



Getting Started with IBM Bluemix

Hands-On Workshop

**Exercise 5a: Creating the project
source code repository**

Exercises 5a - 5d overview

These exercises will demonstrate an approach for developing applications by using principles from agile development and using DevOps processes and tooling.

You'll develop a REST API that will calculate the FizzBuzz result for a given range.

FizzBuzz is a children's numeracy game where any number divisible by 3 is replaced by the word *Fizz*, any number divisible by 5 is replaced with the word *Buzz*, and any number divisible by both 3 and 5 is replaced with the word *FizzBuzz*.

For example, for the range 1 .. 20, the response is:

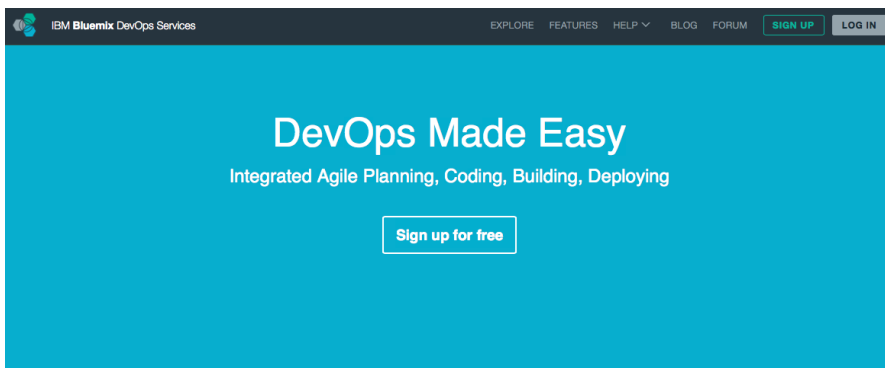
```
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14,  
FizzBuzz, 16, 17, Fizz, 19, Buzz
```

You will use Node.js as the runtime and use test-driven development practices to create the solution.

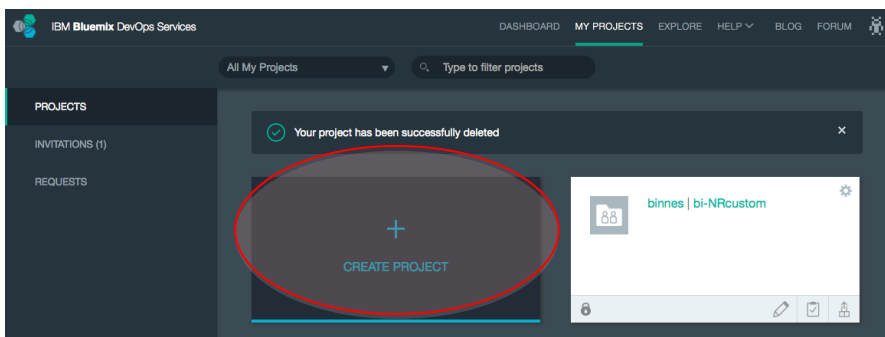
Creating the project source code repository

Before you write any code, you must have the correct tooling and a source code repository. For this exercise, you'll use the Git service from IBM DevOps Services for Bluemix.

1. Log in to DevOps Services for Bluemix: <http://hub.jazz.net>.



2. From your MyProjects page, click **CREATE PROJECT**.



3. Create a Git repository on Bluemix:
 - a. Enter a project name.
 - b. Create a new repository.
 - c. Create a Git repo on Bluemix.

Create a project

binnes | 

Create a new repository or link to an existing one ⓘ



Create a new repository

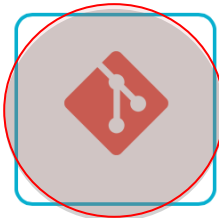


Link to an existing repository

Choose your new repository location ⓘ



Create a Git repo on GitHub



Create a Git repo on Bluemix



Create Jazz SCM on Bluemix

Private Git URL: <https://hub.jazz.net/git/binnes/fizzbuzz-w3>

4. Initialize the repository.

Initialize repository contents

☒ Initialize the repository with a README and license template.

If you want to share your project and make it searchable to the public, do not select the **Private Project** checkbox.

5. If you want to use the Track and Plan tool to support agile methodology, add features for Scrum development.
6. Select the checkbox to make the project a Bluemix project. Then, provide the details of the Bluemix space that the application will be deployed to.

