシピ) を定義します。'%s' はサーバーのベース公開 URL を表し、スラッシュで終了することが保証されています。このプロパティーは String.format（）メソッドへの有効な入力であり、%s への参照が 1 つだけ含まれる必要があります。Ingress のアノテーションとパスを指定する際にこれら 2 つのプロパティーの相互作用を確認するには、che.infra.kubernetes.ingress.annotations_json プロパティーの説明を参照してください。これが定義されていない場合、このプロパティーはデフォルトで '%s' (引用符なし) に設定されます。これは、パスが Ingress コントローラーで使用する場合に変換されないことを意味します。
CHE_INFRA_KUBERNETES_POD_SECURITY__CONTEXT_RUN__AS__USER	**NULL**	OpenShift インフラによって作成される Pod のセキュリティーコンテキストを定義します。これは OpenShift インフラによって無視されます。
CHE_INFRA_KUBERNETES_POD_SECURITY__CONTEXT_FS__GROUP	**NULL**	OpenShift インフラによって作成される Pod のセキュリティーコンテキストを定義します。これは OpenShift インフラによって無視されます。
CHE_INFRA_KUBERNETES_POD_TERMINATION__GRACE__PERIOD__SEC	**0**	OpenShift / OpenShift インフラストラクチャーによって作成される Pod の猶予期間を定義します。これにより、OpenShift / OpenShift ワークスペースの Pod の正常な終了期間がデフォルト設定されます。これにより、Pod をほぼ即時に終了し、ワークスペースを停止するのに必要な時間を短縮できます。注記：**terminationGracePeriodSeconds** が OpenShift / OpenShift recipe で明示的に設定されている場合、これは上書きされません。

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC__REQUESTS_MAX	**1000**	**OpenShiftClient** インスタンスの基礎となる共有 http クライアントでサポートされる同時の非同期 Web リクエスト（http 要求または継続的な Web ソケット呼び出し）の最大数。デフォルト値は 64 および 1 ホストに 5 になります。これは、CodeReady Workspaces が (コマンドまたは ws-agent ログ用などに) 開かれた接続の数を維持することを考慮すると、マルチユーザーのシナリオでは正しい値のように見えない場合があります。
CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC__REQUESTS_MAX__PER__HOST	**1000**	**OpenShiftClient** インスタンスの基礎となる共有 http クライアントでサポートされる同時の非同期 Web リクエスト（http 要求または継続的な Web ソケット呼び出し）の最大数。デフォルト値は 64 および 1 ホストごとに 5 です。これは、**{prod-short}** が（コマンドまたは ws-agent ログ用など）開かれた接続の数を維持することを把握するマルチユーザーシナリオには適切ではありません。
CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION__POOL_MAX__IDLE	**5**	OpenShift クライアント共有 http クライアントの接続プールにおけるアイドル状態の接続の最大数
CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION__POOL_KEEP__ALIVE__MIN	**5**	OpenShift クライアント共有 http クライアントの接続プールのキープアライブのタイムアウト（分単位）
CHE_INFRA_KUBERNETES_TLS__ENABLED	**false**	OpenShift インフラストラクチャーで TLS (Transport Layer Security) を有効にして Ingress を作成します。ルートは TLS 対応になります。
CHE_INFRA_KUBERNETES_TLS__SECRET		OpenShift インフラストラクチャーによって無視される TLS でワークスペース Ingress を作成する際に使用するシークレットの名前

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_TLS__KEY	**NULL**	ワークスペースの Ingresss 証明書およびキーに使用する必要のある TLS Secret のデータは Base64 アルゴリズムでエンコードされる必要があります。これらのプロパティーは OpenShift インフラストラクチャーで無視されます。
CHE_INFRA_KUBERNETES_TLS__CERT	**NULL**	ワークスペースの Ingresss 証明書およびキーに使用する必要のある TLS Secret のデータは Base64 アルゴリズムでエンコードされる必要があります。これらのプロパティーは OpenShift インフラストラクチャーで無視されます。
CHE_INFRA_KUBERNETES_RUNTIMES__CONSISTENCY__CHECK__PERIOD__MIN	**-1**	ランタイムの整合性チェックが実行される期間を定義します。ランタイムに一貫性のない状態がある場合、ランタイムは自動的に停止します。値は 0 をより大きな値、または **-1** である必要があります。ここで、**-1** はチェックが実行されないことを意味します。これはデフォルトで無効にされます。CodeReady Workspaces Server では操作がユーザーによって呼び出しされない場合に OpenShift API と対話できない場合に CodeReady Workspaces サーバーの設定がある可能性があるためです。これは以下の設定で機能します。- ワークスペースオブジェクトは、CodeReady Workspaces Server がある同じ namespace に作成されます。- cluster-admin サービスアカウントトークンは CodeReady Workspaces Server Pod にマウントされます。これは、以下の設定では機能しません。- CodeReady Workspaces Server は OAuth プロバイダーのトークンを使用して OpenShift API と通信します。

表4.6 OpenShift インフラパラメーター

環境変数名	デフォルト値	説明
CHE_INFRA_OPENSHIFT_PROJECT		非推奨: このプロパティーの値は変更しないでください。変更すると、既存のワークスペースでデータが失われます。これは新規インストールに設定しないでください。すべてのワークスペースが作成される OpenShift namespace を定義します。これが設定されないと、すべてのワークスペースが新しい namespace に作成されます。ここで、プロジェクト名 = ワークスペース ID になります。<username> および <userid> プレースホルダー (例: che-workspace-<username>) を使用できます。この場合、ユーザーごとに新規プロジェクトが作成されます。新規プロジェクトを作成するパーミッションを持つ OpenShift oauth またはサービスアカウントを使用する必要があります。このプロパティーで参照されるプロジェクトが存在する場合、これはすべてのワークスペースで使用されます。これが存在しない場合、che.infra.kubernetes.namespace.default によって指定される namespace が作成され、使用されます。
CHE_INFRA_OPENSHIFT_TRUSTED__CA__BUNDLES__CONFIG__MAP	**ca-certs**	CA バンドルが Openshift 4 に保存される trust-store 設定マップの名前を設定します。このマップは、その名前の CodeReady Workspaces インストーラー（operator など）によって最初に作成され、CodeReady Workspaces サーバーはワークスペースの起動時に特定のラベル（以下を参照）でこれを検出し、ワークスペースの namespace に同じマップを作成し、マウントします。プロパティーは、ワークスペース namespace のマップの名前を定義します。
CHE_INFRA_OPENSHIFT_TRUSTED__CA__BUNDLES__CONFIG__MAP__LABELS	**config.openshift.io/inject-trusted-cabundle=true**	Openshift 4 での証明書の自動挿入に使用される設定マップのラベル名。

環境変数名	デフォルト値	説明
CHE_INFRA_OPENSHIFT_TRUSTED__CA__BUNDLES__MOUNT__PATH	**/public-certs**	CA バンドルがマウントされるワークスペースコンテナーでパスを設定します。
CHE_SINGLEPORT_WILDCARD__DOMAIN_HOST	**NULL**	単一ポートモードワイルドカードドメインホスト & port. nip.io がデフォルトで使用されます
CHE_SINGLEPORT_WILDCARD__DOMAIN_PORT	**NULL**	Singleport mode wildcard domain host & port. nip.io がデフォルトで使用されます
CHE_SINGLEPORT_WILDCARD__DOMAIN_IPLESS	**false**	IP を挿入せずに単一ポートのカスタム DNS を有効にします。

表4.7 実験的なプロパティー

環境変数名	デフォルト値	説明
CHE_WORKSPACE_PLUGIN__BROKER_METADATA_IMAGE	**quay.io/eclipse/che-plugin-metadata-broker:v3.3.0**	ワークスペースツール設定を解決し、プラグインの依存関係をワークスペースにコピーする CodeReady Workspaces プラグインブローカーアプリケーションの Docker イメージ
CHE_WORKSPACE_PLUGIN__BROKER_ARTIFACTS_IMAGE	**quay.io/eclipse/che-plugin-artifacts-broker:v3.3.0**	ワークスペースツール設定を解決し、プラグインの依存関係をワークスペースにコピーする **{prod-short}** プラグインブローカーアプリケーションの DockerImage
CHE_WORKSPACE_PLUGIN__BROKER_PULL__POLICY	**Always**	ワークスペースツール設定を解決し、プラグインの依存関係をワークスペースにコピーする CodeReady Workspaces プラグインブローカーアプリケーションの Docker イメージ
CHE_WORKSPACE_PLUGIN__BROKER_WAIT__TIMEOUT__MIN	**3**	プラグインブローカーの待機中に結果の最大期間を制限するタイムアウトを分単位で定義します。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_PLUGIN__REGISTRY__URL	**https://che-plugin-registry.prod-preview.openshift.io/v3**	ワークスペースツールプラグインレジストリーのエンドポイント。有効な HTTP URL でなければなりません。**Example: http://che-plugin-registry-eclipse-che.192.168.65.2.nip.io** CodeReady Workspaces プラグインツールが不要な場合、値 'NULL' を使用する必要があります。
CHE_WORKSPACE_DEVFILE__REGISTRY__URL	**https://che-devfile-registry.prod-preview.openshift.io/**	devfile レジストリーエンドポイント。有効な HTTP URL でなければなりません。**Example: http://che-devfile-registry-eclipse-che.192.168.65.2.nip.io** CodeReady Workspaces プラグインツールが不要な場合、値 'NULL' を使用する必要があります。
CHE_WORKSPACE_STORAGE_AVAILABLE__TYPES	**persistent,ephemeral,async**	ダッシュボードなどのクライアントがワークスペースの作成/更新時にユーザーに提案するストレージタイプに使用できる値を定義する設定プロパティー。使用できる値: - 'persistent': 永続ストレージの I/O は低速だが永続性がある。- 'ephemeral': 一時ストレージは、高速 I/O を可能にするが、ストレージには制限があり、永続性がない。- 'async': 実験的機能: 非同期ストレージは一時ストレージと永続ストレージの組み合わせ。高速な I/O を可能にし、変更を維持し、停止時にバックアップを実行し、ワークスペースの開始時に復元します。 che.infra.kubernetes.pvc.strategy='common' - che.limits.user.workspaces.run.count=1 - che.infra.kubernetes.namespace.allow_user_defined=false - che.infra.kubernetes.namespace.default に <username> が含まれる場合にのみ機能します。それ他の場合は、一覧から 'async' を削除します。

