Tutorial 11 – Eclipse GitHub Workflow 1

Contents

[1 Introduction 1](#_Toc78196917)

[2 Person 1 – Establish Local Version Control 2](#_Toc78196918)

[3 Person 1 – Commit Locally 5](#_Toc78196919)

[4 Person 1 – Push to GitHub 7](#_Toc78196920)

[5 Person 2 – Clone Remote Repo 9](#_Toc78196921)

[6 Submission 15](#_Toc78196922)

# Introduction

This lab serves two purposes.

1. A reference for how your team can use GitHub.
2. A tutorial for you as an individual illustrating how your team can use GitHub.

This tutorial will introduce you to working with Eclipse and GitHub to do Java development. The most recent versions of Eclipse come with a plugin, EGit which integrates Git into Eclipse.

Tutorials 11 & 12 are very similar. I believe the major differences are:

|  |  |  |
| --- | --- | --- |
| **Activity** | **Tutorial 11** | **Tutorial 12** |
| Create local version-controlled project | Configure git in Eclipse and then use Team, Share Project | Only uses: Team, Share Project. Not sure why the configuration for 11 was not needed. |
| Location of local repo | *users/youID/git/myRepo* | Inside the project folder (not recommended, but OK) |
| Initial Pull | Cloning | Download zip file |
| Pulling & Pushing | master branch (not the approach we should use) | Branch off of master |

Note: Both Tutorials 11 & 12 define a workflow that your team can use for the group project. Elements of either can be interchanged. The point is to show you the different possibilities and then you can choose and/or create the best workflow for you and your team.

One thing I know these tutorials do not address is how to roll back (revert) changes in Eclipse.

This is a workflow you can use for your team project.

1. One person creates a local Java project in Eclipse that is version controlled and pushes it to GitHub. This project may simply have one class with no code. The object, as explained next is so that other group members can clone this project and begin their own work.
2. Other members clone this project, work locally, commit locally, then push changes.
3. At any time, other members can pull changes others have committed. Thus, all members can have a complete, local copy of the remote repository.

This tutorial starts completely from scratch. You will create new local and remote repositories.

To make the steps in the tutorial more concrete, we need to think about your group project where you will be on a team with 2 or more members. We refer to *Person 1* as the team member who will initialize the project’s GitHub repo. One person will do this one time. We refer to *Person 2* as any other person in the group. We demonstrate what *Person 2* must do to use the GitHub repo locally. This lab is broken down into activities that these two roles will carry out.

This tutorial is based on: <https://eclipsesource.com/blogs/tutorials/egit-tutorial/>

Finally, as we move towards the project, you (and your team) MUST have a workflow that employs version control. As the labs showed, you can do it a number of ways:

1. Use the command line to clone, pull, commit, push, etc while you work in your IDE of choice (Eclipse, IntelliJ).
2. Use Eclipse (or IntelliJ) connected directly with GitHub. This tutorial, and the next show this approach.
3. Use GitHub desktop to clone, pull, etc while you work in your IDE of choice (Eclipse, IntelliJ). This is addressed in a later tutorial.
4. Last resort (and not the best choice) is to use no local version control. Simply download code, work in your favorite IDE, then drag/drop into GitHub. If you use this approach, you need to make sure and commit often (e.g. drag/drop).

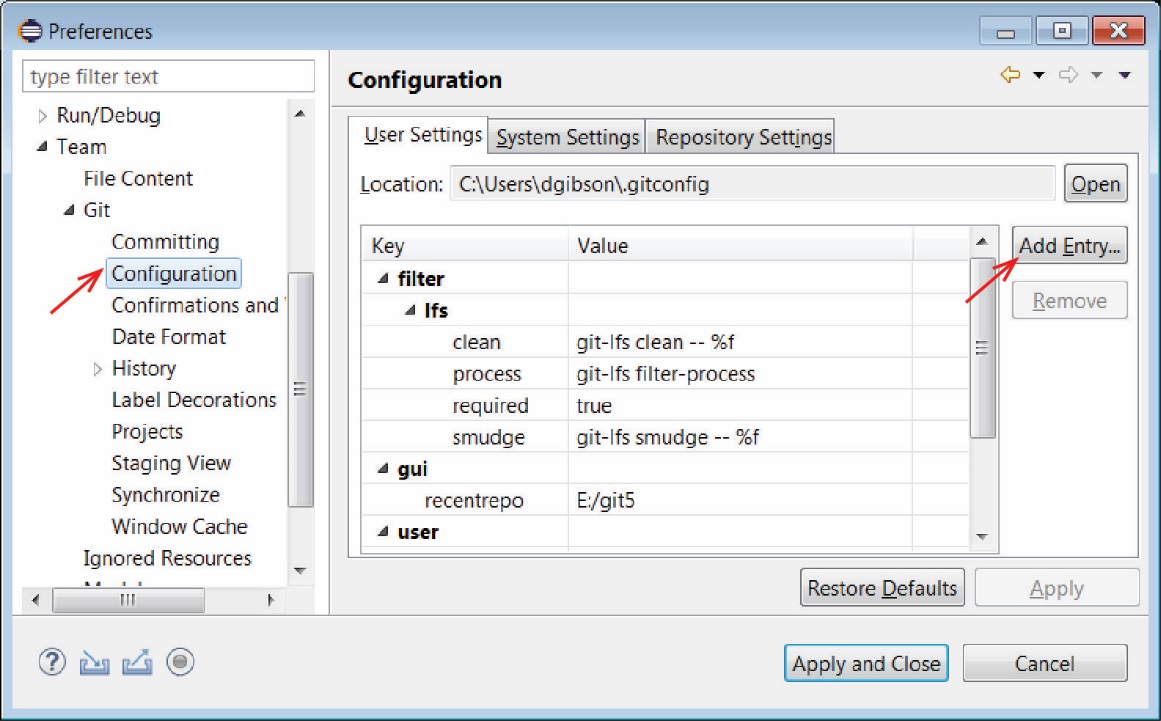
You need to confirm that all members of your team have a workflow.

