Importing an existing codebase into CodeReady Toolchain

Red Hat Developers Documentation Team

2018-12-11 13:07:31 UTC

Table of Contents

This document provides instructions on importing your existing codebases to CodeReady Toolchain. In this tutorial we import one of the existing sample repositories in GitHub.

Creating a new space and importing an existing project into CodeReady Toolchain

This section contains instructions for adding an existing project to CodeReady Toolchain instead of creating a new project using quickstarts.

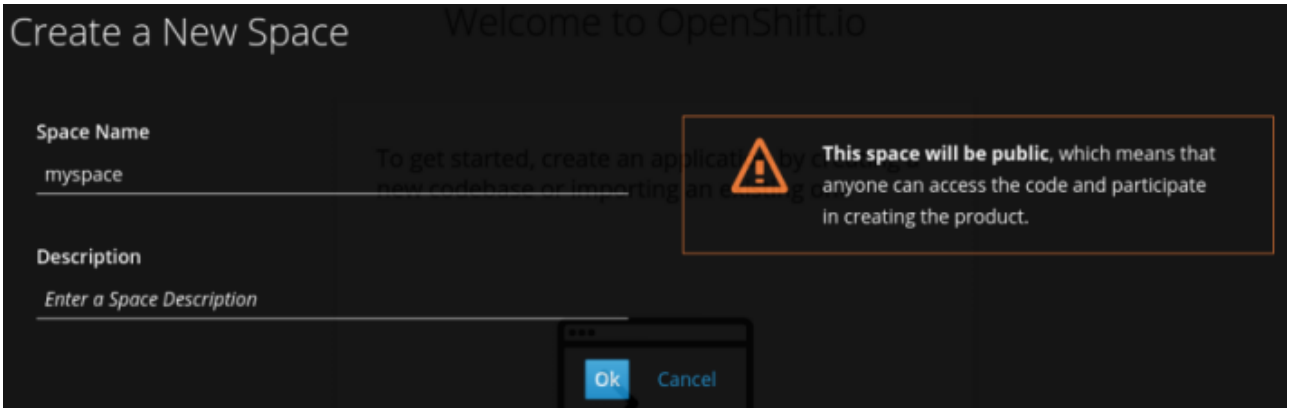
Chapter 1. Creating a new space

In CodeReady Toolchain, the first step for any new project is to create a new space with a unique name. You can then create or import codebases within the space, add collaborators, and plan your work using work items and iterations.

Procedure

Create a new space as follows:

1. In the CodeReady Toolchain home page, click **[Create a space]** in the **Recent Spaces** section. If your account has an existing space, click + to create an additional space.
2. In the **Create a New Space** dialog box, type **mynewspace** as the unique name for your space and click **[Ok]**.



You have now created your first space.

Chapter 2. Forking an example repository

In addition to creating applications using quickstarts, you can import an existing project into CodeReady Toolchain.

Prerequisites

Reset your CodeReady Toolchain environment by following instructions in resetting your CodeReady Toolchain account.

Procedure

- You can fork any one of the sample repositories in GitHub. For this tutorial, fork and use the `vertx-eventbus` repository.

Sample repositories available in Github:

- https://github.com/burrsutter/vertx-eventbus
- https://github.com/burrsutter/vertx-paint
- https://github.com/burrsutter/vertx-wiki2

If needed, you can import multiple projects into CodeReady Toolchain simultaneously, but this can result in a 12-15 minute delay. Import one project at a time to avoid delays.

The OpenShift Online Free Tier provides two pods, and each project requires one pod.

Chapter 3. Importing your project

After creating a space, use the **Create an Application** wizard to import an application. If you closed the wizard earlier, click **[Add to space]** in the upper-right corner of your space dashboard.

1. In the **Name your application** field, type a unique name for your new project.
2. Select the **Import an existing codebase** radio button and then click **[Continue]** to import an existing project codebase into CodeReady Toolchain.
3. In the **Authorize Git Provider** step:
    a. Click the **Location** drop-down to select the location of your codebase. The default option is your personal GitHub account name.
    b. In the **Repository** field, click **Select Repository** to select the repository from which you want to import the codebase.
    c. Click the blue arrow at the bottom of the screen to continue.
4. In the **Select pipeline** step, select an option for your pipeline build, then click the blue arrow to continue to the next step. We recommend using the first option for most use cases because it provides an end to end pipeline that is suitable for most projects.
5. The **Confirm Application Summary & Import** step displays a summary for your imported code. Review the information and click **[Import Application]** to confirm. The progress screen displays when the code is successfully imported.
6. Click **[View New Application]** to continue when all the steps have a green check mark next to them.

