



# Red Hat CodeReady Studio 12.16

## コンテナおよびクラウドベースの開発のスタートガイド

Red Hat CodeReady Studio を使用したコンテナおよびクラウドベースのアプリケーション開発



# Red Hat CodeReady Studio 12.16 コンテナおよびクラウドベースの開発のスタートガイド

---

Red Hat CodeReady Studio を使用したコンテナおよびクラウドベースのアプリケーション開発

Levi Valeeva  
levi@redhat.com

Supriya Takkhi  
sbharadw@redhat.com

Yana Hontyk  
yhontyk@redhat.com

## 法律上の通知

Copyright © 2021 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 概要

本書のトピックでは、コンテナ化されたアプリケーションの開発や、クラウドデプロイメントのアプリケーションの開発を始める方法を説明します。

# 目次

<b>第1章 コンテナおよびクラウドを使用した開発</b>	<b>3</b>
1.1. CODEREADY STUDIO での RED HAT CODEREADY CONTAINERS ツールの使用	3
1.1.1. CodeReady Studio での Red Hat CodeReady コンテナのダウンロードおよびインストール	3
1.1.2. OpenShift Container Platform ツールの使用	6
1.2. CODEREADY STUDIO での RED HAT CONTAINER DEVELOPMENT KIT ツールの使用	12
1.2.1. CodeReady Studio での Container Development Kit のインストール	12
1.2.2. Docker ツールの使用	17
1.2.2.1. Dockerfile の作成	17
1.2.2.2. Container Development Environment を使用した Docker イメージのビルド	23
1.2.2.3. その他のリソース	25
1.2.3. OpenShift Container Platform ツールの使用	25
1.2.4. その他のリソース	31
<b>第2章 OPENSIFT を使用したクラウド向けの開発</b>	<b>32</b>
2.1. CODEREADY STUDIO での OPENSIFT CONTAINER PLATFORM アプリケーションの作成	32
2.1.1. 新規 OpenShift Container Platform コネクションの作成	32
2.1.2. 新規 OpenShift Container Platform プロジェクトの作成	35
2.1.3. 新規 OpenShift Container Platform アプリケーションの作成	37
2.1.4. 既存の OpenShift Container Platform アプリケーションの IDE へのインポート	42
2.1.5. サーバーアダプターを使用したアプリケーションのデプロイ	45
2.1.6. OpenShift Container Platform プロジェクトの削除	48
2.2. OPENSIFT CONTAINER PLATFORM アプリケーションの設定およびリモート監視	50
2.2.1. OpenShift Client Binaries の設定	50
2.2.2. ポート転送の設定	51
2.2.3. Pod ログのストリーミング	54
2.2.4. ビルドログのストリーミング	56
2.3. その他のリソース	58
<b>第3章 DOCKER での開発</b>	<b>59</b>
3.1. DOCKER コネクションの管理	59
3.1.1. CodeReady Studio での Docker アカウントの設定	59
3.1.2. Docker コネクションのテスト	60
3.1.3. Docker コネクションの編集	63
3.2. DOCKER イメージの管理	65
3.2.1. Docker イメージのプル	65
3.2.2. Docker イメージのプッシュ	70
3.2.3. Docker イメージの実行	74
3.2.4. Dockerfile でのイメージのビルド	78
3.3. DOCKER コンテナの管理	80



## 第1章 コンテナおよびクラウドを使用した開発

### 1.1. CODEREADY STUDIO での RED HAT CODEREADY CONTAINERS ツールの使用

Red Hat CodeReady Containers (CRC) は、ローカルシステムに最小限の OpenShift 4 クラスターを提供します。このクラスターは、開発およびテストを目的とする最低限の環境を提供します。これは主に開発者のデスクトップでの実行を目的としています。ヘッドレス、複数開発者、またはチームベースのセットアップなどの他のユースケースでは、[完全な OpenShift インストーラー](#) を使用することが推奨されます。

OpenShift の詳細は、[OpenShift のドキュメント](#) を参照してください。

#### 前提条件

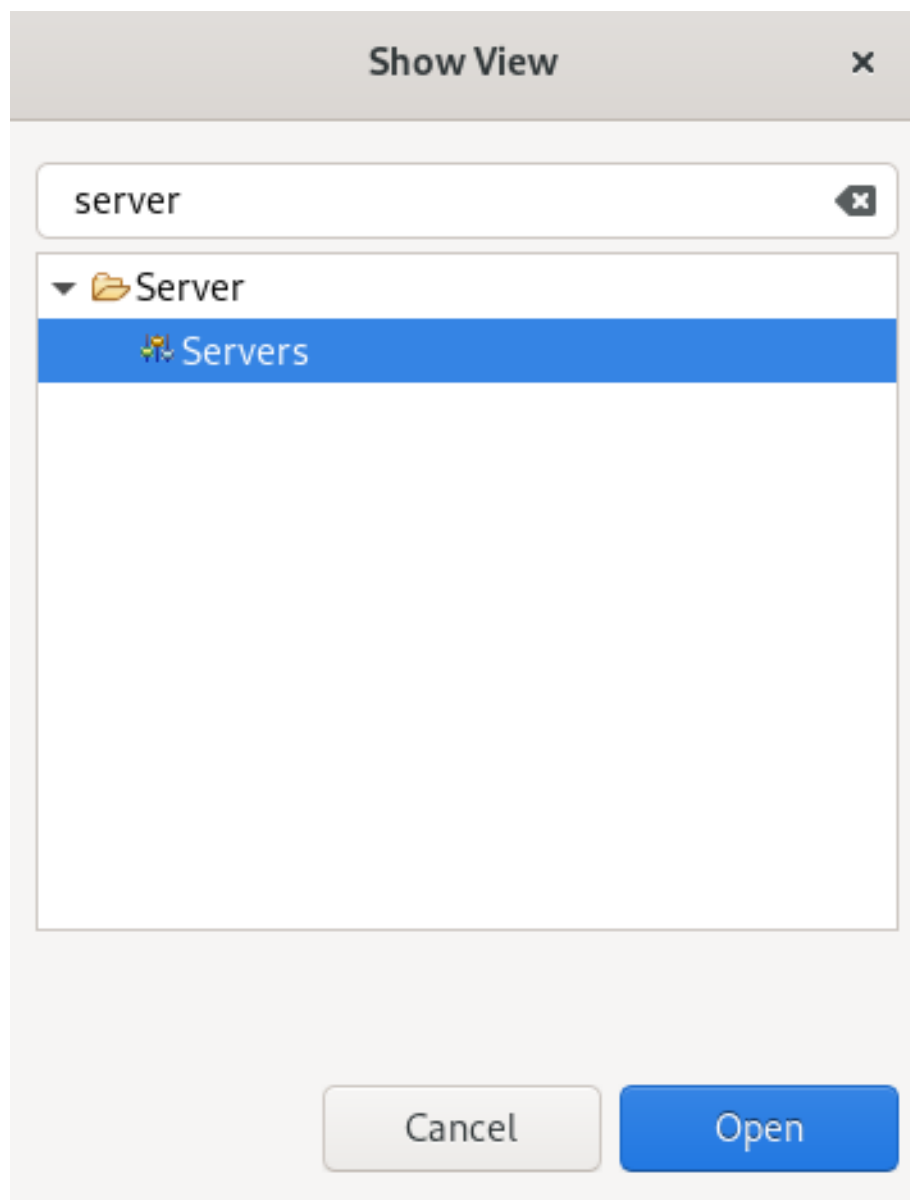
1. [最新リリースの CodeReady Containers とプルシークレットをダウンロード](#) しておく必要があります。
2. CRC ファイルの展開しておく必要があります。  
CRC のインストールおよび設定方法の詳細は、『Getting started with CodeReady Containers Guide』の「[Installation](#)」の章を参照してください。

#### 1.1.1. CodeReady Studio での Red Hat CodeReady コンテナのダウンロードおよびインストール

CodeReady Studio で CodeReady コンテナを設定する方法を説明します。ここでは、本章の前提条件に記載されている手順が完了済みであることを仮定しています。

#### 手順

1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。

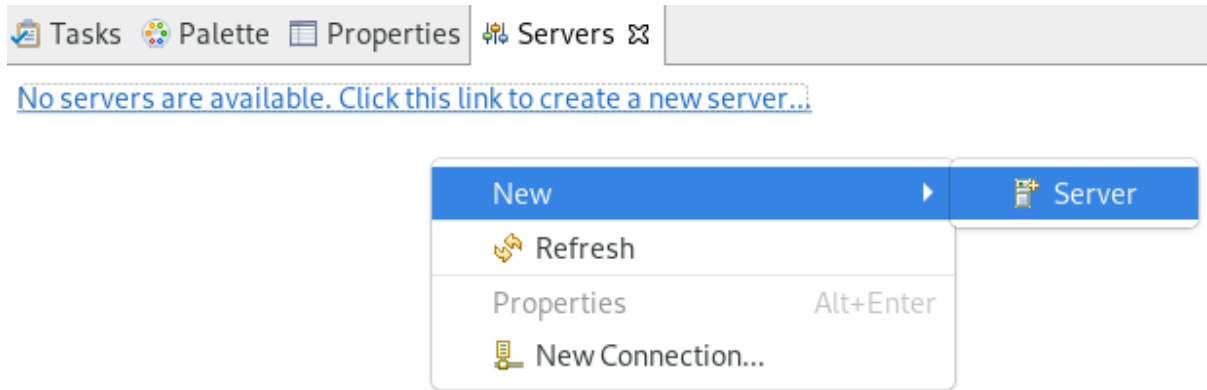


3. 検索フィールドに **Server** と入力します。
4. **Servers** を選択します。
5. **Open** をクリックします。  
**Servers** ビューが表示されます。

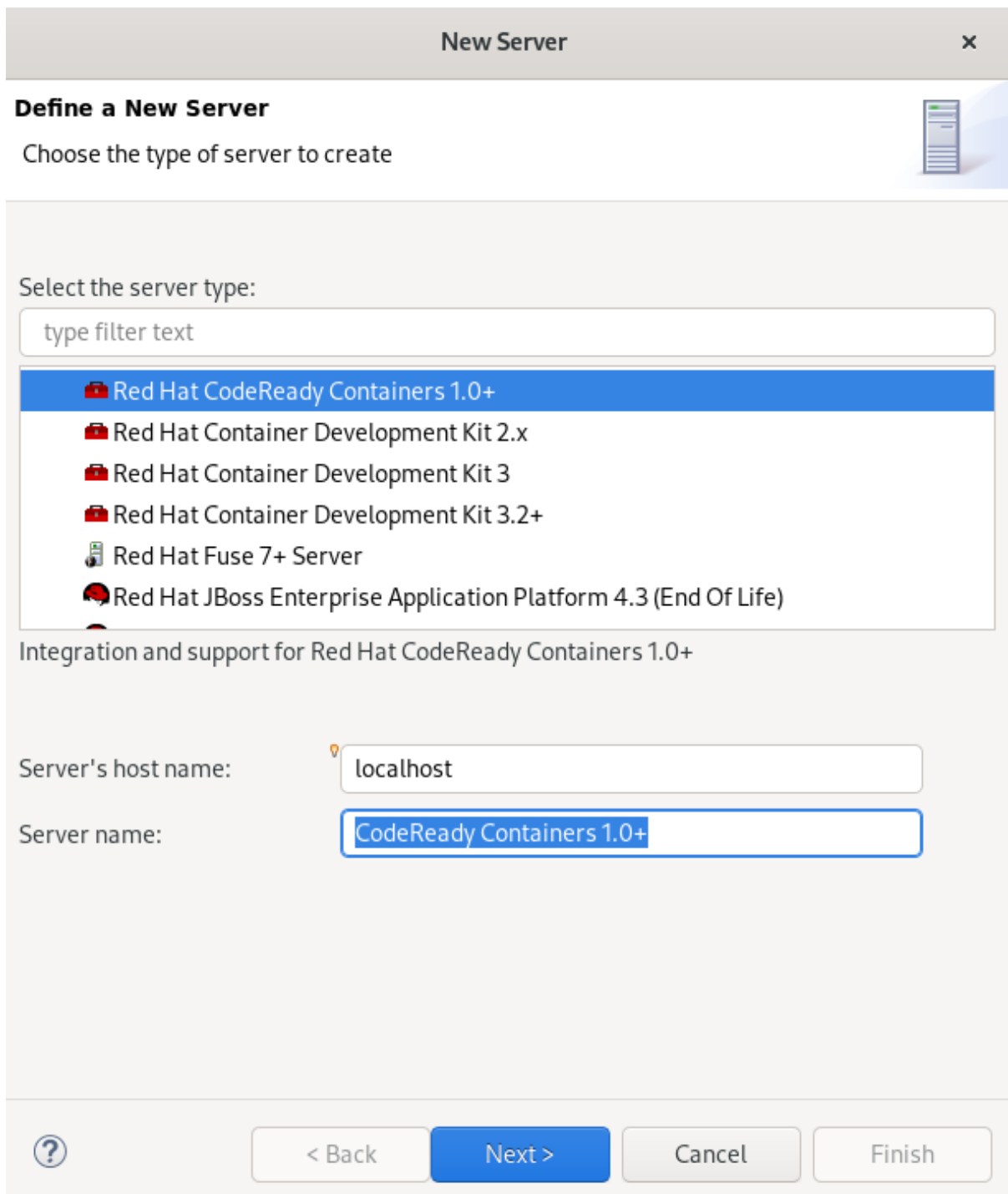


6. **Servers** ビューの任意の場所をクリックします。

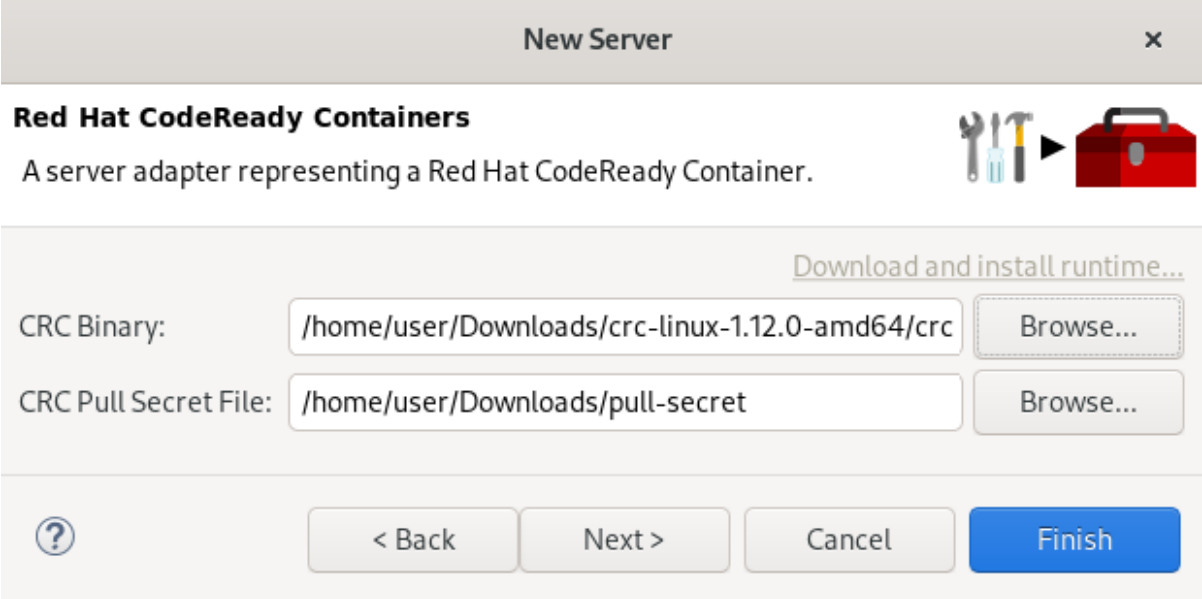




7. **New** → **Server** とクリックします。  
Define a New Server ウィンドウが表示されます。



8. **CodeReady Containers 1.0+** を選択します。
9. **Next** をクリックします。  
CodeReady Containers ウィンドウが表示されます。



**New Server** [Close]

**Red Hat CodeReady Containers**  
A server adapter representing a Red Hat CodeReady Container.

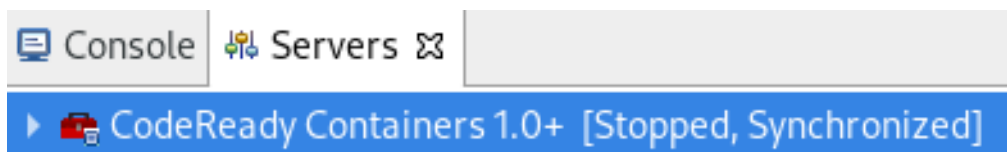
[Download and install runtime...](#)

CRC Binary:

CRC Pull Secret File:

10. **Browse** をクリックして **CRC binary** を見つけます。
11. **Browse** をクリックして **CRC Pull Secret File** を見つけます。
12. **Finish** をクリックします。

新たに追加された CodeReady Containers 1.0+ サーバーアダプターが **Servers** ビューに表示されます。



### 1.1.2. OpenShift Container Platform ツールの使用

CodeReady Studio で OpenShift コンテナを使用する方法を説明します。

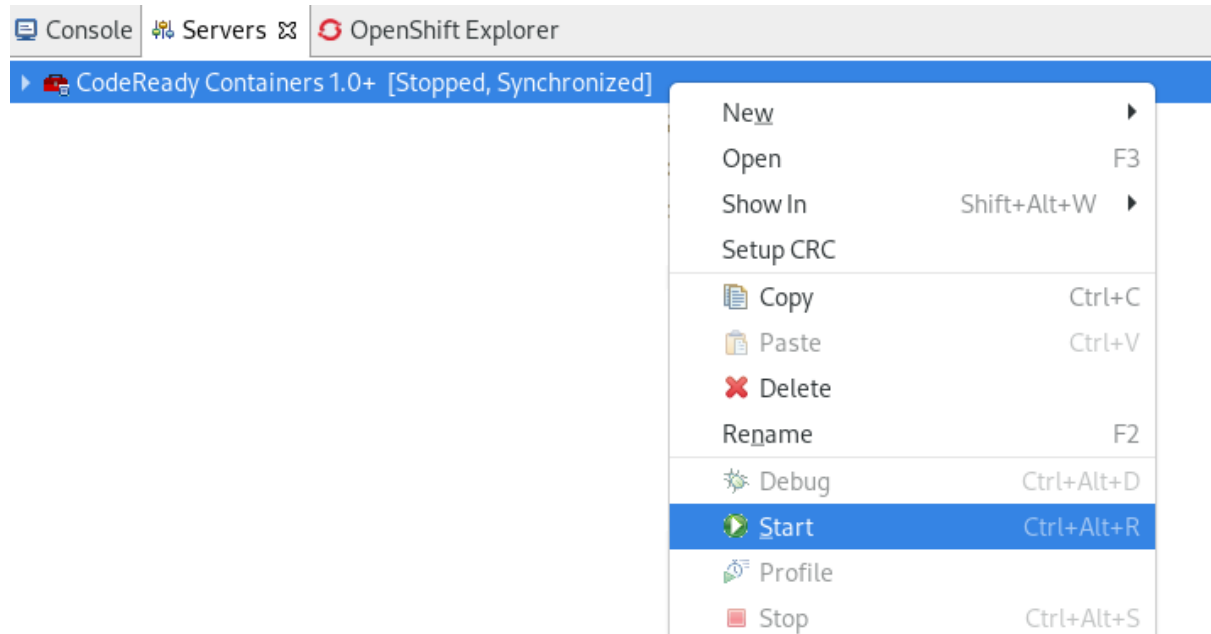
#### 前提条件

- CRC サーバーアダプターが設定済みである必要があります。  
詳細は、「[CodeReady Studio での Red Hat CodeReady コンテナのダウンロードおよびインストール](#)」を参照してください。

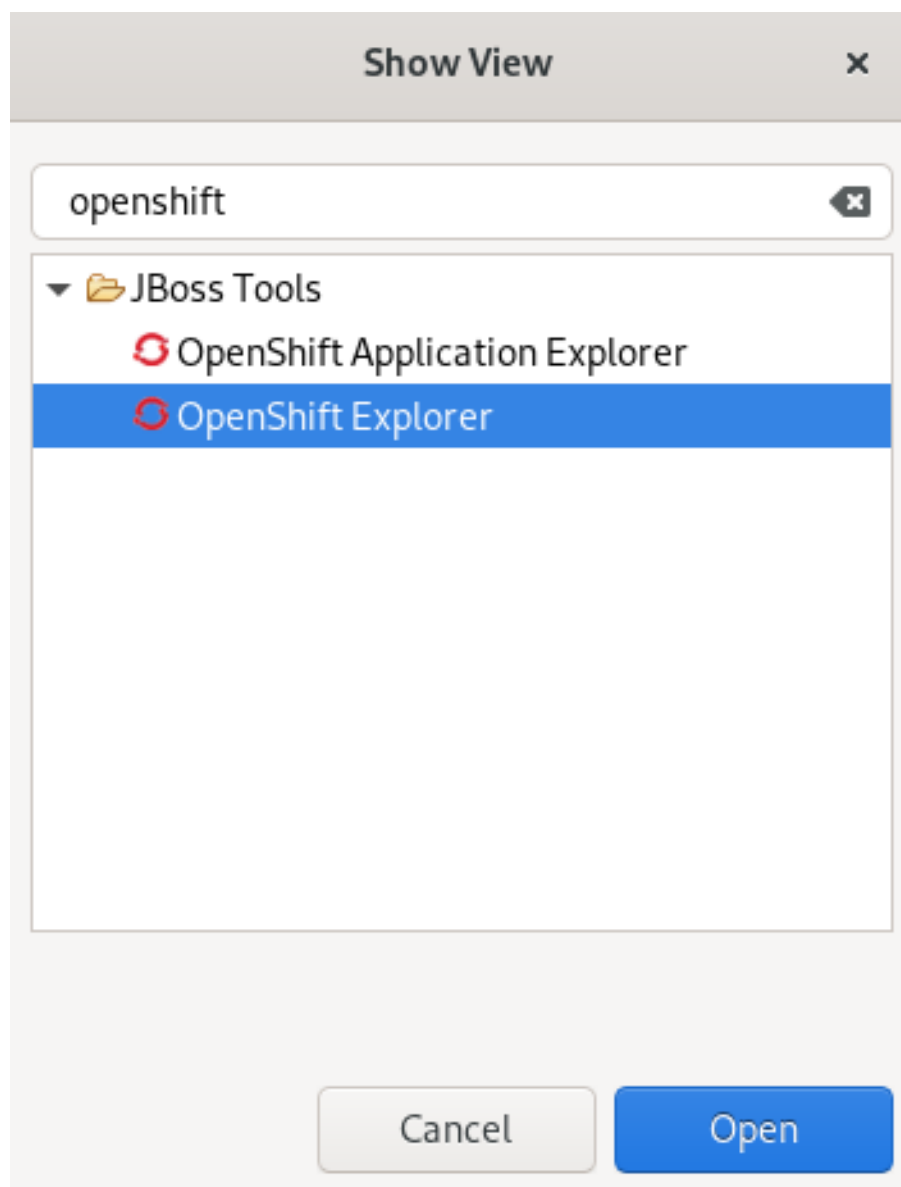
#### 手順

1. CodeReady Studio を起動します。

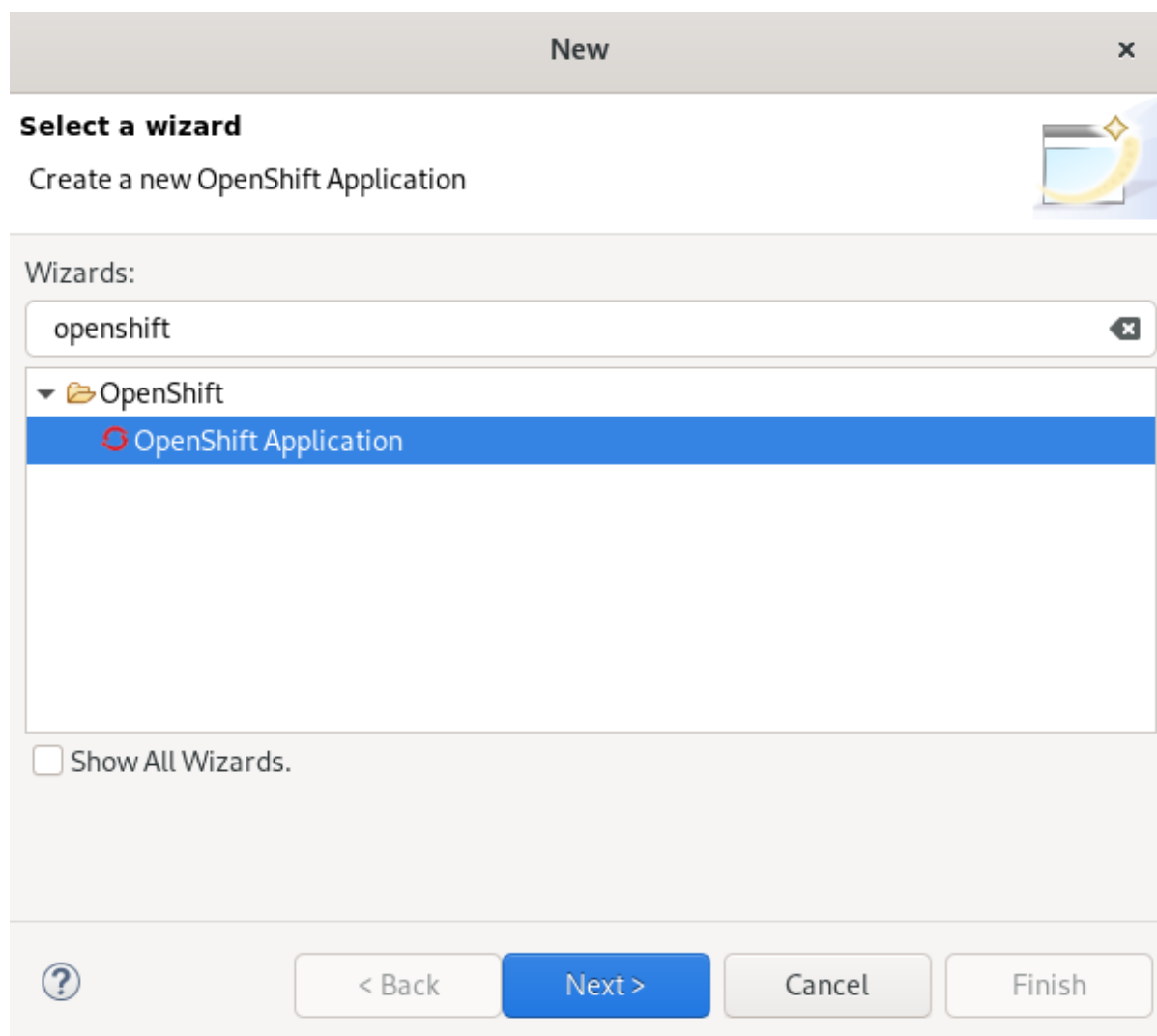
2. CRC サーバーアダプターを起動します。



3. Window → Show View → Other とクリックします。  
Show View ウィンドウが表示されます。



4. 検索フィールドに **OpenShift** と入力します。
5. **OpenShift Explorer** を選択します。
6. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
7. **Ctrl+N** キーを押します。  
**Select a wizard** ウィンドウが表示されます。




8. 検索フィールドに **OpenShift** と入力します。
9. **OpenShift Application** を選択します。
10. **Next** をクリックします。  
**Sign in to OpenShift** ウィンドウが表示されます。

New OpenShift Application

×

Sign in to OpenShift

  
OPENSIFT

⚠️ OpenShift client oc wasn't recognized. You may download and/or configure a different OpenShift client.

Want to try OpenShift online? You can sign up for an account [here](#)

Connection: developer - https://api.crc.testing:6443

Server: https://api.crc.testing:6443 Paste Login Command

Authentication

Protocol: Basic

Username: developer

Password: ●●●●●●●●

☒ Save password (could trigger secure storage login)

Advanced >>

?

< Back

Next >

Cancel


Finish

11. **Next** をクリックします。  
Create OpenShift Project ウィンドウが表示されます。
12. プロジェクトに名前を付けます。
13. **Finish** をクリックします。  
Select template ウィンドウが表示されます。

New OpenShift Application

Select template

Server template choices may be filtered by typing the name of a tag in the text field.

  
OPENSHIFT

OpenShift project: my-openshift-project

New...

Refresh...

Eclipse Project:

Browse...

Server application source

Custom template

dotnet

⚡ dotnet-example (quickstart, dotnet, .net) - openshift

⚡ dotnet-pgsql-persistent (quickstart, dotnet) - openshift

📦 dotnet:2.1 (builder, .net, dotnet, dotnetcore, rh-dotnet21) - openshift

📦 dotnet:3.0 (builder, .net, dotnet, dotnetcore, rh-dotnet30) - openshift

📦 dotnet:3.1 (builder, .net, dotnet, dotnetcore, rh-dotnet31) - openshift

📦 dotnet:latest (builder, .net, dotnet, dotnetcore) - openshift

Details

📦 An example .NET Core application.

Defined Resources...

?

< Back

Next >

Cancel


Finish

14. テンプレートを選択します。

15. **Next** をクリックします。  
**Build Configuration** ウィンドウが表示されます。

10

New OpenShift Application
×

**Build Configuration**


Name:

Git Repository URL:

Git Reference:

Context Directory:

Build Triggers:

- ☒ Configure a webhook build trigger
- ☒ Automatically build a new image when the builder image changes
- ☒ Automatically build a new image when the build configuration changes

Build environment variables (Build and Runtime):

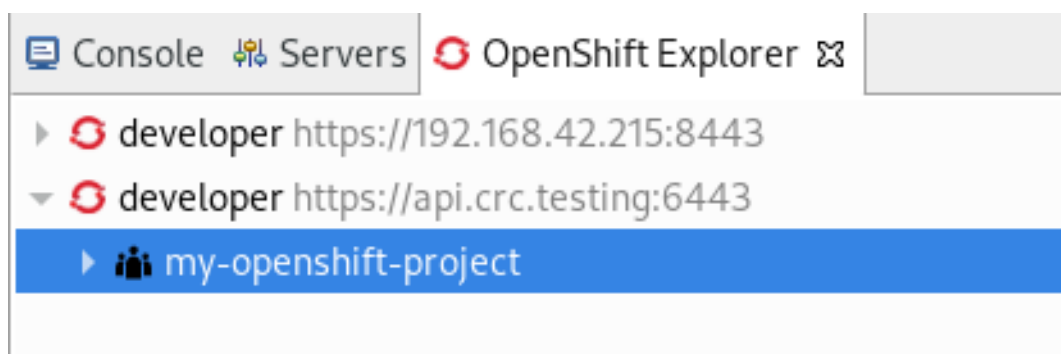
Name	Value

?