# Person 1 – Establish Local Version Control

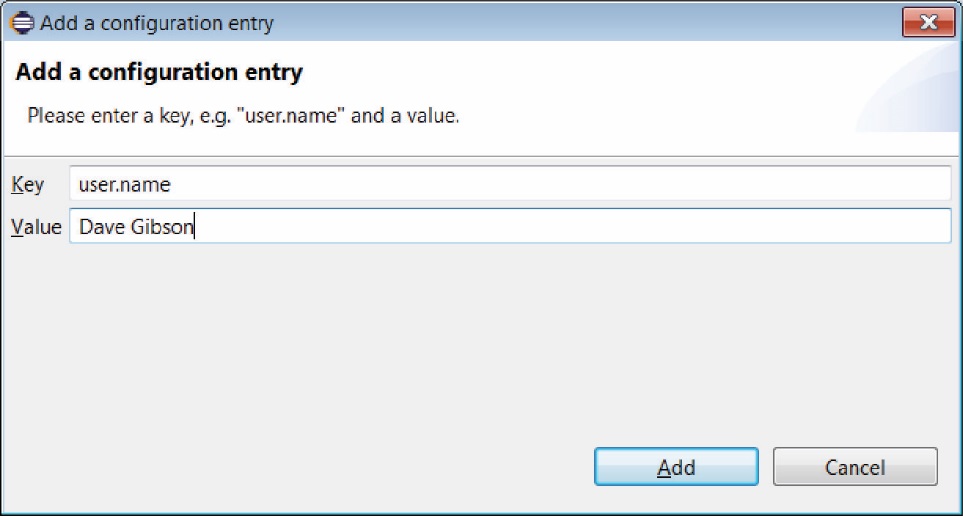
1. Do the following:
2. Create a folder named *eclipse\_ws* on your local computer.
3. Open Eclipse and set the workspace to this folder.
4. Create a Java project named: *Acme Systems*
5. Create a package named: *ver1*
6. Create a class named: *HelloWorld* (select checkbox to create a *main*)
7. Add a print statement to *main*.

System.***out***.println("Hello from Eclipse");

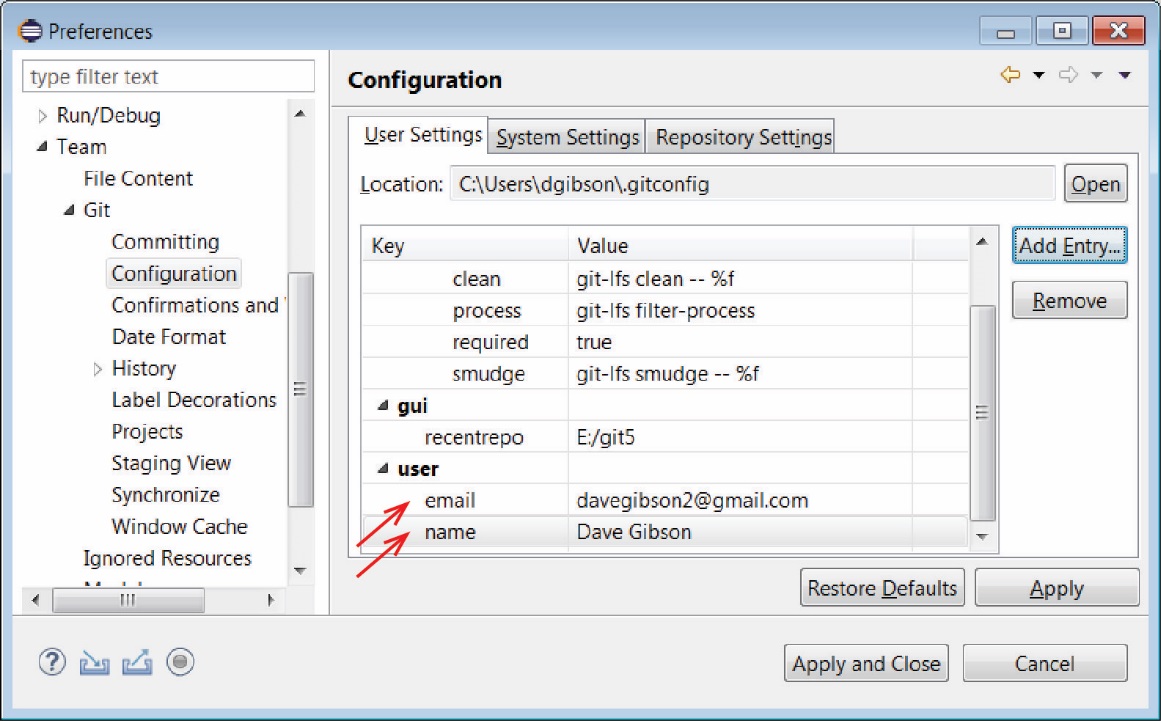
1. Setup up Eclipse to use version control locally. Do the following:
2. Choose: Window, Preferences, Team, Git, Configuration. Then, choose: Add Entry



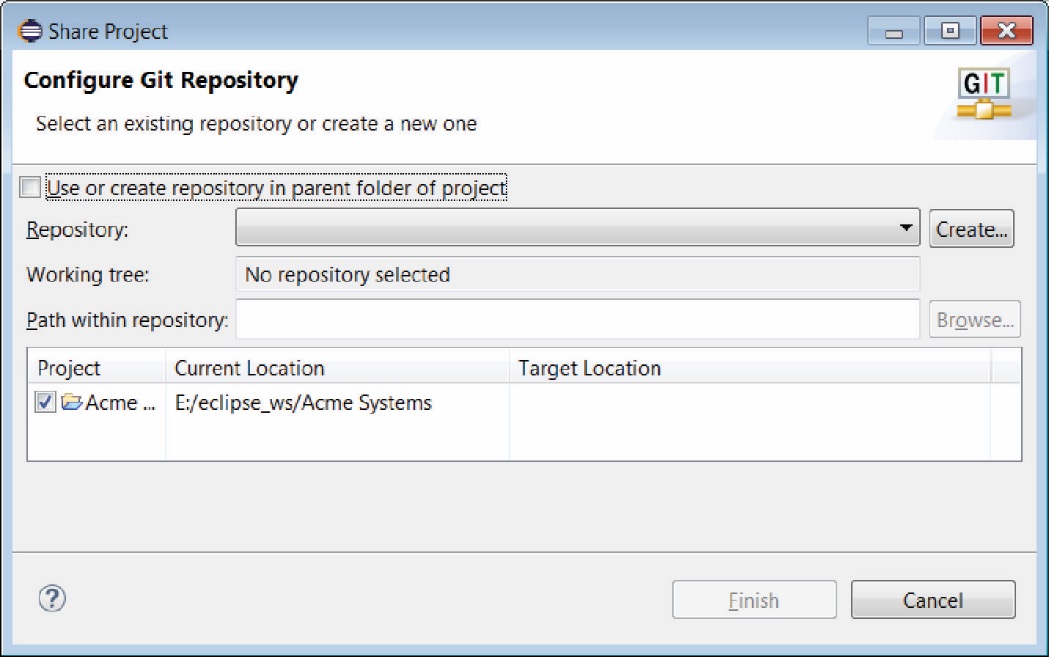
1. Enter *user.name* as Key and your name as Value and Add



