**Automated Testing – Eclipse**

In this lab, you will learn how to create new JUnit tests to the Parts Unlimited MRP App using the Eclipse IDE, and then get them to run in an automated build in Visual Studio Team Services. After finishing this lab, you will have added unit tests to the Parts Unlimited MRP App and have those tests running during build time.

### **Lab Tasks:**

* **Set up the development environment:** This step will get your local environment set up to work on the Parts Unlimited MRP App.
* **Add JUnit tests to the Parts Unlimited MRP App:** In this step, you will add automated unit tests to the Parts Unlimited MRP App and run them locally.
* **Run the automated tests in an automated build:** In this step, you will run the JUnit tests as part of the Continuous Integration Build that was set up in the Continuous Integration HOL with Parts Unlimited MRP.

### **Estimated Lab Time:**

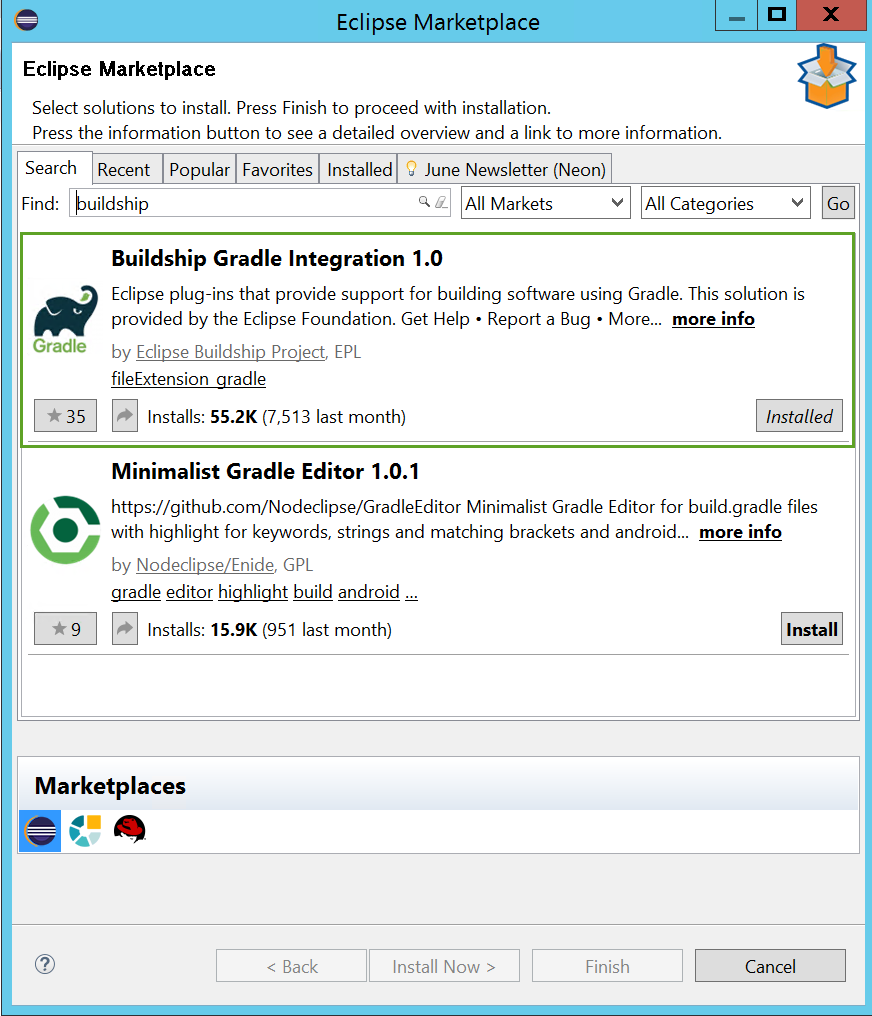
* approx. 40 minutes

### **Task 1. Set up the development environment**

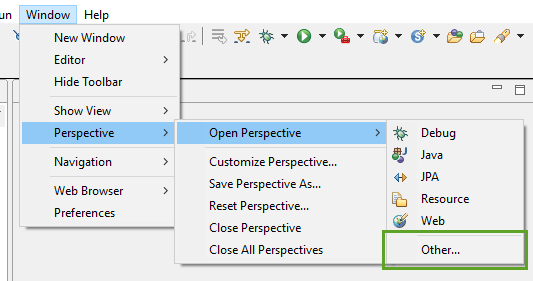
First, we need to set up Eclipse in order to work on the Parts Unlimited MRP App on our local machine. The Parts Unlimited MRP App uses [Gradle](https://gradle.org/), so we need to set up the code as Gradle projects in Eclipse.

**1.** Open up **Eclipse**.

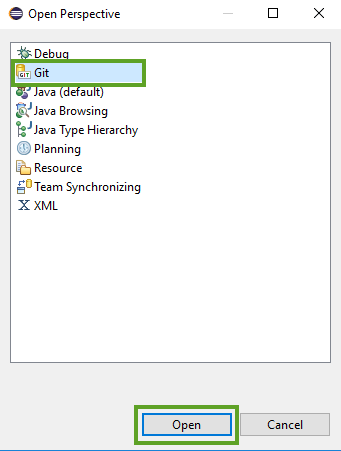
**2.** Navigate to the application toolbar and select Help -> Eclipse Marketplace. Search for the [Gradle Buildship plugin](https://projects.eclipse.org/projects/tools.buildship). If the plugin is not installed already, install it and restart Eclipse. If the plugin is installed already, click on the **Installed** tab and select **Update** on the plugin if possible.



