Github Tutorial

Table of Contents

[Introduction 2](#_Toc442988274)

[What is GitHub and why do we want to use it? 2](#_Toc442988275)

[How will we be using GitHub? 2](#_Toc442988276)

[Basic concept of GitHub. 2](#_Toc442988277)

[How to set up a GitHub account? 3](#_Toc442988278)

[Getting code from Github 3](#_Toc442988279)

[How to work with an existing git project (code already in Github)? 3](#_Toc442988280)

[Adding new project to Github 14](#_Toc442988281)

## Introduction

### What is GitHub and why do we want to use it?

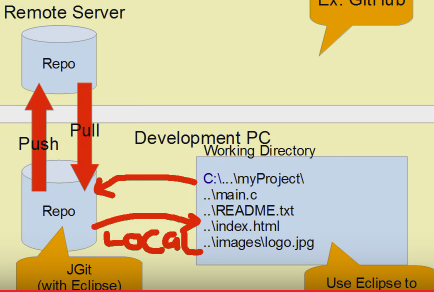
GitHub is an application that allows multiple people to work on a coding project together. Through the tools that GitHub provides, a team will be able to work on a project in a coordinated and organized way. Furthermore, GitHub allows projects to be backed up on the “cloud” and accessed from anywhere with internet connection. Github is the de facto standard source code repository used by the software development industry.

### How will we be using GitHub?

GitHub is free and everyone who will be coding should set up an account. The team has an account on GitHub already (called Team4456). The team account is where we will hold all of the “master” code. From your personal accounts, you will be able to access all the “master” code and add or change it. If you add or change any of the master code you can submit the code to the team account and one of the moderators will either approve it and add it to the master code, or send it back for revision.

### Basic concept of GitHub.

The basic concept of GitHub is that you have a working directory on your computer (shown as the blue box in the lower right hand corner of the picture below), you have a local repository (or “Repo” shown as blue cylinder in lower left), and you have a remote repository (blue cylinder in top left). From your working directory you can “commit” your code to a local repository stored somewhere on your computer. If you wish to put the code online (onto your GitHub account) you can “push” the code up to your remote repository (your online Github account). You can also “Pull” code down from your remote repository. Keep the picture below and the terms described above (commit, push, pull) in mind.



\*Picture taken from https://www.youtube.com/watch?v=C0bFLGJqnI8&list=PL-suslzEBiMo0B5RcAikOaqDLKoG9Okub

### How to set up a GitHub account?

Go to [www.github.com](http://www.github.com), and click the “sign up” button in the upper right hand corner. Select a username, email, and password, and afterwards, select the free account option (this should be the default option anyways).

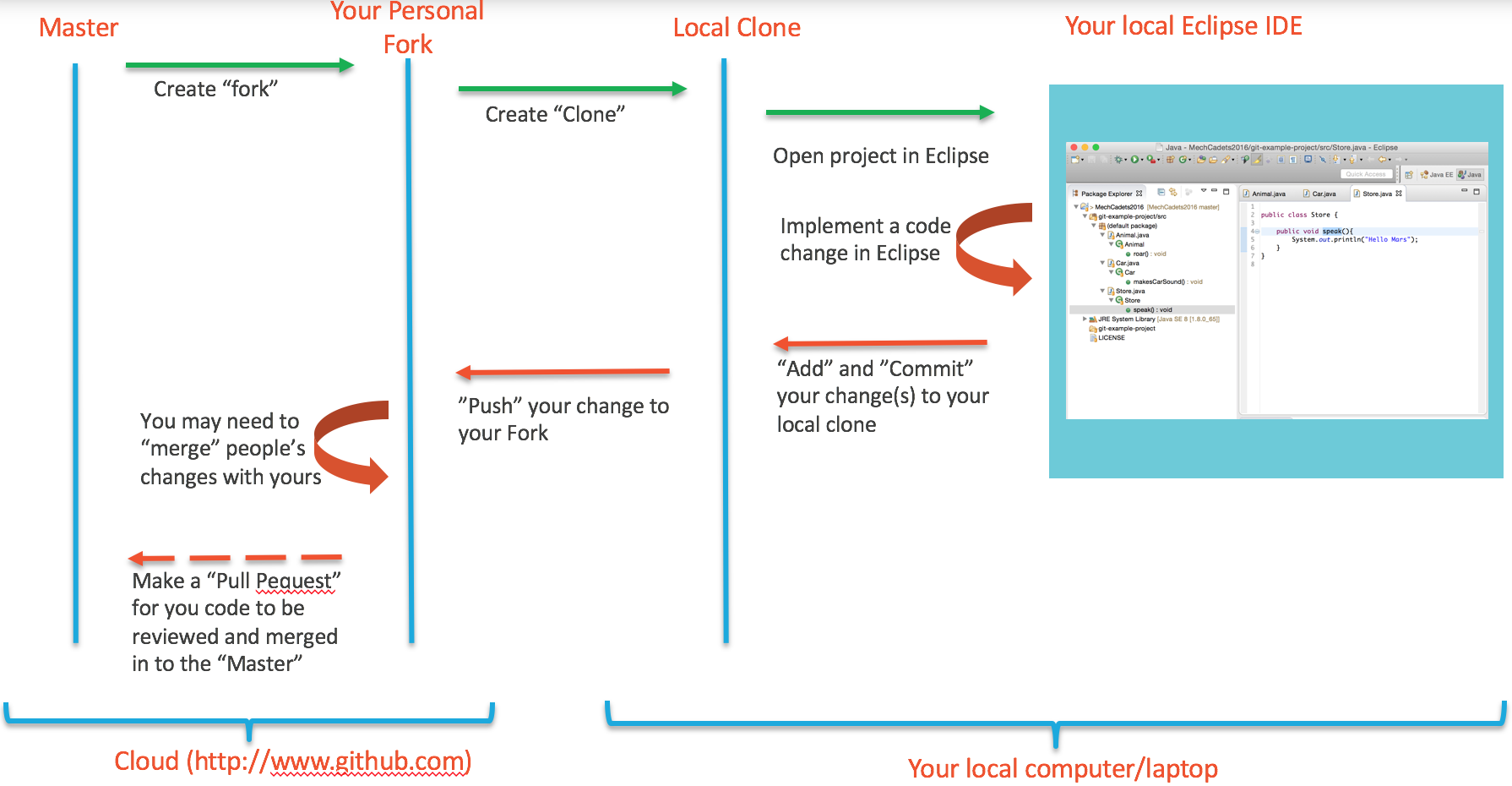
## Getting code from Github

### How to work with an existing git project (code already in Github)?

More often than not, a developer will be joining an ongoing project that already has source code in Github. In this common scenario, the new developer will be:

1. fetching the current code from the “Master” repository on Github.
2. making changes to the code, and
3. adding those changes back into the “Master” repository.