16. ビルド設定が正しいことを確認します。

17. **Finish** をクリックします。

新たに作成された OpenShift アプリケーションプロジェクトが **OpenShift Explorer** ビューに表示されます。



## その他のリソース

- OpenShift Container Platform プロジェクトおよびアプリケーションで追加のタスクを実行する方法の詳細は、「[OpenShift を使用したクラウド向けの開発](#)」を参照してください。

## 1.2. CODEREADY STUDIO での RED HAT CONTAINER DEVELOPMENT KIT ツールの使用

Red Hat Container Development Kit (CDK) は、Red Hat Enterprise Linux (RHEL) をベースとした事前にビルドされたコンテナ開発環境です。CDK は、コンテナベースのアプリケーションをすぐに開発できるようにします。CodeReady Studio 内で CDK を簡単に設定できます。

### 1.2.1. CodeReady Studio での Container Development Kit のインストール

CodeReady Studio 内から CDK をインストールする方法を説明します。

#### 前提条件

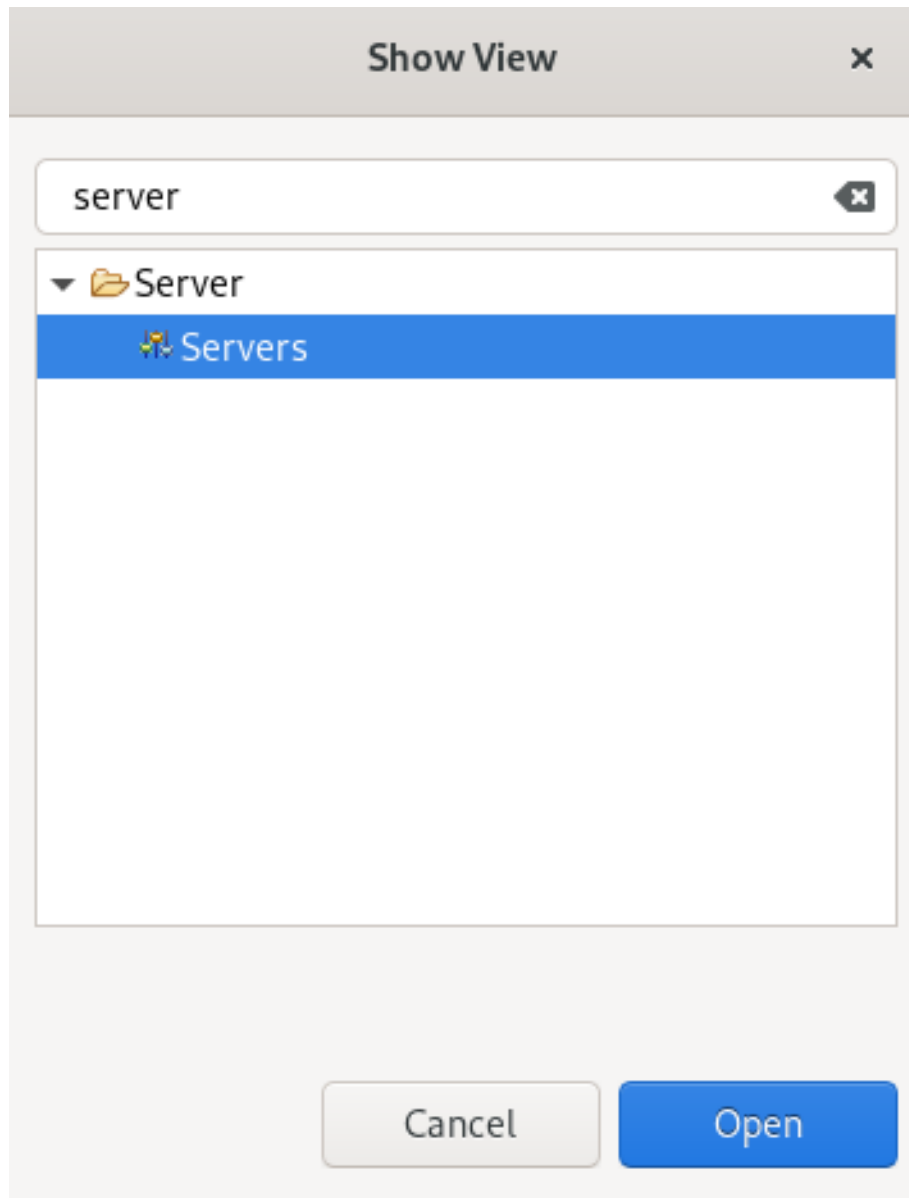
- ハイパーバイザーがシステムにインストールされ、設定されていることを確認します。
  - VirtualBox、Linux KVM/libvirt (Linux)
  - xhyve (macOS)
  - Hyper-V (Windows)
- システムでハードウェアの仮想化が有効になっていることを確認します。  
詳細は『[Setting Up the Virtualization Environment](#)』を参照してください。
- Red Hat Developer アカウントがあることを確認してください。  
新規アカウントを作成する場合は、<https://developers.redhat.com/> にアクセスします。

CDK の詳細は『[Red Hat Container Development Kit Getting Started Guide](#)』を参照してください。

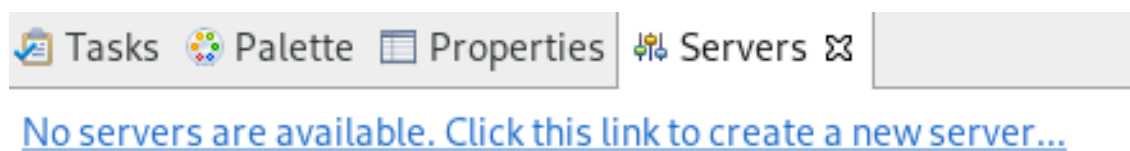
#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。

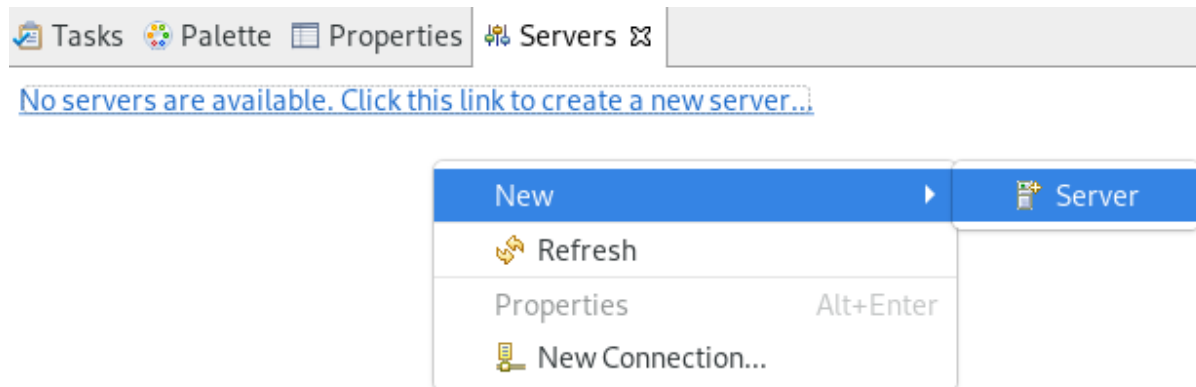




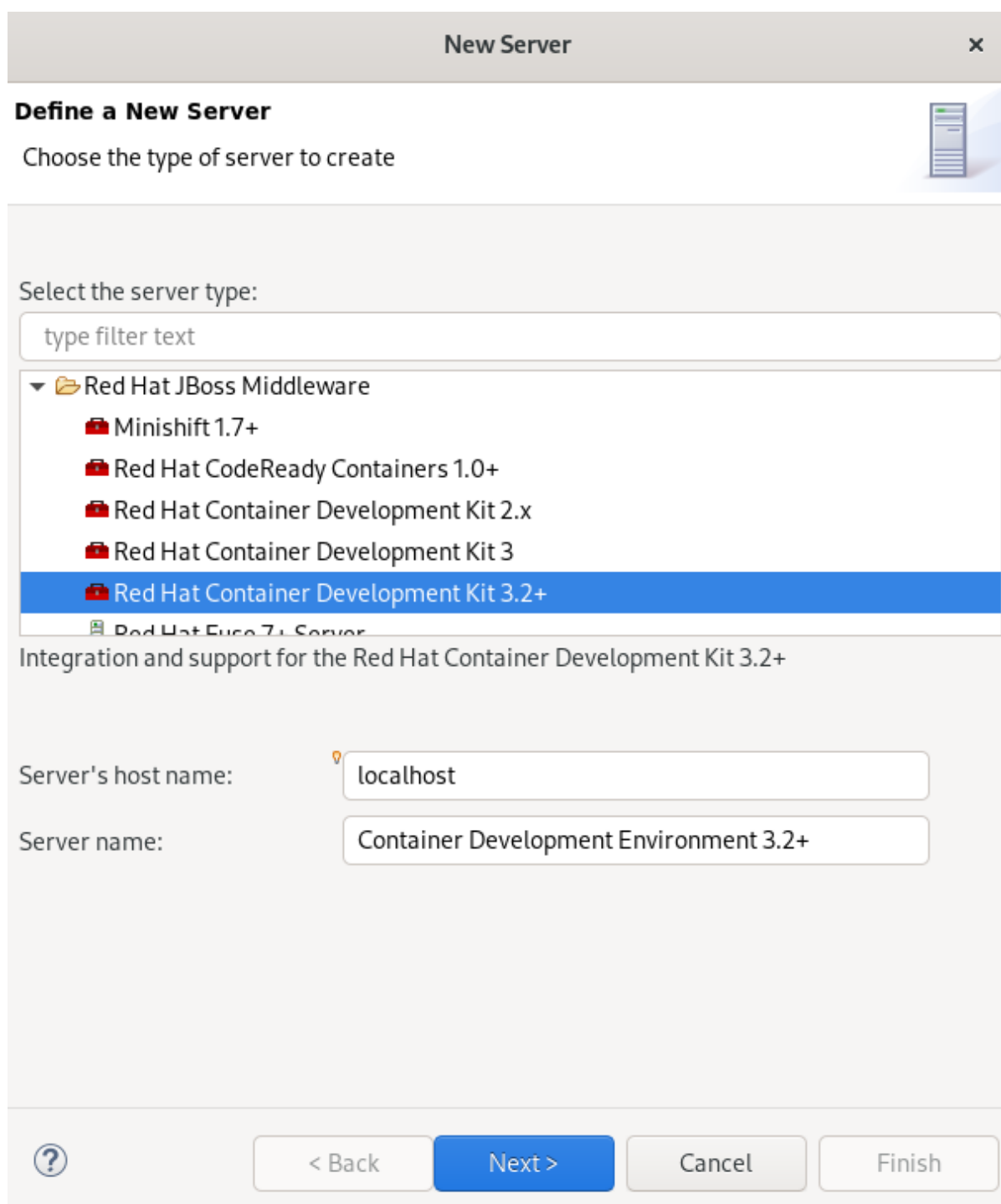
3. 検索フィールドに **Server** と入力します。
4. **Servers** を選択します。
5. **Open** をクリックします。  
**Servers** ビューが表示されます。



6. **Servers** ビューの任意の場所をクリックします。



7. **New** → **Server** とクリックします。  
Define a New Server ウィンドウが表示されます。



8. Red Hat Container Development Kit 3.2+ を選択します。
9. **Next** をクリックします。  
Red Hat Container Development Environment ウィンドウが表示されます。

here if you do not have one already.' Below this are several input fields: 'Domain:' with a dropdown menu showing 'access.redhat.com'; 'Username:' with a dropdown menu showing 'developer@redhat.com', an 'Edit...' button, and an 'Add...' button; 'Hypervisor:' with a dropdown menu showing 'kvm'; 'Minishift Binary:' with an empty text field and a 'Browse...' button; 'Minishift Home:' with a text field containing '/home/ developer / .minishift' and a 'Browse...' button; and 'Minishift Profile:' with a text field containing 'minishift'. A blue link 'Download and install runtime...' is positioned above the 'Minishift Binary' and 'Minishift Home' fields. At the bottom, there is a help icon (question mark in a circle) and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'." data-bbox="154 128 912 566"/>

**New Server** ×

**Red Hat Container Development Environment**

A server adapter representing Red Hat Container Development Kit  
Version 3.2+

Register a Red Hat account [here](#) if you do not have one already.

Domain: access.redhat.com ▼

Username: developer@redhat.com ▼ Edit... Add...

Hypervisor: kvm ▼

[Download and install runtime...](#)

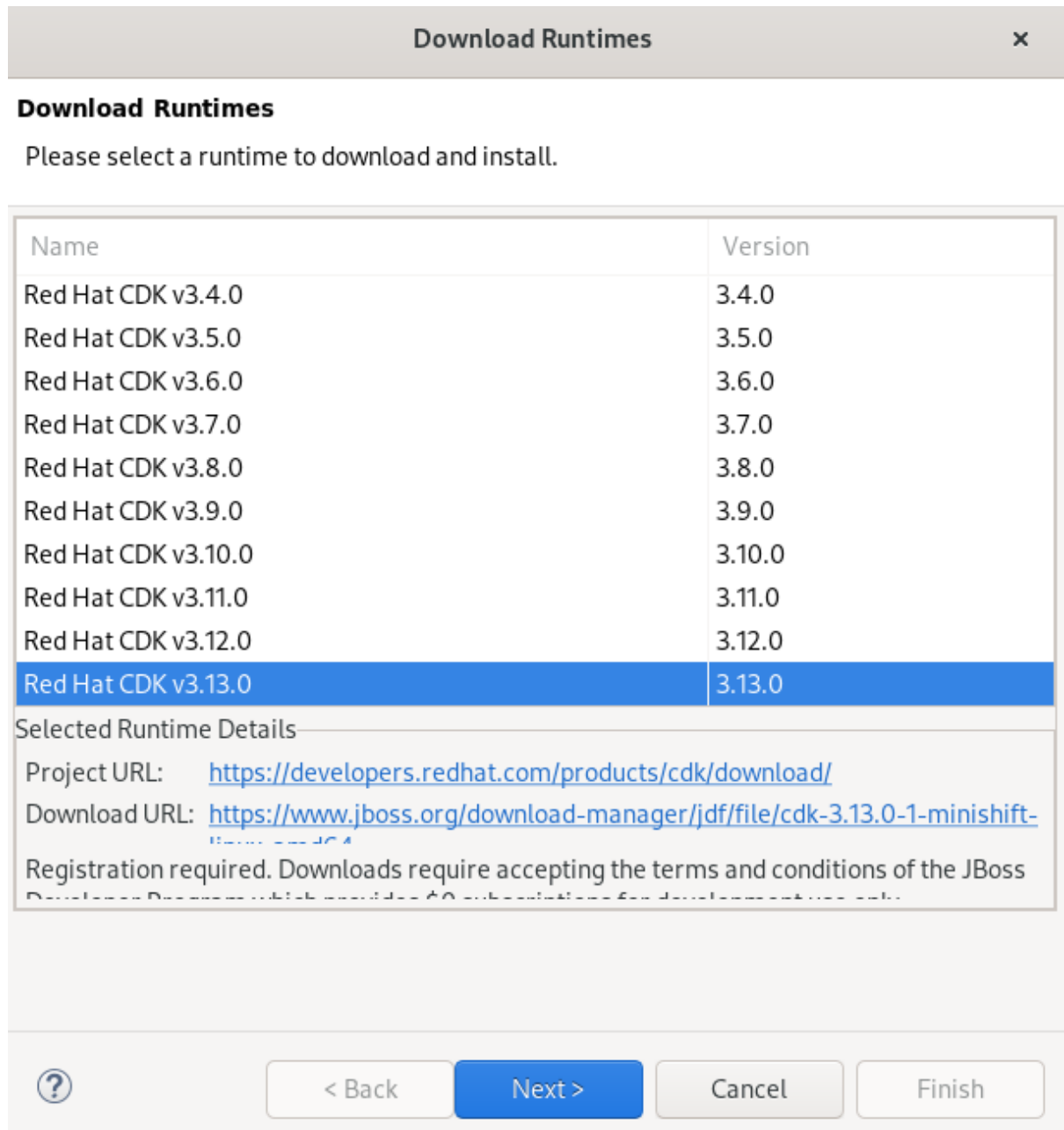
Minishift Binary: Browse...

Minishift Home: /home/ developer / .minishift Browse...

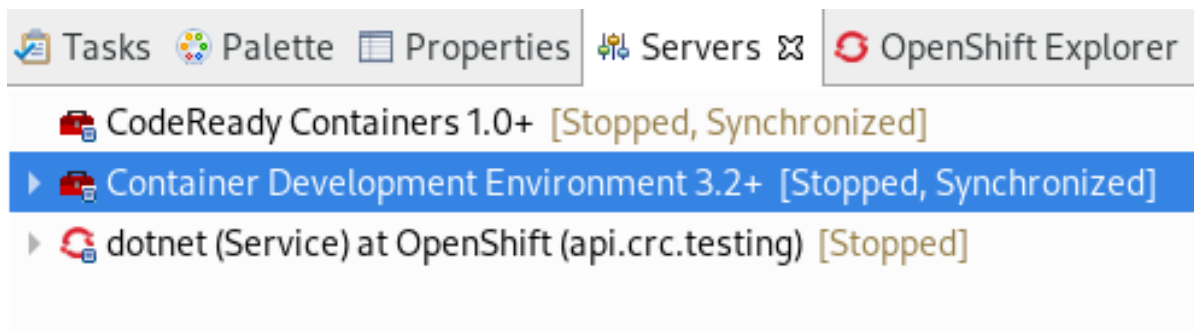
Minishift Profile: minishift

ⓘ < Back Next > Cancel Finish

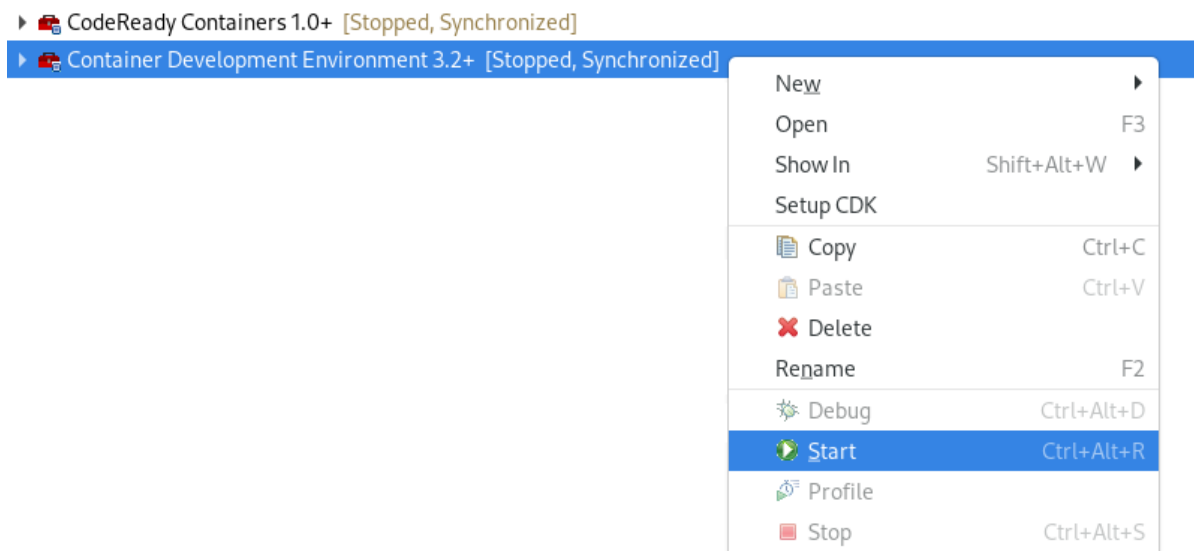
10. **Download and install runtime** をクリックします。  
**Download Runtimes** ウィンドウが表示されます。



11. **Red Hat CDK**バージョンを選択します。
12. **Next** をクリックします。
13. `access.redhat.com` のログインクレデンシャルが正しいことを確認します。
14. **Next** をクリックします。
15. ライセンス契約書を読み、承諾してから **Next** をクリックします。
16. インストールフォルダーを選択し、**Finish** をクリックします。  
ランタイムのダウンロードおよびインストールが完了するまで時間がかかることがあるため注意してください。
17. **Finish** をクリックします。  
新たに作成された **Container Development Environment 3.2+** サーバーが **Servers** ビューに表示されます。



18. CDK server adapter → Start を右クリックします。



### 注記

サーバーアダプターを起動する前に CDK を設定しなかった場合は、**CDK has not been properly initialized!** という警告が表示されます。



画面の指示に従って CDK を初期化します。

## 1.2.2. Docker ツールの使用

IDE で CDK サーバーを起動したら、2 つあるコンテナ開発ワークフローの 1 つに従って Docker ツールを使用できます。

### 1.2.2.1. Dockerfile の作成


#### 前提条件


- CDK サーバーアダプターが設定済みである必要があります。

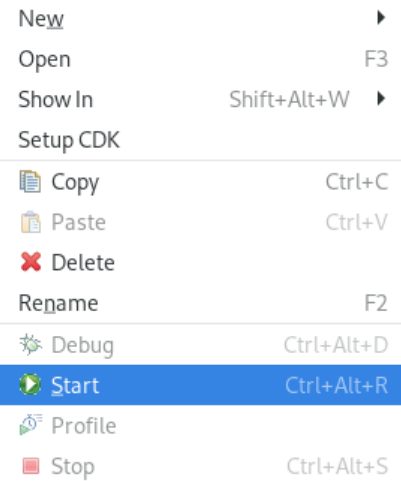
詳細は、「[CodeReady Studio での Container Development Kit のインストール](#)」を参照してください。

## 手順

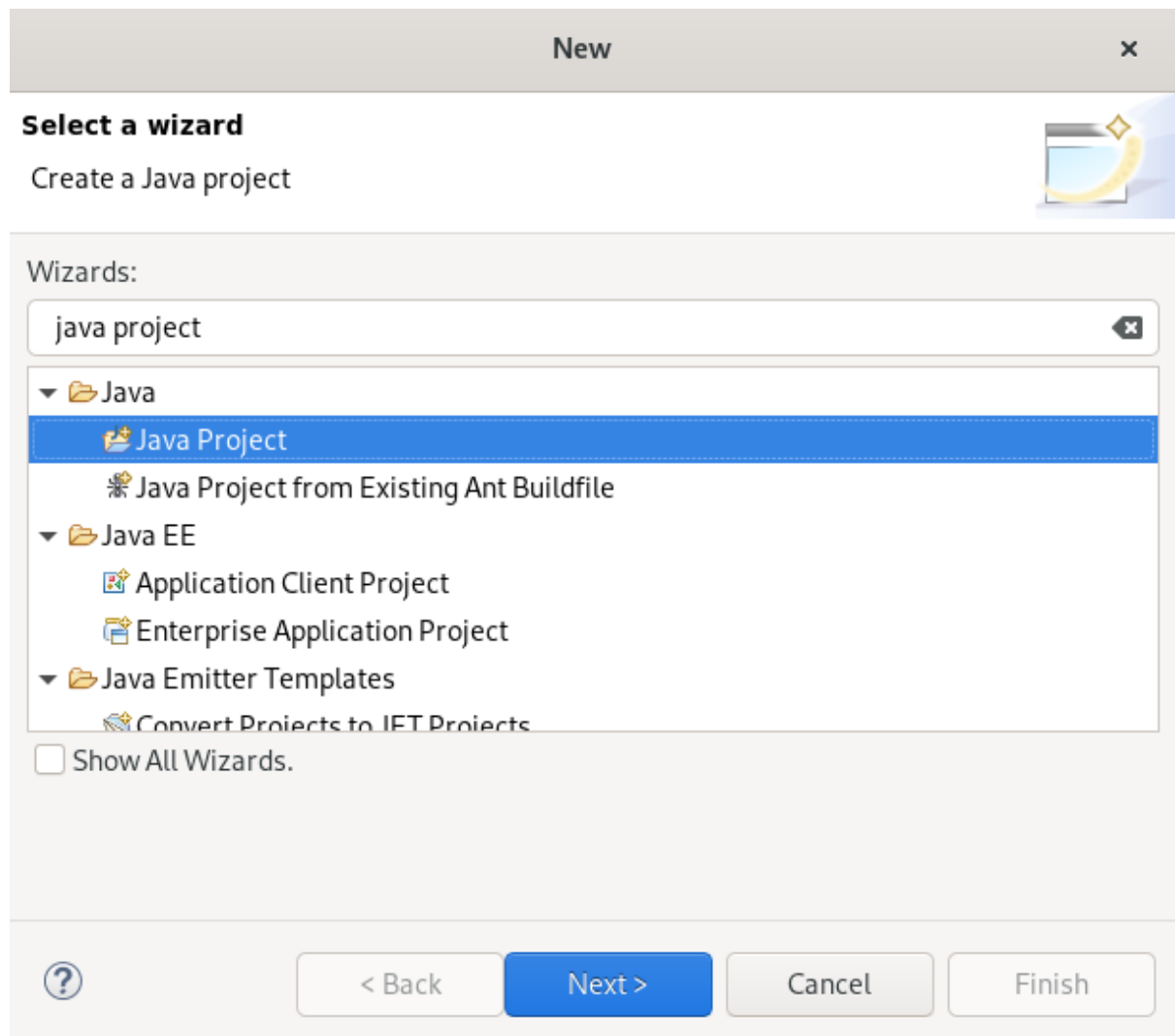
1. CodeReady Studio を起動します。
2. CDK サーバーアダプターを起動します。

▶  CodeReady Containers 1.0+ [Stopped, Synchronized]

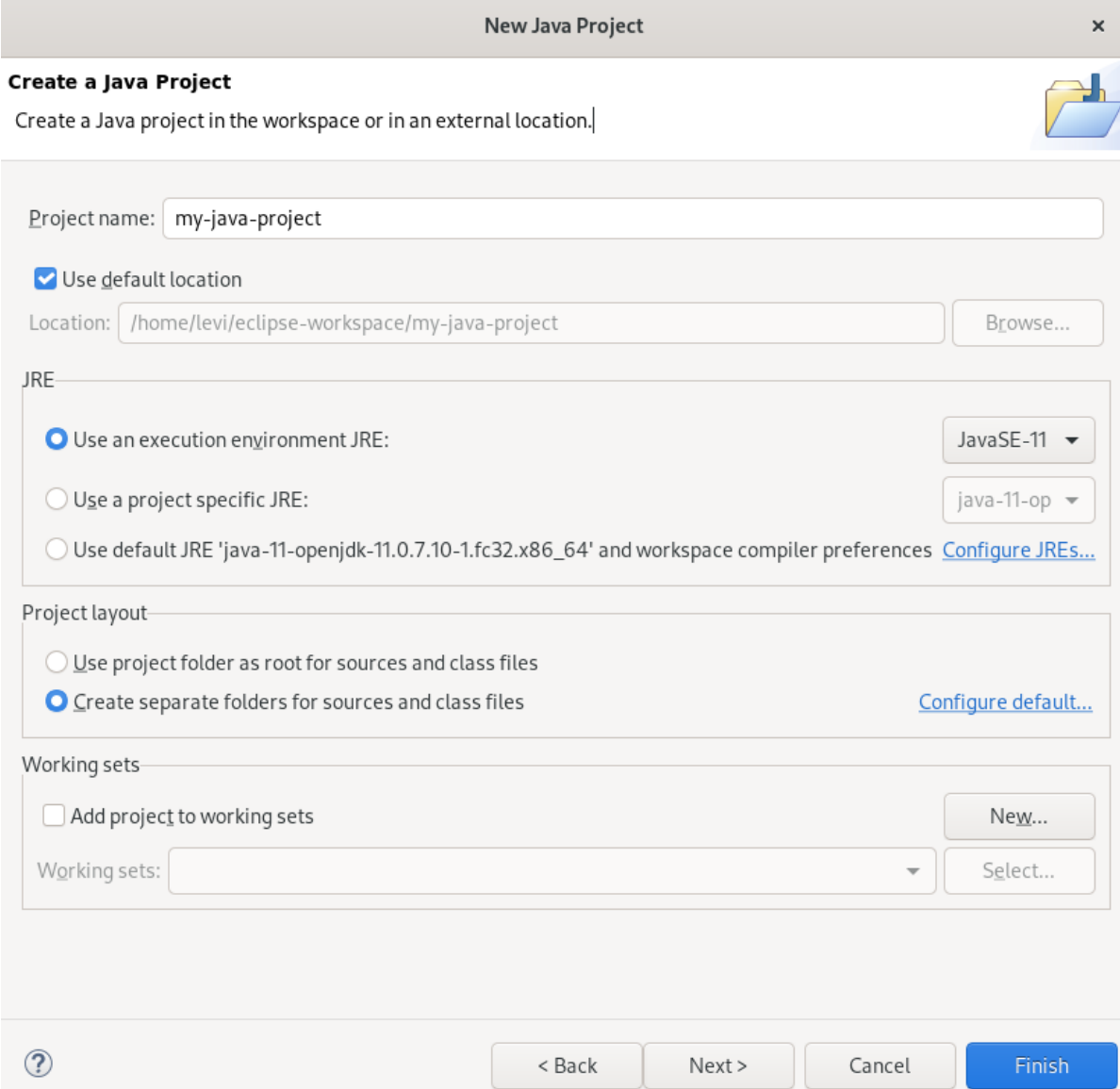
▶  Container Development Environment 3.2+ [Stopped, Synchronized]



3. **Ctrl+N** キーを押します。  
**Select a wizard** ウィンドウが表示されます。



4. 検索フィールドに **Java Project** と入力します。
5. **Java Project** を選択します。
6. **Next** をクリックします。  
New Java Project ウィンドウが表示されます。



**New Java Project**

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: my-java-project

☒ Use default location

Location: /home/levi/eclipse-workspace/my-java-project [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-11

☐ Use a project specific JRE: java-11-op

☐ Use default JRE 'java-11-openjdk-11.0.7.10-1.fc32.x86\_64' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

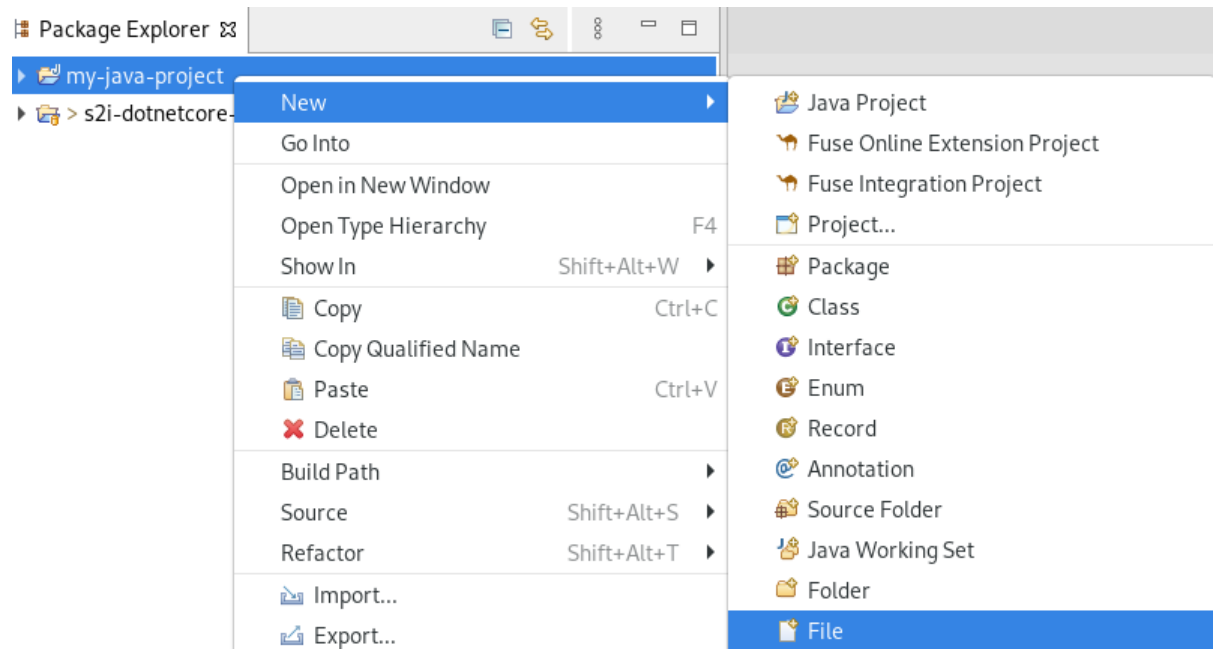
☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

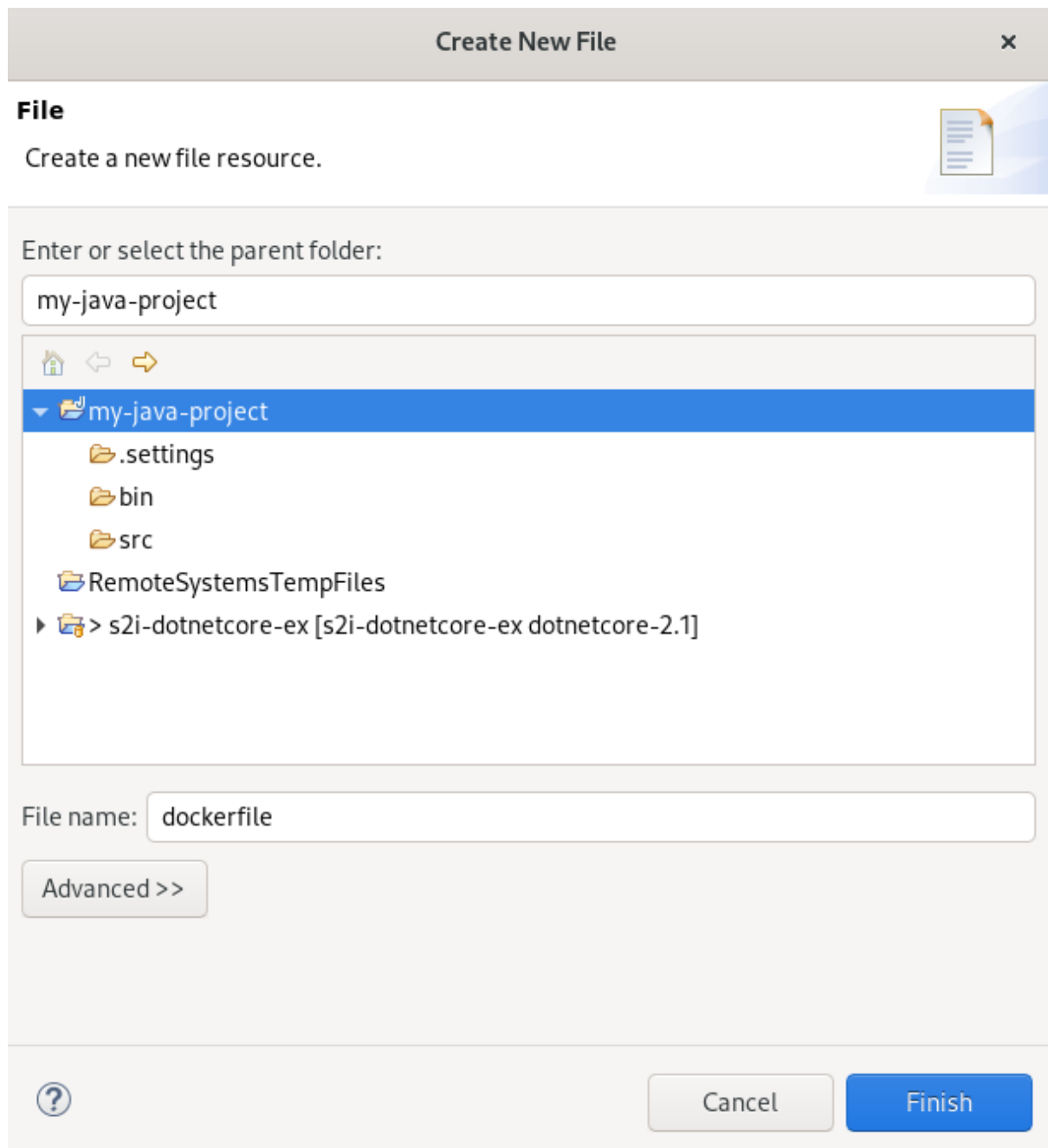
[?](#) [< Back](#) [Next >](#) [Cancel](#) [Finish](#)

7. プロジェクトに名前を付けます。
8. プロジェクトの場所を選択します。
9. **Finish** をクリックします。  
新たに作成された Java プロジェクトが CodeReady Studio ビューに表示されます。
10. **Java project** → **New** → **File** を右クリックします。





Create New File ウィンドウが表示されます。



11. 親フォルダーを選択します。
12. ファイルに名前を付けます。
13. **Finish** をクリックします。  
新たに作成されたファイルが CodeReady Studio エディターに表示されます。
14. 新たに作成されたファイルに以下の内容を貼り付けます。

```
# Use latest jboss/base-jdk:8 image as the base
FROM jboss/base-jdk:8

# Set the WILDFLY_VERSION env variable
ENV WILDFLY_VERSION 10.1.0.Final
ENV WILDFLY_SHA1 9ee3c0255e2e6007d502223916cefad2a1a5e333
ENV JBOSS_HOME /opt/jboss/wildfly

USER root
```

```
# Add the WildFly distribution to /opt, and make wildfly the owner of the extracted tar content
# Make sure the distribution is available from a well-known place
RUN cd $HOME \
    && curl -O https://download.jboss.org/wildfly/$WILDFLY_VERSION/wildfly-
$WILDFLY_VERSION.tar.gz \
    && sha1sum wildfly-$WILDFLY_VERSION.tar.gz | grep $WILDFLY_SHA1 \
    && tar xf wildfly-$WILDFLY_VERSION.tar.gz \
    && mv $HOME/wildfly-$WILDFLY_VERSION $JBOSS_HOME \
    && rm wildfly-$WILDFLY_VERSION.tar.gz \
    && chown -R jboss:0 ${JBOSS_HOME} \
    && chmod -R g+rw ${JBOSS_HOME}

# Ensure signals are forwarded to the JVM process correctly for graceful shutdown
ENV LAUNCH_JBOSS_IN_BACKGROUND true

USER jboss

# Expose the ports we're interested in
EXPOSE 8080

# Set the default command to run on boot
# This will boot WildFly in the standalone mode and bind to all interface
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0"]
```

15. **Ctrl+S** キーを押して変更を保存します。

## その他のリソース

- Dockerfile の詳細は、「[Dockerfile reference](#)」を参照してください。

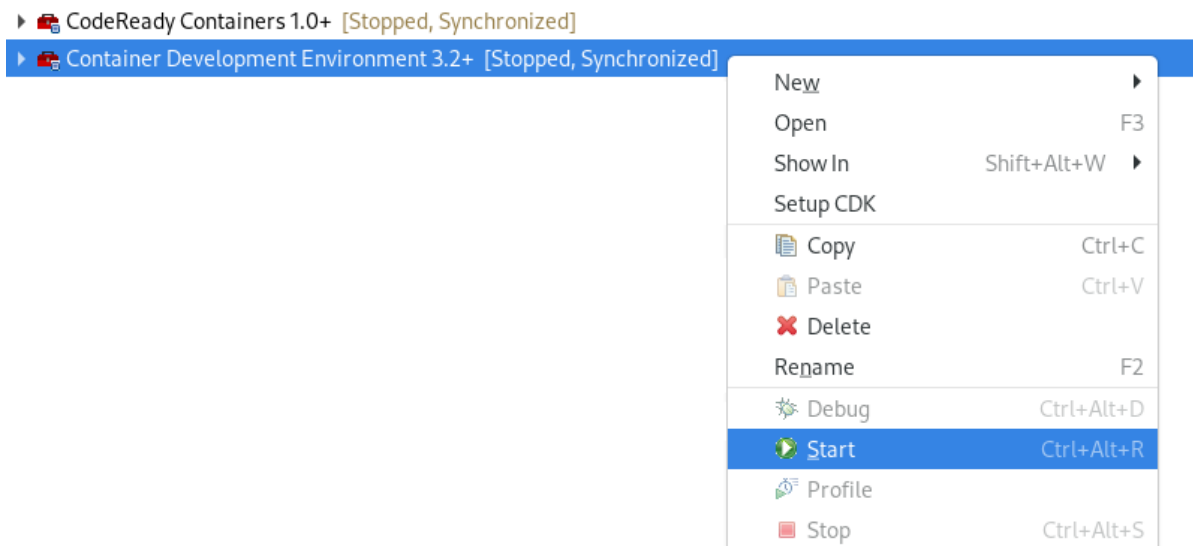
### 1.2.2.2. Container Development Environment を使用した Docker イメージのビルド