**3.** From the menu of Eclipse, select **Window -> Perspective -> Open Perspective -> Other…** to select the perspective that we want.

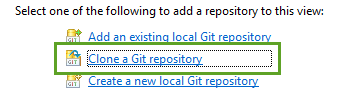


**4.** Select **Git** and click **Open**.



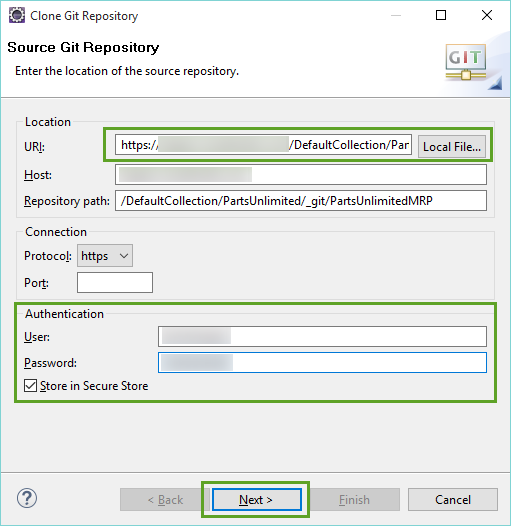
This will open up the Git perspective in Eclipse.

**5.** If the Visual Studio Team Services Git repository already exists on your local machine, **skip to step 8**. Otherwise, click the **Clone a Git repository** from the Git Repositories Window:

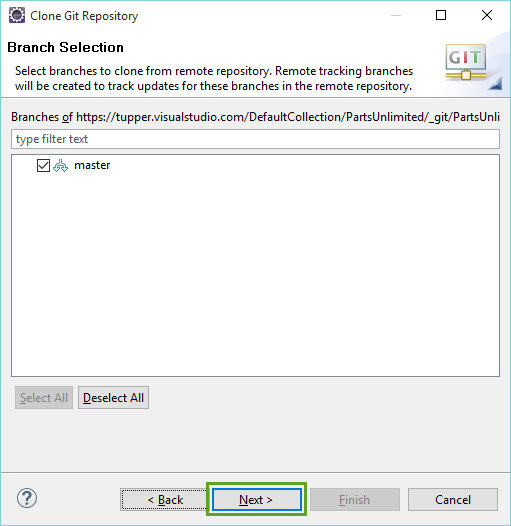


In the Clone Git Repository, **paste the URL for your Visual Studio Team Services Git repo** into the URI text field, **enter your alternate credentials** in the Authentication section , and click **Next**.

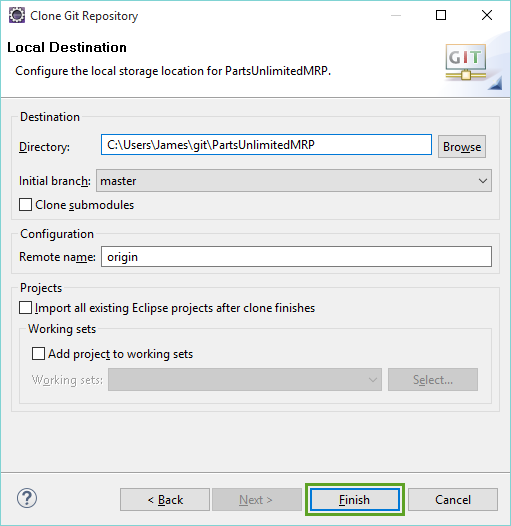
If you need to set up alternate credentials in your VSTS account, go to this link (replace your account name first): https://youraccount.visualstudio.com/\_details/security/altcreds



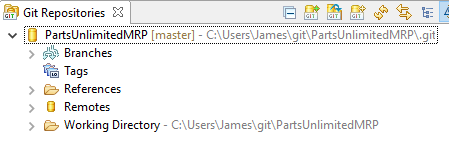
Select any branch that you want, and click **Next**.