You have now imported the code from your existing repository to CodeReady Toolchain.

Now that you have imported the existing project into CodeReady Toolchain, you can change the code and stage changes.

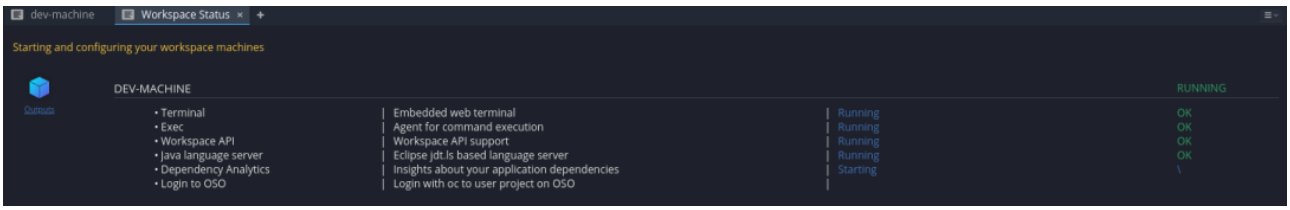
Chapter 4. Creating a Che workspace

To edit your application codebase, create a new Che workspace:

1. Click **Create** from the top of the CodeReady Toolchain page. The default view for this tab is **Codebases**.
2. In the **WORKSPACES** column, click **Create workspace** for your project. The workspace gets created.
3. Click **[Open]** next to the workspace to see your Che workspace in a new browser tab.

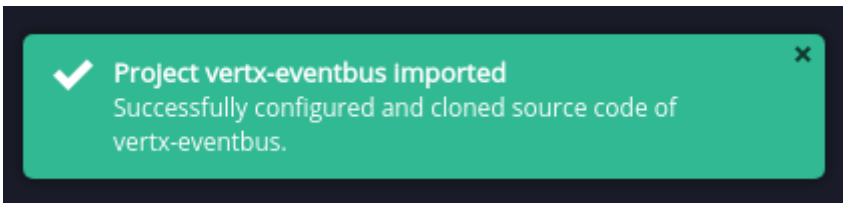
If a new tab does not appear, see enable_popups for troubleshooting information.

As the workspace loads the codebase for your application, the **Workspace Status** window at the bottom of the Che workspace tab displays the progress:



If the **Workspace Status** shows **Stopped**, click **[Start]** in the **Workspace is not running** pane at the top, to restart your workspace.

When loaded and ready to use, the new Che workspace tab displays the following confirmation message:



Chapter 5. Running your project in the Che workspace

After your Che workspace loads, you can see your project code listed in the file explorer panel, in the upper-left side of the screen.

1. Click the **run** option from the Run button (

). Maven then downloads the required dependencies, compiles the application, and starts the *verticle* (Vert.x uses this name for deployed code). For Vert.x projects, this also sets up the server and hot deploy options. The hot deploy option automatically updates the application when you make a change.

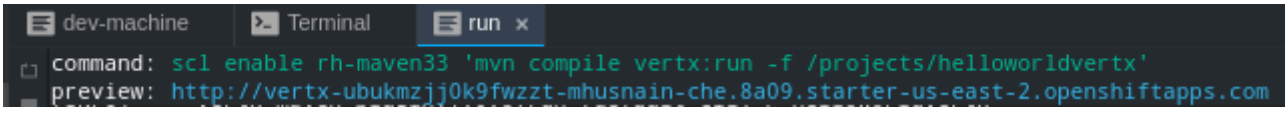
If the **Run** command macro is not defined, it displays the **Create run command** option instead of **Run**:

See create_run_command_macro for instructions on setting up a run command.

