Using Eclipse Che IDE to develop your codebase

Red Hat Developers Documentation Team

2018-12-11 13:07:31 UTC

Table of Contents

This guide describes the features of Eclipse Che IDE hosted on Red Hat CodeReady Toolchain for OpenShift, which provides you with a containerized development and testing environment. It describes the usage of Che workspaces to edit and develop your codebases.

Using Eclipse Che IDE to develop your codebase

Chapter 1. About workspaces

CodeReady Toolchain provides hosted instances of Eclipse Che within your browser to edit, test, and debug your project code. One of the key features of Eclipse Che is Che *workspaces*, which provide a fully configured runtime environment for running your code. As a result, CodeReady Toolchain provides a lightweight, next generation IDE, and a containerized development and testing environment.

When you use the quickstart to create an application in CodeReady Toolchain, the development and testing environment in the workspace is automatically configured with the necessary runtime components. Now you can use the Che workspaces as your personal development machine to modify your application codebase.

Editing your project code and reviewing changes

After you create a project, you can edit the code in Che and stage the changes.

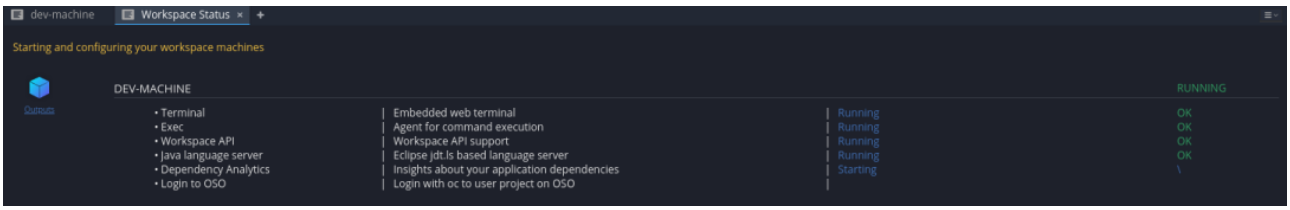
Chapter 2. Creating a Che workspace

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 3. 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.

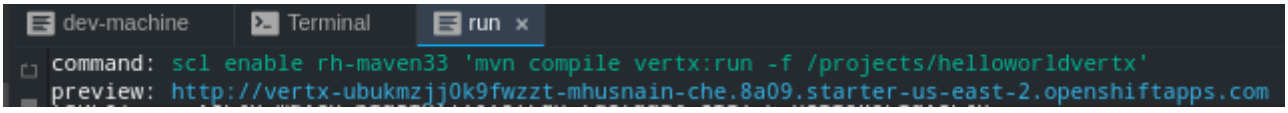
If the **Run** command macro is not defined, it displays the **Create run command** option instead of **Run**:

See create_run_command_macro for instructions on setting up a run command.