環境変数名	デフォルト値	説明
CHE_WORKSPACE_STORAGE_PREFERRED__TYPE	**persistent**	ダッシュボードなどのクライアントがワークスペースの作成/更新時にユーザーに提案するストレージタイプのデフォルト値を定義する設定プロパティー。「async」値は実験的な値のため、デフォルトタイプとしての使用は推奨されません。
CHE_SERVER_SECURE__EXPOSER	**default**	セキュアなサーバーが認証で保護される方法を設定します。適切な値: 'default': jwtproxy はパススルーモードで設定されます。そのため、サーバーは要求を認証する必要があります。'jwtproxy': jwtproxy は要求を認証します。そのため、サーバーは認証済みのもののみを受信します。
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_TOKEN_ISSUER	**wsmaster**	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_TOKEN_TTL	**8800h**	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_AUTH_LOADER_PATH	**/_app/loader.html**	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_IMAGE	**quay.io/eclipse/che-jwtproxy:0.10.0**	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_MEMORY__LIMIT	**128mb**	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。
CHE_SERVER_SECURE__EXPOSER_JWTPROXY_CPU__LIMIT	**0.5**	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。

表4.8 主な「/websocket」エンドポイントの設定

環境変数名	デフォルト値	説明
CHE_CORE_JSONRPC_PROCESSOR__MAX__POOL__SIZE	**50**	JSON RPC 処理プールの最大サイズ。プールサイズが超過すると、メッセージの実行が拒否されます。
CHE_CORE_JSONRPC_PROCESSOR__CORE__POOL__SIZE	**5**	初期 json 処理プール。主な JSON RPC メッセージを処理するために使用されるスレッドの最小数。
CHE_CORE_JSONRPC_PROCESSOR__QUEUE__CAPACITY	**100000**	Json RPC メッセージの処理に使用するキューの設定。

表4.9 主な「/websocket-minor」エンドポイントの設定

環境変数名	デフォルト値	説明
CHE_CORE_JSONRPC_MINOR__PROCESSOR__MAX__POOL__SIZE	**100**	JSON RPC 処理プールの最大サイズ。プールサイズが超過すると、メッセージの実行が拒否されます。
CHE_CORE_JSONRPC_MINOR__PROCESSOR__CORE__POOL__SIZE	**15**	初期 json 処理プール。マイナー JSON RPC メッセージを処理するために使用されるスレッドの最小数。
CHE_CORE_JSONRPC_MINOR__PROCESSOR__QUEUE__CAPACITY	**10000**	Json RPC メッセージの処理に使用するキューの設定。
CHE_METRICS_PORT	**8087**	Prometheus メトリクスで公開される http サーバーエンドポイントのポート

表4.10 CORS 設定

環境変数名	デフォルト値	説明
CHE_CORS_ALLOWED__ORIGINS	*	WS Master の CORS フィルターはデフォルトで無効にされます。環境変数 'CHE_CORS_ENABLED=true' を使用して 'cors.allowed.origins' でこれを有効にし、許可される要求元を示唆します。

環境変数名	デフォルト値	説明
CHE_CORS_ALLOW__CREDENTIALS	**false**	'cors.support.credentials' は、認証情報 (cookie、ヘッダー、TLS クライアント証明書) を使用して要求の処理を許可するかどうかを示します。

表4.11 Factory のデフォルト

環境変数名	デフォルト値	説明
CHE_FACTORY_DEFAULT__EDITOR	**eclipse/che-theia/7.18.2**	CodeReady Workspaces 固有のワークスペース記述子が含まれないリモート git リポジトリーから作成されるファクトリーに使用されるエディターおよびプラグイン（.factory.json の .devfile など）複数のプラグインはコンマで区切る必要があります。例：pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion
CHE_FACTORY_DEFAULT__PLUGINS	**eclipse/che-machine-exec-plugin/7.18.2**	**{prod-short}**固有のワークスペース記述子（.factory.json など）複数のプラグインがコンマで区切られていないリモート git リポジトリーから作成されるファクトリーに使用されるエディターおよびプラグイン。例：pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion
CHE_FACTORY_DEFAULT__DEVFILE__FILENAMES	**devfile.yaml,.devfile.yaml**	リポジトリーベースの Factory (GitHub など) を検索する devfile のファイル名。Factory は、プロパティーで列挙される順序でこれらのファイルの特定を試みます。

表4.12 devfile のデフォルト

環境変数名	デフォルト値	説明

環境変数名	デフォルト値	説明
CHE_WORKSPACE_DEVFILE_DEFAULT__EDITOR	**eclipse/che-theia/7.18.2**	指定されていない場合に Devfile にプロビジョニングする必要があるデフォルトのエディター。エディター形式は、**editorPublisher/editorName/editorVersion** 値になります。**NULL** または値がない場合は、デフォルトのエディターはプロビジョニングされません。
CHE_WORKSPACE_DEVFILE_DEFAULT__EDITOR_PLUGINS	**eclipse/che-machine-exec-plugin/7.18.2**	デフォルトのエディター用にプロビジョニングする必要があるデフォルトのプラグイン。ユーザー定義の devfile で明示的に参照されていないこの一覧のすべてのプラグインはプロビジョニングされますが、これはデフォルトのエディターが使用されているか、またはユーザー定義のエディターが (異なるバージョンの場合でも) デフォルトと同じである場合に限ります。形式は、コンマで区切られた **pluginPublisher/pluginName/pluginVersion** の値、および URL です。例: eclipse/che-theia-exec-plugin/0.0.1,eclipse/che-theia-terminal-plugin/0.0.1,https://cdn.pluginregistry.com/vi-mode/meta.yaml プラグインが URL の場合、プラグインの meta.yaml はその URL から取得されます。
CHE_WORKSPACE_PROVISION_SECRET_LABELS	**app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspace-secret**	ユーザー namespace からシークレットを選択するためにラベルのコンマ区切りの一覧を定義します。これは、ファイルまたは環境変数としてワークスペースコンテナーにマウントされます。すべての指定されるラベルに一致するシークレットのみが選択されます。
CHE_WORKSPACE_DEVFILE_ASYNC_STORAGE_PLUGIN	**eclipse/che-async-pv-plugin/nightly**	非同期ストレージ機能がワークスペース設定で有効にされ、環境でサポートされる場合に、プラグインが追加されます

環境変数名	デフォルト値	説明
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_IMAGE	**quay.io/eclipse/che-workspace-data-sync-storage:latest**	CodeReady Workspaces 非同期ストレージの Docker イメージ
CHE_WORKSPACE_POD_NODE__SELECTOR	**NULL**	オプションでワークスペース Pod のノードセレクターを設定します。形式は、コンマ区切りの key=value ペアです (例: disktype=ssd,cpu=xlarge,foo=bar)。
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN__TIMEOUT__MIN	**120**	最後に使用されたワークスペースの停止後の非同期ストレージ Pod のシャットダウンのタイムアウト。0 以下の値は、シャットダウン機能を無効にするものとして解釈されます。
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN__CHECK__PERIOD__MIN	**30#**	非同期ストレージ Pod が機能を停止する期間を定義します (デフォルトでは 30 分ごと)。

表4.13 Che システム

環境変数名	デフォルト値	説明
CHE_SYSTEM_SUPER__PRIVILEGED__MODE	**false**	System Super Privileged Mode (システムのスーパー特権モード)。getByKey、getByNameSpace、stopWorkspaces、および getResources の manageSystem パーミッションの追加パーミッションをユーザーに付与します。これらは、デフォルトでは管理者には提供されず、これらのパーミッションにより、管理者は admin 権限でそれらのワークスペースに名前を指定し、ワークスペースへの可視性を得ることができます。

環境変数名	デフォルト値	説明
CHE_SYSTEM_ADMIN__NAME	**admin**	'che.admin.name' ユーザーのシステムパーミッションを付与します。ユーザーがすでに存在する場合は、これはコンポーネントの起動時に生じます。ユーザーがすでに存在しない場合は、ユーザーがデータベースで永続化される初回のログイン時に発生します。

表4.14 Workspace の制限

環境変数名	デフォルト値	説明
CHE_LIMITS_WORKSPACE_ENV_RAM	**16gb**	ワークスペースは、開発を行う際のユーザー向けの基本的なランタイムです。ワークスペースの作成方法や、消費されるリソースを制限するパラメーターを設定できます。ユーザーが新規ワークスペースの作成時にワークスペースに割り当てることができる RAM の最大量。RAM スライダーは、この最大値に合わせて調整されます。
CHE_LIMITS_WORKSPACE_IDLE_TIMEOUT	**1800000**	システムがワークスペースを一時停止した後にこれを停止する際に、ユーザーがワークスペースでアイドル状態になる期間。アイドル状態は、ユーザーがワークスペースと対話しない期間です。つまり、エージェントのいずれも対話を受け取っていない期間を意味します。ブラウザーウィンドウを開いたままにするとアイドル状態になります。
CHE_LIMITS_WORKSPACE_RUN_TIMEOUT	**0**	システムが一時停止するまでの、アクティビティーを問わず、ワークスペースが実行される期間 (ミリ秒単位)。一定期間後にワークスペースを自動的に停止する場合は、このプロパティーを設定します。デフォルトはゼロで、実行タイムアウトがないことを意味します。