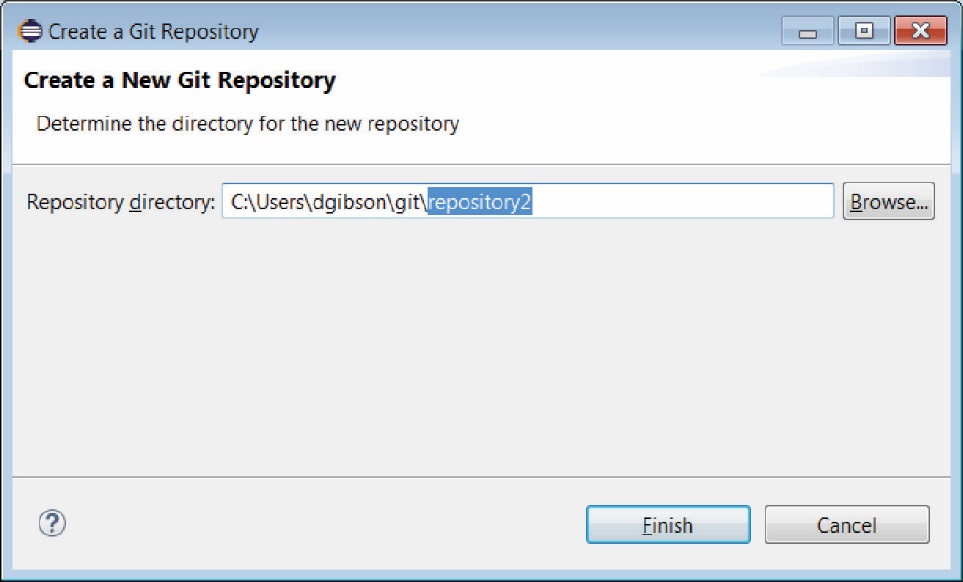
1. Choose: Add Entry and enter *user.email* as Key and your email address as Value. You will see the two entries as shown below.



1. Choose: Apply and Close
2. Do the following:
3. Right-click the project node and choose: Team, Share Project

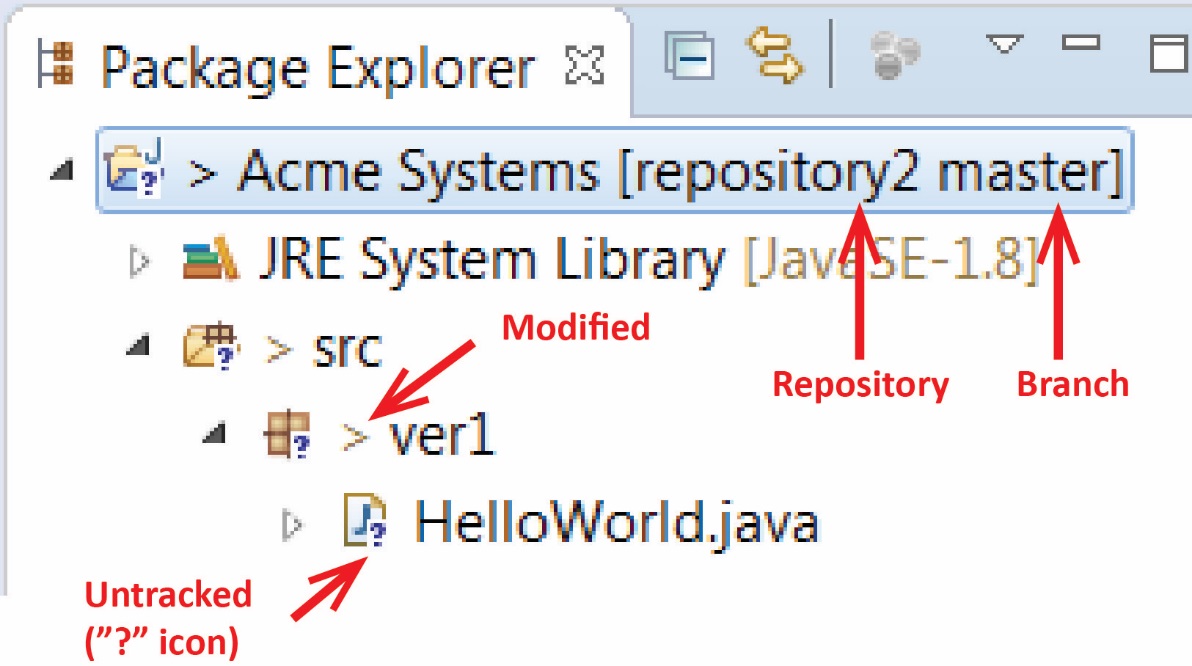


1. Choose: Create. Note that the repository will be stored in a location separate from the workspace, which is the recommended approach. We will accept the default location.