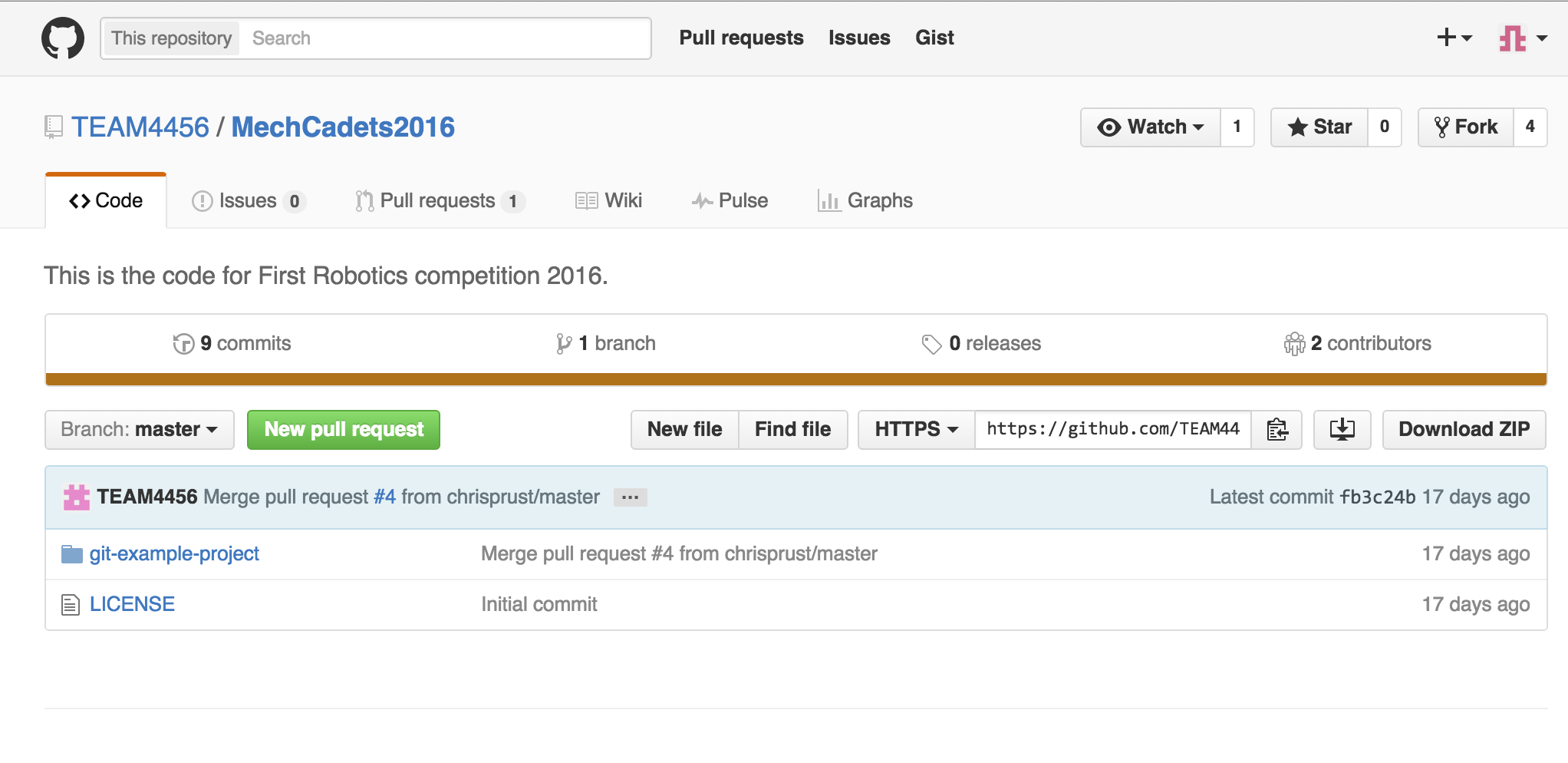
This scenario can be depicted as follows:

****

**The following is a step-by-step walkthrough:**

Step 1:

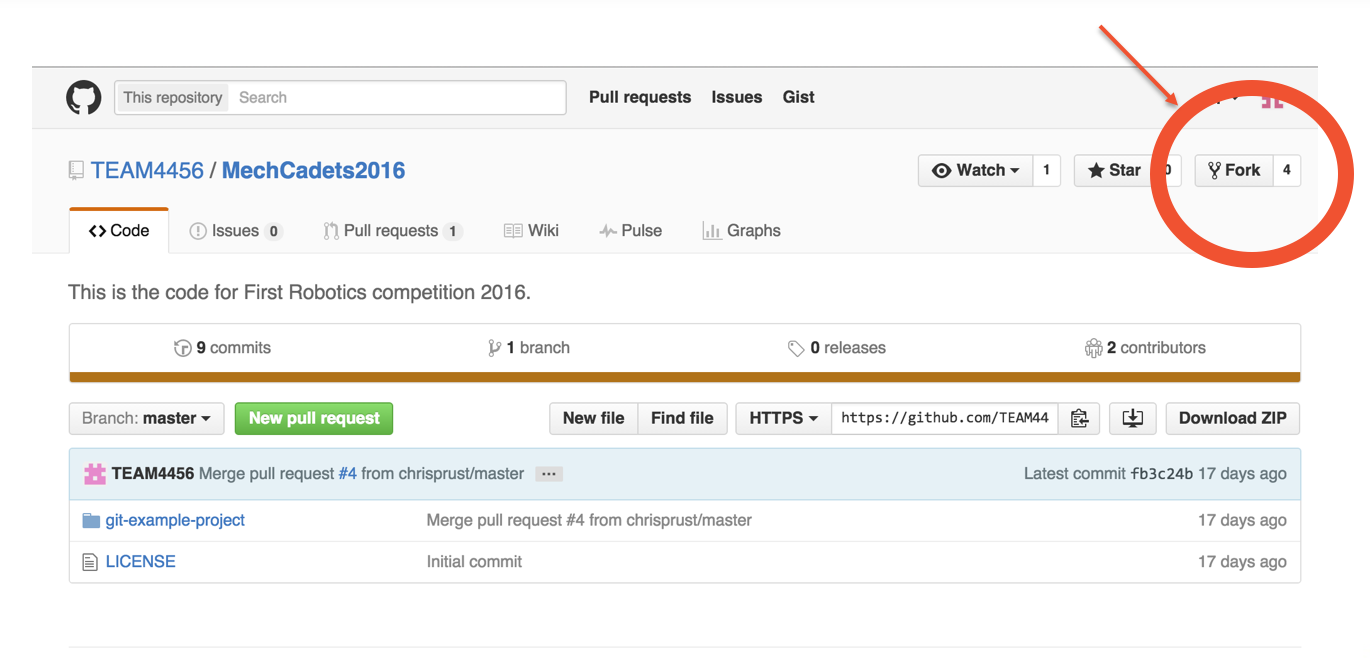
Find the “Master” repository. In our case, it’s <https://github.com/TEAM4456/MechStorm2016/>



Step 2:

Create a “Fork” using the top-right button on the Github interface. Github will create a new repository, which is a copy of the Master. This “Fork” repository belongs to you. That is, you manage all changes to it.