Select project contents

☒ Private Project
Private projects are only accessible by invited team members. [Learn more](#)

☒ Add features for Scrum development (This option can only be added at project creation time.)
Select this if you're familiar with Scrum and plan to deliver software on regular sprints. ⓘ

☒ Make this a Bluemix Project
Select this if you want to deploy your application to the IBM Bluemix cloud platform. [Find out how](#) ⓘ

Bluemix projects are charged for Track & Plan and Delivery Pipeline (Build & Deploy) usage in accordance with the [Bluemix pricing plan](#).

Select a Bluemix space to bill your services to:

Region	IBM Bluemix (United Kingdom)
Organization	binnes@uk.ibm.com
Space	dev

These selections can be changed later in the options for your Project Settings.

Your project must be connected to a Bluemix space for billing purposes. Normally you should select the "dev" (Development) space you want to use with this application.

If you prefer to do this later just uncheck "Make this a Bluemix Project" to continue.

Under the Standard plan, Track and Plan is free for 3 users. Delivery Pipeline offers 60 minutes of build time and 2 deployers per month for free. To continue using these services beyond the free tier, you must add one or both of the Track and Plan and Delivery Pipeline (Build & Deploy) services to the space you have configured for billing. [Learn more](#).

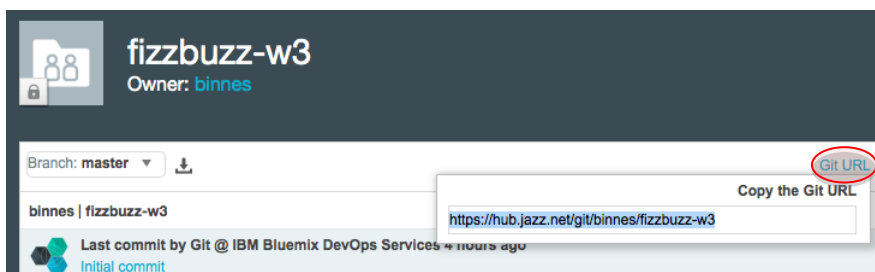
After you've added either of these services to your billing space, you can monitor usage in the Bluemix Account usage page.

CREATE

7. Click **CREATE** to create the project.

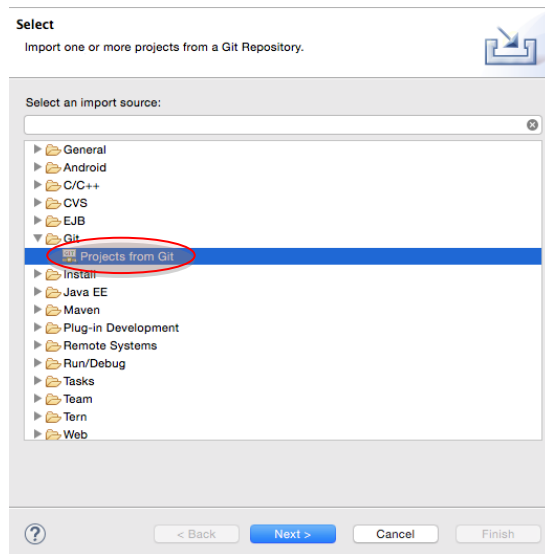
The code repository has been created, and you can create a new Eclipse project that is based on the repository.