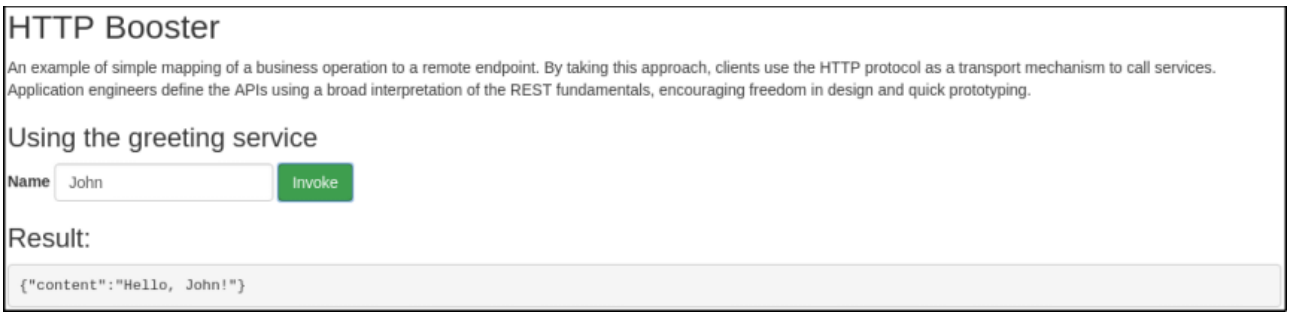
2. A **run** terminal appears at the bottom pane of the Che workspace. When the `mvn build` command finishes executing, the **run** view displays the following message:

```
[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.



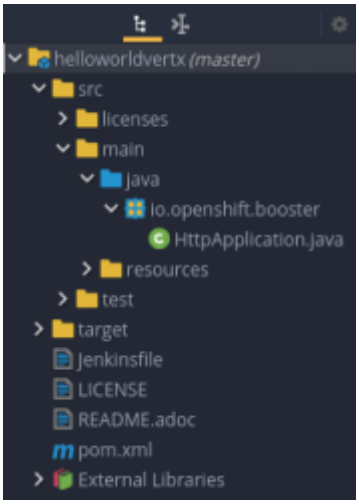
This is the same version of the application that the pipeline deployed to **Stage** and you subsequently promoted to **Run**. The URL for this build of the application is different from the URLs used by OpenShift Online for **Stage** and **Run**. This