



Red Hat CodeReady Workspaces 2.11

インストールガイド

Red Hat CodeReady Workspaces 2.11のインストール

Red Hat CodeReady Workspaces 2.11 インストールガイド

Red Hat CodeReady Workspaces 2.11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 ® is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

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

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

All other trademarks are the property of their respective owners.

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	22
3.1. OPERATORHUB を使用した OPENSHIFT 4 への CODEREADY WORKSPACES のインストール	22
3.1.1. Red Hat CodeReady Workspaces Operator のインストール	22
3.1.2. Red Hat CodeReady Workspaces Operator のインスタンスの作成	23
3.2. CLI を使用した CODEREADY WORKSPACES の OPENSHIFT 4 へのインストール	24
3.3. CODEREADY WORKSPACES の OPENSHIFT CONTAINER PLATFORM 3.11 へのインストール	24
3.3.1. crwctl CLI 管理ツールのインストール	24
3.3.2. Operator を使用した CodeReady Workspaces の OpenShift 3 へのインストール	25
3.4. 制限された環境での CODEREADY WORKSPACES のインストール	26
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	27
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	28
3.4.2.1. プライベートレジストリーの準備	28
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	34
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	34
3.4.2.2.2. 制限された環境での CheCluster カスタムリソース のカスタマイズ	34
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	35
3.4.3. プロキシーの後ろでインストールするための CodeReady Workspaces カスタムリソースの準備	35
第4章 CODEREADY WORKSPACES の設定	37
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	37
4.1.1. Operator を使用した CodeReady Workspaces サーバーの詳細設定について	37
4.1.2. CodeReady Workspaces サーバーコンポーネントのシステムプロパティー参照	38
4.1.2.1. CodeReady Workspaces サーバー	38
4.1.2.2. 認証パラメーター	45
4.1.2.3. 内部	47
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	61
4.1.2.6. 実験的なプロパティー	62
4.1.2.7. 主なWebSocketエンドポイントの設定	66
4.1.2.8. CORS 設定	66
4.1.2.9. Factory のデフォルト	66
4.1.2.10. devfile のデフォルト	67
4.1.2.11. Che システム	70
4.1.2.12. Workspace の制限	70
4.1.2.13. ユーザーワークスペースの制限	71
4.1.2.14. 組織ワークスペースの制限	72
4.1.2.15. マルチユーザー固有の OpenShift インフラストラクチャー設定	72
4.1.2.16. Keycloak の設定	73
4.2. ワークスペーススタートプロジェクトの設定	76
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	78
4.2.2. 互換性のないユーザー名またはユーザー ID の処理	78
4.2.3. 各ユーザーのプロジェクトの事前作成	78
4.2.4. namespace のラベル付け	79

4.3. ストレージストラテジーの設定	80
4.3.1. codeready-workspaces ワークスペースのストレージストラテジー	80
4.3.1.1. common PVC ストラテジー	81
4.3.1.2. per-workspace PVC ストラテジー	82
4.3.1.3. unique PVC ストラテジー	82
4.3.1.4. サブパスが PVC で使用される方法	83
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	83
4.3.2.1. Operator を使用した PVC ストラテジーの設定	84
4.4. ストレージタイプの設定	84
4.4.1. 永続ストレージ	85
4.4.2. 一時ストレージ	85
4.4.3. 非同期ストレージ	85
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	86
4.4.5. 非同期ストレージ Pod のアイドリング	87
4.5. ユーザーが実行できるワークスペース数の設定	87
4.5.1. Operator を使用したユーザーが実行できるワークスペースの数の設定	87
4.6. ユーザーが作成できるワークスペースの数の設定	88
4.6.1. Operator を使用したユーザーが作成できるワークスペースの数の設定	88
4.7. ワークスペース公開ストラテジーの設定	89
4.7.1. Operator を使用したワークスペース公開ストラテジーの設定	89
4.7.2. ワークスペース公開ストラテジー	90
4.7.2.1. Multihost ストラテジー	90
4.7.2.2. 単一ホストストラテジー	90
4.7.2.2.1. devfile エンドポイント : single-host	91
4.7.2.2.2. devfile エンドポイント : multi-host	91
4.7.3. セキュリティーに関する考慮事項	92
4.7.3.1. JSON Web トークン (JWT) プロキシー	92
4.7.3.2. 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	92
4.7.3.3. 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	92
4.7.3.4. クロスサイトリクエストフォージェリー攻撃	93
4.7.3.5. フィッシング攻撃	93
4.8. ワークスペース NODESELECTOR の設定	93
4.9. RED HAT CODEREADY WORKSPACES サーバーのホスト名の設定	93
4.10. OPENSHIFT ルートの設定	95
4.11. ルーターのシャード化と連携するように OPENSHIFT ルートを設定	96
4.12. 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	100
4.13. ストレージクラスを使用した CODEREADY WORKSPACES のインストール	101
4.14. 信頼できない TLS 証明書の CODEREADY WORKSPACES へのインポート	105
4.14.1. 新規 CA 証明書の CodeReady Workspaces への追加	106
4.14.2. CodeReady Workspaces のインストールレベルでの検証	107
4.14.3. ワークスペースレベルでの検証	108
4.15. 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	109
4.16. 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
4.16.1. RH-SSO へのログイン	110
4.16.2. RH-SSO codeready-workspaces-username-readonly テーマの設定	111
4.17. シークレットまたは CONFIGMAP をファイルまたは環境変数として CODEREADY WORKSPACES コンテナーにマウントする	111
4.17.1. シークレットまたは ConfigMap をファイルとして CodeReady Workspaces コンテナーにマウントする	112
4.17.2. シークレットまたは ConfigMap を環境変数として CodeReady Workspaces コンテナーにマウントする	114
4.18. DEV WORKSPACE エンジンの有効化	117

第5章 CODEREADY WORKSPACES のアップグレード	119
5.1. OPERATORHUB を使用した CODEREADY WORKSPACES のアップグレード	119
5.1.1. OperatorHub での CodeReady Workspaces の承認ストラテジーの指定	119
5.1.2. OperatorHub での CodeReady Workspaces の手動によるアップグレード	120
5.2. CLI 管理ツールを使用した CODEREADY WORKSPACES のアップグレード	121
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	121
5.3.1. 制限された環境でのネットワーク接続について	122
5.3.2. オフラインレジストリーイメージのビルド	122
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	122
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	123
5.3.3. プライベートレジストリーの準備	124
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	130
5.4. 「PER USER」以外のプロジェクトストラテジーを使用する CODEREADY WORKSPACES のアップグレード	131
5.4.1. CodeReady Workspaces のアップグレードおよびユーザーデータのバックアップ	131
5.4.2. CodeReady Workspaces のアップグレードおよびユーザーデータの損失	132
第6章 CODEREADY WORKSPACES のアンインストール	133
6.1. OPENSHIFT WEB コンソールを使用した OPERATORHUB のインストール後の CODEREADY WORKSPACES のアンインストール	133
6.2. OPENSHIFT CLI を使用した OPERATORHUB のインストール後の CODEREADY WORKSPACES のアンインストール	134
6.3. CRWCTL インストール後の CODEREADY WORKSPACES のアンインストール	135

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 プロパティー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、[弊社の CTO、Chris Wright のメッセージ](#) を参照してください。

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 の CodeReady Workspaces 2.11 の可用性およびサポートされるインストール方法を説明します。

表1.1 OpenShift Container Platform および OpenShift Dedicated での CodeReady Workspaces 2.11 でサポートされるデプロイメント環境

プラットフォーム	アーキテクチャー	デプロイメント方法
OpenShift Container Platform 3.11	AMD64 および Intel 64 (x86_64)	crwctl
OpenShift Container Platform 4.6	AMD64 および Intel 64 (x86_64)	OperatorHub, crwctl
OpenShift Container Platform 4.6	IBM Z (s390x)	OperatorHub, crwctl
OpenShift Container Platform 4.6	IBM Power Systems (ppc64le)	OperatorHub, crwctl
OpenShift Container Platform 4.8	AMD64 および Intel 64 (x86_64)	OperatorHub, crwctl
OpenShift Container Platform 4.8	IBM Z (s390x)	OperatorHub, crwctl
OpenShift Container Platform 4.8	IBM Power Systems (ppc64le)	OperatorHub, crwctl
OpenShift Dedicated 4.8	AMD64 および Intel 64 (x86_64)	アドオン



注記

IBM Z (s390x) での OpenShift Container Platform への CodeReady Workspaces のデプロイのサポートは、現在テクノロジープレビュー機能としてのみ利用できます。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。テクノロジープレビュー機能のサポートレベルの詳細は、「テクノロジープレビュー機能のサポート範囲」を参照してください。

第2章 CODEREADY WORKSPACES インストールの設定

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

2.1. CHECLUSTER カスタムリソースについて

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

CheCluster カスタムリソース

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

Red Hat CodeReady Workspaces Operator の役割

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

OpenShift プラットフォームの役割

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

例2.1 CodeReady Workspaces サーバーコンポーネントの主なプロパティーの設定

1. ユーザーは、**server** に関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。

関連資料

- [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

このセクションでは、**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 メトリクス収集に関する **CheCluster** カスタムリソース **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する **CheCluster** カスタムリソースの **server** 設定。

プロパティ	詳細
airGapContainerRegistryHostname	イメージのプルに使用する別のコンテナーレジストリーに対する、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ー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。

プロパティ	詳細
airGapContainerRegistryOrganization	イメージのプルに使用する別のコンテナーレジストリー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ーアイメージで定義されるコンテナーレジストリーの組織を上書きします。これは、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。
allowUserDefinedWorkspacesNamespaces	非推奨。このフラグの値は無視されます。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 true 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 のコンマ区切りの一覧。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 false です。
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 che 、CodeReady Workspaces インストールの場合は codeready 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。 cheHostTLSSecret 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ー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。 cheHost フィールドを参照してください。
chelImage	Che デプロイメントで使用されるコンテナーアイメージを上書きします。これには、コンテナーアイメージタグは含まれません。Operator によって提供されるデフォルトのコンテナーア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。
chelImagePullPolicy	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、 nightly または latest イメージの場合は Always で、他の場合は IfNotPresent です。
chelImageTag	Che デプロイメントで使用されるコンテナーア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 : INFO または DEBUG 。デフォルトは INFO です。

プロパティ	詳細
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	CheCluster カスタムリソース (CR) の他のフィールドからすでに生成されている値に加えて、Che サーバーによって使用される生成された che ConfigMap に適用される追加の環境変数のマップ。 custom CheProperties に他の CR フィールドから che ConfigMap に生成されるプロパティーが含まれる場合、代わりに カスタム CheProperties に定義された値が使用されます。
dashboardCpuLimit	ダッシュボード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。
dashboardCpuRequest	ダッシュボード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。
dashboardImage	ダッシュボードのデプロイメントで使用されるコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。
dashboardImagePullPolicy	ダッシュボード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、 nightly または latest イメージの場合は Always で、他の場合は IfNotPresent です。
dashboardIngress	ダッシュボード Ingress のカスタム設定。
dashboardMemoryLimit	ダッシュボード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。
dashboardMemoryRequest	ダッシュボード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。
dashboardRoute	ダッシュボードルートのカスタム設定。
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	詳細
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ーアイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーア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、 nightly または latest イメージの場合は Always で、他の場合は IfNotPresent です。
devfileRegistryRoute	devfile レジストリールートのカスタム設定。
devfileRegistryUrl	externalDevfileRegistries フィールドが導入されたため、非推奨になりました。
disableInternalClusterSVCNames	内部クラスターの SVC 名の使用を無効にして、トラフィックを高速化し、プロキシーの問題を回避します。
externalDevfileRegistries	外部 devfile レジストリーで、すぐに使用可能な devfiles を提供します。 (externalDevfileRegistry が false の場合) 専用の devfile レジストリーに加えて、(externalDevfileRegistry が true の場合) 専用の devfile レジストリーの代わりに、外部の devfile レジストリーを追加します。
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。 externalDevfileRegistry が true の場合には、このような専用サーバーは Operator によって起動されず、 externalDevfileRegistries フィールドで少なくとも 1 つの devfile レジストリーが設定されます。
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。 externalPluginRegistry が true の場合には、このような専用サーバーは Operator によって起動されず、 pluginRegistryUrl フィールドを手動で設定する必要があります。
gitSelfSignedCert	che-git-self-signed-cert 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。

プロパティ	詳細
nonProxyHosts	プロキシーをバイパスして、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 .<DOMAIN> および を区切り文字として使用します。たとえば、localhost my.host.com 123.42.12.32 のようになります。これは、プロキシー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ー設定に対応し、追加の設定は必要ありませんが、カスタムリソースで nonProxyHosts を定義すると、クラスター プロキシー設定からのプロキシー以外のホストと、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 (https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html) を参照してください。 proxyURL フィールドも参照してください。
pluginRegistryCpuLimit	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 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。
pluginRegistryImage	プラグインレジストリーのデプロイメントで使用されるコンテナーイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーイメージを使用するには、これを省略するか、または空のままにします。
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。
pluginRegistryMemoryRequest	プラグインレジストリーのデプロイメントで使用されるメモリー要求を上書きします。デフォルトは 16Mi に設定されます。
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、 nightly または latest イメージの場合は Always で、他の場合は IfNotPresent 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。 externalPluginRegistry 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	プロキシーサーバーのパスワード。プロキシー設定が必要である場合にのみ使用します。 proxyURL 、 proxyUser および proxySecret フィールドを参照してください。
proxyPort	プロキシーサーバーのポート。プロキシーの設定が必要な場合にのみ使用します。 proxyURL および nonProxyHosts フィールドも参照してください。

プロパティ	詳細
proxySecret	プロキシサーバーの user と password が含まれるシークレット。シークレットが定義されると、 proxyUser および proxyPassword 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ーの JAVA_OPTS および https_proxy 変数に適切な変更が加えられます。プロキシーの設定が必要な場合にのみ使用します。Operator は OpenShift クラスター全体のプロキシー設定に対応し、追加の設定は必要ありませんが、カスタムリソースで proxyUrl を定義すると、クラスタープロキシー設定が、カスタムリソースのフィールドの proxyUrl 、 proxyPort 、 proxyUser および proxyPassword で上書きされます。ドキュメント (https://docs.openshift.com/container-platform/4.4/networking/enable-cluster-wide-proxy.html) を参照してください。 proxyPort および nonProxyHosts 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。プロキシーの設定が必要な場合にのみ使用します。 proxyURL 、 proxyPassword および proxySecret フィールドも参照してください。
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。設定可能な値は、 multi-host 、 single-host 、 default-host 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 multi-host にデフォルト設定されます。 single-host は、ワークスペースがサブパスで公開された状態で Che を单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、 singleHostExposureType プロパティを参照してください。 default-host 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ー制限を上書きします。デフォルトは 1Gi に設定されます。

プロパティ	詳細
serverMemoryRequest	Che サーバーのデプロイメントで使用されるメモリー要求を上書きします。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 証明書を認識できる必要があります。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。
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ーア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ーア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	disableInternalClusterSVCNames が導入されたため非推奨になりました。
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。<username>、<userid> および <workspaceid> プレースホルダー (例: che-workspace-<username>) を使用できます。この場合、新規 namespace が各ユーザーまたはワークスペースについて作成されます。

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

プロパティ	詳細
chePostgresContainerResources	PostgreSQL コンテナーのカスタム設定
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 dbche です。

プロパティ	詳細
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 postgres 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。 externalDb 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。
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。 externalDb 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 password が含まれるシークレット。シークレットが定義されると、 chePostgresUser および chePostgresPassword は無視されます。値が省略されるか、空のままになると、以下のいずれかのシナリオが適用されます。1. chePostgresUser および chePostgresPassword が定義され、DB への接続に使用されます。2. chePostgresUser または chePostgresPassword が定義されていないため、 user には pgche のデフォルト値を、 password には自動生成された値を使用して、 che-postgres-secret という名前の新しいシークレットが作成されます。
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 pgche です。
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。 externalDb が true 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。 chePostgres 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。
postgresImagePullPolicy	PostgreSQL データベースのデプロイメントで使用されるイメージプルポリシーを上書きします。デフォルト値は、 nightly または latest イメージの場合は Always で、他の場合は IfNotPresent です。