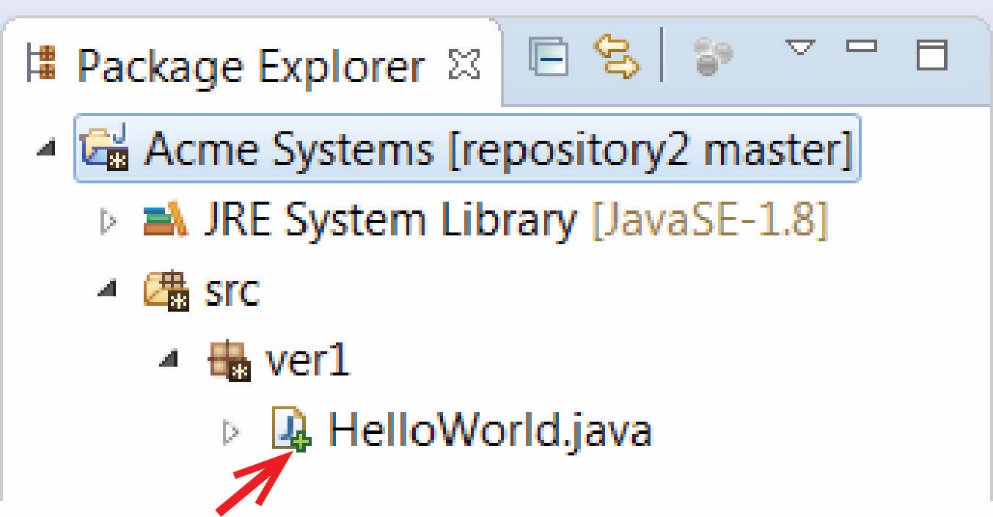
1. Choose: Finish, Finish. Note the following in the Package Explorer. To see all the different icon annotations, see, “Additional Information” towards the bottom of this page:

<https://eclipsesource.com/blogs/tutorials/egit-tutorial/>

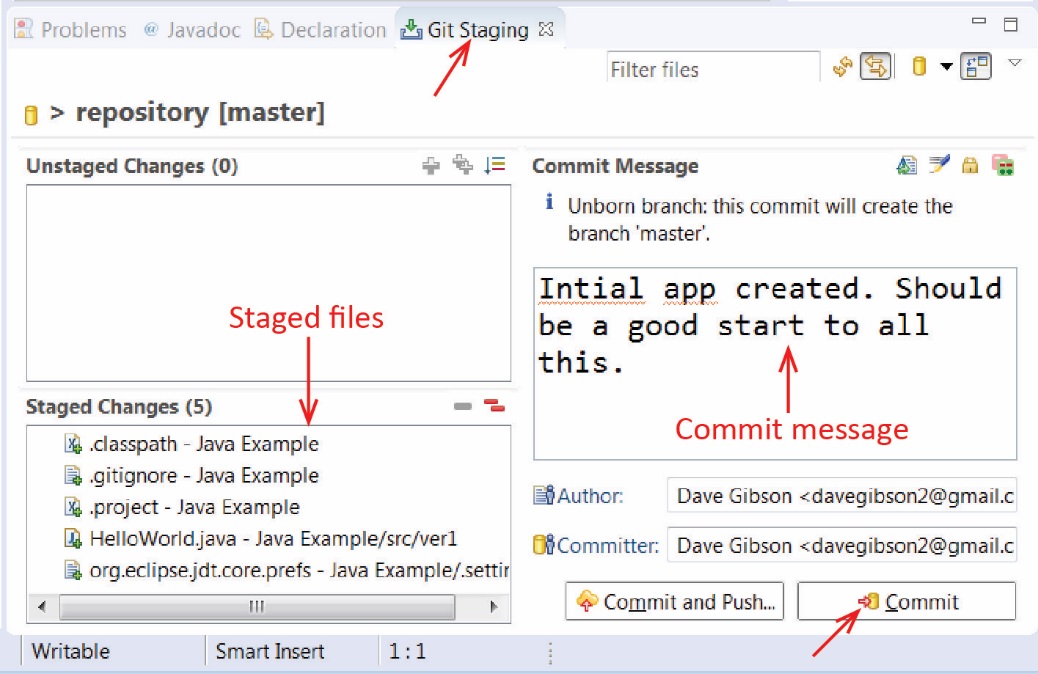


# Person 1 – Commit Locally

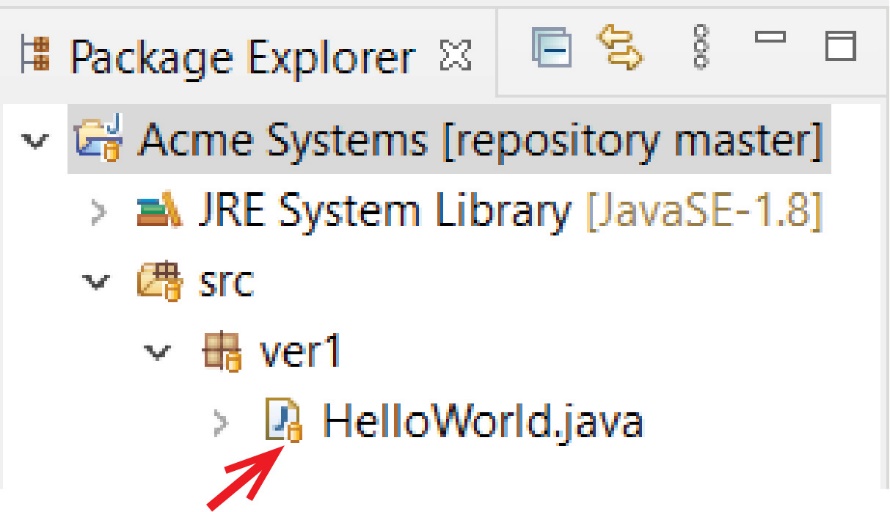
1. Next, we need to add the project to the index. Right-click the project (or the *HelloWorld* class) node and choose: Team, Add to Index. After this operation, the question mark should change to a plus symbol.



1. Next, we need to commit the files. Do the following:
2. Right-click the project node and choose: Team, Commit…
3. Enter a commit message (the first line should be headline-like, as it will appear in the history view) and hit the Commit button (NOT Commit and Push). If the commit was successful, the plus symbols will have turned into repository icons.