is your private sandbox hosted within Che. You can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

run it in their browser.

Chapter 4. Changing the quickstart code

To change your project code and preview the results:

1. In your Che workstation view, navigate to: `src` > `main` > `Java` > `io.openshift.booster` > `HttpApplication.java`.



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

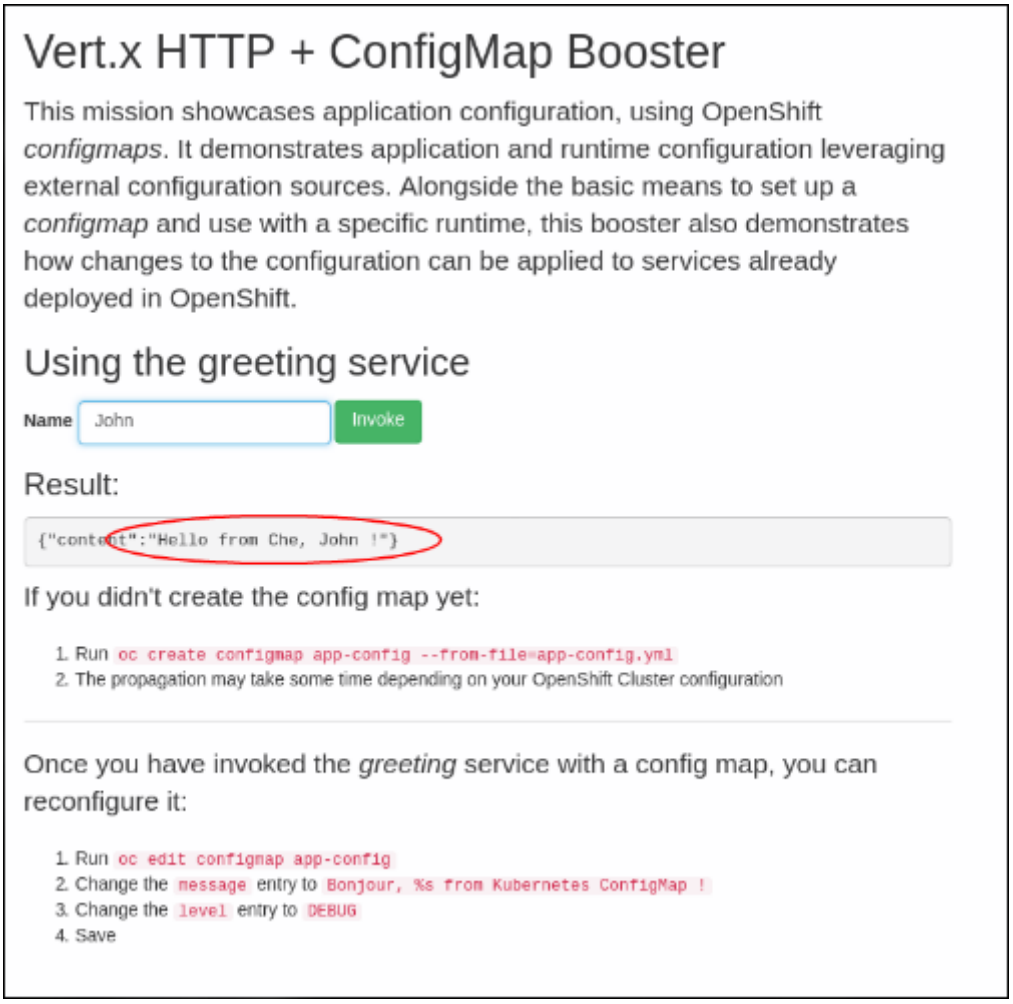
```
static final String template = "Hello, %s!";
```

3. Change the greeting message and then save your changes.