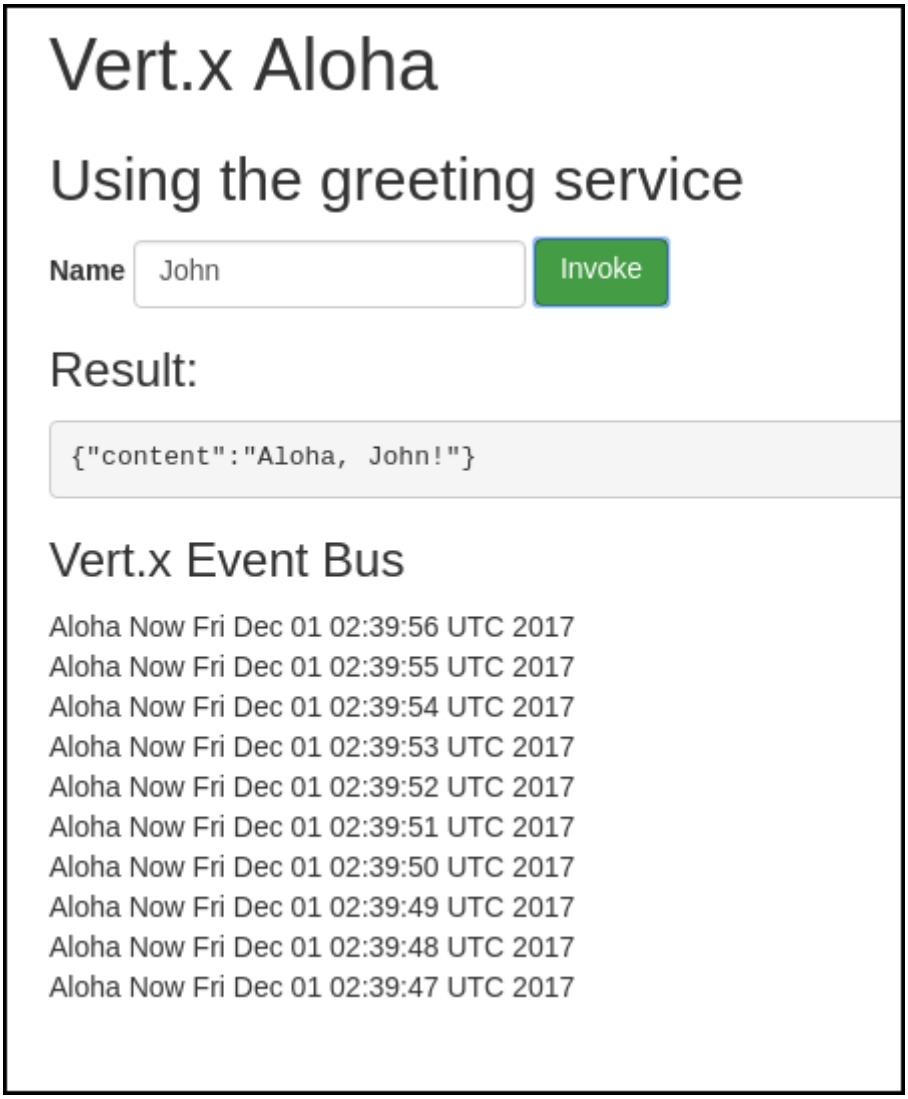
2. A **run** terminal appears at the bottom pane of the Che workspace. When the `mvn build` command finishes executing, the **run** view displays the following message:

```
[INFO] INFO: Succeeded in deploying verticle
```

3. Click the blue preview URL at the top of the **run** view to see your application running in Che.



4. In the application, enter a name in the **Name** field and click **[Invoke]** to test the application.



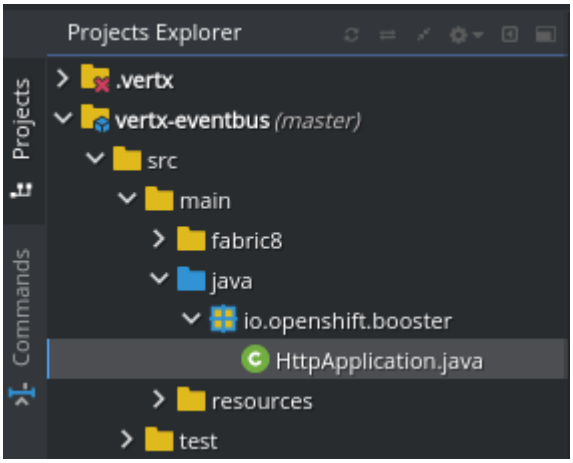
This is the same version of the application that the pipeline deployed to **Stage** and you subsequently promoted to **Run**. The URL for this build of the application is different from the URLs used by OpenShift Online for **Stage** and **Run**. This is your private sandbox hosted within Che. You can still share this URL with others and interactively debug the application while they run it in their browser.