表4.15 ユーザーワークスペースの制限

環境変数名	デフォルト値	説明
CHE_LIMITS_USER_WORKSPACES_RAM	**-1**	単一ユーザーがワークスペースの実行に割り当てることができる RAM の合計量。ユーザーは、この RAM を単一のワークスペースに割り当てるか、または複数のワークスペースに分散することができます。
CHE_LIMITS_USER_WORKSPACES_COUNT	**-1**	ユーザーが作成できるワークスペースの最大数。追加のワークスペースを作成しようとすると、ユーザーにはエラーメッセージが表示されます。これは、実行中および停止中のワークスペースの合計数に適用されます。
CHE_LIMITS_USER_WORKSPACES_RUN_COUNT	**1**	単一ユーザーが持てる実行中のワークスペースの最大数。ユーザーがこのしきい値に達し、追加のワークスペースを開始しようとすると、エラーメッセージと共にプロンプトが表示されます。ユーザーは、実行中のワークスペースを停止してから別のワークスペースをアクティベートする必要があります。

表4.16 組織ワークスペースの制限

環境変数名	デフォルト値	説明
CHE_LIMITS_ORGANIZATION_WORKSPACES_RAM	**-1**	単一組織 (チーム) がワークスペースの実行に割り当てることができる RAM の合計量。組織の所有者はこの RAM を割り当てることができますが、チームのワークスペース全体で適切に割り当てられているように見えます。
CHE_LIMITS_ORGANIZATION_WORKSPACES_COUNT	**-1**	組織が所有できるワークスペースの最大数。追加のワークスペースを作成しようとすると、組織にはエラーメッセージが表示されます。これは、実行中および停止中のワークスペースの合計数に適用されます。

環境変数名	デフォルト値	説明
CHE_LIMITS_ORGANIZATION_WORKSPACES_RUN_COUNT	**-1**	単一組織が持てる実行中のワークスペースの最大数。組織がこのしきい値に達し、追加のワークスペースを開始しようとすると、エラーメッセージと共にプロンプトが表示されます。組織は、実行中のワークスペースを停止してから別のワークスペースをアクティベートする必要があります。
CHE_MAIL_FROM__EMAIL__ADDRESS	**che@noreply.com**	メール通知の送信元のメールに使用されるアドレス

表4.17 組織通知の設定

環境変数名	デフォルト値	説明
CHE_ORGANIZATION_EMAIL_MEMBER__ADDED__SUBJECT	**You'vebeen added to a Che Organization**	組織の通知の件名およびテンプレート
CHE_ORGANIZATION_EMAIL_MEMBER__ADDED__TEMPLATE	**st-html-templates/user_added_to_organization**	組織の通知の件名およびテンプレート
CHE_ORGANIZATION_EMAIL_MEMBER__REMOVED__SUBJECT	**You'vebeen removed from a Che Organization**	
CHE_ORGANIZATION_EMAIL_MEMBER__REMOVED__TEMPLATE	**st-html-templates/user_removed_from_organization**	
CHE_ORGANIZATION_EMAIL_ORG__REMOVED__SUBJECT	**CheOrganization deleted**	
CHE_ORGANIZATION_EMAIL_ORG__REMOVED__TEMPLATE	**st-html-templates/organization_deleted**	
CHE_ORGANIZATION_EMAIL_ORG__RENAMED__SUBJECT	**CheOrganization renamed**	

環境変数名	デフォルト値	説明
CHE_ORGANIZATION_EMAIL_ORG__RENAMED__TEMPLATE	**st-html-templates/organization_renamed**	

表4.18 マルチユーザー固有の OpenShift インフラストラクチャー設定

環境変数名	デフォルト値	説明
CHE_INFRA_OPENSHIFT_OAUTH__IDENTITY__PROVIDER	**NULL**	Keycloak に登録されている Openshift アイデンティティープロバイダーのエイリアス。これは、現行の CodeReady Workspaces ユーザーが所有する Openshift namespace にワークスペース OpenShift リソースを作成するために使用されます。**che.infra.openshift.project** が空でない値に設定されている場合は NULL に設定する必要があります。詳細は、https://www.keycloak.org/docs/latest/server_admin/index.html#openshift-4 のドキュメントを参照してください。

表4.19 Keycloak の設定

環境変数名	デフォルト値	説明
CHE_KEYCLOAK_AUTH__SERVER__URL	**http://${CHE_HOST}:5050/auth**	**che.keycloak.oidcProvider** を使用している場合のみ、keycloak アイデンティティープロバイダーサーバーの URL を NULL に設定できます。
CHE_KEYCLOAK_REALM	**che**	Keycloak レルムはユーザーを認証するために使用されます。**che.keycloak.oidcProvider** を使用している場合のみ NULL に設定できます。
CHE_KEYCLOAK_CLIENT__ID	**che-public**	ユーザーの認証用にダッシュボード、ide、および cli で使用される che.keycloak.realm の Keycloak クライアント ID

表4.20 RedHat Che 固有の設定

環境変数名	デフォルト値	説明
CHE_KEYCLOAK_OSO_ENDPOINT	**NULL**	OSO oauth トークンにアクセスするための URL
CHE_KEYCLOAK_GITHUB_ENDPOINT	**NULL**	Github oauth トークンにアクセスするための URL
CHE_KEYCLOAK_ALLOWED__CLOCK__SKEW__SEC	**3**	exp または nbf 要求を検証する際にクロックスキューについて許容される秒数。
CHE_KEYCLOAK_USE__NONCE	**true**	OIDC オプションの **nonce** 機能を使用して、セキュリティーを強化します。
CHE_KEYCLOAK_JS__ADAPTER__URL	**NULL**	使用する Keycloak Javascript アダプターの URL。NULL に設定した場合は、デフォルトで使用される値は **${che.keycloak.auth_server_url}/js/keycloak.js** になり、別の **oidc_provider** を使用する場合は **<che-server>/api/keycloak/OIDCKeycloak.js** になります。
CHE_KEYCLOAK_OIDC__PROVIDER	**NULL**	以下の仕様で詳細に検出エンドポイントを指定する別の OIDC プロバイダーのベース URL。https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig
CHE_KEYCLOAK_USE__FIXED__REDIRECT__URLS	**false**	固定されたリダイレクト URL のみをサポートする別の OIDC プロバイダーを使用する場合は true に設定します。このプロパティーは、**che.keycloak.oidc_provider** が NULL の場合は無視されます。
CHE_KEYCLOAK_USERNAME__CLAIM	**NULL**	定義されていない場合、JWT トークンの解析時にユーザー名の要求がユーザーの表示名として使用されます。フォールバック値は「preferred_username」です。

環境変数名	デフォルト値	説明
CHE_OAUTH_SERVICE__MODE	**delegated**	「embedded」モードまたは「delegated」モードで使用できる OAuth 認証サービスの設定。「embedded」に設定すると、サービスは、(Single User モードの場合のように) CodeReady Workspaces の OAuthAuthenticator のラッパーとして機能します。「delegated」に設定すると、サービスは Keycloak IdentityProvider メカニズムを使用します。このプロパティーが正しく設定されていない場合は、ランタイム例外がスローされます。

4.2. プロジェクトストラテジーの設定

OpenShift プロジェクトストラテジーは、**CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** 環境変数を使用して設定されます。

警告

CHE_INFRA_KUBERNETES_NAMESPACE および **CHE_INFRA_OPENSHIFT_PROJECT** はレガシー変数です。新規インストールでは、これらの変数は未設定のままにします。更新時にこれらの変数を変更すると、データが失われる可能性があります。

警告

デフォルトでは、同じプロジェクト内で同時に実行できるワークスペースは 1 つだけです。「一度に複数のワークスペースの実行」を参照してください。

4.2.1. ワークスペースストラテジーごとに 1 つのプロジェクト

ストラテジーは、新規ワークスペースごとに新規プロジェクトを作成します。

ストラテジーを使用するには、**CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** 変数に **<workspaceID>** 識別子が含まれている必要があります。これは単独で使用することも、他の ID または任意の文字列と組み合わせることもできます。

例4.2 ワークスペースごとに1つのプロジェクト

'codeready-ws' プレフィックスおよびワークスペース ID で構成されるプロジェクト名を割り当てるには、以下を設定します。

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

4.2.2. すべてのワークスペースストラテジーに1つのプロジェクト

ストラテジーは、すべてのワークスペースに1つの事前に定義されたプロジェクトを使用します。

ストラテジーを使用するには、**CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** 変数の値は必要なプロジェクトの名前である必要があります。

例4.3 すべてのワークスペースに1つのプロジェクト

すべてのワークスペースを 'codeready-ws' プロジェクトで作成する場合は、以下を設定します。

```
CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT=codeready-ws
```

4.2.3. ユーザーストラテジーごとに1つのプロジェクト

ストラテジーは、独自のプロジェクトの各ユーザーを分離します。

ストラテジーを使用するには、**CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT** 変数値に1つまたは複数のユーザー識別子が含まれている必要があります。現在サポートされている ID は **<username>** および **<userId>** です。

例4.4 ユーザーごとに1つのプロジェクト

'codeready-ws' プレフィックスおよび個々のユーザー名（**codeready-ws-user1**、**codeready-ws-user2**) で構成されるプロジェクト名を割り当てるには、以下を設定します。

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

4.2.4. ユーザー定義のワークスペースプロジェクトの許可

CodeReady Workspaces サーバーは、ワークスペースの作成時にプロジェクトのユーザー選択を有効にするように設定できます。この機能はデフォルトでは無効にされます。ユーザー定義のワークスペースプロジェクトを許可するには、以下を実行します。

- Operator デプロイメントの場合、CheCluster カスタムリソースに以下のフィールドを設定します。

```
allowUserDefinedWorkspaceNamespaces
```

4.3. 一度に複数のワークスペースの実行

この手順では、複数のワークスペースを同時に実行する方法を説明します。これにより、ユーザーごとに複数のワークスペースのコンテキストを並行して実行できます。

前提条件

- **'`oc'** ツールを使用できる。
- OpenShift で実行される CodeReady Workspaces のインスタンス。

注記

以下のコマンドは、**-n** オプションのユーザー例として、デフォルトの OpenShift プロジェクト **openshift-workspaces** を使用します。

手順

1. ユーザーごとの同時ワークスペースの数を無制限にするには、**1** のデフォルト制限を **-1** に変更します。

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

1. **per-workspace** または **unique** PVC ストラテジーを設定します。永続ボリュームストラテジーを使用した CodeReady Workspaces ワークスペースの設定について参照して ください。

