



# Red Hat CodeReady Workspaces 2.7

## インストールガイド

Red Hat CodeReady Workspaces 2.7 のインストール



# Red Hat CodeReady Workspaces 2.7 インストールガイド

---

## Red Hat CodeReady Workspaces 2.7 のインストール

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

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Installation\_Guide.ent file |.

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

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

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

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

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

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

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

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

All other trademarks are the property of their respective owners.

## 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

多様性を受け入れるオープンソースの強化 .....	5
第1章 サポートされるプラットフォーム .....	6
第2章 CODEREADY WORKSPACES インストールの設定 .....	7
2.1. CHECLUSTER カスタムリソースについて .....	7
2.2. CHECLUSTER カスタムリソースフィールドの参照 .....	7
第3章 CODEREADY WORKSPACES のインストール .....	23
3.1. OPERATORHUB を使用した OPENSIFT 4 への CODEREADY WORKSPACES のインストール .....	23
3.1.1. OpenShift Web コンソールでのプロジェクトの作成 .....	23
3.1.2. Red Hat CodeReady Workspaces Operator のインストール .....	23
3.1.3. Red Hat CodeReady Workspaces Operator のインスタンスの作成 .....	24
3.2. CLI を使用した CODEREADY WORKSPACES の OPENSIFT 4 へのインストール .....	25
3.3. CODEREADY WORKSPACES の OPENSIFT CONTAINER PLATFORM 3.11 へのインストール .....	26
3.3.1. crwctl CLI 管理ツールのインストール .....	26
3.3.2. Operator を使用した CodeReady Workspaces の OpenShift 3 へのインストール .....	26
3.4. 制限された環境での CODEREADY WORKSPACES のインストール .....	28
3.4.1. OperatorHub を使用した制限された環境での CodeReady Workspaces のインストール .....	29
3.4.2. 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 .....	29
3.4.2.1. プライベートレジストリーの準備 .....	30
3.4.2.2. 制限された環境用の CodeReady Workspaces カスタムリソースの準備 .....	36
3.4.2.2.1. デフォルトの CheCluster カスタムリソースのダウンロード .....	36
3.4.2.2.2. 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 .....	36
3.4.2.3. 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 .....	37
3.4.3. プロキシ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 .....	38
第4章 CODEREADY WORKSPACES の設定 .....	39
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 .....	39
4.1.1. Operator を使用した CodeReady Workspaces サーバーの詳細設定について .....	39
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参照 .....	40
4.1.2.1. Che サーバー .....	40
4.1.2.2. 認証パラメーター .....	46
4.1.2.3. 内部 .....	48
4.1.2.4. OpenShift インフラパラメーター .....	48
4.1.2.5. OpenShift インフラパラメーター .....	64
4.1.2.6. 実験的なプロパティ .....	65
4.1.2.7. 主な「/websocket」エンドポイントの設定 .....	69
4.1.2.8. CORS 設定 .....	69
4.1.2.9. Factory のデフォルト .....	70
4.1.2.10. devfile のデフォルト .....	70
4.1.2.11. Che システム .....	72
4.1.2.12. Workspace の制限 .....	73
4.1.2.13. ユーザーワークスペースの制限 .....	74
4.1.2.14. 組織ワークスペースの制限 .....	75
4.1.2.15. 組織通知の設定 .....	75
4.1.2.16. マルチユーザー固有の OpenShift インフラストラクチャー設定 .....	76
4.1.2.17. Keycloak の設定 .....	77
4.2. プロジェクトストラテジーの設定 .....	80
4.2.1. ユーザーストラテジーごとに1つのプロジェクト .....	82
4.2.2. 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 .....	82

4.2.3. 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	83
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	83
4.2.5. 互換性のないユーザー名またはユーザー ID の処理	83
4.2.6. ユーザーのプロジェクトの事前作成	84
4.2.7. namespace のラベル付け	85
4.3. ストレージストラテジーの設定	86
4.3.1. codeready-workspaces ワークスペースのストレージストラテジー	86
4.3.1.1. common PVC ストラテジー	87
4.3.1.2. per-workspace PVC ストラテジー	88
4.3.1.3. unique PVC ストラテジー	88
4.3.1.4. サブパスが PVC で使用される方法	88
4.3.2. 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	89
4.3.2.1. Operator を使用した PVC ストラテジーの設定	89
4.4. ストレージタイプの設定	90
4.4.1. 永続ストレージ	90
4.4.2. 一時ストレージ	91
4.4.3. 非同期ストレージ	91
4.4.4. 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	92
4.4.5. 非同期ストレージ Pod のアイドリング	92
4.5. 一度に複数のワークスペースの実行	93
4.6. ワークスペース公開ストラテジーの設定	93
4.6.1. Operator を使用したワークスペース公開ストラテジーの設定	93
4.6.2. ワークスペース公開ストラテジー	95
4.6.2.1. Multi-host ストラテジー	95
4.6.2.2. 単一ホストストラテジー	95
4.6.2.2.1. devfile エンドポイント: single-host	96
4.6.2.2.2. devfile エンドポイント: multi-host	96
4.6.3. セキュリティーに関する考慮事項	96
4.6.3.1. JSON Web トークン (JWT) プロキシ	97
4.6.3.2. 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	97
4.6.3.3. セキュリティー保護されたコンテナイメージコンポーネント	97
4.6.3.4. クロスサイトリクエストフォージェリー攻撃	97
4.6.3.5. フィッシング攻撃	98
4.7. ワークスペース NODESELECTOR の設定	98
4.8. RED HAT CODEREADY WORKSPACES サーバーのホスト名の設定	98
4.9. OPENSIFT ROUTE のラベルの設定	100
4.10. OPENSIFT ROUTE のラベルの設定	100
4.11. 自己署名証明書を使用した GIT リポジトリをサポートする CODEREADY WORKSPACES のデプロイ	101
4.12. ストレージクラスを使用した CODEREADY WORKSPACES のインストール	102
4.13. 信頼できない TLS 証明書の CODEREADY WORKSPACES へのインポート	106
4.13.1. 新規 CA 証明書の CodeReady Workspaces への追加	107
4.13.2. CodeReady Workspaces のインストールレベルでの検証	108
4.13.3. ワークスペースレベルでの検証	108
4.14. コンポーネント間の通信での外部 DNS 名と内部 DNS 名間の切り替え	110
4.15. RED HAT CODEREADY WORKSPACES ログインページの RH-SSO CODEREADY-WORKSPACES-USERNAME-READONLY テーマの設定	110
4.15.1. RH-SSO へのログイン	111
4.15.2. RH-SSO codeready-workspaces-username-readonly テーマの設定	111
4.16. シークレットをファイルまたは環境変数として RED HAT CODEREADY WORKSPACES コンテナにマウントする	112
4.16.1. シークレットをファイルとして Red Hat CodeReady Workspaces コンテナにマウントする	112
4.16.2. シークレットを環境変数として Red Hat CodeReady Workspaces コンテナにマウントする	113

---

<b>第5章 CODEREADY WORKSPACES のアップグレード .....</b>	<b>116</b>
5.1. OPERATORHUB を使用した CODEREADY WORKSPACES のアップグレード	116
5.1.1. OperatorHub での CodeReady Workspaces の承認ストラテジーの指定	116
5.1.2. OperatorHub での CodeReady Workspaces の手動によるアップグレード	117
5.2. CLI 管理ツールを使用した CODEREADY WORKSPACES のアップグレード	118
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	118
5.3.1. 制限された環境でのネットワーク接続について	119
5.3.2. オフラインレジストリーイメージのビルド	119
5.3.2.1. オフラインの devfile レジストリーイメージのビルド	119
5.3.2.2. オフラインプラグインレジストリーイメージのビルド	120
5.3.3. プライベートレジストリーの準備	121
5.3.4. 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	127
<b>第6章 CODEREADY WORKSPACES のアンインストール .....</b>	<b>129</b>
6.1. OPENSIFT 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	129
6.2. OPENSIFT 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	130
6.3. CRWCTL インストール後の CODEREADY WORKSPACES のアンインストール	131





## 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#)をご覧ください。

## 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 4.6、3.11、および OpenShift Dedicated での CodeReady Workspaces 2.7 の可用性およびサポートされるインストール方法について説明します。

表1.1 OpenShift Container Platform および OpenShift Dedicated での CodeReady Workspaces 2.7 の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)	<b>crwctl</b>
OpenShift Container Platform 4.6	AMD64 および Intel 64 (x86_64)	OperatorHub、 <b>crwctl</b>
OpenShift Container Platform 4.6	IBM Z (s390x)	OperatorHub、 <b>crwctl</b>
OpenShift Container Platform 4.6	IBM Power Systems (ppc64le)	OperatorHub、 <b>crwctl</b>
OpenShift Container Platform 4.7	AMD64 および Intel 64 (x86_64)	OperatorHub、 <b>crwctl</b>
OpenShift Container Platform 4.7	IBM Z (s390x)	OperatorHub、 <b>crwctl</b>
OpenShift Container Platform 4.7	IBM Power Systems (ppc64le)	OperatorHub、 <b>crwctl</b>
OpenShift Dedicated 4.6	AMD64 および Intel 64 (x86_64)	アドオン

## 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](#)
- [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2.1 「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2.2 「CodeReady Workspaces で使用されるデータベースに関連する **CheCluster** カスタムリソース **database** 設定。」
- 表2.3 「CodeReady Workspaces で使用される認証に関連するカスタムリソース **auth** 設定。」
- 表2.4 「CodeReady Workspaces で使用される永続ストレージに関連する **CheCluster** カスタムリソース **storage** 設定。」
- 表2.5 「OpenShift の CodeReady Workspaces インストールに固有の **CheCluster** カスタムリソース **k8s** 設定。」
- 表2.6 「CodeReady Workspaces によって使用される CodeReady Workspaces メトリクスの収集に関連する **CheCluster** カスタムリソースの **metrics** 設定。」
- 表2.7 「**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	説明
-------	----

プロパティ	説明
airGapContainerRegistryHostname	イメージのプルに使用する別のコンテナレジストリーに対する、オプションのホスト名または URL。この値は、Che 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イメージで定義されるコンテナレジストリーのホスト名を上書きします。これは、制限された環境で Che をインストールする場合にとくに便利です。
airGapContainerRegistryOrganization	イメージのプルに使用する別のコンテナレジストリーのオプションのリポジトリ名。この値は、Che 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イメージで定義されるコンテナレジストリーの組織を上書きします。これは、制限された環境で CodeReady Workspaces をインストールする場合にとくに便利です。
allowUserDefinedWorkspaceNamespaces	ユーザーが OpenShift プロジェクトまたはデフォルトとは異なる OpenShift プロジェクトを指定できるように定義します。OpenShift OAuth を設定せずに <b>true</b> に設定することは推奨されていません。OpenShift インフラストラクチャーは、このプロパティも使用します。
cheClusterRoles	Che ServiceAccount に割り当てられる ClusterRole のコマ区切りの一覧。Che Operator には、これらの ClusterRole のすべてのパーミッションがすでにあり、これらを付与できる必要があることに注意してください。
cheDebug	Che サーバーのデバッグモードを有効にします。デフォルトは <b>false</b> に設定されます。
cheFlavor	インストールのバリエーションを指定します。このオプションは、アップストリームの Che インストールの場合は <b>che</b> 、CodeReady Workspaces インストールの場合は <b>codeready</b> です。デフォルト値は、必要な場合にのみ上書きします。
cheHost	インストールされた Che サーバーのパブリックホスト名。値を省略すると、値は Operator によって自動的に設定されます。 <b>cheHostTLSSecret</b> フィールドを参照してください。
cheHostTLSSecret	インストールされた Che サーバーのカスタムホスト名の Ingress またはルートのセキュリティを保護するための証明書が含まれるシークレットの名前。 <b>cheHost</b> フィールドを参照してください。

プロパティ	説明
cheImage	Che デプロイメントで使用されるコンテナイメージを上書きします。これには、コンテナ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。
cheImagePullPolicy	Che 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。デフォルト値は <b>nightly</b> または <b>latest</b> イメージの場合は <b>Always</b> であり、その他の場合は <b>IfNotPresent</b> です。
cheImageTag	Che デプロイメントで使用されるコンテナ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。
cheLogLevel	Che サーバーのログレベル: <b>INFO</b> または <b>DEBUG</b> 。デフォルトは <b>INFO</b> に設定されます。
cheServerIngress	Che サーバー Ingress のカスタム設定。
cheServerRoute	Che サーバルートのカスタム設定。
cheWorkspaceClusterRole	Che 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。デフォルトのロールは、省略されているか、または空白のままの場合に使用されます。
customCheProperties	<b>CheCluster</b> 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、Che サーバーによって使用される生成された <b>che</b> ConfigMap に適用される追加の環境変数のマップ。 <b>customCheProperties</b> に、他の CR フィールドから <b>che</b> ConfigMap で生成されるプロパティが含まれる場合、代わりに <b>customCheProperties</b> に定義される値が使用されます。
devfileRegistryCpuLimit	devfile レジストリーのデプロイメントで使用される CPU 制限を上書きします。コア (500m = .5 コア)。デフォルトは 500m に設定されます。
devfileRegistryCpuRequest	devfile レジストリーのデプロイメントで使用される CPU 要求を上書きします。コア (500m = .5 コア)。デフォルトは 100m に設定されます。

プロパティ	説明
devfileRegistryImage	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。
devfileRegistryIngress	devfile レジストリー Ingress のカスタム設定。
devfileRegistryMemoryLimit	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。デフォルトは 256Mi に設定されます。
devfileRegistryMemoryRequest	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。デフォルトは 16Mi に設定されます。
devfileRegistryPullPolicy	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。デフォルト値は <b>nightly</b> または <b>latest</b> イメージの場合は <b>Always</b> であり、その他の場合は <b>IfNotPresent</b> です。
devfileRegistryRoute	devfile レジストリールートのカスタム設定。
devfileRegistryUrl	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必要がある場合は、この ONLY を設定します。 <b>externalDevfileRegistry</b> フィールドを参照してください。デフォルトで、これは Operator によって自動的に計算されます。
externalDevfileRegistry	専用の devfile レジストリーサーバーをデプロイするかどうかについて Operator に指示します。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。 <b>externalDevfileRegistry</b> が <b>true</b> の場合、このような専用サーバーは Operator によって起動されず、 <b>devfileRegistryUrl</b> フィールドを手動で設定する必要があります。
externalPluginRegistry	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かどうかについて Operator に指示します。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。 <b>externalPluginRegistry</b> が <b>true</b> の場合、このような専用サーバーは Operator によって起動されず、 <b>pluginRegistryUrl</b> フィールドを手動で設定する必要があります。

プロパティ	説明
gitSelfSignedCert	<b>che-git-self-signed-cert</b> ConfigMap からの証明書が有効にされている場合、これは Che コンポーネントに伝播し、Git の特定の設定が提供されます。
nonProxyHosts	<p>プロキシをバイパスして、直接到達されるホストの一覧。ワイルドカードのドメインを指定するには、以下の <b>.&lt;DOMAIN&gt;</b> および <b> </b> を区切り文字として使用します。たとえば、<b>localhost .my.host.com 123.42.12.32</b> のようになります。これは、プロキシの設定が必要な場合にのみ使用します。Operator は OpenShift クラスター全体のプロキシ設定に対応し、追加の設定は必要ありませんが、カスタムリソースで <b>nonProxyHosts</b> を定義すると、クラスタープロキシ設定からのプロキシ以外のホストと、カスタムリソースで定義されるホストをマージします。ドキュメント (<a href="https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html">https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html</a>) を参照してください。 <b>proxyURL</b> フィールドも参照してください。</p>
pluginRegistryCpuLimit	プラグインレジストリーのデプロイメントで 사용되는 CPU 制限を上書きします。コア(500m = .5 コア)。デフォルトは 500m に設定されます。
pluginRegistryCpuRequest	プラグインレジストリーのデプロイメントで 사용되는 CPU 要求を上書きします。コア(500m = .5 コア)。デフォルトは 100m に設定されます。
pluginRegistryImage	プラグインレジストリーのデプロイメントで 사용되는コンテナ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。
pluginRegistryIngress	プラグインレジストリー Ingress のカスタム設定。
pluginRegistryMemoryLimit	プラグインレジストリーのデプロイメントで 사용되는メモリ制限を上書きします。デフォルトは 256Mi に設定されます。
pluginRegistryMemoryRequest	プラグインレジストリーのデプロイメントで 사용되는メモリ要求を上書きします。デフォルトは 16Mi に設定されます。