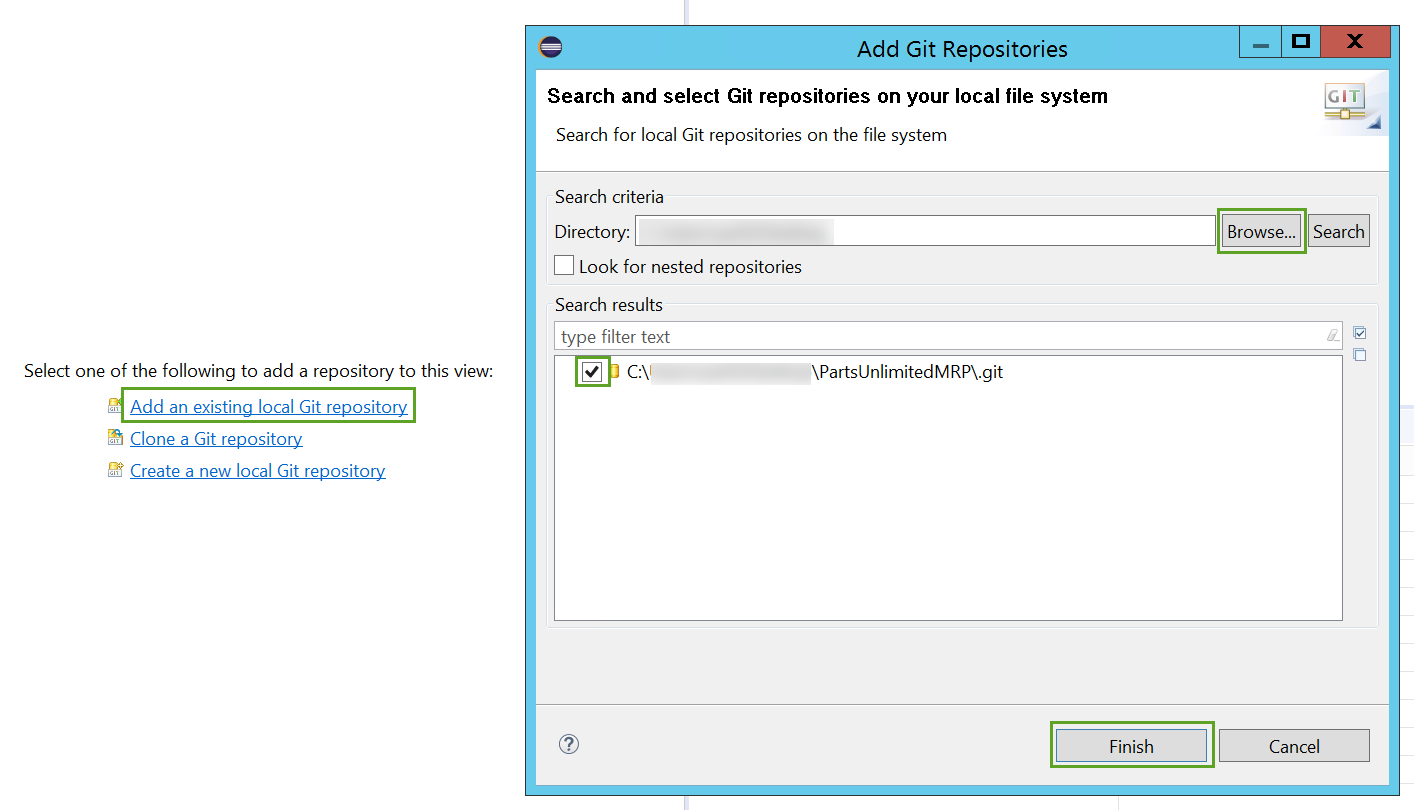
On the next page, leave the defaults and click **Finish**.



This will clone the repository to your local machine. You should be able to see the repository in the Git Repositories window.



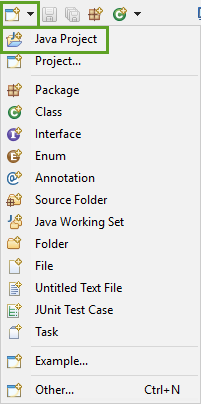
**6.** If you already have the Visual Studio Team Services Git repo on your machine, select the option to **Add an existing local Git repository**, then navigate to the folder of the Git repo and check the box next to it to add the repo reference into Eclipse.



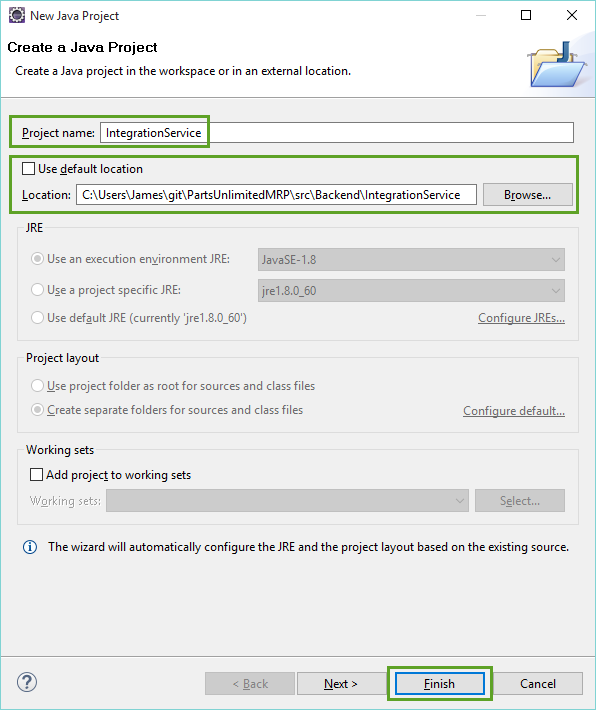
**7.** Open the Java perspective by clicking the shortcut in the top right corner of Eclipse.