注記

common PVC ストラテジーを使用する場合、永続ボリュームを **ReadWriteMany** アクセスモードを使用するように設定します。これにより、ユーザーの同時ワークスペースのいずれかが共通 PVC からの/への読み取り/書き込みを実行できます。

4.4. ワークスペース公開ストラテジーの設定

CodeReady Workspaces サーバーのワークスペース公開ストラテジーを設定し、内部で実行されているアプリケーションが外部からの攻撃を受けないようにする方法を説明します。

ワークスペースの公開ストラテジーは、**che.infra.kubernetes.server_strategy** 設定プロパティーまたは **CHE_INFRA_KUBERNETES_SERVER__STRATEGY** 環境変数を使用して、CodeReady Workspaces サーバーごとに設定されます。

che.infra.kubernetes.server_strategy のサポートされる値は、以下のとおりです。

- **multi-host**

multi-host ストラテジーの有効化：

1. 以下を設定します。
   - **che.infra.kubernetes.ingress.domain** 設定プロパティー
     または
   - **CHE_INFRA_KUBERNETES_INGRESS_DOMAIN** 環境変数
     ワークスペースコンポーネントのサブドメインをホストするドメイン名と一致する。

4.4.1. ワークスペース公開ストラテジー

ワークスペースの特定のコンポーネントは、OpenShift クラスター外からアクセスできるようにする必要があります。通常、これはワークスペースの IDE のユーザーインターフェースですが、開発されるアプリケーションの Web UI である可能性もあります。これにより、開発プロセスでの開発者のアプリケーションとの対話が可能になります。

ワークスペースをユーザーが使用できるようにするためのサポートされる方法は、**ストラテジー** と呼ばれます。このストラテジーは、ワークスペースコンポーネントに新しいサブドメインが作成されるかどうか、およびこれらのコンポーネントを利用可能にするホストを定義します。

CodeReady Workspaces は以下をサポートします： * **multi-host** ストラテジー

4.4.1.1. Multi-host ストラテジー

このストラテジーにより、各ワークスペースコンポーネントには、CodeReady Workspaces サーバーに設定された主なドメインの新規サブドメインが割り当てられます。OpenShift では、これは唯一のストラテジーであり、ワークスペースの公開ストラテジーの手動設定は常に無視されます。

このストラテジーは、コンポーネントへの URL に存在するいずれのパスもコンポーネントごとにそのまま受信されるため、コンポーネントのデプロイメントの点で最も理解しやすいストラテジーです。

Transport Layer Security (TLS) プロトコルの使用によってセキュリティーが保護された CodeReady Workspaces サーバーで、各ワークスペースの各コンポーネントに新規のサブドメインを作成するには、CodeReady Workspaces デプロイメントが機能するため、このようなサブドメインすべてについてワイルドカード証明書が利用可能である必要があります。

4.4.2. セキュリティーに関する考慮事項

本セクションでは、さまざまな CodeReady Workspaces ワークスペースの公開ストラテジーを使用するセキュリティー上の影響について説明します。

本セクションのすべてのセキュリティー関連の考慮事項は、マルチユーザーモードの CodeReady Workspaces のみに適用されます。単一ユーザーモードは、セキュリティー制限を課しません。

4.4.2.1. JSON Web トークン (JWT) プロキシー

すべての CodeReady Workspaces プラグイン、エディター、およびコンポーネントには、それらにアクセスするユーザーの認証が必要になる場合があります。この認証は、その設定に基づいて対応するコンポーネントのリバースプロキシーとして機能し、コンポーネントの代わりに認証を実行する JSON Web トークン (JWT) プロキシーを使用して実行されます。

認証では、CodeReady Workspaces サーバーの特別なページへのリダイレクトを使用して、ワークスペースおよびユーザー固有の認証トークン (ワークスペースアクセストークン) を最初に要求されたページに伝播します。

JWT プロキシーは、受信要求の以下の場所からのワークスペースアクセストークンを受け入れます。

1. トークンクエリーパラメーター
2. bearer-token 形式の Authorization ヘッダー
3. **access_token** cookie

4.4.2.2. セキュリティーが保護されたプラグインおよびエディター

CodeReady Workspaces ユーザーはワークスペースのプラグインやワークスペースのエディター (Che-Theia など) のセキュリティーを保護する必要はありません。これは、JWT プロキシー認証はユーザーに透過的であり、**meta.yaml** 記述子のプラグインまたはエディター定義によって制御されるためです。

4.4.2.3. セキュリティー保護されたコンテナーイメージコンポーネント

コンテナーイメージのコンポーネントは、必要に応じて devfile の作成者側が CodeReady Workspaces が提供する認証を要求するカスタムエンドポイントを定義できます。この認証は、エンドポイントの 2 つのオプション属性を使用して設定されます。

- **secure** - CodeReady Workspaces サーバーに対し、エンドポイントの前に JWT プロキシーを配置するよう指示するブール値属性。このエンドポイントでは、「JSON Web トークン (JWT) プロキシー」 で説明されているいくつかの方法のいずれかを使用して、ワークスペースのアクセストークンが提供される必要があります。属性のデフォルト値は **false** です。

- **cookiesAuthEnabled** - 「JSON Web トークン (JWT) プロキシー」 で説明されているように、CodeReady Workspaces サーバーに対し、現在のユーザー認証の非認証要求を自動的にリダイレクトするように指示するブール値属性。この属性を **true** に設定すると、CSRF (クロスサイトリクエストフォージェリー) 攻撃が可能になり、セキュリティー上の影響が発生します。属性のデフォルト値は **false** です。

4.4.2.4. クロスサイトリクエストフォージェリー攻撃

cookie ベースの認証に、JWT プロキシーによってセキュリティーが保護されたアプリケーションは CSRF (Cross-site Request forgery) 攻撃の対象となりやすくする場合があります。アプリケーションに脆弱性がないことを確認するには、CSRF (Cross-site request forgery) についての Wikipedia ページやその他のリソースを参照してください。

4.4.2.5. フィッシング攻撃

JWT プロキシーの背後にあるサービスとホストを共有するワークスペースを使用してクラスター内に Ingress またはルートを作成できる攻撃者は、サービスの作成やとくに偽造された Ingress オブジェクトの作成が可能になる場合があります。このようなサービスまたは Ingress がワークスペースで以前に認証されている適切なユーザーによってアクセスされる際に、攻撃者は偽の URL への適切なユーザーのブラウザーが送信する cookie からワークスペースアクセストークンを盗むことができる可能性があります。この攻撃ベクトルを排除するには、Ingress のホストの設定を禁止するように OpenShift を設定します。

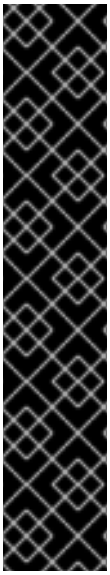
4.5. ワークスペース NODESELECTOR の設定

このセクションでは、CodeReady Workspaces ワークスペースの Pod について **nodeSelector** を設定する方法を説明します。

手順

CodeReady Workspaces は **CHE_WORKSPACE_POD_NODE__SELECTOR** 環境変数を使用して **nodeSelector** を設定します。この変数には、nodeSelector ルールを形成するためにコンマ区切りの **key=value** ペアのセットが含まれるか、またはこれを無効にする **NULL** が含まれる場合があります。

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



重要

nodeSelector は CodeReady Workspaces のインストール時に設定する必要があります。これにより、既存のワークスペース PVC および Pod が異なるゾーンにスケジュールされることによってボリュームのアフィニティーの競合が生じ、既存のワークスペースが実行できなくなることを防ぐことができます。

大規模なマルチゾーンクラスターの複数のゾーンで Pod および PVC がスケジュールされるのを防ぐには、PVC の作成プロセスを調整する追加の **StorageClass** オブジェクトを作成します (**allowedTopologies** フィールドに注目してください)。

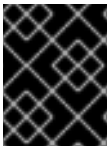
この新たに作成された **StorageClass** の名前を **CHE_INFRA_KUBERNETES_PVC_STORAGECLASSNAME** 環境変数を使用して CodeReady Workspaces に渡します。この変数のデフォルトの空の値の場合、CodeReady Workspaces に対し、クラスターのデフォルト **StorageClass** を使用するように指示します。

4.6. RED HAT CODEREADY WORKSPACES サーバーのホスト名の設定

この手順では、カスタムホスト名を使用するように Red Hat CodeReady Workspaces を設定する方法を説明します。

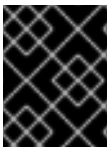
前提条件

- **oc** ツールが利用可能である。
- 証明書とプライベートキーファイルが生成されます。



重要

プライベートキーと証明書のペアを生成するには、他の Red Hat CodeReady Workspaces ホストの場合と同じ CA を使用する必要があります。



重要

DNS プロバイダーに対し、カスタムホスト名をクラスター Ingress を参照するよう要求します。

手順

1. CodeReady Workspaces のプロジェクトを事前に作成します。

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

2. tls シークレットを作成します。

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

1 tls シークレット名

2 プライベートキーを含むファイル

3 証明書を含むファイル

3. カスタムリソースに以下の値を設定します。

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

1 カスタム Red Hat CodeReady Workspaces サーバーのホスト名

2 tls シークレット名

4. CodeReady Workspaces がすでにデプロイされており、CodeReady Workspaces を新しい CodeReady Workspaces ホスト名を使用するように再設定する必要がある場合には、RH-SSO を使用してログインし、**CodeReady Workspaces** レルムで **codeready-public** クライアントを選択し、CodeReady Workspaces ホスト名の値で **Validate Redirect URIs** および **Web Origins** フィールドを更新します。

4.7. 自己署名証明書を使用した GIT リポジトリーをサポートする CODEREADY WORKSPACES のデプロイ

この手順では、自己署名証明書を使用するリポジトリーで Git 操作のサポートのあるデプロイメント用に CodeReady Workspaces を設定する方法を説明します。