#### 前提条件

- CDK サーバーアダプターが設定済みである必要があります。  
詳細は、「[CodeReady Studio での Container Development Kit のインストール](#)」を参照してください。
- Java プロジェクトと Dockerfile を作成します。  
詳細は、「[Dockerfile の作成](#)」を参照してください。

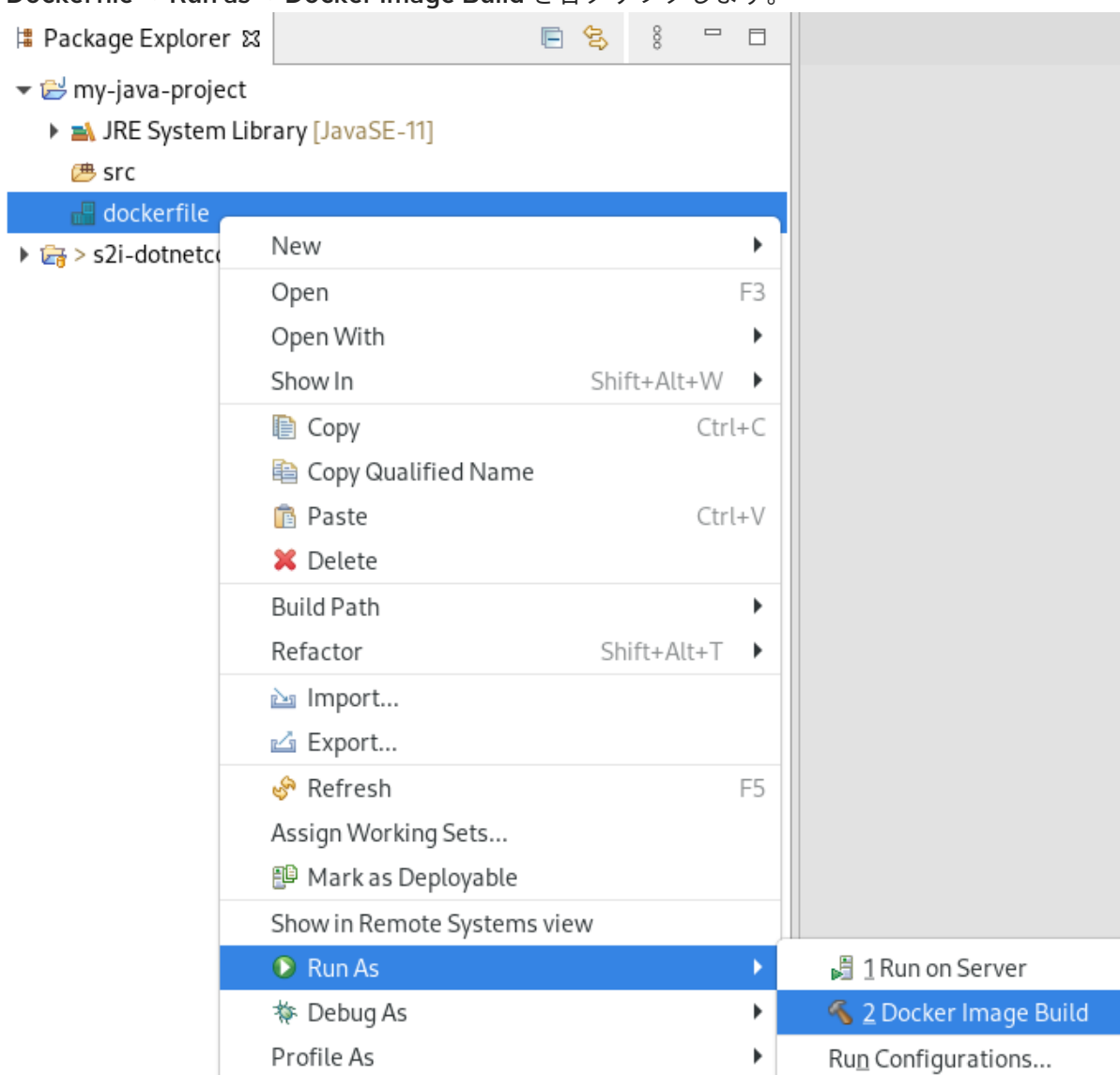
#### 手順

1. CodeReady Studio を起動します。
2. CDK サーバーアダプターを起動します。

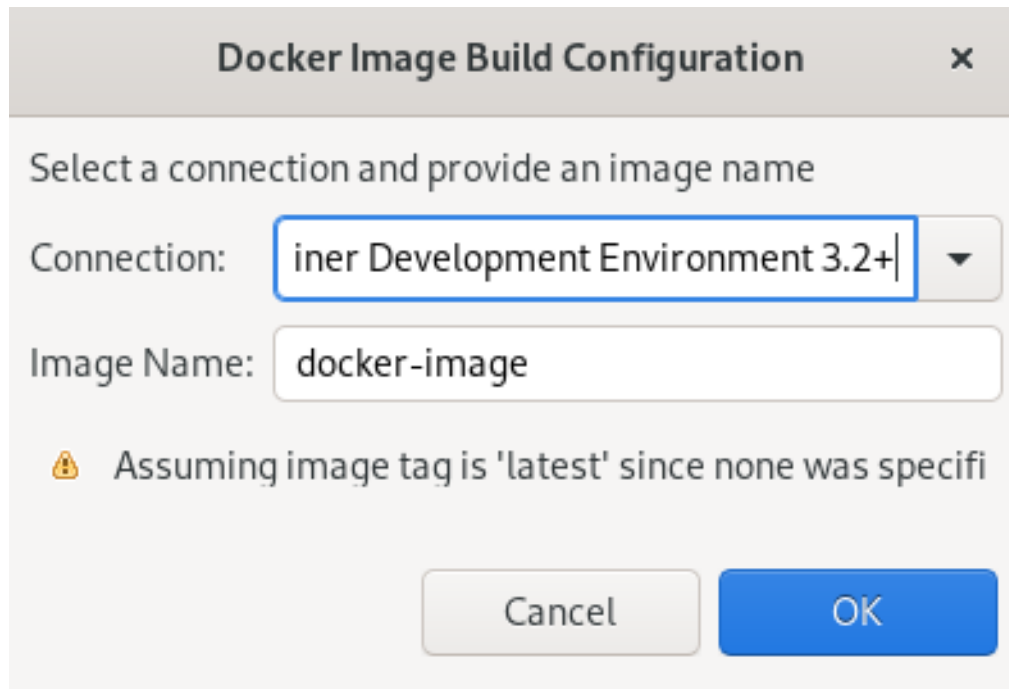


3. Java プロジェクトを展開します。

4. **Dockerfile** → **Run as** → **Docker Image Build** を右クリックします。



Docker Image Build Configuration ウィンドウが表示されます。



5. コネクションとして Container Development Environment サーバーアダプターを選択します。
6. イメージに名前を付けます。
7. **OK** をクリックします。

Console ビューが表示され、docker イメージのビルドプロセスが表示されます。

#### 1.2.2.3. その他のリソース

- Docker Tooling の基本に関する詳細は、「[Docker ツールの使用](#)」を参照してください。

#### 1.2.3. OpenShift Container Platform ツールの使用

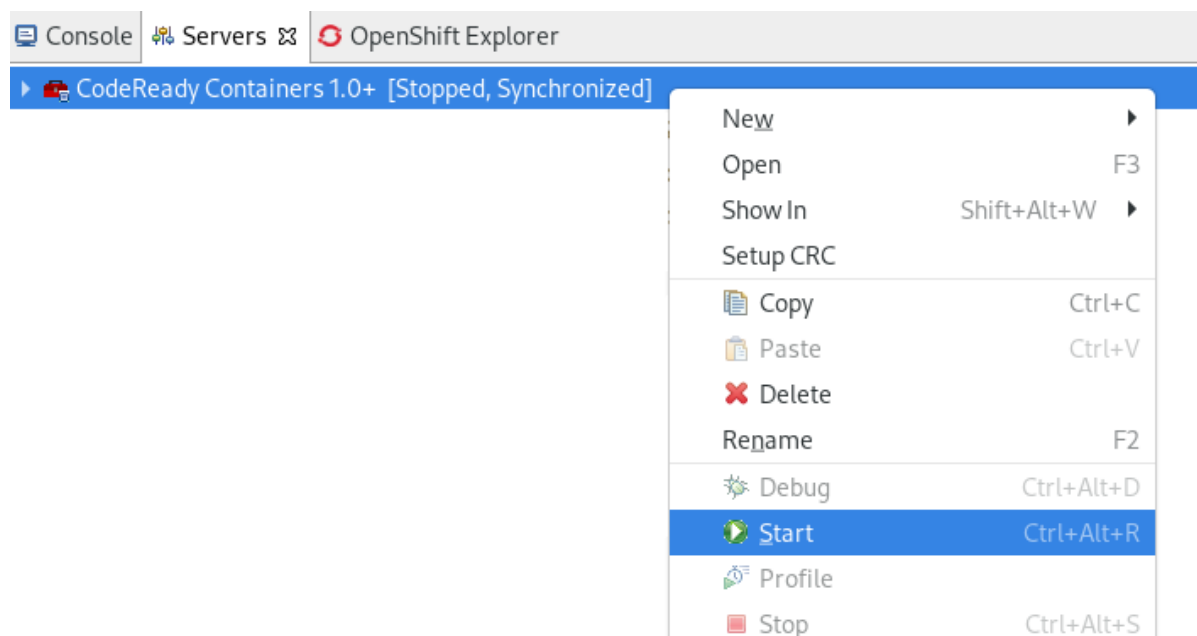
CodeReady Studio で OpenShift コンテナを使用する方法を説明します。

##### 前提条件

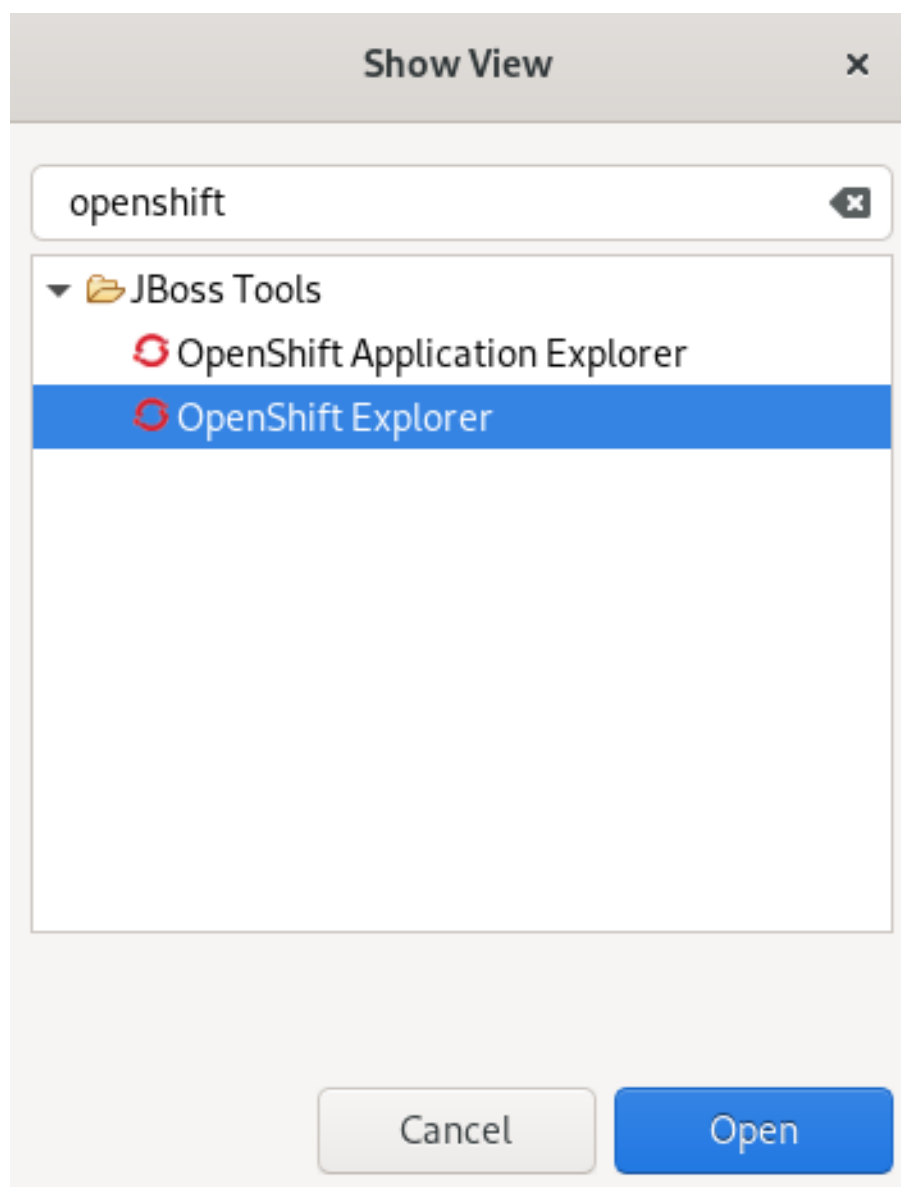
- CRC サーバーアダプターが設定済みである必要があります。  
詳細は、「[CodeReady Studio での Red Hat CodeReady コンテナのダウンロードおよびインストール](#)」を参照してください。

##### 手順

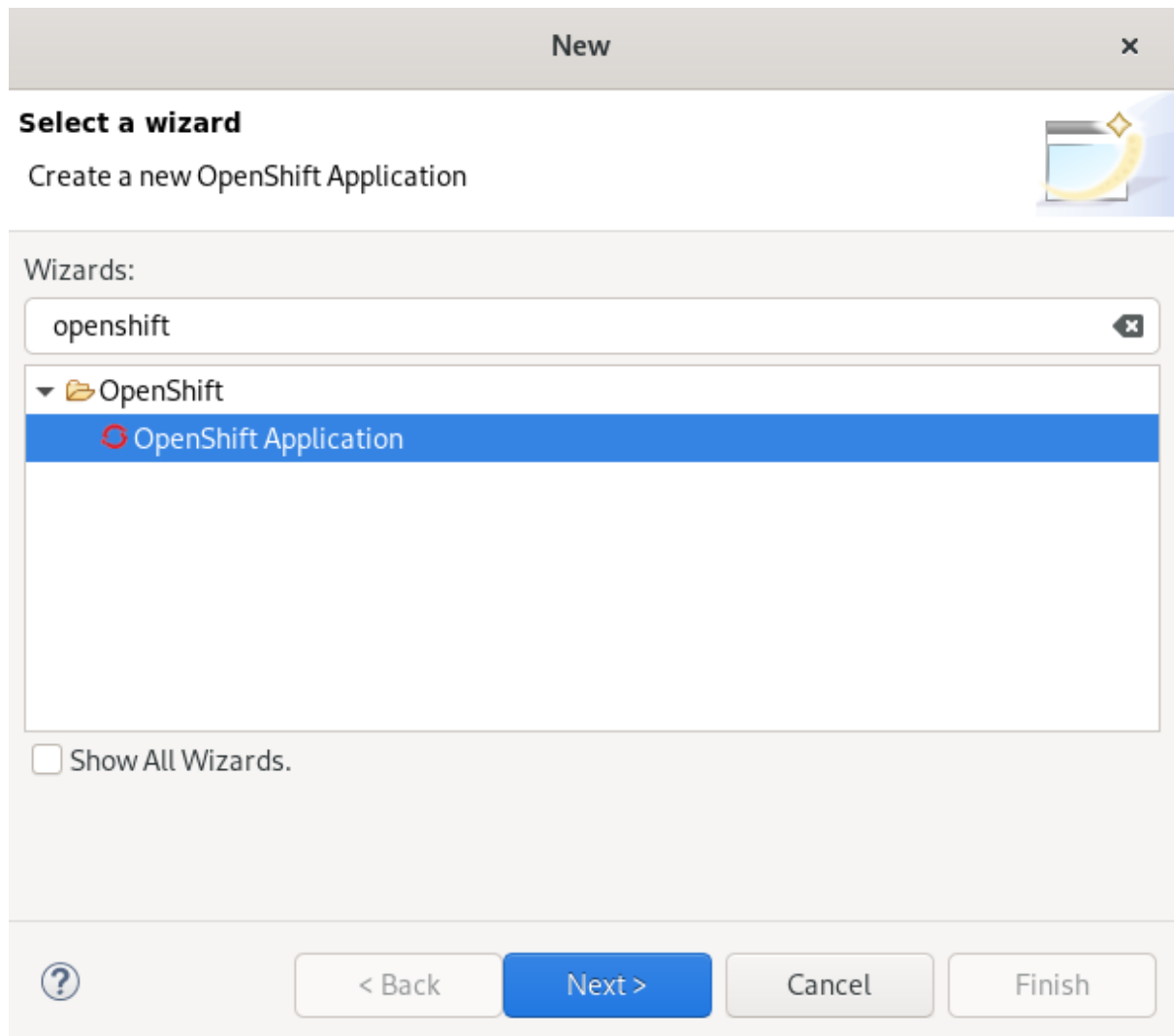
1. CodeReady Studio を起動します。
2. CRC サーバーアダプターを起動します。



3. **Window → Show View → Other** とクリックします。  
Show View ウィンドウが表示されます。




4. 検索フィールドに **OpenShift** と入力します。
5. **OpenShift Explorer** を選択します。
6. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
7. **Ctrl+N** キーを押します。  
**Select a wizard** ウィンドウが表示されます。




8. 検索フィールドに **OpenShift** と入力します。
9. **OpenShift Application** を選択します。
10. **Next** をクリックします。  
**Sign in to OpenShift** ウィンドウが表示されます。

New OpenShift Application

Sign in to OpenShift

  
OPENSIFT

 OpenShift client oc wasn't recognized. You may download and/or configure a different OpenShift client.

Want to try OpenShift online? You can sign up for an account [here](#)

Connection: developer - https://api.crc.testing:6443

Server: https://api.crc.testing:6443 Paste Login Command

Authentication


Protocol: Basic

Username: developer

Password: ●●●●●●●●

☒ Save password (could trigger secure storage login)

Advanced >>



< Back

Next >

Cancel

Finish

11. **Next** をクリックします。  
Create OpenShift Project ウィンドウが表示されます。
12. プロジェクトに名前を付けます。
13. **Finish** をクリックします。  
Select template ウィンドウが表示されます。


28



New OpenShift Application

Select template

Server template choices may be filtered by typing the name of a tag in the text field.

  
OPENSHIFT

OpenShift project: my-openshift-project

New...

Refresh...

Eclipse Project:

Browse...

Server application source

Custom template

dotnet

⚡ dotnet-example (quickstart, dotnet, .net) - openshift

⚡ dotnet-pgsql-persistent (quickstart, dotnet) - openshift

📦 dotnet:2.1 (builder, .net, dotnet, dotnetcore, rh-dotnet21) - openshift

📦 dotnet:3.0 (builder, .net, dotnet, dotnetcore, rh-dotnet30) - openshift

📦 dotnet:3.1 (builder, .net, dotnet, dotnetcore, rh-dotnet31) - openshift

📦 dotnet:latest (builder, .net, dotnet, dotnetcore) - openshift

Details

📦 An example .NET Core application.

Defined Resources...

?

< Back

Next >

Cancel

Finish


14. テンプレートを選択します。
15. **Next** をクリックします。  
**Build Configuration** ウィンドウが表示されます。

29

New OpenShift Application

×

Build Configuration

  
OPENSHIFT

Name:

dotnet

Git Repository URL:

https://github.com/redhat-developer/s2i-dotnetcore-ex.git

Git Reference:

dotnetcore-2.1

Context Directory:

app

Build Triggers:

☒ Configure a webhook build trigger

☒ Automatically build a new image when the builder image changes

☒ Automatically build a new image when the build configuration changes

Build environment variables (Build and Runtime):

Name	Value
------	-------

Add...

Edit...

Reset

Reset All

Remove

?

< Back

Next >

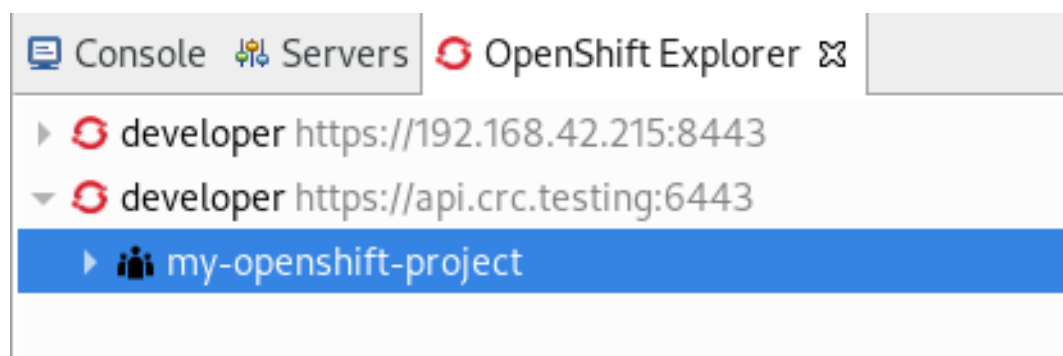
Cancel

Finish

16. ビルド設定が正しいことを確認します。

17. **Finish** をクリックします。

新たに作成された OpenShift アプリケーションプロジェクトが **OpenShift Explorer** ビューに表示されます。



## その他のリソース

- OpenShift Container Platform プロジェクトおよびアプリケーションで追加のタスクを実行する方法の詳細は、「[OpenShift を使用したクラウド向けの開発](#)」を参照してください。

### 1.2.4. その他のリソース

- OpenShift Container Platform ツールでタスクを実行する方法の詳細は、「[OpenShift を使用したクラウド向けの開発](#)」を参照してください。
- CodeReady Studio で OpenShift を使用方法の詳細は、「[CodeReady Studio での OpenShift の基本](#)」を参照してください。

## 第2章 OPENSIFT を使用したクラウド向けの開発

### 2.1. CODEREADY STUDIO での OPENSIFT CONTAINER PLATFORM アプリケーションの作成

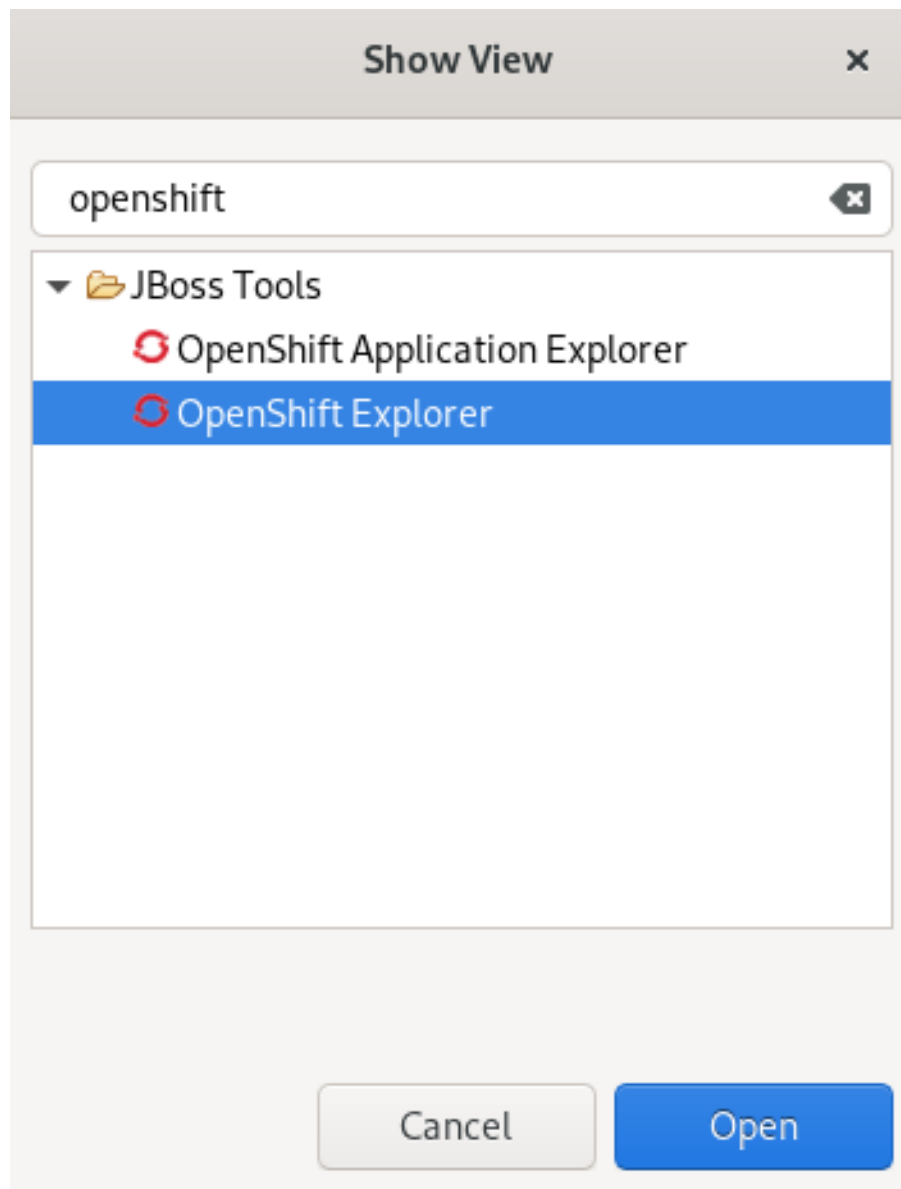
OpenShift Container Platform ツールを使用すると、OpenShift Container Platform アプリケーションを作成、インポート、および変更できます。

#### 2.1.1. 新規 OpenShift Container Platform コネクションの作成

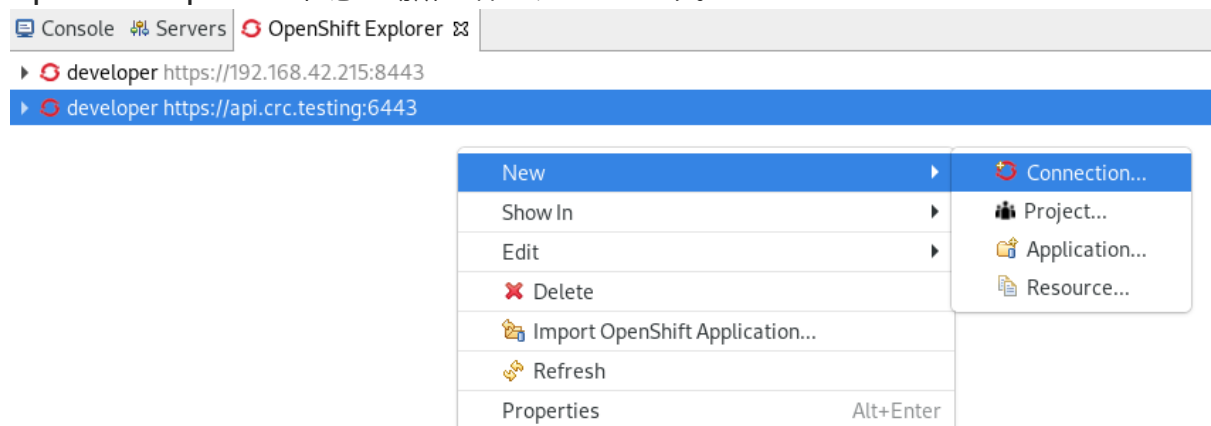
IDE で OpenShift ツールを使用するには、CodeReady Studio の **OpenShift Explorer** ビューで OpenShift コネクションを作成する必要があります。OpenShift コネクションは、IDE を OpenShift インスタンス (CDK、OpenShift Online、Kubernetes、minishift をベースとした) に接続します。コネクションは **OpenShift Explorer** ビューに表示されます。IDE に複数の OpenShift コネクションを設定できます。

#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. **OpenShift Explorer** の任意の場所を右クリックします。



7. **New** → **Connection** とクリックします。  
**Sign in to OpenShift** ウィンドウが表示されます。

**New OpenShift Connection** [X]

**Sign in to OpenShift**  
Please sign in to your OpenShift server.

Want to try OpenShift online? You can sign up for an account [here](#)

Connection: <New Connection> ▼

Server:  ▼

Authentication

Protocol:  ▼

Enter a token or [retrieve](#) a new one.

Token

☐ Save token (could trigger secure storage login)

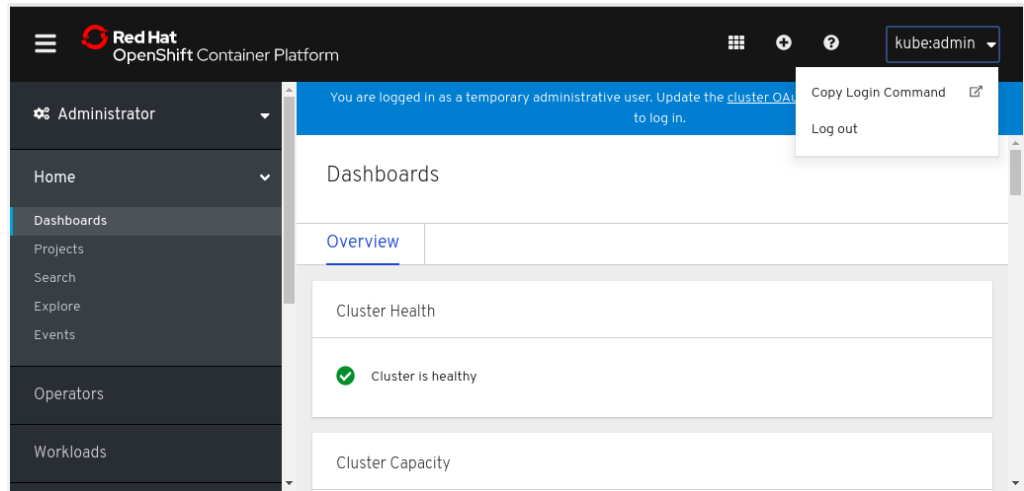
[?]

8. **Server** フィールドに OpenShift サーバーの URL を貼り付けます。
9. トークンまたはログインクレデンシャルを使用して認証します。

## 注記

または、ログインコマンドを OpenShift Container Platform Web UI からコピーすることもできます。

ログインクレデンシャルを取得するには、**drop-down menu in the top right corner → Copy Login Command** とクリックします。



10. **Finish** をクリックします。

新たに追加されたコネクションが **OpenShift Explorer** ビューに表示されます。

### 2.1.2. 新規 OpenShift Container Platform プロジェクトの作成

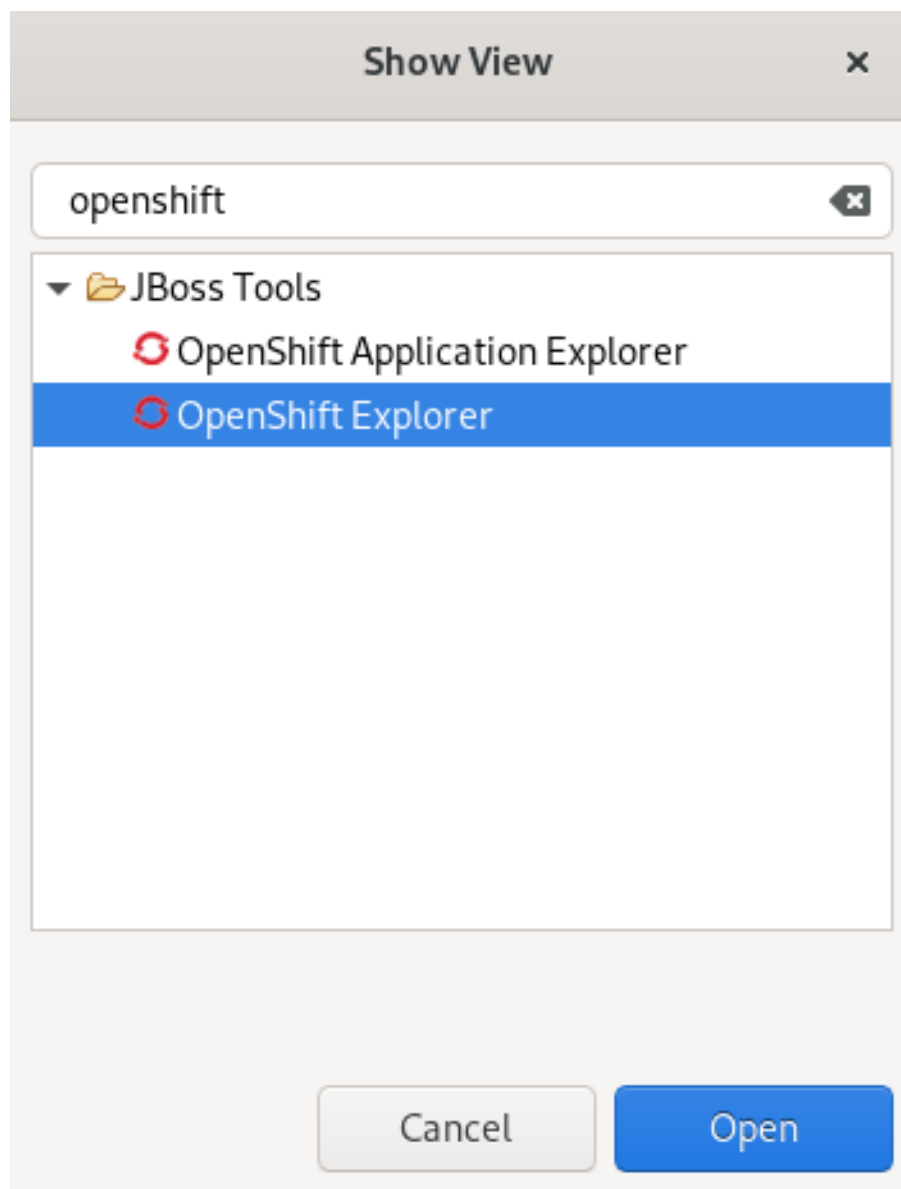
通常ユーザーのリソースへのアクセスを一元管理するには、追加のアノテーションを持つ namespace であるプロジェクトを作成する必要があります。

#### 前提条件

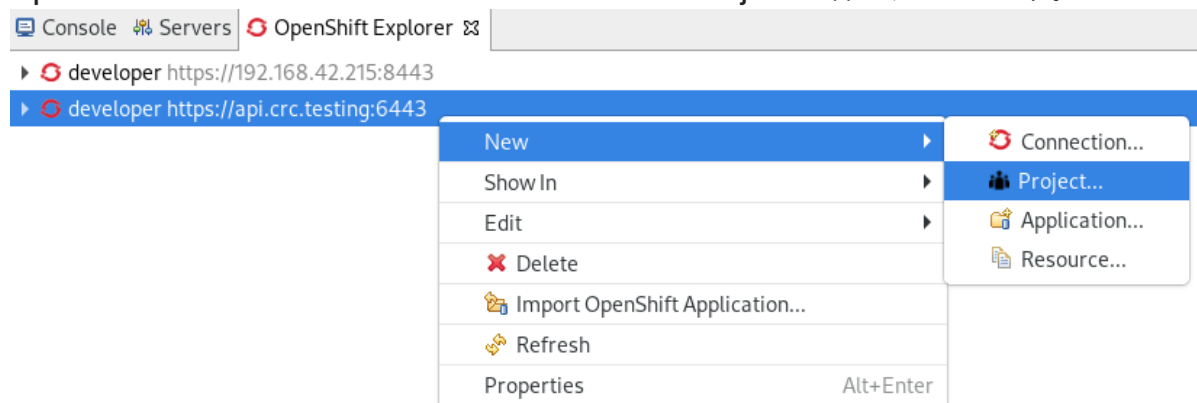
- OpenShift Container Platform コネクション。  
新たに OpenShift Container Platform コネクションを作成する方法の詳細は、「[新規 OpenShift Container Platform コネクションの作成](#)」を参照してください。