Chapter 6. Changing the project code

The example project code includes a new route for **eventbus** and its supporting method. The project sets up **eventbus** to only allow outbound communications for the **my-feed** channel. Messages arriving in the **my-feed** channel are pushed to the browser using the SockJS bridge. This project includes a `vertx-eventbus.js` file along with the `index.html` file and also includes tweaks to the `index.html` file.

Change the code of your project and preview the results as follows:

1. In the file directory view, double-click the following directories to expand them: `src` > `main` > `Java`> `io.openshift.booster` > `HttpApplication.java`.



2. Double click the `HttpApplication.java` file to open it and find the following line:

```
protected static final String template = "Aloha, %s!";
```

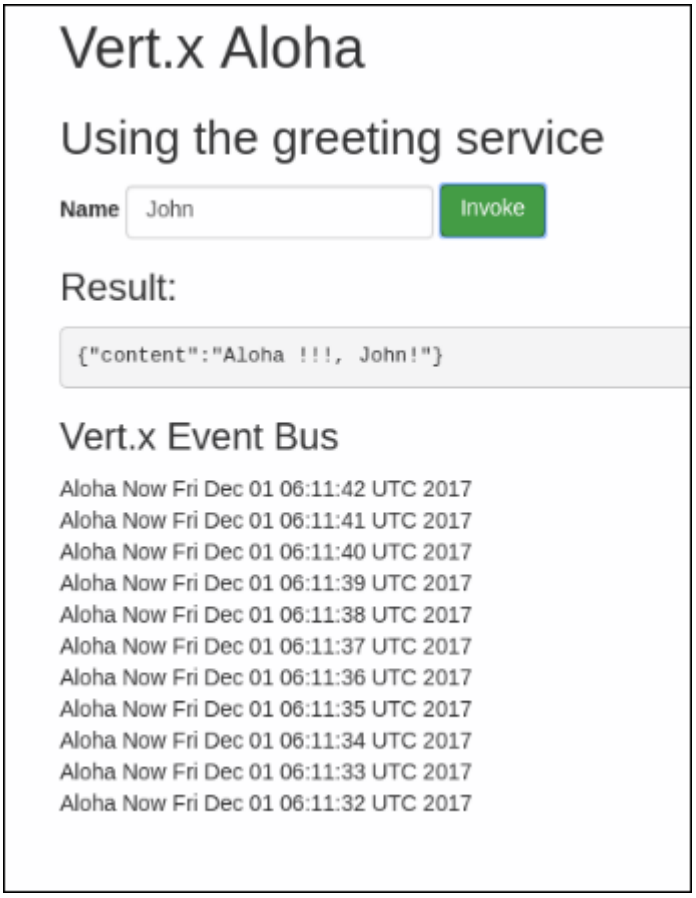
3. Add exclamation marks to the string. The line should now be:

```
protected static final String template = "Aloha !!!, %s!";
```

4. Save the changes (`Ctrl+s` or `Cmd+s` for macOS).

   If you already ran the application earlier as instructed in Running your project in the Che workspace, your changes are instantly implemented. Maven uses the Vert.X hot deploy feature to automatically update the application when you make a change.

5. Return to the browser tab running the application, type a name in the text box and click **[Invoke]** to test the application.



You have now changed your application code and tested the change.

Chapter 7. Committing and pushing changes to GitHub

After making the required changes to your code, commit and push the modifications to your project GitHub repository.

Before committing your changes, ensure that your project pipeline build is promoted and at the **Rollout to Run** stage.

1. In your Che workspace, click **Git** from the menu bar options and select **Commit** from the displayed options.
2. In the **Commit to repository** dialog box:
    a. If they are not already, select all the changed and new files to add them to the commit.
    b. Add a commit message describing your changes.
    c. Select the **Push committed changes to** check box.
    d. Click **[Commit]**.