前提条件

- Git バージョン 2 以降

手順

自己署名の Git リポジトリーのサポートの設定。

1. Git サーバーの詳細情報を使用して新規の **configMap** を作成します。

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

このコマンドで、**<host:port>** を Git サーバーの HTTPS 接続のホストおよびポートに置き換えます (オプション)。

注記

- **githost** を指定しないと、指定された証明書がすべての HTTPS リポジトリーに使用されます。
- 証明書ファイルの名前は **ca.crt** にする必要があります。
- 証明書ファイルは、通常、以下のような Base64 ASCII ファイルとして保存されます。**.pem**、 **.crt**、**.ca-bundle**。また、これらはバイナリーデータとしてエンコードすることもできます (例: **.cer**)。証明書ファイルを保持するすべての **Secrets** は、バイナリーデータ証明書ではなく、Base64 ASCII 証明書を使用する必要があります。

2. ワークスペースの公開ストラテジーを設定します。
gitSelfSignedCert プロパティーを更新します。これを行うには、以下を実行します。

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

3. 新規ワークスペースを作成および開始します。ワークスペースによって使用されるすべてのコンテナーは、自己署名証明書のあるファイルを含む特殊なボリュームをマウントします。リポジトリーの **.git/config** ファイルには、Git サーバーホスト (その URL) と **http** セクションの証明書へのパスについての情報が含まれます（ git-configに関する Git ドキュメントを参照してください)。以下に例を示します。

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

4.8. ストレージクラスを使用した CODEREADY WORKSPACES のインストール

設定済みのインフラストラクチャーストレージを使用するように CodeReady Workspaces を設定するには、ストレージクラスを使用して CodeReady Workspaces をインストールします。これは、ユーザーがデフォルト以外のプロビジョナーによって提供される永続ボリュームをバインドする必要がある場合にとくに役立ちます。これを実行するには、ユーザーは CodeReady Workspaces のデータを節約するためにこのストレージをバインドし、そのストレージのパラメーターを設定します。これらのパラメーターは、以下を決定します。

- 特殊なホストパス
- ストレージ容量
- ボリューム mod
- マウントオプション
- ファイルシステム
- アクセスモード
- ストレージタイプ
- その他多数

CodeReady Workspaces には、データの格納に永続ボリュームが必要な 2 つのコンポーネントがあります。

- PostgreSQL データベース。
- CodeReady Workspaces ワークスペース。CodeReady Workspaces ワークスペースは、ボリューム (例: **/projects** ボリューム) を使用してソースコードを保存します。

注記

CodeReady Workspaces ワークスペースのソースコードは、ワークスペースが一時的ではない場合にのみ永続ボリュームに保存されます。

永続ボリューム要求 (PVC）のファクト:

- CodeReady Workspaces はインフラストラクチャーに永続ボリュームを作成しません。
- CodeReady Workspaces は永続ボリューム要求 (PVC）を使用して永続ボリュームをマウントします。
- CodeReady Workspaces サーバーは永続ボリューム要求を作成します。
ユーザーは、CodeReady Workspaces PVC でストレージクラス機能を使用するために、CodeReady Workspaces 設定でストレージクラス名を定義します。ストレージクラスを使用すると、ユーザーは追加のストレージパラメーターを使用してインフラストラクチャーストレージを柔軟に設定します。クラス名を使用して、静的にプロビジョニングされた永続ボリュームを CodeReady Workspaces PVC にバインドすることもできます。

手順

CheCluster カスタムリソース定義を使用してストレージクラスを定義します。

1. ストレージクラス名を定義します。
これを行うには、以下のいずれかの方法を使用します。
   - **server:start コマンドの引数を使用します。**
     i. PostgreSQL PVC のストレージクラス名を指定します。
     --postgres-pvc-storage-class-name フラグを指定して **crwctl server:start** コマンドを使用します。

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

     ii. CodeReady Workspaces ワークスペースのストレージクラス名を指定します。
     --workspace-pvc-storage-class-name フラグを指定して **server:start** コマンドを使用します。

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

     CodeReady Workspaces ワークスペースでは、ワークスペースの PVC ストラテジーに応じてストレージクラスの名前の動作が異なります。

注記

postgres-pvc-storage-class-name=postgress-storage および **workspace-pvc-storage-class-name** は Operator インストーラーおよび Helm インストーラーで機能します。

- **カスタムリソース YAML ファイルを使用してストレージクラス名を定義します。**

  i. CodeReady Workspaces インストールに定義されたカスタムリソースで YAML ファイルを作成します。

  ii. フィールド **spec#storage#postgresPVCStorageClassName** および **spec#storage#workspacePVCStorageClassName** を定義します。

```
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. カスタムリソースで codeready-workspaces サーバーを起動します。

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

2. CodeReady Workspaces を、ワークスペースを 1 つ目の永続ボリュームに、PostreSQL データベースを 2 つ目の永続ボリュームに保存するように設定します。

   a. カスタムリソース YAML ファイルを変更します。

   - **pvcStrategy** を **common** に設定します。
   - 単一のプロジェクトでワークスペースを開始するように CodeReady Workspaces を設定します。
   - **postgresPVCStorageClassName** および **workspacePVCStorageClassName** のストレージクラス名を定義します。
   - YAML ファイルの例:

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
metadata:
  name: codeready-workspaces
spec:
```

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

b. カスタムリソースで codeready-workspaces サーバーを起動します。

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

3. クラス名を使用して静的にプロビジョニングされたボリュームをバインドします。

a. PostgreSQL データベースの永続ボリュームを定義します。

```
# 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. CodeReady Workspaces ワークスペースの永続ボリュームを定義します。

```
# 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. 2 つの永続ボリュームをバインドします。

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

注記

ボリュームの有効なファイルパーミッションを指定する必要があります。これは、ストレージクラスの設定を使用して実行することも、手動で実行することもできます。パーミッションを手動で定義するには、**storageClass#mountOptions uid** および **gid** を定義します。PostgreSQL ボリュームには、**uid=26** および **gid=26** が必要です。

4.9. ストレージタイプの設定

Red Hat CodeReady Workspaces は、さまざまな機能を備えた 3 種類のストレージをサポートします。

- Persistent (永続)
- Ephemeral (一時)
- Asynchronous (非同期)

4.9.1. 永続ストレージ

永続ストレージにより、マウントされた永続ボリュームにユーザーの変更を直接保存できます。とくに小さなファイルが数多くある場合に I/O が低速になりますが、ユーザーの変更は OpenShift インフラストラクチャー (ストレージバックエンド) によって保護されます。たとえば、Node.js プロジェクトには多くの依存関係が含まれることがあり、**node_modules/** ディレクトリーには数千の小さなファイルが含まれます。

注記

I/O の速度は、環境内で設定されているストレージクラスによって異なります。

永続ストレージは、新規ワークスペースのデフォルトモードです。この設定をワークスペース設定で表示できるようにするには、以下を devfile に追加します。

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

4.9.2. 一時ストレージ

一時ストレージは、ファイルを **emptyDir** ボリュームに保存します。このボリュームは最初は空の状態です。Pod がノードから削除されると、**emptyDir** ボリュームのデータは永久に削除されます。つまり、ワークスペースの停止または再起動時にすべての変更が失われます。

重要

変更を保存するには、一時ワークスペースを停止する前に、リモートへのコミットおよびプッシュを実行します。

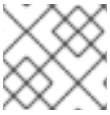
一時モードは、永続ストレージよりも高速な I/O を提供します。このストレージタイプを有効にするには、以下をワークスペース設定に追加します。

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

表4.21 AWS EBSでの一時モード(emptyDir)とサイトごとのモードの比較

コマンド	Ephemeral (一時)	Persistent (永続)
Red Hat CodeReady Workspaces のクローン作成	0 m 19 s	1 m 26 s
1000 のランダムなファイルの生成	1 m 12 s	44 m 53 s

4.9.3. 非同期ストレージ



注記

非同期ストレージは実験的な機能です。

非同期ストレージは、永続ストレージと一時モードの組み合わせです。初期ワークスペースコンテナーは **emptyDir** ボリュームをマウントします。次に、ワークスペースの停止時にバックアップが実行され、変更がワークスペースの起動時に復元されます。非同期ストレージは、(一時モードと同様の）高速 I/O を提供し、ワークスペースプロジェクトの変更は永続化されます。

同期は、SSH プロトコルを使用して rsync ツールで実行されます。ワークスペースが非同期ストレージで設定されている場合、workspace-data-sync プラグインはワークスペース設定に自動的に追加されます。プラグインはワークスペースの開始時に **rsync** コマンドを実行して変更を復元します。ワークスペースが停止したら、変更を永続ストレージに送信します。

比較的小規模なプロジェクトの場合、復元手順は高速で、Che-Theia が初期化されるとプロジェクトのソースファイルはすぐに利用可能になります。**rsync** にかかる時間が長いと、同期プロセスは Che-Theia のステータスバーの領域に表示されます。(Che-Theia リポジトリーの拡張)。

注記

非同期モードには、以下の制限があります。

- **common** PVC ストラテジーのみをサポートします。
- **ユーザーごとの** プロジェクトストラテジーのみをサポートします。
- 1 度に実行できるワークスペースは 1 つのみです。

ワークスペースの非同期ストレージを設定するには、以下をワークスペース設定に追加します。

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

4.9.4. CodeReady Workspaces ダッシュボードのストレージタイプのデフォルトの設定

fo を使用します。以下の 2 つの **che.properties** に従って CodeReady Workspaces ダッシュボードでデフォルトのクライアント値を設定します。

che.workspace.storage.available_types

ワークスペースの作成または更新時に、ダッシュボードなどのクライアントがユーザーに提案するストレージタイプに使用できる値を定義します。使用できる値: **persistent**、**ephemeral**、および **async**。複数の値をコンマで区切ります。以下に例を示します。