#### 手順

1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. **OpenShift Container Platform connection** → **New** → **Project** を右クリックします。



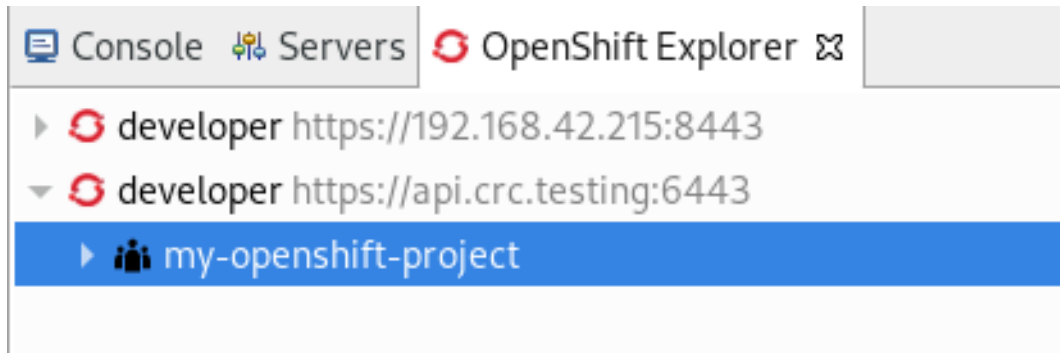
**New OpenShift Project** ウィンドウが表示されます。



7. プロジェクトに名前を付けます。

8. **Finish** をクリックします。

新たに作成された OpenShift プロジェクトが **OpenShift Explorer** ビューに表示されます。



### 2.1.3. 新規 OpenShift Container Platform アプリケーションの作成

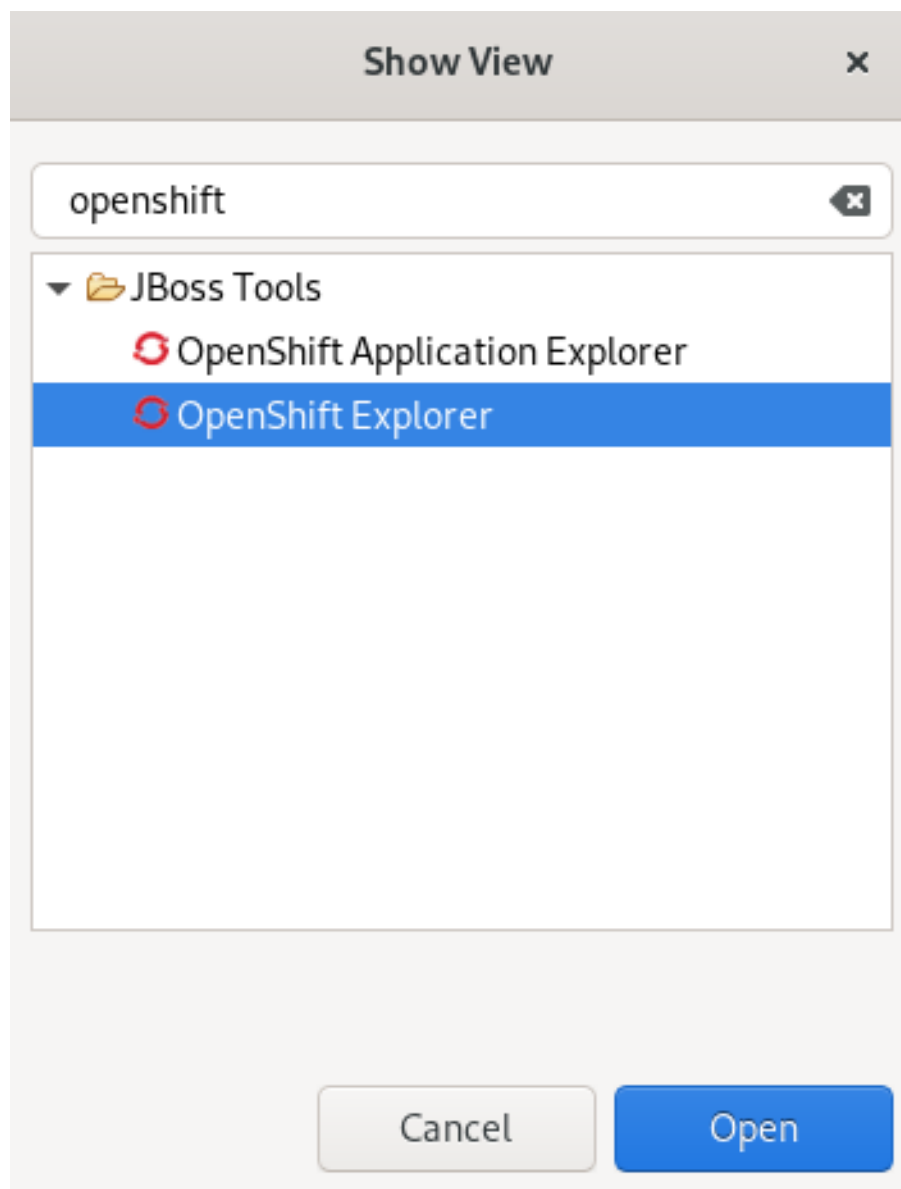
IDE の **OpenShift Application** ウィザードを使用して、デフォルトまたはカスタムテンプレートから OpenShift Container Platform アプリケーションを作成できます。

#### 前提条件

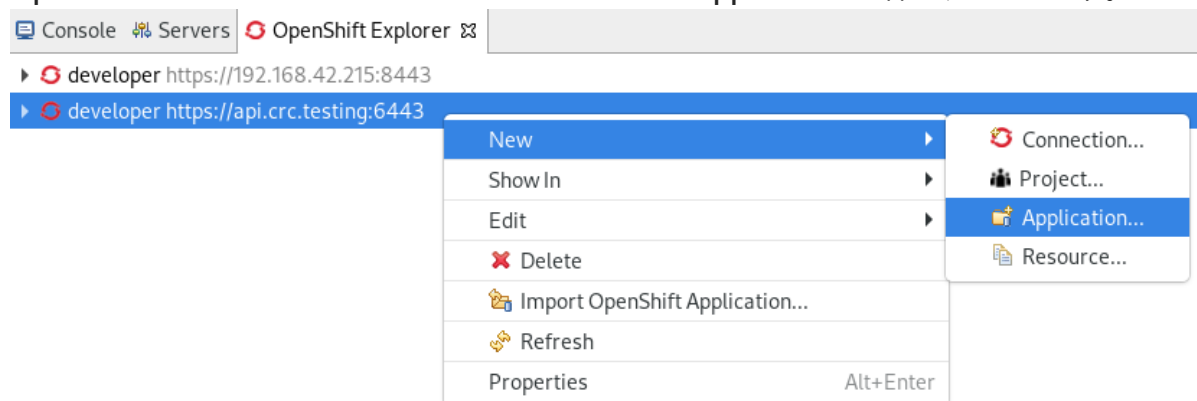
- OpenShift Container Platform コネクション。  
新たに OpenShift Container Platform コネクションを作成する方法の詳細は、[「新規 OpenShift Container Platform コネクションの作成」](#) を参照してください。
- OpenShift Container Platform プロジェクト。  
新たに OpenShift Container Platform プロジェクトを作成する方法の詳細は、[「新規 OpenShift Container Platform プロジェクトの作成」](#) を参照してください。

#### 手順

1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. **OpenShift Container Platform connection** → **New** → **Application** を右クリックします。




Select template ウィンドウが表示されます。

New OpenShift Application

Select template

Server template choices may be filtered by typing the name of a tag in the text field.

  
OPENSIFT

OpenShift project: my-openshift-project

New...

Refresh...

Eclipse Project:

Browse...

Server application source

Custom template

dotnet

⚡ dotnet-example (quickstart, dotnet, .net) - openshift

⚡ dotnet-pgsql-persistent (quickstart, dotnet) - openshift

📦 dotnet:2.1 (builder, .net, dotnet, dotnetcore, rh-dotnet21) - openshift

📦 dotnet:3.0 (builder, .net, dotnet, dotnetcore, rh-dotnet30) - openshift

📦 dotnet:3.1 (builder, .net, dotnet, dotnetcore, rh-dotnet31) - openshift

📦 dotnet:latest (builder, .net, dotnet, dotnetcore) - openshift

Details

📦 An example .NET Core application.

Defined Resources...

?

< Back

Next >

Cancel

Finish


7. テンプレートを選択します。
8. **Next** をクリックします。  
Template Parameters ウィンドウが表示されます。
9. **Next** をクリックします。  
Resource Labels ウィンドウが表示されます。

39

New OpenShift Application

Resource Labels

Add or edit the labels to be added to each resource. Labels are used to organize, group, or select objects and resources, such as pods and

  
OPENSHIFT


Labels

Key	Value

Add...

Edit...

Remove...



< Back

Next >


Cancel

Finish

10. **Add** をクリックしてラベルを追加します。
11. **Finish** をクリックします。  
**Create Application Summary** ウィンドウが表示されます。

Create Application Summary

×

  
OPENSIFT

Results of creating the resources from the dotnet-example template.

New Resources Created:

✔Route - dotnet-example

✔Service - dotnet-example

✔ImageStream - dotnet-example

✔BuildConfig - dotnet-example

✔DeploymentConfig - dotnet-example

▼ Resource Details

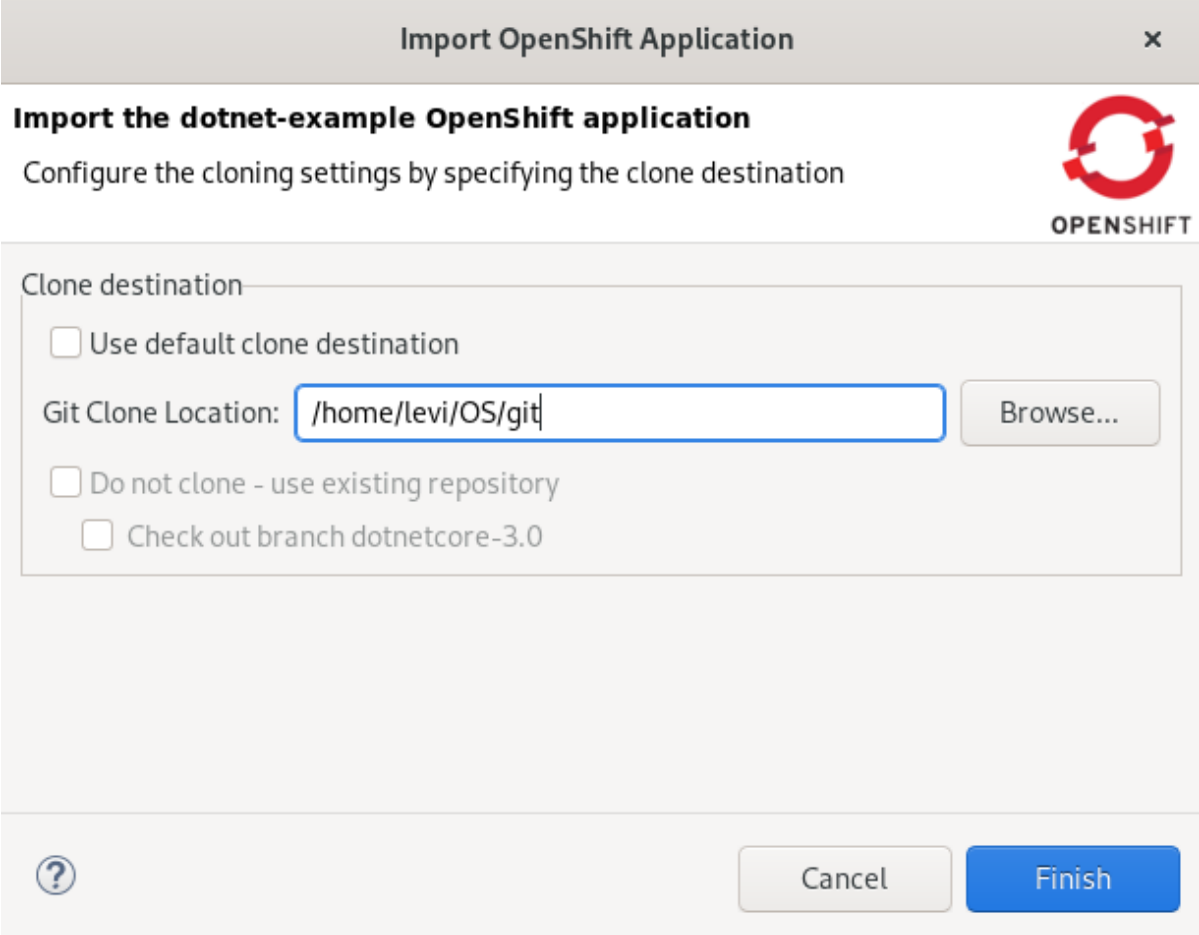
Click [here](#) for webhooks available to automatically trigger builds.

Note the following parameters required to administer your resources:

Name	Value
APPLICATION_DOMAIN	
CONTEXT_DIR	
DOTNET_ASSEMBLY_NAME	
DOTNET_CONFIGURATION	Release
DOTNET_IMAGE_STREAM_TAG	dotnet:3.0
DOTNET_NPM_TOOLS	
DOTNET_PUBLISH_READYTORUN	

OK

12. **OK** をクリックします。  
Import OpenShift application ウィンドウが表示されます。



**Import OpenShift Application** ×

**Import the dotnet-example OpenShift application**

Configure the cloning settings by specifying the clone destination


**Clone destination**

☐ Use default clone destination

Git Clone Location:

☐ Do not clone - use existing repository

☐ Check out branch dotnetcore-3.0



13. **Git Clone Location** を選択します。

14. **Finish** をクリックします。

新たに作成された OpenShift Container Platform アプリケーションが **OpenShift Explorer** ビューに表示されます。

#### その他のリソース

- OpenShift Container Platform でテンプレートを使用および作成する方法については、「[Using Templates](#)」を参照してください。

#### 2.1.4. 既存の OpenShift Container Platform アプリケーションの IDE へのインポート

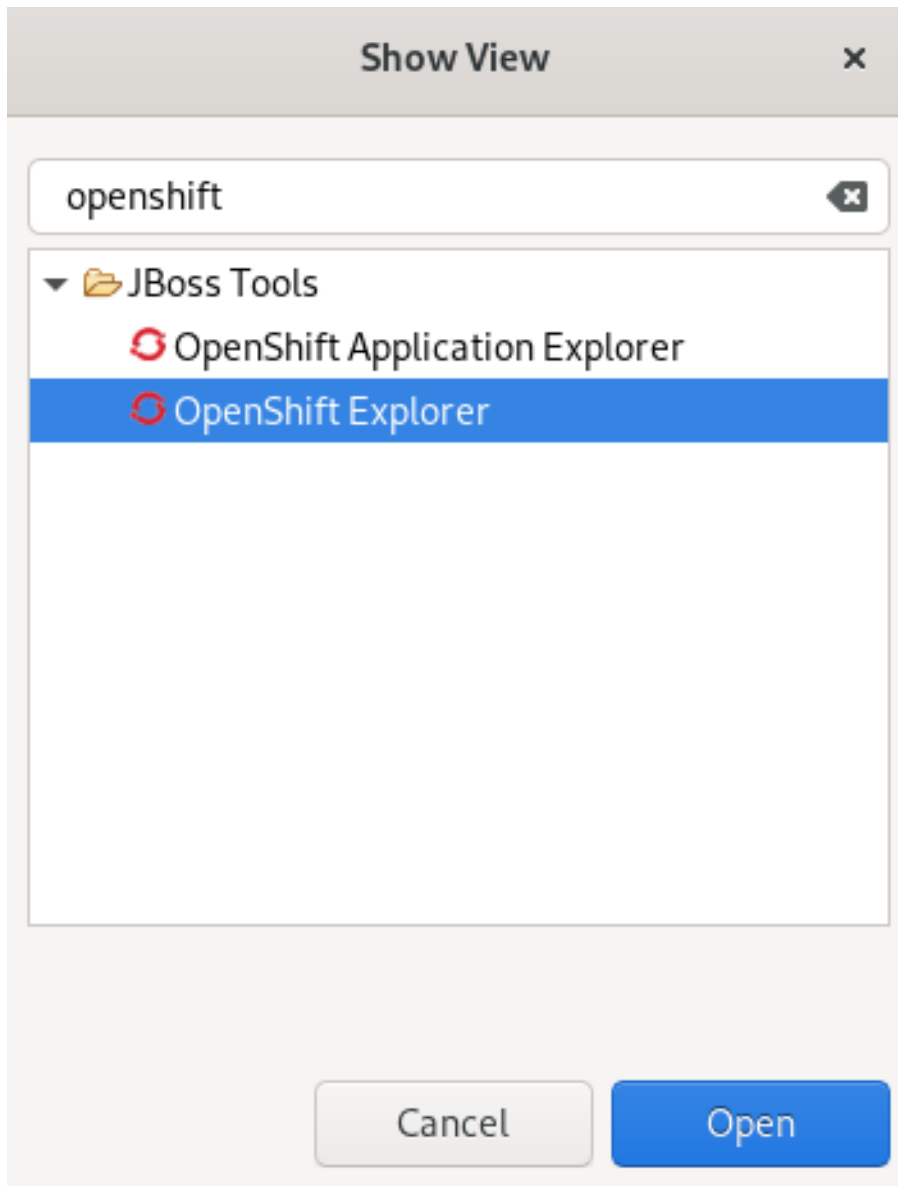
IDE の **OpenShift Explorer** ビューには、OpenShift Container Platform アカウントに関連付けられたアプリケーションが表示されます。**Import OpenShift Application** ウィザードを使用すると、これらのアプリケーションのソースコードを個別に IDE にインポートできます。アプリケーションをインポートした後、アプリケーションのソースコードの変更、アプリケーションのビルド、および Web ブラウザーでの表示を簡単に行うことができます。

#### 前提条件

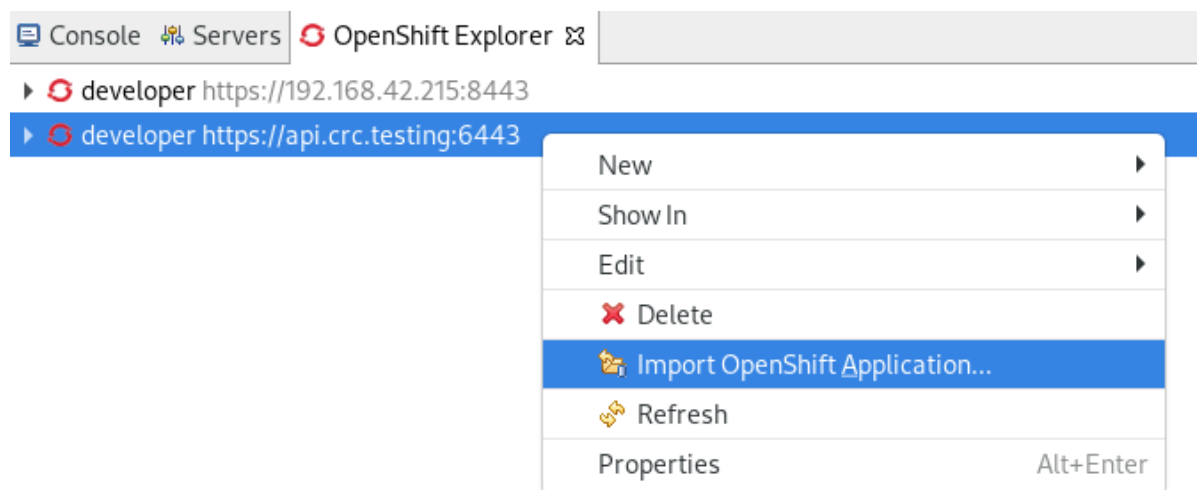
- IDE にインポートするアプリケーションのソースは **build config** ファイルに指定されている必要があります。
- OpenShift Container Platform コネクション。  
新たに OpenShift Container Platform コネクションを作成する方法の詳細は、「[新規 OpenShift Container Platform コネクションの作成](#)」を参照してください。

## 手順

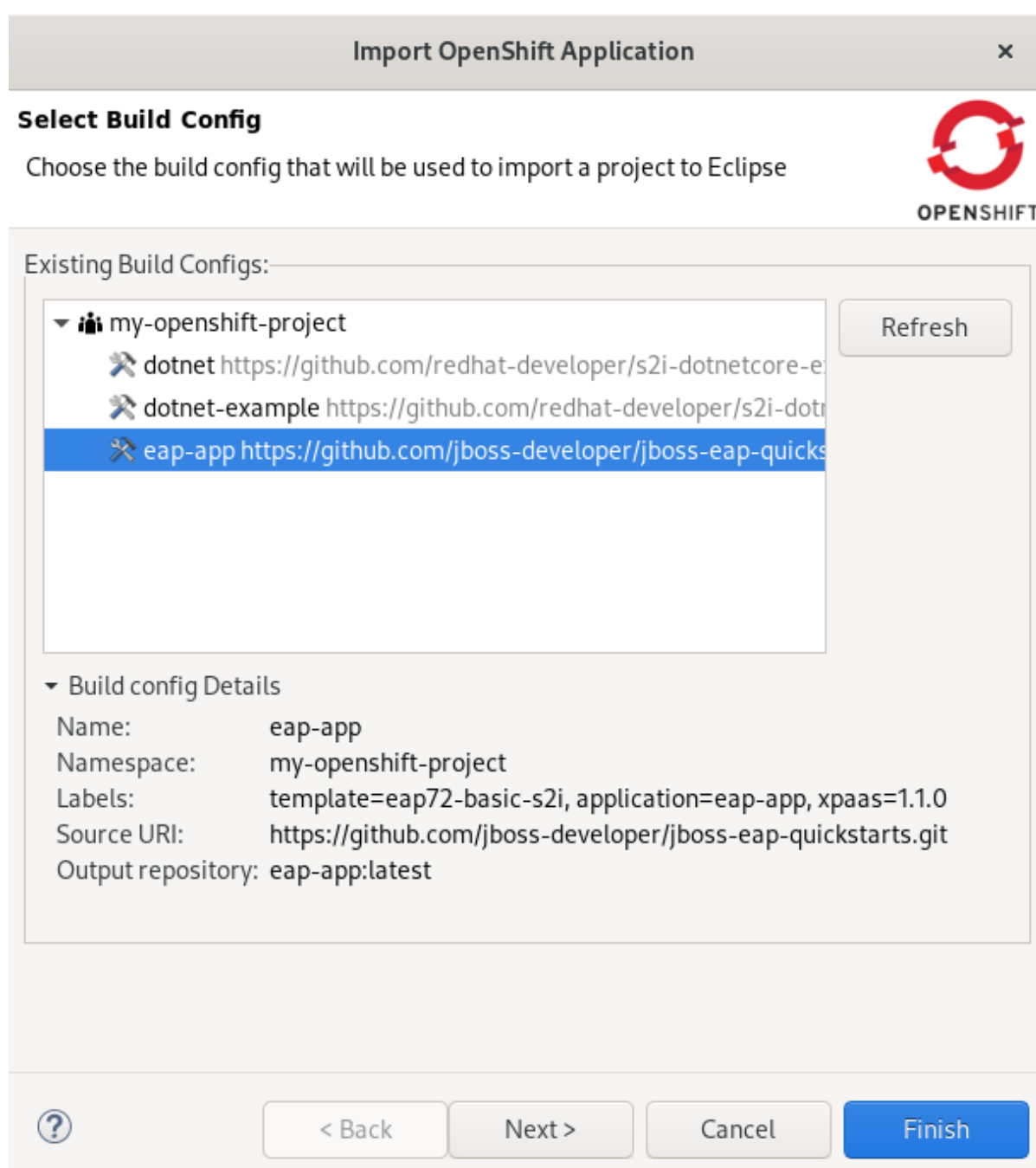
1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
Show View ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
OpenShift Explorer ビューが表示されます。
6. **OpenShift Container Platform connection** → **Import OpenShift Application** を右クリックします。

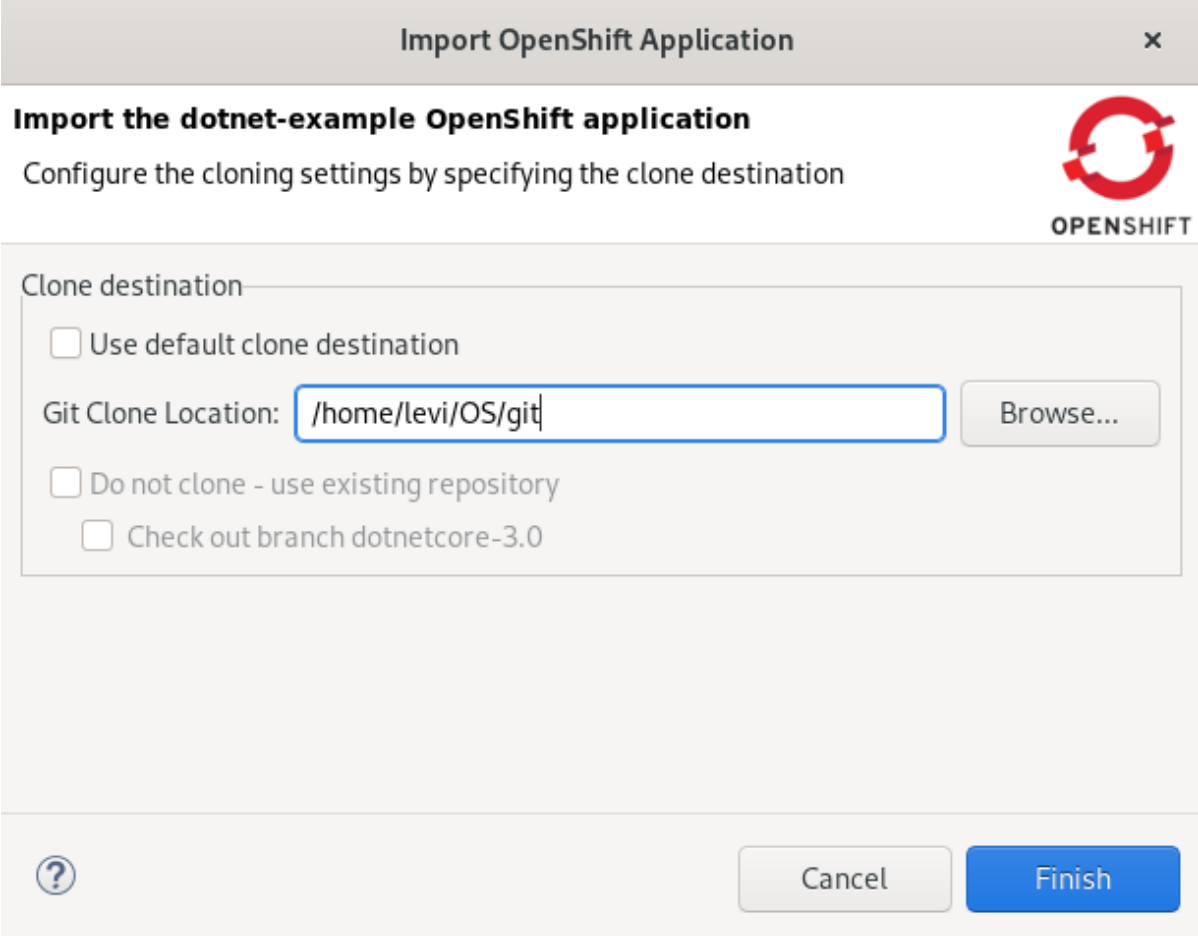


Select Build Config ウィンドウが表示されます。





7. インポートするアプリケーションを選択します。
8. **Next** をクリックします。  
Import OpenShift application ウィンドウが表示されます。



Import OpenShift Application

Import the dotnet-example OpenShift application

Configure the cloning settings by specifying the clone destination

Clone destination

☐ Use default clone destination

Git Clone Location:

☐ Do not clone - use existing repository

☐ Check out branch dotnetcore-3.0

9. Git Clone Location を選択します。

10. **Finish** をクリックします。

新たにインポートされた OpenShift Container Platform アプリケーションが **OpenShift Explorer** ビューに表示されます。

### 2.1.5. サーバーアダプターを使用したアプリケーションのデプロイ

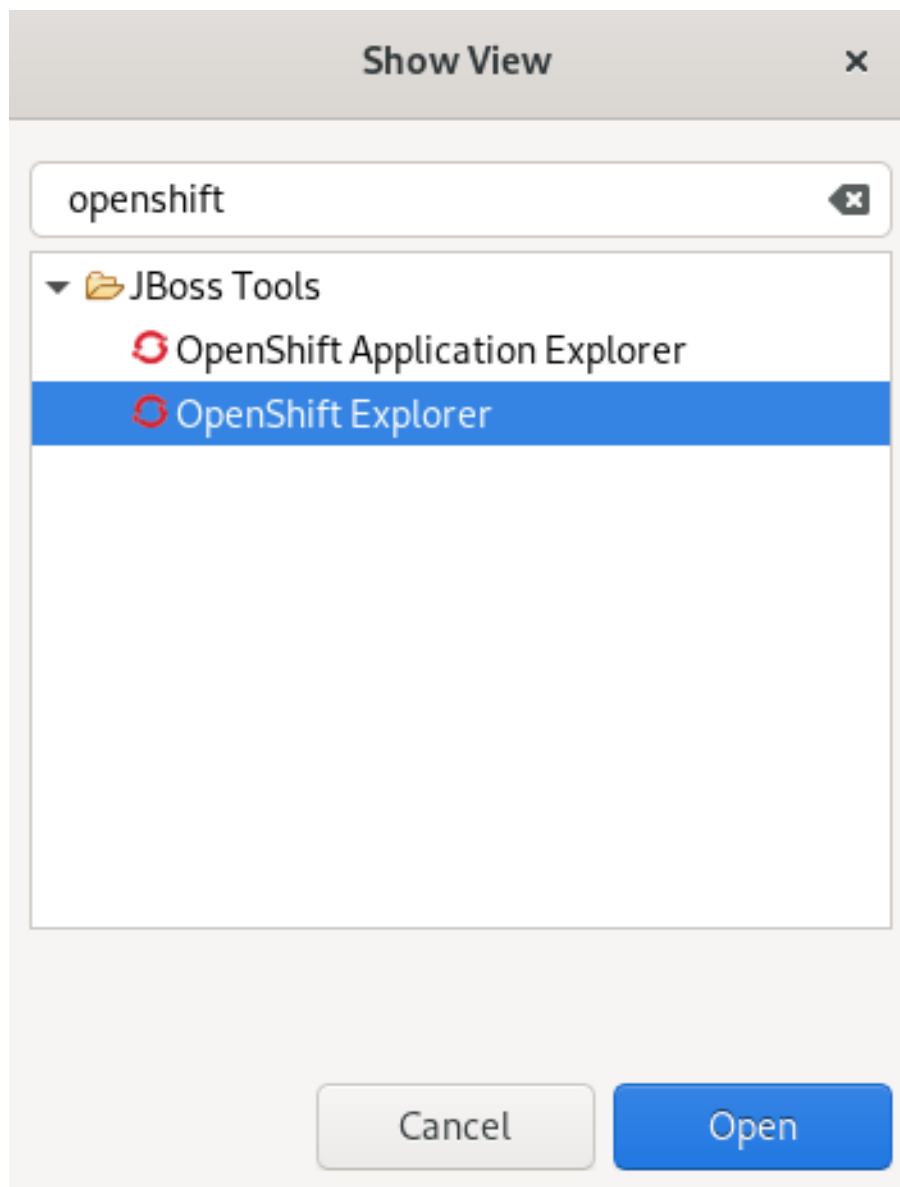
サーバーアダプターを使用すると、ワークスペースプロジェクトに追加した変更を OpenShift インスタンス上で実行中の OpenShift アプリケーションにパブリッシュできます。これにより、アプリケーションの変更部分を OpenShift にデプロイされた Pod に直接デプロイできます。サーバーアダプターを使用すると、ソースコードを Git リポジトリにコミットせずに、アプリケーションの変更を直接実行中の OpenShift アプリケーションにプッシュできます。

#### 前提条件

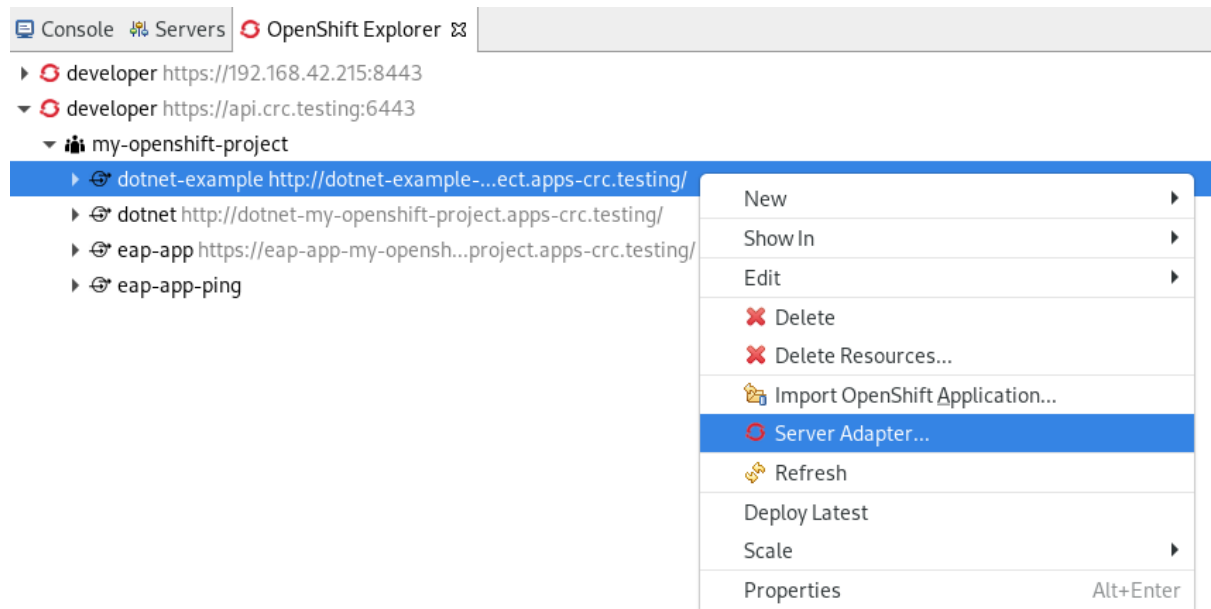
- OpenShift Container Platform コネクション。  
新たに OpenShift Container Platform コネクションを作成する方法の詳細は、[「新規 OpenShift Container Platform コネクションの作成」](#) を参照してください。