The “Fork” will have the URL: [https://github.com/<your\_github\_username>/MechStorm2016/](https://github.com/%3Cyour_github_username%3E/MechStorm2016/)



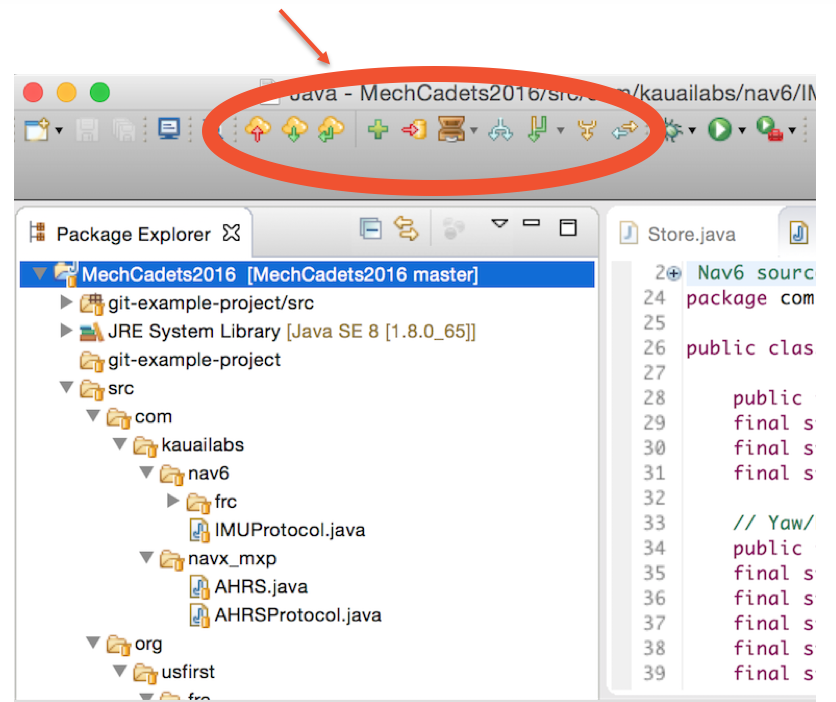
For example, in my case, it created <https://github.com/mevsam/MechStorm2016>

Step 3:

Setup your Eclipse with the “Git” shortcut icons. In Eclipse, go to:

1. Window -> Perspectives
2. Then “Customize Perspectives”
3. Select “Git” the checkbox.
   * If you get an error, expand the “Git” section, and select one of the sub-items uner “Git”. In the subsequent popup, select “Git”.

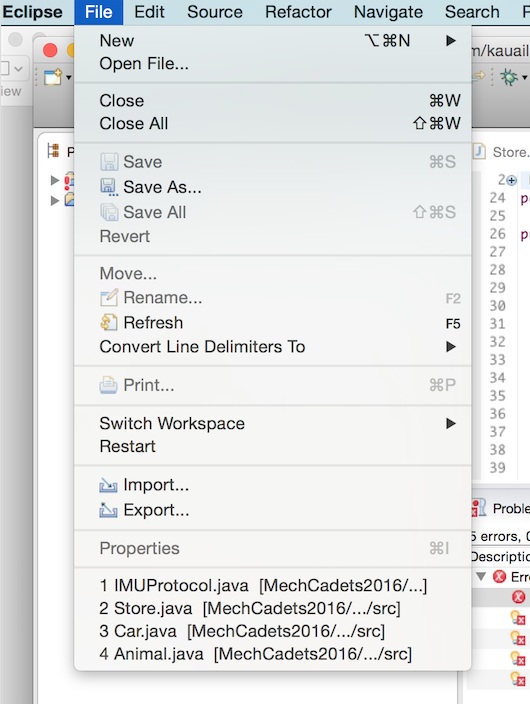
You should now see the “Git” shortcuts at the top of your Eclipse window.



Step 3:

Create a “Clone” on your local computer.

To do this, open Eclipse on your laptop/PC, and click on File -> Import…

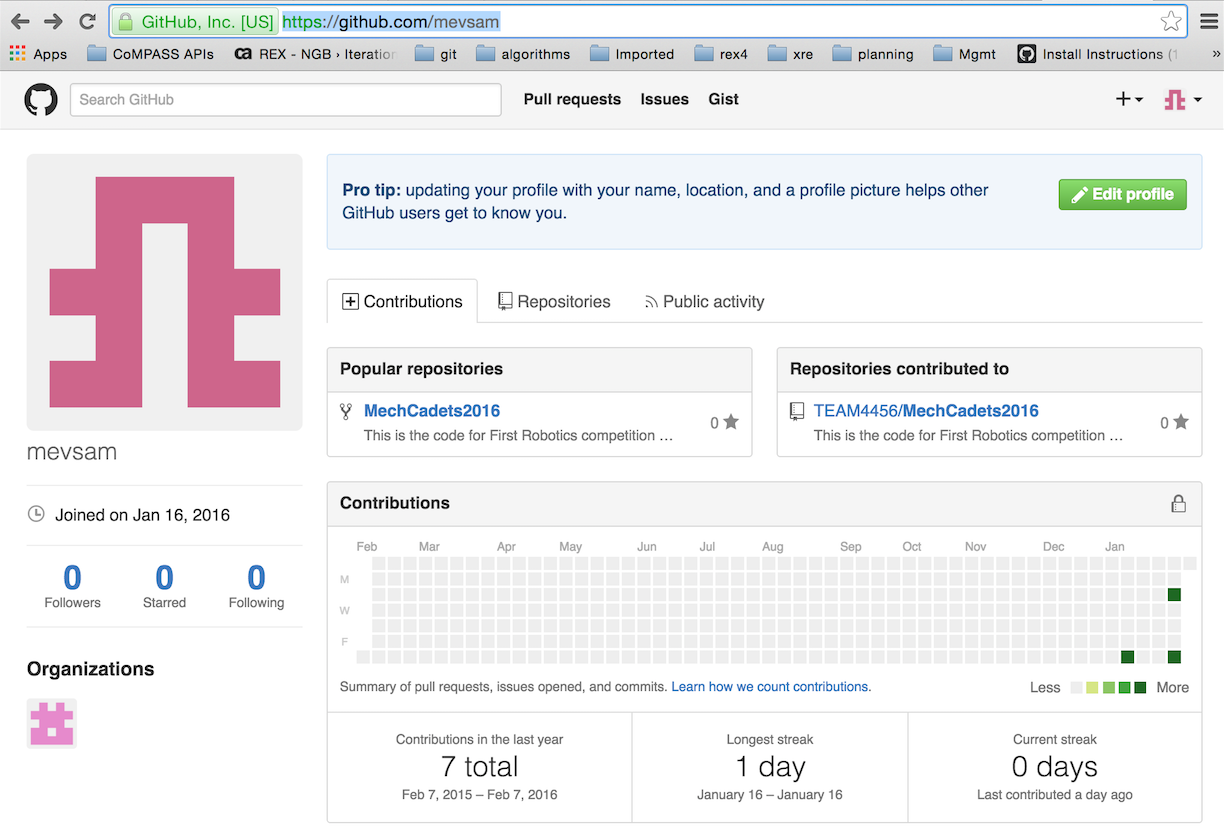


A pop up will appear and you should select “Projects from Git” and click “Next”. Select “Clone URI” and click “Next”. You should see a pop up asking for a URI.