表2.3 CodeReady Workspaces で使用される認証に関するカスタムリソース auth 設定。

プロパティ	詳細
externalIdentityProvider	専用のアイデンティティープロバイダー (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。専用のアイデンティティープロバイダー (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ー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。 externalIdentityProvider が true の場合、専用のアイデンティティープロバイダーは Operator によってデプロイされず、使用する外部アイデンティティープロバイダーの詳細を指定する必要があります。 identityProvider で始まるその他のフィールドもすべて参照してください。
gatewayAuthenticationSidecarImage	NativeUserMode が有効な場合に認証を行うゲートウェイサイドカー。 oauth2-proxy または openshift/oauth-proxy を参照してください。
gatewayAuthorizationSidecarImage	NativeUserMode が有効な場合に承認を行うゲートウェイサイドカー。 kube-rbac-proxy または openshift/kube-rbac-proxy を参照してください。
gatewayHeaderRewriteSidecarImage	非推奨。このフラグの値は無視されます。サイドカー機能が Traefik プラグインに実装されるようになりました。
identityProviderAdminUserName	アイデンティティー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 admin です。
identityProviderClientId	Che に使用されるアイデンティティープロバイダー、Keycloak、または RH-SSO の client-id の名前。外部アイデンティティー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。 externalIdentityProvider 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は、サフィックスが -public の flavour フィールドの値に設定されます。
identityProviderContainerResources	アイデンティティープロバイダーコンテナーのカスタム設定。
identityProviderImage	アイデンティティープロバイダー、Keycloak、または RH-SSO デプロイメントで使用するコンテナーアイメージを上書きします。これには、イメージタグが含まれます。Operator によって提供されるデフォルトのコンテナーア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。
identityProviderImagePullPolicy	アイデンティティープロバイダー、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、 nightly または latest イメージの場合は Always で、他の場合は IfNotPresent です。
identityProviderIngress	Ingress のカスタム設定。

プロパティ	詳細
identityProviderPassword	Keycloak 管理者ユーザーのパスワードを上書きします。外部アイデンティー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。 externalIdentityProvider 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ープロバイダー、Keycloak、または RH-SSO のパスワード外部アイデンティティー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。 externalIdentityProvider 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ープロバイダー、Keycloak、または RH-SSO の パスワード を含むシークレット。シークレットが定義されると、 identityProviderPostgresPassword 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. identityProviderPostgresPassword が定義されると、これがデータベースへの接続に使用されます。 2. identityProviderPostgresPassword が定義されていない場合は、 password に自動生成された値を使用して、 che-identity-postgres-secret という名前の新規シークレットが作成されます。
identityProviderRealm	Che に使用されるアイデンティティープロバイダー、Keycloak、または RH-SSO のレルムの名前。外部アイデンティティー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。 externalIdentityProvider 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、 flavour 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。
identityProviderSecret	アイデンティティープロバイダーの user と password が含まれるシークレット。シークレットが定義されると、 identityProviderAdminUserName および identityProviderPassword は無視されます。値が省略されるか、空のままですると、以下のいずれかのシナリオが適用されます。 1. identityProviderAdminUserName および identityProviderPassword が定義され、これらが使用されます。 2. identityProviderAdminUserName または identityProviderPassword が定義されていない場合は、 user には admin のデフォルト値を、 password には自動生成された値を使用して、 che-identity-secret という名前の新規シークレットが作成されます。
identityProviderURL	アイデンティティー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ー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。 externalIdentityProvider 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。

プロパティ	詳細
initialOpenShiftOAuthUser	OpenShift OAuth 認証の操作には、kubeadm を使用できないため、新規ユーザー アカウントを作成します。値が true の場合には、HTPasswd アイデンティティープロバイダーに新しい OpenShift OAuth ユーザーが作成されます。値が false で、ユーザーがすでに作成されている場合は、削除されます。値が空白の場合は何も実行されません。ユーザーの認証情報は、Operator によって 'openshift-config' namespace にある openshift-oauth-user-credentials シークレットに保存されます。このソリューションは OpenShift 4 プラットフォーム固有であることに注意してください。
nativeUserMode	ネイティブユーザー モードを有効にします。現在、OpenShift および DevWorkspace エンジンでのみ動作します。ネイティブユーザー モードは、Keycloak なしで OpenShift OAuth を直接アイデンティティープロバイダーとして使用します。
oAuthClientName	OpenShift 側でアイデンティティーフェデレーションを設定するために使用される OpenShift OAuthClient リソースの名前。空のままにすると自動生成されます。 OpenShiftAuth フィールドも参照してください。
oAuthSecret	OpenShift 側でアイデンティティーフェデレーションを設定するために使用される OpenShift OAuthClient リソースに設定されたシークレットの名前。空のままにすると自動生成されます。 OAuthClientName フィールドも参照してください。
openShiftAuth	アイデンティティープロバイダー (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: kubeadm 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。
updateAdminPassword	デフォルトの admin 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 false です。

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

プロパティ	詳細
postgresPVCStorageClassName	PostgreSQL データベース専用の Persistent Volume Claim (永続ボリューム要求、PVC) のストレージクラス。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 false 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 10Gi に設定されます。

プロパティ	詳細
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ーアイメージを上書きします。これには、イメージタグが含まれます。Operatorによって提供されるデフォルトのコンテナーアイメージを使用するには、これを省略するか、または空のままにします。 preCreateSubPaths フィールドも参照してください。
pvcStrategy	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)、 per-workspace (すべての宣言されたボリュームについてワークスペースごとに1つの PVC)、および unique (宣言されたボリュームごとに1つの PVC) を指定できます。デフォルトは common です。
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	詳細
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 nginx です。注意: これは Che 関連の Ingress で kubernetes.io/ingress.class 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: multi-host (ホストは Ingress で明示的に指定されます)、 single-host (ホストは指定される、パスベースのルール)、および default-host (ホストは指定されない、パスベースのルール)。server セクションの serverExposureStrategy が優先されるため、デフォルトで multi-host 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、 serverExposureStrategy 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 コンテナーが実行される FSGroup。デフォルト値は 1724 です。
securityContextRunAsUser	Che Pod およびワークスペース Pod コンテナーの実行に使用するユーザーの ID。デフォルト値は 1724 です。

プロパティ	詳細
singleHostExposureType	serverExposureStrategy が single-host 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 native 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 Traefik 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 gateway を使用して公開されます。Ingress またはゲートウェイ route 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 native です。
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。 tlsSupport 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 メトリクス収集に関する CheCluster カスタムリソース metrics 設定。

プロパティ	詳細
enable	Che サーバーエンドポイント メトリクス を有効にします。デフォルトは true です。

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	詳細
cheClusterRunning	Che インストールのステータス。 Available 、 Unavailable までやあ Available 、 Rolling Update in Progress を使用できます。
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。
devworkspaceStatus	Devworkspace サブシステムのステータス
gitHubOAuthProvisioned	アイデンティティー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。

プロパティ	詳細
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	アイデンティティー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	アイデンティティー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。
openShiftOAuthUserCredentialsSecret	HTPasswd アイデンティティープロバイダーのユーザー認証情報を含む openshift-config namespace の OpenShift OAuth シークレット。
openShiftOAuthProvisioned	アイデンティティー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 OPENSHIFT 4 への CODEREADY WORKSPACES のインストール

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

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

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

前提条件

- OpenShift 4 の実行中のインスタンスの管理者アカウント

3.1.1. 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 OpenShift Web コンソールへのアクセス。

手順

1. 左側のパネルで Operators → OperatorHub ページに移動します。
2. Filter by keyword フィールドに **Red Hat CodeReady Workspaces** を入力します。
3. **Red Hat CodeReady Workspaces** タイルをクリックします。
4. **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。
5. **Install Operator** ページで、**Install** ボタンをクリックします。

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 ページに移動します。

2. Installed Operators ページで、Red Hat CodeReady Workspaces名をクリックし、Details タブに移動します。
3. ClusterServiceVersion details セクションで、以下のメッセージが表示されるまで待機します。
 - 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まで待機します。

3.1.2. 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. 左側のパネルを使用して、Operators → Installed Operators ページに移動します。
2. Installed Operators ページで、Red Hat CodeReady Workspaces名をクリックします。
3. Operator details ページで、Details タブの Provided APIs セクションの Create instance リンクをクリックします。
これにより、Create CheCluster ページに移動します。このページには、CheCluster カスタムリソースに保存される CodeReady Workspaces インスタンスの作成に必要な設定が含まれます。
4. デフォルト値を使用し、ページの最後にある Create ボタンを使用して codeready-workspaces クラスターを作成します。
5. Operator details ページの、Red Hat CodeReady Workspaces Cluster タブで、codeready-workspaces リンクをクリックします。
6. Red Hat CodeReady Workspaces URLの出力に表示されるリンクを使用して codeready-workspaces インスタンスに移動します。



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 のインストールが完了すると、URL が表示されます。

検証

1. 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、Operator details ページの CodeReady Workspaces Cluster タブに移動します。CheClusters ページには、CodeReady Workspaces インスタンスの一覧およびそれらのステータスが表示されます。
2. 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 フィールドにはエラーメッセージが含まれます。予想される内容は **None** です。

- Red Hat CodeReady Workspaces URL フィールドには、Red Hat CodeReady Workspaces インスタンスの URL が含まれます。デプロイメントが正常に終了すると、URL が表示されます。
- Resources タブに移動します。CodeReady Workspaces デプロイメントに割り当てられたり ソースとそれらのステータスの一覧が表示されます。

関連資料

- Dashboard を使用した CodeReady Workspaces のナビゲーション
- OpenShift 4 CLI ツールを使用した CodeReady Workspaces クラスター デプロイメントの状態 の表示

3.2. CLI を使用した CODEREADY WORKSPACES の OPENSHIFT 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** を利用できます。「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

- server:deploy** コマンドの出力は以下で終わります。


```
Command server:deploy has completed successfully.
```
- CodeReady Workspaces クラスターインスタンス:\https://codeready-
<openshift_deployment_name>.<domain_name> に移動します。

3.3. CODEREADY WORKSPACES の OPENSHIFT 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 2.11 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. `$HOME/crwctl` または `/opt/crwctl` などのフォルダーにアーカイブを展開します。
4. 展開したフォルダーから `crwctl` の実行可能ファイルを実行します。この例では `$HOME/crwctl/bin/crwctl version` です。
5. オプションで、`bin` フォルダーを `$PATH`（例：`PATH=$PATH:$HOME/crwctl/bin`）に追加し、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、ツールの現在の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。
- openshift-workspaces namespace をデフォルトのインストールプロジェクトとして使用します。
- イメージを **registry.redhat.com** からプルするように OpenShift を設定します。[Red Hat コンテナーレジストリーの認証](#) を参照してください。

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、openshift-workspaces というデフォルトプロジェクトに CodeReady Workspaces インスタンスを作成します。

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

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

3.4.1. 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にのみ、Operator を OperatorHub からインストールできます。

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

手順

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

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

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

表3.1サンプルで使用されるプレースホルダー

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

<target-password>

ターゲットのコンテナーアイメージレジストリーのユーザーパスワード。

手順

- 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>**に追加します。
- /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>']
```

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

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

注記

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

使用	プレフィックスまたはキーワード
Essential	stacks- , plugin- または -openj9- ではない
Workspaces	stacks- , plugin-
IBM Z および IBM Power Systems	-openj9-

注記

openj9 のサフィックスがつくイメージは、x86_64 で使用される OpenJDK イメージと同等の **Eclipse OpenJ9** イメージです。IBM Power Systems および IBM Z は、これらのシステムでパフォーマンスを改善するために Eclipse OpenJ9 を使用します。

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@sha256:20fd31c45d769526d45eaf6738a6d4af1520a844126a2a2e510c304a81b7249a	configbump-rhel8@sha256:20fd31c45d769526d45eaf6738a6d4af1520a844126a2a2e510c304a81b7249a
registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:a41f7b950c5131a6bc08b1e094db2da9b784e6083ddaa4aa68512f3947798702	crw-2-rhel8-operator@sha256:a41f7b950c5131a6bc08b1e094db2da9b784e6083ddaa4aa68512f3947798702
registry.redhat.io/codeready-workspaces/dashboard-rhel8@sha256:1c37bdffae8cdc154d88b94ab38e868f7e33486c81b6c3bded36dfdf85b81a4	dashboard-rhel8@sha256:1c37bdffae8cdc154d88b94ab38e868f7e33486c81b6c3bded36dfdf85b81a4
registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:b164968dbd52c72f39533bec4efd3ad3cce3acb6060495e472dd9c3f2908fbcc	devfileregistry-rhel8@sha256:b164968dbd52c72f39533bec4efd3ad3cce3acb6060495e472dd9c3f2908fbcc
registry.redhat.io/codeready-workspaces/devworkspace-controller-rhel8@sha256:c88242524a9074a58bc7d20cb8411d37e7e752358ab80366533b8165bb9f95b0	devworkspace-controller-rhel8@sha256:c88242524a9074a58bc7d20cb8411d37e7e752358ab80366533b8165bb9f95b0
registry.redhat.io/codeready-workspaces/devworkspace-rhel8@sha256:c18f166f570ca572c94472b7a3bd5127b48521e777ea09dcad6f78ad66cd7a13	devworkspace-rhel8@sha256:c18f166f570ca572c94472b7a3bd5127b48521e777ea09dcad6f78ad66cd7a13
registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:44acafb02cce3d3fe8b57da2e27547b502c4088624935ffe7f3aa06a55d08bba	jwtproxy-rhel8@sha256:44acafb02cce3d3fe8b57da2e27547b502c4088624935ffe7f3aa06a55d08bba
registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:bfdd8cf61a6fad757f1e8334aa84dbf44baddf897ff8def7496bf6dbc066679d	machineexec-rhel8@sha256:bfdd8cf61a6fad757f1e8334aa84dbf44baddf897ff8def7496bf6dbc066679d

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:8d9930cd3c0b2fa72a6c0d880b4d0b330b1a7a51491f09175134dcc79f2cb376	plugin-java11-openj9-rhel8@sha256:8d9930cd3c0b2fa72a6c0d880b4d0b330b1a7a51491f09175134dcc79f2cb376
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:d0337762e71fd4badabcb38a582b2f35e7e7fc1c9c0f2e841e339d45b7bd34ed	plugin-java11-rhel8@sha256:d0337762e71fd4badabcb38a582b2f35e7e7fc1c9c0f2e841e339d45b7bd34ed
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:d7ec33ce2fa61a06fade63e2b516409c465bd5516030dd482e2f4bdb2d676c9f	plugin-java8-openj9-rhel8@sha256:d7ec33ce2fa61a06fade63e2b516409c465bd5516030dd482e2f4bdb2d676c9f
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:b2ceb0039c763e6a38aa370157b476ecb08faf8b2fb680bada774e149583d62	plugin-java8-rhel8@sha256:b2ceb0039c763e6a38aa370157b476ecb08faf8b2fb680bada774e149583d62
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:45535630e37e3e317772f36b28b47859d32ad1e82505a796139682cdbefb03b8	plugin-kubernetes-rhel8@sha256:45535630e37e3e317772f36b28b47859d32ad1e82505a796139682cdbefb03b8
registry.redhat.io/codeready-workspaces/plugin.openshift-rhel8@sha256:d2384cafc870c497913168508be0d846412c68ace9724baa37ca3c6be9aa4772	plugin-openshift-rhel8@sha256:d2384cafc870c497913168508be0d846412c68ace9724baa37ca3c6be9aa4772
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:a9bf68e6dabbaaf3e97afe4ac6e97a317e8fd9c05c88e5801fbf01aaa1ebb99	pluginbroker-artifacts-rhel8@sha256:a9bf68e6dabbaaf3e97afe4ac6e97a317e8fd9c05c88e5801fbf01aaa1ebb99
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:727f80af1e1f6054ac93cad165bc392f43c951681936b979b98003e06e759643	pluginbroker-metadata-rhel8@sha256:727f80af1e1f6054ac93cad165bc392f43c951681936b979b98003e06e759643

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/pluginregistry-rhel8@sha256:5d19f7c5c0417940c52e552c51401f729b9ec16868013e016d7b80342cd8de4e	pluginregistry-rhel8@sha256:5d19f7c5c0417940c52e552c51401f729b9ec16868013e016d7b80342cd8de4e
registry.redhat.io/codeready-workspaces/server-rhel8@sha256:e79e0a462b4dd47ecaac2f514567287c44e32437496b2c214ebc2bc0055c4aa9	server-rhel8@sha256:e79e0a462b4dd47ecaac2f514567287c44e32437496b2c214ebc2bc0055c4aa9
registry.redhat.io/codeready-workspaces/stacks-cpp-rhel8@sha256:31ef0774342bc1dbcd91e3b85d68d7a28846500f04ace7a5dfa3116c0cedfeb1	stacks-cpp-rhel8@sha256:31ef0774342bc1dbcd91e3b85d68d7a28846500f04ace7a5dfa3116c0cedfeb1
registry.redhat.io/codeready-workspaces/stacks-dotnet-rhel8@sha256:6ca14e5a94a98b15f39a353e533cf659b2b3937a86bd51af175dc3eadd8b80d5	stacks-dotnet-rhel8@sha256:6ca14e5a94a98b15f39a353e533cf659b2b3937a86bd51af175dc3eadd8b80d5
registry.redhat.io/codeready-workspaces/stacks-golang-rhel8@sha256:30e71577cb80ffaf1f67a292b4c96ab74108a2361347fc593cbb505784629db2	stacks-golang-rhel8@sha256:30e71577cb80ffaf1f67a292b4c96ab74108a2361347fc593cbb505784629db2
registry.redhat.io/codeready-workspaces/stacks-php-rhel8@sha256:bb7f7ef0ce58695aaf29b3355dd9ee187a94d1d382f68f329f9664ca01772ba2	stacks-php-rhel8@sha256:bb7f7ef0ce58695aaf29b3355dd9ee187a94d1d382f68f329f9664ca01772ba2
registry.redhat.io/codeready-workspaces/theia-endpoint-rhel8@sha256:abb4f4c8e1328ea9fc5ca4fe0c809ec007fe348e3d2ccd722e5ba75c02ff448f	theia-endpoint-rhel8@sha256:abb4f4c8e1328ea9fc5ca4fe0c809ec007fe348e3d2ccd722e5ba75c02ff448f
registry.redhat.io/codeready-workspaces/theia-rhel8@sha256:5ed38a48d18577120993cd3b673a365e31aeb4265c5b4a95dd9d0ac747260392	theia-rhel8@sha256:5ed38a48d18577120993cd3b673a365e31aeb4265c5b4a95dd9d0ac747260392

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:6704bd086f0d971ecedc1dd6dc7a90429231fdfa86579e742705b31cbedbd8b2	traefik-rhel8@sha256:6704bd086f0d971ecedc1dd6dc7a90429231fdfa86579e742705b31cbedbd8b2
registry.redhat.io/jboss-eap-7/eap-xp3-openj9-11-openshift-rhel8@sha256:53684e34b0dbe8560d2c330b0761b3eb17982edc1c947a74c36d29805bda6736	eap-xp3-openj9-11-openshift-rhel8@sha256:53684e34b0dbe8560d2c330b0761b3eb17982edc1c947a74c36d29805bda6736
registry.redhat.io/jboss-eap-7/eap-xp3-openjdk11-openshift-rhel8@sha256:3875b2ee2826a6d8134aa3b80ac0c8b5ebc4a7f718335d76dfc3461b79f93d19	eap-xp3-openjdk11-openshift-rhel8@sha256:3875b2ee2826a6d8134aa3b80ac0c8b5ebc4a7f718335d76dfc3461b79f93d19
registry.redhat.io/jboss-eap-7/eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d	eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:4ff9d6342dfd3b85234ea554b92867c649744ece9aa7f8751aae06bf9d2d324c	sso74-openj9-openshift-rhel8@sha256:4ff9d6342dfd3b85234ea554b92867c649744ece9aa7f8751aae06bf9d2d324c
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:b98f0b743dd406be726d8ba8c0437ed5228c7064015c1d48ef5f87eb365522bc	sso74-openshift-rhel8@sha256:b98f0b743dd406be726d8ba8c0437ed5228c7064015c1d48ef5f87eb365522bc
registry.redhat.io/rhel8/postgresql-96@sha256:ed53ca7b191432f7cf9da0fd8629d7de14ade609ca5f38aba443716f83616f2e	postgresql-96@sha256:ed53ca7b191432f7cf9da0fd8629d7de14ade609ca5f38aba443716f83616f2e
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73

<source-image>	<target-image>
registry.redhat.io/ubi8/ubi-minimal@sha256:31ccb79b1b2c2d6eff1bee0db23d5b8ab598eaf6238417d9813f1346f717c11	ubi8ubi-minimal@sha256:31ccb79b1b2c2d6eff1bee0db23d5b8ab598eaf6238417d9813f1346f717c11

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>
```