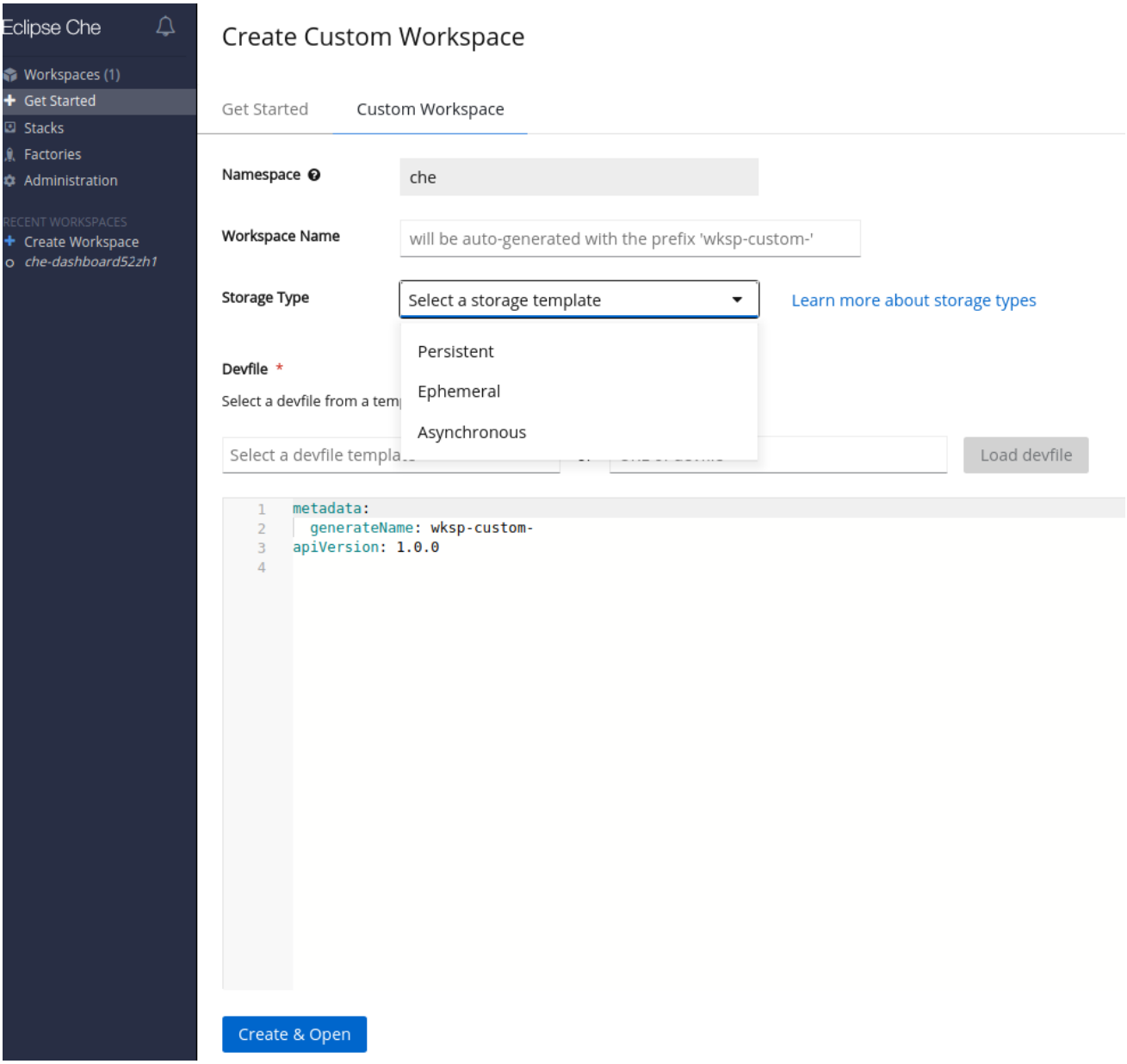
```
che.workspace.storage.available_types=persistent,ephemeral,async
```

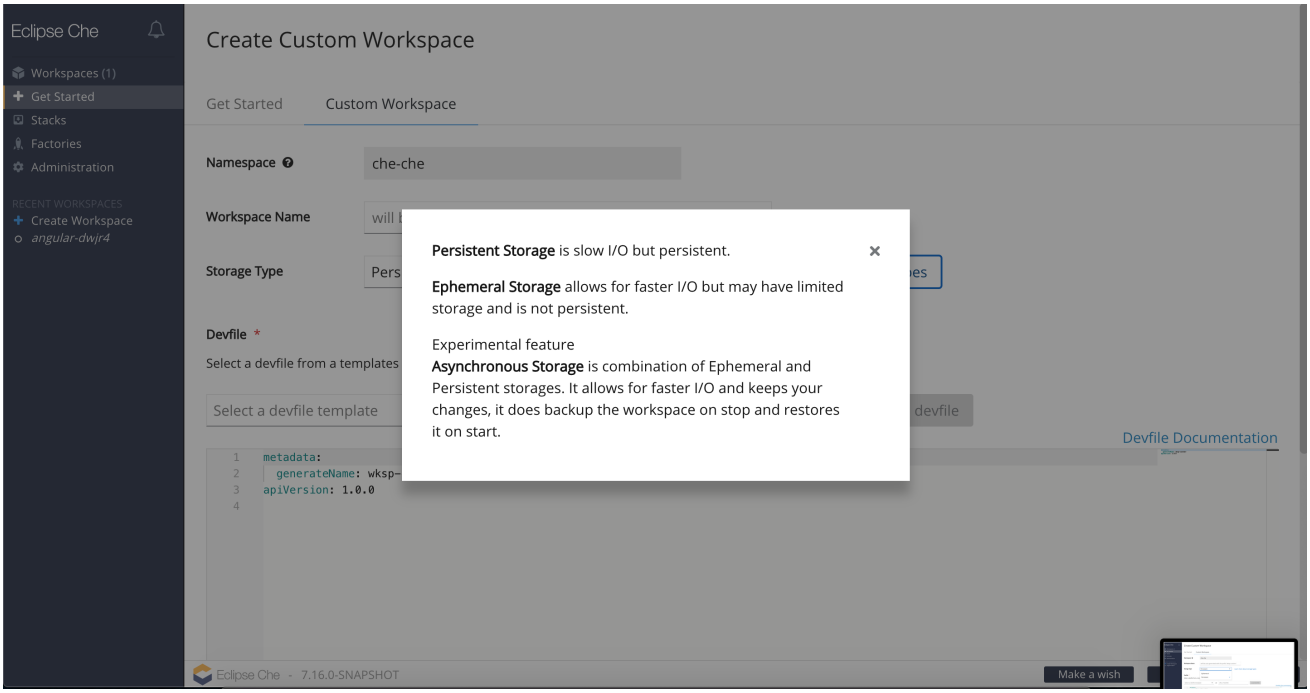
che.workspace.storage.preferred_type

ワークスペースの作成または更新時に、ダッシュボードなどのクライアントがユーザーに提案するストレージタイプのデフォルト値を定義します。**async** 値は、実験的な取組であるため、デフォルトタイプとしての使用は推奨されません。以下に例を示します。

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

Storage Type ドロップダウンメニューは、ユーザーダッシュボードの **Create Custom Workspace** ページで利用できます。


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


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

4.9.5. 非同期ストレージ Pod のアイドリング

CodeReady Workspaces は、設定された期間に使用されていない場合に、非同期ストレージ Pod をシャットダウンできます。

動作を調整するには、以下の設定プロパティーを使用します。

che.infra.kubernetes.async.storage.shutdown_timeout_min

最後のアクティブなワークスペースの停止後に非同期ストレージ Pod が停止されるアイドル時間を定義します。デフォルト値は 120 分です。

che.infra.kubernetes.async.storage.shutdown_check_period_min

非同期ストレージ Pod でアイドル状態をチェックする頻度を定義します。デフォルト値は 30 分です。

4.10. TLS 証明書の CODEREADY WORKSPACES サーバーの JAVA トラストストアへのインポート

CodeReady Workspaces サーバーが HTTPS 要求を RH-SSO、プロキシー、または git サーバーとして外部サービスに送信しようとすると、CodeReady Workspaces が外部サービスによって使用される TLS 証明書を信頼しない場合には接続に失敗します。

この問題を修正するには、TLS 証明書に関する情報を CodeReady Workspaces 設定に追加することで、アイデンティティーや Git サーバーなどの外部サービスとの HTTPS 通信を承認するように CodeReady Workspaces を設定します。

前提条件

- **oc** ツールが利用可能である。

手順

1. 外部サービス証明書をローカルファイルシステムに保存します。
2. 必要な TLS 証明書で新規の configMap を作成します。

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

複数の証明書を適用するには、上記のコマンドに別の **--from-file=<certificate-file-path>** オプションを追加します。

3. 既存の CodeReady Workspaces サーバー設定を更新します。

注記

これらのステップは、CodeReady Workspaces の既存のインスタンスで使用します。自己署名 TLS 証明書で CodeReady Workspaces の新規インスタンスをインストールするには、既存の設定を更新する代わりに、選択したインストール方法に基づいて、新しい **CheCluster** カスタムリソースまたは Helm Chart プロパティーを作成します。

- CodeReady Workspaces Operator デプロイメントの 場合：

- 以前に作成した configMap に一致するように **spec.server.ServerTrustStoreConfigMapName CheCluster** カスタムリソースプロパティーを編集して、新規に作成された configMap の名前を定義します。

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

検証

証明書が正常に追加されると、CodeReady Workspaces サーバーは起動し、HTTPS で RH-SSO 設定を取得します。これ以外の場合は、以下を確認する一覧を以下に示します。

- CheCluster 属性 **serverTrustStoreConfigMapName** の値は ConfigMap の名前と一致します。以下のコマンドを使用して値を取得します。

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

- CodeReady Workspaces Pod ボリューム一覧には、ConfigMap をデータソースとして使用する 1 つのボリュームが含まれます。CodeReady Workspaces Pod のボリュームの一覧を取得するには、以下を実行します。

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

- 証明書は、CodeReady Workspaces サーバーコンテナーのフォルダー **/public-certs/** にマウントされます。このコマンドは、そのフォルダー内のファイルの一覧を返します。

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

- CodeReady Workspaces サーバーログに、CodeReady Workspaces の自己署名証明書を含む、Java トラストストアに追加されたすべての証明書についての行があります。