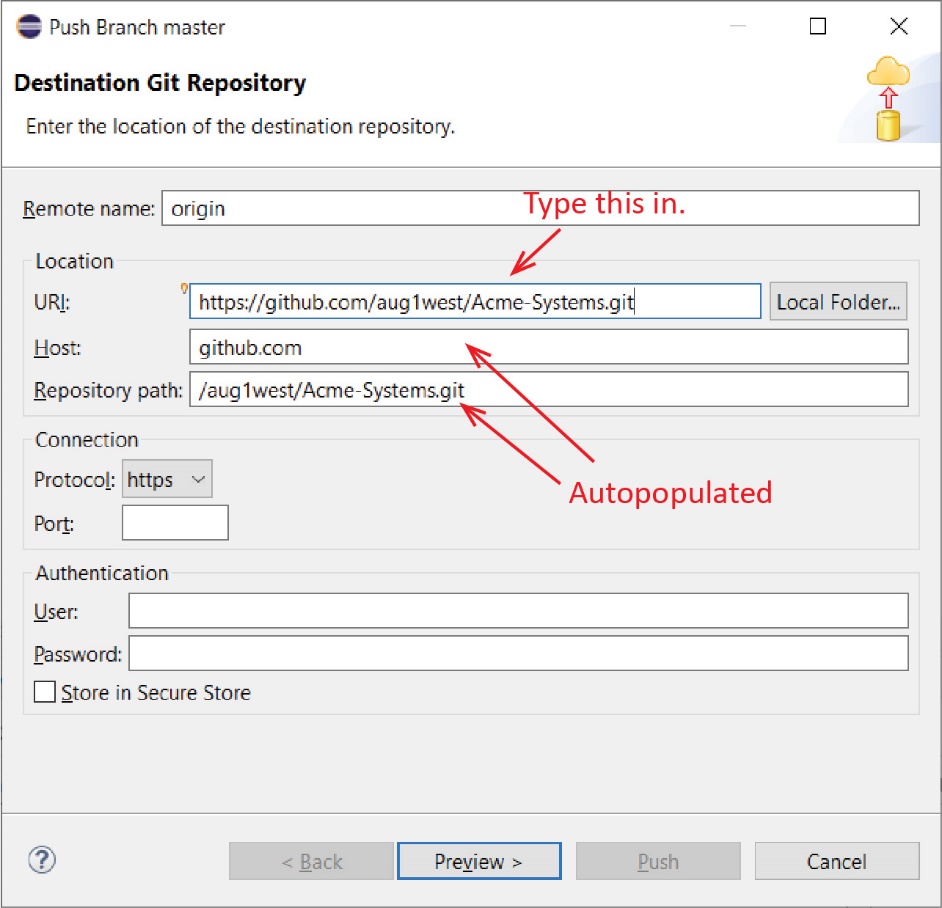


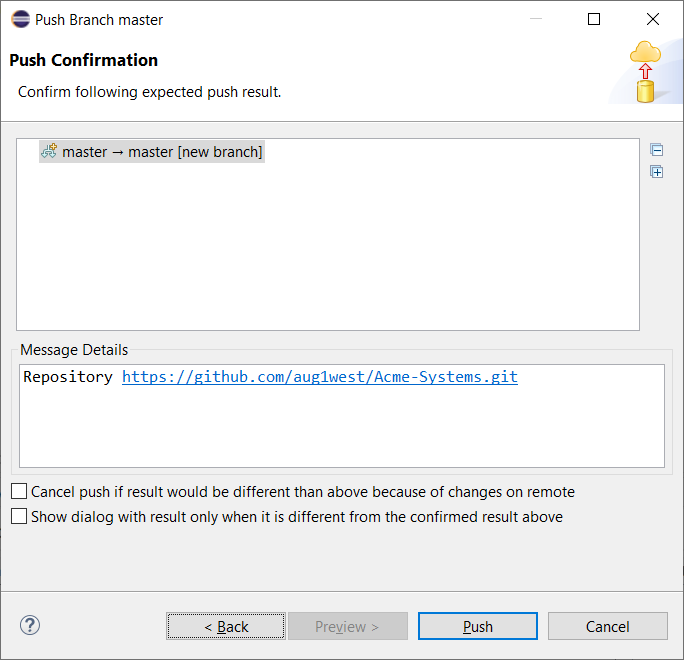
1. If the commit was successful, the plus symbols will have turned into a repository icon.



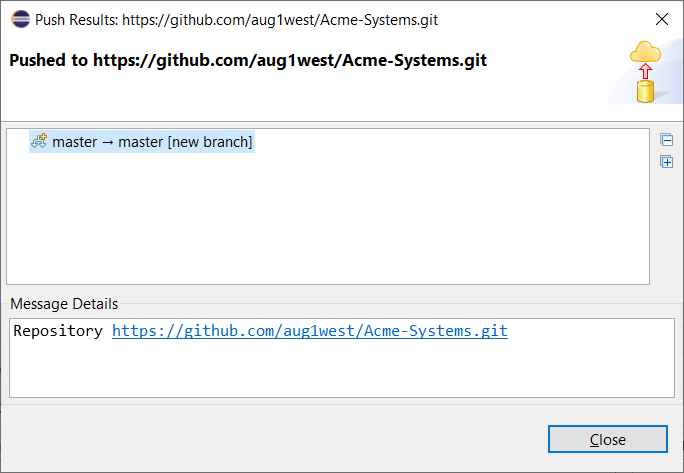
1. [Read, no action required] If you later realize that your previous commit was incomplete (e.g. you forgot to commit a file) or your commit message was wrong, you might want to use *Amend* feature. This will merge the current commit and the previous commit into one, so you don’t have to perform an extra commit (and maybe cause confusion). However, this should only be used if the previous commit hasn’t already been published to a shared repository. To do this, choose: Team, Commit…. Then, there is an *Amend* icon beside *Commit Message.*
2. Add a class named, *HelloDolly* and a print statement to *main.* The new file will, again, appear with a question mark.
3. Next, add this to the index and commit it by repeating steps 4 and 5 above.