関連資料

- 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. 制限された環境での 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 ${OPENSHIFT_API_URL} --username ${OPENSHIFT_USERNAME} \  
--password ${OPENSHIFT_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:2.11 \  
--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. プロキシーの後ろでインストールするための CodeReady Workspaces カスタムリソースの準備

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

手順

- 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>'  
# [...]
```

- これらの基本設定のほかに、プロキシー設定では通常、プロキシーを使用せずに CodeReady Workspaces からアクセスされるホストの一覧に外部 OpenShift クラスター API URL のホストを追加する必要があります。

このクラスター API ホストを取得するには、OpenShift クラスターに対して以下のコマンドを実行します。

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

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

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

第4章 CODEREADY WORKSPACES の設定

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

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

次のセクションでは、特定のユーザーストーリーを説明します。

- 「[ワークスペーススターゲットプロジェクトの設定](#)」
- 「[ユーザーが作成できるワークスペースの数の設定](#)」
- 「[ユーザーが実行できるワークスペース数の設定](#)」
- 「[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 ルートの設定](#)」
- 「[ルーターのシャード化と連携するように OpenShift ルートを設定](#)」
- 「[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](#)」
- 「[シークレットまたは ConfigMap をファイルまたは環境変数として CodeReady Workspaces コンテナーにマウントする](#)」
- 「[Dev Workspace エンジンの有効化](#)」

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

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

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

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

詳細設定は以下を実行するために必要です。

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

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

例4.1 ワークスペースのデフォルトのメモリー制限の上書き

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

```
apiVersion: org.eclipse.che/v1
kind: CheCluster
# [...]
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 **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。

関連資料

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

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

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

4.1.2.1. CodeReady Workspaces サーバー

表4.1 CodeReady Workspaces サーバー

環境変数名	デフォルト値	詳細
CHE_DATABASE	<code> \${che.home}/storage</code>	CodeReady Workspaces が内部データオブジェクトを保存するフォルダー。
CHE_API	<code>http://\${CHE_HOST}:\${CHE_PORT}/api</code>	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。
CHE_API_INTERNAL	<code>http://\${CHE_HOST}:\${CHE_PORT}/api</code>	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。
CHE_WEBSOCKET_ENDPOINT	<code>ws://\${CHE_HOST}:\${CHE_PORT}/api/websocket</code>	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。
CHE_WORKSPACE_PROJECTS_STORAGE	<code>/projects</code>	プロジェクトは、CodeReady Workspaces サーバーから各ワークスペースを実行するマシンに同期されます。これは、プロジェクトが配置されているマシンのディレクトリーです。
CHE_WORKSPACE_PROJECTS_STORAGE_DEFAULT_SIZE	<code>1Gi</code>	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 che.infra.kubernetes.pvc.quantity プロパティーの値で書き換えられます)。

環境変数名	デフォルト値	詳細
<code>CHE_WORKSPACE_LOGS_ROOT_DIR</code>	<code>/workspace_logs</code>	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。
<code>CHE_WORKSPACE_HTTP_PROXY</code>		環境変数 <code>HTTP_PROXY</code> は、ワークスペースを起動するコンテナーで指定された値に設定します。
<code>CHE_WORKSPACE_HTTPS_PROXY</code>		環境変数 <code>HTTPS_PROXY</code> は、ワークスペースを起動するコンテナーで指定された値に設定します。
<code>CHE_WORKSPACE_NO_PROXY</code>		環境変数 <code>NO_PROXY</code> は、ワークスペースを起動するコンテナーで指定された値に設定します。
<code>CHE_WORKSPACE_AUTO_START</code>	<code>true</code>	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 <code>false</code> に設定します。
<code>CHE_WORKSPACE_POOL_TYPE</code>	固定:	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。設定可能な値は <code>fixed</code> および <code>cached</code> です。
<code>CHE_WORKSPACE_POOL_EXACT_SIZE</code>	30	プールタイプが <code>fixed</code> 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。設定されると、 <code>multiplier</code> プロパティーは無視されます。このプロパティーが設定されていない場合 (<code>0 < 0, NULL</code>)、プールサイズはコア数と等しくなります。 <code>che.workspace.pool.core_size_multiplier</code> も参照してください。

環境変数名	デフォルト値	詳細
CHE_WORKSPACE_POOL_CORES_MULTIPLIER	2	プールタイプが fixed に設定されており、 che.workspace.pool.executor_size が設定されている場合は、このプロパティーは無視されます。設定されている場合、プールサイズは N_CORES * multiplier になります。
CHE_WORKSPACE_PROBE_POOL_SIZE	10	このプロパティーは、ワークスペースサーバーの liveness プローブに使用するスレッドの数を指定します。
CHE_WORKSPACE_HTTP_PROXY_JAVA_OPTIONS	NULL	ワークスペース JVM の HTTP プロキシー設定。
CHE_WORKSPACE_JAVA_OPTIONS	<pre>-XX:MaxRAM=150m- XX:MaxRAMFraction=2 - XX:+UseParallelGC - XX:MinHeapFreeRatio=10 - XX:MaxHeapFreeRatio=20 - XX:GCTimeRatio=4 - XX:AdaptiveSizePolicyWeight=90 - Dsun.zip.disableMemoryMap ping=true -Xms20m - Djava.security.egd=file:/dev/ /urandom</pre>	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。
CHE_WORKSPACE_MAVEN_OPTIONS	<pre>-XX:MaxRAM=150m- XX:MaxRAMFraction=2 - XX:+UseParallelGC - XX:MinHeapFreeRatio=10 - XX:MaxHeapFreeRatio=20 - XX:GCTimeRatio=4 - XX:AdaptiveSizePolicyWeight=90 - Dsun.zip.disableMemoryMap ping=true -Xms20m - Djava.security.egd=file:/dev/ /urandom</pre>	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。
CHE_WORKSPACE_DEFAULT_MEMORY_LIMIT_MB	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。

環境変数名	デフォルト値	詳細
CHE_WORKSPACE_DEFAULT_MEMORY_REQUEST_MB	200	環境内に明示的な RAM 設定のない各コンテナーの RAM 要求。この量はワークスペースコンテナー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。現時点では、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。
CHE_WORKSPACE_DEFAULT_CPU_LIMIT_CORES	-1	環境に CPU 設定のない各コンテナーの CPU 制限。浮動小数点のコア数(例: 0.125)で、または OpenShift 形式(125m 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。
CHE_WORKSPACE_DEFAULT_CPU_REQUEST_CORES	-1	環境内に CPU 設定のない各コンテナーの CPU 要求。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。
CHE_WORKSPACE_SIDECA_R_DEFAULT_MEMORY_LIMIT_MB	128	CodeReady Workspaces プラグイン設定に RAM 設定のない各サイドカーの RAM 制限。0以下の値は、制限を無効にするものとして解釈されます。
CHE_WORKSPACE_SIDECA_R_DEFAULT_MEMORY_REQUEST_MB	64	CodeReady Workspaces プラグイン設定に RAM 設定のない各サイドカーの RAM 要求。

環境変数名	デフォルト値	詳細
<code>CHE_WORKSPACE_SIDECA R_DEFAULT_CPU_LIMIT_ _CORES</code>	-1	CodeReady Workspaces プラグイン設定に CPU 設定のない各サイドカーの CPU 制限のデフォルト。浮動小数点のコア数(例: 0.125)で、または OpenShift 形式 (125m 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。
<code>CHE_WORKSPACE_SIDECA R_DEFAULT_CPU_REQUEST_ _CORES</code>	-1	CodeReady Workspaces プラグイン設定に CPU 設定のない各サイドカーの CPU 要求のデフォルト。浮動小数点のコア数(例: 0.125)で、または OpenShift 形式 (125m などの整数のミリコア数)を使用して指定します。
<code>CHE_WORKSPACE_SIDECA R_IMAGE_PULL_POLICY</code>	Always	サイドカーのイメージプルストラテジーを定義します。使用できる値は Always 、 Never 、 IfNotPresent です。その他の値については、 Always は <code>:latest</code> タグが付いたイメージに、他の場合は IfNotPresent が想定されます。
<code>CHE_WORKSPACE_ACTIVIT Y_CHECK_SCHEDULER_ PERIOD_S</code>	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。
<code>CHE_WORKSPACE_ACTIVIT Y_CLEANUP_SCHEDULER_ PERIOD_S</code>	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の障害が発生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。
<code>CHE_WORKSPACE_ACTIVIT Y_CLEANUP_SCHEDULER_ INITIAL_DELAY_S</code>	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。

環境変数名	デフォルト値	詳細
<code>CHE_WORKSPACE_ACTIVITY_CHECK_SCHEDULER_DELAY_S</code>	180	CodeReady Workspaces サーバ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。
<code>CHE_WORKSPACE_CLEANUP_TEMPORARY_INITIAL_DELAY_MIN</code>	5	一時ワークスペースのクリーンアップジョブの最初の実行を遅延させる時間。
<code>CHE_WORKSPACE_CLEANUP_TEMPORARY_PERIOD_MIN</code>	180	実行を終了してから次の一時的ワークスペースのクリーンアップジョブの実行を開始するまでの間に遅延する時間
<code>CHE_WORKSPACE_SERVER_PING_SUCCESS_THRESHOLD</code>	1	サーバーへの正常に順次実行される ping の数。この数を超えると、サーバーは利用可能な状態にあるものとして処理されます。 CodeReady Workspaces Operator: このプロパティーは、ワークスペース、エージェント、ターミナル、exec などの全サーバーに共通します。
<code>CHE_WORKSPACE_SERVER_PING_INTERVAL_MILLISECONDS</code>	3000	ワークスペースサーバーへの連続する ping の間隔(ミリ秒単位)。
<code>CHE_WORKSPACE_SERVER_LIVENESS_PROBES</code>	<code>wsagent/http,exec-agent/http,terminal,theia,jupyter,dirigible,cloud-shell,intellij</code>	liveness プローブを必要とするサーバー名の一覧
<code>CHE_WORKSPACE_STARTUP_DEBUG_LOG_LIMIT_BYTES</code>	10485760	ワークスペースの起動をデバッグする際に che-server で観察される單一コンテナーから収集されるログの制限サイズ。デフォルト値は 10MB=10485760 です。
<code>CHE_WORKSPACE_STOP_ROLE_ENABLED</code>	true	true の場合、OpenShift OAuth が有効な場合に、編集権限を持つ「stop-workspace」ロールが「che」ServiceAccount に付与されます。この設定は、OpenShift OAuth が有効な場合にワークスペースのアーリングに主に必要になります。

環境変数名	デフォルト値	詳細
CHE_DEVWORKSPACES_ENABLED	false	DevWorkspaces を有効にして CodeReady Workspaces をデプロイするかどうかを指定します。このプロパティは、DevWorkspace のサポートもインストールされている場合に CodeReady Workspaces Operator によって設定されます。このプロパティを使用して、このファクトを CodeReady Workspaces ダッシュボードにアドバタイズします。このプロパティの値を手動で変更することは推奨されません。

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	詳細
CHE_AUTH_USER_SELF_CREATION	false	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クレデンシャルへの変更はありません。true の場合、API レベルでのユーザー作成を有効にします。
CHE_AUTH_ACCESS_DENIED_ERROR_PAGE	/error-oauth	認証エラーページアドレス
CHE_AUTH_RESERVED_USER_NAMES		予約済みのユーザー名