**8.** In the top left, click the **New** drop down and select **Java project**.

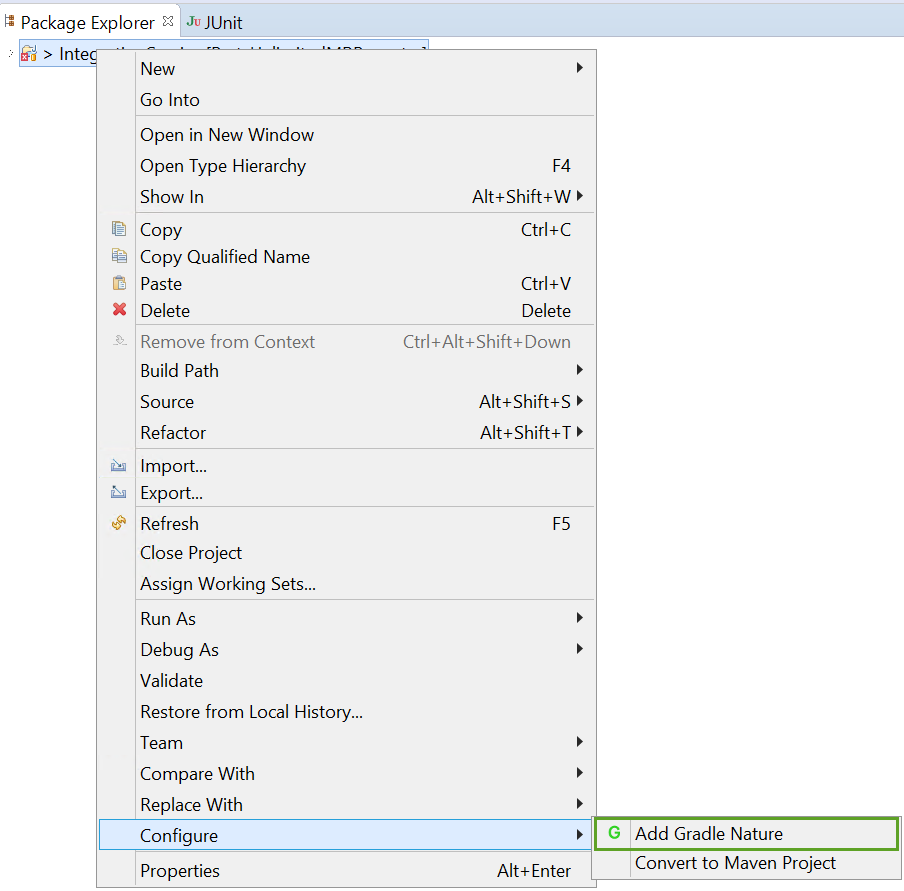


**9.** In the New Java Project dialog name the project **IntegrationService**, uncheck the **Use default location**, choose the **IntegrationService** folder located in src/Backend/ of the PartsUnlimitedMRP repository, and click **Finish**.



**10.** In the Package Explorer window, **right-click on IntegrationService** and then select **Configure -> Add Gradle Nature**.

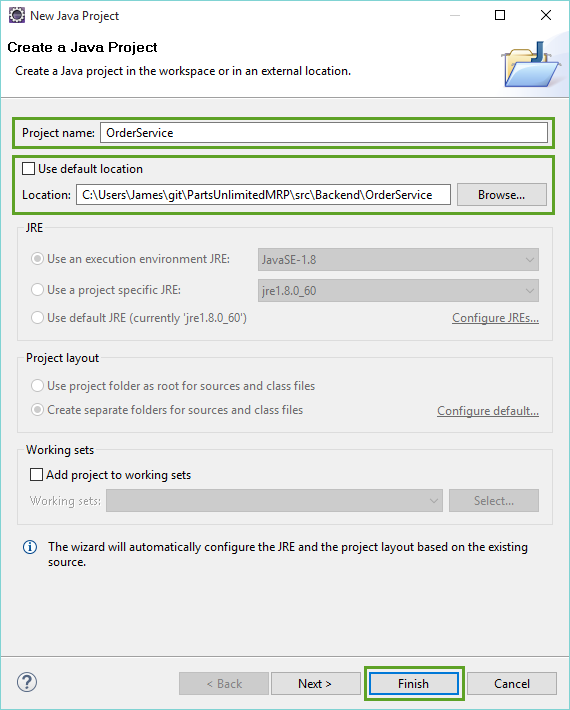
**Note:** Allow network access when requested.



Note: at this point, you should see the errors disappear from the project.

**11.** Follow **Step 8** to create a **new Java Project**.

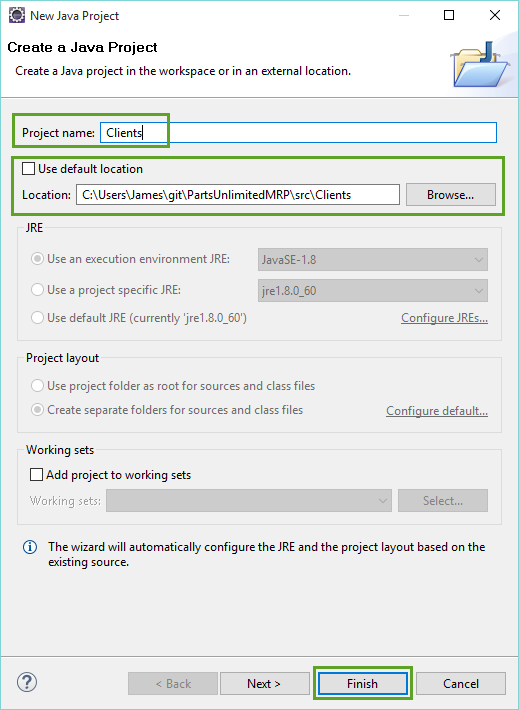
**12.** In the New Java Project dialog name the project **OrderService**, uncheck the **Use default location**, choose the **OrderService** folder located in src/Backend/ of the PartsUnlimitedMRP repository, and click **Finish**.