#### 手順

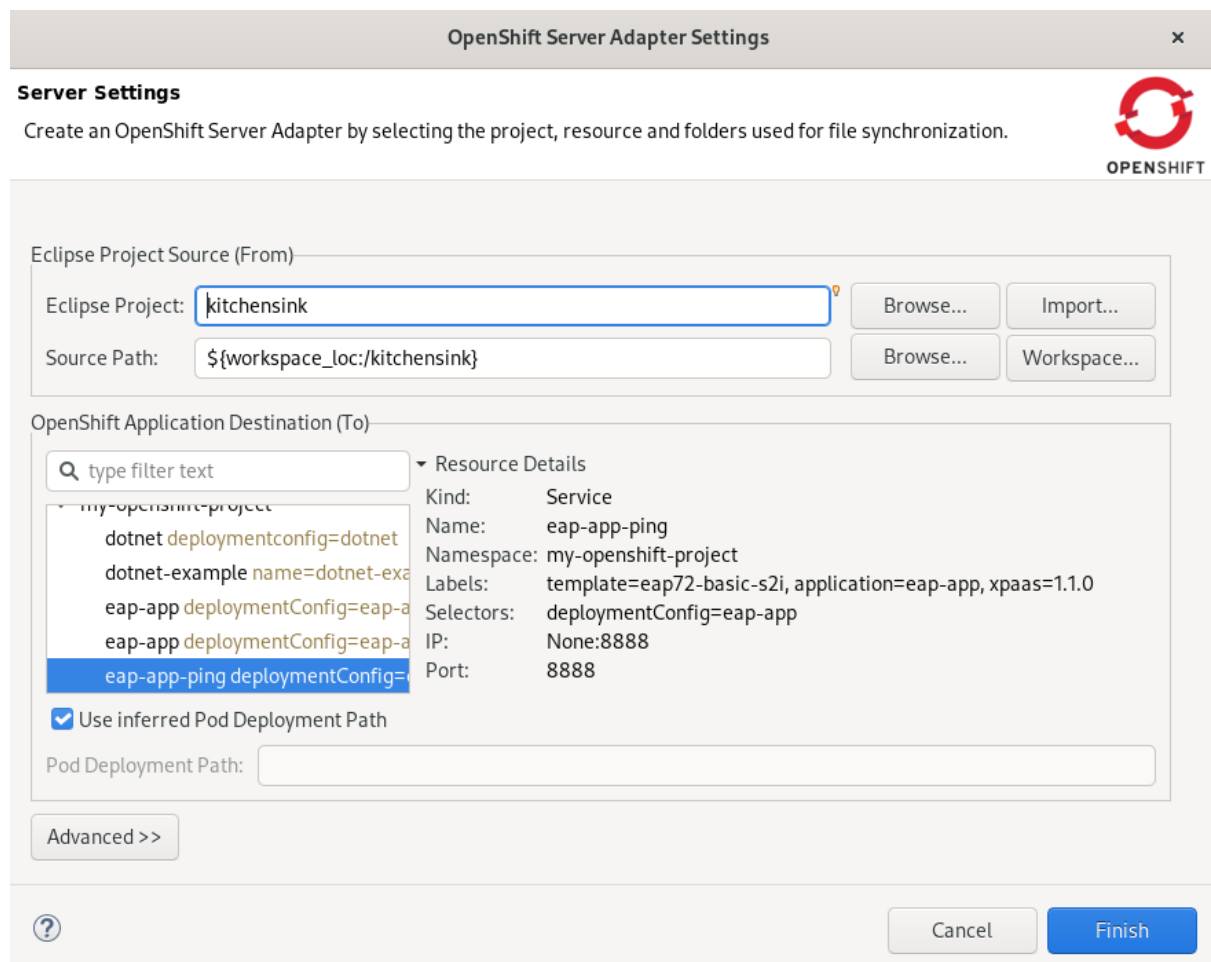
1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. OpenShift Container Platform コネクションを展開します。
7. **application** → **Server Adapter**を右クリックします。



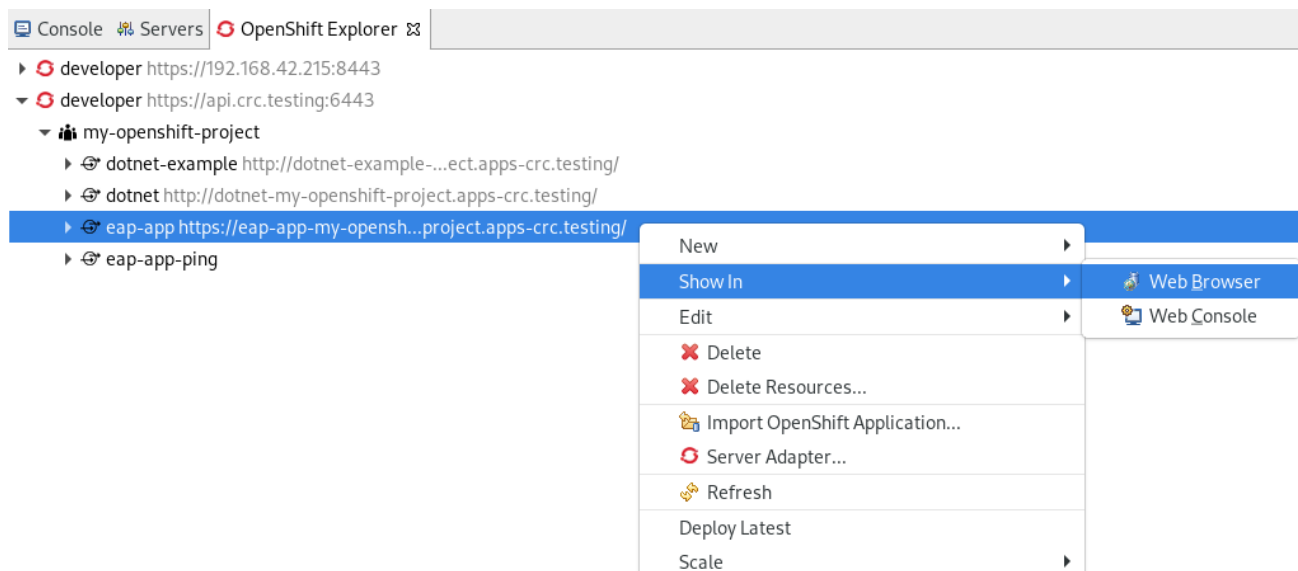
Server Settings ウィンドウが表示されます。



8. **Finish** をクリックします。

**Servers** ビューが表示され、サーバーアダプターを起動します。

ブラウザーでアプリケーションを開くには、**application** → **Show In** → **Web Browser** を右クリックします。



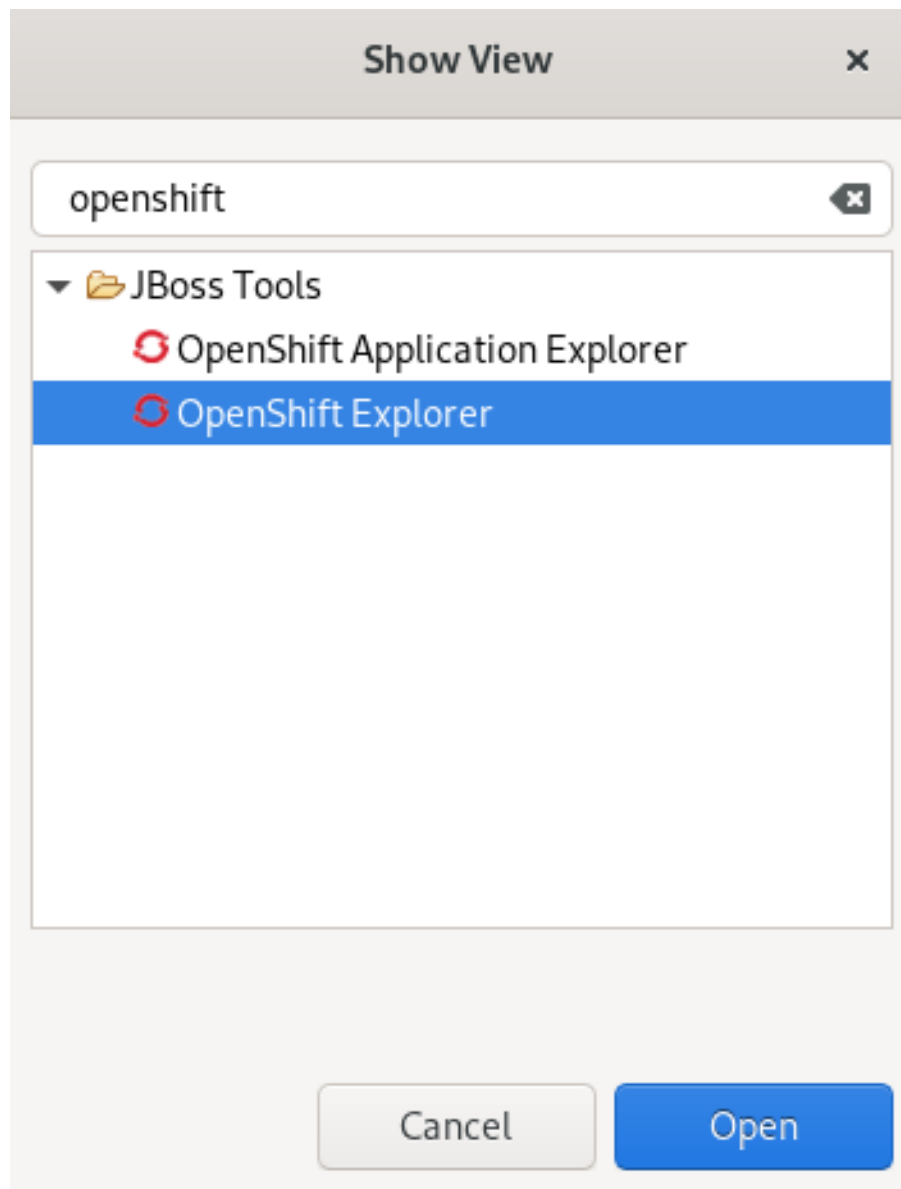
CodeReady Studio の組み込み Web ブラウザーが開かれ、アプリケーションが表示されます。

## 2.1.6. OpenShift Container Platform プロジェクトの削除

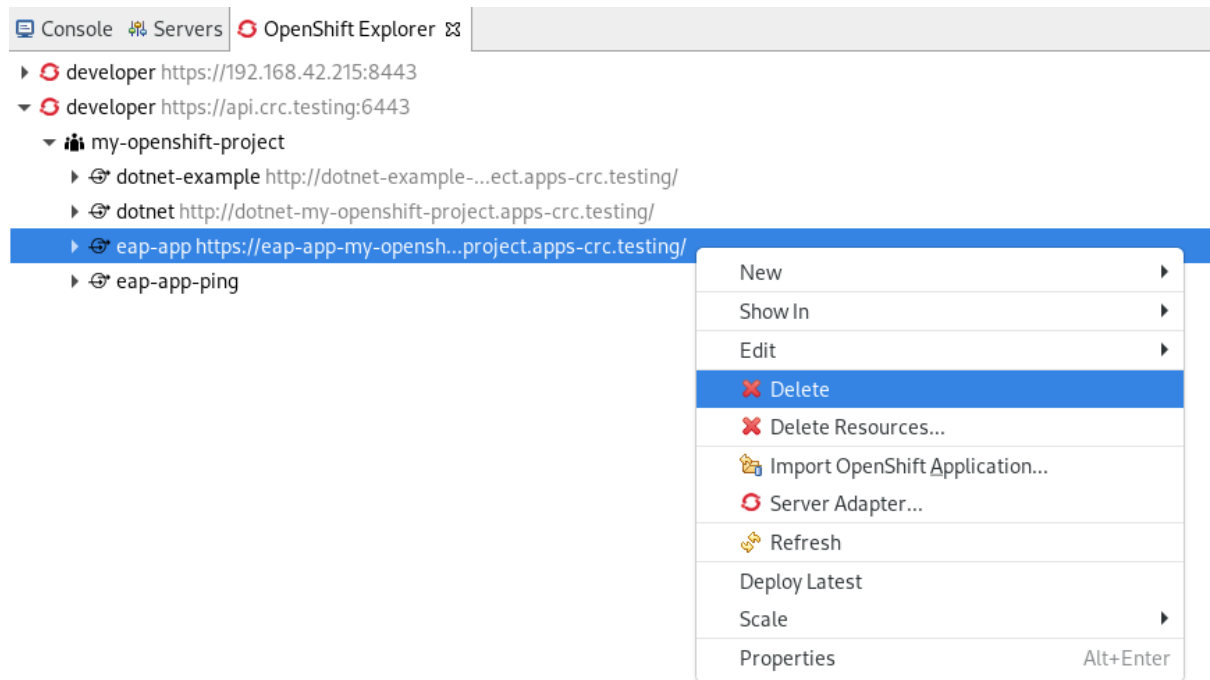
プロジェクトの開発を新たに開始する場合や、プロジェクトの開発が終了した後に、ワークスペースからプロジェクトを削除することがあります。プロジェクトを削除すると、プロジェクトに関連付けられたリソースはすべて削除されます。

### 手順

1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. OpenShift Container Platform コネクションを展開します。
7. **project** → **Delete** を右クリックします。



Delete OpenShift Resource ウィンドウが表示されます。

8. **OK** をクリックします。

これでプロジェクトが削除されます。

## 2.2. OPENSIFT CONTAINER PLATFORM アプリケーションの設定およびリモート監視

IDE を使用すると、ユーザーは OpenShift Container Platform のリモートインスタンスへの接続を設定し、ログ (アプリケーションログおよびビルドログ) を使用して、実行中のアプリケーションをトラブルシューティングおよび監視できます。

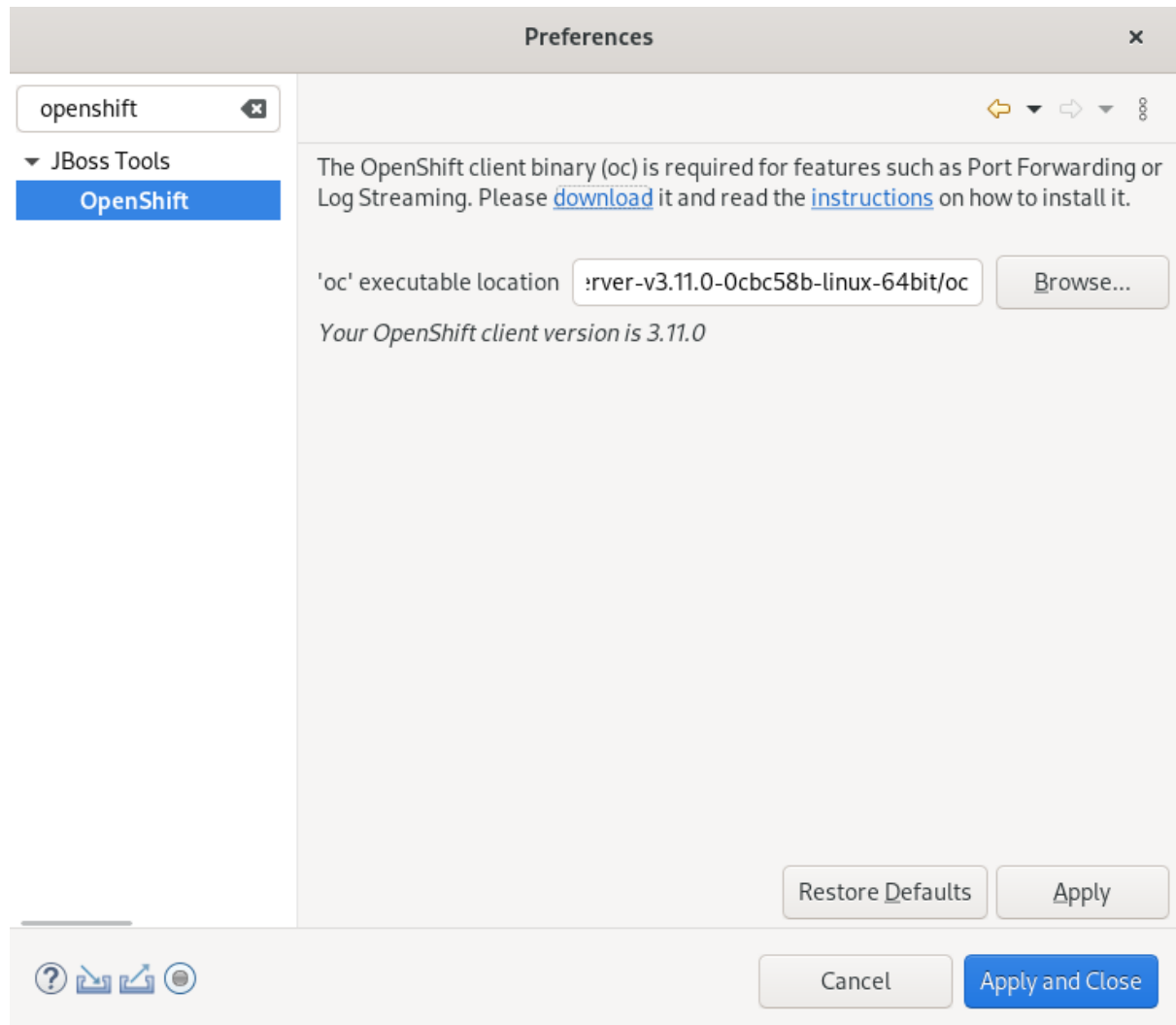
### 2.2.1. OpenShift Client Binaries の設定

#### 前提条件

ポート転送の設定や、アプリケーションおよびビルドログのストリーミングを行う前に、必ず OpenShift Client Binaries を設定する必要があります。

#### 手順

1. CodeReady Studio を起動します。
2. **Window → Preferences** をクリックします。  
**Preferences** ウィンドウが表示されます。



3. 検索フィールドに OpenShift と入力します。
4. OpenShift を選択します。
5. **Browse** をクリックして、**oc** 実行可能ファイルを見つけます。
6. **Apply and Close** をクリックします。

これで、OpenShift Client Binaries が設定されます。

### 2.2.2. ポート転送の設定

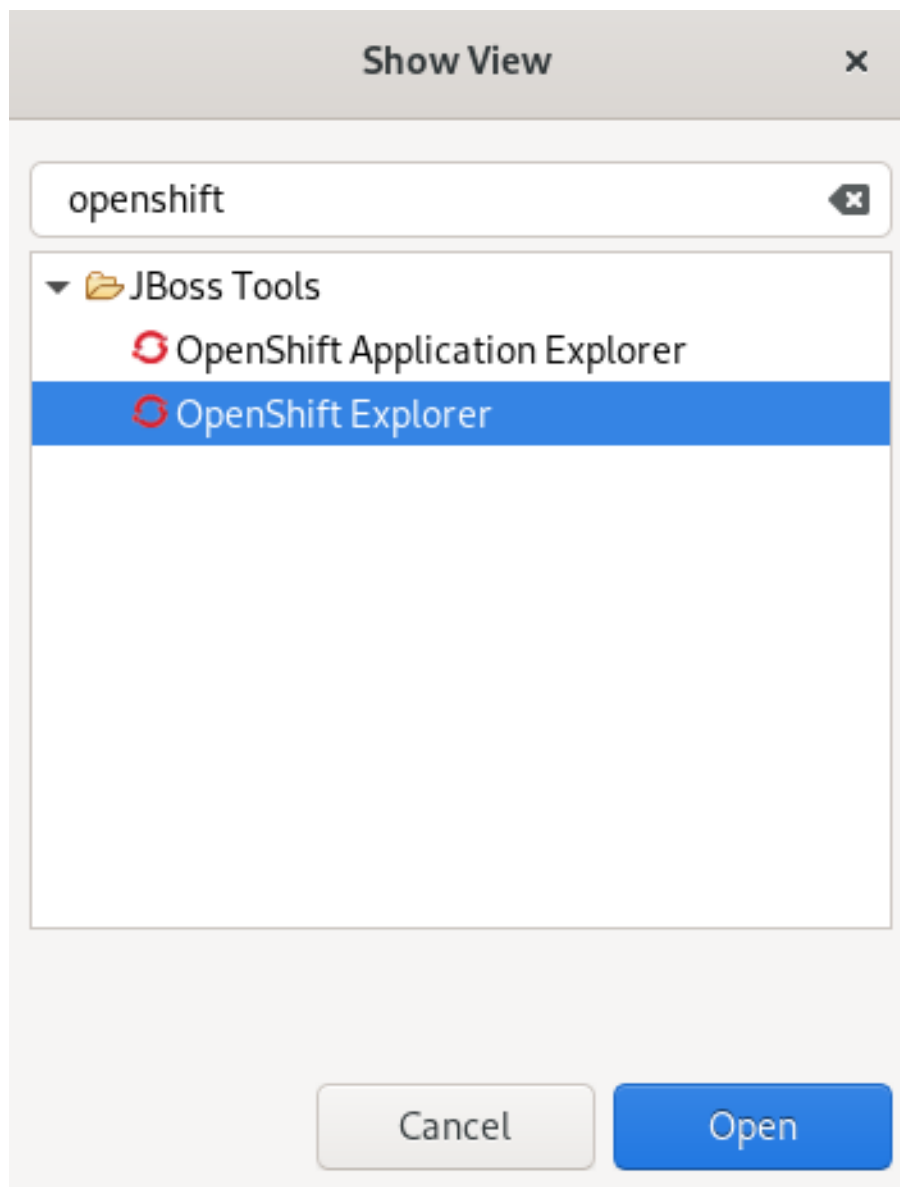
**Application Port Forwarding** ウィンドウを使用すると、ローカルポートをリモートポートに接続し、データのアクセスやアプリケーションのデバッグを行うことができます。以下のいずれかの理由で、ポート転送は自動的に停止します。

- OpenShift Container Platform コネクションの終了
- IDE のシャットダウン
- ワークスペースの変更

ポート転送は、IDE から OpenShift Container Platform に接続するたびに有効にする必要があります。

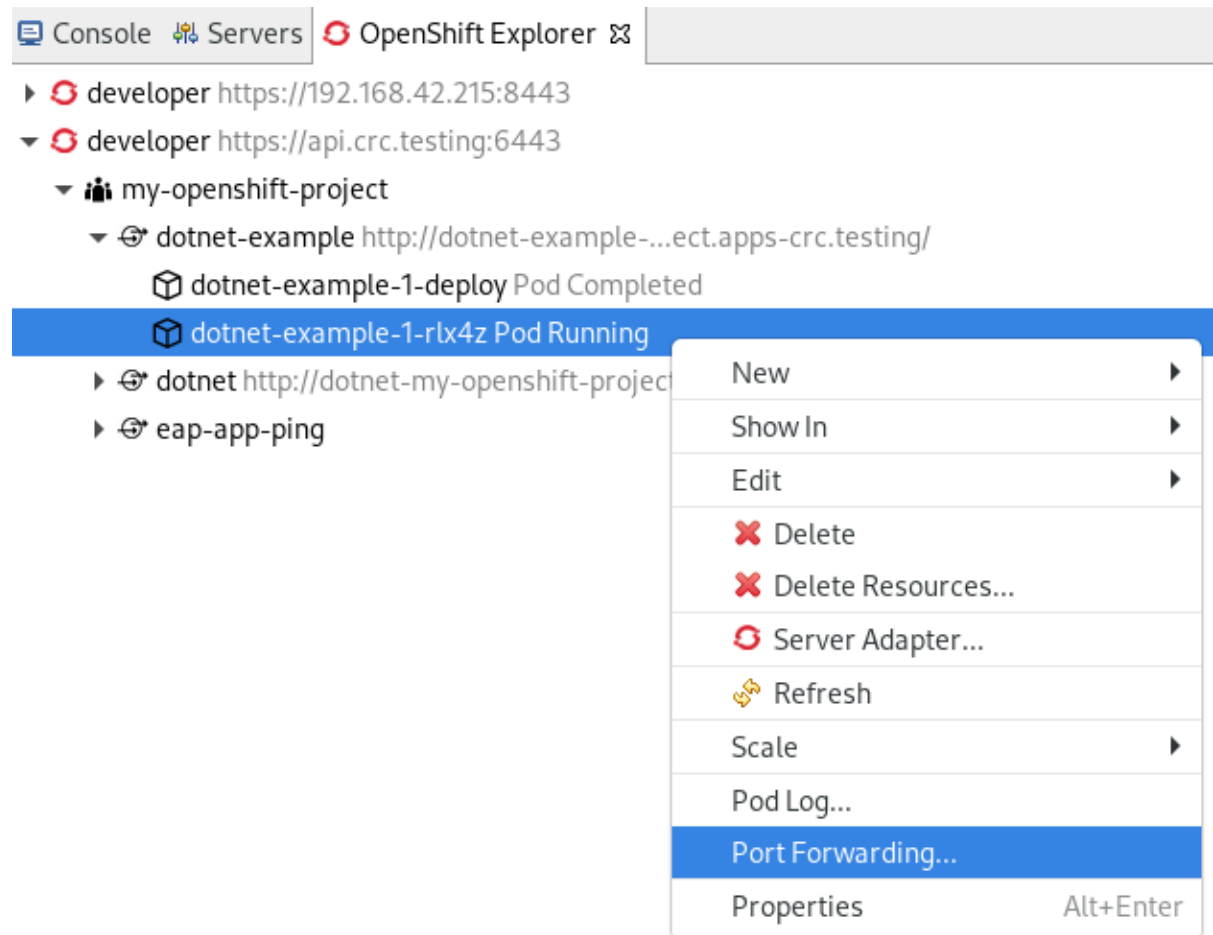
#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。

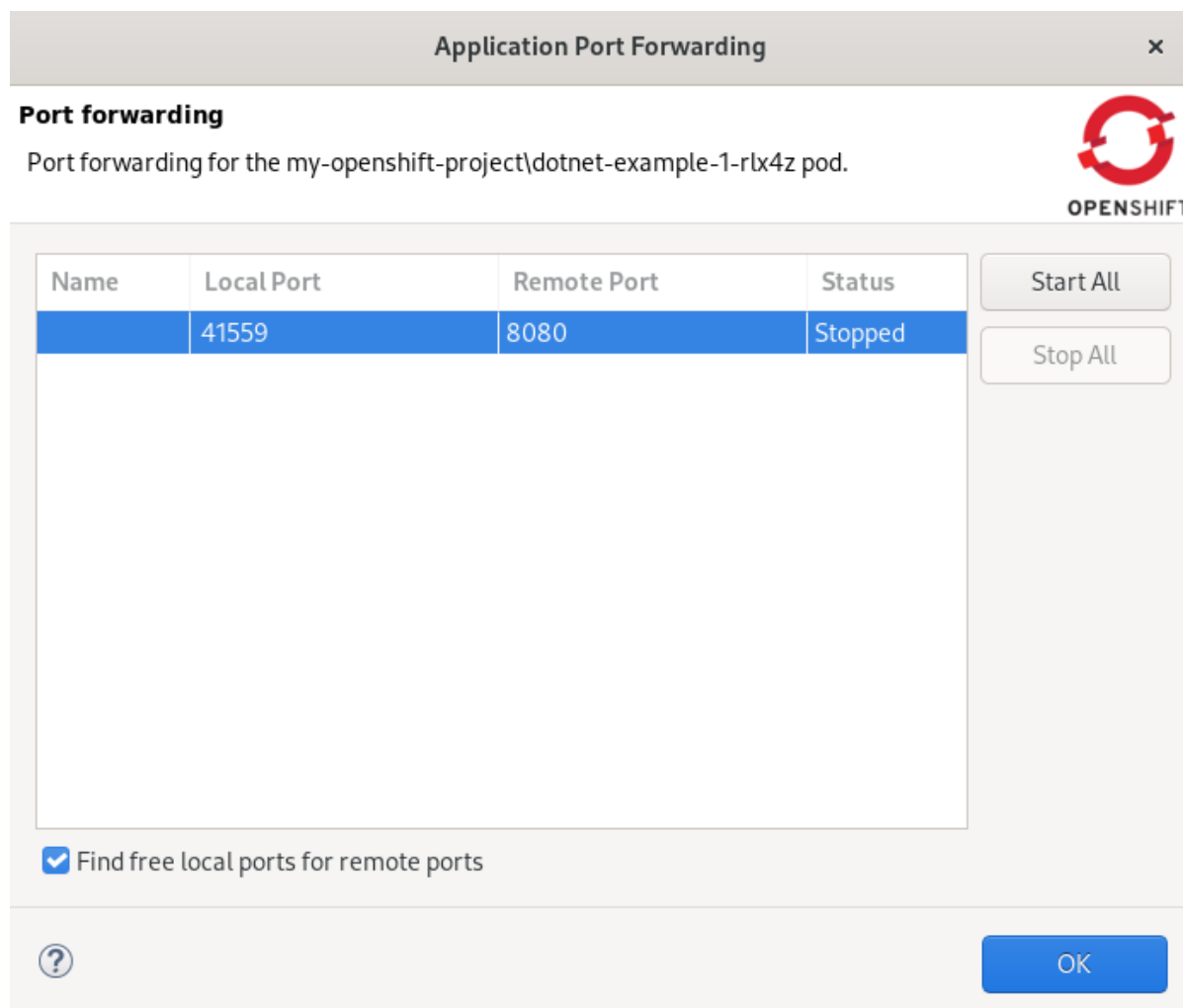


3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. OpenShift Container Platform コネクションを展開します。
7. **application** → **Port Forwarding** を右クリックします。





Port Forwarding ウィンドウが表示されます。



8. Find free local ports for remote ports ボックスを選択します。
9. **Start All** をクリックします。
10. **OK** をクリックします。

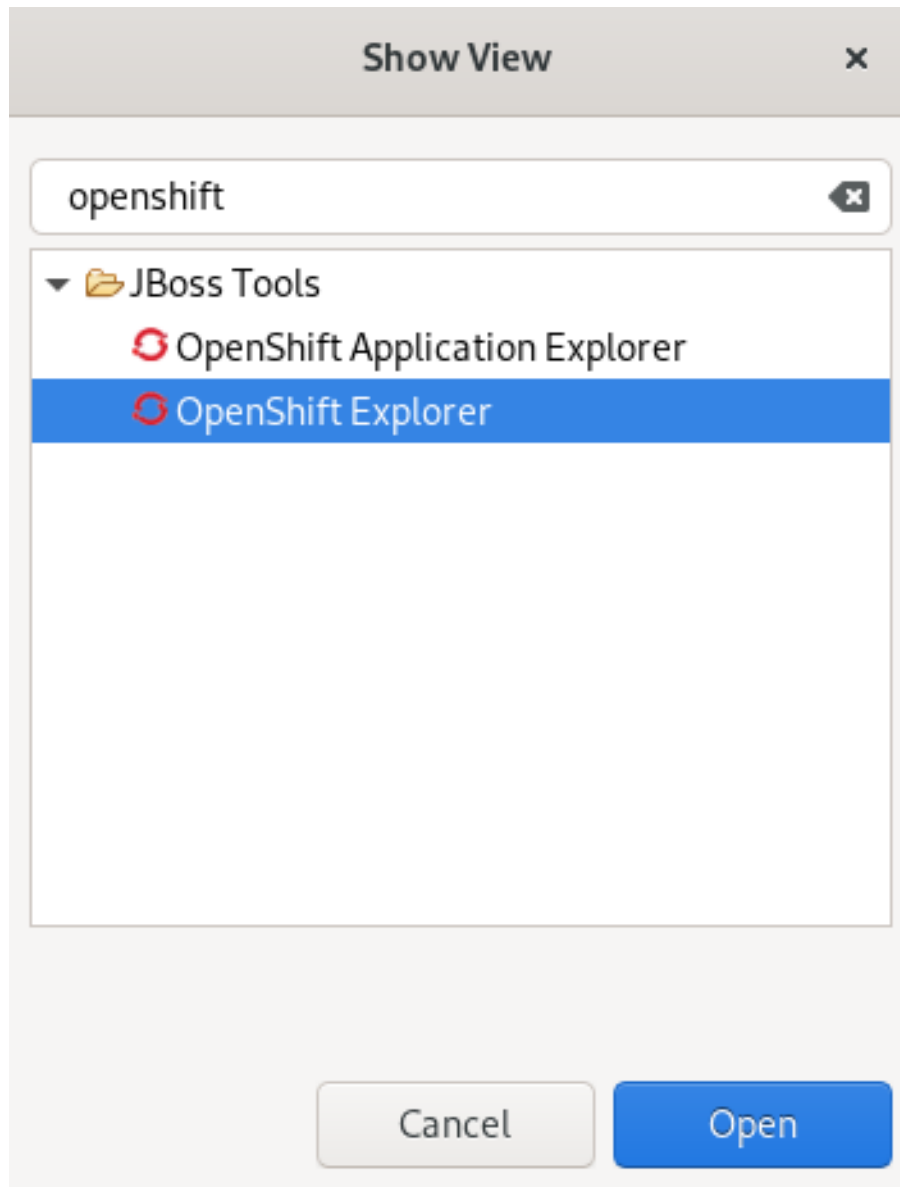
**Console** ビューが表示され、ポート転送の開始プロセスが表示されます。

### 2.2.3. Pod ログのストリーミング

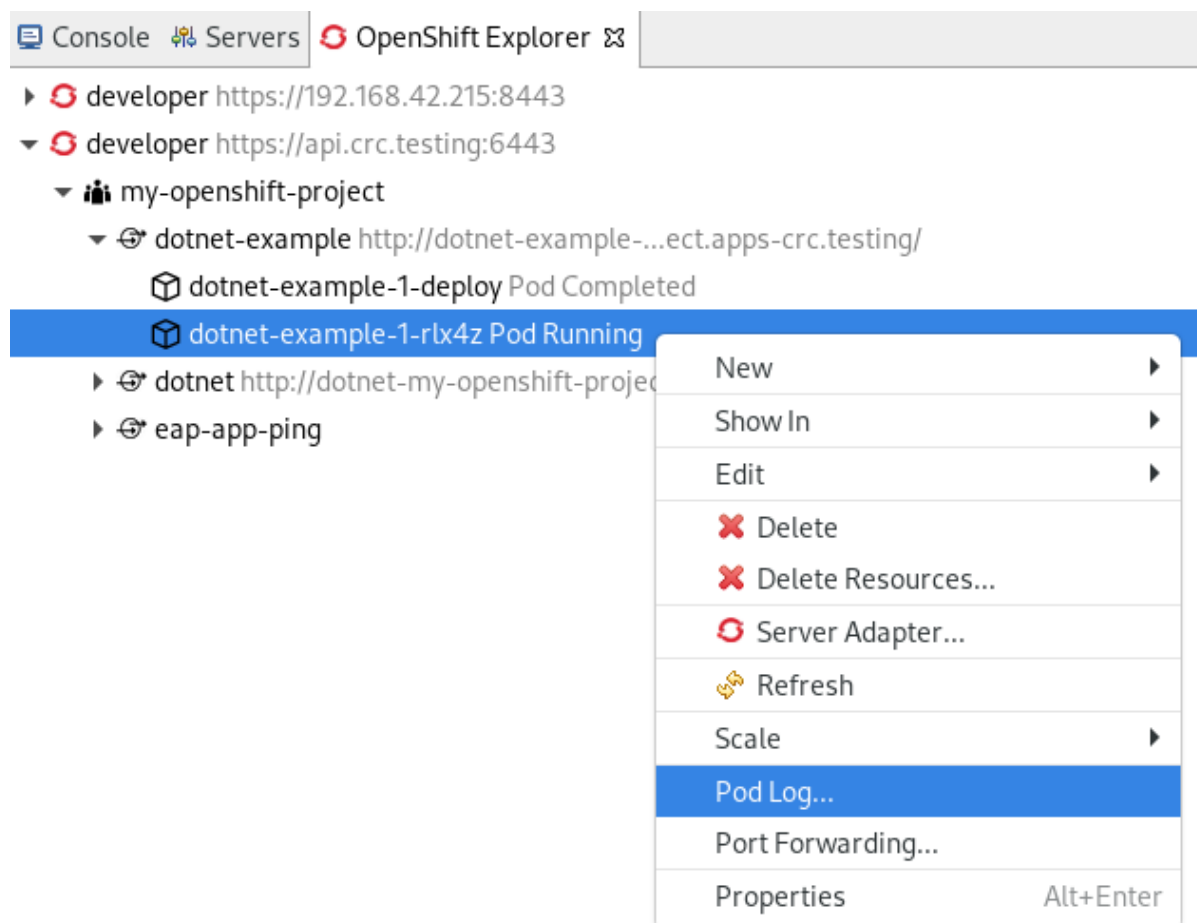
Pod ログは、リモート OpenShift Container Platform インスタンスで実行されるアプリケーションの一般的なログです。IDE のストリーミングアプリケーションログ機能は、アプリケーションを監視し、アプリケーションが失敗したりエラーを返した場合に以前の Pod ログを使用してトラブルシューティングを行います。