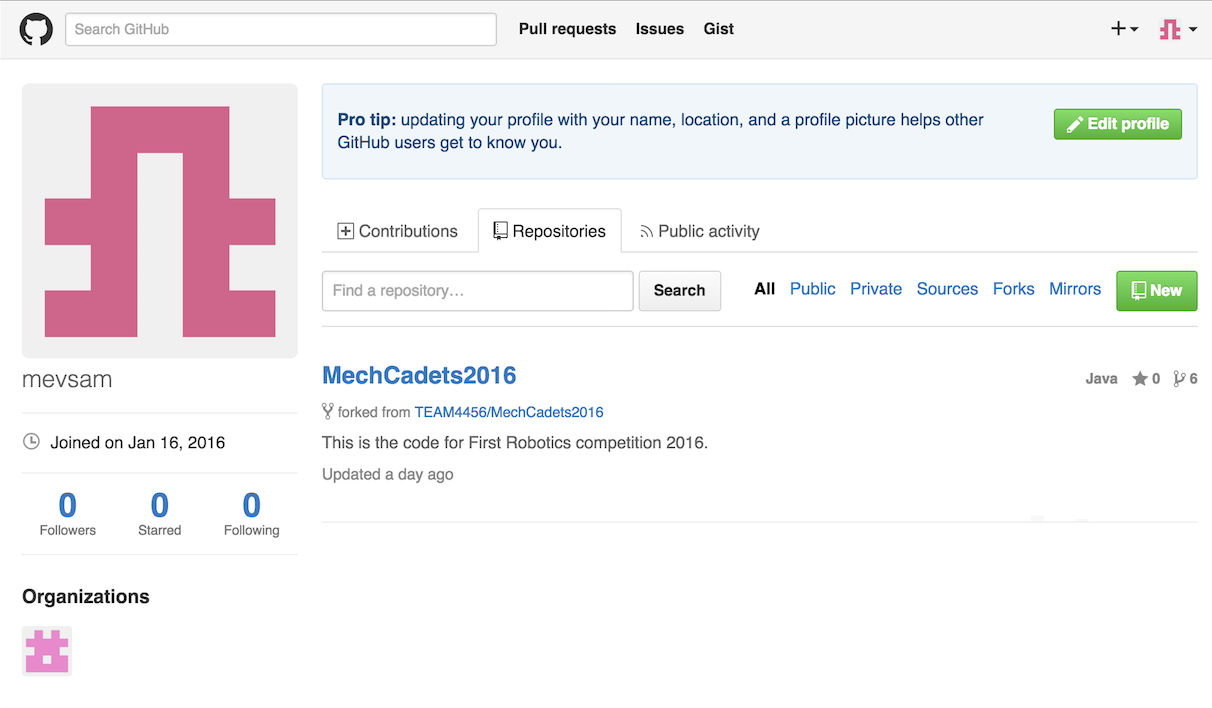
Screen Shot 2016-01-17 at 6.43.31 PM.pngScreen Shot 2016-01-17 at 6.45.37 PM.png

In this example, we will be cloning your “Fork” (that you just created) onto your local laptop/PC.

Go to your github.com dashboard. It should look similar to mine:

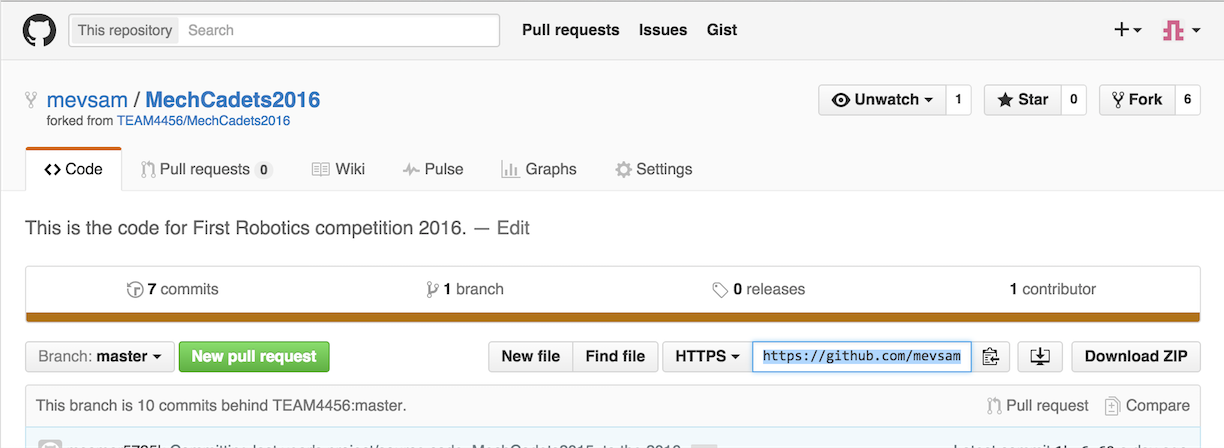


Click on the “Repositories” tab in the middle of the page. You will see the “MechStorm2016” project.

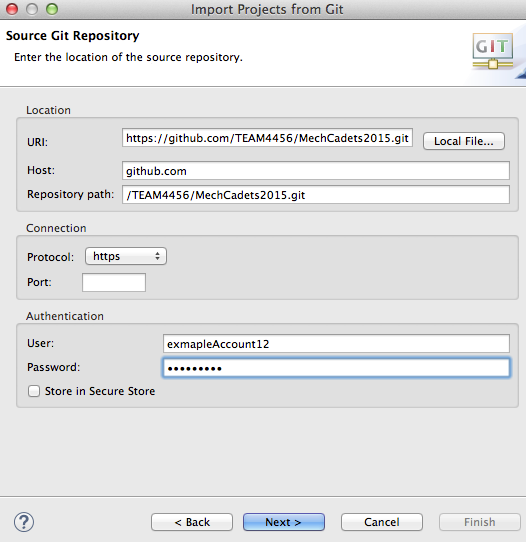


Click on the “MechStorm2016” project. This will open the page showing your fork. In my case: <https://github.com/mevsam/MechStorm2016>.