**13.** In the Package Explorer window, **right-click on OrderService** and then select **Configure -> Add Gradle Nature**.

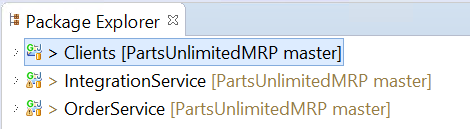
**14.** Follow **Step 8** again to create a **new Java Project**.

**15.** In the New Java Project dialog name the project **Clients**, uncheck the **Use default location**, choose the **Clients** folder located in src/ of the PartsUnlimitedMRP repository, and click **Finish**.

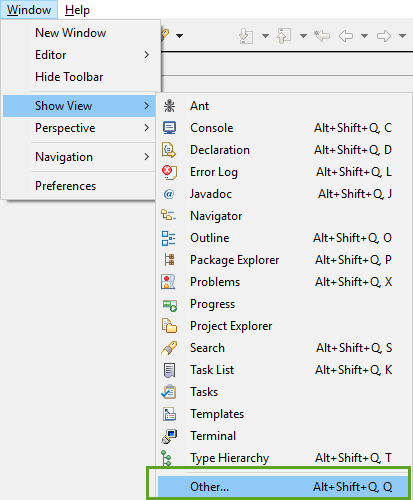


**16.** In the Package Explorer window, **right-click on Clients** and then select **Configure -> Add Gradle Nature**.

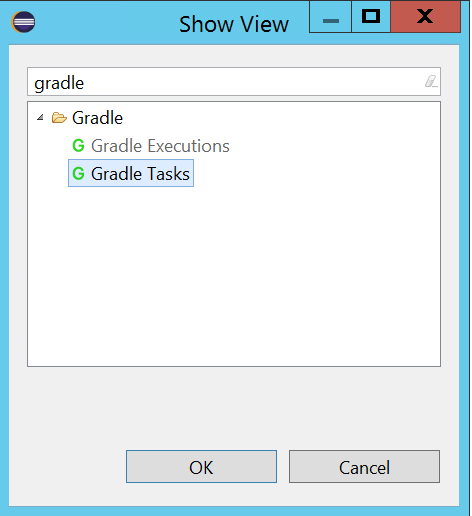
You should now have 3 projects in Eclipse that are all Gradle projects (signified by the G in the glyph of the projects in the Package Explorer window).



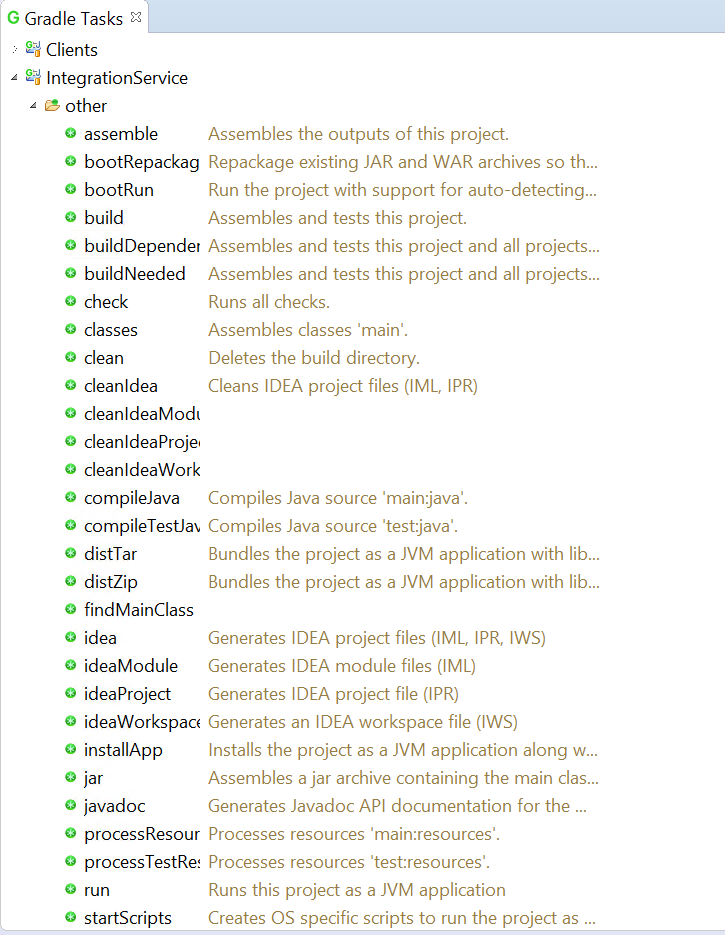
**17.** In Eclipse’s menu select **Window -> Show View -> Other…**