8. In the DevOps Services console, select the Git URL link and copy the URL. Use Ctrl+C or Cmd C.

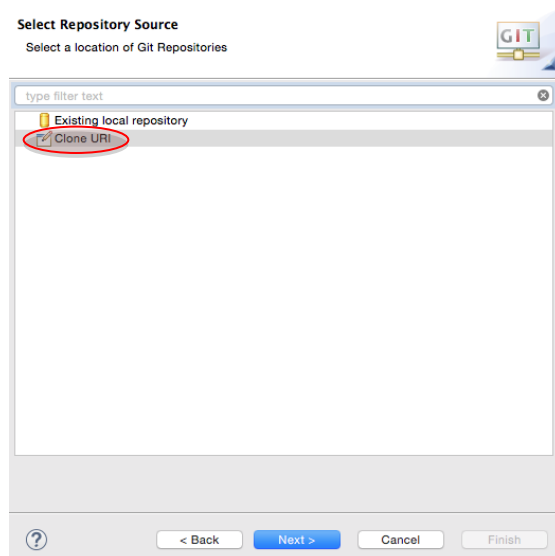


9. Start Eclipse and then from the main menu, click **File > Import**.

10. From the Select dialog, click **Git > Project from Git** and then click **Next**.



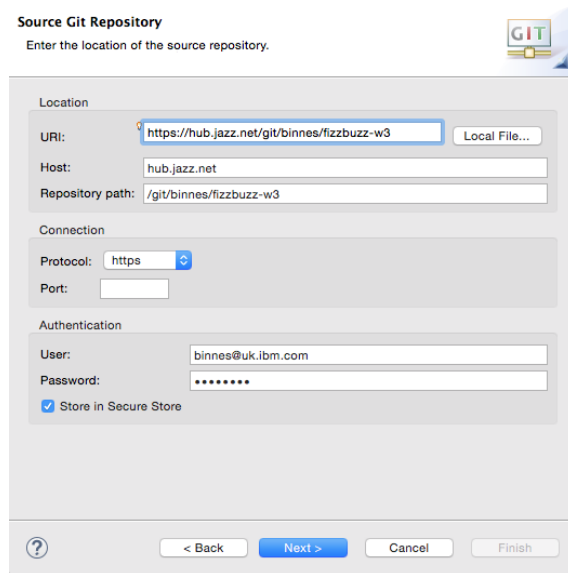
11. In the Git dialog, select Clone URI and then click **Next**.



12. In the next dialog, paste the Git repository URL into the **URI Location** field. This will automatically populate the Host and Repository path.

13. Enter your DevOps services user name and password. Optionally, select to store the Git credentials in a secure store in Eclipse. You will need to provide additional information to initialize the secure store if this is the first time that you store credentials.

14. Click **Next**.



Source Git Repository
Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

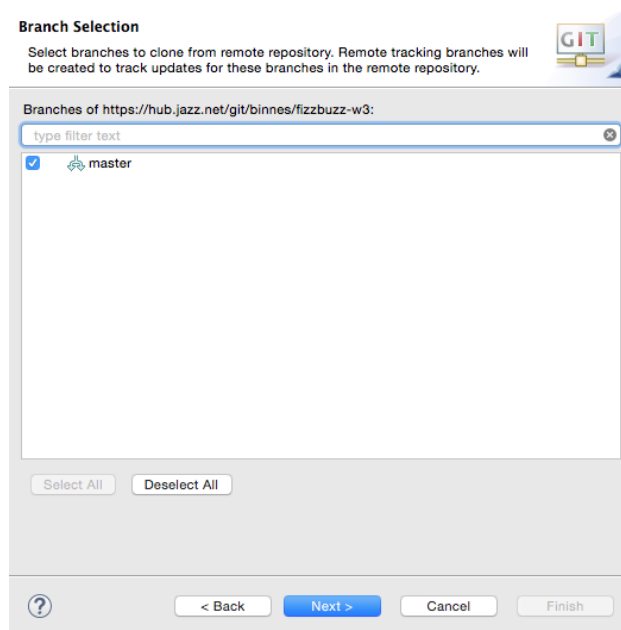
User:

Password:

☒ Store in Secure Store


Eclipse will query the Git repository and list the next branch that's available. Because you just created the repository, only the master branch exists.

15. Leave the master branch selected and click **Next**.



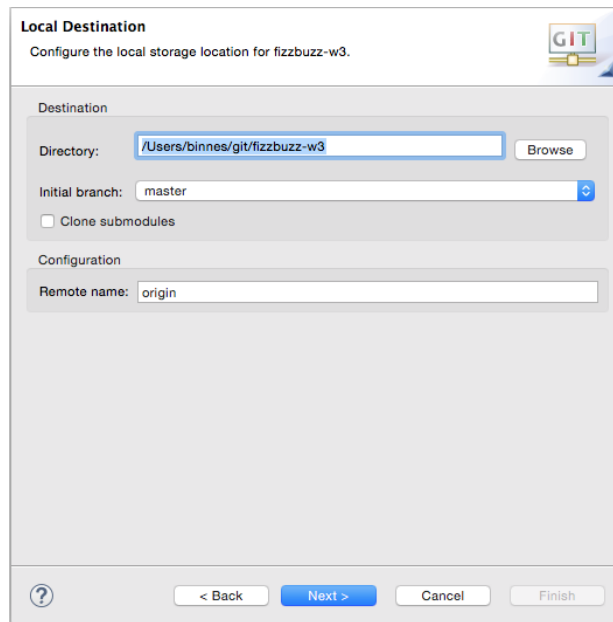
Branch Selection
Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.