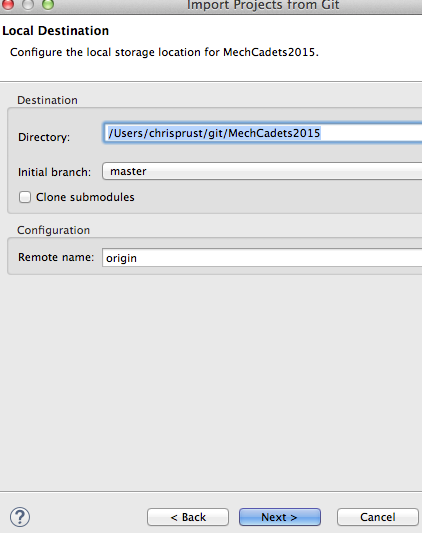
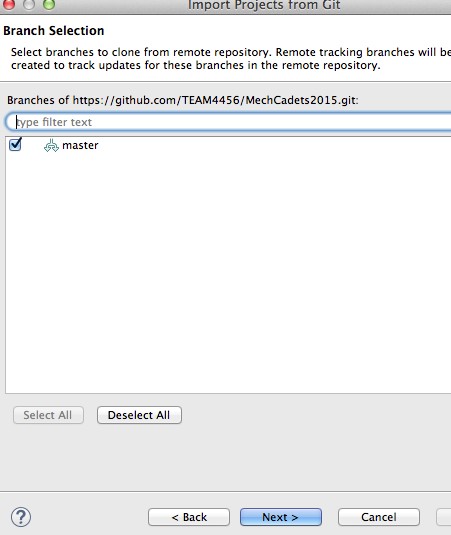
In the middle of the page, you will see the “HTTPS” box. Copy the URL onto your clipboard. Here’s a screenshot:

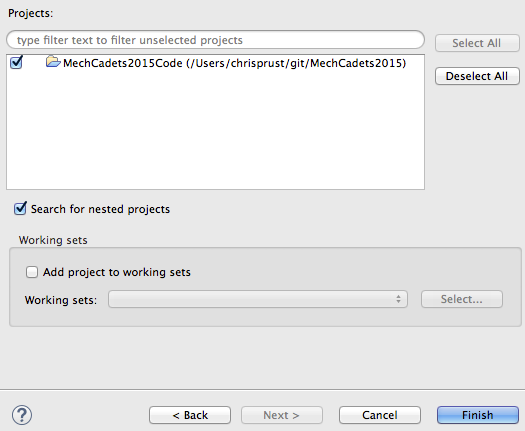
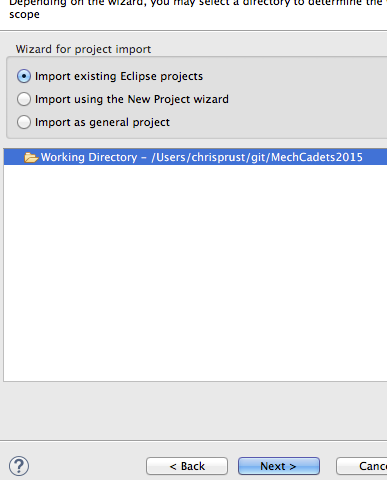


Going back to Eclipse enter the URI and enter your username and password and click “Next”.



Various screens will pop up (shown below), the default options should be fine so just click “Next” through all the screens and finally “Finish”. You should now see “MechStorm2016” in you Package Explorer.





Step 4:

Using Eclipse, make changes to the code as needed. Depending on your need, your change could be:

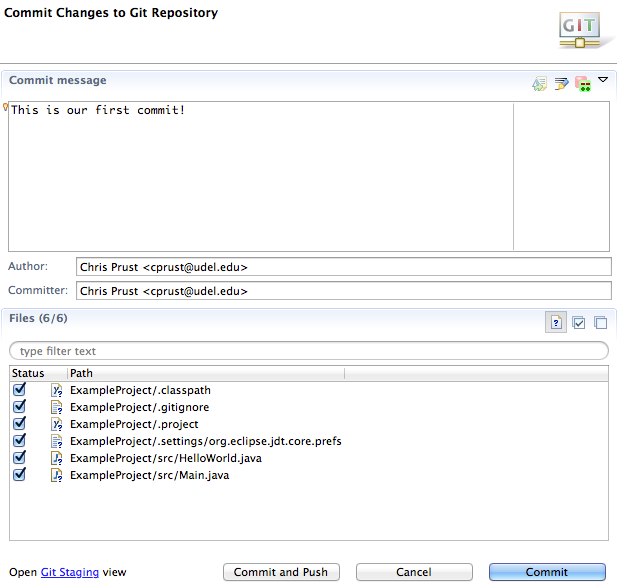
1. A change to an existing class or file
2. Adding a new file such as a Java class, configuration file, or a new “JAR” library
3. Any other change or addition to your Eclipse project

Step 5:

Now we are ready to “commit” your code changes. This means that we will save what we have to our local “clone” repository. We can click the commit button on the tool bar.

Screen Shot 2016-01-17 at 4.29.09 PM.png

A pop up will appear. You MUST fill out the commit message to document what you are doing on your project. Have all the boxes checked in the lower part of the pop up (these are all the files that you will be committing). Then click “Commit”.



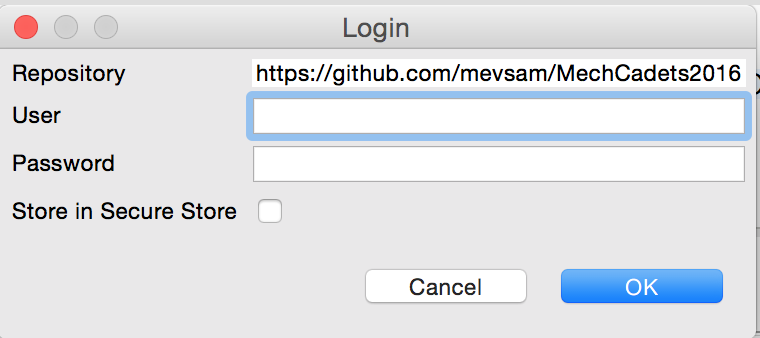
Step 7:

“Push” your commits to your “Fork” on [www.github.com](http://www.github.com) (e.g. [https://github.com/mevsam/MechStorm2016)](https://github.com/mevsam/MechStorm2016%20)).