プロパティ	説明
pluginRegistryPullPolicy	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。デフォルト値は <b>nightly</b> または <b>latest</b> イメージの場合は <b>Always</b> であり、その他の場合は <b>IfNotPresent</b> です。
pluginRegistryRoute	プラグインレジストリールートのカスタム設定。
pluginRegistryUrl	サンプルのすぐに使できる devfile を提供するプラグインレジストリーの公開 URL。外部 devfile レジストリーを使用する必要がある場合は、この ONLY を設定します。 <b>externalPluginRegistry</b> フィールドを参照してください。デフォルトで、これは Operator によって自動的に計算されます。
proxyPassword	プロキシサーバーのパスワード。プロキシ設定が必要である場合にのみ使用します。 <b>proxyURL</b> 、 <b>proxyUser</b> 、および <b>proxySecret</b> フィールドを参照してください。
proxyPort	プロキシサーバーのポート。プロキシの設定が必要な場合にのみ使用します。 <b>proxyURL</b> および <b>nonProxyHosts</b> フィールドも参照してください。
proxySecret	プロキシサーバーの <b>user</b> および <b>password</b> が含まれるシークレット。シークレットが定義されると、 <b>proxyUser</b> および <b>proxyPassword</b> は無視されます。
proxyURL	プロキシサーバーの URL (プロトコル+ホスト名)。これにより、Che サーバーおよびワークスペースコンテナの <b>JAVA_OPTS</b> および <b>https(s)_proxy</b> 変数の適切な変更が実行されます。プロキシの設定が必要な場合にのみ使用します。Operator は OpenShift クラスター全体のプロキシ設定に対応し、追加の設定は必要ありませんが、カスタムリソースで <b>proxyUrl</b> を定義すると、クラスタープロキシ設定が、カスタムリソースのフィールドの <b>proxyUrl</b> 、 <b>proxyPort</b> 、 <b>proxyUser</b> および <b>proxyPassword</b> で上書きされます。ドキュメント ( <a href="https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html">https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html</a> ) を参照してください。 <b>proxyPort</b> および <b>nonProxyHosts</b> フィールドも参照してください。
proxyUser	プロキシサーバーのユーザー名。プロキシの設定が必要な場合にのみ使用します。 <b>proxyURL</b> 、 <b>proxyPassword</b> および <b>proxySecret</b> フィールドも参照してください。

プロパティ	説明
selfSignedCert	非推奨。このフラグの値は無視されます。Che Operator は、ルーター証明書が自己署名されているかどうかを自動的に検知し、これを Che サーバーなどの他のコンポーネントに伝播します。
serverCpuLimit	Che サーバーのデプロイメントで使用される CPU 制限を上書きします (コア単位)。 (500m = .5 コア)。デフォルトは 1 に設定されます。
serverCpuRequest	Che サーバーのデプロイメントで使用される CPU 要求を上書きします (コア単位)。 (500m = .5 コア)。デフォルトは 100m に設定されます。
serverExposureStrategy	サーバーおよびワークスペースの公開タイプを設定します。使用できる値は <b>multi-host</b> 、 <b>single-host</b> 、 <b>default-host</b> です。必要なエンドポイントごとに個別の Ingress または OpenShift ルートを作成する <b>multi-host</b> にデフォルト設定されます。 <b>single-host</b> は、ワークスペースがサブパスで公開された状態で Che を単一のホスト名で公開します。この方法の制限については、関連ドキュメントを参照してください。さらに、Operator および Che サーバーが Kubernetes でこれを実行する方法の追加の設定については、 <b>singleHostExposureType</b> プロパティを参照してください。 <b>default-host</b> は、クラスターのホストで Che サーバーを公開します。この方法の制限については、関連ドキュメントを参照してください。
serverMemoryLimit	Che サーバーのデプロイメントで使用されるメモリ制限を上書きします。デフォルトは 1Gi に設定されます。
serverMemoryRequest	Che サーバーのデプロイメントで使用されるメモリ要求を上書きします。デフォルトは 512Mi に設定されます。
serverTrustStoreConfigMapName	Che サーバーの Java トラストストアに追加するパブリック証明書のある ConfigMap の名前。これは、HTTPS エンドポイントが自己署名の証明書で署名されている OpenShift OAuth プロバイダーを追加する際に必要になります。Che サーバーは、要求できるように CA 証明書を認識する必要があります。これはデフォルトで無効にされます。
singleHostGatewayConfigMapLabels	ゲートウェイ設定を表す ConfigMap に存在する必要があるラベル。

プロパティ	説明
singleHostGatewayConfigSidecarImage	ゲートウェイに設定を提供するゲートウェイサイドカーに使用されるイメージ。Operator によって提供されるデフォルトのコンテナ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。
singleHostGatewayImage	単一ホストモードでゲートウェイに使用されるイメージ。Operator によって提供されるデフォルトのコンテナ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。
tlsSupport	非推奨。Operator に対して Che を TLS モードでデプロイするように指示します。これはデフォルトで有効になります。TLS を無効にすると、Che コンポーネントが正しく機能しないことがあります。
useInternalClusterSVCNames	内部クラスターの SVC 名を使用してコンポーネント間の通信を行い、トラフィックを加速し、プロキシの問題を回避します。デフォルト値は <b>false</b> です。
workspaceNamespaceDefault	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デフォルトの OpenShift プロジェクトを定義します。che-workspace- <code>&lt;username&gt;</code> などの <code>&lt;username&gt;</code> 、 <code>&lt;userid&gt;</code> 、および <code>&lt;workspaceid&gt;</code> プレースホルダーを使用できます。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。

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

プロパティ	説明
chePostgresContainerResources	PostgreSQL コンテナのカスタム設定
chePostgresDb	Che サーバーが DB への接続に使用する PostgreSQL データベース名。デフォルトは <b>dbche</b> に設定されます。
chePostgresHostName	Che サーバーが接続する PostgreSQL データベースのホスト名。デフォルトは <b>postgres</b> です。外部データベースを使用する場合、この値のみを上書きします。 <b>externalDb</b> フィールドを参照してください。デフォルトでは、これは Operator によって自動的に設定されます。

プロパティ	説明
chePostgresPassword	Che サーバーが DB への接続に使用する PostgreSQL パスワード。これは、省略されるか、または空のままの場合は、自動的に生成される値に設定されます。
chePostgresPort	Che サーバーが接続する PostgreSQL データベースのポート。デフォルトは 5432 に設定されます。外部データベースを使用する場合、この値のみを上書きします。 <b>externalDb</b> フィールドを参照してください。デフォルトでは、これは Operator によって自動的に設定されます。
chePostgresSecret	Che サーバーが DB への接続に使用する PostgreSQL の `user` および <b>password</b> が含まれるシークレット。シークレットが定義されると、 <b>chePostgresUser</b> および <b>chePostgresPassword</b> は無視されます。値が省略されるか、空のままにすると、以下のいずれかのシナリオが適用されます。1. <b>chePostgresUser</b> および <b>chePostgresPassword</b> が定義され、それらは DB への接続に使用されます。2. <b>chePostgresUser</b> または <b>chePostgresPassword</b> が定義されていない場合には、 <b>che-postgres-secret</b> という名前の新しいシークレットが <b>user</b> のデフォルト値 <b>pgche</b> で作成され、 <b>password</b> の自動生成される値で作成されます。
chePostgresUser	Che サーバーが DB への接続に使用する PostgreSQL ユーザー。デフォルトは <b>pgche</b> に設定されます。
externalDb	専用のデータベースをデプロイするかどうかについて Operator に指示します。デフォルトでは、専用の PostgreSQL データベースは Che インストールの一部としてデプロイされます。 <b>externalDb</b> が <b>true</b> の場合、専用データベースは Operator によってデプロイされず、使用する外部 DB への接続の詳細を指定する必要があります。 <b>chePostgres</b> で始まるすべてのフィールドも参照してください。
postgresImage	PostgreSQL データベースのデプロイメントで使われるコンテナ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	説明
postgresImagePullPolicy	PosgreSQL データベースのデプロイメントで使われるイメージプルポリシーを上書きします。デフォルト値は <b>nightly</b> または <b>latest</b> イメージの場合は <b>Always</b> であり、その他の場合は <b>IfNotPresent</b> です。

表2.3 CodeReady Workspaces で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	説明
externalIdentityProvider	専用のアイデンティティプロバイダー (Keycloak または RH-SSO インスタンス) をデプロイするかどうかについて Operator に指示します。デフォルトで、専用のアイデンティティプロバイダーサーバーは Che インストールの一部としてデプロイされます。 <b>externalIdentityProvider</b> が <b>true</b> の場合、専用のアイデンティティプロバイダーは Operator によってデプロイされず、使用する予定の外部アイデンティティプロバイダーについての詳細情報を独自に提供する必要があります。 <b>identityProvider</b> 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	アイデンティティプロバイダーの管理者ユーザーの名前を上書きします。デフォルトは <b>admin</b> に設定されます。
identityProviderClientId	Che に使用されるアイデンティティプロバイダー、Keycloak または RH-SSO の <b>client-id</b> の名前。外部アイデンティティプロバイダーが使用されている場合にこれを上書きします。 <b>externalIdentityProvider</b> フィールドを参照してください。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 <b>-public</b> が付けられた <b>flavour</b> フィールドの値に設定されます。
identityProviderContainerResources	アイデンティティプロバイダーコンテナのカスタム設定。
identityProviderImage	アイデンティティプロバイダー、Keycloak、または RH-SSO デプロイメントで使用するコンテナ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	説明
identityProviderImagePullPolicy	アイデンティティプロバイダー、Keycloak、または 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。デフォルト値は <b>nightly</b> または <b>latest</b> イメージの場合は <b>Always</b> であり、その他の場合は <b>IfNotPresent</b> です。
identityProviderIngress	Ingress のカスタム設定。
identityProviderPassword	Keycloak 管理者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プロバイダーが使用されている場合にこれを上書きします。 <b>externalIdentityProvider</b> フィールドを参照してください。省略されているか、または空のままにすると、自動生成されたパスワードに設定されます。
identityProviderPostgresPassword	データベースに接続するために使用するアイデンティティプロバイダー、Keycloak、または RH-SSO のパスワード外部アイデンティティプロバイダーが使用されている場合にこれを上書きします。 <b>externalIdentityProvider</b> フィールドを参照してください。省略されているか、または空のままにすると、自動生成されたパスワードに設定されます。
identityProviderPostgresSecret	データベースに接続するために使用するアイデンティティプロバイダー、Keycloak、または RH-SSO の <b>password</b> が含まれるシークレット。シークレットが定義されると、 <b>identityProviderPostgresPassword</b> は無視されます。値が省略されるか、空のままにすると、以下のいずれかのシナリオが適用されます。 1. <b>identityProviderPostgresPassword</b> が定義され、これがデータベースへの接続に使用されます。 2. <b>identityProviderPostgresPassword</b> が定義されていない場合、 <b>che-identity-postgres-secret</b> という名前の新規シークレットが <b>password</b> の自動生成された値で作成されます。
identityProviderRealm	Che に使用されるアイデンティティプロバイダー、Keycloak、または RH-SSO のレルムの名前。外部アイデンティティプロバイダーが使用されている場合にこれを上書きします。 <b>externalIdentityProvider</b> フィールドを参照してください。省略されているか、または空のままの場合、これは <b>flavour</b> フィールドの値に設定されます。
identityProviderRoute	ルートのカスタム設定。

プロパティ	説明
identityProviderSecret	<p>アイデンティティプロバイダーの <b>user</b> および <b>password</b> が含まれるシークレット。シークレットが定義される</p> <p>と、<b>identityProviderAdminUserName</b> および <b>identityProviderPassword</b> は無視されます。値が省略されるか、空のままにすると、以下のいずれかのシナリオが適用されます。</p> <p>1.<b>identityProviderAdminUserName</b> また、<b>identityProviderPassword</b> が定義されてから使用されます。</p> <p>2.<b>identityProviderAdminUserName</b> または、<b>identityProviderPassword</b> が定義されていない場合、<b>che-identity-secret</b> という名前の新しいシークレットが <b>user</b> のデフォルト値の <b>admin</b> と、<b>password</b> の自動生成される値で作成されます。</p>
identityProviderURL	<p>アイデンティティプロバイダーサーバー (Keycloak/RH-SSO サーバー) の公開 URL。外部アイデンティティプロバイダーを使用する必要がある場合は、これのみを設定します。<b>externalIdentityProvider</b> フィールドを参照してください。デフォルトで、これは Operator によって自動的に計算され、設定されます。</p>
oAuthClientName	<p>OpenShift 側でアイデンティティフェデレーションを設定するために使用される OpenShift <b>OAuthClient</b> リソースの名前。空のままにすると自動生成されます。<b>OpenShifttoAuth</b> フィールドも参照してください。</p>
oAuthSecret	<p>OpenShift 側でアイデンティティフェデレーションを設定するために使用される OpenShift <b>OAuthClient</b> リソースに設定されたシークレットの名前。空のままにすると自動生成されます。<b>OAuthClientName</b> フィールドも参照してください。</p>
openShifttoAuth	<p>アイデンティティプロバイダー (Keycloak/RHSSO) と OpenShift OAuth の統合を有効にします。デフォルトでは OpenShift の値は空になります。これにより、ユーザーは OpenShift ログインで OpenShift ユーザーとして直接ログインでき、独自のワークスペースを個人の OpenShift namespace の下に作成できます。警告: <b>kubeadmin</b> ユーザーはサポートされておらず、これを使用してログインしても Che Dashboard にはアクセスできません。</p>

プロパティ	説明
updateAdminPassword	デフォルトの <b>admin</b> Che ユーザーの初回ログイン時のパスワードの更新を強制します。デフォルトは <b>false</b> に設定されます。

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

プロパティ	説明
postgresPVCStorageClassName	PosgreSQL 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。省略されるか、または空のままの場合は、デフォルトのストレージクラスが使用されます。
preCreateSubPaths	Che 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。デフォルトは <b>false</b> に設定されますが、OpenShift クラスターの設定に応じてこれを有効にする必要があります。
pvcClaimSize	ワークスペースの永続ボリューム要求 (PVC) のサイズ。デフォルトは <b>1Gi</b> に設定されます。
pvcJobsImage	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。 <b>preCreateSubPaths</b> フィールドも参照してください。
pvcStrategy	Che サーバーの永続ボリューム要求ストラテジー。これには、'common' (1つのボリュームにすべてのワークスペース PVC)、 <b>per-workspace</b> (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 <b>unique</b> (宣言されたボリュームごとに1つの PVC) を指定できます。デフォルトは <b>common</b> に設定されます。
workspacePVCStorageClassName	Che 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。省略されるか、または空のままの場合は、デフォルトのストレージクラスが使用されます。

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



プロパティ	説明
ingressClass	Ingress を管理するコントローラーを定義する Ingress クラス。デフォルトは <b>nginx</b> に設定されます。注意: これは、Che 関連の Ingress で <b>kubernetes.io/ingress.class</b> アノテーションを実行します。
ingressDomain	OpenShift クラスターのグローバル Ingress ドメイン。これは明示的に指定する必要があります。デフォルト値はありません。
ingressStrategy	Ingress 作成のストラテジー。オプション: <b>multi-host</b> (ホストは Ingress で明示的に指定されます)、 <b>single-host</b> (ホストは指定される、パスペースのルール)、および <b>default-host</b> (ホストは指定されない、パスペースのルール)。 <b>server</b> セクションの <b>serverExposureStrategy</b> が優先されるため、デフォルトで <b>multi-host</b> が非推奨になりました。これは、クラスタータイプに関係なく定義されます。両方が定義されている場合は、 <b>serverExposureStrategy</b> オプションが優先されます。
securityContextFsGroup	Che Pod およびワークスペース Pod コンテナが実行される FSGroup。デフォルト値は <b>1724</b> です。
securityContextRunAsUser	Che Pod およびワークスペース Pod コンテナの実行に使用するユーザーの ID。デフォルト値は <b>1724</b> です。
singleHostExposureType	serverExposureStrategy が <b>single-host</b> に設定されている場合、サーバー、レジストリー、およびワークスペースを公開する方法は、このプロパティによって追加で設定されます。使用できる値は <b>native</b> です。つまり、サーバーおよびワークスペースは K8s の ingress、またはサーバーおよびワークスペースが <b>Traefik</b> をベースとするカスタムゲートウェイを使用して公開される <b>gateway</b> を使用して公開されます。Ingress またはゲートウェイ <b>route</b> でサポートされる場合でも、すべてのエンドポイントは常に同じドメインのサブパスを参照します。デフォルトは <b>native</b> に設定されます。
tlsSecretName	TLS が有効にされている場合に 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。フィールドが空の文字列である場合、デフォルトのクラスター証明書が使用されます。 <b>tlsSupport</b> フィールドも参照してください。

表2.6 CodeReady Workspaces によって使用される CodeReady Workspaces メトリクスの収集に関連する CheCluster カスタムリソースの metrics 設定。

プロパティ	説明
enable	<b>metrics</b> Che サーバーエンドポイントを有効にします。デフォルトは <b>true</b> に設定されます。

表2.7 **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	説明
cheClusterRunning	Che インストールのステータス。 <b>Available</b> 、 <b>Unavailable</b> 、または <b>Available, Rolling Update in Progress</b> を指定できます。
cheURL	Che サーバーへの公開 URL。
cheVersion	現在のインストールされている Che 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。
gitHubOAuthProvisioned	アイデンティティプロバイダーインスタンス、Keycloak または RH-SSO が GitHub 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。
helpLink	現在の Operator ステータスに関連するヘルプの検索に使用する URL を参照する URL。
keycloakProvisioned	アイデンティティプロバイダーインスタンス、Keycloak または 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	アイデンティティプロバイダーサーバー (Keycloak/RH-SSO) の公開 URL。
message	Pod がこの状態にある理由の詳細を示す、人が判読できるメッセージ。
openShiftOAuthProvisioned	アイデンティティプロバイダーインスタンス、Keycloak または 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を示す簡単な 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 OPENSIFT 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。

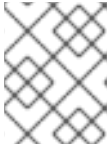
### 3.1.3. Red Hat CodeReady Workspaces Operator のインスタンスの作成

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/2.7/html-single/end-user\\_guide/index#navigating-codeready-workspaces-using-the-dashboard\\_crw](https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.7/html-single/end-user_guide/index#navigating-codeready-workspaces-using-the-dashboard_crw).
- [https://access.redhat.com/documentation/en-us/red\\_hat\\_codeready\\_workspaces/2.7/html-single/administration\\_guide/index#viewing-the-state-of-the-codeready-workspaces-cluster-deployment-using-openshift-4-cli-tools\\_crw](https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.7/html-single/administration_guide/index#viewing-the-state-of-the-codeready-workspaces-cluster-deployment-using-openshift-4-cli-tools_crw).