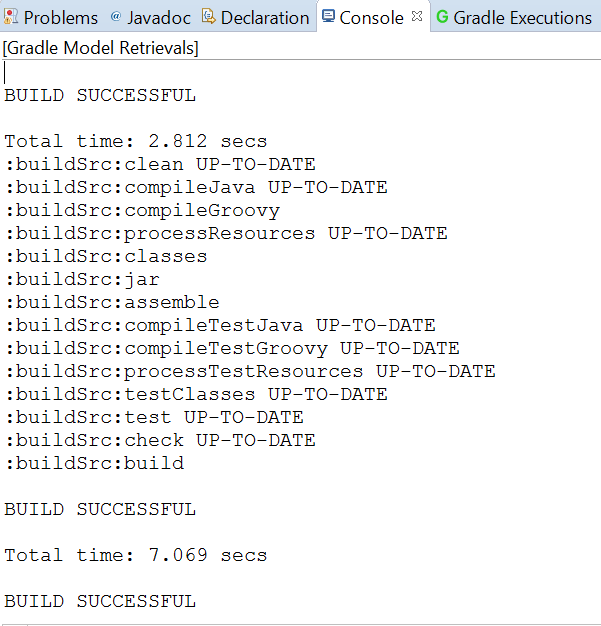
**18.** Type gradle in the filter box, and select “Gradle Tasks” then click **Open**.



You can now select one of the three projects that were created, and be able to view all of the Gradle tasks in each on via the Gradle Tasks window.



The development environment is now set up, and you should be ready to make write some automated tests. Go ahead and look over the different tasks - you may even want to try out the build task.

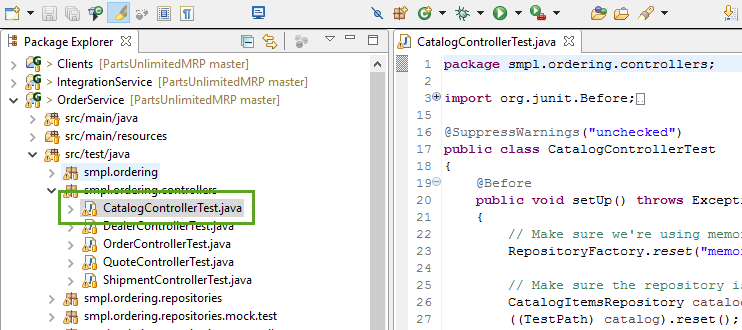


### **Task 2. Add JUnit tests to Parts Unlimited MRP App**

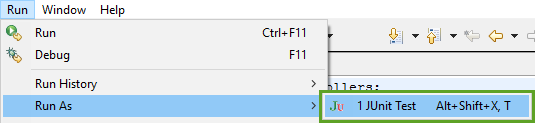
This task will focus on creating a unit test to test a part of the Parts Unlimited App.

**1.** There are already many JUnit tests that exist within the various projects of the Parts Unlimited MRP App. **Open up the file** in the **OrderService** Eclipse project located here:

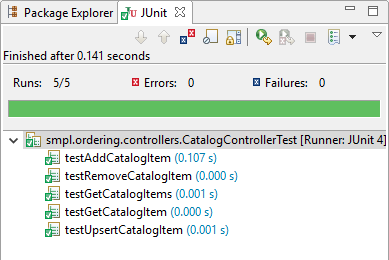
src/test/java/smpl.ordering.controllers/CatalogControllerTest.java



**2.** Lets go ahead and run these tests. In the menu select **Run -> Run As -> JUnit Test**.



This should open up the JUnit window and give you a summary of the test run.



**3.** Paste the following code on **line 31** of the CatalogControllerTest.java file :

@Test

public void testAddCatalogItems() throws Exception

{

ResponseEntity<List<CatalogItem>> oldItems = controller.getCatalogItems();

int sizeBeforeAdds = 0;

if (oldItems.getBody() != null)

{

sizeBeforeAdds = oldItems.getBody().size();

}

ResponseEntity<CatalogItem> response =

controller.addCatalogItem(new CatalogItem("ACC-0123", "Storage", 15.75, 5, 2));

assertNotNull(response);

assertEquals(HttpStatus.CREATED, response.getStatusCode());

response =

controller.addCatalogItem(new CatalogItem("ACC-0124", "Storage", 11.25, 7, 6));

assertNotNull(response);

assertEquals(HttpStatus.CREATED, response.getStatusCode());

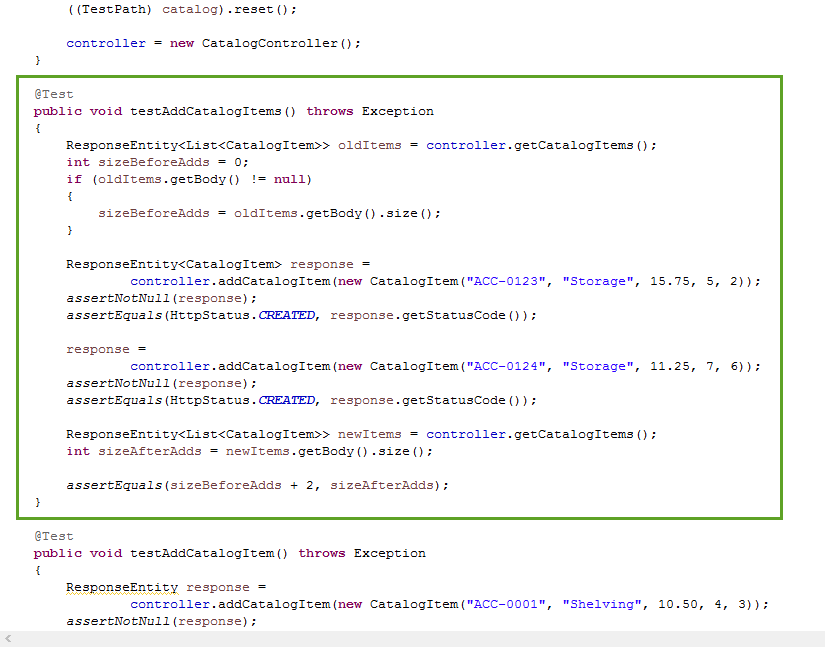
ResponseEntity<List<CatalogItem>> newItems = controller.getCatalogItems();