環境変数名	デフォルト値	詳細
CHE_OAUTH_GITHUB_CLIENTID	NULL	GitHub OAuth クライアントの設定。GitHub OAuth を設定して、リモー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。GitHub OAuth クライアント ID。
CHE_OAUTH_GITHUB_CLIENTSECRET	NULL	GitHub OAuth クライアントシークレット。
CHE_OAUTH_GITHUB_AUTHURI	https://github.com/login/oauth/authorize	GitHub OAuth 認証 URI。
CHE_OAUTH_GITHUB_TOKENURI	https://github.com/login/oauth/access_token	GitHub OAuth トークン URI。
CHE_OAUTH_GITHUB_REDIRECTURIS	http://localhost:\${CHE_PORT}/api/oauth/callback	GitHub OAuth リダイレクト URI。複数の値をコンマで区切ります（例：URI,URI,URI）。
CHE_OAUTH_OPENSOURCE_CLIENTID	NULL	OpenShift OAuth クライアントの設定。OpenShift OAuth トークンの取得に使用されます。OpenShift OAuth クライアント ID。
CHE_OAUTH_OPENSOURCE_CLIENTSECRET	NULL	OpenShift OAuth クライアントの設定。OpenShift OAuth トークンの取得に使用されます。OpenShift OAuth クライアント ID。OpenShift OAuth クライアントシークレット。
CHE_OAUTH_OPENSOURCE_OAUTH_ENDPOINT	NULL	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。OpenShift OAuth クライアント ID。OpenShift OAuth クライアントシークレット。OpenShift OAuth エンドポイント。

環境変数名	デフォルト値	詳細
CHE_OAUTH_OPENSIFT_V ERIFY_TOKEN_URL	NULL	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。OpenShift OAuth クライアント ID。OpenShift OAuth クライアントシークレット。OpenShift OAuth エンドポイント。OpenShift OAuth 検証トークン URL。
CHE_OAUTH1_BITBUCKET_ CONSUMERKEYPATH	NULL	Bitbucket Server OAuth1 クライアントの設定。パーソナルアクセストークンの取得に使用されます。Bitbucket Server アプリケーションのコンシューマーキーが含まれるファイルの場所（ユーザー名と同等）。
CHE_OAUTH1_BITBUCKET_ PRIVATEKEYPATH	NULL	Bitbucket Server OAuth1 クライアントの設定パーソナルアクセストークンの取得に使用されます。Bitbucket Server アプリケーションのコンシューマーキーが含まれるファイルの場所（ユーザー名と同等）。Bitbucket Server アプリケーションの秘密鍵が含まれるファイルの場所
CHE_OAUTH1_BITBUCKET_ ENDPOINT	NULL	Bitbucket Server OAuth1 クライアントの設定パーソナルアクセストークンの取得に使用されます。Bitbucket Server アプリケーションのコンシューマーキーが含まれるファイルの場所（ユーザー名と同等）。Bitbucket Server アプリケーションの秘密鍵の Bitbucket Server URL が含まれるファイルの場所ファクトリーと正しく連携するには、同じ URL を che.integration.bitbucket.ser ver_endpoints に含める必要があります。

4.1.2.3. 内部

表4.3 内部

環境変数名	デフォルト値	詳細
SCHEDULE_CORE_POOL_SIZE	10	CodeReady Workspaces 拡張には、時間ベースでスケジュールされる実行をスケジュールできます。これにより、繰り返されるスケジュールで起動する拡張に割り当てるスレッドプールのサイズが設定されます。
DB_SCHEMA_FLYWAY_BASELINE_ENABLED	true	DB の初期化および移行設定。true の場合には、baseline.version で設定したバージョンのスクリプトを無視します。
DB_SCHEMA_FLYWAY_BASELINE_VERSION	5.0.0.8.1	これまでのバージョンを含むスクリプトは無視されます。ベースラインバージョンと同じバージョンのスクリプトも無視されることに注意してください。
DB_SCHEMA_FLYWAY_SCRIPTS_PREFIX		移行スクリプトの接頭辞
DB_SCHEMA_FLYWAY_SCRIPTS_SUFFIX	.sql	移行スクリプトの接尾辞。
DB_SCHEMA_FLYWAY_SCRIPTS_VERSION_SEPARATOR	—	スクリプト名を他の部分からバージョンを区切るための区切り文字。
DB_SCHEMA_FLYWAY_SCRIPTS_LOCATIONS	classpath:che-schema	移行スクリプトを検索する場所。

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	詳細
CHE_INFRA_KUBERNETES_MASTER_URL		インフラが使用する OpenShift クライアントのマスター URL の設定。
CHE_INFRA_KUBERNETES_TRUST_CERTS	false	信頼済み証明書を使用するように OpenShift クライアントを設定する布尔値。

環境変数名	デフォルト値	詳細
<code>CHE_INFRA_KUBERNETES_CLUSTER_DOMAIN</code>	<code>NULL</code>	OpenShift クラスター ドメイン。設定されていない場合は、svc 名にはクラスター ドメインに関する情報が含まれません。
<code>CHE_INFRA_KUBERNETES_SERVER_STRATEGY</code>	<code>multi-host</code>	サーバーが Kubernetes 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
<code>CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_EXPOSURE</code>	<code>native</code>	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 ルート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 :: reverse-proxy ゲートウェイを使用してサーバーを公開します。
<code>CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE_ENDPOINT_EXPOSURE</code>	<code>multi-host</code>	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ます。 single-host :: サブパスに公開されます。
<code>CHE_INFRA_KUBERNETES_SINGLEHOST_GATEWAY_CONFIGMAP_LABELS</code>	<code>app=che,component=che-gateway-config</code>	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。
<code>CHE_INFRA_KUBERNETES_INGRESS_DOMAIN</code>		<code>che.infra.kubernetes.server_strategy</code> プロパティーが <code>multi-host</code> 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	詳細
<code>CHE_INFRA_KUBERNETES_NAMESPACE_CREATION_ALLOWED</code>	<code>true</code>	CodeReady Workspaces サーバーがユーザー ワークスペースの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。
<code>CHE_INFRA_KUBERNETES_NAMESPACE_DEFAULT</code>	<code><username>-che</code>	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。 <code><username></code> および <code><userid></code> プースホルダー (例: <code>che-workspace-<username></code>) 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。OpenShift インフラでプロジェクトの指定にも使用されます。 <code><username></code> または <code><userid></code> プースホルダーは必須です。
<code>CHE_INFRA_KUBERNETES_NAMESPACE_LABEL</code>	<code>true</code>	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。
<code>CHE_INFRA_KUBERNETES_NAMESPACE_LABELS</code>	<code>app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspaces-namespaces</code>	CodeReady Workspaces に使用されるプロジェクトの検索に使用するラベルの一覧。これらは <code>che.infra.kubernetes.namespace.annotations</code> と組み合わせてユーザー用に準備されたプロジェクトを検索し、ワークスペースでプロジェクトをアクティブにラベル付けするのに使用されます。

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_NAMESPACE_ANNOTATIONS	<code>che.eclipse.org/username=<username></code>	CodeReady Workspaces ユーザー ワークスペース用に用意されたプロジェクトの検索に使用するアノテーションの一覧。これらのアノテーションと照合されるのは、 che.infra.kubernetes.namespace.labels と一致するプロジェクトのみです。このプロジェクトは、 che.infra.kubernetes.namespace.labels と che.infra.kubernetes.namespace.annotations 両方に一致するプロジェクトは、優先的にユーザーのワークスペースに使用されます。<username> プレースホルダーを使用して、具体的なユーザーにプロジェクトを指定できます。
CHE_INFRA_KUBERNETES_SERVICE_ACCOUNT_NAME	NULL	すべてのワークスペース Pod にバインドされるように指定する必要のある Kubernetes サービスアカウント名を定義します。OpenShift インフラストラクチャーがサービスアカウントを作成しない CodeReady Workspaces Operator は存在すべきです。OpenShift インフラストラクチャーは、プロジェクトが事前に定義されているかどうかをチェックします (che.infra.openshift.project が空でない場合)。これが事前に定義されている場合はサービスアカウントが存在するはずです。これが 'NULL' または空の文字列の場合、インフラストラクチャーはワークスペースごとに新しい OpenShift プロジェクトを作成し、必要なロールを持つワークスペースのサービスアカウントをここに準備します。

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_WORKSPACE_SA_CLUSTER_ROLES	NULL	ワークスペースサービスアカウントで使用するオプションの追加の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ます、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ります。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。
CHE_INFRA_KUBERNETES_WORKSPACE_START_TIMEOUT_MIN	8	Kubernetes ワークスペースの開始時間を制限する待機時間を定義します。
CHE_INFRA_KUBERNETES_INGRESS_START_TIMEOUT_MIN	5	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。
CHE_INFRA_KUBERNETES_WORKSPACE_UNRECOVERABLE_EVENTS	FailedMount,FailedScheduling,MountVolume.SetUpfailed, Failed to pull image,FailedCreate,ReplicaSetCreateError	ワークスペースの起動中に、プロパティで定義されたリカバリー不可能なイベントが発生する場合は、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終了します。CodeReady Workspaces Operator: リカバリー不能なイベントが発生する可能性があるため「Failed」の理由だけを追加できません。失敗したコンテナーの起動は、CodeReady Workspaces サーバーで明示的に処理されます。
CHE_INFRA_KUBERNETES_PVC_ENABLED	true	CodeReady Workspaces 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を定義します (例: プロジェクトやログのバックアップや、無効化)。

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_PVC_STRATEGY	common	<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:: 同じプロジェクトに含まれる全ワークスペースが同じ PVC を再利用します。PVC の名前は <code>{che.infra.kubernetes.pvc.name}</code> 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 <code>{che.infra.kubernetes.pvc.name} + '-' + {generated_8_chars}</code> 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:: 各ワークスペースに別個の PVC が使用されます。PVC の名前は <code>{che.infra.kubernetes.pvc.name} + '-' + {WORKSPACE_ID}</code> 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	詳細
<code>CHE_INFRA_KUBERNETES_PVC_PRECREATE_SUBPATHS</code>	<code>true</code>	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 <code>true</code> ですが、OpenShift のバージョンがユーザーパーミッションでサブディレクトリーを作成する場合は <code>false</code> に設定する必要があります。 subPath in volumeMount is not writable for non-root users #41638 the CodeReady Workspaces Operator that this property that this property that only effect for the common PVC strategy used を参照してください。
<code>CHE_INFRA_KUBERNETES_PVC_NAME</code>	<code>claim-che-workspace</code>	CodeReady Workspaces 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 プロパティに関するドキュメントを参照してください。
<code>CHE_INFRA_KUBERNETES_PVC_STORAGE_CLASS_NAME</code>		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。
<code>CHE_INFRA_KUBERNETES_PVC_QUANTITY</code>	<code>10Gi</code>	CodeReady Workspaces ワークスペースの Persistent Volume Claim (永続ボリューム要求、PVC) のサイズを定義します。 「永続ストレージについて」 を参照してください。

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE	registry.access.redhat.com/ubi8-minimal:8.3-230	OpenShift で永続ボリューム要求 (PVC) のメンテナンスジョブを実行する際に起動する Pod
CHE_INFRA_KUBERNETES_PVC_JOBS_IMAGE_PULL_POLICY	IfNotPresent	OpenShift クラスターのメンテナンスジョブに使用されるコンテナーのイメージプルポリシー
CHE_INFRA_KUBERNETES_PVC_JOBS_MEMORYLIMIT	250Mi	永続ボリューム要求のメンテナンスジョブの Pod メモリー制限を定義します。
CHE_INFRA_KUBERNETES_PVC_ACCESS_MODE	ReadWriteOnce	Persistent Volume Claim (永続ボリューム要求、PVC) のアクセスモードを定義します。アクセスモードを変更する一般的な PVC ストラテジー向けの CodeReady Workspaces Operator は、同時に実行されるワークスペースの数に影響します。CodeReady Workspace を実行している OpenShift インスタンスが RWX アクセスマードで永続ボリュームを使用している場合には、同時に実行するワークスペース数の上限は、CodeReady Workspace の制限設定 (メモリー、CPU など) でのみバインドされます。「 永続ストレージについて 」を参照してください。
CHE_INFRA_KUBERNETES_PVC_WAIT_BOUND	true	CodeReady Workspaces Server がワークスペース Persistent Volume Claim (永続ボリューム要求、PVC) の作成後にバインドする必要があるかどうかを定義します。デフォルト値は true です。このパラメーターは、全 Persistent Volume Claim (永続ボリューム要求、PVC) ストラテジーにより使用されます。 volumeBindingMode が WaitForFirstConsumer に設定されている場合は、 false に設定する必要があります。それ以外の場合は、ワークスペースの起動が PVC の待機フェーズでハングします。
	NULL	サーバーを公開するために使用される Ingress のアノテーションを

CHE_INFRA_KUBERNETES_I 環境変数名 WORKSC_ANNOTATIONS_J	デフォルト値	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 ではなくルートを使用するため、このプロパティは無視されます。单一ホストデプロイメントストラテジーが機能する CodeReady Workspaces Operator 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（そのため、サーバーはアプリケーションルートの変更をサポートする必要があります）。 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 以降では、 <code>{'ingress.kubernetes.io/rewrite-target': '/\$1', 'ingress.kubernetes.io/ssl-redirect': 'false', 'ingress.kubernetes.io/proxy-connect-timeout': '3600', 'ingress.kubernetes.io/proxy-read-timeout': '3600', 'nginx.org/websocket-services': '<service-name>'}</code> の値が推奨されます。 che.infra.kubernetes.ingress.path.rewrite_transform は <code>%s(.*)</code> に設定する必要があります。0.22.0 よりも古い nginx Ingress コントローラーの場合には、rewrite-target は / に設定するだけで、パスは %s に変換されます（ che.infra.kubernetes.ingress.path.rewrite_transform プロパティを参照）。Ingress コントローラーが Ingress パスにある正規表現を使用する方法と、URL の書き換えを実行する方法についての説明は、nginx Ingress コントローラーのドキュメントを参照してください。
SON		

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_INGRESS_PATH_TRANSFORM	NULL	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 <code>String.format()</code> メソッドへの有効な入力であり、%s への参照が1つだけ含まれる必要があります。Ingress のアノテーションとパスを指定する際にこれら2つのプロパティの相互作用を確認するには、 <code>che.infra.kubernetes.ingress.annotations_json</code> 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。
CHE_INFRA_KUBERNETES_INGRESS_LABELS	NULL	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。
CHE_INFRA_KUBERNETES_POD_SECURITY_CONTEXT_RUN_AS_USER	NULL	OpenShift インフラによって作成される Pod のセキュリティーコンテキストを定義します。これは OpenShift インフラによって無視されます。
CHE_INFRA_KUBERNETES_POD_SECURITY_CONTEXT_FS_GROUP	NULL	OpenShift インフラによって作成される Pod のセキュリティーコンテキストを定義します。Pod の全コンテナーに適用される特別な補助グループです。OpenShift インフラは、このグループを無視します。

環境変数名	デフォルト値	詳細
<code>CHE_INFRA_KUBERNETES_POD_TERMINATION_GRACE_PERIOD_SEC</code>	0	OpenShift インフラストラクチャーによって作成される Pod の猶予期間を定義します。デフォルト値: 0 これにより、Pod をすぐに停止し、ワークスペースの停止に必要な時間を短縮できます。 CodeReady Workspaces Operator: terminationGracePeriodSeconds が OpenShift レシピで明示的に設定されている場合は上書きされません。
<code>CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC_REQUESTS_MAX</code>	1000	KubernetesClient 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並行非同期 Web 要求 (HTTP 要求または継続的な WebSocket 呼び出し) の最大数。デフォルト値 : max=64 および max_per_host:5 CodeReady Workspaces はコマンドや ws-agent ログなど、コネクションを開いたままにするため、デフォルトの値はマルチユーザーシナリオには適していません。
<code>CHE_INFRA_KUBERNETES_CLIENT_HTTP_ASYNC_REQUESTS_MAX_PER_HOST</code>	1000	ホストごとの並行非同期 Web 要求の最大数。
<code>CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION_POOL_MAX_IDLE</code>	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
<code>CHE_INFRA_KUBERNETES_CLIENT_HTTP_CONNECTION_POOL_KEEP_ALIVE_MIN</code>	5	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)
<code>CHE_INFRA_KUBERNETES_TLS_ENABLED</code>	false	Transport Layer Security(TLS)を有効にして Ingress を作成します。OpenShift インフラストラクチャーではルートは TLS に対応します。