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

- $CodeReady Workspaces サーバーの Java 信頼には証明書が含まれます。証明書の SHA1 fingerpints は、以下のコマンドで返される信頼に含まれる証明書の SHA1 の一覧にあります。

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

(...)
```

ローカルファイルシステムの証明書の SHA1 ハッシュを取得するには、以下のコマンドを実行します。

```
$ 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
```

第5章 CODEREADY WORKSPACES のアップグレード

本章では、CodeReady Workspaces インスタンスをバージョン 2.3 から CodeReady Workspaces 2.4 にアップグレードする方法を説明します。

CodeReady Workspaces インスタンスのインストールするために使用する方法により、アップグレードする方法が決まります。

- 「OperatorHub を使用した CodeReady Workspaces のアップグレード」
- 「CLI 管理ツールを使用した CodeReady Workspaces のアップグレード」
- 「制限された環境での CLI 管理ツールを使用した CodeReady Workspaces のアップグレード」

5.1. OPERATORHUB を使用した CODEREADY WORKSPACES のアップグレード

このセクションでは、OpenShift Web コンソールの OperatorHub から Operator を使用して、以前のマイナーバージョンからアップグレードする方法を説明します。

前提条件

- OpenShift インスタンスの管理者アカウント。
- OpenShift の同じインスタンス上で OperatorHub の Operator を使用してインストールされた、以前のマイナーバージョンの CodeReady Workspaces のインスタンス。

手順

1. OpenShift Web コンソールを開きます。
2. **Operators → Installed Operators** セクションに移動します。
3. インストールされた Operator の一覧で **Red Hat CodeReady Workspaces**をクリックします。
4. **Subscription** タブに移動し、以下のオプションを有効にします。
   - **Channel**: **latest**
   - **Approval**: **Automatic**

検証手順

1. CodeReady Workspaces インスタンスに移動します。
2. 2.4 のバージョン番号がページ下部に表示されます。

5.2. CLI 管理ツールを使用した CODEREADY WORKSPACES のアップグレード

本セクションでは、CLI 管理ツールを使用して以前のマイナーバージョンからアップグレードする方法を説明します。

前提条件

- OpenShift インスタンスの管理者アカウント
- 以前のマイナーバージョンの Red Hat CodeReady Workspaces の実行中のインスタンス。これは、OpenShift の同じインスタンスで CLI 管理ツールを使用して **<openshift-workspaces>** プロジェクトにインストールされています。
- **crwctl** 2.4 バージョン管理ツールのインストール。「crwctl CLI 管理ツールのインストール」を参照してください。

手順

1. CodeReady Workspaces 2.3 インスタンスで実行されているすべてのワークスペースで、変更を保存し、Git リポジトリーに再度プッシュします。
2. CodeReady Workspaces 2.3 インスタンスのすべてのワークスペースをシャットダウンします。
3. 次のコマンドを実行します。

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

注記

低速なシステムまたはインターネット接続の場合は、**--k8spodwaittimeout=1800000** フラグオプションを **crwctl server:update** コマンドに追加し、Pod のタイムアウト期間を 1800000 ms 以上に拡張します。

検証手順

1. CodeReady Workspaces インスタンスに移動します。
2. 2.4 のバージョン番号がページ下部に表示されます。

5.3. 制限された環境での CLI 管理ツールを使用した CODEREADY WORKSPACES のアップグレード

本セクションでは、制限された環境で CLI 管理ツールを使用して Red Hat CodeReady Workspaces をアップグレードする方法を説明します。アップグレードパスは、CodeReady Workspaces バージョン 2.3 からバージョン 2.4 へのマイナーバージョンの更新をサポートします。

前提条件

- OpenShift インスタンスの管理者アカウント。
- **<openshift-workspaces>** プロジェクトの crwctl **--installer operator** メソッドを使用してインストールされた、CLI 管理ツールでインストールされた Red Hat CodeReady Workspaces の実行中のインスタンスバージョン 2.3。「制限された環境での CodeReady Workspaces のインストール」を参照してください。
- **crwctl** 2.4 管理ツールが利用できる。「crwctl CLI 管理ツールのインストール」 を参照してください。

5.3.1. 制限された環境でのネットワーク接続について

CodeReady Workspaces では、CodeReady Workspaces 用に作成された各 OpenShift Route が OpenShift クラスター内からアクセスできる必要があります。これらの CodeReady Workspaces コンポーネントには OpenShift Route の **codeready-workspaces-server**、**keycloak**、**devfile-registry**、**plugin-registry** があります。

環境のネットワークトポロジーを考慮して、これを実行する最善の方法を判断してください。

> **例5.1 公開インターネットから切断された、会社または組織が所有するネットワーク**
>
> ネットワーク管理者は、クラスターからのトラフィックを OpenShift Route ホスト名にルーティングできるようにする必要があります。

> **例5.2 クラウドプロバイダーのプライベートサブネットワーク**
>
> トラフィックがノードから出て、外部に表示されるロードバランサーに到達できるようにするプロキシー設定を作成します。

5.3.2. プライベートレジストリーの準備

前提条件

- **oc** ツールが利用可能である。
- **skopeo** ツール (バージョン 0.1.40 以降) が利用できる。
- **podman** ツールが利用できる。
- OpenShift クラスターからアクセスできるイメージ、および V2 イメージマニフェスト (スキーマバージョン 2) フォーマットのサポート。インターネットへのアクセスが一時的に可能な場所から、これにプッシュできることを確認します。

表5.1 サンプルで使用されるプレースホルダー

<source-image>	レジストリー、組織、およびダイジェストなどのソースイメージの詳細な組み合わせ (coordinate)。
<target-registry>	ターゲットコンテナーイメージレジストリーのホスト名およびポート。
<target-organization>	ターゲットのコンテナーイメージレジストリー内の組織
<target-image>	ターゲットのコンテナーイメージレジストリーのイメージ名とダイジェスト。
<target-user>	ターゲットのコンテナーイメージレジストリーのユーザー名。
<target-password>	ターゲットのコンテナーイメージレジストリーのユーザーパスワード。

手順

1. 内部イメージレジストリーにログインします。

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

ヒント

内部レジストリーへのプッシュを試行する際に **x509: certificate signed by unknown authority** などのエラーが発生した場合には、以下のいずれかの回避策を試してください。

- OpenShift クラスターの証明書を **/etc/containers/certs.d/<target-registry>** に追加する。
- **/etc/containers/registries.conf** にある Podman 設定ファイルに以下の行を追加して、レジストリーを非セキュアなレジストリーとして追加する。

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

2. ダイジェストを変更せずにイメージをコピーします。以下の表のすべてのイメージに対して、この手順を繰り返します。

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

注記

表5.2 名前に含まれるプレフィックスまたはキーワードからの container-images の使用について

使用	プレフィックスまたはキーワード
Essential	**stacks-**、**plugin-** ではない。 **-openj-**
Workspaces	**stacks-**、 **plugin-**
z および Power	**-openj-**

表5.3 プライベートレジストリーでコピーするイメージ

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

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:7702adb0ed28b635e45804e87fe5dd98bdd3aa766fed7845a8ce509b91c22e36	devfileregistry-rhel8@sha256:7702adb0ed28b635e45804e87fe5dd98bdd3aa766fed7845a8ce509b91c22e36
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:8afecd5b0edc7734532ee76ff9eac1fc4814d8aaa6c9be440a2a88a20c014e4e	jwtproxy-rhel8@sha256:8afecd5b0edc7734532ee76ff9eac1fc4814d8aaa6c9be440a2a88a20c014e4e
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:c9bebc895e5fa5a0bd4ecaedfd5384ab75a45a96b6314ba5d4a6f4c1e8e109f9	machineexec-rhel8@sha256:c9bebc895e5fa5a0bd4ecaedfd5384ab75a45a96b6314ba5d4a6f4c1e8e109f9
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:27a71612f9bd3bee77adb4e164c44c61cf5085458d592215b2fe74c55d11abc6	plugin-java11-openj9-rhel8@sha256:27a71612f9bd3bee77adb4e164c44c61cf5085458d592215b2fe74c55d11abc6
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:e9deebbc320d28a2f425e858ed3dcf87fc67a40f6654d6eb7c2b6feea022b7d6	plugin-java11-rhel8@sha256:e9deebbc320d28a2f425e858ed3dcf87fc67a40f6654d6eb7c2b6feea022b7d6
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:14f2774e92b70d85280e506f81e2ea9a89c26490fd53a4421df8a694bd944d2d	plugin-java8-openj9-rhel8@sha256:14f2774e92b70d85280e506f81e2ea9a89c26490fd53a4421df8a694bd944d2d
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:d04f70c8340abaee1a282b77158d054f4faf2225bc17c79aafb413396c367782	plugin-java8-rhel8@sha256:d04f70c8340abaee1a282b77158d054f4faf2225bc17c79aafb413396c367782
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:d87aed64704369a50d1e54a57815b699f74d4efad1401d1a638808e655a37e48	plugin-kubernetes-rhel8@sha256:d87aed64704369a50d1e54a57815b699f74d4efad1401d1a638808e655a37e48

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:9c43a02b0dd0f66744359c5ccdb1f1780ecd92c3dc31b14d73b553ba763af8ab	plugin-openshift-rhel8@sha256:9c43a02b0dd0f66744359c5ccdb1f1780ecd92c3dc31b14d73b553ba763af8ab
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:d0eebf2c8b460adb75dc6bc5200aa9fd40d030b7b17c6b1c3b9d3c879f4652ee	pluginbroker-artifacts-rhel8@sha256:d0eebf2c8b460adb75dc6bc5200aa9fd40d030b7b17c6b1c3b9d3c879f4652ee
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:cff23432d1d397bbbc7df65be9d6ddf4a97a3ef1801708bb7bb7d2fa72dbcce3	pluginbroker-metadata-rhel8@sha256:cff23432d1d397bbbc7df65be9d6ddf4a97a3ef1801708bb7bb7d2fa72dbcce3
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:9f37917122c20fc83e6558a5484efab42650958b513a22920f449f948e50cd51	pluginregistry-rhel8@sha256:9f37917122c20fc83e6558a5484efab42650958b513a22920f449f948e50cd51
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:63bf304cd04576048012693e7e8544a5a703790f99551554a75798bc799b112b	server-rhel8@sha256:63bf304cd04576048012693e7e8544a5a703790f99551554a75798bc799b112b
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:56543cfeeeac030821557ac4937db40f6845e874193c79c30267a680f9b2cbe7	stacks-cpp-rhel8@sha256:56543cfeeeac030821557ac4937db40f6845e874193c79c30267a680f9b2cbe7