int sizeAfterAdds = newItems.getBody().size();

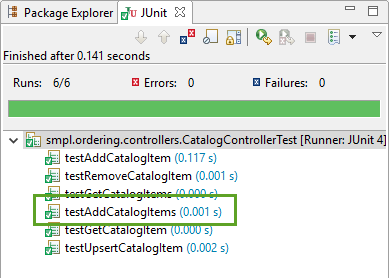
assertEquals(sizeBeforeAdds + 2, sizeAfterAdds);

}

The final result should look something like this:



**4.** Go ahead and re-run the JUnit tests from the menu **Run -> Run As -> JUnit Test**.



You should now be able to see that the test in the JUnit Window.

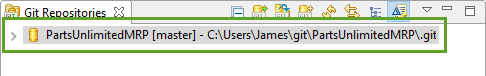
### **Task 3. Run the automated tests in an automated build**

Now that we have the automated tests written, and successfully running, it’s important we push the new tests to the shared repository. This will allow other team members to benefit from the automated test and give the automated build more verification that our code is working as expected.

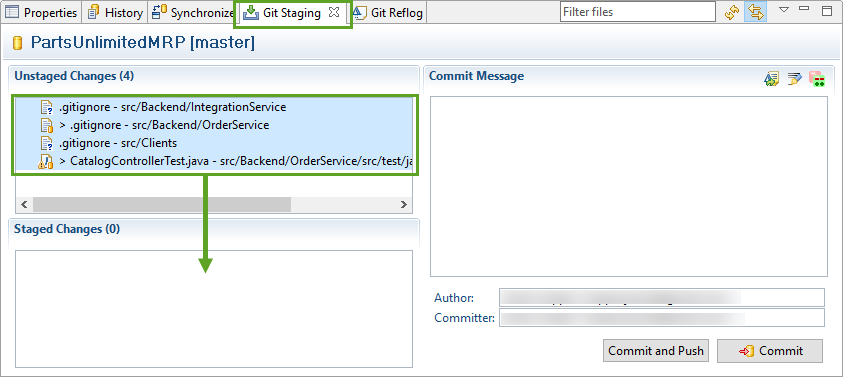
**1.** In Eclipse, **switch to the Git perspective** in the top right of the screen.



**2.** In the Git Repositories window, select the PartsUnlimitedMRP git repository.



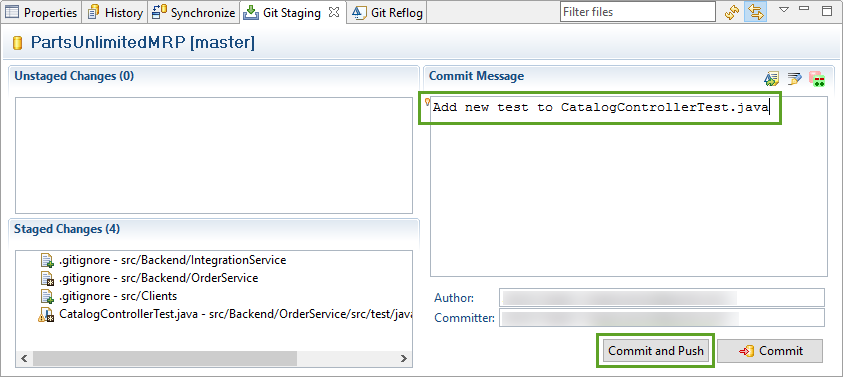
**3.** Select the **Git Staging** window. **Highlight** all of the Unstaged Changes and **drag and drop them into Stage Changes**.



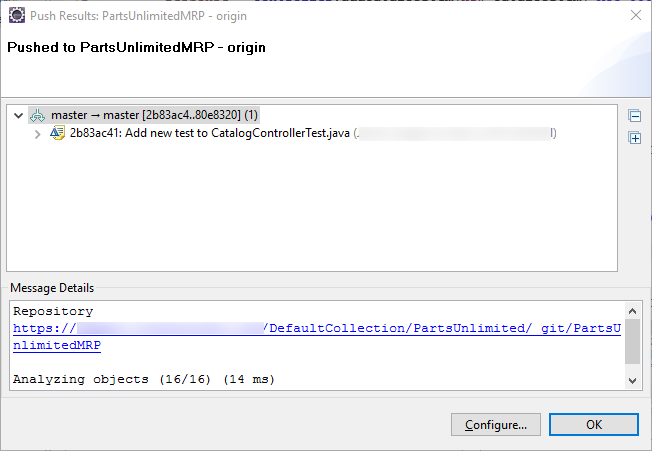
Note: Three .gitignore flies have changed. Eclipse automatically added the bin/ folder here, so any binaries that are built locally will not be tracked. You can choose not to stage these.