```
static final String template = "Hello from Che, %s !"
```

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.

4. Return to the browser tab running the application, add a name in the **Name** field, and click **[Invoke]** to test the application. The displayed message shows the amended text template.



You have now learned how the workspace automatically saves and applies your changes.

Chapter 5. 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:

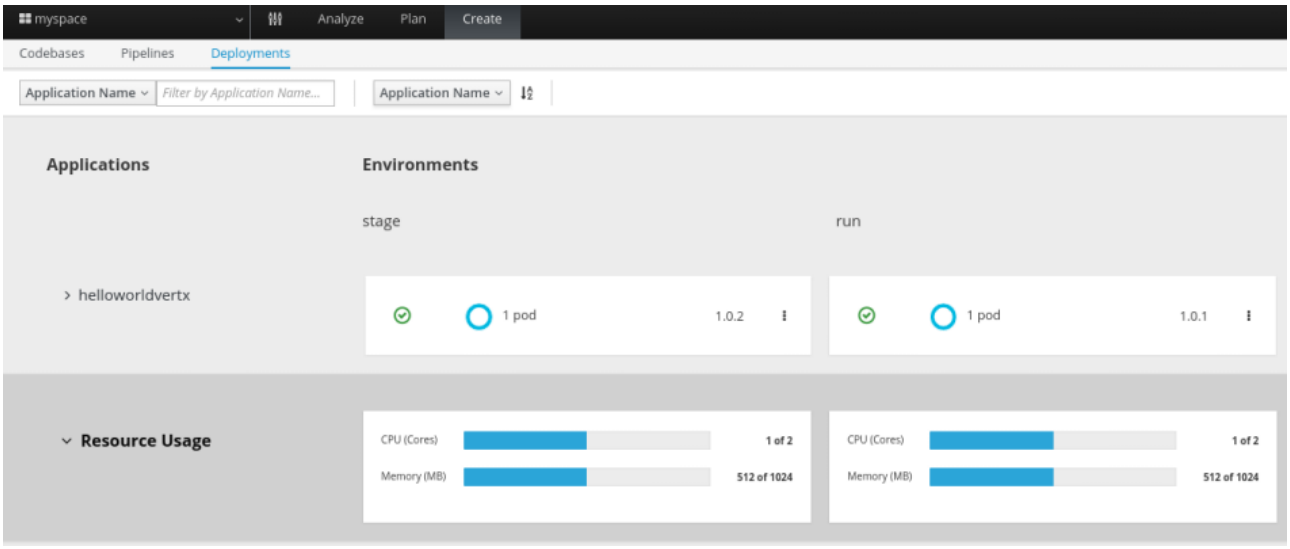
You have now committed your code changes to GitHub.

Chapter 6. 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 created your first quickstart project in CodeReady Toolchain, made changes to your

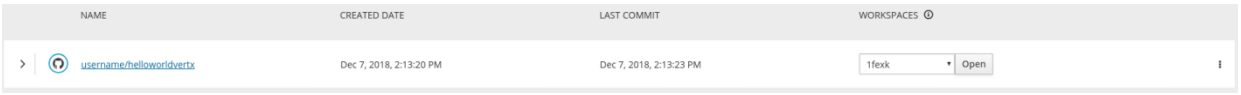
project code, committed the changes to GitHub, and published the new version of your project.

Using advanced Che features

Chapter 7. Using the code assistant

1. Return to the Che workspace browser tab. If you closed your earlier workspace tab, use the following instructions:

   a. In CodeReady Toolchain, click the **Create** tab. The default **Codebases** view lists your project workspace:

   b. Click **[Open]** to view the existing workspace in a new tab. If the new tab does not load, ensure your browser is not blocking pop-up windows.

2. When the workspace loads, expand the following directory path in the top left panel: `src` > `main` > `Java` > `io.openshift.booster`.

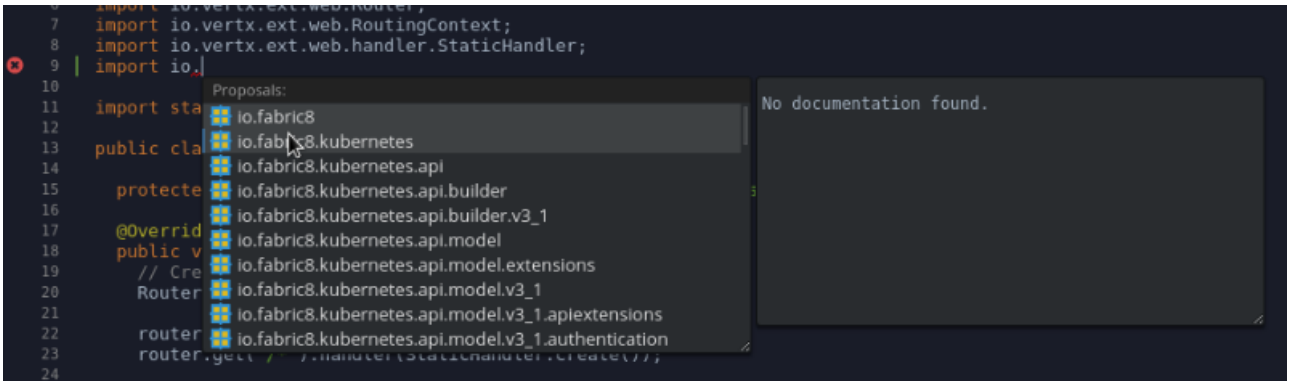
3. Double click `HttpApplication.java` to edit this file.

4. In the text editor, add a line break at line 9 after the **import** section.

5. Type the following in the new line:

```
import io.
```

6. Press `Ctrl`+`Space` to view the autocomplete options for this line. Select any one of the options.

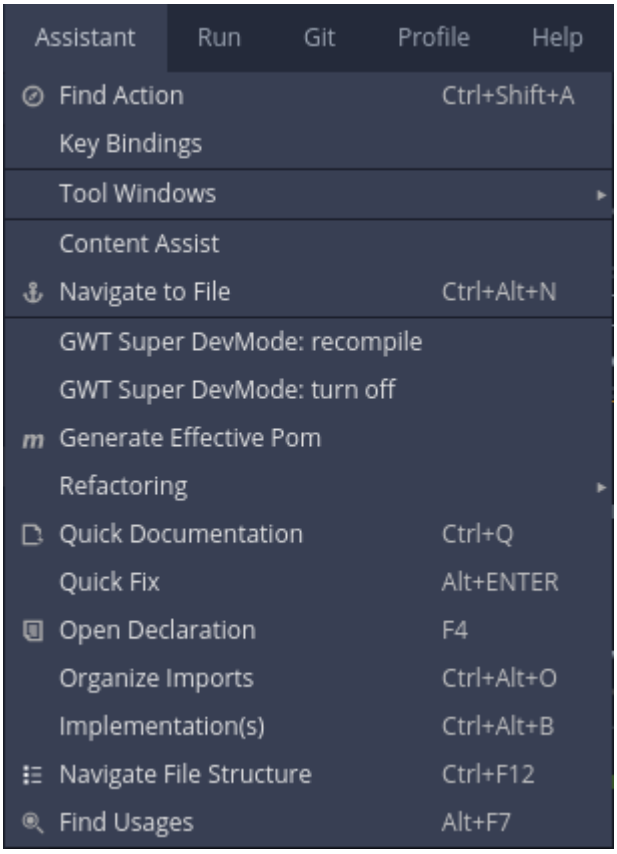


7. Move the mouse pointer over the red X error icon next to the line number to view the error, which warns you that import is not used.

8. Delete the added line to clear the error.

You have now learned how the Che workspace uses Code Assistant to make recommendations and identify errors.

To view additional IDE features, click **Assistant** at the top of the page.


Assistant
Run
Git
Profile
Help
Find Action Ctrl+Shift+A
Key Bindings
Tool Windows
Content Assist
Navigate to File Ctrl+Alt+N
GWT Super DevMode: recompile
GWT Super DevMode: turn off
Generate Effective Pom
Refactoring
Quick Documentation Ctrl+Q
Quick Fix Alt+ENTER
Open Declaration F4
Organize Imports Ctrl+Alt+O
Implementation(s) Ctrl+Alt+B
Navigate File Structure Ctrl+F12
Find Usages Alt+F7

Chapter 8. Using the terminal tab

Your Che workspace displays a terminal tab at the bottom of the screen. You can use this terminal to run commands in your private Linux container (your workspace) as follows:

1. In your Che workspace, if your application is not running, Click the **run** option (

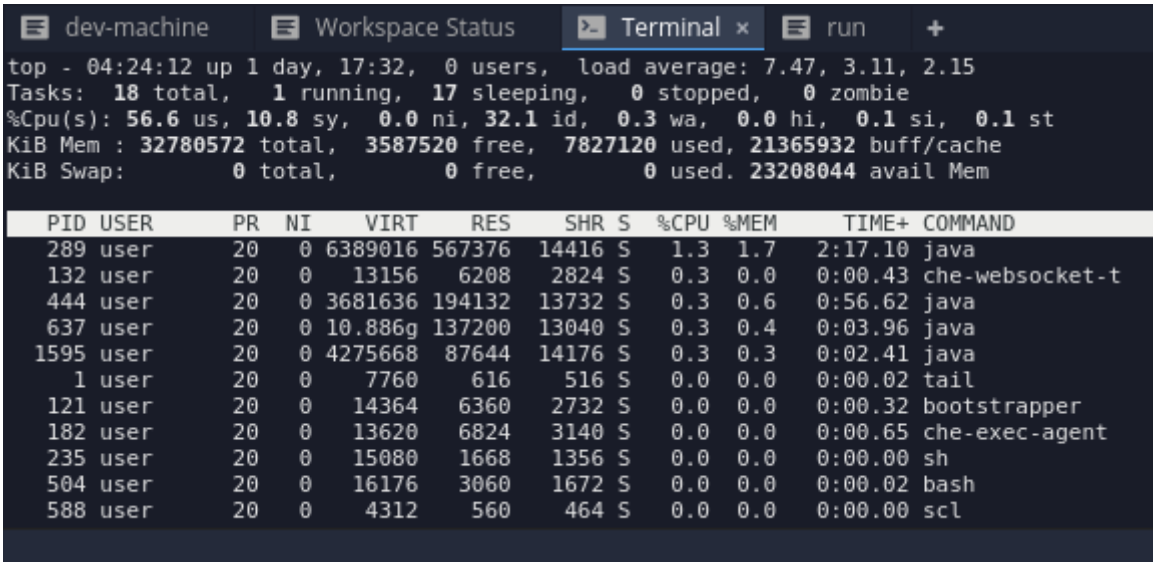
). The following message appears in the terminal tab when the run process completes:

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

2. Click **terminal** next to the **run** tab to view your workspace terminal.
3. Type the following in the terminal and press `Enter` to run the commands.