Branches of https://hub.jazz.net/git/binnes/fizzbuzz-w3:

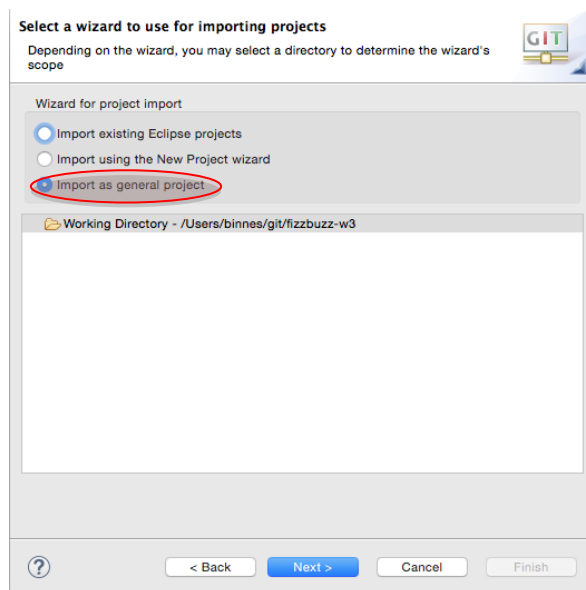
☒  master

The local destination is then chosen. The default location is `<user home directory>/git/<project name>`. Use the suggested default location. If you want to change it, you must remember the location because you will need it later in the exercise.

16. Click **Next**.

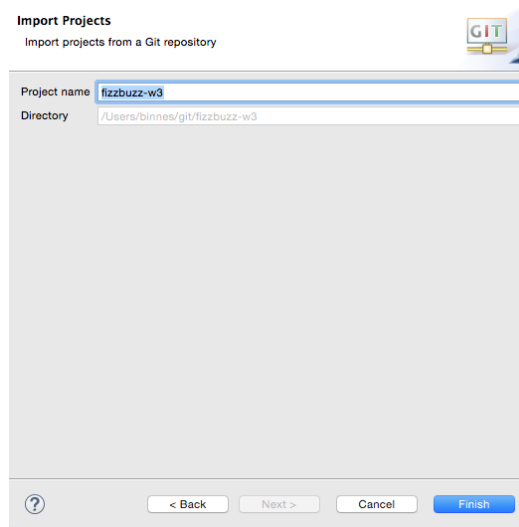


17. Select the wizard to import as a general project and then click **Next**.



18. In the Import Projects dialog, enter an Eclipse project name. You should use the same name as the DevOps service project name, but you can change it if needed.

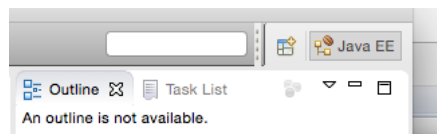
19. Click **Finish**.



You now have an Eclipse project linked to your DevOps services Git repository. To complete the project setup, configure Eclipse to support JavaScript development.

20. Switch Eclipse to the JavaScript perspective:

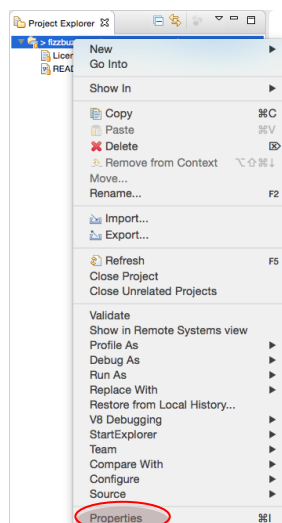
- a. Click **Add Perspective** at the top right of the Eclipse window.



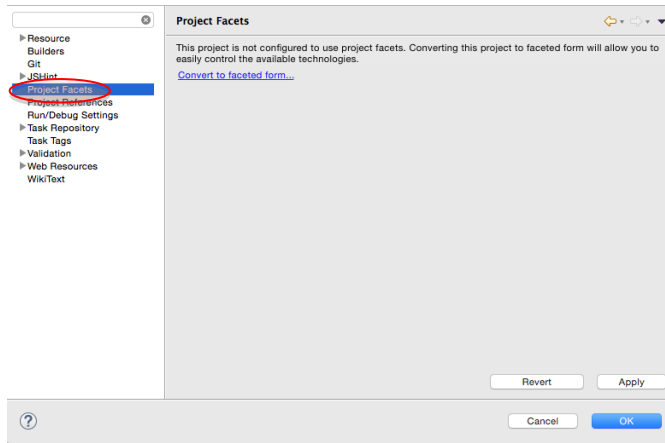
- b. Select **JavaScript** from the list.