**4.** Add the following as a commit message, and then click **Commit and Push**.

Add new test to CatalogControllerTest.java



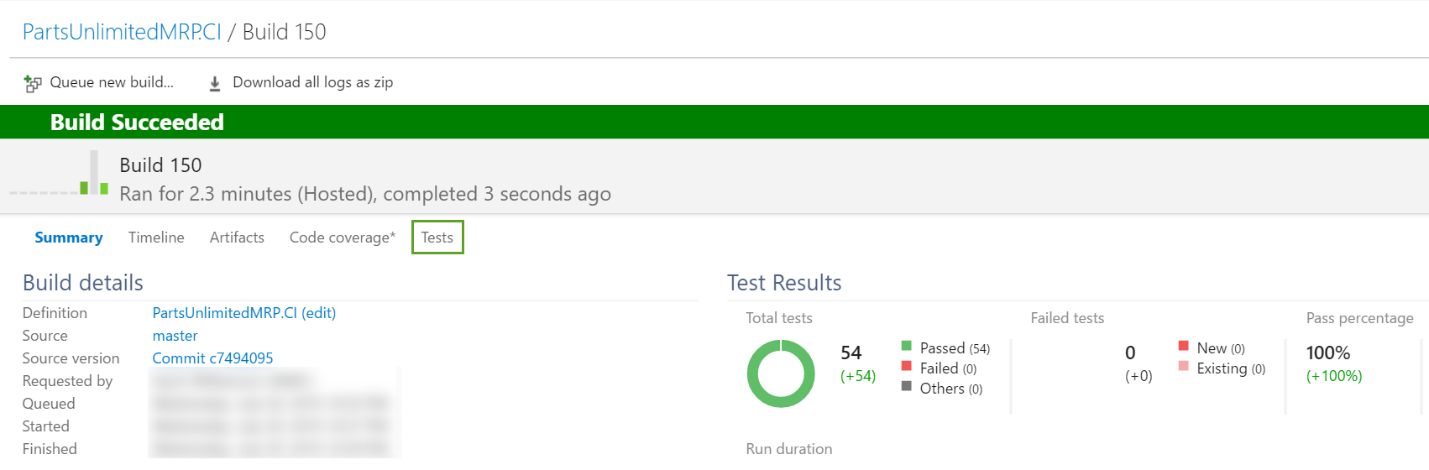
The Push Results window should pop up after you have successfully pushed:



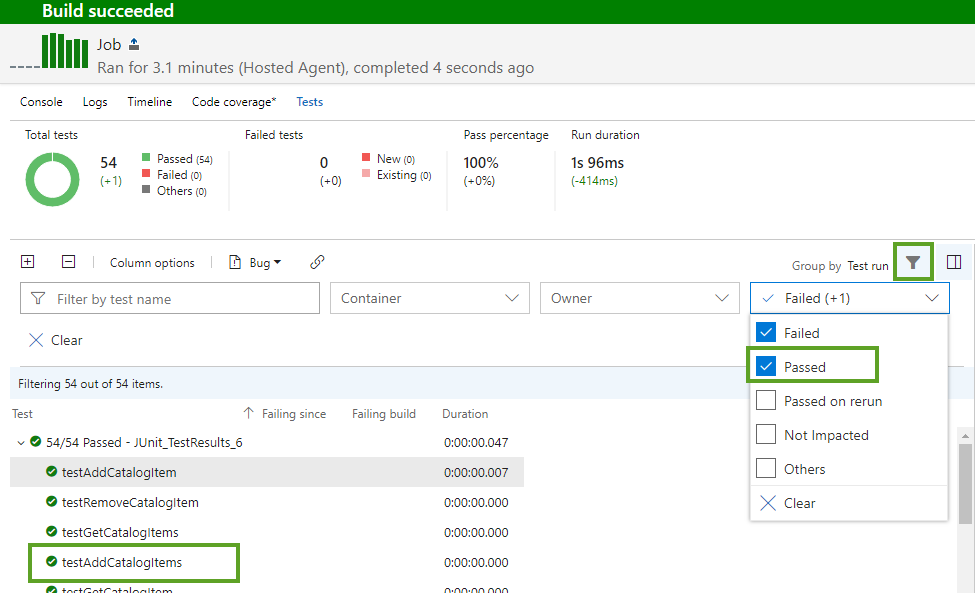
**5.** Navigate to the **BUILD** hub in your Visual Studio Team Services Team Project, and then click the **Queued** tab. You should see that your continuous integration build was triggered, and is running.

Once the build is done, open up the build summary page by clicking the build number.

**6.** On the build summary page, click to open the **Tests** tab, note that 54 tests have run successfully.



**7.** Click the filter icon, and select **Passed** in the dropdown area, to include the passed tests (by default failed tests are shown), and you should now see that your new automated test is a part of the Continuous Integration Build.



### **Summary**

In this lab, you set up your development environment to work with Gradle, you added a new JUnit test to the Parts Unlimited MRP App, and you were able to see that reflected in the automated build process.