```
cat /etc/os-release
top
```

The terminal tab displays the following information about your Linux container:



4. Experiment further with the following commands in your terminal tab:
   a. Press `Ctrl+c` to stop the **top** command.
   b. Enter the following:

```
curl localhost:8080
```

   c. Enter the following:

```
curl localhost:8080/api/greeting
```

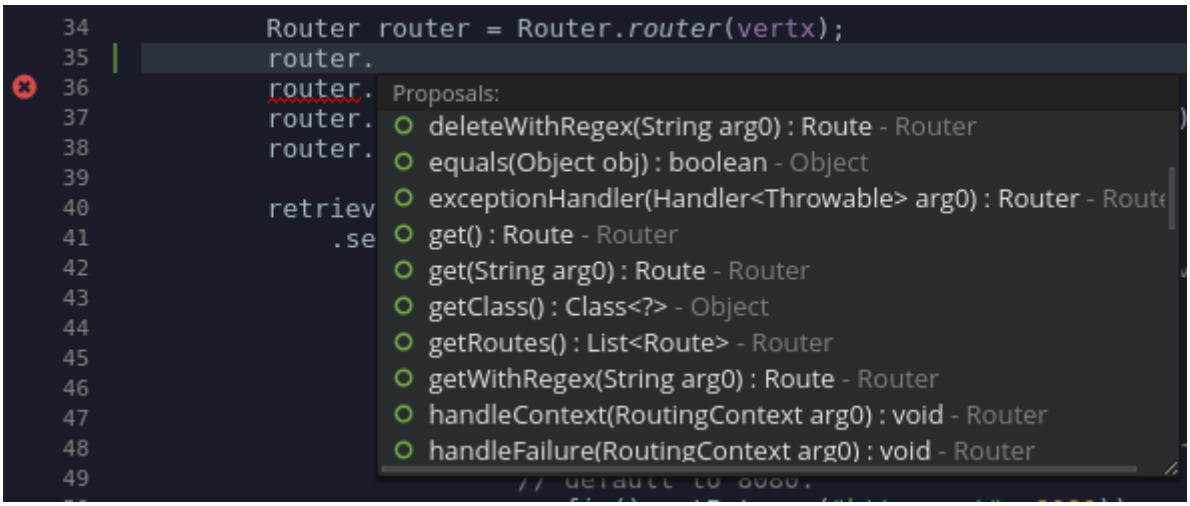
5. After experimenting with the terminal commands, use the stop icon at the top of the page to stop the application.

Chapter 9. Debugging using your Che workspace

After creating your CodeReady Toolchain project, you can debug your application code in the Che workspace. The following is a tutorial using the quickstart project you have already created.

To debug your project code:

1. Return to your project Che workspace browser tab.
2. If the `HttpApplication.java` file is not already loaded, double-click it in your file explorer view (`src` > `main` > `Java` > `io.openshift.booster`) to view the contents.
3. In your `HttpApplication.java` file, add a line break after line 34, then type `router.` and press `Ctrl+Space` to see the context-aware assistance.
4. Scroll down in the displayed list and double-click the `get():Route` method. This is the Vert.x Router method that maps to the HTTP verb GET. You now have the syntax for the get method.



5. Edit the new line to include the `goodbye` endpoint. The following is an example of the completed new line:

```
router.get("/api/goodbye").handler(this::goodbye);
```

The completed line shows a red x indicating an error because the referenced `goodbye` endpoint is not available in the code. This endpoint is added in the next step.

6. Add a new line after line 74 and then add the following method for the `goodbye` endpoint:

```
private void goodbye(RoutingContext rc) {
  String name = rc.request().getParam("name");
  if (name == null) {
    name = "World";
  }
  JsonObject response = new JsonObject()
    .put("content", "Goodbye " + name);

    rc.response()
      .putHeader(CONTENT_TYPE, "application/json; charset=utf-8")
        .end(response.encodePrettily());
}
```

7. Use `Ctrl+s` (or `Cmd+s` for macOS) to save your changes.
8. If your application is already deployed, skip this step. To deploy the application, click the **run** option (

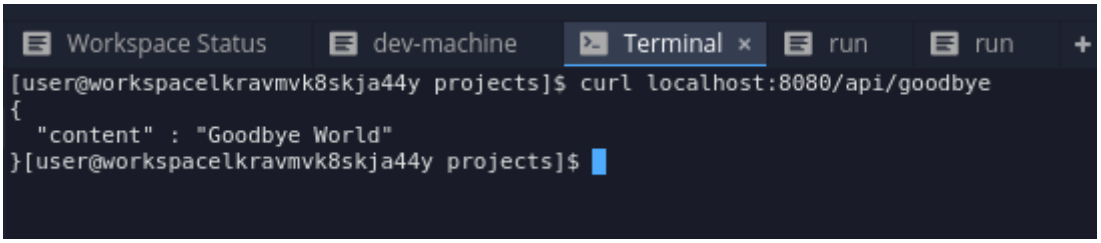
). The following message displays when the run process completes:

```
INFO: Succeeded in deploying verticle
```

9. In the **Terminal** tab at the bottom of the page, test the new endpoint by typing the following command:

```
curl localhost:8080/api/goodbye
```

The following result appears if the changes were successful:



10. To stop the application, click the stop button in the **EXEC** bar on the top of the workspace page.

When the application stops, the square icon becomes a circular arrow ().

Chapter 10. Setting up the debugger

To debug your project code, set up the Che workspace Debugger feature:

1. In your workspace, click the **debug** option in the debug button.

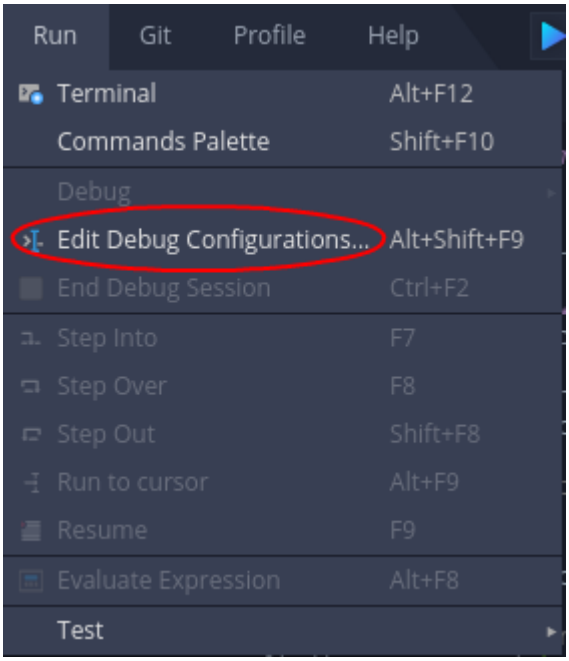
2. In the new **debug** tab in the **Processes** pane at the bottom of the screen, look for the following lines when the debugging starts and wait for the process to complete:

```
[INFO] The application will wait for a debugger to attach on debugPort 5005

[INFO] Launching Vert.x Application

[INFO] Listening for transport dt_socket at address: 5005
```

3. From the menu bar, click **Run** and select **Edit Debug Configurations**.



4. In the **Debug Configurations** dialog box:
    a. In the left pane, click **[+]** for the **JAVA** item in the dialog box to add a **Remote Java** port.
    b. In the right pane, change the **Port** value to `5005`.
    c. Click **[Save]** and then **[Close]**.
5. To create a new breakpoint for your method:
    a. In the `HttpApplication.java` file, go to the following line of code at line 77:

```
    if (name == null) {
```

b. Click the line number (77 in the margin) for this line of code:

6. Click the debug icon () at the bottom left corner of the workspace to verify that the breakpoint is added.

The **Breakpoints** pane at the bottom left corner of the screen lists all added breakpoints named after the file they are added in and then the line number. For example, your new breakpoint is listed as the following:

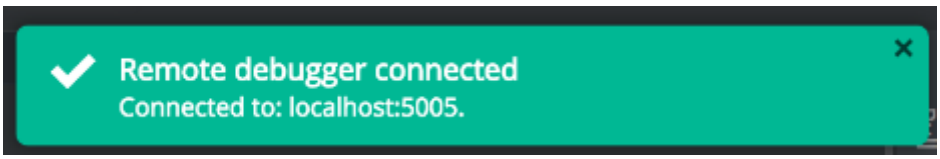
```
HttpApplication.java:77
```

The Debugger is now set up for your project.

Chapter 11. Running the debug command

Now that you have set up the Debugger for your project, you can use the debug command as follows:

1. In your workspace tab, in the top menu bar options, click **Run › Debug › Remote Java**. A success message displays when the remote debugger connects.



If your remote debugger connection fails, restart your browser and try again.

2. In the top menu bar options, click **Run › Terminal** to view the terminal tab.
3. Run the following command in the terminal to start debugging:

```
curl localhost:8080/api/goodbye
```

4. Click the debug icon () at the bottom left corner of the workspace. The **Variables** panel on the bottom right side of the screen lists your debugger variables.

You can explore the debugger features by using the **Resume**, **Step Into**, **Step Over**, and **Step Out** options during the debugging process. When finished, close the workspace tab and return to the CodeReady Toolchain tab.