## 3.2. CLI を使用した CODEREADY WORKSPACES の OPENSIFT 4 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 4 に CodeReady Workspaces をインストールする方法を説明します。

#### 前提条件

- OpenShift クラスターと管理者アカウント。

- **oc** が利用できる。「[Getting started with the OpenShift CLI](#)」を参照してください。**oc** バージョンは OpenShift クラスターのバージョンと一致する必要があります。
- OpenShift にログインしている。「[Logging in to the CLI](#)」を参照してください。
- **crwctl** 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

- **server:deploy** コマンドを実行して、CodeReady Workspaces インスタンスを作成します。

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

#### 検証手順

1. **server:deploy** コマンドの出力は以下で終わります。

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

2. CodeReady Workspaces クラスターインスタンス ([https://codeready-  
<openshift\\_deployment\\_name>.<domain\\_name>](https://codeready-openshift_deployment_name.domain_name)) に移動します。

## 3.3. CODEREADY WORKSPACES の OPENSIFT CONTAINER PLATFORM 3.11 へのインストール

### 3.3.1. 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 2.7 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。

### 3.3.2. 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。

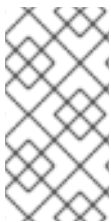


### 注記

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。



### 注記

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。

- openshift-workspaces プロジェクトで以下を実行します。

```
$ crwctl server:deploy -n openshift-workspaces -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:deploy -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:deploy has completed successfully.
```

2. CodeReady Workspaces クラスターインスタンス ([https://codeready-  
<openshift\\_deployment\\_name>.<domain\\_name>](https://codeready-openshift_deployment_name.<domain_name>)) に移動します。

## 3.4. 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ポリシーを考慮して、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が必要な場合は、「[プロキシ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。

### 3.4.1. 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。

### 3.4.2. 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。

### 3.4.2.1. 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

<b>&lt;source-image&gt;</b>	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)。
<b>&lt;target-registry&gt;</b>	ターゲットコンテナイメージレジストリーのホスト名およびポート。
<b>&lt;target-organization&gt;</b>	ターゲットのコンテナイメージレジストリー内の組織
<b>&lt;target-image&gt;</b>	ターゲットのコンテナイメージレジストリーのイメージ名とダイジェスト。
<b>&lt;target-user&gt;</b>	ターゲットのコンテナイメージレジストリーのユーザー名。
<b>&lt;target-password&gt;</b>	ターゲットのコンテナ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>
```

## 注記

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	<b>stacks-</b> 、 <b>plugin-</b> または <b>-openj9-</b> ではない
Workspaces	<b>stacks-</b> 、 <b>plugin-</b>
IBM Z および IBM Power Systems	<b>-openj9-</b>

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/configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac	configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55	devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7	jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd	machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99	plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfb9da505d715bd75	plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfb9da505d715bd75
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfb9da505d715bd75	plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfb9da505d715bd75
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfb9da505d715bd75	plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cdb2e3d9bbca51db128c7289bcfb9da505d715bd75
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763b431744790a89acd5c6d197316b694693b58c19770	plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763b431744790a89acd5c6d197316b694693b58c19770
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f990e9fab9baa47f659ef9450b7881072e3cb40399	plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f990e9fab9baa47f659ef9450b7881072e3cb40399
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a950cc3cbe21ed7301c65f84bb509fa2b6e71fa81d	plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a950cc3cbe21ed7301c65f84bb509fa2b6e71fa81d
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6ccdb8c303f32d046f7eccf9f452eb30e615f2a0bf0e	pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6ccdb8c303f32d046f7eccf9f452eb30e615f2a0bf0e
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505a0fa6f96969cf8cc4aa8c4db95b40626f95854a525	pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505a0fa6f96969cf8cc4aa8c4db95b40626f95854a525

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:c9f48f247cff27280587aeff54cea5d8a27e0eb55c99a73726cd7d575db7fbcc	pluginregistry-rhel8@sha256:c9f48f247cff27280587aeff54cea5d8a27e0eb55c99a73726cd7d575db7fbcc
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57	server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb	stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2	stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0	stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db	stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3	theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0	theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed	traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c 1992663258c0016c80247f507fd1c39cf9a73 da833	eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c 1992663258c0016c80247f507fd1c39cf9a73 da833
registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b4 6ade182d0f981362f36b7e6fb9c7d8531ac6 39fe0	eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b4 6ade182d0f981362f36b7e6fb9c7d8531ac6 39fe0
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f	sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f	sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f

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

## 検証手順

- 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>
```

## 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。

### 3.4.2.2. 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。

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

## 手順

- 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。
- 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。

#### 3.4.2.2.2. 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

- 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: '<target-registry>'
    airGapContainerRegistryOrganization: '<target-organization>'
# [...]
```

### 3.4.2.3. 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

- OpenShift Container Platform にログインします。

```
$ oc login ${OPENSSHIFT_API_URL} --username ${OPENSSHIFT_USERNAME} \
  --password ${OPENSSHIFT_PASSWORD}
```

- 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=<target-registry>/<target-organization>/crw-2-rhel8-operator:2.7 \
  --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** フラグオプションを **crwctl server:start** 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。

### 3.4.3. プロキシ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、プロキシ設定では通常、プロキシを使用せずに 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、いくつかのユース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、特定のユース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](#)」
- 「[OpenShift Route のラベルの設定](#)」
- 「[OpenShift Route のラベルの設定](#)」
- 「[自己署名証明書を使用した Git リポジトリをサポートする 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](#)」
- 「[信頼できない TLS 証明書の CodeReady Workspaces へのインポート](#)」
- 「[コンポーネント間の通信での外部 DNS 名と内部 DNS 名間の切り替え](#)」
- 「[Red Hat CodeReady Workspaces ログインページの RH-SSO codeready-workspaces-username-readonly テーマの設定](#)」
- 「[シークレットをファイルまたは環境変数として Red Hat CodeReady Workspaces コンテナにマウントする](#)」

### 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** プロパティを **customCheProperties** に追加します。

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# [...]
spec:
  server:
    # [...]
    customCheProperties:
      CHE_WORKSPACE_DEFAULT_MEMORY_LIMIT_MB: "2048"
    # [...]
```



#### 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について説明します。

#### 4.1.2.1. Che サーバー

表4.1 Che サーバー

環境変数名	デフォルト値	説明
<b>CHE_DATABASE</b>	<b>`\${che.home}/storage`</b>	CodeReady Workspaces が内部データオブジェクトを保存するフォルダー。

環境変数名	デフォルト値	説明
<b>CHE_API</b>	<b>http://\${CHE_HOST}:\${CHE_PORT}/api</b>	API サービス。ブラウザーは、この URL を使用して CodeReady Workspaces サーバーへの REST 通信を開始します。
<b>CHE_API_INTERNAL</b>	<b>http://\${CHE_HOST}:\${CHE_PORT}/api</b>	API サービスの内部ネットワーク URL。バックエンドサービスは、この URL を使用した CodeReady Workspaces サーバーへの REST 通信を開始する必要があります。
<b>CHE_WEBSOCKET_ENDPOINT</b>	<b>ws://\${CHE_HOST}:\${CHE_PORT}/api/websocket</b>	CodeReady Workspaces websocket の主なエンドポイント。主な websocket の対話と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。
<b>CHE_WORKSPACE_PROJECTS_STORAGE</b>	<b>/projects</b>	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。
<b>CHE_WORKSPACE_PROJECTS_STORAGE_DEFAULT_SIZE</b>	<b>1Gi</b>	devfile 要求の OpenShift 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 ('unique' および 'per workspace' 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は、これは <b>che.infra.kubernetes.pvc.quantity</b> プロパティの値で書き換えられます)。
<b>CHE_WORKSPACE_LOGS_ROOT_DIR</b>	<b>/workspace_logs</b>	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。環境変数などの値として、この値をマシンに指定します。これは、エージェントの開発者がこのディレクトリーを使用してエージェントのログをバックアップできるようにするためのものです。
<b>CHE_WORKSPACE_HTTP_PROXY</b>		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を設定します。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_HTTPS_PROXY</b>		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を設定します。
<b>CHE_WORKSPACE_NO_PROXY</b>		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を設定します。
<b>CHE_WORKSPACE_AUTO_START</b>	<b>true</b>	デフォルトでは、ユーザーがこの URL を使用してワークスペースにアクセスすると、ワークスペースは自動的に起動します (現時点で停止している場合)。この動作を無効にするには、このパラメーターを <b>false</b> に設定します。
<b>CHE_WORKSPACE_POOL_TYPE</b>	<b>fixed</b>	ワークスペーススレッドプールの設定。このプールは、非同期の実行が必要なワークスペース関連の操作 (例: 起動/停止) に使用されます。使用できる値は <b>fixed</b> および <b>cached</b> です。
<b>CHE_WORKSPACE_POOL_EXACT_SIZE</b>	<b>30</b>	プールタイプが <b>fixed</b> と異なる場合に、このプロパティは無視されます。これはプールのサイズを設定します。これが設定すると、 <b>multiplier</b> プロパティは無視されます。このプロパティが設定されていない場合 ( <b>0</b> 、 <b>&lt;0</b> 、 <b>NULL</b> )、プールサイズはコア数と等しくなります。 <b>che.workspace.pool.core_s_multiplier</b> も参照してください。
<b>CHE_WORKSPACE_POOL_CORES_MULTIPLIER</b>	<b>2</b>	プールタイプが <b>fixed</b> に設定されておらず、 <b>che.workspace.pool.exact_size</b> が設定される場合に無視されます。設定される場合、プールサイズは <b>N_CORES * multiplier</b> になります。
<b>CHE_WORKSPACE_PROBE_POOL_SIZE</b>	<b>10</b>	このプロパティは、ワークスペースサーバーの liveness プロブに使用するスレッドの数を指定します。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_HTTP_PROXY__JAVA__OPTIONS</b>	NULL	ワークスペース JVM の HTTP プロキシ設定。
<b>CHE_WORKSPACE_JAVA__OPTIONS</b>	-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	ワークスペースで実行されている JVM に追加される Java コマンドラインオプション。
<b>CHE_WORKSPACE_MAVEN__OPTIONS</b>	-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。
<b>CHE_WORKSPACE_DEFAULT_MEMORY_LIMIT_MB</b>	1024	環境に 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。
<b>CHE_WORKSPACE_DEFAULT_MEMORY_REQUEST_MB</b>	200	環境内に明示的な RAM 設定のない各コンテナの RAM 要求。この量はワークスペースコンテナの作成時に割り当てられます。このプロパティは、すべてのインフラストラクチャー実装でサポートされる訳ではありません。現時点で、これは OpenShift によってサポートされます。メモリー制限を超えるメモリー要求は無視され、制限サイズのみが使用されます。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。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_DEFAULT_CPU_LIMIT_CORES</b>	<b>-1</b>	環境に CPU 設定のない各コンテナの CPU 制限。浮動小数点のコア数 (例: <b>0.125</b> ) で、または OpenShift 形式 ( <b>125m</b> などの整数のミリコア数) を使用して指定します。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。
<b>CHE_WORKSPACE_DEFAULT_CPU_REQUEST_CORES</b>	<b>-1</b>	環境内に CPU 設定のない各コンテナの CPU 要求。CPU 制限を超える CPU 要求は無視され、制限の数値のみが使用されます。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。
<b>CHE_WORKSPACE_SIDECAR_DEFAULT_MEMORY_LIMIT_MB</b>	<b>128</b>	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。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。
<b>CHE_WORKSPACE_SIDECAR_DEFAULT_MEMORY_REQUEST_MB</b>	<b>64</b>	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。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。
<b>CHE_WORKSPACE_SIDECAR_DEFAULT_CPU_LIMIT_CORES</b>	<b>-1</b>	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: <b>0.125</b> ) で、または OpenShift 形式 ( <b>125m</b> などの整数のミリコア数) を使用して指定します。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。
<b>CHE_WORKSPACE_SIDECAR_DEFAULT_CPU_REQUEST_CORES</b>	<b>-1</b>	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: <b>0.125</b> ) で、または OpenShift 形式 ( <b>125m</b> などの整数のミリコア数) を使用して指定します。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。



環境変数名	デフォルト値	説明
CHE_WORKSPACE_SIDECA R_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 <b>Always</b> 、 <b>Never</b> 、 <b>IfNotPresen t</b> です。その他の値について は、 <b>Always</b> は <b>:latest</b> タグが付 いたイメージに、その他の場合は <b>IfNotPresent</b> が想定されます。
CHE_WORKSPACE_ACTIVIT Y__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_ACTIVIT Y__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_ACTIVIT Y__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_ACTIVIT Y__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_CLEANU P__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_CLEANU P__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。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_SERVER_PING_SUCCESS_THRES HOLD</b>	<b>1</b>	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。
<b>CHE_WORKSPACE_SERVER_PING_INTERVAL_MILLIS ECONDS</b>	<b>3000</b>	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)。
<b>CHE_WORKSPACE_SERVER_LIVENESS_PROBES</b>	<b>wsagent/http,exec-agent/http,terminal,theia,jupyter,dirigible,cloud-shell,intellij</b>	liveness プロブを必要とするサーバー名の一覧
<b>CHE_WORKSPACE_STARTUP_DEBUG_LOG_LIMIT_BYTES</b>	<b>10485760</b>	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。
<b>CHE_WORKSPACE_STOP_ROLE_ENABLED</b>	<b>true</b>	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。

#### 4.1.2.2. 認証パラメーター

表4.2 認証パラメーター

環境変数名	デフォルト値	説明
<b>CHE_AUTH_USER_SELF_CREATION</b>	<b>false</b>	CodeReady Workspaces には単一のアイデンティティ実装があるため、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。
<b>CHE_AUTH_ACCESS_DENIED_ERROR_PAGE</b>	<b>/error-oauth</b>	認証エラーページアドレス

環境変数名	デフォルト値	説明
<b>CHE_AUTH_RESERVED__USER__NAMES</b>		予約済みのユーザー名
<b>CHE_OAUTH_GITHUB_CLIENTID</b>	<b>NULL</b>	GitHub OAuth を設定して、リモトリポジトリ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。
<b>CHE_OAUTH_GITHUB_CLIENTSECRET</b>	<b>NULL</b>	GitHub OAuth を設定して、リモトリポジトリ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。
<b>CHE_OAUTH_GITHUB_AUTH_URI</b>	<b>https://github.com/login/oauth/authorize</b>	GitHub OAuth を設定して、リモトリポジトリ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。
<b>CHE_OAUTH_GITHUB_TOKEN_URI</b>	<b>https://github.com/login/oauth/access_token</b>	GitHub OAuth を設定して、リモトリポジトリ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。
<b>CHE_OAUTH_GITHUB_REDIRECTURI</b>	<b>http://localhost:\${CHE_PORT}/api/oauth/callback</b>	GitHub OAuth を設定して、リモトリポジトリ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。
<b>CHE_OAUTH_OPENSHIFT_CLIENTID</b>	<b>NULL</b>	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。
<b>CHE_OAUTH_OPENSHIFT_CLIENTSECRET</b>	<b>NULL</b>	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。
<b>CHE_OAUTH_OPENSHIFT_OAUTH__ENDPOINT</b>	<b>NULL</b>	Configuration of OpenShift OAuth クライアント。OpenShift OAuth トークンの取得に使用されます。
<b>CHE_OAUTH_OPENSHIFT_VERIFY__TOKEN__URL</b>	<b>NULL</b>	Configuration of OpenShift OAuth クライアント。OpenShift OAuth トークンの取得に使用されます。