21. Enable the NodeJS facet on the project:

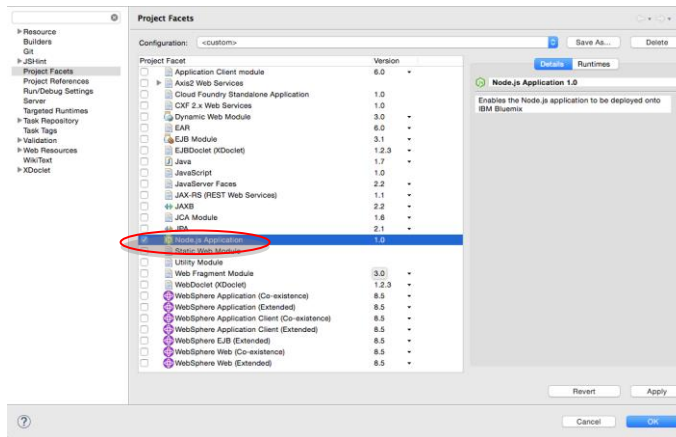
- a. Right-click the project name and click **Properties**.



- b. In the Project Facet dialog, click **Project Facets** and then click **Convert to faceted form**.



- c. When you see the list of facets, click **Node.js Application**. Click **OK**.



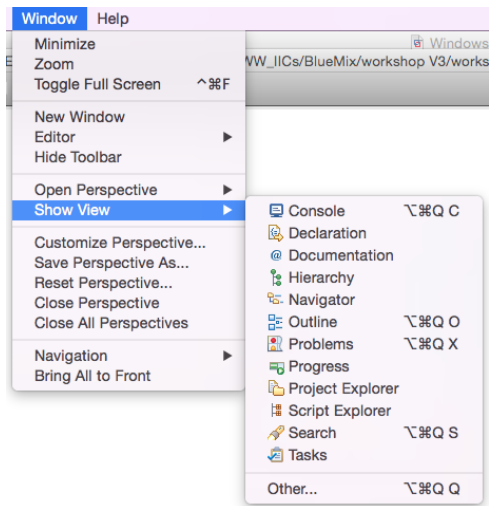
Adding the facet automatically creates a sample application in the project.

- d. Remove the `app.js` and `package.json` files. Right-click each file, click **Delete**, and confirm the deletion.

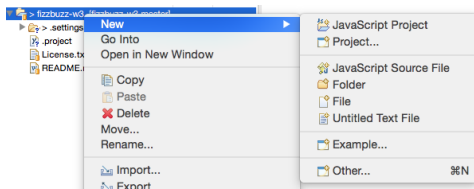
The project shows that there are changes to the local copy of the project. These are the Eclipse project setting files. Don't save these as part of the source code.

22. Add a navigator view to see all the files in the project:

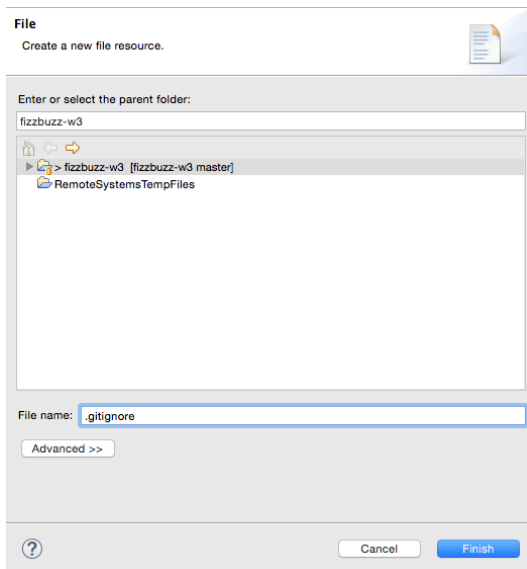
- a. Click **Window > Show View** and then click **Navigator**.



- b. Right-click the project name in the Navigator view and click **New > File**.



- c. Name the file `.gitignore` and then click **Finish**.