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_TLS_SECRET		TLS でワークスペース Ingress を作成する際に使用すべきシークレットの名前。OpenShift インフラストラクチャーでは、このプロパティは無視されます。
CHE_INFRA_KUBERNETES_TLS_KEY	NULL	ワークスペース Ingress に使用する必要のある TLS Secret のデータ。 cert および key は Base64 アルゴリズムでエンコードする必要があります。OpenShift インフラストラクチャーでは、これらのプロパティは無視されます。
CHE_INFRA_KUBERNETES_TLS_CERT	NULL	ワークスペース Ingress に使用する必要のある TLS Secret の証明書データ。証明書は、Base64 アルゴリズムでエンコードする必要があります。OpenShift インフラストラクチャーでは、このプロパティは無視されます。
CHE_INFRA_KUBERNETES_RUNTIMES_CONSISTENCY_CHECK_PERIOD_MIN	-1	ランタイムの整合性チェックが実行される期間を定義します。ランタイムに一貫性のない状態がある場合、ランタイムは自動的に停止します。値は 0 をより大きな値、または -1 である必要があります。ここで、-1 はチェックが実行されることを意味します。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、CodeReady Workspace Server の配置先と同じ namespace でワークスペースオブジェクトが作成される場合、 cluster-admin サービスアカウントトークンが CodeReady Workspace Server Pod にマウントされる場合に機能します。これは、CodeReady Workspaces Server が OAuth プロバイダーからトークンを使用して Kubernetes API と通信する場合には機能しません。

環境変数名	デフォルト値	詳細
<code>CHE_INFRA_KUBERNETES_TRUSTED_CA_SRC_CONFIGMAP</code>	NULL	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。プロパティーを OpenShift 4 インフラストラクチャーに設定し、 <code>che.infra.openshift.trusted_ca.dest_configmap_labels</code> に <code>config.openshift.io/inject-trusted-cabundle=true</code> ラベルが含まれる場合に、クラスター CA バンドルも伝播されます。
<code>CHE_INFRA_KUBERNETES_TRUSTED_CA_DEST_CONFIGMAP</code>	<code>ca-certs</code>	追加の CA TLS 証明書を含むワークスペース namespace の設定マップの名前。ワークスペース namespace にある <code>che.infra.kubernetes.trusted_ca.src_configmap</code> のコピーを保持します。この設定マップの内容は、プラグインブローカーを含むすべてのワークスペースコンテナーにマウントされます。既存の設定マップと競合しない限り、設定マップ名は変更しないでください。CodeReady Workspaces Operator: 結果として作成される設定マップ名を最終的に調整してプロジェクトで一意にすることができます。元の名前は <code>che.original_name</code> ラベルに保存されます。
<code>CHE_INFRA_KUBERNETES_TRUSTED_CA_MOUNT_PATH</code>	<code>/public-certs</code>	CA バンドルがマウントされるワークスペースコンテナーでパスを設定します。 <code>che.infra.kubernetes.trusted_ca.dest_configmap</code> で指定される設定マップの内容がマウントされます。
<code>CHE_INFRA_KUBERNETES_TRUSTED_CA_DEST_CONFIGMAP_LABELS</code>		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。 <code>che.infra.kubernetes.trusted_ca.dest_configmap</code> プロパティーを参照してください。

環境変数名	デフォルト値	詳細
CHE_INFRA_KUBERNETES_ENABLE_UNsupported_K8S	false	/unsupported/OpenShift エンドポイントを有効にして、Kubernetes インフラストラクチャーの呼び出しを解決します。基礎となるインフラストラクチャーの REST API に直接アクセスできます。この設定では、大幅に権限が昇格されます。これは Kubernetes インフラストラクチャーにだけ影響します。そのため、OAuth が使用されている OpenShift では、セキュリティリスクがないことを意味します。リスクを把握していない場合は有效地にしないでください。

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	詳細
CHE_INFRA_OPENSIFT_TRUSTED_CA_DEST_CONFIGMAP_LABELS	config.openshift.io/inject-trusted-cabundle=true	ユーザーアクセススペースの 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 プロパティー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。
CHE_INFRA_OPENSIFT_ROUTE_LABELS	NULL	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。
CHE_INFRA_OPENSIFT_ROUTE_HOST_DOMAIN_SUFFIX	NULL	ワークスペースルートの接尾辞として使用する必要のあるホスト名。たとえば、domain_suffix=<codeready-openshift_deployment_name>.<domain_name> を使用する場合には、ルートは routed3qrtk.<codeready-openshift_deployment_name>.<domain_name> のようになります。有効な DNS 名である必要があります。

環境変数名	デフォルト値	詳細
CHE_INFRA_OPENSOURCE_PROJECT_INIT_WITH_SERVER_SA	true	OpenShift OAuth が有効な場合には、CodeReady Workspaces サーバーのサービスアカウントで OpenShift プロジェクトを初期化します。

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	詳細
CHE_WORKSPACE_PLUGIN_BROKER_METADATA_IMAGE	quay.io/eclipse/che-plugin-metadata-broker:v3.4.0	ワークスペースツール設定を解決し、プラグインの依存関係をワークスペースにコピーする CodeReady Workspaces プラグインブローカーアプリケーションの Docker イメージCodeReady Workspaces Operator はデフォルトでこれらのイメージを上書きします。CodeReady Workspaces が Operator を使用してインストールされている場合は、ここでイメージを変更しても、効果がありません。
CHE_WORKSPACE_PLUGIN_BROKER_ARTIFACTS_IMAGE	quay.io/eclipse/che-plugin-artifacts-broker:v3.4.0	CodeReady Workspaces プラグインアーティファクトブローカーの Docker イメージ。このブローカーは、ワークスペース Pod で init コンテナーとして実行されます。このジョブは、プラグインの ID (レジストリー内のプラグインへの参照または、プラグインの meta.yaml へのリンク) の一覧を取り、ワークスペース向けに要求されたプラグインごとに、正しい .vsix and .theia 拡張子が /plugins ディレクトリーにダウンロードされていることを確認します。

環境変数名	デフォルト値	詳細
CHE_WORKSPACE_PLUGIN__BROKER_DEFAULT_MERGE_PLUGINS	false	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プ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、プラグインプ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ア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で mergePlugins 属性が指定されていない場合に使用されるデフォルト設定です。
CHE_WORKSPACE_PLUGIN__BROKER_PULL_POLICY	Always	ワークスペースツール設定を解決し、プラグインの依存関係をワークスペースにコピーするCodeReady Workspaces プラグインプローカーアプリケーションのDockerイメージ
CHE_WORKSPACE_PLUGIN__BROKER_WAIT_TIMEOUT_MIN	3	プラグインプ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。
CHE_WORKSPACE_PLUGIN__REGISTRY_URL	https://che-plugin-registry.prod-preview.openshift.io/v3	ワークスペース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 プラグインレジストリーが不要な場合、値'NULL'を使用する必要があります。
CHE_WORKSPACE_PLUGIN__REGISTRY_INTERNAL_URL	NULL	ワークスペースプラグインレジストリーの内部エンドポイント。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 プラグインレジストリーが不要な場合、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	詳細
<code>CHE_WORKSPACE_DEVFILE__REGISTRY_URL</code>	<code>https://che-devfile-registry.prod-preview.openshift.io/</code>	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: <code>http://che-devfile-registry-eclipse-che.192.168.65.2.nip.io</code> CodeReady Workspaces プラグインレジストリーが不要な場合、値 'NULL' を使用する必要があります。
<code>CHE_WORKSPACE_DEVFILE__REGISTRY_INTERNAL_URL</code>	<code>NULL</code>	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: <code>http://plugin-registry.che.svc.cluster.local:8080</code> CodeReady Workspaces プラグインレジストリーが不要な場合、値 'NULL' を使用する必要があります。
<code>CHE_WORKSPACE_STORAGE_AVAILABLE_TYPES</code>	<code>persistent,ephemeral,async</code>	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。 <code>che.infra.kubernetes.pvc.strategy='common'</code> - <code>che.limits.user.workspaces.run.count=1</code> - <code>che.infra.kubernetes.namespace.default</code> に <code><username></code> が含まれる場合にのみ機能します。それ他の場合は、一覧から async を削除します。

環境変数名	デフォルト値	詳細
CHE_WORKSPACE_STORAGE_PREFERRED_TYPE	永続	ダッシュボードなどのクライアントがワークスペースの作成/更新時にユーザーに提案するストレージタイプのデフォルト値を定義する設定プロパティー。 async 値は実験的な機能であるため、デフォルトタイプとしての使用は推奨されません。
CHE_SERVER_SECURE_EXPOSER	jwtproxy	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要求のみを受信します。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_TOKEN_ISSUER	wsmaster	署名のない要求をルーティングするための Jwtproxy 発行側の文字列、トークンの有効期間およびオプションの認証ページのパス。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_TOKEN_TTL	8800h	jwtproxy 発行者トークンの有効期間。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_AUTH_LOADER_PATH	/app/loader.html	署名なしの要求をルーティングする認証ページのパス(任意)。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_IMAGE	quay.io/eclipse/che-jwtproxy:0.10.0	jwtproxy イメージ。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_MEMORY_REQUEST	15mb	jwtproxy メモリー要求。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_MEMORY_LIMIT	128mb	jwtproxy メモリー制限。
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_CPU_REQUEST	0.03	jwtproxy CPU 要求。

環境変数名	デフォルト値	詳細
CHE_SERVER_SECURE_EXPOSER_JWTPROXY_CPU_LIMIT	0.5	jwtproxy CPU 制限。

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	詳細
CHE_CORE_JSONRPC_PROCESSOR_MAX_POOL_SIZE	50	JSON RPC 処理プールの最大サイズ。プールサイズが超過すると、メッセージの実行が拒否されます。
CHE_CORE_JSONRPC_PROCESSOR_CORE_POOL_SIZE	5	初期 JSON 処理プール。主な JSON RPC メッセージを処理するに使用されるスレッドの最小数。
CHE_CORE_JSONRPC_PROCESSOR_QUEUE_CAPACITY	100000	Json RPC メッセージの処理に使用するキューの設定。
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	詳細
CHE_CORS_ALLOWED_ORIGINS	*	許可される要求元を指定します。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' を使用してオンにします。
CHE_CORS_ALLOW_CREDENTIALS	false	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	詳細
CHE_FACTORY_DEFAULT_PLUGINS	redhat/vscode-commons/latest	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: <code>pluginFooPublisher/pluginFooName/pluginFooVersion,pluginBarPublisher/pluginBarName/pluginBarVersion</code>
CHE_FACTORY_DEFAULT_DEVFILE_Filenames	devfile.yaml,.devfile.yaml	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、プロパティーで列挙される順序でこれらのファイルの特定を試みます。

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	詳細
CHE_FACTORY_DEFAULT_EDITOR	eclipse/che-theia/latest	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 に使用されるエディター。
CHE_FACTORY_SCM_FILE_FETCHER_LIMIT_BYTES	102400	SCM リポジトリからファイルを取得する URL フェッチャーのファイルサイズ制限。
CHE_FACTORY_DEVFILE2_FILES_RESOLUTION_LIST	.che/che-editor.yaml,.che/che-theia-plugins.yaml,.vscode/extensions.json	devfile v2 を補完する追加ファイルで、リポジトリに含まれる場合があり、取得するには Factory の SCM リゾルバーサービスへのリンクとして参照する必要があります。

環境変数名	デフォルト値	詳細
<code>CHE_WORKSPACE_DEVFILE_DEFAULT_EDITOR</code>	<code>eclipse/che-theia/latest</code>	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、 <code>editorPublisher/editorName/editorVersion</code> 値になります。 <code>NULL</code> 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。
<code>CHE_WORKSPACE_DEVFILE_DEFAULT_EDITOR_PLUGINS</code>	<code>NULL</code>	デフォルトのエディター用にプロビジョニングする必要があるデフォルトのプラグイン。ユーザー定義の devfile で明示的に参照されていないこの一覧のすべてのプラグインはプロビジョニングされますが、これはデフォルトのエディターが使用されているか、またはユーザー定義のエディターが(異なるバージョンの場合でも)デフォルトと同じである場合に限ります。形式は、コンマ区切りの <code>pluginPublisher/pluginName/pluginVersion</code> 値および URL です。例: <code>eclipse/che-theia-exec-plugin/0.0.1,eclipse/che-theia-terminal-plugin/0.0.1,https://cdn.pluginregistry.com/vimode/meta.yaml</code> プラグインが URL の場合、プラグインの <code>meta.yaml</code> はその URL から取得されます。
<code>CHE_WORKSPACE_PROVISION_SECRET_LABELS</code>	<code>app.kubernetes.io/part-of=che.eclipse.org,app.kubernetes.io/component=workspace-secret</code>	ユーザー namespace からシークレットを選択するためにラベルのコンマ区切りの一覧を定義します。これは、ファイルまたは環境変数としてワークスペースコンテナーにマウントされます。すべての指定されるラベルに一致するシークレットのみが選択されます。
<code>CHE_WORKSPACE_DEVFILE_ASYNC_STORAGE_PLUGIN</code>	<code>eclipse/che-async-pv-plugin/latest</code>	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	詳細
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_IMAGE	quay.io/eclipse/che-workspace-data-sync-storage:0.0.1	CodeReady Workspaces 非同期ストレージの Docker イメージ
CHE_WORKSPACE_POD_NODE_SELECTOR	NULL	オプションでワークスペース Pod のノードセレクターを設定します。形式は、コンマ区切りの key=value ペアです (例: disktype=ssd,cpu=xlarge,foo=bar)。
CHE_WORKSPACE_POD_TOLERATIONS_JSON	NULL	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、テイントの容認の JSON 配列を表す文字列か、または NULL 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、 toleration v1 コア仕様 に準拠する必要があります。例: [{'effect': 'NoExecute', 'key': 'a NodeTaint', 'operator': 'Equal', 'value': 'aValue'}]
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN_TIMEOUT_MIN	120	最後に使用されたワークスペースの停止後の非同期ストレージ Pod のシャットダウンのタイムアウト。0 以下の値は、シャットダウン機能を無効にするものとして解釈されます。
CHE_INFRA_KUBERNETES_ASYNC_STORAGE_SHUTDOWN_CHECK_PERIOD_MIN	30	非同期ストレージ Pod が機能を停止する期間を定義します (デフォルトでは 30 分ごと)。
CHE_INTEGRATION_BITBUCKET_SERVER_ENDPOINTS	NULL	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。
CHE_INTEGRATION_GITLAB_SERVER_ENDPOINTS	NULL	Factory の統合に使用される GitLab エンドポイント。GitLab 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。
CHE_INTEGRATION_GITLAB_OAUTH_ENDPOINT	NULL#	OAuth 2 統合が設定された GitLab サーバーのアドレス

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	詳細
CHE_SYSTEM_SUPER_PRIVILEGED_MODE	false	System Super Privileged Mode (システムのスーパー特権モード)。getByKey、getByNameSpace、stopWorkspaces、およびgetResources の manageSystem パーミッションの追加パーミッションをユーザーに付与します。これらは、デフォルトでは管理者には提供されず、これらのパーミッションにより、管理者は admin 権限でそれらのワークスペースに名前を指定し、ワークスペースへの可視性を得ることができます。
CHE_SYSTEM_ADMIN_NAME	admin	che.admin.name ユーザーのシステムパーミッションを付与します。ユーザーがすでに存在する場合は、これはコンポーネントの起動時に生じます。ユーザーがすでに存在しない場合は、ユーザーがデータベースで永続化される初回のログイン時に発生します。

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	詳細
CHE_LIMITS_WORKSPACE_ENV_RAM	16gb	ワークスペースは、開発を行う際のユーザー向けの基本的なランタイムです。ワークスペースの作成方法や、消費されるリソースを制限するパラメーターを設定できます。ユーザーが新規ワークスペースの作成時にワークスペースに割り当てることができる RAM の最大量。RAM スライダーは、この最大値に合わせて調整されます。