## 4.1.2.3. 内部

表4.3 内部

環境変数名	デフォルト値	説明
<b>SCHEDULE_CORE__POOL__SIZE</b>	<b>10</b>	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。
<b>DB_SCHEMA_FLYWAY_BASELINE_ENABLED</b>	<b>true</b>	DB の初期化および移行設定
<b>DB_SCHEMA_FLYWAY_BASELINE_VERSION</b>	<b>5.0.0.8.1</b>	DB の初期化および移行設定
<b>DB_SCHEMA_FLYWAY_SCRIPTS_PREFIX</b>		DB の初期化および移行設定
<b>DB_SCHEMA_FLYWAY_SCRIPTS_SUFFIX</b>	<b>.sql</b>	DB の初期化および移行設定
<b>DB_SCHEMA_FLYWAY_SCRIPTS_VERSION_SEPARATOR</b>	<b>—</b>	DB の初期化および移行設定
<b>DB_SCHEMA_FLYWAY_SCRIPTS_LOCATIONS</b>	<b>classpath:che-schema</b>	DB の初期化および移行設定

## 4.1.2.4. OpenShift インフラパラメーター

表4.4 OpenShift インフラパラメーター

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_MASTER__URL</b>		インフラが使用する OpenShift クライアントの設定
<b>CHE_INFRA_KUBERNETES_TRUST__CERTS</b>		インフラが使用する OpenShift クライアントの設定
<b>CHE_INFRA_KUBERNETES_SERVER__STRATEGY</b>	<b>multi-host</b>	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

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_EXPOSURE</b>	<b>native</b>	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。Kubernetes でのみ機能します。'gateway': リバースプロキシゲートウェイを使用してサーバーを公開します。
<b>CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE_ENDPOINT_EXPOSURE</b>	<b>multi-host</b>	single-host サーバーストラテジーで devfile エンドポイント、エンドユーザーのアプリケーションを公開する方法を定義します。これらは single-host ストラテジーに従い、サブパスで公開されるか、またはサブドメイン上で公開できます。'multi-host': サブドメインで公開される - 'single-host': サブパスで公開される
<b>CHE_INFRA_KUBERNETES_SINGLEHOST_GATEWAY_CONFIGMAP_LABELS</b>	<b>app=che,component=che-gateway-config</b>	single-host ゲートウェイを設定する ConfigMap に設定されるラベルを定義します。
<b>CHE_INFRA_KUBERNETES_INGRESS_DOMAIN</b>		プロパティー <b>che.infra.kubernetes.server_strategy</b> が <b>multi-host</b> 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
<b>CHE_INFRA_KUBERNETES_NAMESPACE</b>		<p>非推奨: このプロパティ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 プロジェクトを定義します。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。</p> <p>&lt;username&gt; および &lt;userid&gt; プレースホルダー (例: che-workspace-&lt;username&gt;) 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。</p> <p><b>che.infra.openshift.project</b> を代わりに使用します。このプロパティで参照される 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。</p>
<b>CHE_INFRA_KUBERNETES_NAMESPACE_CREATION_ALLOWED</b>	<b>true</b>	<p>CodeReady Workspaces サーバーがユーザーワークスペースの namespace/プロジェクトを作成できるかどうか、またはそれらはクラスター管理者によって手動で作成されるかどうかを示します。このプロパティは OpenShift infra によっても使用されます。</p>

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT</b>	<b>&lt;username&gt;-che</b>	<p>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デフォルトの OpenShift プロジェクトを定義します。</p> <p>&lt;username&gt;、&lt;userid&gt; および &lt;workspaceid&gt; プレースホルダー (例: che-workspace-&lt;username&gt;) を使用できます。</p> <p>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。</p> <p>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。</p>
<b>CHE_INFRA_KUBERNETES_NAMESPACE_LABEL</b>	<b>true</b>	che-server がワークスペース namespace にラベルを付けるかどうかを定義します。
<b>CHE_INFRA_KUBERNETES_NAMESPACE_LABELS</b>	<b>app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspaces-namespace</b>	<p>CodeReady Workspaces に使用される namespace/プロジェクトの検索に使用するラベルの一覧。これらは以下を実行するために使用されます。 -</p> <p><b>che.infra.kubernetes.namespace.annotations</b> と組み合わせて、ユーザー用に準備された namespace/プロジェクトを見つけます。 - アクティブに namespace に任意のワークスペースでラベルを付けます。</p>

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_NAMESPACE_ANNOTATIONS</b>	<b>che.eclipse.org/username=&lt;username&gt;</b>	<p>CodeReady Workspaces ユーザーワークスペース用に用意された namespace/プロジェクトの検索に使用するアノテーションの一覧。<b>che.infra.kubernetes.namespace.labels</b> に一致する namespace/プロジェクトのみがこれらのアノテーションに対して一致します。<b>che.infra.kubernetes.namespace.labels</b> と <b>che.infra.kubernetes.namespace.annotations</b> の両方に一致する namespace/プロジェクトは、ユーザーのワークスペースに優先して使用されます。<b>&lt;username&gt;</b> プレースホルダーを使用して、具体的なユーザーに namespace/プロジェクトを指定することができます。</p>
<b>CHE_INFRA_KUBERNETES_NAMESPACE_ALLOW_USER_DEFINED</b>	<b>false</b>	<p>ユーザーがデフォルトとは異なる OpenShift プロジェクト (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。このプロパティは OpenShift infra によっても使用されます。</p>



環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_SERVICE__ACCOUNT__NAME</b>	<b>NULL</b>	<p>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がある Kubernetes サービスアカウント名を定義します。</p> <p>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 ( <b>che.infra.openshift.project</b> 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スのスサービスアカウントをここに準備します。</p>
<b>CHE_INFRA_KUBERNETES_WORKSPACE__SA__CLUSTER__ROLES</b>	<b>NULL</b>	<p>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。このプロパティは 'che.infra.kubernetes.cluster_role_name' を非推奨にします。</p>
<b>CHE_INFRA_KUBERNETES_WORKSPACE__START__TIMEOUT__MIN</b>	<b>8</b>	Kubernetes 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。
<b>CHE_INFRA_KUBERNETES_INGRESS__START__TIMEOUT__MIN</b>	<b>5</b>	OpenShift Route 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。

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_WORKSPACE__UNRECOVERABLE__EVENTS</b>	<b>FailedMount,FailedScheduling,MountVolume.SetUpfailed,Failed to pull image,FailedCreate,ReplicaSetCreateError</b>	ワークスペースの起動中に、プロパティ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' の理由が含まれ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。失敗したコンテナの起動は、CodeReady Workspaces サーバーで明示的に処理されます。
<b>CHE_INFRA_KUBERNETES_PVC_ENABLED</b>	<b>true</b>	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
<b>CHE_INFRA_KUBERNETES_PVC_STRATEGY</b>	<b>common</b>	<p>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 Kubernetes 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</p> <p>'{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</p> <p>'{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。</p>

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_PVC_PRECREATE__SUBPATHS</b>	<b>true</b>	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。ワークスペースのサブパスの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 Workspace 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が表示される)、一部の OpenShift のバージョンで必要です。デフォルトは「true」ですが、Openshift/Kubernetes 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: <a href="https://github.com/kubernetes/kubernetes/issues/41638">https://github.com/kubernetes/kubernetes/issues/41638</a> 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。
<b>CHE_INFRA_KUBERNETES_PVC_NAME</b>	<b>claim-che-workspace</b>	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。
<b>CHE_INFRA_KUBERNETES_PVC_STORAGE__CLASS__NAME</b>		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。
<b>CHE_INFRA_KUBERNETES_PVC_QUANTITY</b>	<b>10Gi</b>	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: <a href="https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html">https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html</a>

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE</b>	<b>centos:centos7</b>	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
<b>CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE_PULL_POLICY</b>	<b>IfNotPresent</b>	Kubernetes/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のイメージプルポリシー
<b>CHE_INFRA_KUBERNETES_PVC_JOBS_MEMORYLIMIT</b>	<b>250Mi</b>	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。
<b>CHE_INFRA_KUBERNETES_PVC_ACCESS_MODE</b>	<b>ReadWriteOnce</b>	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、 <a href="https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html">https://docs.openshift.com/container-platform/4.4/storage/understanding-persistent-storage.html</a> を参照してください。
<b>CHE_INFRA_KUBERNETES_PVC_WAIT_BOUND</b>	<b>true</b>	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。 <b>volumeBindingMode</b> が <b>WaitForFirstConsumer</b> に設定されている場合は、 <b>false</b> 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があります) です。

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_INSTALLER__SERVER__MIN__PORT</b>	<b>10000</b>	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。
<b>CHE_INFRA_KUBERNETES_INSTALLER__SERVER__MAX__PORT</b>	<b>20000</b>	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
<b>CHE_INFRA_KUBERNETES_INGRESS_ANNOTATIONS_JSON</b>	<b>NULL</b>	<p>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)。</p> <p>che.infra.kubernetes.ingress.path.rewrite_transform プロパティは、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)。たとえば、nginx ingress コントローラー 0.22.0 以降の場合、以下の値が推奨されます:</p> <pre>{'ingress.kubernetes.io/rewrite-target': '/\$1','ingress.kubernetes.io/ssl-redirect': 'false',\ 'ingress.kubernetes.io/proxy-connect-timeout': '3600','ingress.kubernetes.io/proxy-read-timeout': '3600'}</pre> <p>che.infra.kubernetes.ingress.path.rewrite_transform は 0.22.0 よりも前の '%s(*)' nginx ingress コントローラーに設定する必要があり、rewrite-target は単に '/' に設定し、path transform は '%s' に設定する必要があります (che.infra.kubernetes.ingress.path.rewrite_transform プロパティ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</p>

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_INGRESS_PATH_TRANSFORM</b>	<b>NULL</b>	この説明は、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。 サーバーを公開する 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 のアンノテーションとパスを指定する際にこれら2つのプロパティの相互作用を確認するには、che.infra.kubernetes.ingress.annotations_json プロパティ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。
<b>CHE_INFRA_KUBERNETES_INGRESS_LABELS</b>	<b>NULL</b>	明確化できるように、CodeReady Workspaces サーバーによって作成されるすべての Ingress に追加する追加のラベル。
<b>CHE_INFRA_KUBERNETES_POD_SECURITY_CONTEXT_RUN_AS_USER</b>	<b>NULL</b>	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。
<b>CHE_INFRA_KUBERNETES_POD_SECURITY_CONTEXT_FS_GROUP</b>	<b>NULL</b>	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。



環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_POD_TERMINATION__GRACE_PERIOD__SEC</b>	0	Kubernetes / OpenShift インフラストラクチャーによって作成される Pod の強制終了の猶予期間を定義します。Kubernetes / OpenShift ワークスペースの Pod の強制終了の猶予期間はデフォルトで '0' に設定されます。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: <b>terminationGracePeriodSeconds</b> が Kubernetes / 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。
<b>CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC_REQUESTS_MAX</b>	1000	<b>KubernetesClient</b> 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。
<b>CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC_REQUESTS_MAX_PER_HOST</b>	1000	<b>KubernetesClient</b> 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。
<b>CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION_POOL_MAX_IDLE</b>	5	Kubernetes 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

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION_POOL_KEEP_ALIVE_MIN</b>	<b>5</b>	Kubernetes 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)
<b>CHE_INFRA_KUBERNETES_TLS_ENABLED</b>	<b>false</b>	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。
<b>CHE_INFRA_KUBERNETES_TLS_SECRET</b>		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
<b>CHE_INFRA_KUBERNETES_TLS_KEY</b>	<b>NULL</b>	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は OpenShift インフラストラクチャーで無視されます。
<b>CHE_INFRA_KUBERNETES_TLS_CERT</b>	<b>NULL</b>	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は OpenShift インフラストラクチャーで無視されます。

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_RUNTIMES__CONSISTENCY__CHECK__PERIOD__MIN</b>	<b>-1</b>	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 <b>-1</b> である必要があります。ここで、 <b>-1</b> 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 は操作がユーザーによって呼び出しされない場合に Kubernetes API と対話できなくなる CodeReady Workspaces サーバーの設定があることが予想されるためです。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が CodeReady Workspaces Server がある同じ namespace で作成される。- クラスター管理のサービスアカウントトークンが CodeReady Workspaces Server Pod にマウントされる。これは、以下の設定では機能しません。- CodeReady Workspaces Server が OAuth プロバイダーのトークンを使用して Kubernetes API と通信する。
<b>CHE_INFRA_KUBERNETES_TRUSTED__CA_SRC__CONFIGMAP</b>	<b>NULL</b>	すべてのユーザーのワークスペースに伝播される追加の CA TLS 証明書を含む、CodeReady Workspaces サーバー namespace の設定マップの名前。プロパティを OpenShift 4 インフラストラクチャーに設定し、che.infra.openshift.trusted_ca.dest_configmap_labels に config.openshift.io/inject-trusted-cabundle=true ラベルが含まれる場合、クラスター CA バンドルも伝播されます。
<b>CHE_INFRA_KUBERNETES_TRUSTED__CA_DEST__CONFIGMAP</b>	<b>ca-certs</b>	

環境変数名	デフォルト値	説明
<b>CHE_INFRA_KUBERNETES_TRUSTED__CA_MOUNT__PATH</b>	<b>/public-certs</b>	CA バンドルがマウントされるワークスペースコンテナでパスを設定します。 che.infra.kubernetes.trusted_ca.default_configmap で指定される設定マップの内容がマウントされます。
<b>CHE_INFRA_KUBERNETES_TRUSTED__CA_DEST_CONFIGMAP__LABELS</b>		ユーザーワークスペースの CA 証明書の設定マップに追加するラベルのコンマ区切りの一覧。 che.infra.kubernetes.trusted_ca.default_configmap プロパティを参照してください。

#### 4.1.2.5. OpenShift インフラパラメーター

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
<b>CHE_INFRA_OPENSHIFT_PROJECT</b>		非推奨: このプロパティ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。

環境変数名	デフォルト値	説明
<b>CHE_INFRA_OPENSHIFT_TRUSTED__CA_DEST_CONFIGMAP__LABELS</b>	<b>config.openshift.io/inject-trusted-cabundle=true</b>	ユーザーワークスペースの CA 証明書の設定マップに追加するラベルのコンマ区切りの一覧。 che.infra.kubernetes.trusted_ca.dest_configmap プロパティを参照してください。このデフォルト値は、OpenShift 4 でのクラスター CA バンドルの自動挿入に使用されます。
<b>CHE_INFRA_OPENSHIFT_ROUTE_LABELS</b>	<b>NULL</b>	明確化できるように、CodeReady Workspaces サーバーによって作成されるすべての Route に追加する追加のラベル。

#### 4.1.2.6. 実験的なプロパティ

表4.6 実験的なプロパティ

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_PLUGIN__BROKER_METADATA_IMAGE</b>	<b>quay.io/eclipse/che-plugin-metadata-broker:v3.4.0</b>	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。これらのイメージはデフォルトで CodeReady Workspaces Operator によって上書きされることに注意してください。ここでイメージを変更しても、CodeReady Workspaces が Operator でインストールされている場合は影響がありません。
<b>CHE_WORKSPACE_PLUGIN__BROKER_ARTIFACTS_IMAGE</b>	<b>quay.io/eclipse/che-plugin-artifacts-broker:v3.4.0</b>	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。これらのイメージはデフォルトで CodeReady Workspaces Operator によって上書きされることに注意してください。ここでイメージを変更しても、CodeReady Workspaces が Operator でインストールされている場合は影響がありません。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_PLUGIN__BROKER_DEFAULT_MERGE_PLUGINS</b>	<b>false</b>	プラグインをワークスペースにプロビジョニングする際にプラグインブローカーのデフォルト動作を設定します。true に設定すると、プラグインブローカーは可能な場合にプラグインのマージを試行します（つまり、それらは同じサイドカーイメージで実行され、設定が競合することはありません）。この値は、devfile が指定していない場合に、'mergePlugins' 属性を使用して使用されるデフォルト設定です。
<b>CHE_WORKSPACE_PLUGIN__BROKER_PULL_POLICY</b>	<b>Always</b>	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
<b>CHE_WORKSPACE_PLUGIN__BROKER_WAIT_TIMEOUT_MIN</b>	<b>3</b>	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。
<b>CHE_WORKSPACE_PLUGIN__REGISTRY_URL</b>	<b>https://che-plugin-registry.prod-preview.openshift.io/v3</b>	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。例: 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。
<b>CHE_WORKSPACE_PLUGIN__REGISTRY_INTERNAL_URL</b>	<b>NULL</b>	ワークスペースツールプラグインレジストリーの 'internal' エンドポイントです。有効な HTTP URL でなければなりません。例: http://devfile-registry.che.svc.cluster.local:8080 CodeReady Workspaces プラグインが不要な場合、値 'NULL' を使用する必要があります。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_DEVFILE__REGISTRY__URL</b>	<b>https://che-devfile-registry.prod-preview.openshift.io/</b>	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。例: http://che-devfile-registry-eclipse-che.192.168.65.2.nip.io CodeReady Workspaces プラグインが不要な場合、値 'NULL' を使用する必要があります。
<b>CHE_WORKSPACE_DEVFILE__REGISTRY__INTERNAL__URL</b>	<b>NULL</b>	devfile レジストリー 'internal' エンドポイント。有効な HTTP URL でなければなりません。例: http://plugin-registry.che.svc.cluster.local:8080 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。
<b>CHE_WORKSPACE_STORAGE_AVAILABLE__TYPES</b>	<b>persistent,ephemeral,async</b>	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。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。その他の場合は、一覧から 'async' を削除します。