- d. Add the following two lines to the `.gitignore` file:

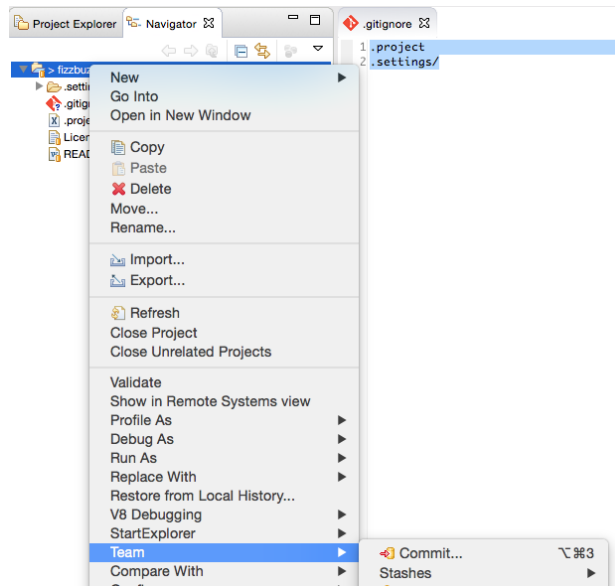
Windows: Leave the slash as specified; do not change it to `.settings\`:

```
.project
.settings/
```

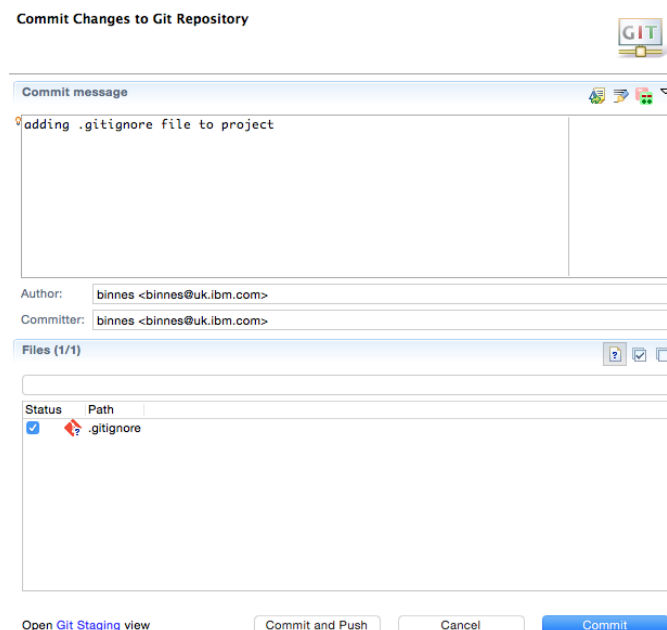
The project still shows that there are changes. This occurs because the `.gitignore` file is a new file that is not committed and pushed to the master branch on the Git server.

23. Commit and push the `.gitignore` file:

- a. Right-click the project name and click **Team > Commit**.



- b. In the Commit Changes dialog, enter a commit message and select the `.gitignore` file. Then, click **Commit and Push**.



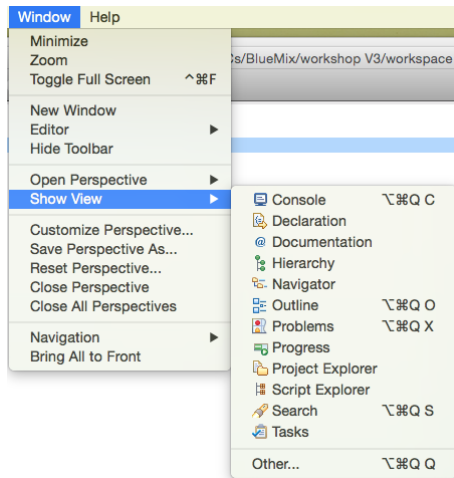
- c. Close the confirmation dialog by clicking **OK**.

The project will no longer show that there are outstanding changes.

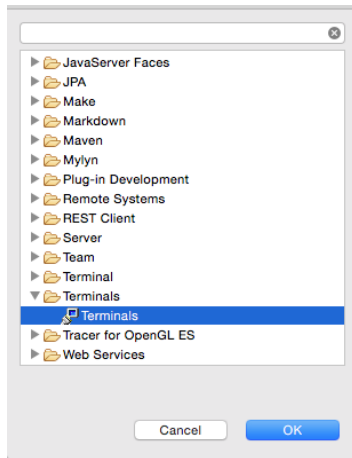
Finally, you should add a new window to access the command line. However, this is not necessary. Instead, you can use a command window outside Eclipse, but it's more convenient to have everything in Eclipse because it provides a better working environment.

24. Create a terminals view:

- a. From the main menu, click **Window > Show View** and then click **Other**.



- b. In the dialog, click **Terminals > Terminals** and then click **OK**.



- c. In the Terminals window, drag the tab to split the bottom section of the screen to allow concurrent viewing of the terminal and the problem views.



If a command prompt isn't shown, click the **Open Terminal** icon to start a new terminal. Use the default setting. On some platforms, this happens automatically.