When the commit succeeds, the following message displays:

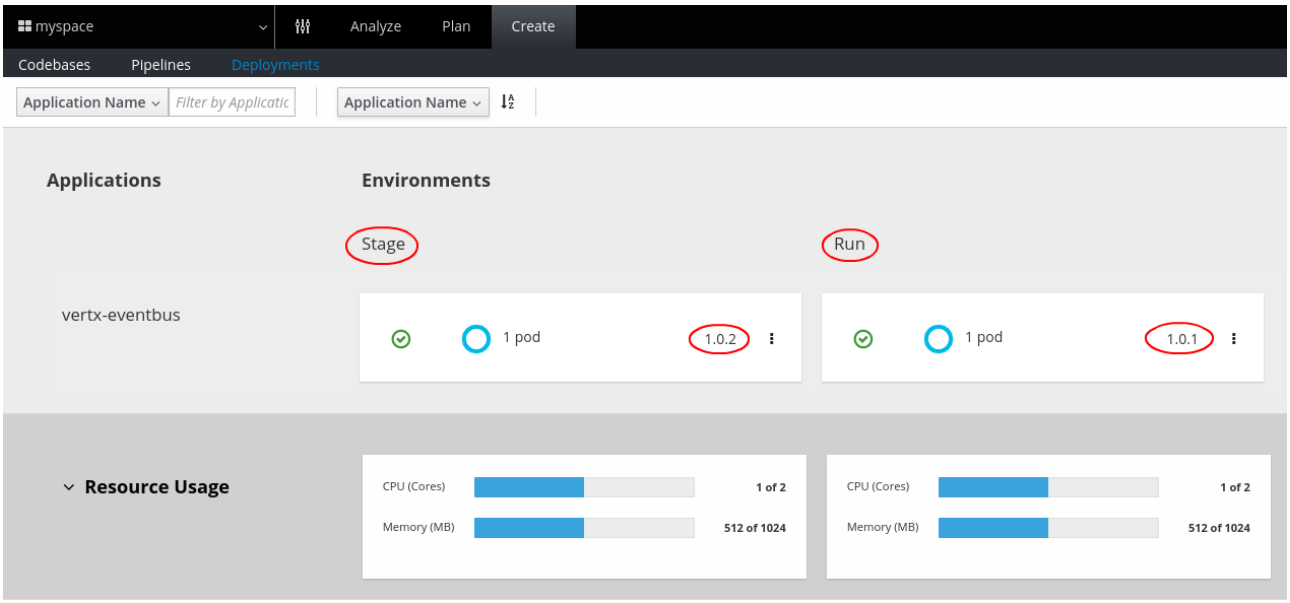
You have now committed your code changes to GitHub.

Chapter 8. Reviewing and publishing your changes

When you commit and push a change to GitHub, a pipeline build is automatically triggered in CodeReady Toolchain.

To review the build and publish your changes:

1. Return to the CodeReady Toolchain browser tab.
2. Click **Create** and then **Pipelines** to view the build pipelines. Wait for the build pipeline to progress to the **Approve** stage.
3. In the **Create** tab, click **Deployments** to see the following information:



   - Different versions of your application are now deployed to **Stage** and **Run**. Version **1.0.2** of the application, which includes your committed change to the code, is deployed to **Stage** because you have not yet promoted it to **Run**. The older version, **1.0.1** , is deployed to **Run** because you approved it the last time the pipeline build executed.
   - The green check marks indicate that both builds are operational.
   - The **1 pod** indicates that each of the application builds scale to one pod in OpenShift Online. The number of pods indicates the number of running instances of the application.
   - The version numbers link to individual running applications. You can use these separate staging areas to share different versions of your application before promoting a change. Click the version numbers to view the details for that deployment.
4. Click **Pipelines** to return to the pipelines view and click **[Input Required]** at the **Approve** stage of the pipeline.
5. Click **[Promote]** to promote version 1.0.2 of the application to **Run**.

   Your changes are now available on both **Stage** and **Run**. If you return to the **Deployments** tab, you can see that version 1.0.2 is deployed to both **Stage** and **Run**.

You have now imported an existing project into CodeReady Toolchain, made changes to your project code, committed the changes to GitHub, and published the new version of your project.