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_STORAGE_PREFERRED__TYPE</b>	<b>persistent</b>	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。「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。
<b>CHE_SERVER_SECURE_EXPOSER</b>	<b>jwtproxy</b>	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。
<b>CHE_SERVER_SECURE_EXPOSER_JWTPROXY_TOKEN_ISSUER</b>	<b>wsmaster</b>	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。
<b>CHE_SERVER_SECURE_EXPOSER_JWTPROXY_TOKEN_TTL</b>	<b>8800h</b>	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。
<b>CHE_SERVER_SECURE_EXPOSER_JWTPROXY_AUTH_LOADER_PATH</b>	<b>/_app/loader.html</b>	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。
<b>CHE_SERVER_SECURE_EXPOSER_JWTPROXY_IMAGE</b>	<b>quay.io/eclipse/che-jwtproxy:0.10.0</b>	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。
<b>CHE_SERVER_SECURE_EXPOSER_JWTPROXY_MEMORY_REQUEST</b>	<b>15mb</b>	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。
<b>CHE_SERVER_SECURE_EXPOSER_JWTPROXY_MEMORY_LIMIT</b>	<b>128mb</b>	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。



環境変数名	デフォルト値	説明
CHE_SERVER_SECURE_EX POSER_JWTPROXY_CPU_ REQUEST	0.03	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_EX POSER_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。

#### 4.1.2.7. 主な「/websocket」エンドポイントの設定

表4.7 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_PRO CESSOR__MAX__POOL__SI ZE	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_PRO CESSOR__CORE__POOL__S IZE	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_PRO CESSOR__QUEUE__CAPACI TY	100000	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

#### 4.1.2.8. CORS 設定

表4.8 CORS 設定

環境変数名	デフォルト値	説明
CHE_CORS_ALLOWED__OR IGINS	*	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
<b>CHE_CORS_ALLOW_CREDENTIALS</b>	<b>false</b>	'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。

#### 4.1.2.9. Factory のデフォルト

表4.9 Factory のデフォルト

環境変数名	デフォルト値	説明
<b>CHE_FACTORY_DEFAULT_EDITOR</b>	<b>eclipse/che-theia/latest</b>	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 リポジトリから作成される Factory 用に作成されるエディターおよびプラグイン。複数のプラグインは、以下のようにコンマで区切る必要があります。例: pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion
<b>CHE_FACTORY_DEFAULT_PLUGINS</b>	<b>eclipse/che-machine-exec-plugin/latest</b>	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 リポジトリから作成される Factory 用に作成されるエディターおよびプラグイン。複数のプラグインは、以下のようにコンマで区切る必要があります。例: pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion
<b>CHE_FACTORY_DEFAULT_DEVFILE__FILENAMES</b>	<b>devfile.yaml,.devfile.yaml</b>	リポジトリベースの 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。

#### 4.1.2.10. devfile のデフォルト

表4.10 devfile のデフォルト

環境変数名	デフォルト値	説明
<b>CHE_WORKSPACE_DEVFILE_DEFAULT_EDITOR</b>	<b>eclipse/che-theia/latest</b>	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、 <b>editorPublisher/editorName/editorVersion</b> 値になります。 <b>NULL</b> 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。
<b>CHE_WORKSPACE_DEVFILE_DEFAULT_EDITOR_PLUGINS</b>	<b>eclipse/che-machine-exec-plugin/latest</b>	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です。形式は、コマンドで区切られた <b>pluginPublisher/pluginName/pluginVersion</b> 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。
<b>CHE_WORKSPACE_PROVISION_SECRET_LABELS</b>	<b>app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspace-secret</b>	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。
<b>CHE_WORKSPACE_DEVFILE_ASYNC_STORAGE_PLUGIN</b>	<b>eclipse/che-async-pv-plugin/latest</b>	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
<b>CHE_INFRA_KUBERNETES_ASYNC_STORAGE_IMAGE</b>	<b>quay.io/eclipse/che-workspace-data-sync-storage:0.0.1</b>	CodeReady Workspaces 非同期ストレージの Docker イメージ
<b>CHE_WORKSPACE_POD_NODE_SELECTOR</b>	<b>NULL</b>	オプションでワークスペース Pod のノードセ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)。
<b>CHE_WORKSPACE_POD_TOLERATIONS_JSON</b>	<b>NULL</b>	オプションでワークスペース Pod の容認を設定します。形式は、テナントの容認の JSON 配列を表す文字列か、または <b>NULL</b> の場合はこれを無効にします。配列に含まれるオブジェクトは、この <a href="#">仕様</a> に準拠する必要があります。例: <pre>[{'effect':'NoExecute','key':'aNodeTaint','operator':'Equal','value':'aValue'}]</pre>
<b>CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN_TIMEOUT_MIN</b>	<b>120</b>	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。
<b>CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN_CHECK_PERIOD_MIN</b>	<b>30</b>	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)。
<b>CHE_INTEGRATION_BITBUCKET_SERVER_ENDPOINTS</b>	<b>NULL#</b>	Factory の統合に使用される Bitbucket エンドポイント。bitbucket サーバー URL のコンマ区切りの一覧、または統合が予想されない場合は NULL。

#### 4.1.2.11. Che システム

表4.11 Che システム

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
<b>CHE_SYSTEM_SUPER_PRIVILEGED_MODE</b>	<b>false</b>	System Super Privileged Mode (システムのスーパー特権モード)。getKey、getByNameSpace、stopWorkspaces、および getResources の manageSystem パーMISSIONの追加パーMISSION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MISSION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。
<b>CHE_SYSTEM_ADMIN_NAME</b>	<b>admin</b>	'che.admin.name' ユーザーのシステムパーMISSION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。

#### 4.1.2.12. Workspace の制限

表4.12 Workspace の制限

環境変数名	デフォルト値	説明
<b>CHE_LIMITS_WORKSPACE_ENV_RAM</b>	<b>16gb</b>	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。

環境変数名	デフォルト値	説明
<b>CHE_LIMITS_WORKSPACE_IDLE_TIMEOUT</b>	<b>1800000</b>	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。
<b>CHE_LIMITS_WORKSPACE_RUN_TIMEOUT</b>	<b>0</b>	システムが一時的に停止するまでの、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。

#### 4.1.2.13. ユーザーワークスペースの制限

表4.13 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
<b>CHE_LIMITS_USER_WORKSPACES_RAM</b>	<b>-1</b>	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。
<b>CHE_LIMITS_USER_WORKSPACES_COUNT</b>	<b>-1</b>	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、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。

環境変数名	デフォルト値	説明
<b>CHE_LIMITS_USER_WORKSPACES_RUN_COUNT</b>	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、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。

#### 4.1.2.14. 組織ワークスペースの制限

表4.14 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
<b>CHE_LIMITS_ORGANIZATION_WORKSPACES_RAM</b>	-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。
<b>CHE_LIMITS_ORGANIZATION_WORKSPACES_COUNT</b>	-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、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。
<b>CHE_LIMITS_ORGANIZATION_WORKSPACES_RUN_COUNT</b>	-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、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。
<b>CHE_MAIL_FROM_EMAIL_ADDRESS</b>	<b>che@noreply.com</b>	メール通知の送信元のメールに使用されるアドレス

#### 4.1.2.15. 組織通知の設定

表4.15 組織通知の設定

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

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

表4.16 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
-------	--------	----



環境変数名	デフォルト値	説明
<b>CHE_INFRA_OPENSHIFT_OAUTH__IDENTITY__PROVIDER</b>	<b>NULL</b>	Keycloak に登録されている OpenShift アイデンティティ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。 <b>che.infra.openshift.project</b> 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、 <a href="https://www.keycloak.org/docs/latest/server_admin/index.html#openshift-4">https://www.keycloak.org/docs/latest/server_admin/index.html#openshift-4</a> のドキュメントを参照してください。

#### 4.1.2.17. Keycloak の設定

表4.17 Keycloak の設定

環境変数名	デフォルト値	説明
<b>CHE_KEYCLOAK_AUTH_SERVER_URL</b>	<b>http://\${CHE_HOST}:5050/auth</b>	<b>che.keycloak.oidcProvider</b> を使用している場合のみ、keycloak アイデンティティプロバイダーサーバーの URL を NULL に設定できます。
<b>CHE_KEYCLOAK_AUTH_INTERNAL_SERVER_URL</b>	<b>NULL</b>	keycloak アイデンティティプロバイダーサーバーへの内部ネットワークサービス URL
<b>CHE_KEYCLOAK_REALM</b>	<b>che</b>	Keycloak レalmはユーザーを認証するために使用されます。 <b>che.keycloak.oidcProvider</b> を使用している場合のみ NULL に設定できます。
<b>CHE_KEYCLOAK_CLIENT_ID</b>	<b>che-public</b>	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
<b>CHE_KEYCLOAK_OSO_ENDPOINT</b>	<b>NULL</b>	OSO oauth トークンにアクセスするための URL

環境変数名	デフォルト値	説明
<b>CHE_KEYCLOAK_GITHUB_ENDPOINT</b>	<b>NULL</b>	Github oauth トークンにアクセスするための URL
<b>CHE_KEYCLOAK_ALLOWED_CLOCK_SKEW_SEC</b>	<b>3</b>	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。
<b>CHE_KEYCLOAK_USE_NONCE</b>	<b>true</b>	OIDC オプションの <b>nonce</b> 機能を使用して、セキュリティを強化します。
<b>CHE_KEYCLOAK_JS_ADAPTER_URL</b>	<b>NULL</b>	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 <b><code>\${che.keycloak.auth_server_url}/js/keycloak.js</code></b> になり、別の <b>oidc_provider</b> を使用する場合は <b><code>&lt;che-server&gt;/api/keycloak/OIDCKeycloak.js</code></b> になります。
<b>CHE_KEYCLOAK_OIDC_PROVIDER</b>	<b>NULL</b>	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。 <a href="https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig">https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig</a>
<b>CHE_KEYCLOAK_USE_FIXED_REDIRECT_URLS</b>	<b>false</b>	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。このプロパティは、 <b>che.keycloak.oidc_provider</b> が NULL の場合は無視されます。
<b>CHE_KEYCLOAK_USERNAME_CLAIM</b>	<b>NULL</b>	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
<b>CHE_OAUTH_SERVICE__MODE</b>	<b>delegated</b>	<p>「embedded」モードまたは「delegated」モードで使用する OAuth 認証サービスの設定。</p> <p>「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。</p> <p>「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。</p>
<b>CHE_KEYCLOAK_CASCADE__USER_REMOVAL_ENABLED</b>	<b>false</b>	<p>CodeReady Workspaces データベースからユーザーを削除する際の Keycloak サーバーからのユーザーの削除を有効にするための設定。デフォルトで、これは無効にされます。CodeReady Workspaces データベースでユーザーを削除する際に Keycloak からの関連ユーザーの削除が実行される特別なケースでは有効にされることがあります。適切に機能するには、管理ユーザー名 <code>\${che.keycloak.admin_username}</code> とパスワード <code>\${che.keycloak.admin_password}</code> を設定する必要があります。</p>
<b>CHE_KEYCLOAK_ADMIN__USERNAME</b>	<b>NULL</b>	<p>Keycloak 管理ユーザー名。CodeReady Workspaces データベースからユーザーを削除する際に Keycloak からユーザーを削除するために使用します。</p> <p><code>\${che.keycloak.cascade_user_removal_enabled}</code> が 'true' に設定されている場合にのみ機能します。</p>
<b>CHE_KEYCLOAK_ADMIN__PASSWORD</b>	<b>NULL</b>	<p>Keycloak 管理者パスワード。CodeReady Workspaces データベースからユーザーを削除する際に Keycloak からユーザーを削除するために使用します。</p> <p><code>\${che.keycloak.cascade_user_removal_enabled}</code> が 'true' に設定されている場合にのみ機能します。</p>

環境変数名	デフォルト値	説明
<b>CHE_KEYCLOAK_USERNAME_REPLACEMENT_PATTERN</b>	<b>NULL</b>	<p>ユーザー名の調整の設定。</p> <p>CodeReady Workspaces は、ユーザー名を K8s オブジェクト名とラベルの一部として使用する必要があるため、アイデンティティプロバイダーが通常許可する場合よりもフォーマットの要件が厳しくなります (DNS に準拠する必要があります)。この調整は、コンマ区切りのキー/値のペアで表されます。これらは元のユーザー名の String.replaceAll 関数への引数として順次使用されます。キーは正規表現で、値は正規表現に一致するユーザー名の文字を置き換える置換文字列です。変更したユーザー名は CodeReady Workspaces データベースのみに保存され、アイデンティティプロバイダーには再び公開されません。DNS に準拠する文字を代替文字列として使用することが推奨されます (キー/値のペアの値)。例: <code>\=-, @=-at-</code> は <code>\</code> を <code>-</code> に、<code>@</code> を <code>-at-</code> に変更して、ユーザー名 <b>org\user@com</b> が <b>org-user-at-com</b> になるようにします。</p>

## 関連情報

- [外部の Keycloak インストールを使用するように Che を設定する](#)

## 4.2. プロジェクトストラテジーの設定

新規ワークスペース Pod がデプロイされる OpenShift プロジェクトは、CodeReady Workspaces サーバー設定によって異なります。デフォルトで、すべてのワークスペースは個別の OpenShift プロジェクトにデプロイされますが、ユーザーは CodeReady Workspaces サーバーを 1 つの特定の OpenShift プロジェクトにすべてのワークスペースをデプロイするように設定できます。OpenShift プロジェクトの名前は CodeReady Workspaces サーバー設定プロパティーとして指定される必要があり、実行時に変更することはできません。

Operator インストーラーでは、OpenShift プロジェクトストラテジーは **server.workspaceNamespaceDefault** プロパティを使用して設定されます。

### Operator CheCluster CR パッチ

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

- 1 - CodeReady Workspaces ワークスペース namespace の設定



#### 注記

CodeReady Workspaces サーバーが使用する基礎となる環境変数は **CHE\_INFRA\_KUBERNETES\_NAMESPACE\_DEFAULT** です。



#### 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。

**警告**

Kubernetes は namespace 名の長さを 63 文字に制限します (これには評価されるプレースホルダーが含まれます)。さらに、名前 (プレースホルダーの評価後) は有効な DNS 名である必要があります。

マルチホストサーバーの脆弱性ストラテジーのある OpenShift では、長さはさらに 49 文字に制限されます。

<userid> プレースホルダーは 36 文字の長さの UUID 文字列として評価されることに注意してください。

**警告**

新規プロジェクトの作成が必要なストラテジーの場合、**che** ServiceAccount にこれを実行するのに十分なパーミッションがあることを確認します。OpenShift OAuth では、認証されたユーザーには、新規プロジェクトを作成するための権限が必要です。

#### 4.2.1. 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、**CodeReady Workspaces workspace namespace** 設定の値を1つ以上のユーザー ID が含まれるように設定します。現在サポートされている ID は <username> および <userid> です。

##### 例4.2 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。

Operator インストーラー (CheCluster CustomResource)

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

#### 4.2.2. 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、CodeReady Workspaces workspace namespace configuration値を **<workspaceID>** ID が含まれるように設定します。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4.3 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。

Operator インストーラー (CheCluster CustomResource)

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

#### 4.2.3. 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、CodeReady Workspaces workspace namespace configuration値を、使用する必要のあるプロジェクトの名前に設定します。

#### 例4.4 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。

Operator インストーラー (CheCluster CustomResource)

```
...
spec:
  server:
    workspaceNamespaceDefault: codeready-ws
...
```

#### 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。

```
...
server:
  allowUserDefinedWorkspaceNamespaces: true
...
```

#### 4.2.5. 互換性のないユーザー名またはユーザー ID の処理

CodeReady Workspaces サーバーは、テンプレートからプロジェクトを作成する前に、Kubernetes オブジェクトの命名規則との互換性についてユーザー名と ID を自動的にチェックします。互換性のないユーザー名または ID は、適切ではないシンボルのグループを - に置き換えることにより減少し、ほぼ有効な名前だけに絞られます。競合を回避するために、ランダムな 6 記号のサフィックスが追加され、結果は再利用できるように設定に保存されます。