環境変数名	デフォルト値	詳細
CHE_LIMITS_WORKSPACE_IDLE_TIMEOUT	1800000	システムがワークスペースを一時停止した後にこれを停止する際に、ユーザーがワークスペースでアイドル状態になる期間(ミリ秒単位)。アイドル状態は、ユーザーがワークスペースと対話しない期間です。つまり、エージェントのいずれも対話を受け取っていない期間を意味します。ブラウザーウィンドウを開いたままになるとアイドル状態になります。
CHE_LIMITS_WORKSPACE_RUN_TIMEOUT	0	システムが一時停止するまでの、アクティビティーを問わず、ワークスペースが実行される期間(ミリ秒単位)。一定期間後にワークスペースを自動的に停止する場合は、このプロパティーを設定します。デフォルトはゼロで、実行タイムアウトがないことを意味します。

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	詳細
CHE_LIMITS_USER_WORKSPACES_RAM	-1	単一ユーザーがワークスペースの実行に割り当てることができるRAMの合計量。ユーザーは、このRAMを单一のワークスペースに割り当てるか、または複数のワークスペースに分散することができます。
CHE_LIMITS_USER_WORKSPACES_COUNT	-1	ユーザーが作成できるワークスペースの最大数。追加のワークスペースを作成しようとすると、ユーザーにはエラーメッセージが表示されます。これは、実行中および停止中のワークスペースの合計数に適用されます。

環境変数名	デフォルト値	詳細
CHE_LIMITS_USER_WORKSPACES_RUN_COUNT	1	単一ユーザーが持てる実行中のワークスペースの最大数。ユーザーがこのしきい値に達し、追加のワークスペースを開始しようとすると、エラーメッセージと共にプロンプトが表示されます。ユーザーは、実行中のワークスペースを停止してから別のワークスペースをアクティベートする必要があります。

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	詳細
CHE_LIMITS_ORGANIZATION_WORKSPACES_RAM	-1	単一組織(チーム)がワークスペースの実行に割り当てることができるRAMの合計量。組織の所有者はこのRAMを割り当てることができますが、チームのワークスペース全体で適切に割り当てられているように見えます。
CHE_LIMITS_ORGANIZATION_WORKSPACES_COUNT	-1	組織が所有できるワークスペースの最大数。追加のワークスペースを作成しようとすると、組織にはエラーメッセージが表示されます。これは、実行中および停止中のワークスペースの合計数に適用されます。
CHE_LIMITS_ORGANIZATION_WORKSPACES_RUN_COUNT	-1	単一組織が持てる実行中のワークスペースの最大数。組織がこのしきい値に達し、追加のワークスペースを開始しようとすると、エラーメッセージと共にプロンプトが表示されます。組織は、実行中のワークスペースを停止してから別のワークスペースをアクティベートする必要があります。

4.1.2.15. 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4.15 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	詳細
CHE_INFRA_OPENSHIFT_OAUTH_IDENTITY_PROVIDER	NULL	Keycloak に登録されている OpenShift アイデンティティープロバイダーのエイリアス。これは、現行の CodeReady Workspaces ユーザーが所有する OpenShift namespace にワークスペース OpenShift リソースを作成するために使用されます。 che.infra.openshift.provider が空白以外の値に設定する場合は NULL に設定する必要があります。 OpenShift アイデンティティープロバイダー を参照してください。

4.1.2.16. Keycloak の設定

表4.16 Keycloak の設定

環境変数名	デフォルト値	詳細
CHE_KEYCLOAK_AUTH_SERVER_URL	<code>http://\${CHE_HOST}:5050/auth</code>	che.keycloak_oidcProvider を使用している場合のみ、keycloak アイデンティティープロバイダーサーバーの URL を NULL に設定できます。
CHE_KEYCLOAK_AUTH_INTERNAL_SERVER_URL	NULL	keycloak アイデンティティー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
CHE_KEYCLOAK_REALM	che	Keycloak レルムを使用してユーザーを認証するために使用されます。 che.keycloak_oidcProvider が使用している場合のみ NULL に設定できます。
CHE_KEYCLOAK_CLIENT_ID	che-public	ダッシュボード、IDE、および CLI でユーザーを認証する che.keycloak.realm の Keycloak クライアント識別子。
CHE_KEYCLOAK_OSO_ENDPOINT	NULL	OSO OAuth トークンにアクセスするための URL
CHE_KEYCLOAK_GITHUB_ENDPOINT	NULL	Github OAuth トークンにアクセスするための URL

環境変数名	デフォルト値	詳細
CHE_KEYCLOAK_ALLOWED_CLOCK_SKEW_SEC	3	exp または nbf 要求を検証する際にクロックスキューについて許容される秒数。
CHE_KEYCLOAK_USE_NO_NCE	true	OIDC オプションの nonce 機能を使用して、セキュリティーを強化します。
CHE_KEYCLOAK_JS_ADAPTER_URL	NULL	使用する Keycloak Javascript アダプターの URL。NULL に設定すると、デフォルト値が \${che.keycloak.auth_server_url}/js/keycloak.js になり、別の oidc_provider を使用する場合には、 <che-server>/api/keycloak/OIDCKeycloak.js になります。
CHE_KEYCLOAK_OIDC_PROVIDER	NULL	この仕様 (Obtaining OpenID Provider Configuration Information) 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。
CHE_KEYCLOAK_USE_FIXED_REDIRECT_URLS	false	固定されたリダイレクト URL のみをサポートする別の OIDC プロバイダーを使用する場合は true に設定します。このプロパティは、 che.keycloak.oidc_provider が NULL の場合は無視されます。
CHE_KEYCLOAK_USERNAME CLAIM	NULL	定義されていない場合、JWT トークンの解析時にユーザー名の要求がユーザーの表示名として使用されます。フォールバック値は「preferred_username」です。

環境変数名	デフォルト値	詳細
CHE_OAUTH_SERVICE_MODE	delegated	「embedded」モードまたは「delegated」モードで使用できる OAuth 認証サービスの設定。 「embedded」に設定すると、サービスは、(Single User モードの場合のように) CodeReady Workspaces の OAuthAuthenticator のラッパーとして機能します。 「delegated」に設定すると、サービスは Keycloak IdentityProvider メカニズムを使用します。このプロパティが正しく設定されていない場合は、ランタイム例外 wii がスローされます。
CHE_KEYCLOAK CASCADE_USER_REMOVAL_ENABLED	false	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場合があります。適切に機能するには、管理ユーザー名 \${che.keycloak.admin_username} とパスワード \${che.keycloak.admin_password} を設定する必要があります。
CHE_KEYCLOAK_ADMIN_USERNAME	NULL	Keycloak 管理者のユーザー名。 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。 \${che.keycloak.cascade_user_remove_enabled} が 'true' に設定されている場合にのみ機能します。
CHE_KEYCLOAK_ADMIN_PASSWORD	NULL	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。 \${che.keycloak.cascade_user_remove_enabled} が 'true' に設定されている場合にのみ機能します。

環境変数名	デフォルト値	詳細
CHE_KEYCLOAK_USERNAME E_REPLACEMENT_PATTERNNS	NULL	<p>ユーザー名の調整の設定。</p> <p>CodeReady Workspaces は、ユーザー名を Kubernetes 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、アイデンティティー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、アイデンティティー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)。</p> <p>例 : \=-,@=-at- では \ は - に、@ は -at- に変更され、ユーザー名 org\user@com は org-user-at-com. になります。</p>

関連資料

- 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. ワークスペーススターゲットプロジェクトの設定

新規ワークスペースがデプロイされる 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。CodeReady Workspaces は、各ワークスペースをユーザー専用のプロジェクトにデプロイして、対象のユーザーが作成した全 CodeReady Workspaces をホストします。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するか、CodeReady Workspaces 管理者が事前に作成しておく必要があります。

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** プロパティー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 ワークスペースプロジェクト設定



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

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。 「[ユーザーが実行できるワークスペース数の設定](#)」を参照してください。



警告

Kubernetes はプロジェクト名の長さを 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

以下の場合は 「[各ユーザーのプロジェクトの事前作成](#)」 を使用します。

- 新規プロジェクトの作成時に、**Che ServiceAccount** に十分なパーミッションがない
- クラスターロールが **self-provisioner** の OpenShift OAuth が **system:authenticated:oauth** グループにリンクされていない
- CodeReady Workspaces が namespace を作成できない

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 プロジェクト設定の値を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識別子は <username> と <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. 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、OpenShift 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、ほぼ有効な名前のみに絞られます。ID が競合しないように、無作為に選択された 6 記号の接尾辞を追加します。再利用できるように、結果は設定に保存されます。

4.2.3. 各ユーザーのプロジェクトの事前作成

各ユーザーにプロジェクトを事前に作成するには、OpenShift のラベルとアノテーションを使用します。このプロジェクトは、**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ーのリファレンス](#) を参照してください。



警告

単一ユーザーに複数の namespace を作成しないでください。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ターゲット namespace の **admin** ロール権限が必要です。

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

- ① 事前に作成される namespace
- ② 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。

4.2.4. 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プロジェクトを更新します。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> ①
rules:
- apiGroups:
  - ""
resources:
- namespaces
verbs:
- update
- get
```

1 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> ①
subjects:
  - kind: ServiceAccount
    name: <service-account-name> ②
    namespace: <service-account-namespace> ③
roleRef:
  kind: ClusterRole
  name: <cluster-role-name> ④
  apiGroup: rbac.authorization.k8s.io
```

① クラスターロールバインディングの名前

② che ServiceAccount の名前

③ CodeReady Workspaces インストール namespace

④ 直前の手順で作成されたクラスターロールの名前



注記

パーミッションがない場合には、CodeReady Workspaces ワークスペースが起動しなくなり、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** を定義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 アクセスマード](#) 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

ストラテジー	詳細	利点	不利な点
per-workspace (デフォルト)	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。
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。	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。 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。

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。

対応するコンテナー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。



注記

common PVC は、ユーザーのワークスペースがすべて削除された時点で削除されます。PVC は、一時ワークスペース以外のワークスペースの開始時に再作成されます。

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 アクセスマードが `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、以下の点を注意してください。

- ワークスペース PVC アクセスマードを **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。[「ユーザーが実行できるワークスペース数の設定」](#) 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用する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。その場合、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> または <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、コンテナー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 はコンテナー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 サーバーのワークプレ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** プロパティーを変更して、目的のストラテジーを設定します。

前提条件

- **oc** ツールが利用できる。

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ーを更新します。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

一時ストレージでは、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。

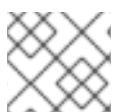
一時モードは、永続ストレージよりも高速な I/O を提供します。このストレージタイプを有効にするには、以下をワークスペース設定に追加します。

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

表4.17 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



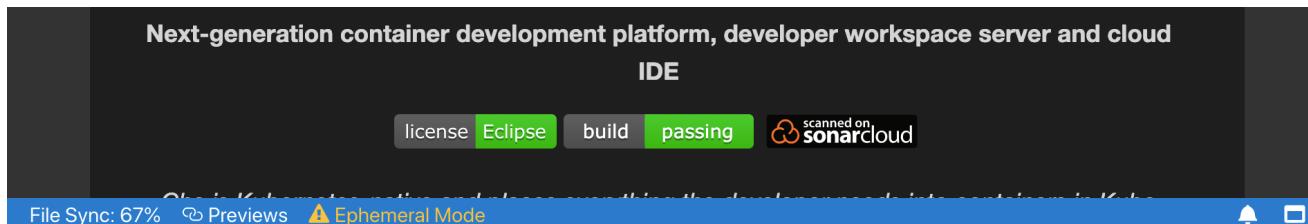
注記

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

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

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



注記

非同期モードには、以下の制限があります。

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

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

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

4.4.4. 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。使用できる値は **persistent**、**ephemeral** および **async** です。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。

CodeReady Workspaces ワークスペースのタイムアウトを増やすには、以下の例を使用して、ワークスペースタイムアウトを 30 分間隔(1800000 ミリ秒)に設定します。

+

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

4.5. ユーザーが実行できるワークスペース数の設定

この記事では、ユーザーが同時に実行できるワークスペースの数を設定する方法を説明します。

4.5.1. Operator を使用したユーザーが実行できるワークスペースの数の設定

この手順では、複数のワークスペースを同時に実行するように、**CodeReady Workspaces**を設定する方法を説明します。複数のワークスペースを実行すると、ユーザーは異なる作業環境を同時に使用できます。

前提条件

- Operator を使用して **CodeReady Workspaces** のインスタンスがインストールされている。
- PVC ストラテジーとアクセスモードの組み合わせは以下の基準を満たしている。
 - **ReadWriteMany** アクセスマードおよび任意の PVC ストラテジー
 - **ReadWriteOnce** アクセスマードおよび **per-workspace** または **unique** PVC ストラテジー

[「ストレージストラテジーの設定」](#) を参照してください。

- <number-of-workspaces> プレースホルダーの値を特定している。



注記

値が **-1** の場合には、各ユーザーが実行できるワークスペース数は無制限になります。値が正の整数である場合には、この整数値と同じ数だけ、ワークスペースを実行できます。デフォルト値は **1** です。

手順

1. CheCluster カスタムリソースの **server** 設定では、**CHE_LIMITS_USER_WORKSPACES_RUN_COUNT** プロパティーを **custom CheProperties** に追加して、ユーザーが実行できるワークスペース数を設定します。

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

4.6. ユーザーが作成できるワークスペースの数の設定

この記事では、ユーザーが作成できるワークスペースの数を設定する方法を説明します。

4.6.1. Operator を使用したユーザーが作成できるワークスペースの数の設定

この手順では、ユーザーが作成できるワークスペースの数を設定する方法を説明します。複数のワークスペースを作成すると、ユーザーは同時に設定が異なるワークスペースにアクセスできます。

前提条件

- Operator を使用して **CodeReady Workspaces** のインスタンスがインストールされている。
- <number-of-workspaces> プレースホルダーの値を特定している。



注記

値が **-1** の場合には、ユーザーが無制限にワークスペースを作成できます。値が正の整数である場合には、この整数値と同じ数だけ、ワークスペースを作成できます。デフォルト値は **-1** です。

手順

- CheCluster カスタムリソースの **server** 設定では、**CHE_LIMITS_USER_WORKSPACES_RUN_COUNT** プロパティーを **custom CheProperties** に追加して、ユーザーが作成できるワークスペース数を設定します。

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

```

kind: CheCluster
# [...]
spec:
server:
# [...]
customCheProperties:
  CHE_LIMITS_USER_WORKSPACES_COUNT: "<number-of-workspaces>"
```

4.7. 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。

4.7.1. 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` ツールが利用できる。

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 でサポートされる値は次のとおりです。

- [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>' ①

```

① - 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ーを更新します。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>"}]' \ ①
-n openshift-workspaces

```

① - ワークスペース公開ストラテジーの使用

4.7.2. 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

4.7.2.1. Multihost ストラテジー

Multihost 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。

4.7.2.2. 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。

单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ーのある特別な 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)リバースプロキシー 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を定義するには、CodeReady Workspaces インスタンスの

CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE_ENDPOINT_EXPOSURE 環境変数を定義します。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。

4.7.2.2.1. devfile エンドポイント : **single-host**

CHE_INFRA_KUBERNETES_SINGLEHOST_WORKSPACE_DEVFILE_ENDPOINT_EXPOSURE: 'single-host'

この单一ホスト設定は、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ーの背後にあるためです。

たとえば、ユーザーが仮の \a href="https://codeready-<openshift_deployment_name>. <domain_name>/component-prefix-djh3d/app/index.php">https://codeready-<openshift_deployment_name>. <domain_name>/component-prefix-djh3d/app/index.php URL にアクセスする場合に、アプリケーションには要求が <https://internal-host/app/index.php> に送信されるように表示されます。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。

4.7.2.2.2. 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://<host>> ページはセキュリティー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。

4.7.3. 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ー上の影響について説明します。

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

4.7.3.1. JSON Web トークン (JWT) プロキシー

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

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

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

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

4.7.3.2. 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

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

4.7.3.3. 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

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

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

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

4.7.3.5. フィッシング攻撃

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

4.8. ワークスペース NODESELECTOR の設定

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

手順

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

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



重要

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

大規模なマルチゾーンクラスターの異なるゾーンに Pod および PVC がスケジュールされないようにするには、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)。