#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. OpenShift Container Platform コネクションを展開します。
7. **application** → **Port Log** を右クリックします。



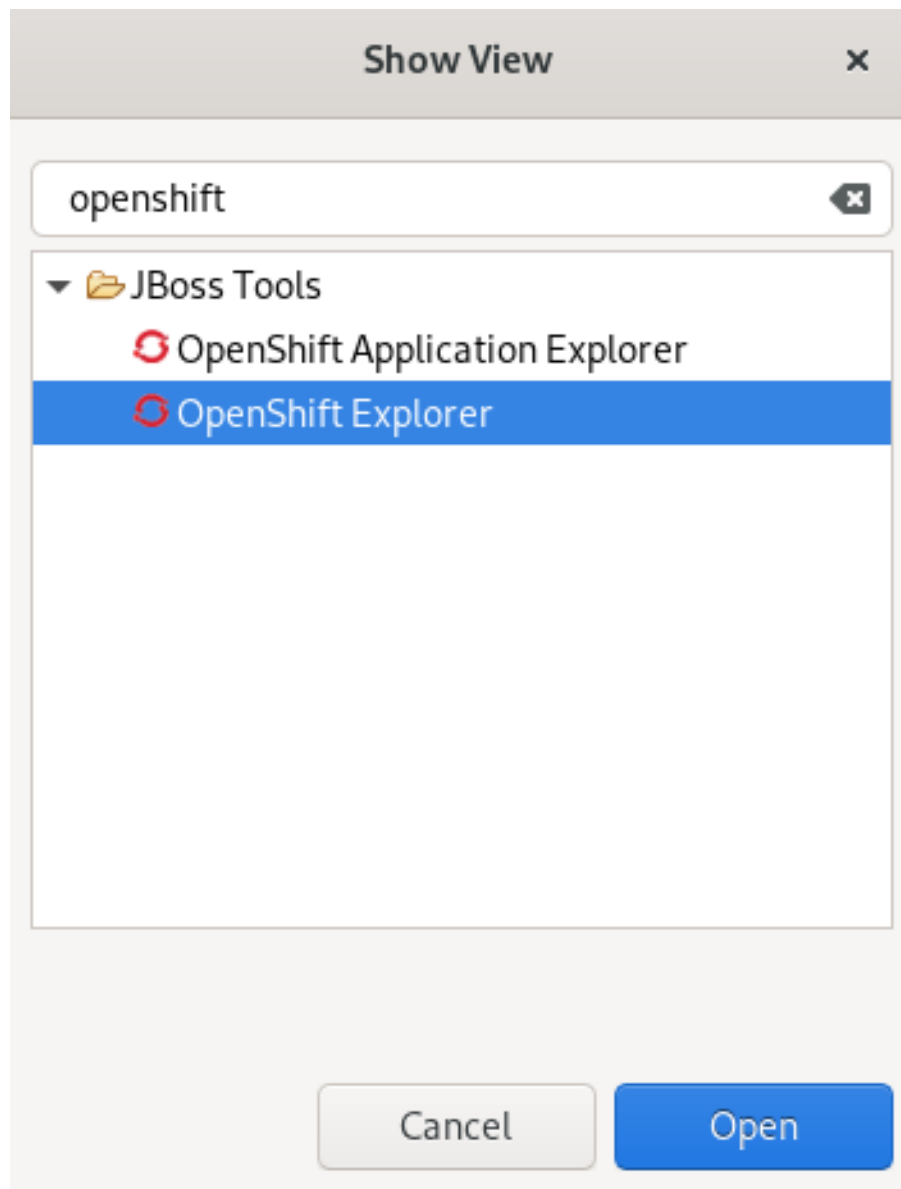
Console ビューが表示され、Pod ログが表示されます。

## 2.2.4. ビルドログのストリーミング

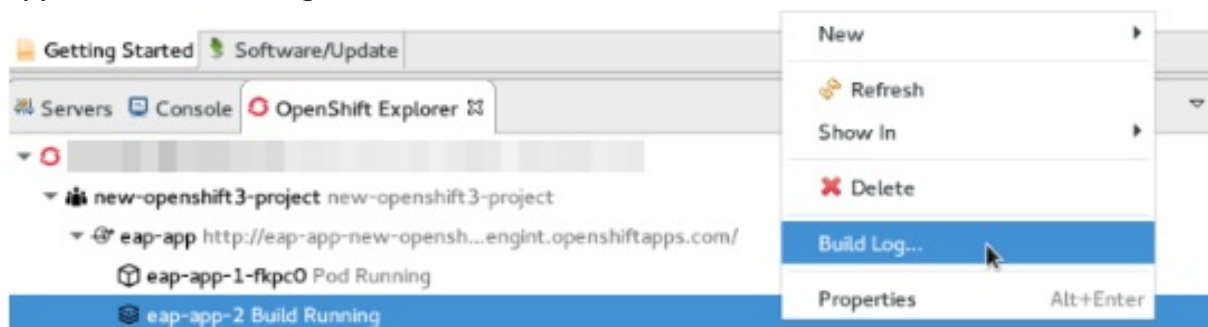
ビルドログは、リモート OpenShift Container Platform インスタンスで実行されているアプリケーションへの変更を記録するログです。IDE のストリーミングビルドログ機能は、アプリケーションのビルドプロセスの進捗や、アプリケーションのデバッグに使用されます。

### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
Show View ウィンドウが表示されます。



3. 検索フィールドに **OpenShift** と入力します。
4. **OpenShift Explorer** を選択します。
5. **Open** をクリックします。  
**OpenShift Explorer** ビューが表示されます。
6. OpenShift Container Platform コネクションを展開します。
7. **application** → **Build Log** を右クリックします。



Console ビューが表示され、ビルドログが表示されます。

## 2.3. その他のリソース

- OpenShift Application Explorer の詳細は、『[CodeReady Studio ツールのスタートガイド](#)』を参照してください。

## 第3章 DOCKER での開発

### 前提条件

- Docker がシステムにインストールされている。  
Docker のインストール方法の詳細は、「[Get Docker](#)」を参照してください。
- Docker ID を取得している。  
Docker ID を取得する方法の詳細は「[Register for a Docker ID](#)」を参照してください。

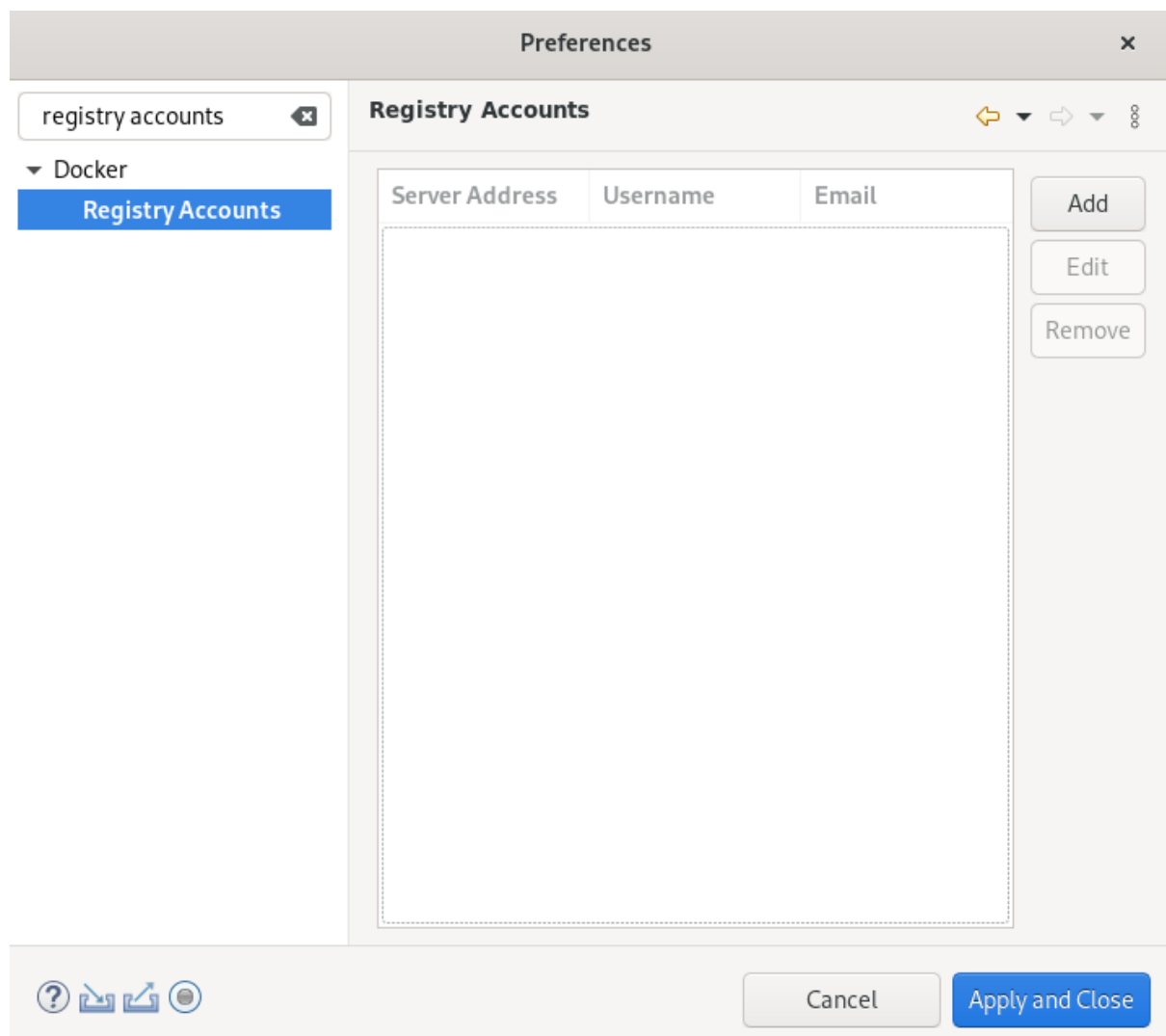
## 3.1. DOCKER コネクションの管理

### 3.1.1. CodeReady Studio での Docker アカウントの設定

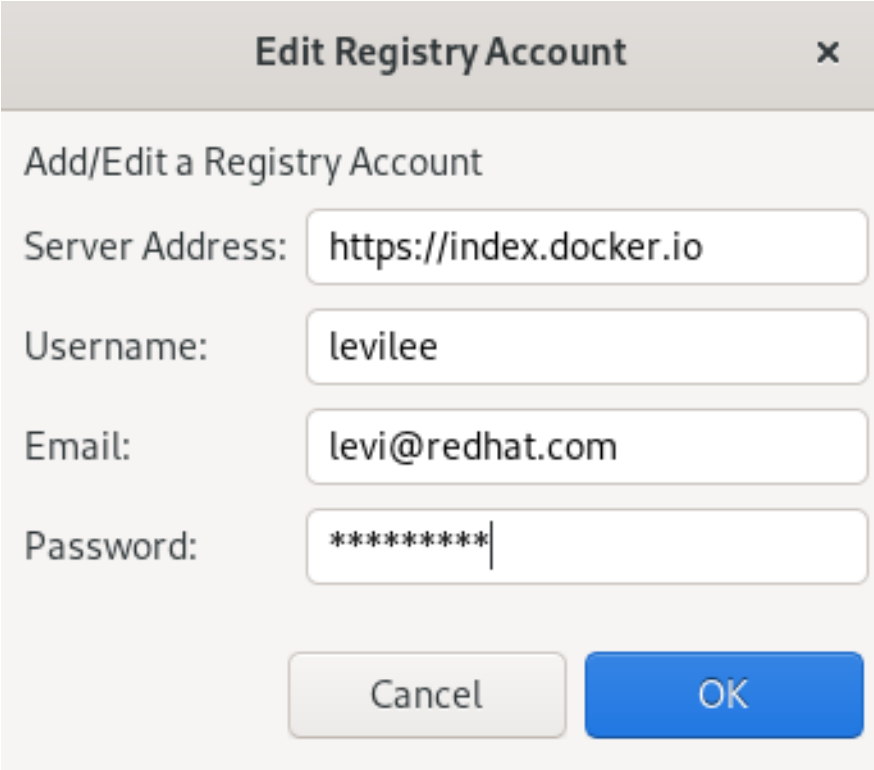
CodeReady Studio で Docker アカウントを設定する方法を説明します。ここでは、本章の前提条件に記載されている手順が完了済みであることを仮定しています。

### 手順

1. CodeReady Studio を起動します。
2. **Window → Preferences** をクリックします。  
**Preferences** ウィンドウが表示されます。



3. 検索フィールドに **Registry Accounts** と入力します。
4. **Registry Accounts** を選択します。
5. **Add** をクリックします。  
**New Registry Account** ウィンドウが表示されます。



**Edit Registry Account** ×

Add/Edit a Registry Account

Server Address:

Username:

Email:

Password:

6. Docker ハブの **Server Address** を入力します。
7. Docker ID を **Username** として入力します。
8. Docker アカウントに関連するメールを入力します。
9. パスワードを入力します。
10. **OK** をクリックします。
11. **Apply and Close** をクリックします。

### 3.1.2. Docker コネクションのテスト

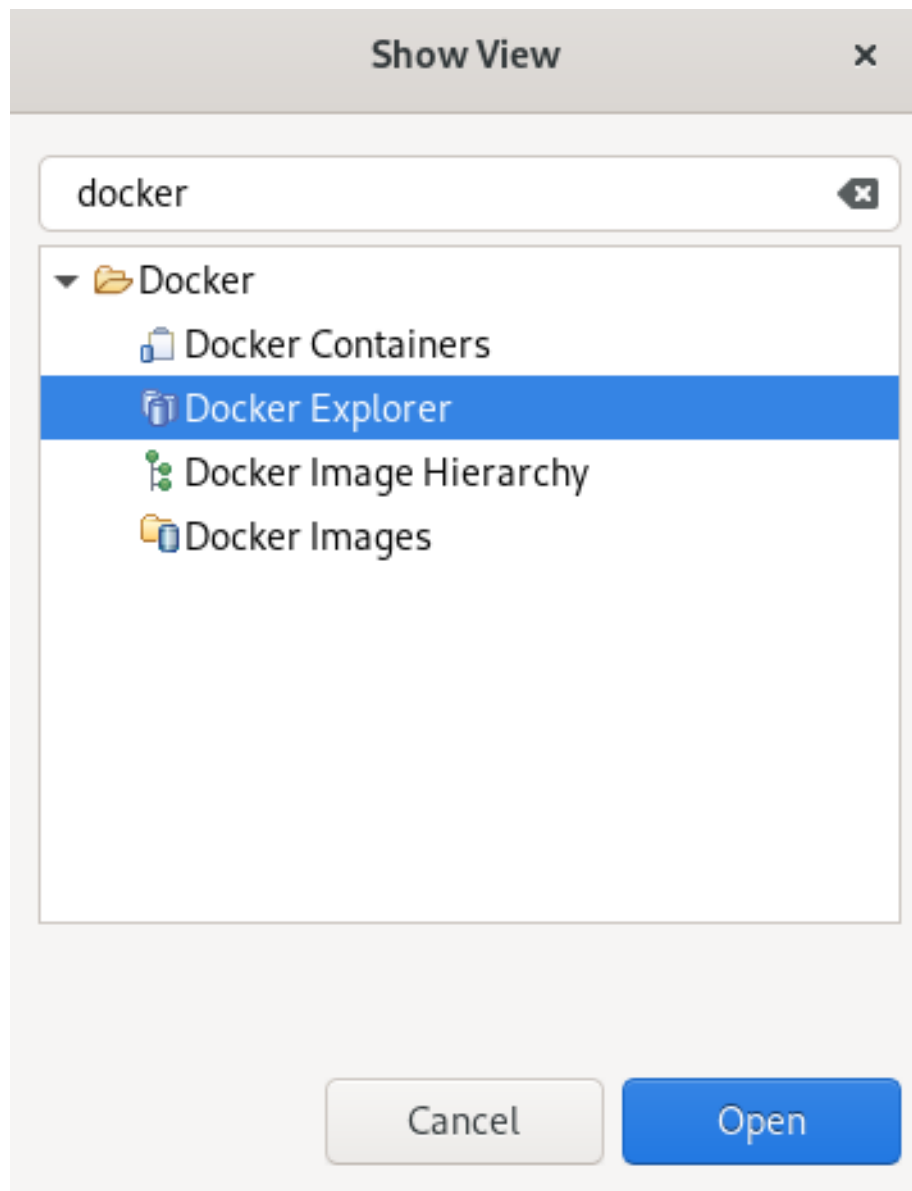
#### 前提条件

- CodeReady Studio に Docker アカウントが設定されている。  
CodeReady Studio に Docker アカウントを設定する方法の詳細は、[「CodeReady Studio での Docker アカウントの設定」](#) を参照してください。

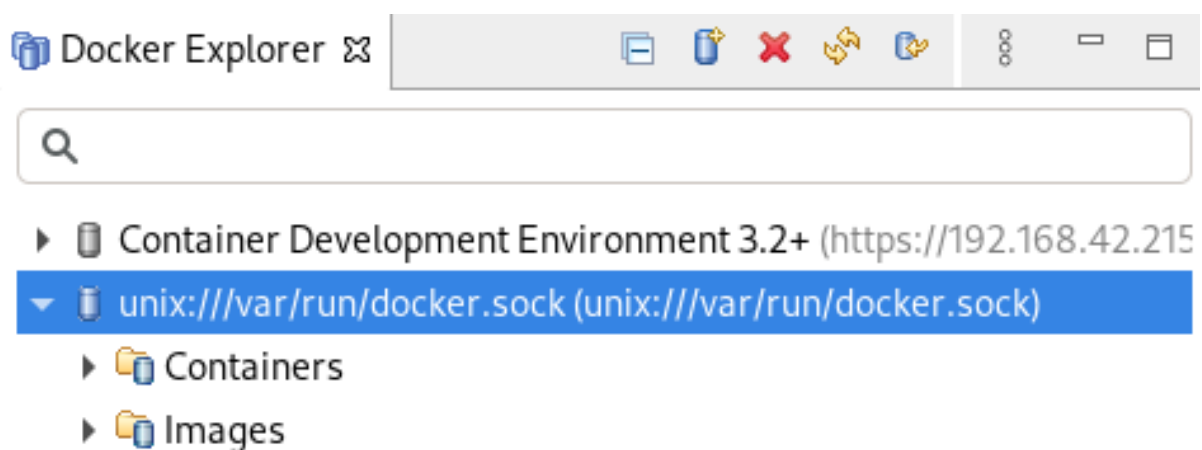
#### 手順

1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。

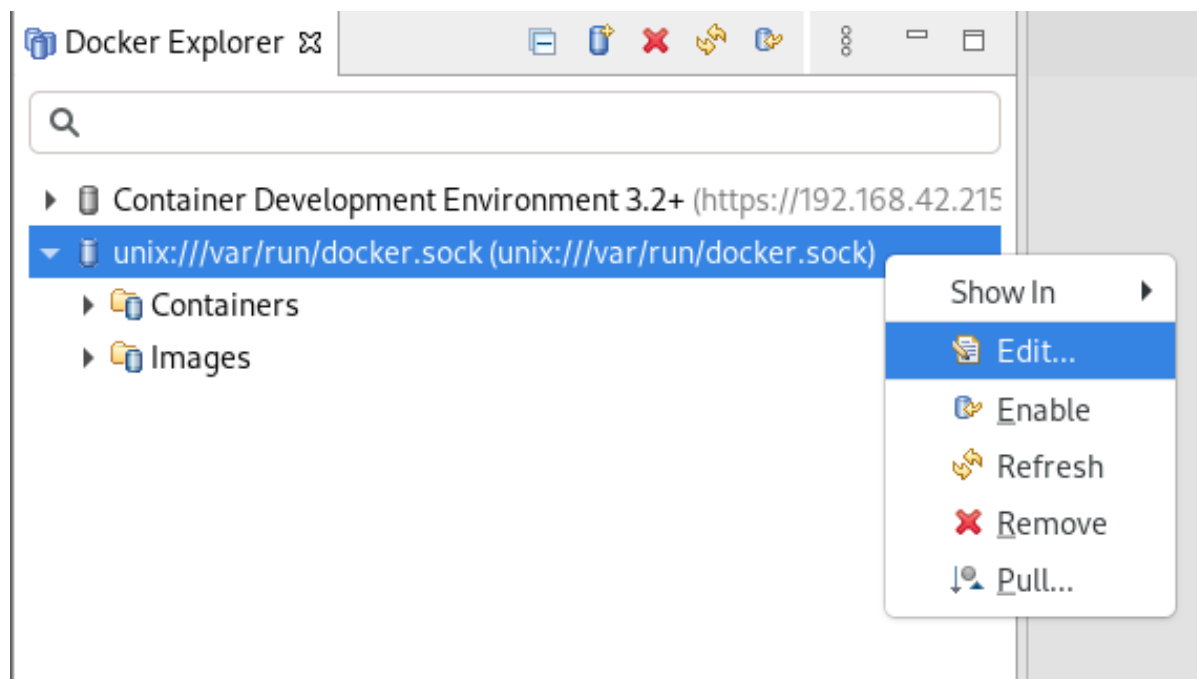




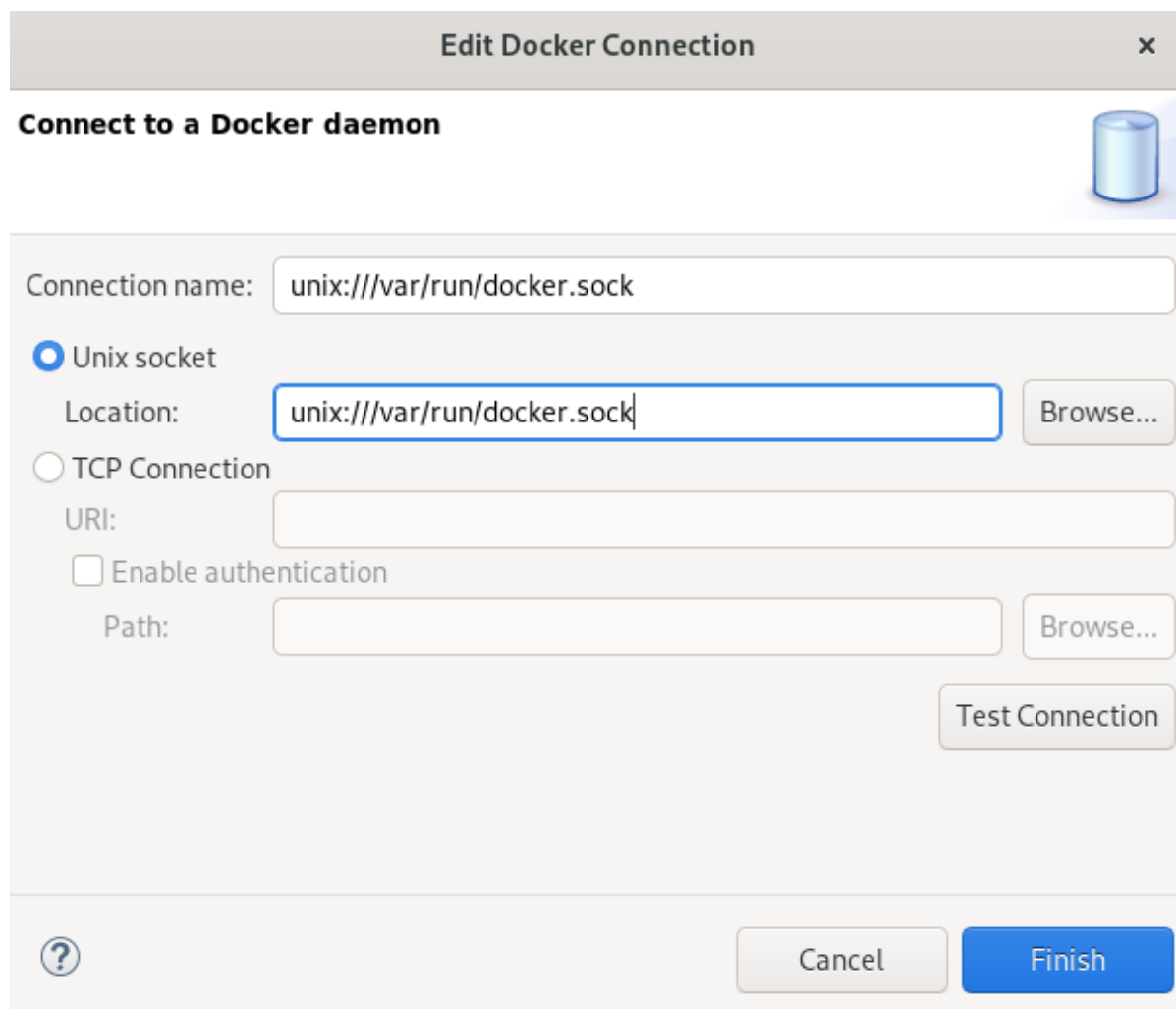
3. 検索フィールドに **Docker** と入力します。
4. **Docker Explorer** を選択します。
5. **Open** をクリックします。  
Docker Explorer ビューが表示されます。



6. **Docker socket** → **Edit** を右クリックします。



Edit Docker Connection ウィンドウが表示されます。



7. **Test Connection** をクリックします。  
コネクションが正しく設定されていれば、**Ping succeeded!** を示すウィンドウが表示されます。
8. **OK** をクリックします。

9. **Finish** をクリックします。

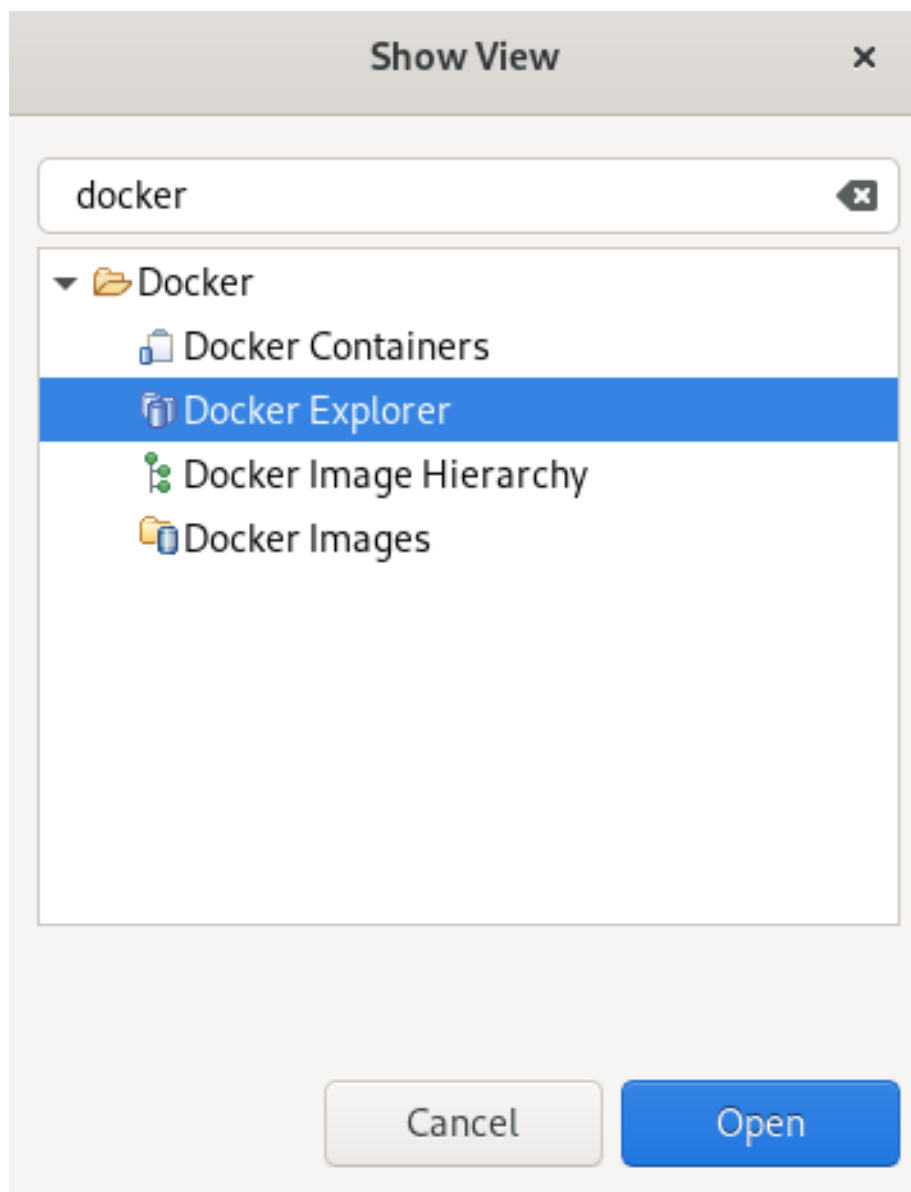
### 3.1.3. Docker コネクションの編集

#### 前提条件

- CodeReady Studio に Docker アカウントが設定されている。  
CodeReady Studio に Docker アカウントを設定する方法の詳細は、[「CodeReady Studio での Docker アカウントの設定」](#) を参照してください。

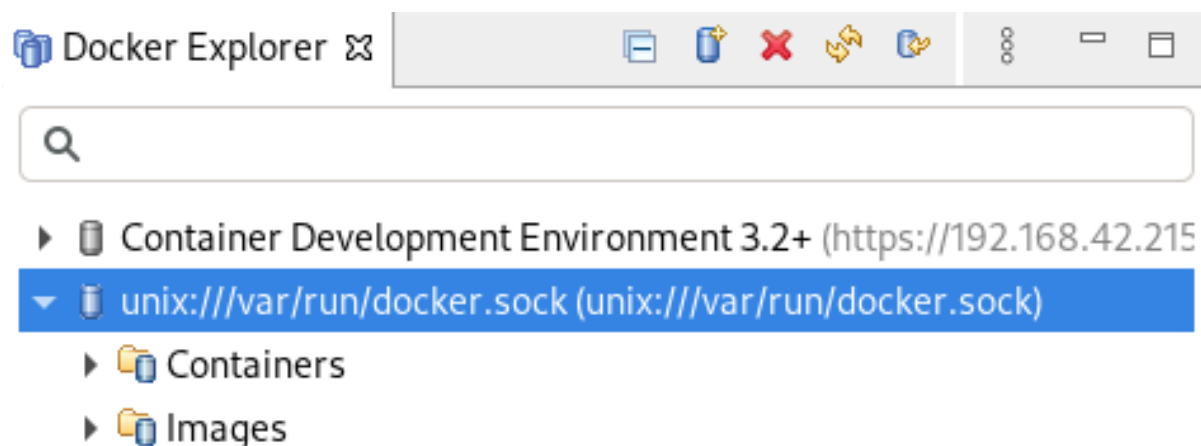
#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。

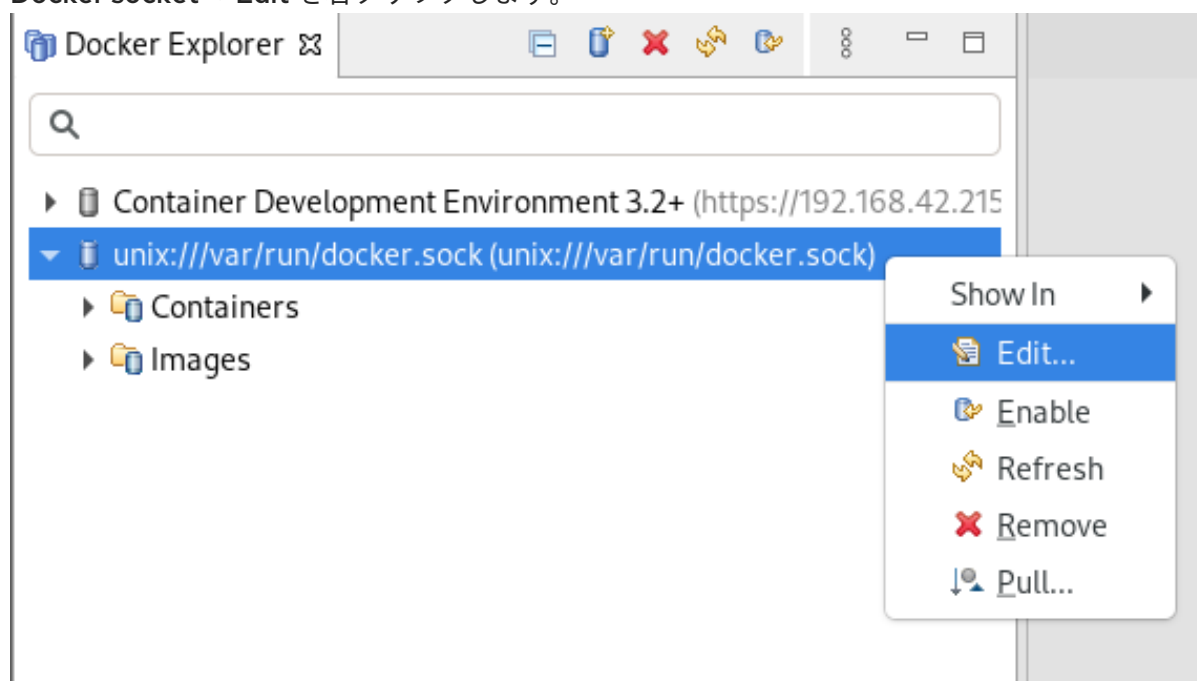


3. 検索フィールドに **Docker** と入力します。
4. **Docker Explorer** を選択します。

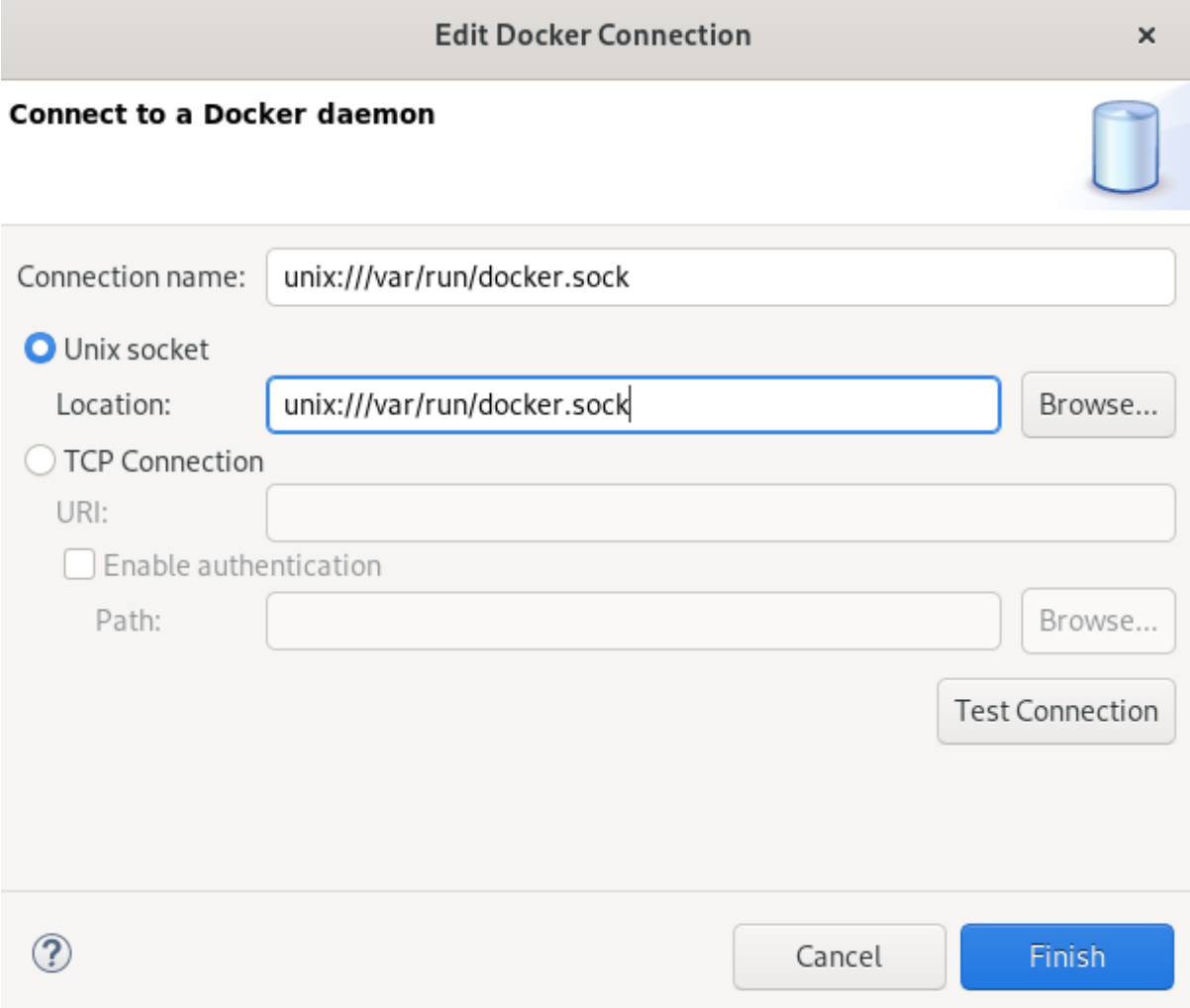
5. **Open** をクリックします。  
Docker Explorer ビューが表示されます。