# Person 1 – Push to GitHub

1. Go to GitHub and create a new repository named, *Acme-Systems.* This is where we will push our local project.
2. Do the following:
3. In Eclipse, right-click the project node and choose: Team, Push Branch ‘master’
4. Paste the URI from GitHub, other fields will be auto-populated.
5. Press Preview, look at dialog,
6. Press Preview again. You’ll be prompted for your github id and password. **Remember that your password is your PAT (See Lab 7, Step 6 if necessary)**



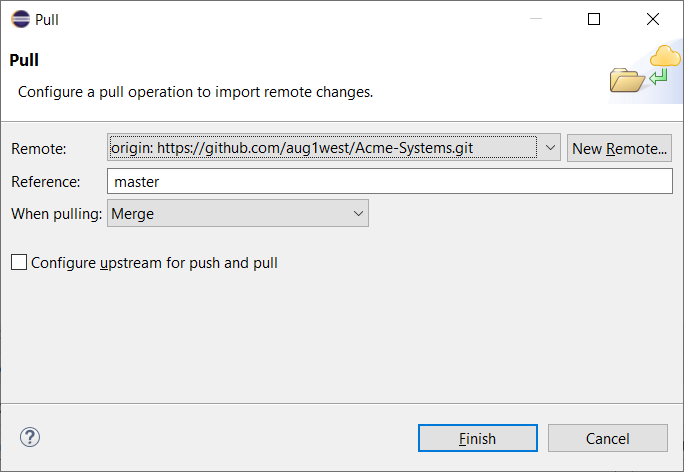
1. Finally, press: Push. The resulting dialog will be similar to the one shown below. When done, press: Close.



1. Do the following in GitHub:
2. Verify that your project was pushed.
3. Edit HelloWorld and add a print statement and Commit.

System.out.println("Hello from GitHub");

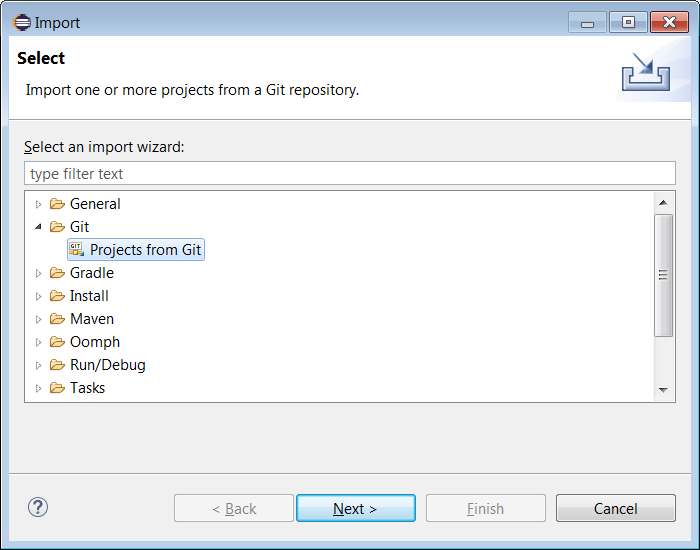
1. Pull from GitHub : Do the following in Eclipse:
2. Right-click the project node and choose: Team, Pull…



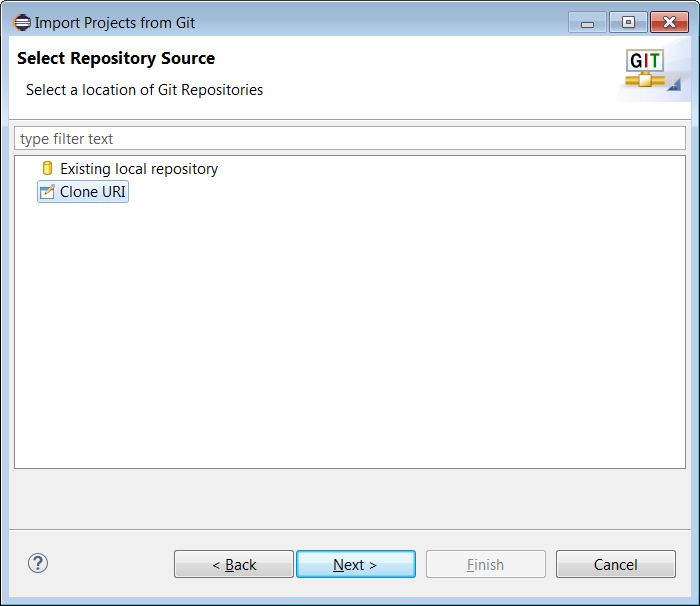
1. Press Finish, and then verify that the change was pulled to your local copy.

# Person 2 – Clone Remote Repo

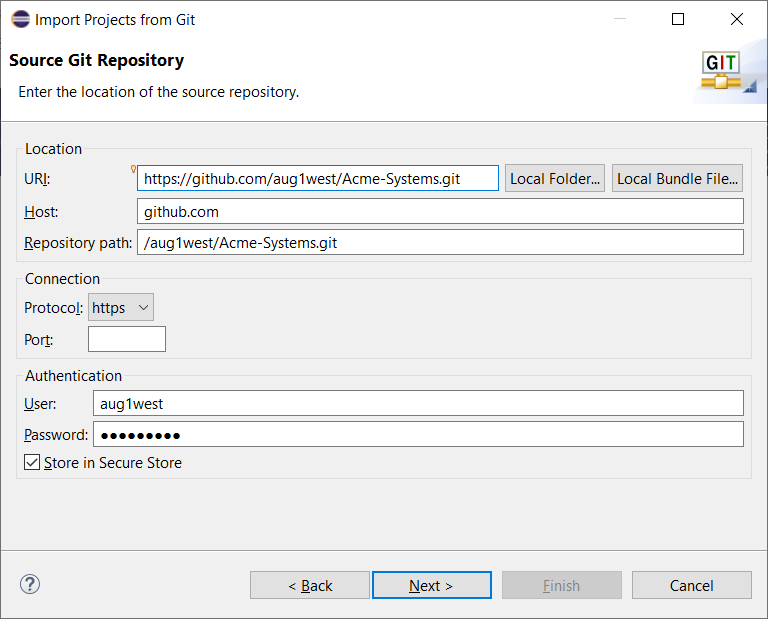
1. Do the following:
2. Create a folder named *eclipse\_ws2* on your local computer.
3. Open Eclipse and set the workspace to this folder.
4. Next, we will clone the *Acme-Systems* repository on GitHub to our local workspace, *eclipse\_ws2.* Do the following:
5. Choose: File, Import, Git, Projects from Git