新規に作成された **StorageClass** の名前を、**CHE_INFRA_KUBERNETES_PVC_STORAGE_CLASS_NAME** 環境変数で CodeReady Workspaces に指定します。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。

4.9. 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 CodeReady Workspaces を設定する方法を説明します。

前提条件

- **oc** ツールが利用できる。
- 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

秘密鍵と証明書のペアを生成するには、他の CodeReady Workspaces ホストと同じ認証局(CA)を使用する必要があります。



重要

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

手順

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

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

2. TLS Secret を作成します。

```
$ oc create secret TLS ${secret} \
--key ${key_file} \
--cert ${cert_file} \
-n openshift-workspaces
```

- 1 TLS Secret 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 Secret 名

4. CodeReady Workspaces がすでにデプロイされている場合は、すべての CodeReady Workspaces コンポーネントのロールアウトが完了するまで待ちます。

4.10. OPENSHIFT ルートの設定

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

前提条件

- oc** ツールが利用できる。
- OpenShift で実行される CodeReady Workspaces のインスタンス。

手順

- OpenShift Route のラベルを設定するには、カスタムリソースを更新します。



重要

コンマを使用して、ラベルを区切れます : **key1=value1,key2=value2**

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

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

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

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

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

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

- OpenShift Route のアノテーションを設定するには、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。



重要

オブジェクトを使用してアノテーションを指定します : **{"key1": "value1", "key2": "value2"}**

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p \
```

```
'[{"op": "replace", "path": "/spec/server/cheServerIngress/annotations", \
"value": <annotations for a codeready-workspaces server ingress>}]'

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

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

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

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

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

4.11. ルーターのシャード化と連携するように OPENSHIFT ルートを設定

この手順では、OpenShift Route が[ルーターのシャード化](#)と連携するようにラベル、アノテーション、およびドメインを設定する方法を説明します。本章では、既存のインスタンスの設定プロセス、またはインストール予定の設定プロセスを説明します。

前提条件

- **oc** および **crwctl** ツールが利用できる。

手順

- 新規の OperatorHub インストールの場合 :

1. OpenShift Container Platform を使用して CodeReady Workspaces クラスターを入力し、CheCluster カスタムリソース(CR)を作成します。[Red Hat CodeReady Workspaces Operator のインスタンスの作成](#) を参照してください。
2. 以下の値を codeready-workspaces カスタムリソース(CR)に設定します。

```
spec:
server:
  devfileRegistryRoute:
    labels: <labels> ①
    domain: <domain> ②
    annotations: ③
      key1: value1
      key2: value2
  pluginRegistryRoute:
```

```

labels: <labels> ④
domain: <domain> ⑤
annotations: ⑥
  key1: value1
  key2: value2
dashboardRoute:
  labels: <labels> ⑦
  domain: <domain> ⑧
  annotations: ⑨
    key1: value1
    key2: value2
cheServerRoute:
  labels: <labels> ⑩
  domain: <domain> ⑪
  annotations: ⑫
    key1: value1
    key2: value2
customCheProperties:
  CHE_INFRA_OPENSHIFT_ROUTE_LABELS: <labels> ⑬
  CHE_INFRA_OPENSHIFT_ROUTE_HOST_DOMAIN_SUFFIX: <domain> ⑭
auth:
  identityProviderRoute:
    labels: <labels> ⑮
    domain: <domain> ⑯
    annotations: ⑰
      key1: value1
      key2: value2

```

① ④ ⑦ ⑩ ⑬ ⑮ ターゲット Ingress コントローラーがルートからサービスのセットを
フィルタリングする時に使用するラベルのコンマ区切りの一覧

② ⑤ ⑧ ⑪ ⑭ ⑯ ターゲット Ingress コントローラーが提供する DNS 名

③ ⑥ ⑨ ⑫ ⑰ リソースに保管された構造化されていないキー値のマップ

- 新規の `crwctl` インストールの場合は、以下のようにになります。

- 以下を使用してインストールを設定します。

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

patch.yaml ファイルには以下を含める必要があります。

```

spec:
server:
  devfileRegistryRoute:
    labels: <labels> ①
    domain: <domain> ②
    annotations: ③
      key1: value1
      key2: value2
  pluginRegistryRoute:

```

```

labels: <labels> ④
domain: <domain> ⑤
annotations: ⑥
  key1: value1
  key2: value2
dashboardRoute:
  labels: <labels> ⑦
  domain: <domain> ⑧
  annotations: ⑨
    key1: value1
    key2: value2
cheServerRoute:
  labels: <labels> ⑩
  domain: <domain> ⑪
  annotations: ⑫
    key1: value1
    key2: value2
customCheProperties:
  CHE_INFRA_OPENSHIFT_ROUTE_LABELS: <labels> ⑬
  CHE_INFRA_OPENSHIFT_ROUTE_HOST_DOMAIN_SUFFIX: <domain> ⑭
auth:
  identityProviderRoute:
    labels: <labels> ⑮
    domain: <domain> ⑯
    annotations: ⑰
      key1: value1
      key2: value2

```

① ④ ⑦ ⑩ ⑬ ⑮ ターゲット Ingress コントローラーがルートからサービスのセットを
フィルタリングする時に使用するラベルのコンマ区切りの一覧

② ⑤ ⑧ ⑪ ⑭ ⑯ ターゲット Ingress コントローラーが提供する DNS 名

③ ⑥ ⑨ ⑫ ⑰ リソースに保管された構造化されていないキー値のマップ

- 既存の CodeReady Workspaces インストールの場合 :

1. **oc** ツールを使用して **codeready-workspaces** CR を更新します。

- a. ラベルを設定するには、以下を実行します。



重要

コンマを使用して、ラベルを区切れます : **key1=value1,key2=value2**

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

```
$ oc patch checluster/codeready-workspaces -n openshift-workspaces --type=json -p
 \  

```

```
'[{"op": "replace", "path": "/spec/server/pluginRegistryRoute/labels", \
"value": "<labels for a plug-ins registry route>"}]'

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

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

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

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

- b. ドメインを設定するには、以下を行います。

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

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

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

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

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

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

- c. アノテーションを設定するには、以下を行います。



重要

オブジェクトを使用してアノテーションを指定します : {"key1": "value1", "key2" : "value2"}

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

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

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

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

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

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

- ① 自己署名証明書へのパス
- ② Git サーバーの HTTPS 接続のホストおよびポート（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. Git リポジトリの自己署名証明書を使用するように CodeReady Workspaces 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ーは、自己署名証明書のあるファイルを含む特殊なボリュームをマウントします。リポジトリの.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/ca.crt
```

4.13. 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ーストレージを柔軟に設定します。クラス名を使用して、静的にプロビジョニングされた永続ボリュームを CodeReady Workspaces PVC にバインドすることもできます。

手順

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

1. ストレージクラス名を定義します。
これを行うには、以下のいずれかの方法を使用します。

- **server:deploy** コマンドの引数の使用

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

- カスタムリソース YAML ファイルを使用してストレージクラス名を定義します。
 - i. CodeReady Workspaces インストールに定義されたカスタムリソースで YAML ファイルを作成します。
 - ii. フィールド **spec#storage#postgresPVCStorageClassName** および **spec#storage#workspacePVCStorageClassName** を定義します。

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

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

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

2. CodeReady Workspaces を、ワークスペースを1つ目の永続ボリュームに、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。
 - a. カスタムリソース YAML ファイルを変更します。
 - **pvcStrategy** を **common** に設定します。
 - 単一のプロジェクトでワークスペースを開始するように CodeReady Workspaces を設定します。

- **postgresPVCStorageClassName** および **workspacePVCStorageClassName** のストレージクラス名を定義します。
- YAML ファイルの例:

```

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

- b. 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** と **gid** を定義します。PostgreSQL ボリュームには **uid=26** と **gid=26** が必要です。

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

CodeReady Workspaces コンポーネントの外部通信は、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ー、ソースコードリポジトリ、アイデンティー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され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るため、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 クラスター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、プロジェクトのラベルが付いた ConfigMaps 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

クラスターに、[クラスター全体のプロキシー設定](#)を使用して追加されたクラスター全体の信頼される 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 に自動的に **config.openshift.io/inject-trusted-cabundle="true"** ラベルを付けます。
- このアノテーションに基づいて、OpenShift は ConfigMap の **ca-bundle.crt** キー内にクラスター全体で信頼される CA 証明書を自動的に挿入します。



重要

一部の CodeReady Workspaces コンポーネントには、エンドポイントを信頼するための完全な証明書チェーンが必要です。クラスターが中間証明書で設定されている場合には、チェーン全体（自己署名ルートを含む）を CodeReady Workspaces に追加する必要があります。

4.14.1. 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** ツールが利用できる。
- CodeReady Workspaces の namespace が存在する。

手順

- 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

- 証明書ファイルは、通常、**.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してください。
 - ca-bundle.crt**
 - ca.crt**

- 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** と **app.kubernetes.io/component=ca-bundle** の両方のラベルを使用して作成した 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。

4.14.2. 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-cl} 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 サーバーコンテナーのフォルダー **/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
```

4.14.3. 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 コンテナー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。さらに、予約されている **ca-bundl.crt** エントリーが含まれる場合があります。

```
$ 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ー(とくに Theia IDE コンテナー)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 コンテナーの /public-certs フォルダーを検査し、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
```

4.15. 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。

- プロキシー、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、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 ツールが利用できる。
- 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。

- 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/disableInternalClusterSVCNames", "value": true}]'
```

- 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/disableInternalClusterSVCNames", "value": false}]'
```

4.16. 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。

4.16.1. 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

- 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
```

- 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
```

- 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。

4.16.2. 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。

The screenshot shows the Codeready Workspaces configuration interface. On the left, there's a sidebar with a 'Codeready' header and a 'Configure' section containing links for 'Clients', 'Client Scopes', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. Below these is a 'Realm Settings' link, which is highlighted in blue. The main area has a 'Codeready' header with a trash can icon. Below it are several tabs: General, Login, Keys, Email, Themes (which is selected and highlighted in blue), Cache, Tokens, and Client Registration. Under the Themes tab, there are four dropdown menus: 'Login Theme' (set to 'codeready-username-readonly'), 'Account Theme' (set to 'rh-sso'), 'Admin Console Theme' (set to 'rh-sso'), and 'Email Theme' (set to 'rh-sso'). At the bottom, there's a 'Save' button and a 'Cancel' button.

2. Themes タブに移動します。
3. 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。

4.17. シークレットまたは CONFIGMAP をファイルまたは環境変数として CODEREADY WORKSPACES コンテナーにマウントする

シークレットは、以下のような機密データを格納する OpenShift オブジェクトです。

- ユーザー名
- パスワード
- 認証トークン

(暗号化された形式)。

ユーザーは、以下のように、機密データを含む OpenShift シークレットまたは CodeReady Workspaces 管理コンテナーの設定を含む ConfigMap をマウントできます。

- 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。

4.17.1. シークレットまたは ConfigMap をファイルとして CodeReady Workspaces コンテナーにマウントする

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 シークレットまたは ConfigMap を作成します。作成される予定のオブジェクトのラベルは、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>-<OBJECT_KIND>**
 - <DEPLOYMENT_NAME> には、以下のデプロイメントのいずれかを使用します。
 - postgres**
 - keycloak**
 - devfile-registry**
 - plugin-registry**
 - codeready**
および
 - <jasper_KIND> は以下のいずれかになります。
 - secret**
または
 - configmap**

例4.3 例:

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

```

または

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

アノテーションは、指定されるオブジェク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。

1. アノテーション値を設定します。

- **che.eclipse.org/mount-as: file** - オブジェクトをファイルとしてマウントするように指定します。
- **che.eclipse.org/mount-path: <TARGET_PATH>** - 必要なマウントパスを指定します。

例4.4 例:

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

または

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

OpenShift オブジェク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ー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。

例4.5 例:

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-data
```

```

labels:
  app.kubernetes.io/part-of: che.eclipse.org
  app.kubernetes.io/component: codeready-secret
annotations:
  che.eclipse.org/mount-as: file
  che.eclipse.org/mount-path: /data
data:
  ca.crt: <base64 encoded data content here>

```

または

```

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

```

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



重要

CodeReady Workspaces コンテナーの変更を表示できるようにするには、オブジェクトを完全に再作成します。

4.17.2. シークレットまたは ConfigMap を環境変数として CodeReady Workspaces コンテナーにマウントする

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 シークレットまたは ConfigMap を作成します。作成される予定のオブジェクトのラベルは、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>-<OBJECT_KIND>**
 - <DEPLOYMENT_NAME>** には、以下のデプロイメントのいずれかを使用します。

- **postgres**
- **keycloak**
- **devfile-registry**
- **plugin-registry**
- **codeready**
および
- <jasper_KIND> は以下のいずれかになります。
 - **secret**
または
 - **configmap**

例4.6 例:

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

```

または

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

```

アノテーションは、指定されるオブジェク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。

1. アノテーション値を設定します。

- **che.eclipse.org/mount-as: env** -: オブジェクトを環境変数としてマウントするように指定します。
- **che.eclipse.org/env-name: <FOOO_ENV>**: オブジェクトキー値のマウントに必要な環境変数名を指定します。

例4.7 例:

```
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

```

または

```

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

```

これにより、2つの環境変数が

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

CodeReady Workspaces コンテナーにプロビジョニングされます。

オブジェク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。

例4.8 例:

```

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

```

または

```

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

```

これにより、2つの環境変数が

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

CodeReady Workspaces コンテナー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、オブジェク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。



重要

CodeReady Workspaces コンテナーの変更を表示できるようにするには、オブジェクトを完全に再作成します。

4.18. DEV WORKSPACE エンジンの有効化

この手順では、Dev Workspace エンジンを有効にして Devfile 2.0.0 ファイル形式をサポートし、既存のインスタンスやインストールを行う方法を説明します。

前提条件

- **oc** および **crwctl** ツールが利用できる。

手順

- 新規の OperatorHub インストールの場合：
 1. OpenShift Container Platform を使用して Red Hat CodeReady Workspaces クラスターを入力し、CheCluster カスタムリソース(CR)を作成します。[Red Hat CodeReady Workspaces Operator のインスタンスの作成](#) を参照してください。
 2. 以下の値を codeready-workspaces カスタムリソース(CR)に設定します。

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

- 新規の **crwctl** インストールの場合は、以下のようになります。

- 以下を使用して **crwctl** インストールを設定します。

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

patch.yaml には以下を含める必要があります。

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

- 既存の CodeReady Workspaces インストールの場合：

- oc** ツールを使用して **codeready-workspaces** CR を更新します。

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

関連資料

本章のインストール方法は、[3章CodeReady Workspaces のインストール](#) を参照してください。

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 2.10 から CodeReady Workspaces 2.11 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 の新規バージョンが公開されるとアップグレードプロセスが開始されます。**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

- OpenShift Web コンソールを開きます。
- Operators** → **Installed Operators** セクションに移動します。
- 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。
- 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**)。
- 1 requires approval** をクリックしてから、**Preview Install Plan** をクリックします。
- 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

- 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。完了時に、ステータスは **Succeeded** および **Up to date** に変更されます。
- 2.11 のバージョン番号がページ下部に表示されます。

関連資料

- 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 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` が利用可能で、更新されている。[「crwctl CLI 管理ツールのインストール」](#) を参照してください。

手順

- 実行中の全 CodeReady Workspaces 2.10 ワークスペースに対する変更を Git リポジトリに保存し、プッシュします。
- CodeReady Workspaces 2.10 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。
- CodeReady Workspaces をアップグレードします。

```
$ 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

- CodeReady Workspaces インスタンスに移動します。
- 2.11 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 2.10 からバージョン 2.11 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、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 2.10。「[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** 2.11 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 (**codeready-workspaces-server**, **keycloak**, **devfile-registry**, **plugin-registry**) があります。

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

例5.1 公開インターネットから切断された、会社または組織が所有するネットワーク

ネットワーク管理者は、クラスターからのトラフィックを OpenShift Route ホスト名にルーティングできるようにする必要があります。

例5.2 クラウドプロバイダーのプライベートサブネットワーク

トラフィックがノードから出て、外部に表示されるロードバランサーに到達できるようにするプロキシー設定を作成します。

5.3.2. オフラインレジストリーアイメージのビルド

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

本セクションでは、オフラインの devfile レジストリーア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-2.11-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 スクリプトの詳細な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。

関連資料

- [レジストリーのカスタマイズ](#)

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。参照: [Installing Yarn](#).
- **./node_modules/.bin** が **PATH** 環境変数にある。
- [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-2.11-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 \
--skip-digest-generation
```