6. Docker socket → Edit を右クリックします。



Edit Docker Connection ウィンドウが表示されます。



Connection name:

☒ Unix socket

Location:

☐ TCP Connection

URI:

☐ Enable authentication

Path:

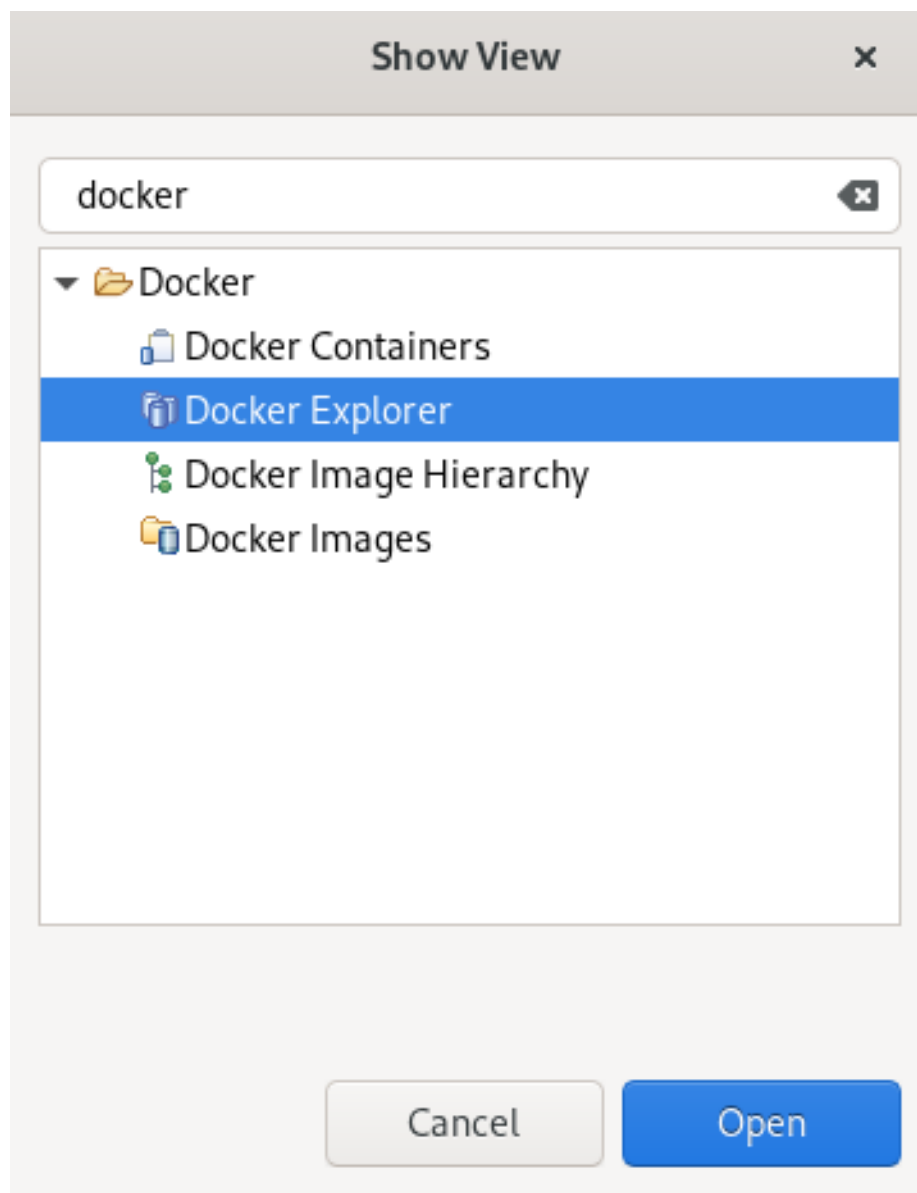
7. **Unix socket Location** フィールドで **Browse** をクリックして新しいソケットを見つけるか、**TCP Connection** オプションを選択して URI を追加します。
8. **Finish** をクリックします。

## 3.2. DOCKER イメージの管理

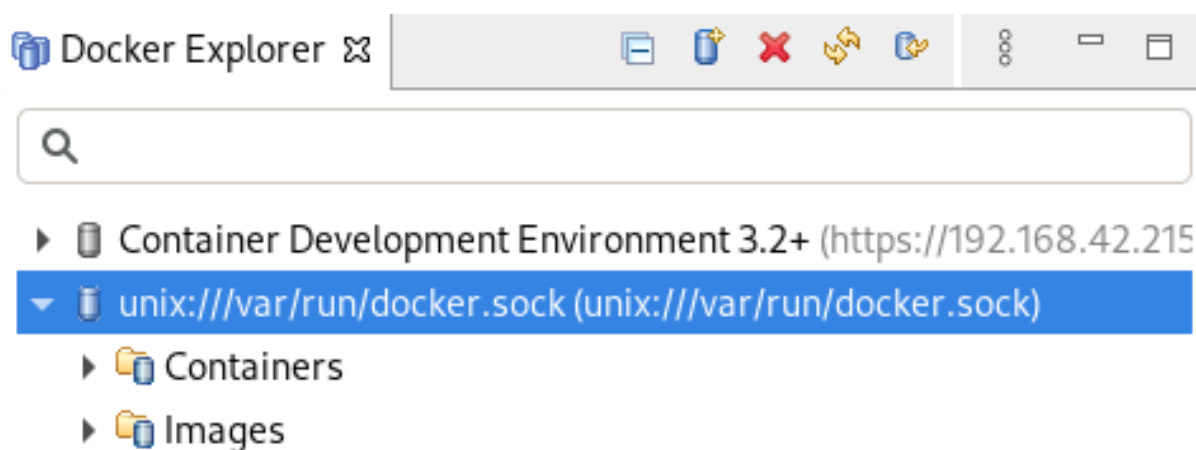
### 3.2.1. Docker イメージのプル

#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。

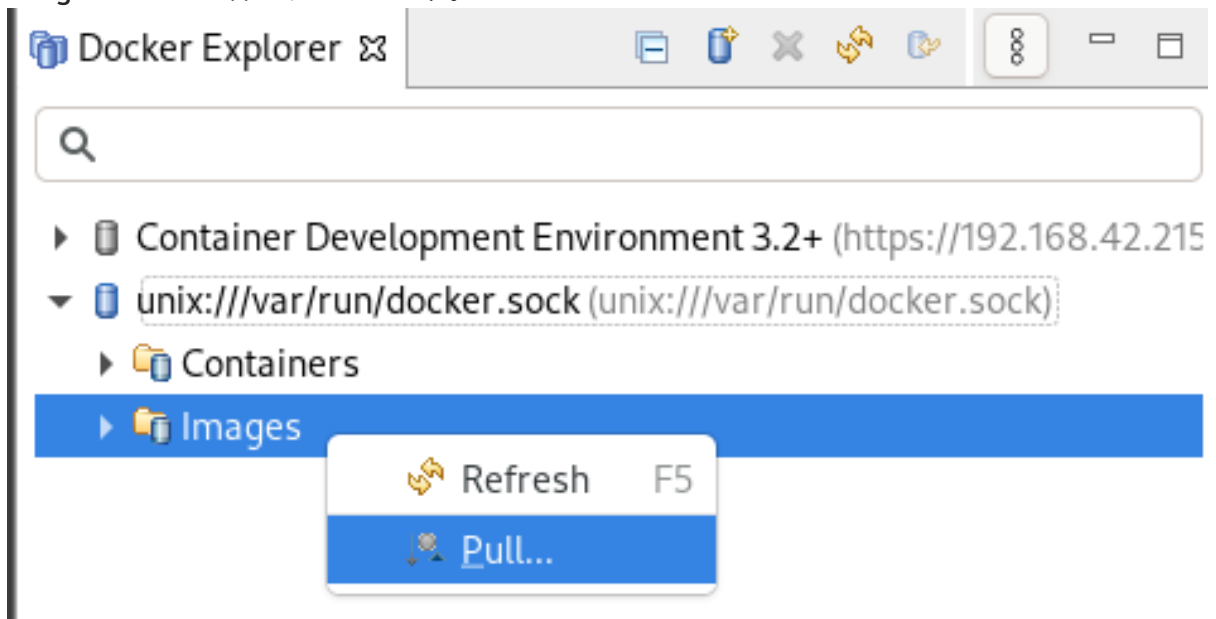


3. 検索フィールドに **Docker** と入力します。
4. **Docker Explorer** を選択します。
5. **Open** をクリックします。  
Docker Explorer ビューが表示されます。

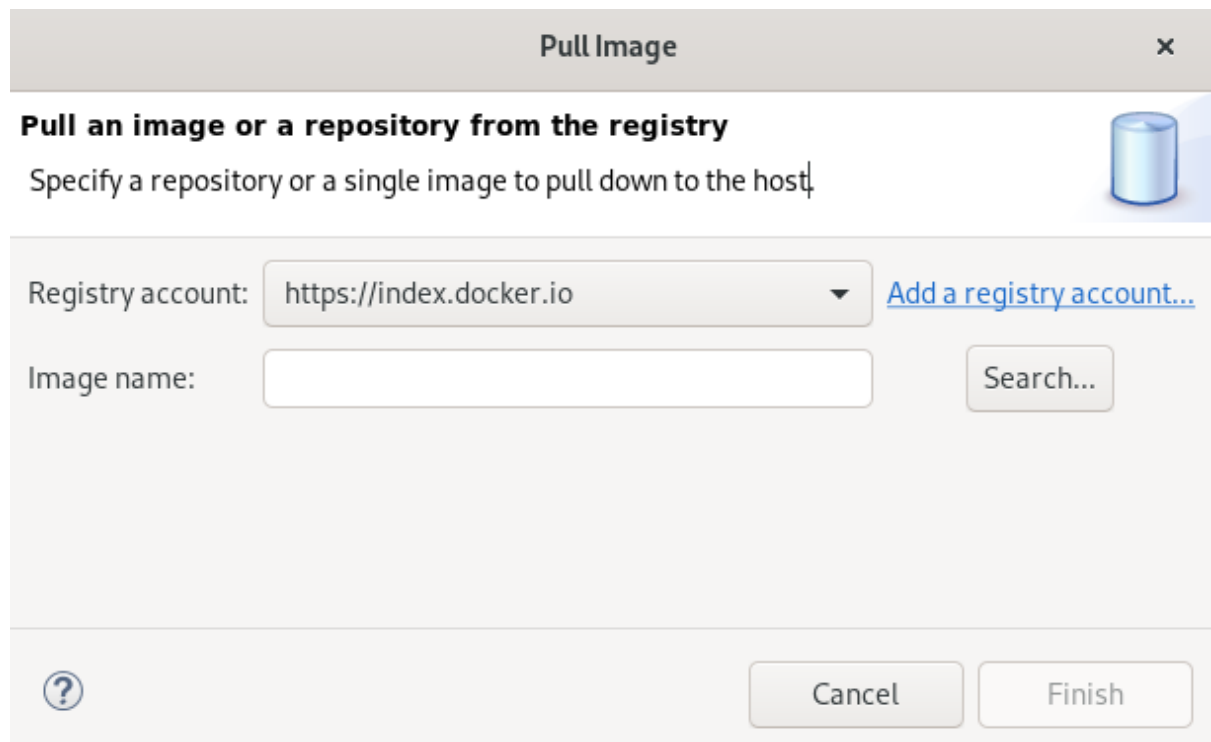


6. Docker socket フォルダーを展開します。

7. Images → Pull を右クリックします。



Pull Image ウィンドウが表示されます。



8. **Search** をクリックします。  
Search the Docker Registry for imagesウィンドウが表示されます。

Search and pull a Docker image

×

Search the Docker Registry for images






Image:  


Search

Matching images

Name	Stars	Official	Automated
jboss/wildfly	534		
openshift/wildfly-101-centos7	8		
openshift/wildfly-81-centos7	1		

Description

WildFly application server image



< Back

Next >

Cancel

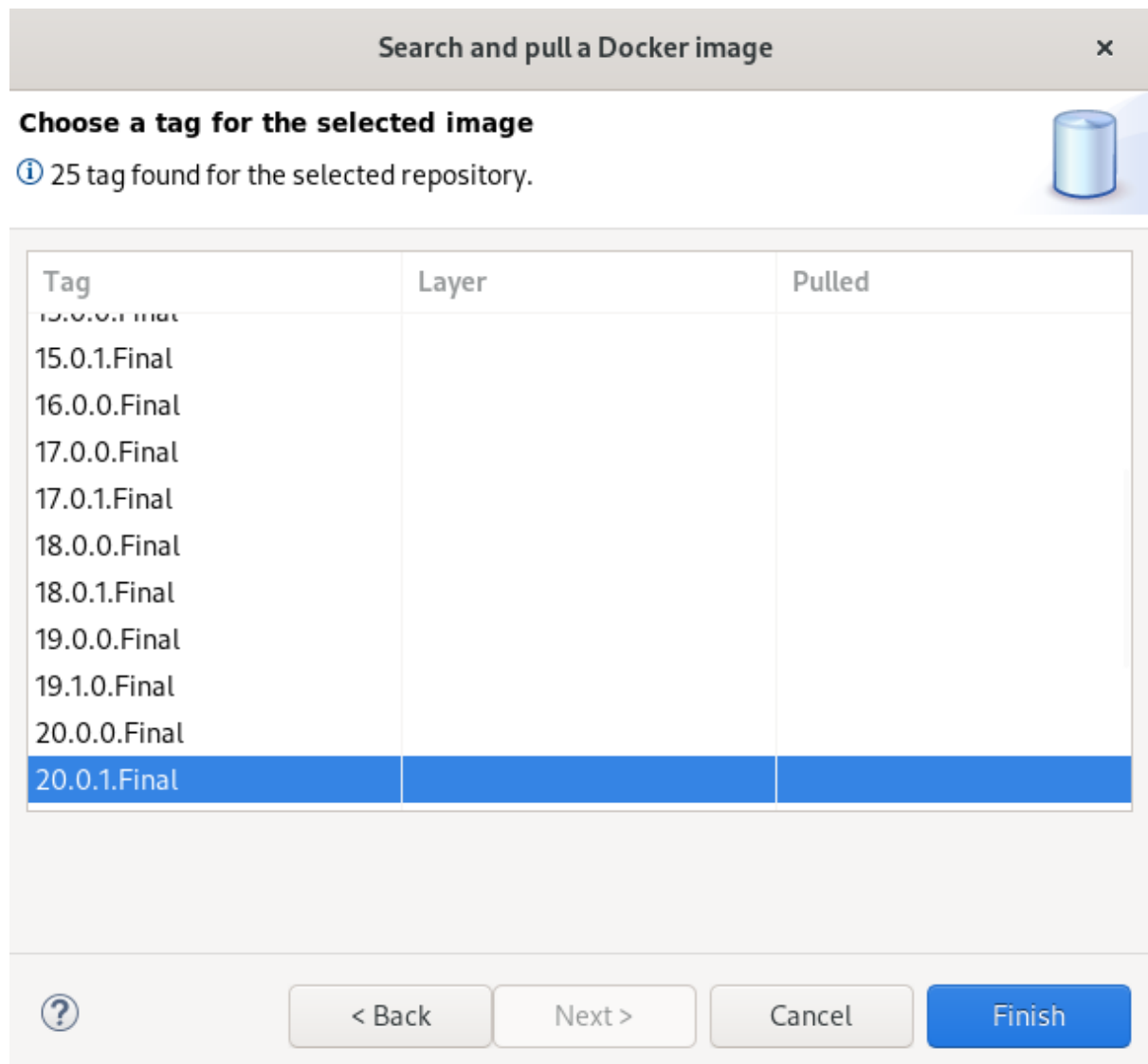
Finish

9. 検索フィールドにイメージ名を入力します。

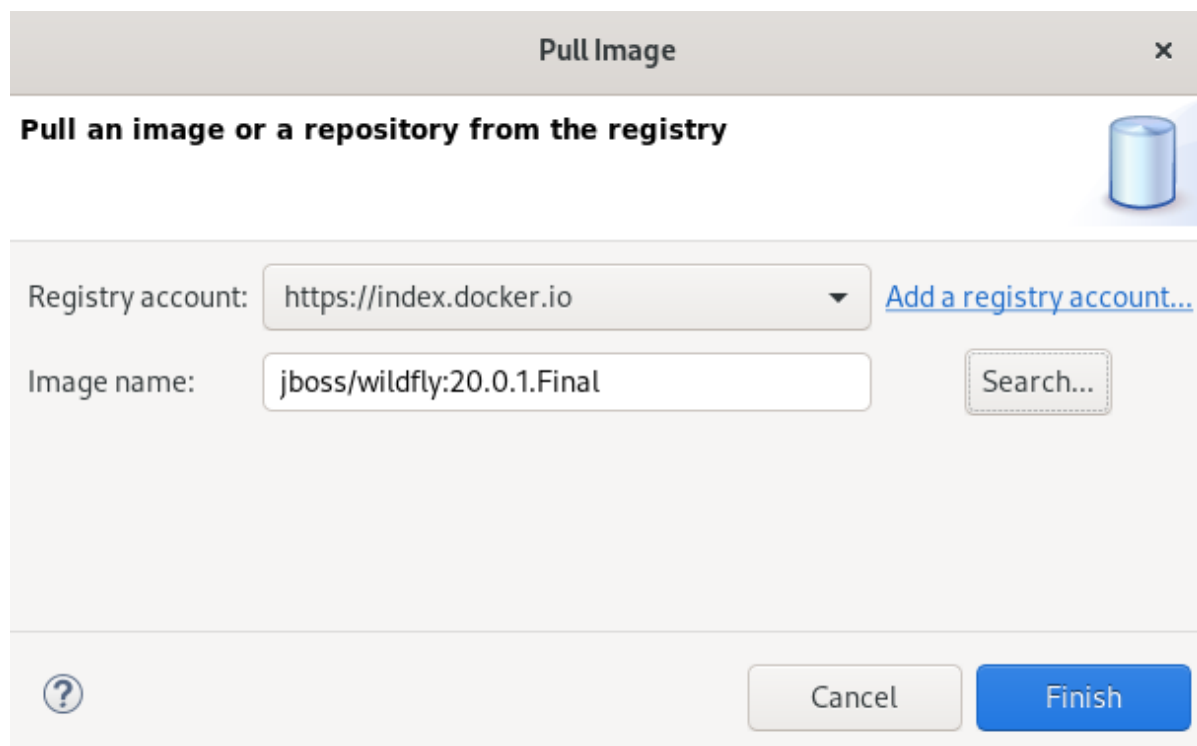
10. **Next** をクリックします。

**Choose a tag for the selected image**ウィンドウが表示されます。



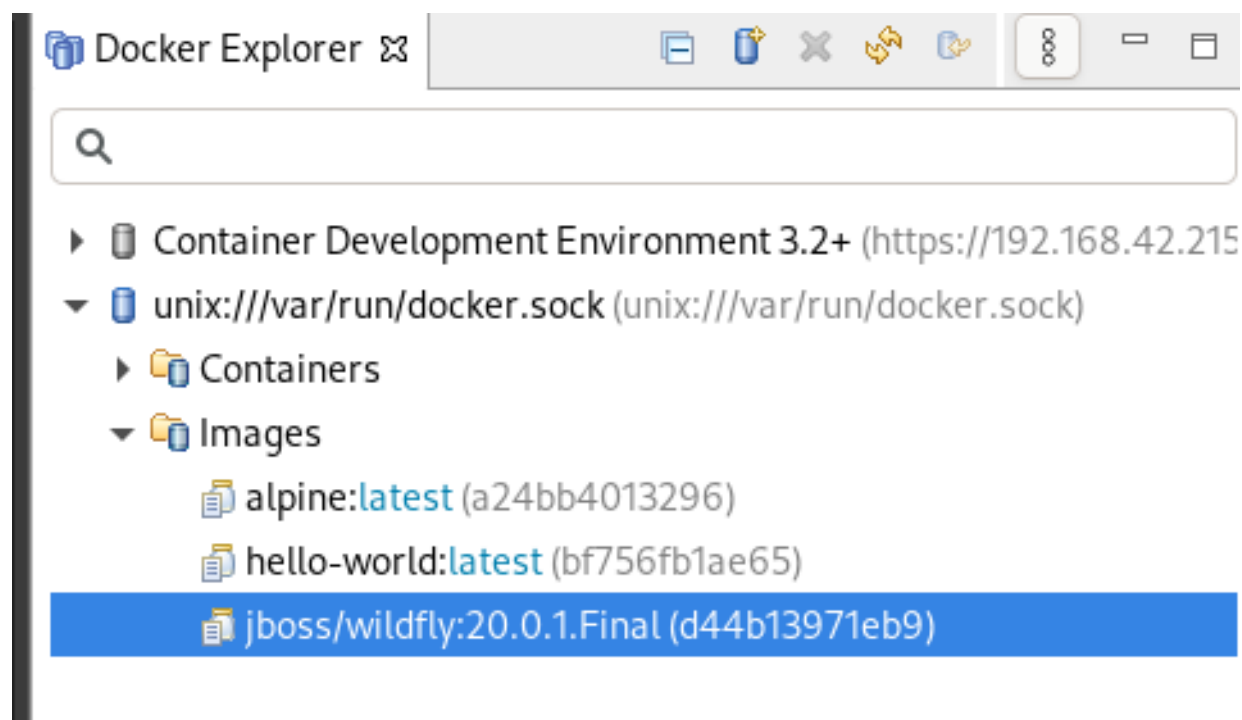


11. イメージのタグを選択します。
12. **Finish** をクリックします。  
Search the Docker Registry for imagesウィンドウが表示されます。



13. **Finish** をクリックします。

新しい Docker イメージが **Docker Explorer** ビューに表示されます。



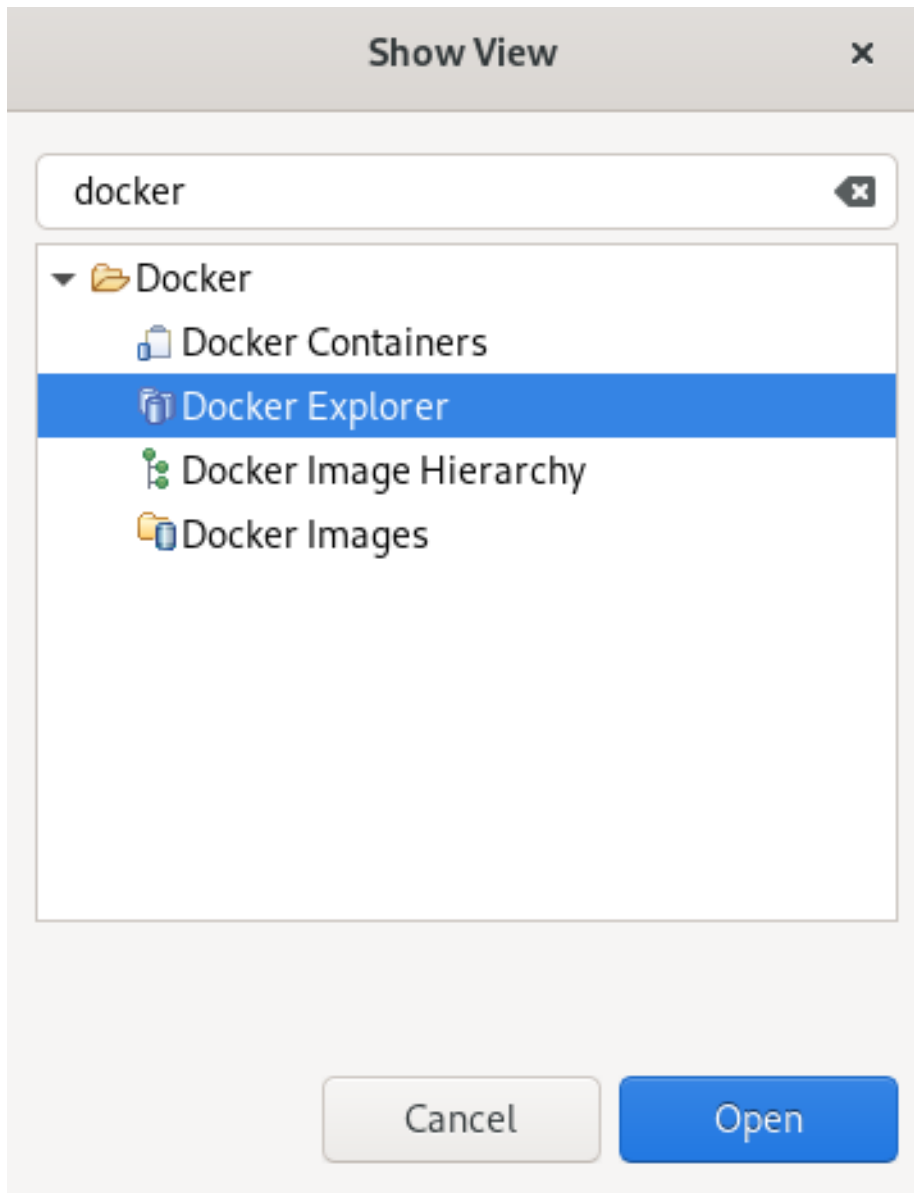
### 3.2.2. Docker イメージのプッシュ

Docker イメージをプッシュする前に、タグを付ける必要があります。CodeReady Studio で Docker イメージにタグを付け、プッシュする方法を説明します。

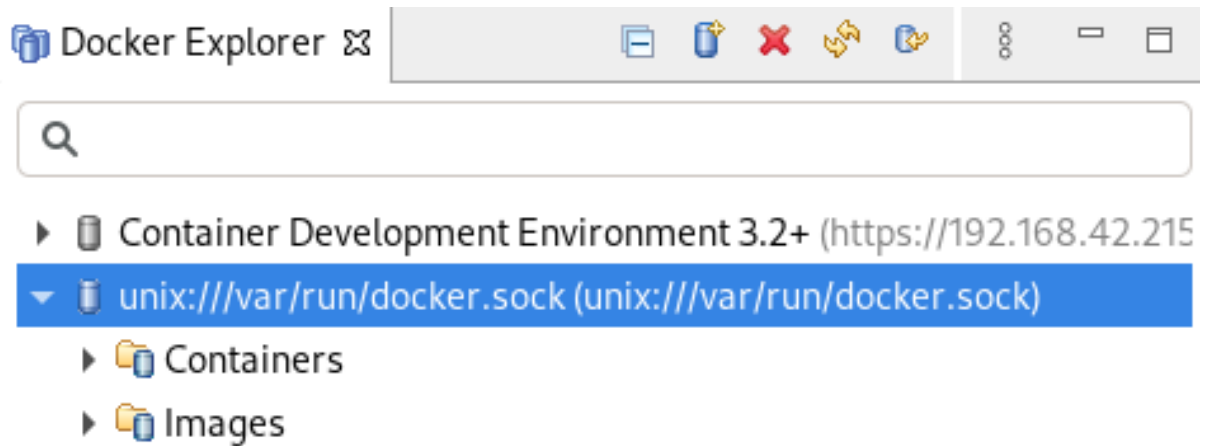
#### 手順

1. CodeReady Studio を起動します。

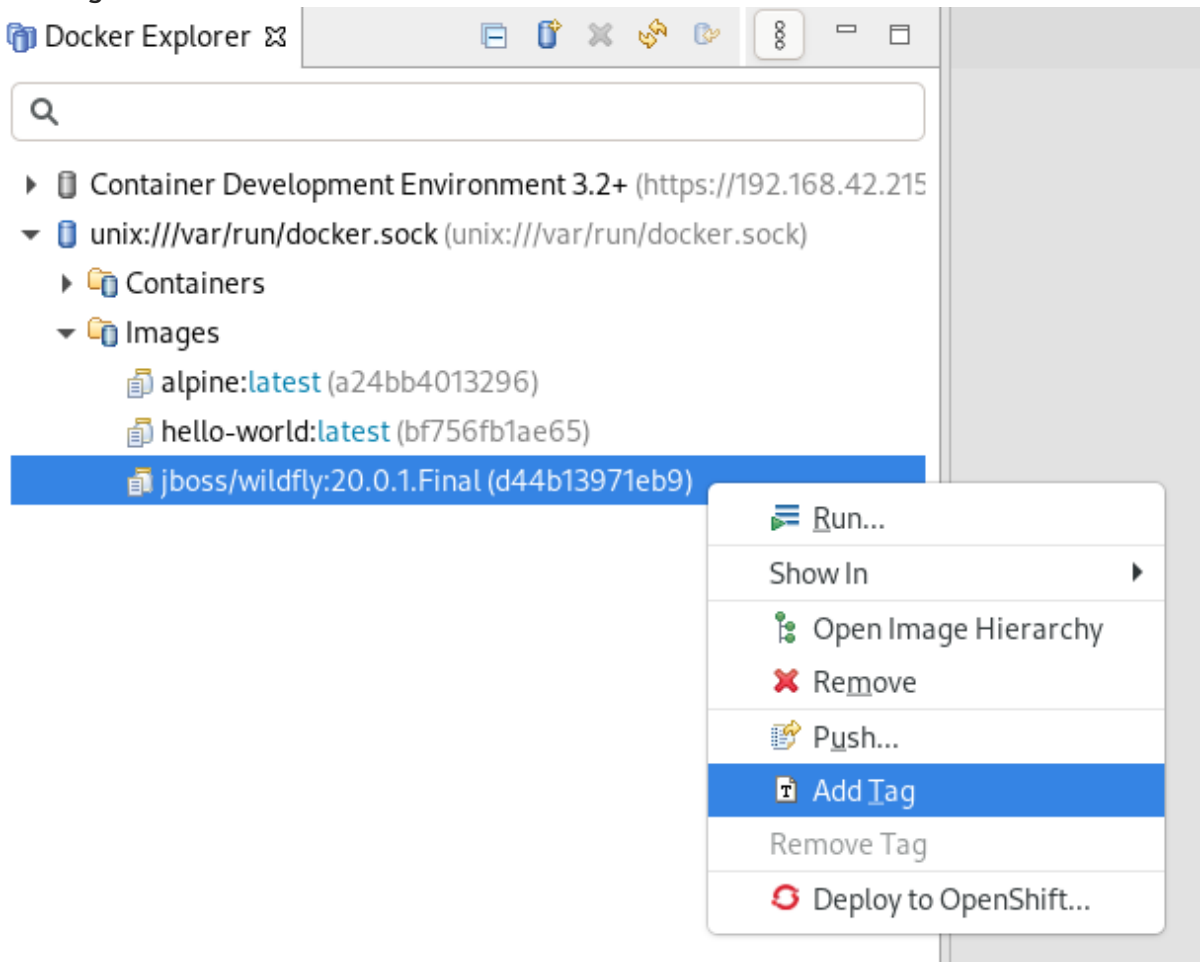
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。



3. 検索フィールドに **Docker** と入力します。
4. **Docker Explorer** を選択します。
5. **Open** をクリックします。  
**Docker Explorer** ビューが表示されます。