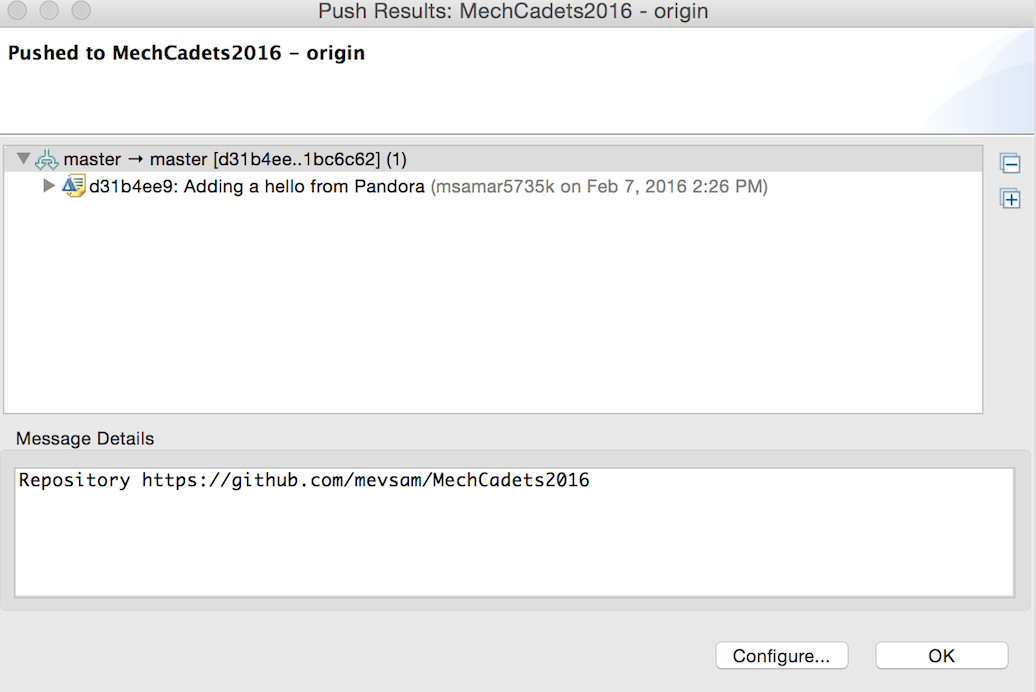
In Eclipse, click on the “Push” icon:



If this is your first “push”, you will get prompted for your github.com username and password. Enter it. Select the “Store in Secure Store” so that Eclipse saves your password for future.



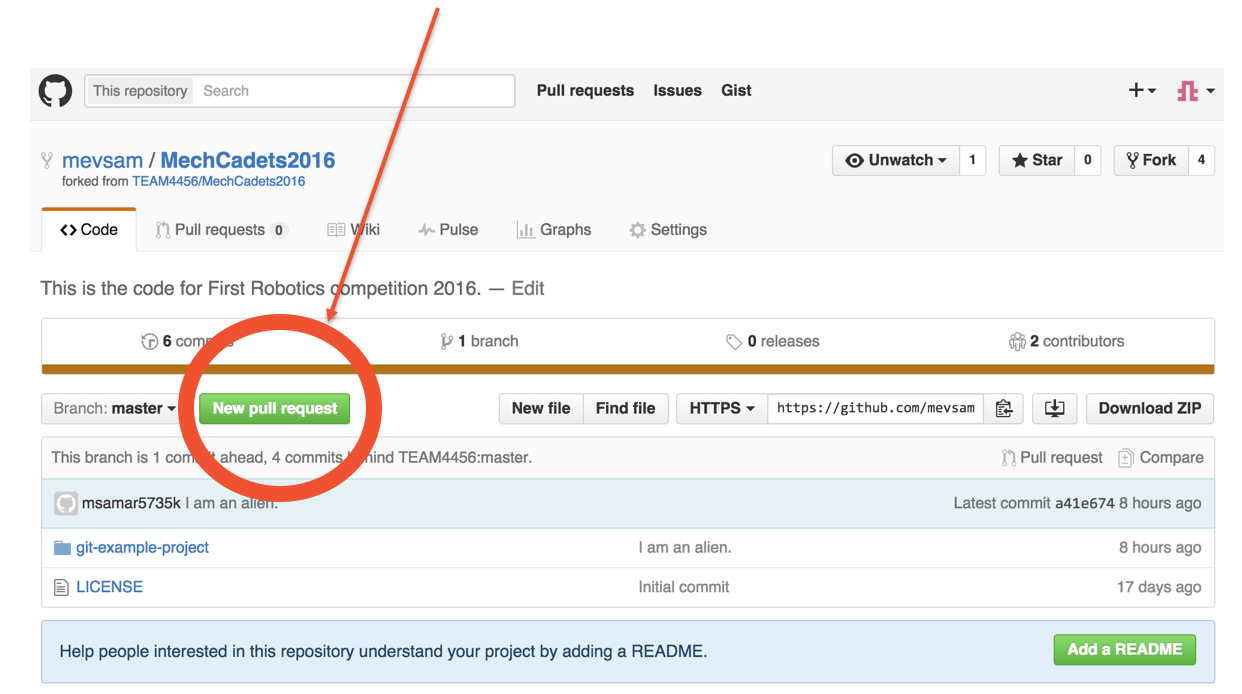
You will get a confirmation box. Click on “OK”:



Your code is now “pushed” to your “Fork” on github.com. To verify, go to your “Fork” on github.com and confirm that your new files/changes are updated. Example: <https://github.com/mevsam/MechStorm2016>.

Step 8:

Make a “pull request”: this is a request to the Admin of the “Master” repository that you have code that should be merged into the “Master” (from your “Fork”). As noted above, in our case the Master is https://github.com/TEAM4456/MechStorm2016.



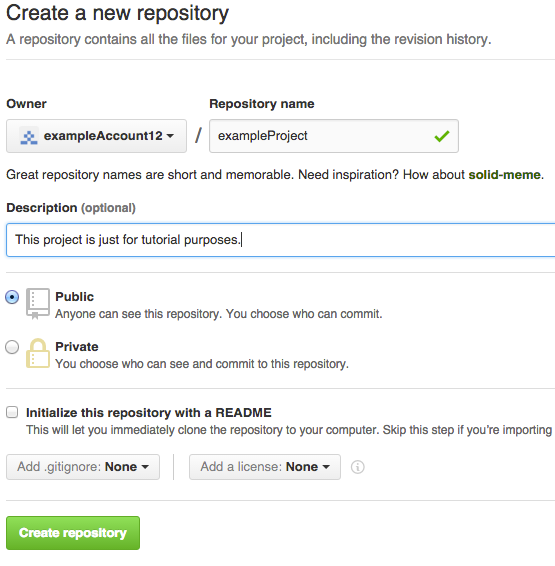
Step 9:

The Admin will review the code and merge it into Master. If there are questions/comments on the code, the Admin will respond to your pull request and ask for further information/clarification prior to merging the code to Master.

*Remember: the “Master” is the source of truth for the project and it contains all the work from everyone writing code for the project. Thus, it should always be kept pristine and in working order (at any time, one should be able to build, deploy and successfully run the code from Master).*

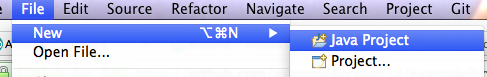
## Adding new project to Github

A times, you may want to create a brand new project in Github. To start your new project you have to first open a “New repository” (should be green button on the middle, right of the screen). It will ask you to confirm your account by going to your email. The Repository will be your project file. Chose a name, add a description if you want, and keep it public. For this tutorial I will call it “exampleProject”.