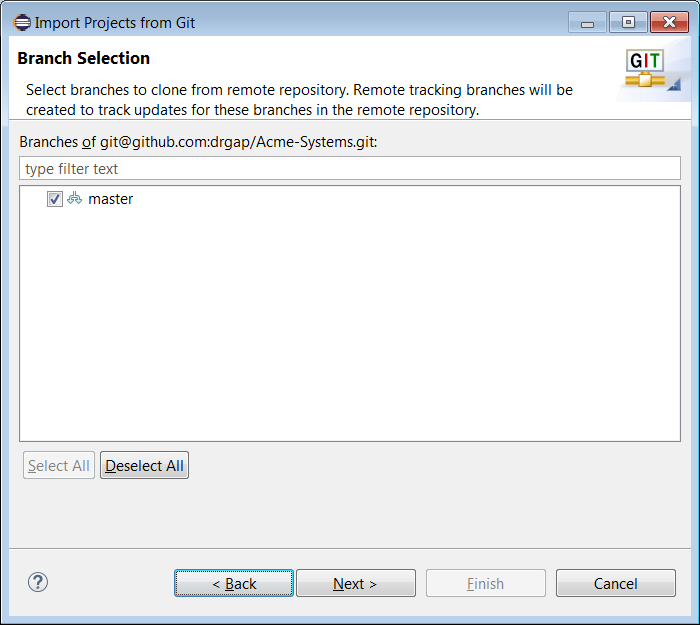
1. Choose: Next, then Clone URI



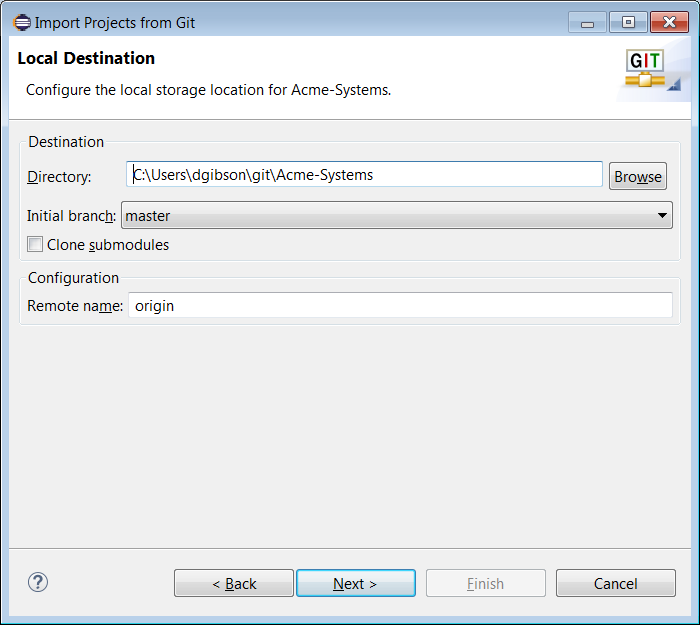
1. Choose: Next, and then supply the URI for the newly created repository