注記

build.sh スクリプトの詳細な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。

関連資料

- レジストリーのカスタマイズ

5.3.3. プライベートレジストリーの準備

前提条件

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

表5.1 サンプルで使用されるプレースホルダー

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

手順

- 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>`に追加します。
- `/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>']
```

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

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

注記

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

使用	プレフィックスまたはキーワード
Essential	stacks- , plugin- または -openj9- ではない
Workspaces	stacks- , plugin-
IBM Z および IBM Power Systems	-openj9-

注記

openj9 のサフィックスがつくイメージは、x86_64 で使用される OpenJDK イメージと同等の **Eclipse OpenJ9** イメージです。IBM Power Systems および IBM Z は、これらのシステムでパフォーマンスを改善するために Eclipse OpenJ9 を使用します。

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@sha256:20fd31c45d769526d45eaf6738a6d4af1520a844126a2a2e510c304a81b7249a	configbump-rhel8@sha256:20fd31c45d769526d45eaf6738a6d4af1520a844126a2a2e510c304a81b7249a

<source-image>

<target-image>

<code>registry.redhat.io/codeready-workspaces/crw-2-rhel8-operator@sha256:a41f7b950c5131a6bc08b1e094db2da9b784e6083ddaa4aa68512f3947798702</code>	<code>crw-2-rhel8-operator@sha256:a41f7b950c5131a6bc08b1e094db2da9b784e6083ddaa4aa68512f3947798702</code>
<code>registry.redhat.io/codeready-workspaces/dashboard-rhel8@sha256:1c37bdffae8cdc154d88b94ab38e868f7e33486c81b6c3bded36dfdf85b81a4</code>	<code>dashboard-rhel8@sha256:1c37bdffae8cdc154d88b94ab38e868f7e33486c81b6c3bded36dfdf85b81a4</code>
<code>registry.redhat.io/codeready-workspaces/devfileregistry-rhel8@sha256:b164968dbd52c72f39533bec4efd3ad3cce3acb6060495e472dd9c3f2908fb8c</code>	<code>devfileregistry-rhel8@sha256:b164968dbd52c72f39533bec4efd3ad3cce3acb6060495e472dd9c3f2908fb8c</code>
<code>registry.redhat.io/codeready-workspaces/devworkspace-controller-rhel8@sha256:c88242524a9074a58bc7d20cb8411d37e7e752358ab80366533b8165bb9f95b0</code>	<code>devworkspace-controller-rhel8@sha256:c88242524a9074a58bc7d20cb8411d37e7e752358ab80366533b8165bb9f95b0</code>
<code>registry.redhat.io/codeready-workspaces/devworkspace-rhel8@sha256:c18f166f570ca572c94472b7a3bd5127b48521e777ea09dcad6f78ad66cd7a13</code>	<code>devworkspace-rhel8@sha256:c18f166f570ca572c94472b7a3bd5127b48521e777ea09dcad6f78ad66cd7a13</code>
<code>registry.redhat.io/codeready-workspaces/jwtproxy-rhel8@sha256:44acafb02cce3d3fe8b57da2e27547b502c4088624935ffe7f3aa06a55d08bba</code>	<code>jwtproxy-rhel8@sha256:44acafb02cce3d3fe8b57da2e27547b502c4088624935ffe7f3aa06a55d08bba</code>
<code>registry.redhat.io/codeready-workspaces/machineexec-rhel8@sha256:bfdd8cf61a6fad757f1e8334aa84dbf44baddf897ff8def7496bf6dbc066679d</code>	<code>machineexec-rhel8@sha256:bfdd8cf61a6fad757f1e8334aa84dbf44baddf897ff8def7496bf6dbc066679d</code>

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/plugin-java11-openj9-rhel8@sha256:8d9930cd3c0b2fa72a6c0d880b4d0b330b1a7a51491f09175134dcc79f2cb376	plugin-java11-openj9-rhel8@sha256:8d9930cd3c0b2fa72a6c0d880b4d0b330b1a7a51491f09175134dcc79f2cb376
registry.redhat.io/codeready-workspaces/plugin-java11-rhel8@sha256:d0337762e71fd4badabcb38a582b2f35e7e7fc1c9c0f2e841e339d45b7bd34ed	plugin-java11-rhel8@sha256:d0337762e71fd4badabcb38a582b2f35e7e7fc1c9c0f2e841e339d45b7bd34ed
registry.redhat.io/codeready-workspaces/plugin-java8-openj9-rhel8@sha256:d7ec33ce2fa61a06fade63e2b516409c465bd5516030dd482e2f4bdb2d676c9f	plugin-java8-openj9-rhel8@sha256:d7ec33ce2fa61a06fade63e2b516409c465bd5516030dd482e2f4bdb2d676c9f
registry.redhat.io/codeready-workspaces/plugin-java8-rhel8@sha256:b2ceb0039c763e6a38aa370157b476ecb08faf8b2bfb680bada774e149583d62	plugin-java8-rhel8@sha256:b2ceb0039c763e6a38aa370157b476ecb08faf8b2bfb680bada774e149583d62
registry.redhat.io/codeready-workspaces/plugin-kubernetes-rhel8@sha256:45535630e37e3e317772f36b28b47859d32ad1e82505a796139682cdbefb03b8	plugin-kubernetes-rhel8@sha256:45535630e37e3e317772f36b28b47859d32ad1e82505a796139682cdbefb03b8
registry.redhat.io/codeready-workspaces/plugin Openshift-rhel8@sha256:d2384caf870c497913168508be0d846412c68ace9724baa37ca3c6be9aa4772	plugin-openshift-rhel8@sha256:d2384caf870c497913168508be0d846412c68ace9724baa37ca3c6be9aa4772
registry.redhat.io/codeready-workspaces/pluginbroker-artifacts-rhel8@sha256:a9bf68e6dabbaaf3e97afe4ac6e97a317e8fd9c05c88e5801fbf01aaa1ebb99	pluginbroker-artifacts-rhel8@sha256:a9bf68e6dabbaaf3e97afe4ac6e97a317e8fd9c05c88e5801fbf01aaa1ebb99
registry.redhat.io/codeready-workspaces/pluginbroker-metadata-rhel8@sha256:727f80af1e1f6054ac93cad165bc392f43c951681936b979b98003e06e759643	pluginbroker-metadata-rhel8@sha256:727f80af1e1f6054ac93cad165bc392f43c951681936b979b98003e06e759643

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/pluginregistry- rhel8@sha256:5d19f7c5c0417940c52e552 c51401f729b9ec16868013e016d7b80342c d8de4e	pluginregistry- rhel8@sha256:5d19f7c5c0417940c52e552 c51401f729b9ec16868013e016d7b80342c d8de4e
registry.redhat.io/codeready-workspaces/server- rhel8@sha256:e79e0a462b4dd47ecaac2f5 14567287c44e32437496b2c214ebc2bc005 5c4aa9	server- rhel8@sha256:e79e0a462b4dd47ecaac2f5 14567287c44e32437496b2c214ebc2bc005 5c4aa9
registry.redhat.io/codeready-workspaces/stacks-cpp- rhel8@sha256:31ef0774342bc1dbcd91e3 b85d68d7a28846500f04ace7a5dfa3116c0c edfeb1	stacks-cpp- rhel8@sha256:31ef0774342bc1dbcd91e3 b85d68d7a28846500f04ace7a5dfa3116c0c edfeb1
registry.redhat.io/codeready-workspaces/stacks-dotnet- rhel8@sha256:6ca14e5a94a98b15f39a353 e533cf659b2b3937a86bd51af175dc3eadd 8b80d5	stacks-dotnet- rhel8@sha256:6ca14e5a94a98b15f39a353 e533cf659b2b3937a86bd51af175dc3eadd 8b80d5
registry.redhat.io/codeready-workspaces/stacks-golang- rhel8@sha256:30e71577cb80ffaf1f67a292 b4c96ab74108a2361347fc593ccb5057846 29db2	stacks-golang- rhel8@sha256:30e71577cb80ffaf1f67a292 b4c96ab74108a2361347fc593ccb5057846 29db2
registry.redhat.io/codeready-workspaces/stacks-php- rhel8@sha256:bb7f7ef0ce58695aaf29b33 55dd9ee187a94d1d382f68f329f9664ca017 72ba2	stacks-php- rhel8@sha256:bb7f7ef0ce58695aaf29b33 55dd9ee187a94d1d382f68f329f9664ca017 72ba2
registry.redhat.io/codeready-workspaces/theia-endpoint- rhel8@sha256:abb4f4c8e1328ea9fc5ca4f e0c809ec007fe348e3d2ccd722e5ba75c02f f448f	theia-endpoint- rhel8@sha256:abb4f4c8e1328ea9fc5ca4f e0c809ec007fe348e3d2ccd722e5ba75c02f f448f
registry.redhat.io/codeready-workspaces/theia- rhel8@sha256:5ed38a48d18577120993cd 3b673a365e31aeb4265c5b4a95dd9d0ac74 7260392	theia- rhel8@sha256:5ed38a48d18577120993cd 3b673a365e31aeb4265c5b4a95dd9d0ac74 7260392

<source-image>	<target-image>
registry.redhat.io/codeready-workspaces/traefik-rhel8@sha256:6704bd086f0d971ecedc1dd6dc7a90429231fdfa86579e742705b31cbedbd8b2	traefik-rhel8@sha256:6704bd086f0d971ecedc1dd6dc7a90429231fdfa86579e742705b31cbedbd8b2
registry.redhat.io/jboss-eap-7/eap-xp3-openj9-11-openshift-rhel8@sha256:53684e34b0dbe8560d2c330b0761b3eb17982edc1c947a74c36d29805bda6736	eap-xp3-openj9-11-openshift-rhel8@sha256:53684e34b0dbe8560d2c330b0761b3eb17982edc1c947a74c36d29805bda6736
registry.redhat.io/jboss-eap-7/eap-xp3-openjdk11-openshift-rhel8@sha256:3875b2ee2826a6d8134aa3b80ac0c8b5ebc4a7f718335d76dfc3461b79f93d19	eap-xp3-openjdk11-openshift-rhel8@sha256:3875b2ee2826a6d8134aa3b80ac0c8b5ebc4a7f718335d76dfc3461b79f93d19
registry.redhat.io/jboss-eap-7/eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d	eap74-openjdk8-openshift-rhel7@sha256:b4a113c4d4972d142a3c350e2006a2b297dc883f8ddb29a88db19c892358632d
registry.redhat.io/rh-sso-7/sso74-openj9-openshift-rhel8@sha256:4ff9d6342dfd3b85234ea554b92867c649744ece9aa7f8751aae06bf9d2d324c	sso74-openj9-openshift-rhel8@sha256:4ff9d6342dfd3b85234ea554b92867c649744ece9aa7f8751aae06bf9d2d324c
registry.redhat.io/rh-sso-7/sso74-openshift-rhel8@sha256:b98f0b743dd406be726d8ba8c0437ed5228c7064015c1d48ef5f87eb365522bc	sso74-openshift-rhel8@sha256:b98f0b743dd406be726d8ba8c0437ed5228c7064015c1d48ef5f87eb365522bc
registry.redhat.io/rhel8/postgresql-96@sha256:ed53ca7b191432f7cf9da0fd8629d7de14ade609ca5f38aba443716f83616f2e	postgresql-96@sha256:ed53ca7b191432f7cf9da0fd8629d7de14ade609ca5f38aba443716f83616f2e
registry.redhat.io/rhscl/mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73	mongodb-36-rhel7@sha256:9f799d356d7d2e442bde9d401b720600fd9059a3d8eefea6f3b2ffa721c0dc73

<source-image>	<target-image>
registry.redhat.io/ubi8/ubi-minimal@sha256:31ccb79b1b2c2d6eff1bee0db23d5b8ab598eaf6238417d9813f1346f717c11	ubi8ubi-minimal@sha256:31ccb79b1b2c2d6eff1bee0db23d5b8ab598eaf6238417d9813f1346f717c11

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>
```

関連資料

- 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、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 2.10。「[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 2.11 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

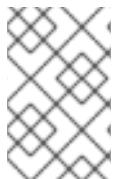
- CodeReady Workspaces 2.10 インスタンスで実行されているすべてのワークスペースで、変更を保存し、Git リポジトリに再度プッシュします。
- CodeReady Workspaces 2.10 インスタンスのすべてのワークスペースを停止します。
- 以下のコマンドを実行します。

```
$ crwctl server:update --che-operator-image=<image-registry>/<organization>/crw-2-rhel8-operator:2.11 -n openshift-workspaces
```

- <image-registry>: 制限された環境でアクセス可能なコンテナーア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. 2.11 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。

5.4. 「PER USER」以外のプロジェクトストラテジーを使用する CODEREADY WORKSPACES のアップグレード

本セクションでは、「per user」以外のプロジェクトストラテジーを使用する CodeReady Workspaces をアップグレードする方法を説明します。

CodeReady Workspaces は、全機密ユーザーデータのストレージとして Kubernetes シークレットを使用することを目的とします。ユーザーごとに1つのプロジェクトとすることで、ワークスペースの設計が簡素化されます。**per user** 以外のプロジェクトストラテジーが非推奨になるのはこのような理由からです。2段階で非推奨化プロセスが進められます。最初の手順では、**per user** 以外のプロジェクトストラテジーは非推奨ですが、使用できます。2番目の手順では、**per user** 以外のプロジェクトストラテジーに対するサポートが削除されます。



注記

最初の手順 と **2番目の手順**との間で、**per user** 以外を使用するプロジェクトストラテジーのシステム環境を自動的にアップグレードするサポートはありません。

前提条件

- **per user** 以外のプロジェクトストラテジーで設定された CodeReady Workspaces。
- **per user** namespace ストラテジー **per user** を設定した CodeReady Workspaces を使用する予定である。

5.4.1. CodeReady Workspaces のアップグレードおよびユーザーデータのバックアップ

手順

1. すべての CodeReady Workspaces ユーザーに今後データが消去されることを通知します。



注記

ワークスペース設定を SCM サーバーにコミットしてデータをバックアップして、ファクトリーを使用して後で復元できます。

2. **per user** namespace ストラテジーを使用して、CodeReady Workspaces を再インストールします。

5.4.2. CodeReady Workspaces のアップグレードおよびユーザーデータの損失

CodeReady Workspaces のアップグレード時にユーザーデータをバックアップしない場合には、ワークスペース設定およびユーザー設定は保持されますが、ランタイムデータはすべて消去されます。

手順

1. すべての CodeReady Workspaces ユーザーに今後データが消去されることを通知します。
2. **per user** プロジェクトストラテジーを変更します。



注記

ユーザーデータをバックアップせずにアップグレードするとデメリットがあります。ランタイムデータが含まれる元の PV は保持されますが、使用されません。これにより、リソースが無駄になることがあります。

関連資料

- 「ワークスペーススターゲットプロジェクトの設定」
- 3章 *CodeReady Workspaces のインストール*
- コードサンプルからのワークスペースの作成
- プロジェクトのソースコードをインポートしてワークスペースを作成する

第6章 CODEREADY WORKSPACES のアンインストール

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

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

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

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

前提条件

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

手順

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



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

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

前提条件

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

手順

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

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

1. クラスターにサインインします。

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

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

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

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

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

4. CodeReady Workspaces クラスターを削除します。

```
$ oc delete checluster red-hat-codeready-workspaces
checluster.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.v2.11** 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.v2.11	Red Hat CodeReady Workspaces	2.11	red-hat-codeready-workspaces.v2.10	Succeeded

- CodeReady Workspaces CSV を削除します。

```
$ oc delete csv red-hat-codeready-workspaces.v2.11
clusterserviceversion.operators.coreos.com "red-hat-codeready-workspaces.v2.11" 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** ツールが利用できる。
- oc** ツールが利用できる。
- 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

- OpenShift クラスターにサインインします。

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

- 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>
```

- 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)
```

- 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
```

- 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. 指定した **ClusterRoleBinding** を削除します。

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