#### 4.2.6. ユーザーのプロジェクトの事前作成

ユーザーのプロジェクトを事前に作成するには、プロジェクトのラベルおよびアノテーションを使用します。この namespace は、**CHE\_INFRA\_KUBERNETES\_NAMESPACE\_DEFAULT** 変数よりも優先して使用されます。

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

① ターゲットユーザーのユーザー名

ラベルを設定するには、**CHE\_INFRA\_KUBERNETES\_NAMESPACE\_LABELS** を必要なラベルに設定します。アノテーションを設定するに

は、**CHE\_INFRA\_KUBERNETES\_NAMESPACE\_ANNOTATIONS** を必要なアノテーションに設定します。詳細は、[CodeReady Workspaces サーバーコンポーネントのシステムプロパティのリファレンス](#)を参照してください。



#### 警告

単一ユーザーに複数の namespace を作成することは推奨されません。これにより、定義されていない動作が生じる可能性があります。



## 重要

OAuth を使用する OpenShift では、ターゲットユーザーにはターゲット namespace で **admin** ロール権限が必要です。

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

- ❶ 事前に作成される namespace
- ❷ ターゲットユーザー

Kubernetes では、**che** ServiceAccount には、クラスター全体の **list** および **get namespaces** パーミッションと、ターゲット namespace の **admin** ロールが必要です。

### 4.2.7. namespace のラベル付け

CodeReady Workspaces は、**CHE\_INFRA\_KUBERNETES\_NAMESPACE\_LABELS** で定義されるラベルを追加して、ワークスペースの起動時にワークスペースの namespace を更新します。これを実行するには、**che** ServiceAccount に **update** および **get namespaces** に対する以下のようなクラスター全体のパーミッションが必要になります。

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

- ❶ クラスターロールの名前

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

```

name: <service-account-name> ❷
namespace: <service-account-namespace> ❸
roleRef:
  kind: ClusterRole
  name: <cluster-role-name> ❹
  apiGroup: rbac.authorization.k8s.io

```

- ❶ クラスターロールバインディングの名前
- ❷ che サービスアカウントの名前
- ❸ CodeReady Workspaces インストール namespace
- ❹ 直前の手順で作成されたクラスターロールの名前



### 注記

CodeReady Workspaces は、パーミッションがないことでワークスペースの起動に失敗することはなく、警告のみがログに記録されます。CodeReady Workspaces ログに警告が表示される場合は、**CHE\_INFRA\_KUBERNETES\_NAMESPACE\_LABEL=false** を設定して機能を無効にすることを検討してください。

## 4.3. ストレージストラテジーの設定

本セクションでは、CodeReady Workspaces ワークスペースのストレージストラテジーを設定する方法を説明します。

### 4.3.1. codeready-workspaces ワークスペースのストレージストラテジー

ワークスペース Pod は、[ReadWriteOnce アクセスモード](#)で物理的な永続ボリューム (PV) にバインドされる Persistent Volume Claim (永続ボリューム要求、PVC) を使用します。CodeReady Workspaces サーバーがワークスペースに PVC を使用する方法を設定できます。この設定の個別の方法は PVC ストラテジーと呼ばれます。

ストラテジー	詳細	利点	不利な点
unique	ワークスペースボリュームまたはユーザー定義 PVC ごとに1つの PVC	ストレージの分離	定義されない数の PV が必要です
per-workspace (default)	1つのワークスペースに1つの PVC	一意のストラテジーと比較すると、ストレージの管理と制御が容易になります。	PV 数は不明で、ワークスペース数により異なります。

ストラテジー	詳細	利点	不利な点
common	1つの OpenShift namespace のすべてのワークスペースに1つの PVC	ストレージの管理と制御が容易になります。	<p>PV が ReadWriteMany (RWX) アクセスモードをサポートしない場合、ワークスペースは別の OpenShift namespace になければなりません。</p> <p>または、1つの namespace で 2 つ以上のワークスペースを同時に実行することはできません。</p>

Red Hat CodeReady Workspaces は、すべての CodeReady Workspaces ワークスペースがユーザーのプロジェクトで動作し、1つの PVC を共有する場合に、**common** PVC ストラテジーを「ユーザーごとに1プロジェクト」のプロジェクトストラテジーと共に使用します。

#### 4.3.1.1. common PVC ストラテジー

OpenShift プロジェクト内のすべてのワークスペースは、宣言されたボリュームに以下のようなデータを保存する際に、デフォルトのデータストレージと同じ Persistent Volume Claim（永続ボリューム要求、PVC）を使用します。

- プロジェクト
- ワークスペースログ
- 使用に基づいて定義される追加のボリューム

**common** PVC ストラテジーが使用されている場合、ユーザー定義 PVC は無視され、これらのユーザー定義 PVC を参照するボリュームは共通 PVC を参照するボリュームに置き換えられます。このストラテジーでは、すべての CodeReady Workspaces ワークスペースが同じ PVC を使用します。ユーザーが1つのワークスペースを実行すると、一度にクラスター内の1つのノードにのみバインドします。

対応するコンテナボリュームのマウントは共通ボリュームにリンクされ、サブパスには **<workspace-ID>** または **<original-PVC-name>** のプレフィックスが付けられます。詳細は、[「サブパスが PVC で使用される方法」](#) を参照してください。

CodeReady Workspaces ボリューム名は、ユーザー定義 PVC の名前と同じです。つまり、マシンがユーザー定義の PVC と同じ名前を持つ CodeReady Workspaces ボリュームを使用するように設定されている場合、それらは共通 PVC で同じ共有フォルダーを使用します。

ワークスペースが削除されると、対応するサブディレクトリー (**\${ws-id}**) が PV ディレクトリーで削除されます。

#### common PVC ストラテジーの使用に関する制限

**common** ストラテジーが使用され、ワークスペース PVC アクセスモードが ReadWriteOnce (RWO) の場合、1つのノードのみが PVC を同時に使用できます。

ノードが複数ある場合には、**common** ストラテジーを使用できますが、以下の点に注意してください。

- ワークスペース PVC アクセスモードを **ReadWriteMany** (RWM) に再設定し、複数のノードがこの PVC を同時に使用できるようにする必要があります。
- 同じプロジェクトの1つのワークスペースのみを実行できます。[「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。

**common** PVC ストラテジーは、大規模なマルチノードクラスターには適していません。そのため、単一ノードクラスターで使うことが最も適しています。ただし、**per-workspace** プロジェクトストラテジーと組み合わせにより、**common** PVC ストラテジーは 75 ノードを超えるクラスターで使用できます。このストラテジーで使用する PVC は、1つのプロジェクトが他のプロジェクトのリソースを使い切る状況を防ぐために、すべてのプロジェクトに対応するのに十分な大きさである必要があります。

#### 4.3.1.2. per-workspace PVC ストラテジー

**per-workspace** ストラテジーは **common** PVC ストラテジーに似ています。すべてのワークスペースではなく、すべてのワークスペースのボリュームが以下についてデフォルトのデータストレージと同じ PVC を使用する点が唯一の違いになります。

- プロジェクト
- ワークスペースログ
- ユーザーが定義する追加のボリューム

このストラテジーでは、CodeReady Workspaces は単一の PVC によって割り当てられる割り当てられた PV にワークスペースのデータを保持します。

**per-workspace** PVC ストラテジーは、利用可能な PVC ストラテジーの中でも最も汎用的なストラテジーであり、ユーザーの量が多い大規模なマルチノードクラスターの適切なオプションとして機能します。**per-workspace** PVC ストラテジーを使用すると、ユーザーは複数のワークスペースを同時に実行できます。これにより、PVC がさらに作成されます。

#### 4.3.1.3. unique PVC ストラテジー

'unique' PVC ストラテジーを使用する場合、ワークスペースのすべての CodeReady Workspaces ボリュームには独自の PVC があります。つまり、ワークスペース PVC は以下のようになります。

ワークスペースの初回起動時に作成されます。対応するワークスペースが削除されると削除されます。

ユーザー定義の PVC は以下の詳細で作成されます。

- これらは、プロジェクトの他の PVC との名前の競合を防ぐために、生成される名前でプロビジョニングされます。
- ユーザー定義の PVC を参照するマウントされた物理的な永続ボリュームのサブパスには、**<workspace-ID>** または **<PVC-name>** のプレフィックスが付けられます。これにより、同じ PV データ構造が異なる PVC ストラテジーで設定されます。詳細は、[「サブパスが PVC で使用される方法」](#)を参照してください。

**unique** PVC ストラテジーは、ユーザーの量が少ない大規模なマルチノードクラスターに適しています。このストラテジーはワークスペースの各ボリュームについて別個の PVC で動作するため、さらに多くの PVC が作成されます。

#### 4.3.1.4. サブパスが PVC で使用される方法

サブパスは、永続ボリューム (PV) のフォルダー階層を示しています。

```

/pv0001
/workspaceID1
/workspaceID2
/workspaceIDn
/che-logs
/projects
/<volume1>
/<volume2>
/<User-defined PVC name 1 | volume 3>
...

```

ユーザーが devfile でコンポーネントのボリュームを定義すると、同じ名前のボリュームを定義するすべてのコンポーネントは、PV 内の **<PV-name> <workspace-ID>, or `<original-PVC-name>** と同じディレクトリーでサポートされます。各コンポーネントでは、コンテナ内の異なるパスにこの場所をマウントすることができます。

## 例

**common** PVC ストラテジーを使用すると、ユーザー定義の PVC は共通 PVC のサブパスに置き換えられます。ユーザーがボリュームを **my-volume** として参照すると、これは **/workspace-id/my-volume** サブパスで common-pvc にマウントされます。

### 4.3.2. 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

永続ボリューム (PV) は、ボリュームをクラスターに追加する仮想ストレージインスタンスとして機能します。

永続ボリューム要求 (PVC) は、以下の CodeReady Workspaces ストレージ設定ストラテジーで利用可能な特定のタイプおよび設定の永続ストレージのプロビジョニング要求です。

- Common
- Per-workspace
- Unique

マウントされた PVC はコンテナのファイルシステムのフォルダーとして表示されます。

#### 4.3.2.1. Operator を使用した PVC ストラテジーの設定

以下のセクションでは、Operator を使用して CodeReady Workspaces サーバーのワークスペースの永続ボリューム要求 (PVC) ストラテジーを設定する方法を説明します。



#### 警告

既存のワークスペースを使用して既存の CodeReady Workspaces クラスターに PVC ストラテジーを再設定することは推奨されません。これを実行すると、データが失われます。

**Operator** は、**カスタムリソース**を使用してアプリケーションとそのコンポーネントを管理する OpenShift に対するソフトウェアの拡張機能です。

Operator を使用して CodeReady Workspaces をデプロイする場合は、CheCluster カスタムリソースオブジェクトの YAML ファイルの **spec.storage.pvcStrategy** プロパティを変更して、目的のストラテジーを設定します。

#### 前提条件

- **oc** ツールが利用可能である。

#### 手順

以下の手順は、OpenShift コマンドラインツール「oc」で使用できます。

CheCluster YAML ファイルに変更を加えるには、以下のいずれかを選択します。

- **oc apply** コマンドを実行して、新規クラスターを作成します。以下に例を示します。

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

- **oc patch** コマンドを実行して、すでに実行中のクラスターの YAML ファイルプロパティを更新します。以下に例を示します。

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

使用されるストラテジーに応じて、上記の例の **per-workspace** オプションを **unique** または **common** に置き換えます。

## 4.4. 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)

### 4.4.1. 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'
```

#### 4.4.2. 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。

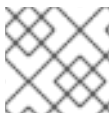
一時モードは、永続ストレージよりも高速な I/O を提供します。このストレージタイプを有効にするには、以下をワークスペース設定に追加します。

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

表4.18 AWS EBS での一時モード (emptyDir) と永続モードの I/O 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

#### 4.4.3. 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度に行うことが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'
```

### 4.4.4. 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

以下の2つの **che.properties** を使用して、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
```

ユーザーは、ワークスペースの作成時に CodeReady Workspaces ダッシュボードの **Create Custom Workspace** タブでストレージタイプを設定できます。既存のワークスペースのストレージタイプは、ワークスペースの詳細について **Overview** タブで設定できます。

### 4.4.5. 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。

## 4.5. 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_WORKSPACES_RUN_COUNT": "-1"} } } }
```

1. **per-workspace** または **unique** PVC ストラテジーを設定します。[「ストレージストラテジーの設定」](#)を参照してください。



#### 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。

## 4.6. 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。

### 4.6.1. Operator を使用したワークスペース公開ストラテジーの設定

Operator は、[カスタムリソース](#)を使用してアプリケーションとそのコンポーネントを管理する OpenShift に対するソフトウェアの拡張機能です。

## 前提条件

- **oc** ツールが利用可能である。

## 手順

Operator を使用して CodeReady Workspaces をデプロイする場合は、CheCluster カスタムリソースオブジェクトの YAML ファイルの **spec.server.serverExposureStrategy** プロパティを変更して、目的のストラテジーを設定します。

**spec.server.serverExposureStrategy** のサポートされる値は、以下のとおりです。

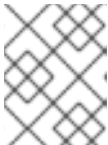
- **multi-host**
- **single-host**

個別のストラテジーの詳細は、「[ワークスペース公開ストラテジー](#)」を参照してください。

CheCluster YAML ファイルに加えた変更を有効にするには、以下のいずれかを実行します。

- パッチを適用して **crwctl** コマンドを実行して、新しいクラスターを作成します。以下に例を示します。

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



### 注記

利用可能な OpenShift デプロイメントプラットフォームの一覧については、**crwctl server:deploy --platform --help** を使用します。

- 以下の **patch.yaml** ファイルを使用します。

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

- ❶ - ワークスペース公開ストラテジーの使用

- **oc patch** コマンドを実行して、すでに実行中のクラスターの YAML ファイルプロパティを更新します。以下に例を示します。

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

## 1 - ワークスペース公開ストラテジーの使用

### 4.6.2. 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
- **single-host** ストラテジー
  - **gateway** サブタイプの使用

#### 4.6.2.1. Multi-host ストラテジー

Multi-host ストラテジーでは、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。これはデフォルトのストラテジーです。

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。

#### 4.6.2.2. 単一ホストストラテジー

single-host ストラテジーでは、すべてのワークスペースが主な CodeReady Workspaces サーバードメインのサブパスにデプロイされます。

これは、すべてのワークスペースコンポーネントのデプロイメントに対応する CodeReady Workspaces サーバーの単一の証明書のみが必要となるため、TLS で保護される CodeReady Workspaces サーバーの場合に便利です。

単一ホストストラテジーには、異なる実装方法が設定された 2 つのサブタイプがあります。最初のサブタイプの名前は **native** です。このストラテジーは Kubernetes でデフォルトで利用できますが、サーバーの公開に Ingress を使用するため、OpenShift では利用できません。**gateway** という名前の 2 つ目のサブタイプは OpenShift の両方で機能し、内部で実行されるリバースプロキシのある特別な Pod を使用して要求をルーティングします。

**警告**

**gateway** single-host ストラテジーでは、クラスターのネットワークポリシーを設定して、ワークスペースのサービスが (通常は CodeReady Workspaces プロジェクトの) リバースプロキシ Pod から到達できるように設定する必要があります。通常、これらは異なるプロジェクトに置かれます。

devfile に指定されたエンドポイントを公開する方法は 2 つあります。これらは、CodeReady Workspaces の

**CHE\_INFRA\_KUBERNETES\_SINGLEHOST\_WORKSPACE\_DEVFILE\_\_ENDPOINT\_\_EXPOSURE** 環境変数を使用して設定できます。この環境変数は、single-host サーバーストラテジーでのみ有効であり、すべてのユーザーのすべてのワークスペースに適用できます。

#### 4.6.2.2.1. devfile エンドポイント:single-host

**CHE\_INFRA\_KUBERNETES\_SINGLEHOST\_WORKSPACE\_DEVFILE\_\_ENDPOINT\_\_EXPOSURE:**  
**'single-host'**

この単一ホスト設定は、サブパスのエンドポイントを公開します (例: **https://<che-host>/serverihzmuqqc/go-cli-server-8080**)。これにより、公開されるコンポーネントおよびユーザーアプリケーションが制限されます。サーバーを参照するサーバー側で生成される絶対 URL は機能しません。これは、サーバーが、コンポーネントまたはユーザーアプリケーションから一意の URL パスのプレフィックスを非表示にするパスが書き換えられるリバースプロキシの背後にあるためです。

たとえば、ユーザーが仮説の [**https://codeready-<openshift\_deployment\_name>.<domain\_name>/component-prefix-djh3d/app/index.php**] URL にアクセスする場合、アプリケーションは要求が **https://internal-host/app/index.php** に送信されるのを認識します。アプリケーションが UI で生成する URL でホストを使用している場合、内部ホストが外部に表示されるホストとは異なるため、これは機能しません。ただし、アプリケーションが URL に絶対パスを使用している場合 (上記の場合は **/app/index.php**)、この URL は依然として機能しません。これは、外部ではこの URL はコンポーネント固有のプレフィックスがなく、アプリケーションを参照しないためです。