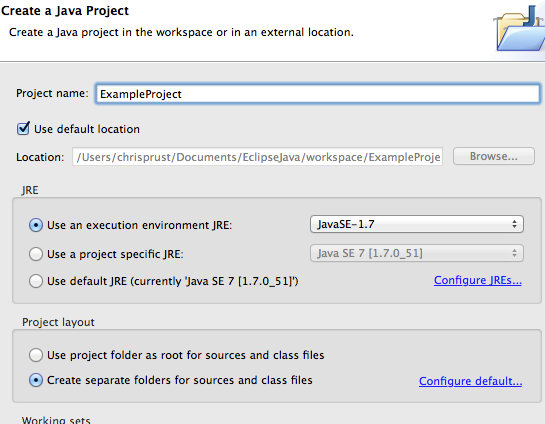


**Lets upload our first file to GitHub.**

- Open Eclipse and start a new project.

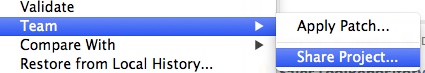


-------------------------------------------------------------

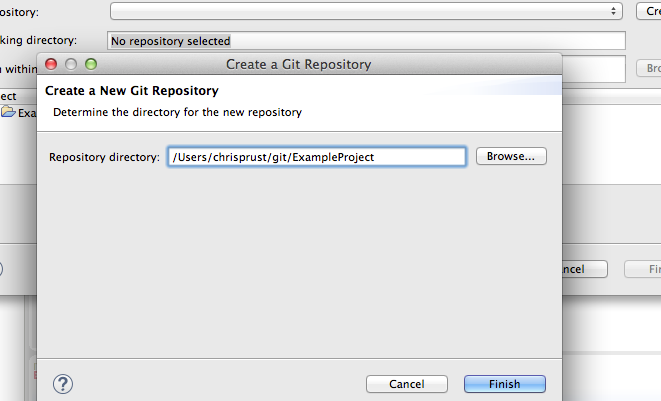


**Making the Eclipse Project into a git Project and Setting up Local Repo.**

-Now that we have a java project, we can make it into a git project. Start by right clicking on the project, go down to “Team” and then “Share Project”.



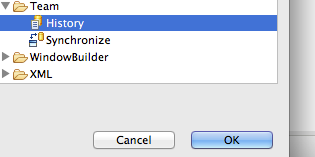
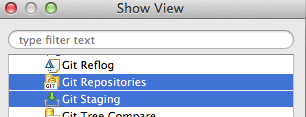
A pop up should appear. On the right side of the pop up click “Create” and this will open another pop up. This pop up tells you where your local repository will reside in your computer. It will suggest a directory and you simply need to add what the file name will be by adding a / and file name (in this example /ExampleProject):



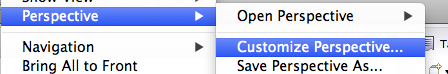
Click “Finish” and again “Finish” on the original pop up. Your local repository is now set up.

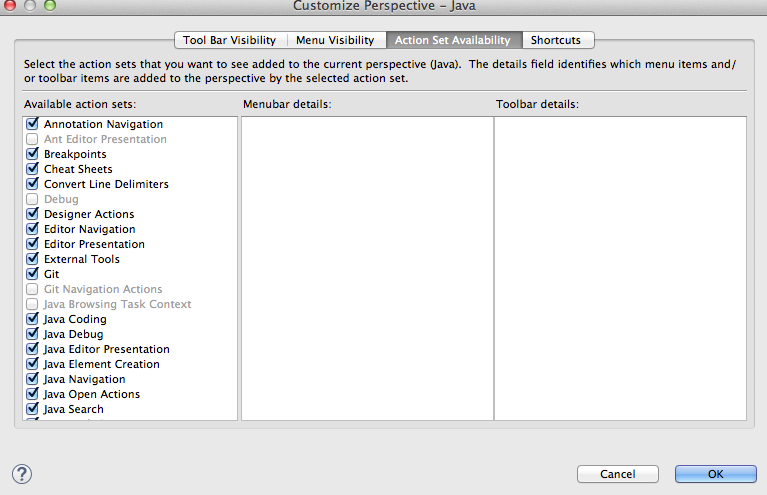
**Bring up Extra Views and git toolbar into Eclipse.**

It will be useful to bring up extra windows in Eclipse to better interact with git. Go to “Window” -> “Show Views” -> “Other”->look up “Git” and highlight “Git Repositories” and “Git Staging” and then click “OK”. Do the same thing again, but this time go to “Team” and highlight “History” and click “OK”.

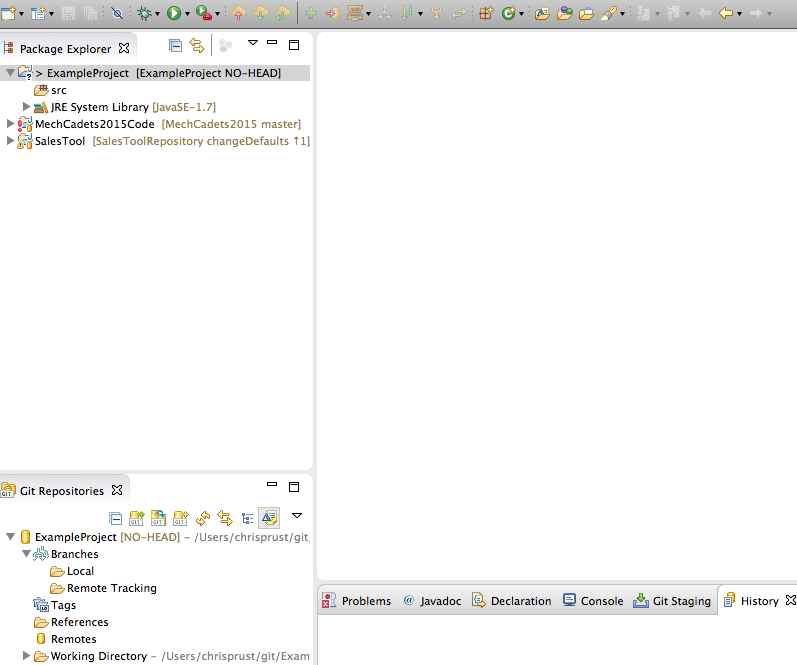


To bring up the git toolbar, go to “Window” -> “Perspective”->”Customize Perspective” -> a pop up will appear, go to “Action Set Availability” and make sure that “Git” is checked off, then click “OK”.