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:13628110b96de0e516ff2dfa29cdcaee64cd8f8978052c8160c294c332dba9f0	stacks-dotnet-rhel8@sha256:13628110b96de0e516ff2dfa29cdcaee64cd8f8978052c8160c294c332dba9f0
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:fef91718ccebc4cd9b89999f6b5df83bf3d60fce657f6f44eda092100549af2c	stacks-golang-rhel8@sha256:fef91718ccebc4cd9b89999f6b5df83bf3d60fce657f6f44eda092100549af2c

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:b75f498954fbe858c74f80a89d132ba3560f40c0f697b0cd9550ed5663078ef6	stacks-php-rhel8@sha256:b75f498954fbe858c74f80a89d132ba3560f40c0f697b0cd9550ed5663078ef6
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:942e1e6328169508e3fff8fd96c575d7a423339ced17dbf5813d61d1971adaef	theia-endpoint-rhel8@sha256:942e1e6328169508e3fff8fd96c575d7a423339ced17dbf5813d61d1971adaef
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:78edc9f75680cbe7f63774d9dfbbc505401486a73c8e420380e1d3078bdf9f2a	theia-rhel8@sha256:78edc9f75680cbe7f63774d9dfbbc505401486a73c8e420380e1d3078bdf9f2a
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:d6a7bdbf4726fe0e0e54c0bce9b2257bbd2a165c37cb4ec68e1f994716ffb15c	eap-xp1-openj9-11-openshift-rhel8@sha256:d6a7bdbf4726fe0e0e54c0bce9b2257bbd2a165c37cb4ec68e1f994716ffb15c
registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833	eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c1992663258c0016c80247f507fd1c39cf9a73da833
registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0	eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b46ade182d0f981362f36b7e6fb9c7d8531ac639fe0
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:8e6c7874247053df431c25552c6e2edb050b2627ae21907149f419e0d9909135	sso74-openj9-openshift-rhel8@sha256:8e6c7874247053df431c25552c6e2edb050b2627ae21907149f419e0d9909135
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:ec6801343eb1ca085154d8d7481552f2e9debc414125413d25e42216aa5922af	sso74-openshift-rhel8@sha256:ec6801343eb1ca085154d8d7481552f2e9debc414125413d25e42216aa5922af

<source-image>	<target-image>
registry.redhat.io/rhel8/postgresql-96@sha256:fdc2398a25530547354714f2538c691d91b700e0cedef5361a3e7d96ddfd4e11	**postgresql-96@sha256:fdc2398a25530547354714f2538c691d91b700e0cedef5361a3e7d96ddfd4e11**
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	**mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73**
registry.redhat.io/ubi8-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6	**ubi8-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6**

3. イメージに同じダイジェストがあることを確認します。

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

4. ダイジェストは、異なる場合に明示的に設定します。

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

関連情報

- イメージ一覧のソースを見つけるには、CodeReady Workspaces Operator ClusterServiceVersion ソースの **relatedImages** 属性の値を参照してください。

5.3.3. 制限された環境での CLI 管理ツールを使用した CodeReady Workspaces のアップグレード

本セクションでは、制限された環境で CLI 管理ツールを使用して Red Hat CodeReady Workspaces をアップグレードする方法を説明します。

前提条件

- OpenShift インスタンスの管理者アカウント
- **<openshift-workspaces>** プロジェクトの crwctl **--installer operator** メソッドを使用してインストールされた、CLI 管理ツールでインストールされた Red Hat CodeReady Workspaces の実行中のインスタンスバージョン 2.3。「制限された環境での CodeReady Workspaces のインストール」を参照してください。
- 必須のコンテナーイメージはクラスターで実行される CodeReady Workspaces サーバーで使用できる。「プライベートレジストリーの準備」 を参照してください。
- **crwctl** 2.4 管理ツールが利用できる。「crwctl CLI 管理ツールのインストール」 を参照してください。

手順

1. CodeReady Workspaces 2.3 インスタンスで実行されているすべてのワークスペースで、変更を保存し、Git リポジトリーに再度プッシュします。
2. CodeReady Workspaces 2.3 インスタンスのすべてのワークスペースを停止します。
3. 次のコマンドを実行します。

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

   - **<image-registry>**: 制限された環境でアクセス可能なコンテナーイメージレジストリーのホスト名およびポート。
   - **<organization>**: container-image レジストリーの組織。「プライベートレジストリーの準備」 を参照してください。

検証手順

1. CodeReady Workspaces インスタンスに移動します。
2. 2.4 のバージョン番号がページ下部に表示されます。

注記

低速なシステムまたはインターネット接続の場合は、**--k8spodwaittimeout=1800000** フラグオプションを **crwctl server:update** コマンドに追加し、Pod のタイムアウト期間を 1800000 ms 以上に拡張します。

第6章 CODEREADY WORKSPACES のアンインストール

本セクションでは、Red Hat CodeReady Workspaces のアンインストール手順を説明します。アンインストールプロセスでは、CodeReady Workspaces 関連のユーザーデータが完全に削除されます。CodeReady Workspaces インスタンスをインストールするために以前に使用された方法により、アンインストール方法が決まります。

- OperatorHub を使用してインストールされた CodeReady Workspaces の場合、OpenShift Web コンソールの方法については、「OpenShift Web コンソールを使用した OperatorHub のインストール後の CodeReady Workspaces のアンインストール」を参照してください。
- CLI の方法を使用してインストールされた CodeReady Workspaces の場合は、「OpenShift CLI を使用した OperatorHub のインストール後の CodeReady Workspaces のアンインストール」を参照してください。
- crwctl を使用してインストールされた CodeReady Workspaces の場合は、「crwctl インストール後の CodeReady Workspaces のアンインストール」を参照してください。

6.1. OPENSHIFT WEB コンソールを使用した OPERATORHUB のインストール後の CODEREADY WORKSPACES のアンインストール

本セクションでは、OpenShift Administrator パースペクティブのメインメニューを使用して、クラスターから CodeReady Workspaces をアンインストールする方法を説明します。

前提条件

- CodeReady Workspaces が OperatorHub を使用して OpenShift クラスターにインストールされている。

手順

1. OpenShift Web コンソールに移動し、Administrator パースペクティブを選択します。
2. **Home > Projects** セクションで、CodeReady Workspaces インスタンスが含まれるプロジェクトに移動します。

   ヒント

   デフォルトのプロジェクト名は **<openshift-workspaces>** です。

3. **Operators > Installed Operators** セクションで、インストールされた Operator の一覧で **Red Hat CodeReady Workspaces** をクリックします。
4. **Red Hat CodeReady Workspaces Cluster** タブで、表示された Red Hat CodeReady Workspaces Cluster をクリックし、右上の **Actions** ドロップダウンメニューで **Delete cluster** オプションを選択します。

   ヒント

   デフォルトの Red Hat CodeReady Workspaces クラスター名は **<red-hat-codeready-workspaces>** です。

5. **Operators > Installed Operators** セクションの、インストールされた Operator 一覧で **Red Hat CodeReady Workspaces** をクリックし、右上の **Actions** ドロップダウンメニューで **Uninstall Operator** オプションを選択します。

6. **Home > Projects** セクションで、CodeReady Workspaces インスタンスが含まれるプロジェクトに移動し、右上の **Actions** ドロップダウンメニューで **Delete Project** オプションを選択します。

6.2. OPENSHIFT CLI を使用した OPERATORHUB のインストール後の CODEREADY WORKSPACES のアンインストール

本セクションでは、**oc** コマンドを使用して、CodeReady Workspaces インスタンスをアンインストールする方法を説明します。

前提条件

- CodeReady Workspaces が OperatorHub を使用して OpenShift クラスターにインストールされている。
- **oc** ツールが利用可能である。

手順

以下の手順では、コマンドラインの出力を例として示します。ユーザーの端末の出力は異なる場合があることに注意してください。

クラスターから CodeReady Workspaces インスタンスをアンインストールするには、以下を実行します。

1. クラスターにサインインします。

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

2. CodeReady Workspaces インスタンスがデプロイされているプロジェクトに切り替えます。

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

3. CodeReady Workspaces クラスター名を取得します。以下は、**red-hat-codeready-workspaces** という名前のクラスターを示しています。

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

4. CodeReady Workspaces クラスターを削除します。

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

5. CodeReady Workspaces クラスターサービスバージョン (CSV) モジュールの名前を取得します。以下は、**red-hat-codeready-workspaces.v2.4** という名前の CSV モジュールを検出します。

```
$ oc get csv
```

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

6. CodeReady Workspaces CSV を削除します。

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

6.3. CRWCTL インストール後の CODEREADY WORKSPACES のアンインストール

本セクションでは、**crwctl** ツールを使用してインストールされた Red Hat CodeReady Workspaces のインスタンスをアンインストールする方法を説明します。

前提条件

- **crwctl** ツールが使用できる。
- **oc** ツールが利用可能である。
- **crwctl** ツールは OpenShift の CodeReady Workspaces インスタンスにインストールされている。

手順

1. OpenShift クラスターにサインインします。

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

2. CodeReady Workspaces namespace の名前を取得します。

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

3. **<namespace>** プロジェクトから CodeReady Workspaces インスタンスを削除します。

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

ヒント

CodeReady Workspaces インスタンスを含むプロジェクトの名前が **openshift-workspaces** の場合、**-n** 引数は必要ありません。

4. **<namespace>** プロジェクトを削除します。

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