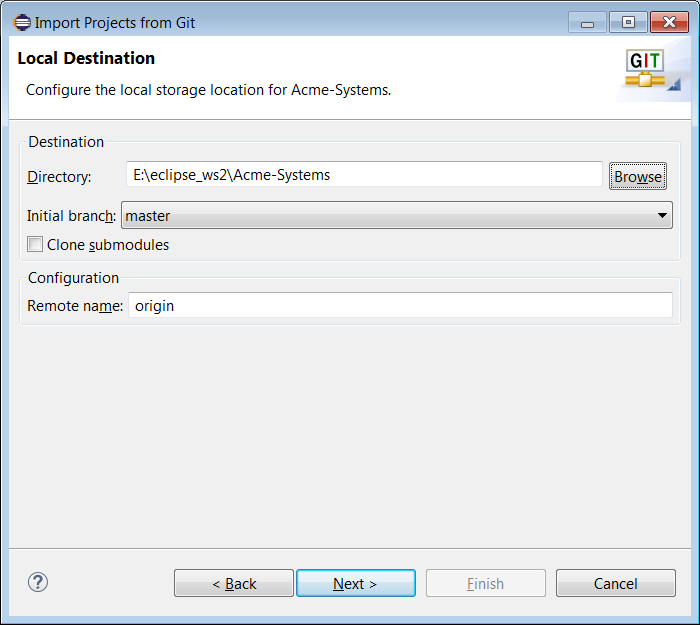
1. Choose: Next



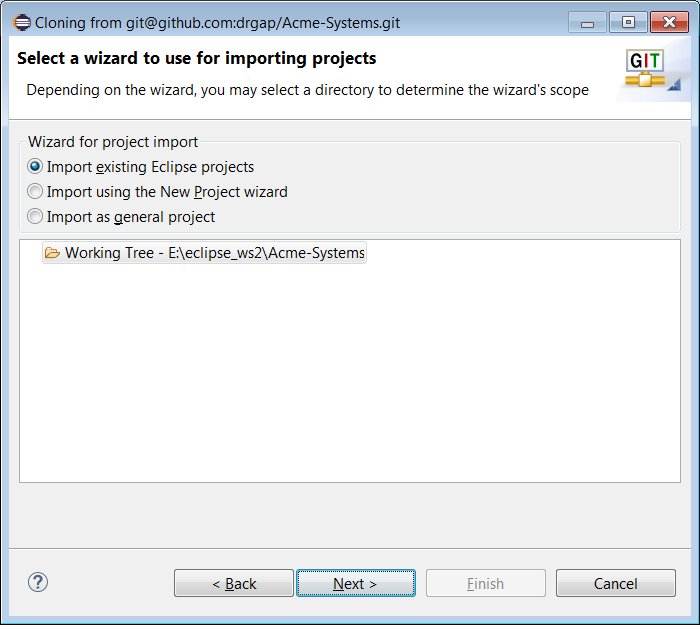
1. Choose: Next



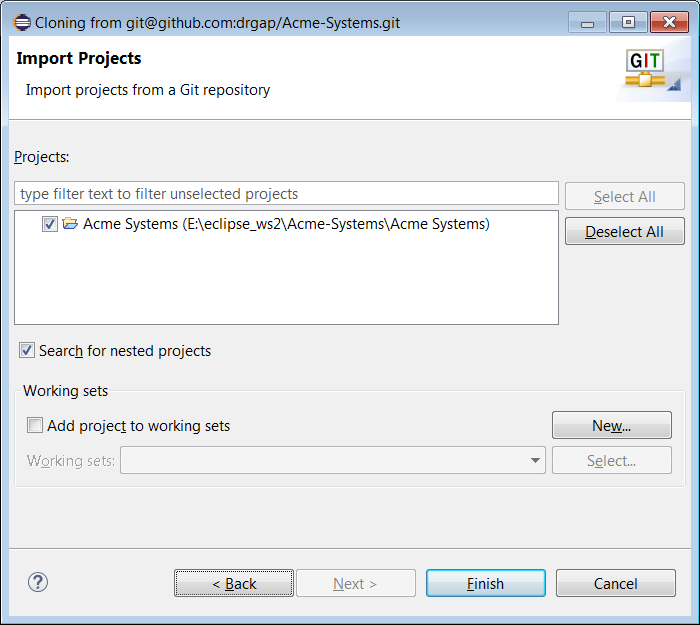
1. Choose: Browse, and navigate to your *eclipse\_ws2* folder



1. Choose, Next



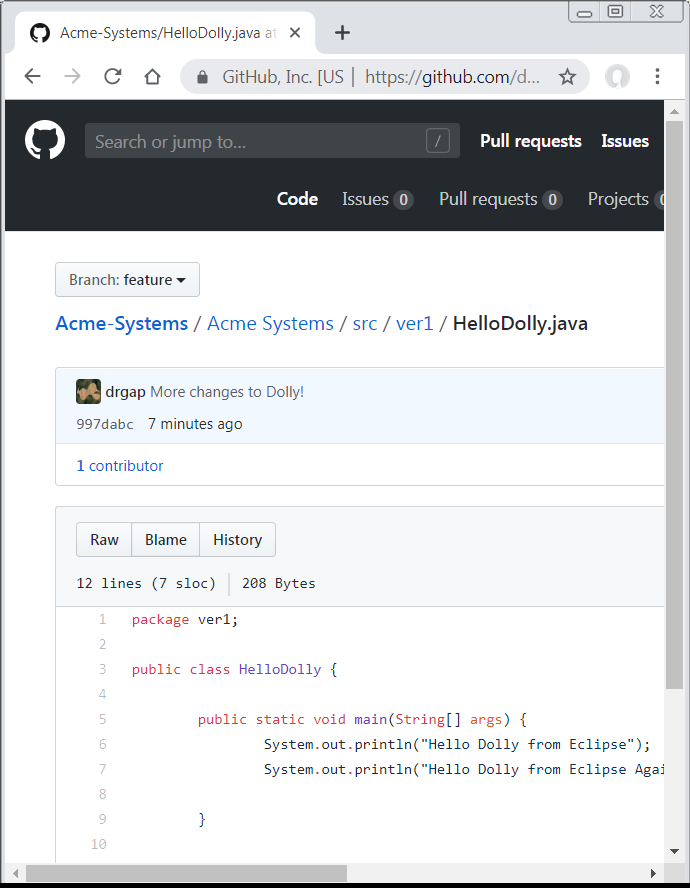
1. Choose: Next



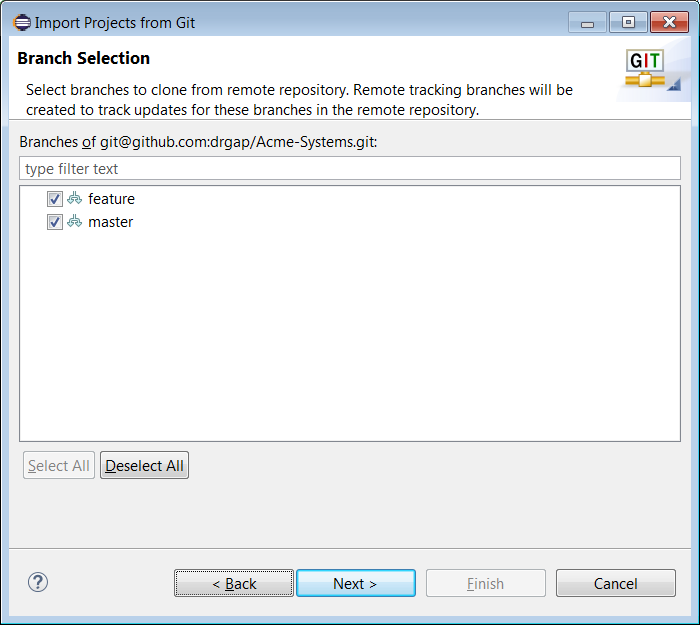
1. Choose: Finish. Your project should now be loaded in Eclipse and under local version control
2. Do the following in Eclipse:
3. Add a print statement to *HelloDolly*:

System.***out***.println("Hello Dolly from Eclipse Again");

1. Commit and then push changes to GitHub.
2. Verify that the changes made it to GitHub as shown below.



1. [Read, no action required] **Important note – You will not be cloning master when you work on your project. Remember that master is only for working code. You will create a branch on master, say, *feature* and clone it. To clone a branch, you will follow the steps above. At one of the dialogs, you will see a dialog like the one shown below where you can choose the branch you want to clone.**



# Submission

1. **Do the following:**
2. Make a screen shot similar to 15c above.
3. Place the image in the *HW VCS* document in the appropriate place.
4. The image should easily readable without zooming in or out.