そのため、UI で相対 URL を使用するアプリケーションのみが、single-host ワークスペース公開ストラテジーで機能します。

#### 4.6.2.2.2. devfile エンドポイント:multi-host

**CHE\_INFRA\_KUBERNETES\_SINGLEHOST\_WORKSPACE\_DEVFILE\_\_ENDPOINT\_\_EXPOSURE:**  
**'multi-host'**

この単一ホスト設定は、サブドメインのエンドポイントを公開します (例: **http://serverihzmuqqc-go-cli-server-8080.<che-host>**)。これらのエンドポイントは、セキュアでない HTTP ポートで公開されます。**gateway** 単一ホストの設定でも、専用の Ingress または Route がこのエンドポイントに使用されます。

この設定により、CodeReady Workspaces が TLS で設定されている場合に、エディターページに直接表示されるプレビューの使用が制限されます。**https** ページはセキュリティが保護されたエンドポイントとの通信のみを許可するため、ユーザーは別のブラウザータブでアプリケーションのプレビューを開く必要があります。

### 4.6.3. 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、セキュリティ制限を課しません。

#### 4.6.3.1. JSON Web トークン (JWT) プロキシ

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

#### 4.6.3.2. セキュリティ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) のセキュリティを保護する必要はありません。これは、JWT プロキシ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。

#### 4.6.3.3. セキュリティ保護されたコンテナイメージコンポーネント

コンテナ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](#)」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](#)」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、セキュリティ上の影響が発生します。属性のデフォルト値は **false** です。

#### 4.6.3.4. 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 プロキシによってセキュリティ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なる場合があります。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。

#### 4.6.3.5. フィッシング攻撃

JWT プロキシ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。

### 4.7. 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\_STORAGE\_CLASS\_NAME** 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。

### 4.8. 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。

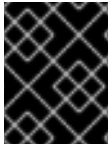
#### 前提条件

- **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 を参照するように要求します。

**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

1. 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。

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

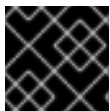
RH-SSO にログインするには、[RH-SSO へのログイン](#) 手順に従います。

## 4.9. OPENSIFT ROUTE のラベルの設定

この手順では、OpenShift Route のラベルを、オブジェクトを整理し、分類 (スコープおよび選択) するように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。
- OpenShift で実行される CodeReady Workspaces のインスタンス。



### 重要

ラベルを分離するにはコンマを使用します。 **key1=value1,key2=value2**

### 手順

1. OpenShift Route のラベルを設定するには、以下のコマンドを使用してカスタムリソースを更新します。

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

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

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

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

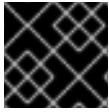
## 4.10. OPENSIFT ROUTE のラベルの設定

この手順では、OpenShift Route のラベルを、オブジェクトを整理し、分類 (スコープおよび選択) するように設定する方法を説明します。



## 別添木竹

- **oc** ツールが利用可能である。
- OpenShift で実行される CodeReady Workspaces のインスタンス。



## 重要

ラベルを分離するにはコンマを使用します。 **key1=value1,key2=value2**

## 手順

1. OpenShift Route のラベルを設定するには、以下のコマンドを使用してカスタムリソースを更新します。

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

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

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

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

## 4.11. 自己署名証明書を使用した GIT リポジトリをサポートする 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
```

## 4.12. 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:deploy** 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:deploy** コマンドを使用します。

```
$ crwctl server:deploy -m -p minikube -a operator --postgres-pvc-storage-class-name=postgres-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:deploy** コマンドを使用します。

```
$ crwctl server:deploy -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=postgres-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。

- 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。
- 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'
    # ...
```

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

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

- CodeReady Workspaces を、ワークスペースを1つ目の永続ボリュームに、PostgreSQL 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。

- 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:deploy -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。

```

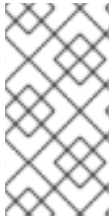
# che-workspace-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: workspace-pv-volume
  labels:
    type: local
spec:
  storageClassName: workspace-storage
  capacity:

```

```
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。

## 4.13. 信頼できない TLS 証明書の CODEREADY WORKSPACES へのインポート

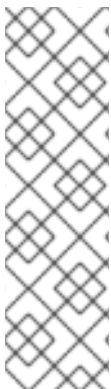
CodeReady Workspaces コンポーネントの内部通信は、デフォルトでは TLS で暗号化されます。CodeReady Workspaces コンポーネントのプロキシ、ソースコードリポジトリ、アイデンティティプロバイダーなどの外部サービスとの通信では、TLS を使用する必要がある場合があります。これらの通信では、信頼できる認証局が署名する TLS 証明書を使用する必要があります。

CodeReady Workspaces コンポーネントまたは外部サービスで使用する証明書が信頼できない CA によって署名される場合、CodeReady Workspaces インストールで CA 証明書をインポートする必要があります。そのため、すべての CodeReady Workspaces コンポーネントがそれらを信頼できる CA によって署名されているものと見なす必要があります。

この追加が必要になる可能性のある典型的なケースは以下のとおりです。

- 基礎となる OpenShift クラスタが信頼されていない CA によって署名される TLS 証明書を使用する場合
- CodeReady Workspaces サーバーまたはワークスペースコンポーネントが、信頼できない CA で署名された TLS 証明書を使用する RH-SSO や Git サーバーなどの外部サービスに接続する場合

CodeReady Workspaces は、CodeReady Workspaces namespace のラベルが付いた ConfigMap を TLS 証明書のソースとして使用します。ConfigMap には、それぞれが任意の数の証明書を持つキーの任意の数を指定できます。



### 注記

クラスタに、[クラスタ全体のプロキシ設定](#)を使用して追加されたクラスタ全体の信頼される CA 証明書が含まれる場合、CodeReady Workspaces Operator はそれらを検知し、それらをこの ConfigMap に自動的に挿入します。

- CodeReady Workspaces は ConfigMap に **config.openshift.io/inject-trusted-cabundle="true"** のラベルを自動的に付けます。
- このアノテーションに基づいて、OpenShift は ConfigMap の **ca-bundle.crt** キー内にクラスタ全体で信頼される CA 証明書を自動的に挿入します。

### 4.13.1. 新規 CA 証明書の CodeReady Workspaces への追加

本書は、CodeReady Workspaces のインストール前や、CodeReady Workspaces がすでにインストールされ、実行されている場合に使用できます。



#### 注記

2.5.1 より前の CodeReady Workspaces バージョンを使用している場合は、[本書](#)で追加の TLS 証明書を適用する方法について参照してください。

#### 前提条件

- **oc** ツールが利用可能である。
- CodeReady Workspaces の namespace が存在する。

#### 手順

1. インポートする必要のある証明書をローカルファイルシステムに保存します。

#### 注意

- 通常、証明書ファイルは **.pem**、**.crt**、**.ca-bundle** などの Base64 ASCII ファイルとして保存されます。ただし、それらにはバイナリーエンコード (例: **.cer** ファイル) を使用することもできます。証明書ファイルを保持するすべてのシークレットでは、バイナリーでエンコードされる証明書ではなく、Base64 ASCII 証明書を使用する必要があります。
- CodeReady Workspaces はすでに予約済みのファイル名を使用して証明書を ConfigMap に自動的に挿入するため、以下の予約済みのファイル名を使用して証明書を保存しないようにしてください。
  - **ca-bundle.crt**
  - **ca.crt**

2. 必要な TLS 証明書で新規 ConfigMap を作成します。

```
$ oc create configmap custom-certs --from-file=<bundle-file-path> -n=openshift-workspaces
```

複数のバンドルを適用するには、上記のコマンドに別の **--from-file=<bundle-file-path>** フラグを追加します。または、別の ConfigMap を作成することもできます。

3. **app.kubernetes.io/part-of=che.eclipse.org** および **app.kubernetes.io/component=ca-bundle** ラベルの両方を使用して作成された ConfigMap にラベルを付けます。

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

4. CodeReady Workspaces がデプロイされていない場合は、CodeReady Workspaces をデプロイします。デプロイしない場合は、CodeReady Workspaces コンポーネントのロールアウトが完了するまで待機します。実行中のワークスペースがある場合は、変更を有効にするためにそれらを再起動する必要があります。

### 4.13.2. CodeReady Workspaces のインストールレベルでの検証

証明書の追加後に、予想通りに機能しない場合は、以下の一覧を確認してください。

- CodeReady Workspaces [Operator](#) デプロイメントの場合、**CheCluster** に適切なコンテンツと共にラベルが付けられた ConfigMap が含まれる namespace。

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

また、ConfigMap の内容を確認するには、以下を実行します。

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

- CodeReady Workspaces Pod ボリューム一覧には、**ca-certs-merged** ConfigMap をデータソースとして使用する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 pod -o json <codeready-workspaces-pod-name> -n openshift-workspaces | jq .spec.volumes
```

- CodeReady Workspaces は、証明書を CodeReady Workspaces サーバーコンテナのフォルダー **/public-certs/** にマウントします。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、設定された CodeReady Workspaces 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
```

- CodeReady Workspaces サーバーの Java トラストストアに証明書が含まれる。証明書の SHA1 フィンガープリントは、以下のコマンドで返されるトラストストア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/jboss/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
```

### 4.13.3. ワークスペースレベルでの検証

- ワークスペースを起動し、これが作成された OpenShift プロジェクトを取得し、これが起動するのを待機します。



- 以下のコマンドを使用してワークスペース Pod の名前を取得します。

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

- 以下のコマンドを使用して、ワークスペース Pod の Theia IDE コンテナの名前を取得します。

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

- ワークスペース namespace 内に作成されている必要がある **ca-certs** ConfigMap を検索します。

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

- ca-certs** ConfigMap のエントリーに事前に追加した追加エントリーがすべて含まれていることを確認します。これには、予約されている **ca-bundle.crt** エントリーが含まれる場合もあります。

```
$ oc get cm ca-certs -n <workspace namespace> -o json | jq -r '.data | keys[]'
ca-bundle.crt
source-config-map-name.data-key.crt
```

- ca-certs** ConfigMap がワークスペース Pod のボリュームとして追加されていることを確認します。

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

- ボリュームがコンテナ（とくに Theia IDE コンテナ）にマウントされていることを確認します。

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

- Theia IDE コンテナの **/public-certs** フォルダを検査し、その内容が **ca-certs** ConfigMap のエントリーの一覧と一致することを確認します。

```
$ oc exec <workspace pod name> -c <theia ide container name> -n <workspace
```

```
namespace> -- ls /public-certs
ca-bundle.crt
source-config-map-name.data-key.crt
```

## 4.14. コンポーネント間の通信での外部 DNS 名と内部 DNS 名間の切り替え

デフォルトでは、新規の CodeReady Workspaces デプロイメントは、CodeReady Workspaces サーバー、RH-SSO、レジストリー間の通信に OpenShift サービス DNS 名を使用します。これは以下に役立ちます。

- プロキシ、証明書、およびファイアウォールの問題の回避
- トラフィックの高速化

このタイプの通信は、OpenShift Route クラスターのホスト名を使用するコンポーネント間の通信の外部の方法に代わるものです。以下の状況では、OpenShift 内部 DNS 名の使用はサポートされません。コンポーネント間の通信で内部クラスターのホスト名の使用を無効にすると、外部 OpenShift Route を使用した通信が有効になります。

### OpenShift における内部のコンポーネント間通信の制限

- CodeReady Workspaces コンポーネントは、マルチクラスター OpenShift 環境全体にデプロイされます。
- OpenShift NetworkPolicies は namespace 間の通信を制限します。

以下のセクションでは、OpenShift Route の外部のコンポーネント間の通信を有効/無効に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。
- OpenShift で実行される CodeReady Workspaces のインスタンス。

### 手順

コンポーネント間の通信方法を外部と内部で切り替えるには、カスタムリソース (CR) に対する更新が必要です。

1. コンポーネント間の通信で外部 OpenShift ルートを使用するには、以下を実行します。

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

2. コンポーネント間の通信で内部 OpenShift DNS 名を使用するには、以下を実行します。

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

## 4.15. RED HAT CODEREADY WORKSPACES ログインページの RH-SSO CODEREADY-WORKSPACES-USERNAME-READONLY テーマの設定

以下の手順は、OpenShift OAuth サービスが有効にされているすべての CodeReady Workspaces インスタンスに関連します。

事前に作成した namespace のユーザーが Red Hat CodeReady Workspaces ダッシュボードに初めてログインする際に、ユーザーがアカウント情報を更新できるページが表示されます。ユーザー名を変更することはできますが、OpenShift ユーザー名に一致しないユーザー名を選択すると、ユーザーのワークスペースは実行されません。これは、CodeReady Workspaces が存在しない namespace、ユーザーの OpenShift ユーザー名から派生する名前の使用を試行し、ワークスペースの作成を試行することによって生じます。これを防ぐには、RH-SSO にログインし、テーマの設定を変更します。

#### 4.15.1. RH-SSO へのログイン

以下の手順では、OpenShift プラットフォームのルートとして機能する RH-SSO にログインする方法を説明します。RH-SSO にログインするには、ユーザーは RH-SSO URL とユーザーの認証情報を最初に取得する必要があります。

##### 前提条件

- **oc** ツールがインストールされている。
- **oc** ツールを使用して OpenShift クラスターにログインしている。

##### 手順

1. ユーザーの RH-SSO ログインを取得します。

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

2. ユーザーの RH-SSO パスワードを取得します。

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

3. RH-SSO URL を取得します。

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

4. ブラウザーで URL を開き、取得したログインとパスワードを使用して RH-SSO にログインします。

#### 4.15.2. RH-SSO codeready-workspaces-username-readonly テーマの設定

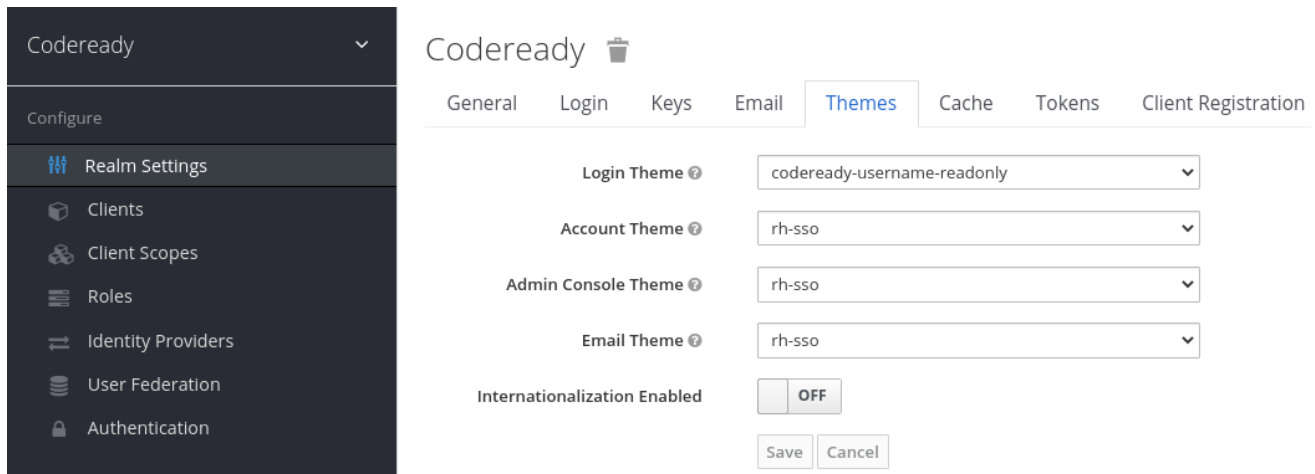
##### 前提条件

- OpenShift で実行される CodeReady Workspaces のインスタンス。
- RH-SSO サービスにログインしている。

##### 手順

ユーザー名を変更したら、Login Theme オプションを **readonly** に設定します。

1. 左側のメイン **Configure** メニューで、**Realm Settings** を選択します。



1. **Themes** タブに移動します。
2. **Login Theme** フィールドで、**codeready-workspaces-username-readonly** オプションを選択し、**Save** ボタンをクリックして変更を適用します。

## 4.16. シークレットをファイルまたは環境変数として RED HAT CODEREADY WORKSPACES コンテナにマウントする

シークレットは、ユーザー名、パスワード、認証トークン、設定などの機密データを暗号化された形式で保存する OpenShift オブジェクトです。

ユーザーは、機密データが含まれる OpenShift シークレットを Red Hat CodeReady Workspaces コンテナにマウントできます。

- ファイル
- 環境変数

マウントプロセスでは、標準の OpenShift マウントメカニズムを使用しますが、追加のアノテーションとラベル付けが必要です。

### 4.16.1. シークレットをファイルとして Red Hat CodeReady Workspaces コンテナにマウントする

#### 前提条件

- CodeReady Workspaces の実行中のインスタンス。CodeReady Workspaces のインスタンスをインストールするには、[CodeReady Workspaces のインストール](#)について参照してください。