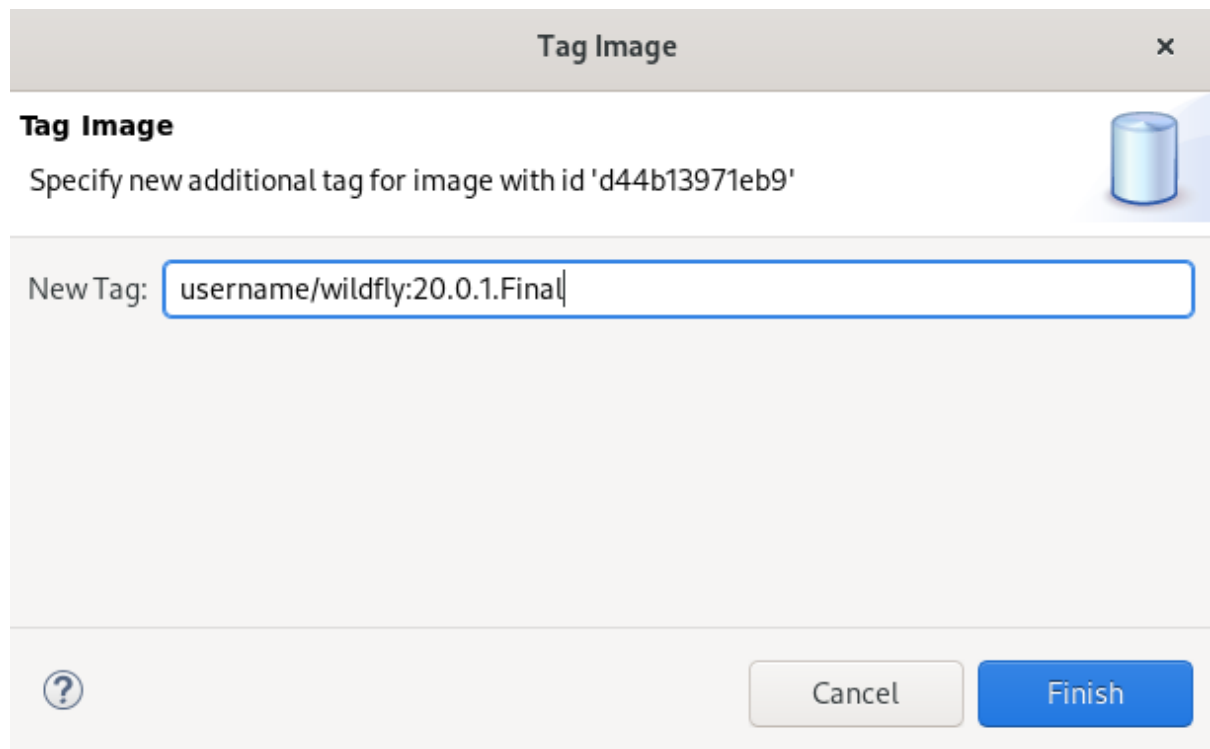


6. Docker socket → Images を展開します。
7. タグを付けるイメージを右クリックします。
8. Add tag をクリックします。

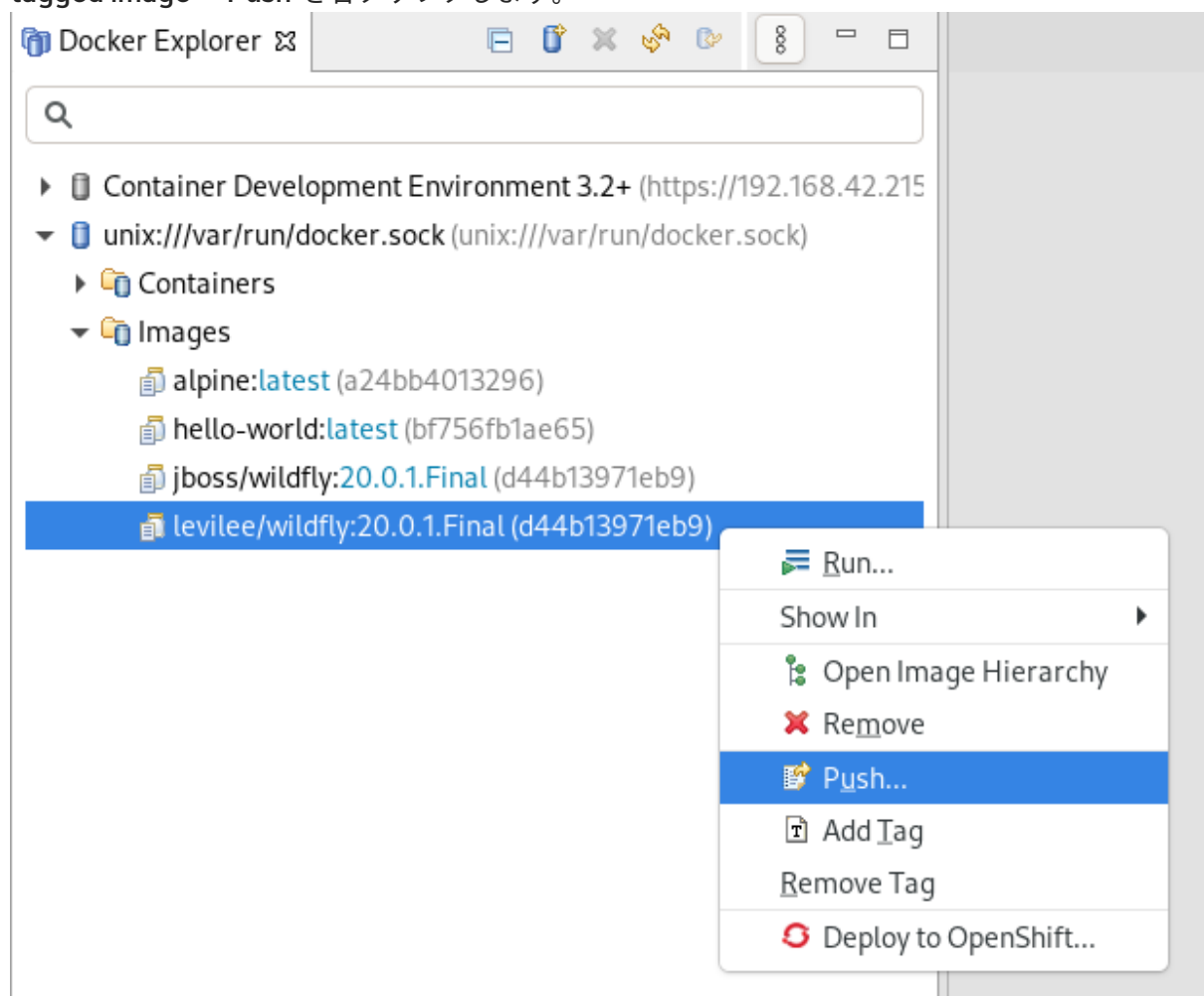


Tag Image ウィンドウが表示されます。

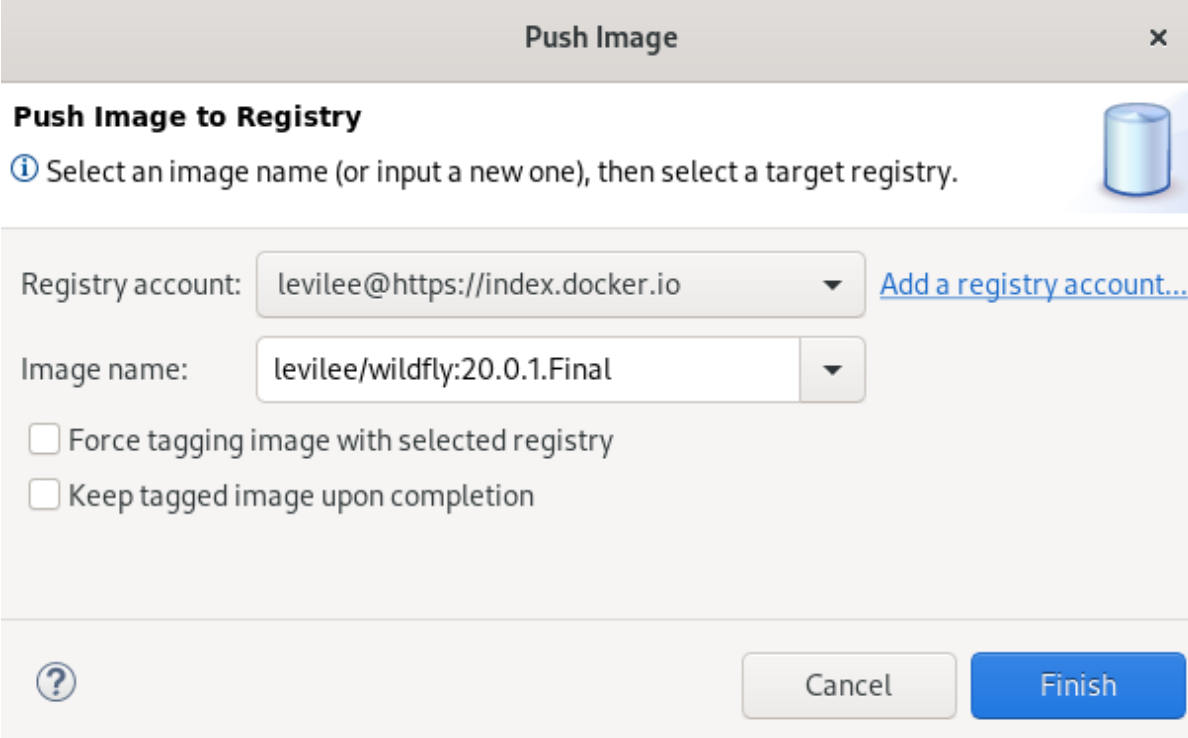
9. New Tag フィールドにタグを入力します。  
タグは **username/image\_name:tag\_name** の形式で指定する必要があります。ここで、**username** は <https://hub.docker.com> 上の Docker ID、**image\_name** はイメージの名前、**tag\_name** はイメージのバージョンに置き換えます。



10. **Finish** をクリックします。
11. **tagged image** → **Push** を右クリックします。



Push image to Registry ウィンドウが表示されます。



**Push Image**

**Push Image to Registry**

① Select an image name (or input a new one), then select a target registry.

Registry account: levilee@https://index.docker.io [Add a registry account...](#)

Image name: levilee/wildfly:20.0.1.Final

☐ Force tagging image with selected registry

☐ Keep tagged image upon completion

?

Cancel Finish

12. Docker ID で始まる **Registry Account** を選択します。

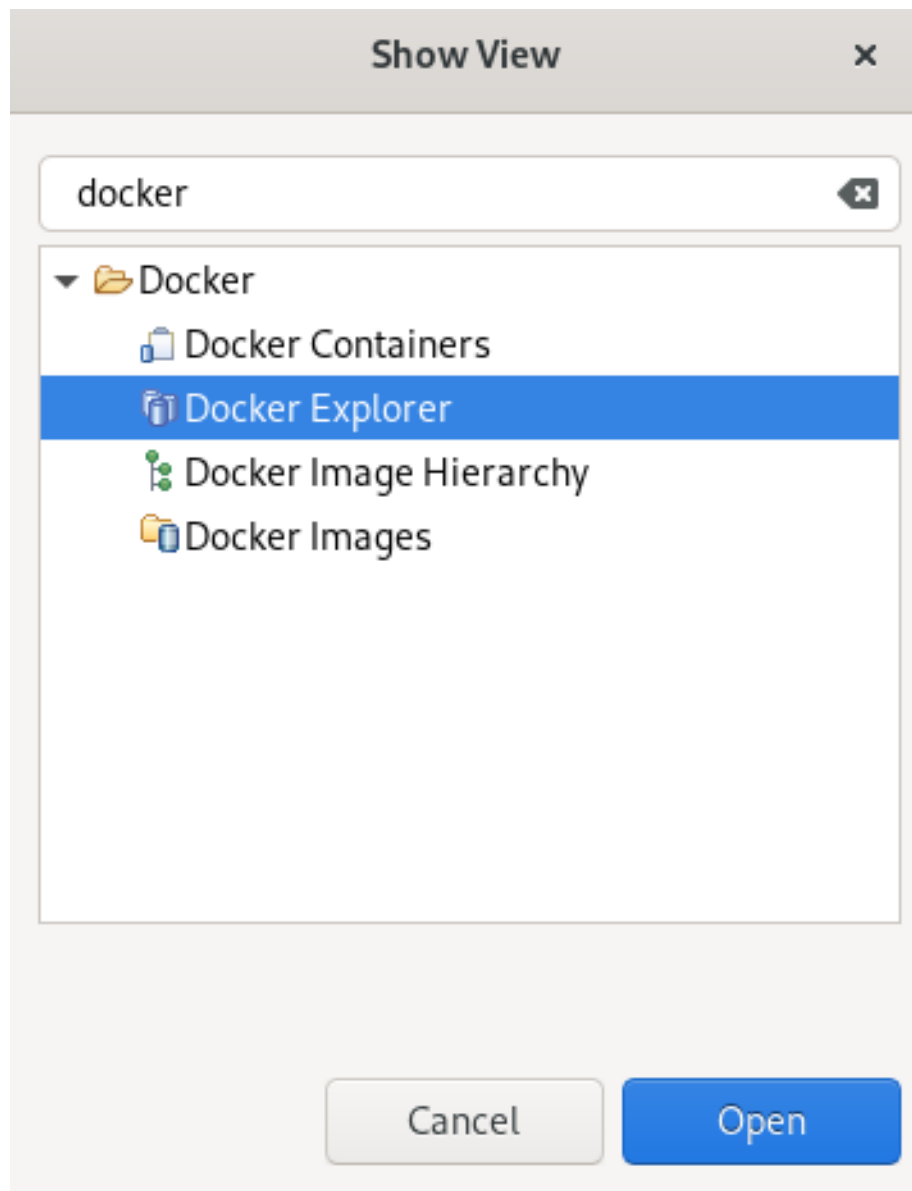
13. **Finish** をクリックします。

イメージをプッシュすると、Docker Cloud に表示されます。その後、このイメージは他の開発者が使用できるようになります。

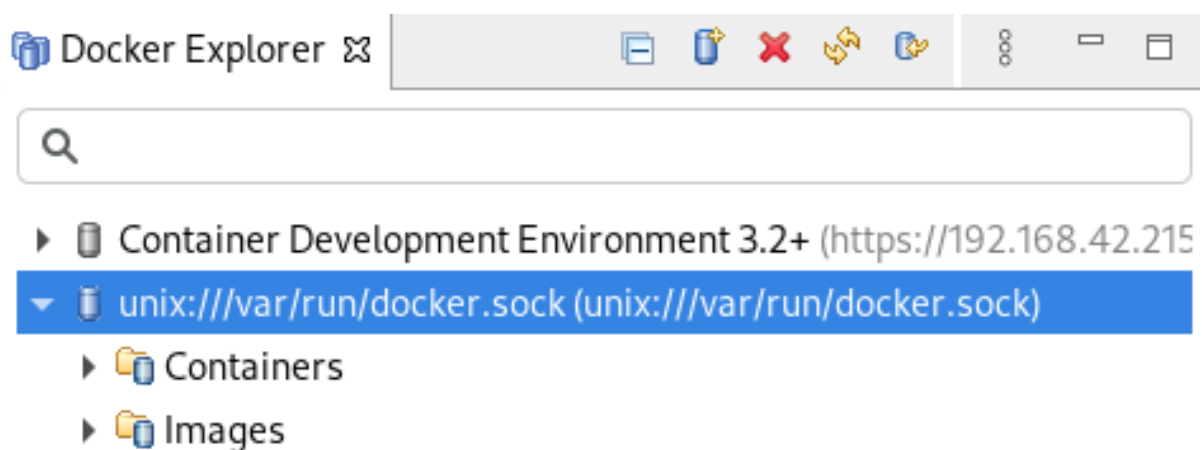
### 3.2.3. Docker イメージの実行

#### 手順

1. CodeReady Studio を起動します。
2. **Window** → **Show View** → **Other** とクリックします。  
**Show View** ウィンドウが表示されます。



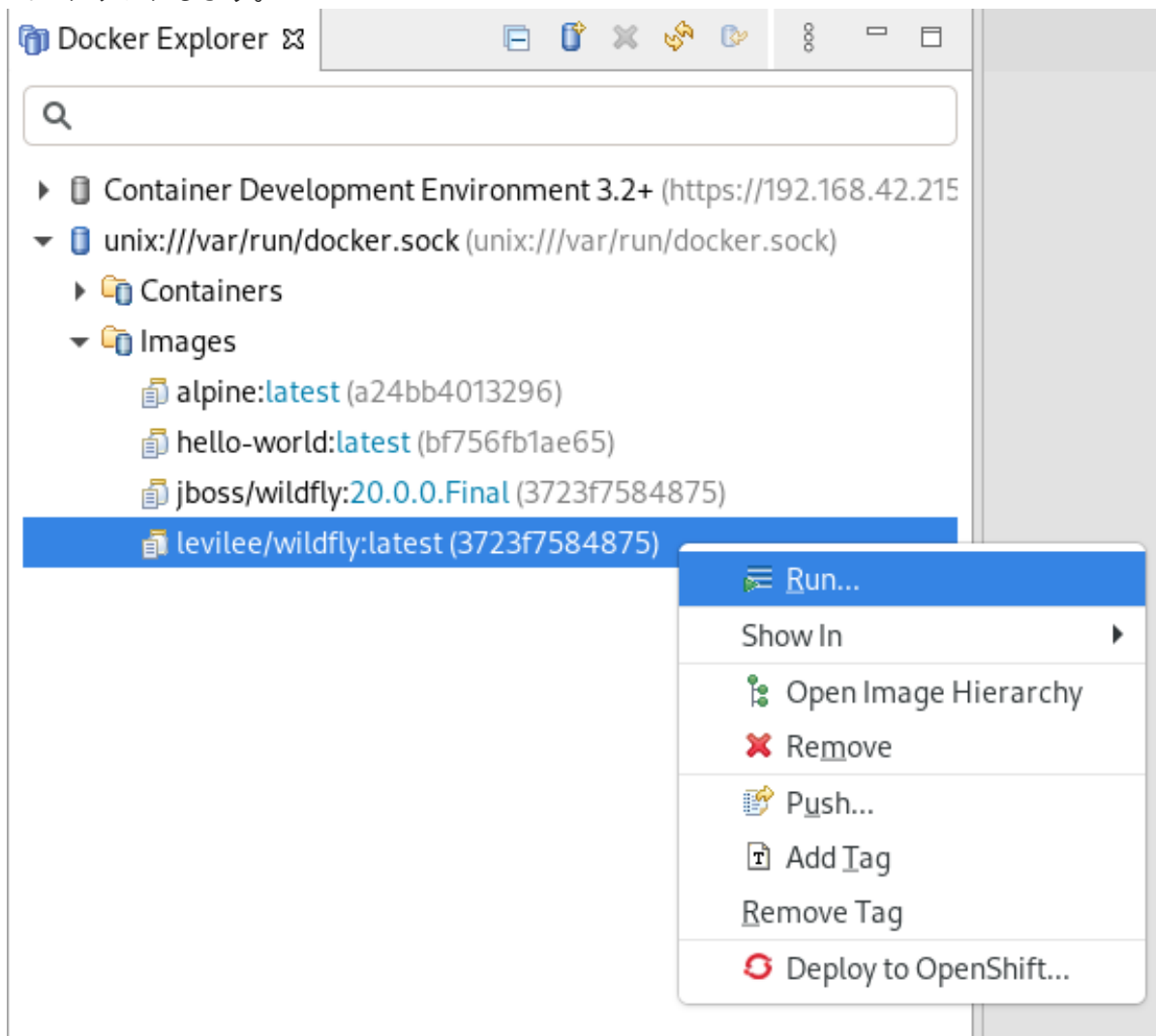
3. 検索フィールドに **Docker** と入力します。
4. **Docker Explorer** を選択します。
5. **Open** をクリックします。  
Docker Explorer ビューが表示されます。



6. Docker socket → Images を展開します。

7. 実行するイメージを右クリックします。

8. **Run** クリックします。



Docker Container settings ウィンドウが表示されます。



Run a Docker Image

×

Docker Container settings




Image:

levilee/wildfly:latest

▼

Search...

[Pull this image...](#)

Container Name:

my-docker-container

Entrypoint:

Command:

/opt/jboss/wildfly/bin/standalone.sh -b 0.0.0.0

☐ Publish all exposed ports to random ports on the host interfaces  
 Only publish the selected container ports below to the host:

Container Port	Type	Host Address	Host Port
<input checked="" type="checkbox"/> 8080	tcp		8080

Add...

Edit...

Remove

Links to other containers:

Container Name	Alias

Add...

Edit...

Remove

☐ Keep STDIN open to Console even if not attached (-i)  
☐ Allocate pseudo-TTY from Console (-t)  
☐ Automatically remove the container when it exits (--rm)  
☐ Give extended privileges to this container (--privileged)  
☐ Use unconfined seccomp profile (--securityOpt seccomp=unconfined)  
☐ Add basic security (--readonly --tmpfs /run --tmpfs /tmp --cap-drop=all)

?

< Back

Next >

Cancel

Finish

9. コンテナに名前を付けます。

10. **Publish all exposed ports to random ports on the host interfaces** チェックボックスを非選択にします。
11. **8080** ポートのチェックボックスを選択します。
12. **Finish** をクリックします。  
**Console** ビューが表示され、イメージを起動するプロセスが表示されます。
13. Web ブラウザーで <http://localhost:8080/> に移動し、実行中のイメージを確認します。



### 3.2.4. Dockerfile でのイメージのビルド

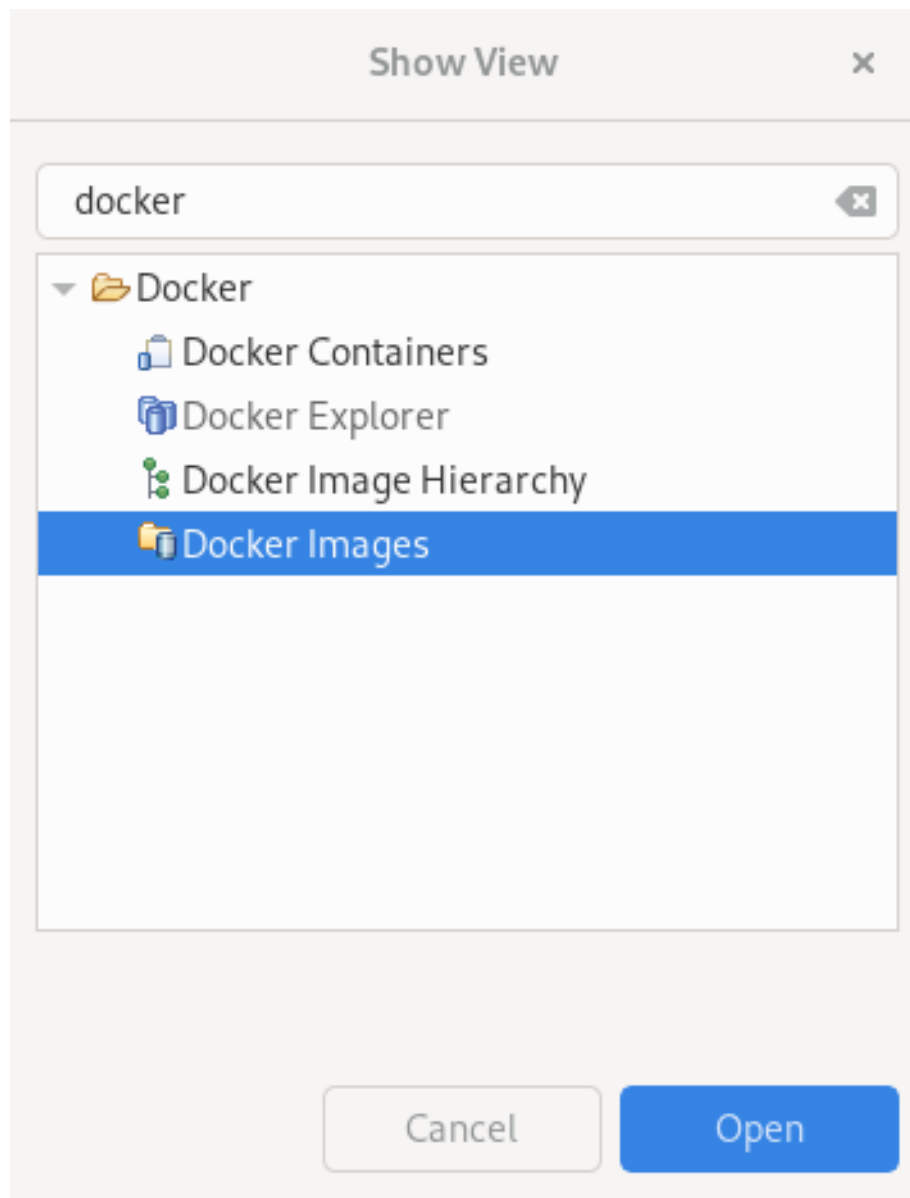
既存のイメージを変更して、イメージをビルドまたは作成できます。通常、これには新しいパッケージをインストールする必要があります。新しい Docker イメージの指定は、**Dockerfile**で行います。

#### 前提条件

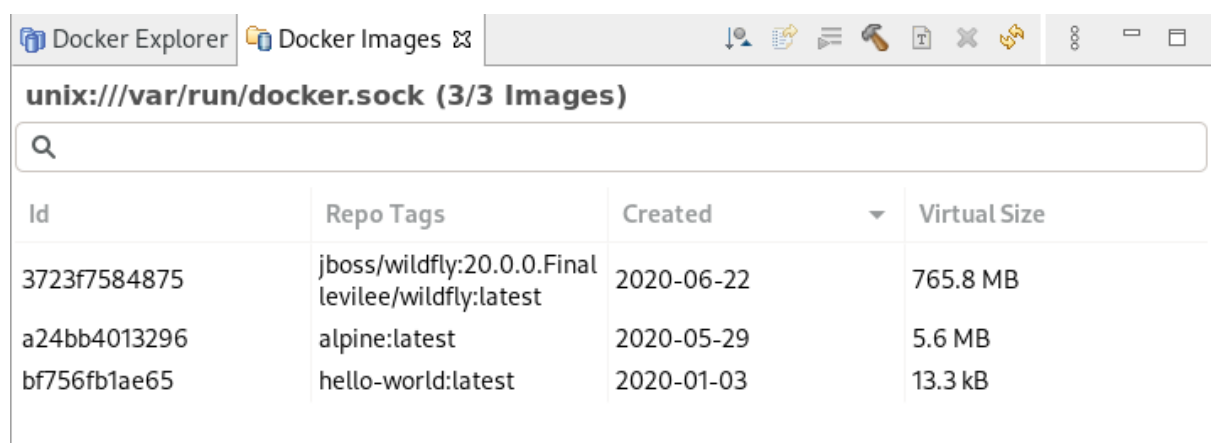
- ローカルマシンに Dockerfile が作成されている。  
Dockerfile の作成方法の詳細は、「[Dockerfile の作成](#)」を参照してください。

#### 手順

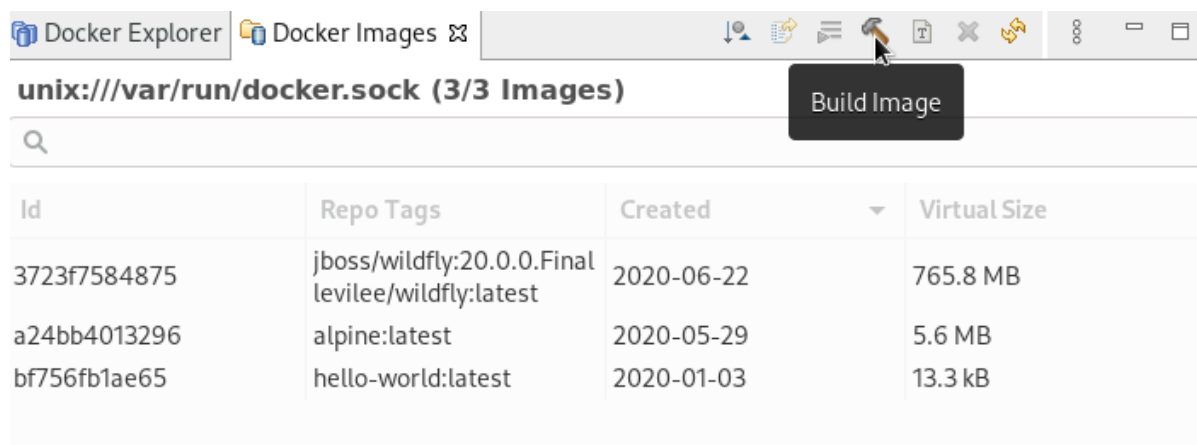
1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
**Show View** ウィンドウが表示されます。



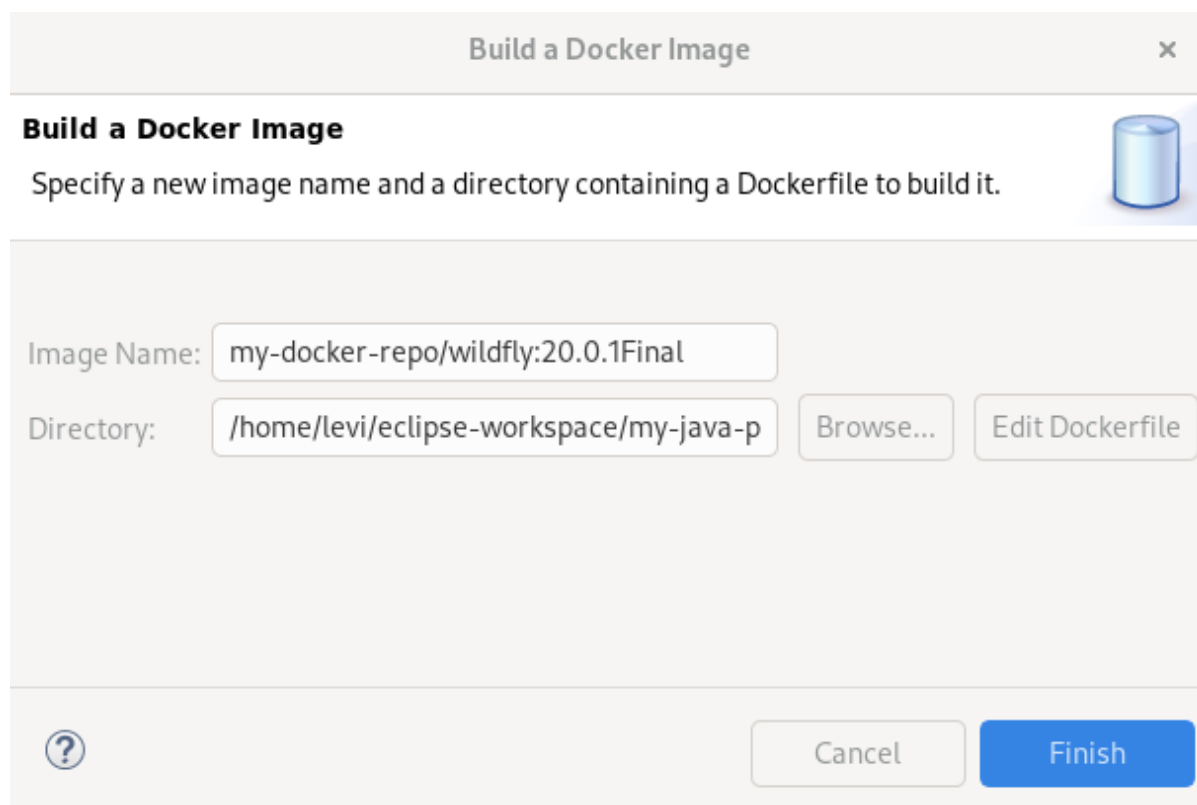
3. 検索フィールドに **Docker** と入力します。
4. **Docker Images** を選択します。
5. **Open** をクリックします。  
Docker Images ビューが表示されます。



6. イメージの **Build Image** アイコンをクリックします。



Build a Docker Image ウィンドウが表示されます。



7. **repo/name:version** の形式で、イメージに名前を付けます。
8. **Browse** をクリックして Dockerfile を見つけます。
9. **Finish** をクリックします。

Console ビューが表示され、ビルドのプロセスが表示されます。

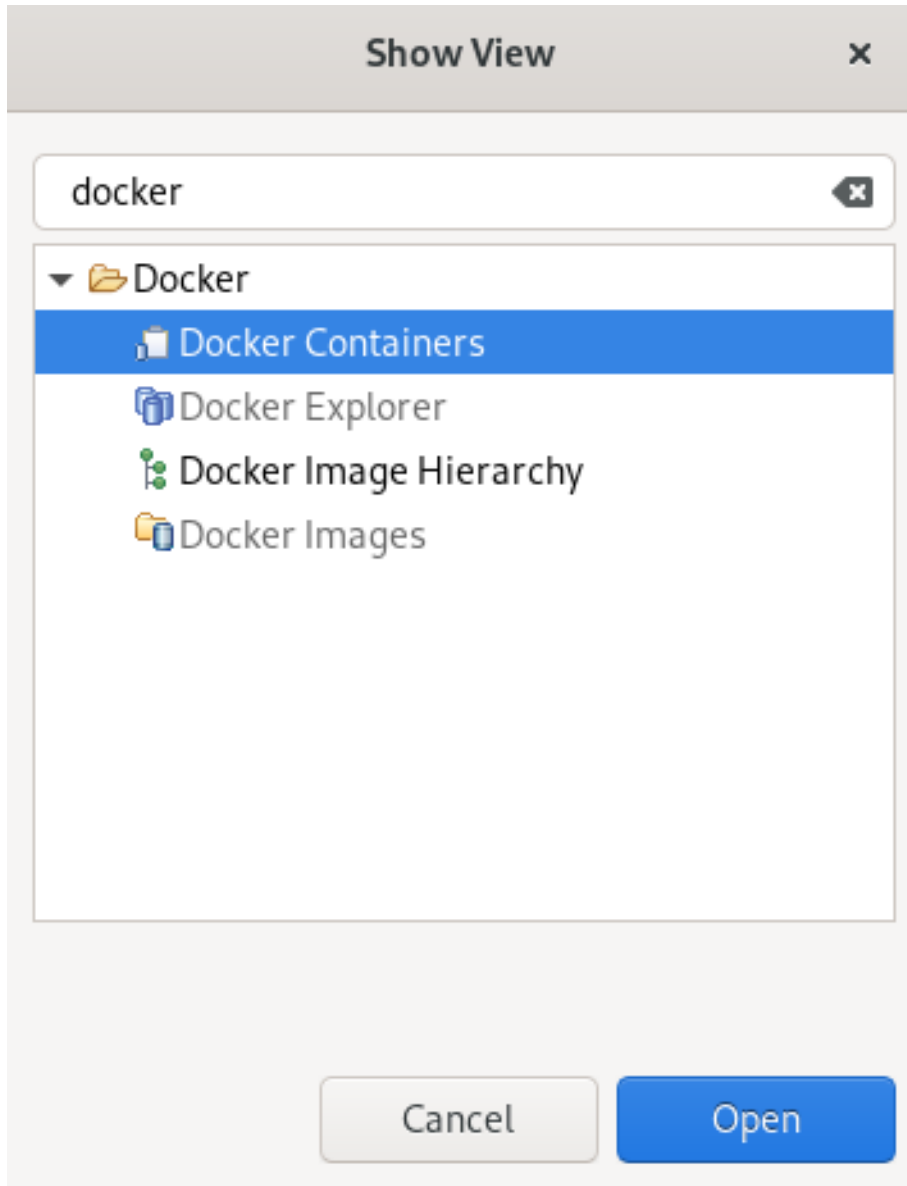
### 3.3. DOCKER コンテナの管理

Docker コンテナは、Docker イメージを基にした分離されたプロセスです。コンテナの作成後、ユーザーはコンテナを停止、起動、一時停止、一時停止解除、kill、または削除できます。また、コンテナのログの読み取りも可能です。

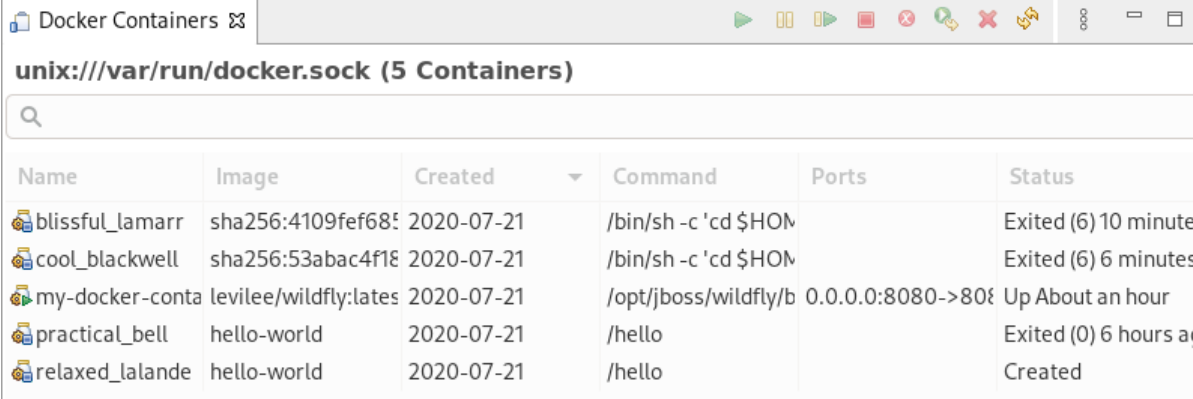
CodeReady Studio で Docker コンテナを管理する方法を説明します。

#### 手順

1. CodeReady Studio を起動します。
2. **Window → Show View → Other** とクリックします。  
Show View ウィンドウが表示されます。



3. 検索フィールドに **Docker** と入力します。
4. **Docker Containers** を選択します。
5. **Open** をクリックします。  
**Docker Containers** ビューが表示されます。



Name	Image	Created	Command	Ports	Status
blissful_lamarr	sha256:4109fef685	2020-07-21	/bin/sh -c 'cd \$HOM		Exited (6) 10 minute
cool_blackwell	sha256:53abac4f18	2020-07-21	/bin/sh -c 'cd \$HOM		Exited (6) 6 minutes
my-docker-conta	levilee/wildfly:lates	2020-07-21	/opt/jboss/wildfly/b	0.0.0.0:8080->808	Up About an hour
practical_bell	hello-world	2020-07-21	/hello		Exited (0) 6 hours ag
relaxed_lalande	hello-world	2020-07-21	/hello		Created

パネルを使用して、コンテナを起動、一時停止、一時停止解除、停止、kill、再開、削除、または更新できます。