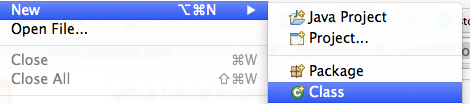


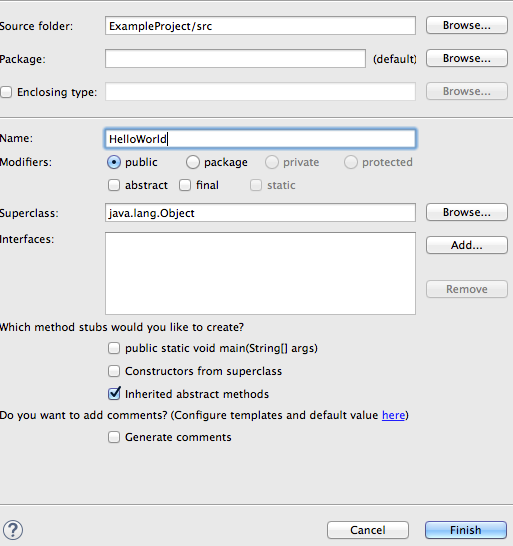
You should notice new windows have appeared in you Eclipse window and that your toolbar has changed slightly.



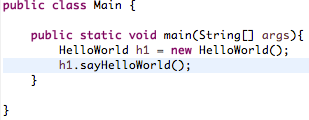
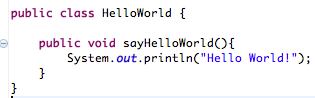
**Make a simple Java class and make our first commit**

We will make two java classes called “HelloWorld” and “Main”.



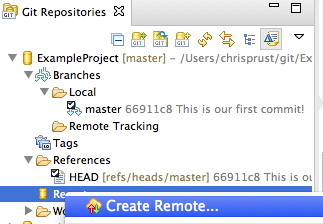


They will look like this:

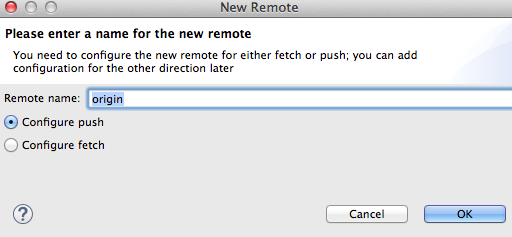


**Push to Git Server**

Now that we have learned to commit our code to our local repo we will push the code from our local repo to our online (server) repo on GitHub. Right click on “Remotes” (under the “Git Repositories” window) -> “Create Remote”



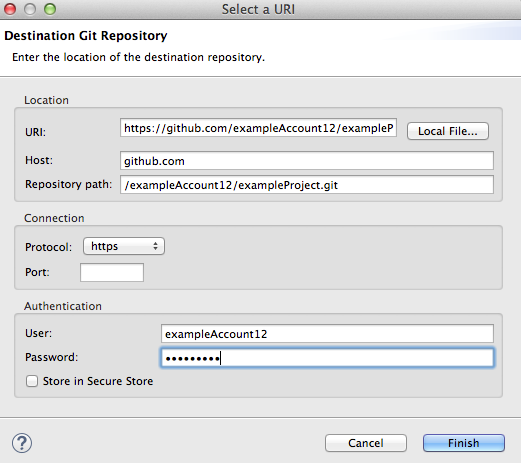
A pop up should appear. The “Remote name” should default to “origin”. Keep this; it is the name of our online repo. Click “OK”. A new pop up should appear.



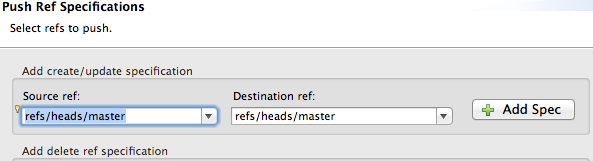
Go to your online GitHub account and look for the HTTPS and copy it. For this example it is (but it will be different for you!): https://github.com/exampleAccount12/exampleProject.git

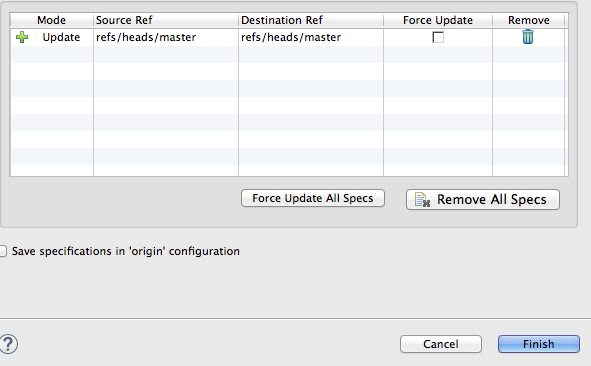


Going back to Eclipse, under the new pop up click “Change” and paste the HTTPS that you just got (Eclipse should auto fill some of the entries below). Then enter your username and password at the bottom of the pop up and click “Finish”.

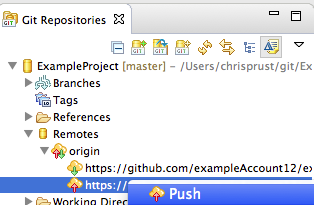


You will return to your original pop up. Click “Advanced” and then a new pop up should appear. Change the “Source ref” and “Destination ref” by pulling down the menu and selecting the “refs/heads/master” option for both and then click “Add Spec”. The table at the bottom of the pop up should have an entry now (see picture below). Now click “Finish”. You should be back at the original pop up.

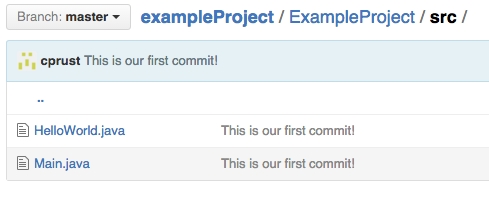
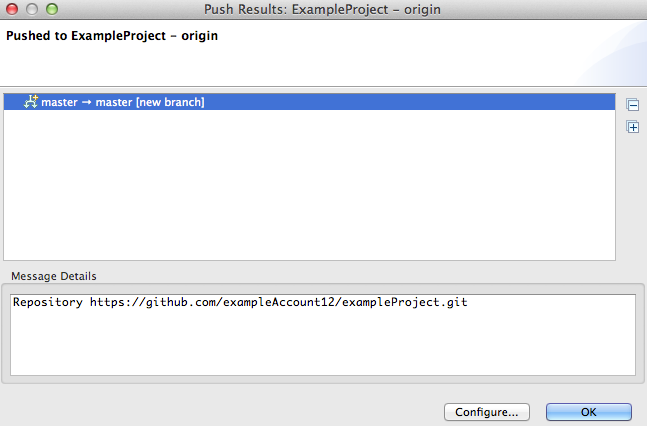




At this point we have the option of saving and pushing, but we will do this in two steps (generally considered good practice). Click “save” and the pop up will disappear. Look under the “Git Repositories” window under the project -> “Remotes”->”origin” -> and right click the red arrow pointing up and click “Push”



It will prompt you for your account information again and bring you to a pop up which looks like the one below. Click “OK”. Now if you look under your GitHub account you should notice that the two code files should be in your account.



Notice that if I change “Hello World!” to “Hello Universe!”, commit the change, and then push it up, you can see the files in the GitHub account change and you can see in the history of the project both of our commits.