#### 手順

1. CodeReady Workspaces ワークスペースがデプロイされる OpenShift プロジェクトで新規の OpenShift シークレットを作成します。作成される予定のシークレットのラベルは、ラベルのセットと一致する必要があります。
  - **app.kubernetes.io/part-of: che.eclipse.org**
  - **app.kubernetes.io/component: <DEPLOYMENT\_NAME>-secret**

ここで、**<DEPLOYMENT\_NAME>** は、 **postgres**、 **keycloak**、 **devfile-registry**、 **plugin-registry** または **codeready** のデプロイメントのいずれかになります。

例4.5 たとえば、以下のようになります。

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

アノテーションは、指定されるシークレットがファイルとしてマウントされていることを示す必要があります。アノテーションの値を設定します。

- **che.eclipse.org/mount-as: file**: シークレットがファイルとしてマウントされていることを示します。
- **che.eclipse.org/mount-path: <FOO\_ENV>**: 必要なマウントパスを指定します。

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

OpenShift シークレットには複数の項目が含まれる可能性があり、その名前はコンテナにマウントされる必要なファイル名と一致する必要があります。

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

これにより、**ca.crt** という名前のファイルが CodeReady Workspaces コンテナの **/custom-certificates** パスにマウントされます。

#### 4.16.2. シークレットを環境変数として Red Hat CodeReady Workspaces コンテナにマウントする

## 前提条件

- Red Hat CodeReady Workspaces の実行中のインスタンス。Red Hat CodeReady Workspaces のインスタンスをインストールするには、[CodeReady Workspaces のインストール](#)について参照してください。

## 手順

- CodeReady Workspaces ワークスペースがデプロイされる OpenShift プロジェクトで新規の OpenShift シークレットを作成します。作成される予定のシークレットのラベルは、ラベルのセットと一致する必要があります。

- app.kubernetes.io/part-of: che.eclipse.org**
- app.kubernetes.io/component: <DEPLOYMENT\_NAME>-secret**

ここで、<DEPLOYMENT\_NAME> は、**postgres**、**keycloak**、**devfile-registry**、**plugin-registry** または **codeready** のデプロイメントのいずれかになります。

例4.6 たとえば、以下ようになります。

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

アノテーションは、指定されるシークレットが環境変数としてマウントされていることを示す必要があります。アノテーションの値を設定します。

- che.eclipse.org/mount-as: env**: シークレットが環境変数としてマウントされていることを示します。
- che.eclipse.org/env-name: <FOO\_ENV>**: シークレットキーの値をマウントするために必要な環境変数名を指定します。

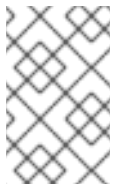
```
apiVersion: v1
kind: Secret
metadata:
  name: custom-settings
annotations:
  che.eclipse.org/env-name: FOO_ENV
  che.eclipse.org/mount-as: env
labels:
  ...
data:
  mykey: myvalue
```

これにより、**FOO\_ENV** という名前の環境変数と値 **myvalue** が CodeReady Workspaces コンテナにプロビジョニングされます。

シークレットに複数のデータ項目がある場合、環境変数の名前は以下のようにそれぞれのデータキーについて指定される必要があります。

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

これにより、**FOO\_ENV**、**OTHER\_ENV** という名前の、**myvalue** および **othervalue** の値を持つ 2 つの環境変数が CodeReady Workspaces コンテナにプロビジョニングされます。



### 注記

OpenShift シークレットのアノテーション名の最大長さは 63 文字です。ここで、9 文字は、/ で終わるプレフィックス用に予約されます。これは、シークレットに使用できるキーの最大長さの制限として機能します。

## 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 2.6 から CodeReady Workspaces 2.7 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。

OperatorHub は **Automatic** および **Manual** アップグレードストラテジーをサポートします。**Automatic**:: Operator アップグレードプロセスは、Operator の新規バージョンが公開されると開始されます。**Manual**:: 更新は、Operator の新規バージョンが公開されるたびに手動で承認される必要があります。

#### 5.1.1. OperatorHub での CodeReady Workspaces の承認ストラテジーの指定

##### 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、承認ストラテジーを指定します。
  - **Approval: Automatic**  
または  
**Approval: Manual**



## Change Update Approval Strategy

What strategy is used for approving updates?

☐ Automatic (default)

New updates will be installed as soon as they become available.

☒ Manual

New updates need to be manually approved before installation begins.

Cancel

Save

### 5.1.2. OperatorHub での CodeReady Workspaces の手動によるアップグレード

#### 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。
- サブスクリプションの承認ストラテジーは、**Manual** に設定されます。

#### 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** タブに移動します。承認を必要とするアップグレードは **Upgrade Status** の横に表示されます (例: **1 requires approval**)。
5. **1 requires approval** をクリックしてから、**Preview Install Plan** をクリックします。
6. アップグレードに利用可能なリソースとして一覧表示されているリソースを確認し、**Approve** をクリックします。

#### 検証手順

1. **Operators → Installed Operators** ページに移動し、アップグレードの進捗をモニターします。完了すると、ステータスは **Succeeded** および **Up to date** に変更されます。
2. 2.7 のバージョン番号がページ下部に表示されます。

## 関連情報

- OpenShift ドキュメントの [インストールされた Operator のアップグレード](#) についてのセクション。

## 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** 2.7 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 2.6 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 リポジトリに再度プッシュします。
2. CodeReady Workspaces 2.6 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 server:update -n <openshift-workspaces>
```



#### 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. 2.7 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 2.6 からバージョン 2.7 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` の方法を使用して、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 2.6。 [「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** 2.7 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設定を作成します。

### 5.3.2. オフラインレジストリーイメージのビルド

#### 5.3.2.1. オフラインの devfile レジストリーイメージのビルド

本セクションでは、オフラインの devfile レジストリーイメージをビルドする方法を説明します。外部インターネットのリソースを使用せずにワークスペースを起動するには、このイメージをビルドする必要があります。このイメージには、devfile で **zip** ファイルとして参照されるすべてのサンプルプロジェクトが含まれます。

#### 前提条件:

- `podman` または `docker` の実行中のインストール。

#### 手順

1. devfile レジストリーリポジトリのクローンを作成し、デプロイするバージョンをチェックアウトします。

```
$ git clone git@github.com:redhat-developer/codeready-workspaces.git
$ cd codeready-workspaces
$ git checkout crw-2.7-rhel-8
```

2. オフラインの devfile レジストリーイメージをビルドします。

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



#### 注記

**build.sh** スクリプトの詳細オプションを表示するには、**--help** パラメーターを使用します。

#### 関連情報

- [https://access.redhat.com/documentation/en-us/red\\_hat\\_codeready\\_workspaces/2.7/html-single/administration\\_guide/index#customizing-the-registries\\_crw](https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.7/html-single/administration_guide/index#customizing-the-registries_crw).

#### 5.3.2.2. オフラインプラグインレジストリーイメージのビルド

本セクションでは、オフラインのプラグインレジストリーイメージをビルドする方法を説明します。外部インターネットのリソースを使用せずにワークスペースを起動するには、このイメージをビルドする必要があります。イメージには、プラグインのメタデータとすべてのプラグインまたは拡張アーティファクトが含まれます。

#### 前提条件

- NodeJS 12.x
- yarn の実行中のバージョン。「[Installing Yarn](#)」を参照してください。
- **./node\_modules/.bin** は **PATH** 環境変数に含まれている。
- [podman](#) または [docker](#) の実行中のインストール。

#### 手順

1. プラグインレジストリーリポジトリのクローンを作成し、デプロイするバージョンをチェックアウトします。

```
$ git clone git@github.com:redhat-developer/codeready-workspaces.git
$ cd codeready-workspaces
$ git checkout crw-2.7-rhel-8
```

2. オフラインプラグインレジストリーイメージをビルドします。

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



## 注記

**build.sh** スクリプトの詳細オプションを表示するには、**--help** パラメーターを使用します。

## 関連情報

- [https://access.redhat.com/documentation/en-us/red\\_hat\\_codeready\\_workspaces/2.7/html-single/administration\\_guide/index#customizing-the-registries\\_crw](https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/2.7/html-single/administration_guide/index#customizing-the-registries_crw).

## 5.3.3. 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

<b>&lt;source-image&gt;</b>	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)。
<b>&lt;target-registry&gt;</b>	ターゲットコンテナイメージレジストリーのホスト名およびポート。
<b>&lt;target-organization&gt;</b>	ターゲットのコンテナイメージレジストリー内の組織
<b>&lt;target-image&gt;</b>	ターゲットのコンテナイメージレジストリーのイメージ名とダイジェスト。
<b>&lt;target-user&gt;</b>	ターゲットのコンテナイメージレジストリーのユーザー名。
<b>&lt;target-password&gt;</b>	ターゲットのコンテナ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>
```

## 注記

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	<b>stacks-</b> 、 <b>plugin-</b> または <b>-openj9-</b> ではない
Workspaces	<b>stacks-</b> 、 <b>plugin-</b>
IBM Z および IBM Power Systems	<b>-openj9-</b>

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/configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac	configbump-rhel8@sha256:30f61524365f0d36bbe1208df77dd5cbe75b3f9e5c979305566e46ccac139dac
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9	crw-2-rhel8-operator@sha256:df78dac12257c42910cc98e3cf7cafab628012c19b3e4104f85f0567346f45d9
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55	devfileregistry-rhel8@sha256:58e961fa91492fd13ccb2c39afb201431f187301a2a192ab683ee202c9fe8c55
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7	jwtproxy-rhel8@sha256:79783bfaedce74edcb9681baab0a33dd40268f721642c31ca5319b4b47219cb7
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd	machineexec-rhel8@sha256:a493fcb94465bdb2c61250a0cacd95b0b5bb46618e9b5fd49e5902341ed0fcd
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc	plugin-java11-openj9-rhel8@sha256:d7facc17f95bcfc23b32487346c82d2e23e6efe4d595a1b782e94f54aa636bbc
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99	plugin-java11-rhel8@sha256:641e223f5efbc32bab3461aa000e3a50a5dcca063331322158d1c959129ffd99

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cd b2e3d9bbca51db128c7289bcfb9da505d7 15bd75	plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cd b2e3d9bbca51db128c7289bcfb9da505d7 15bd75
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cd b2e3d9bbca51db128c7289bcfb9da505d7 15bd75	plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cd b2e3d9bbca51db128c7289bcfb9da505d7 15bd75
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cd b2e3d9bbca51db128c7289bcfb9da505d7 15bd75	plugin-java8-openj9-rhel8@sha256:1e84507ef957ed0ad8384cd b2e3d9bbca51db128c7289bcfb9da505d7 15bd75
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763 b431744790a89acd5c6d197316b694693b5 8c19770	plugin-java8-rhel8@sha256:5b2df65e7ec4676a43b763 b431744790a89acd5c6d197316b694693b5 8c19770
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f 990e9fab9baa47f659ef9450b7881072e3cb 40399	plugin-kubernetes-rhel8@sha256:5821feb70c74ed560a372f 990e9fab9baa47f659ef9450b7881072e3cb 40399
registry.redhat.io/codeready-workspaces/plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a 950cc3cbe21ed7301c65f84bb509fa2b6e71 fa81d	plugin-openshift-rhel8@sha256:7772bc9073e64713ebbf1a 950cc3cbe21ed7301c65f84bb509fa2b6e71 fa81d
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6cc db8c303f32d046f7eccf9f452eb30e615f2a0 bf0e	pluginbroker-artifacts-rhel8@sha256:dc191ef97b01d0afedab6cc db8c303f32d046f7eccf9f452eb30e615f2a0 bf0e
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505 a0fa6f96969cf8cc4aa8c4db95b40626f9585 4a525	pluginbroker-metadata-rhel8@sha256:dbd839715c80db641c1505 a0fa6f96969cf8cc4aa8c4db95b40626f9585 4a525



<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:c9f48f247cff27280587aeff54cea5d8a27e0eb55c99a73726cd7d575db7fbcc	pluginregistry-rhel8@sha256:c9f48f247cff27280587aeff54cea5d8a27e0eb55c99a73726cd7d575db7fbcc
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57	server-rhel8@sha256:feb6c83be2b1e6edc56287d2c9ed66a82522a297f88b495aeddd0778fb9d1f57
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb	stacks-cpp-rhel8@sha256:4bc877635a0feae47d259a232cca84130dc1f36890f76e39f422024372830bcb
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2	stacks-dotnet-rhel8@sha256:a61038e596c0c6104ae86cf4c5af5c60a6126feefa6e6585c540de2c48b723a2
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0	stacks-golang-rhel8@sha256:4ecb4f5fe6917a0e54cdaa8bb8332a06472debc8a12e8c948d7abbb6e90a95f0
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db	stacks-php-rhel8@sha256:d07364b8556e2f6689fa59fafefbaad3bb8c63b47e3e51be59521d38816a13db
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3	theia-endpoint-rhel8@sha256:bbd5b5fce80594d68a266128f607176a2f392829b969deafd848306d90c265e3
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0	theia-rhel8@sha256:3713798c7f61c3863afd4f501806df2fe462d8e3be37ab9e572940bf7a6facc0

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed	traefik-rhel8@sha256:c7ab18087c660f35386268053f29ebd2dc55163d2fd7956f0fdc227938b136ed
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce	eap-xp1-openj9-11-openshift-rhel8@sha256:42d7a7264314b9ef8399bd a08ea61362887e4c1a88addb4c4f9f3b5d9 d3169ce
registry.redhat.io/jboss-eap-7/eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c 1992663258c0016c80247f507fd1c39cf9a73 da833	eap-xp1-openjdk11-openshift-rhel8@sha256:94e1cd4eb4196a358e301c 1992663258c0016c80247f507fd1c39cf9a73 da833
registry.redhat.io/jboss-eap-7/eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b4 6ade182d0f981362f36b7e6fb9c7d8531ac6 39fe0	eap73-openjdk8-openshift-rhel7@sha256:24dea0cfc154a23c1aeb6b4 6ade182d0f981362f36b7e6fb9c7d8531ac6 39fe0
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f	sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f	sso74-openj9-openshift-rhel8@sha256:9297414d1cad8f86871f240 c1c0ae324f7d1a3285c22ac7dd878bfcf3c5 9a75f

<source-image>	<target-image>
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:c0045cd676e06eb17083a44c4b90b29b11ddb40e1fb6a7b651384cf0960f5158	sso74-openshift-rhel8@sha256:c0045cd676e06eb17083a44c4b90b29b11ddb40e1fb6a7b651384cf0960f5158
registry.redhat.io/rhel8/postgresql-96@sha256:5b5bf623d89deda89250f422d352b122bce9533b902b5474f9c63a9facc7a6f1	postgresql-96@sha256:5b5bf623d89deda89250f422d352b122bce9533b902b5474f9c63a9facc7a6f1
registry.redhat.io/rhsc1/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73
registry.redhat.io/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6	ubi8ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187aadb32e89fd83fa455ebaa6

## 検証手順

- 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>
```

## 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。

## 5.3.4. 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** の方法を使用して、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 2.6。「[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** 2.7 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

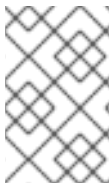
1. CodeReady Workspaces 2.6 インスタンスで実行されているすべてのワークスペースで、変更を保存し、Git リポジトリに再度プッシュします。
2. CodeReady Workspaces 2.6 インスタンスのすべてのワークスペースを停止します。
3. 次のコマンドを実行します。

```
$ crwctl server:update --che-operator-image=<image-registry>/<organization>/crw-2-rhel8-operator:2.7 -n openshift-workspaces
```

- **<image-registry>**: 制限された環境でアクセス可能なコンテナ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. 2.7 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. OPENSIFT 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

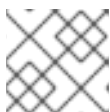
本セクションでは、OpenShift Administrator パースペクティブのメインメニューを使用して、クラスターから CodeReady Workspaces をアンインストールする方法を説明します。

#### 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。



#### 注記

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。



#### 注記

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. OPENSIFT 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ります。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.v2.7** 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.v2.7	Red Hat CodeReady Workspaces	2.7	red-hat-codeready-workspaces.v2.6	Succeeded

6. CodeReady Workspaces CSV を削除します。

```
$ oc delete csv red-hat-codeready-workspaces.v2.7
clusterserviceversion.operators.coreos.com "red-hat-codeready-workspaces.v2.7" 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. 削除する CodeReady Workspaces namespace の名前をエクスポートします。

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

3. ユーザーのアクセストークンおよび Keycloak URL をエクスポートします。

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

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

4. UAT を使用してサーバーを停止します。

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

5. プロジェクトおよび CodeReady Workspaces デプロイメントを削除します。

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

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

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

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

6. プロジェクトについての情報を一覧表示して、削除が正常に実行されていることを確認します。

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

7. 指定された **ClusterRoleBinding** を削除します。

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