Setting Up Your CSE 332 Environment

Gitlab and Submission

We will be using `gitlab.cs.washington.edu` to submit homeworks and give feedback. `gitlab` is a version of `github` that is local to the CSE Department. As such, you will need to learn basic `git` to be able to work on and submit your homework.

Downloading Eclipse

The first thing you should do is download **the newest version** of Eclipse. This is important, because we will be using plugins that were not included in old versions of Eclipse. To download Eclipse go to

`https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/marsr`

Generating and Installing a Private Key

The first time you clone from gitlab, you will need to create a *private key*. To do this, follow these steps:

(1) Open Eclipse Preferences and type "ssh" into the box.

(2) Click the "Key Management" tab and click "Generate RSA Key".

(3) Type your `UWNetID` into the "Comment" box.

(4) Then, click "Save Private Key" and choose somewhere safe.

(5) Copy the text starting with "ssh-rsa" in the box and go to

`https://gitlab.cs.washington.edu/profile/keys`.

(6) Click "Add SSH Key", choose a name for the key, and paste the text you copied into the box.

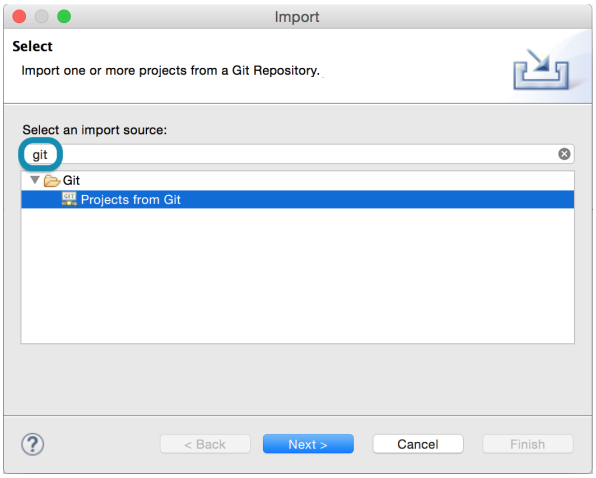
(7) You have now installed an RSA public key, and you're ready to use `git`!

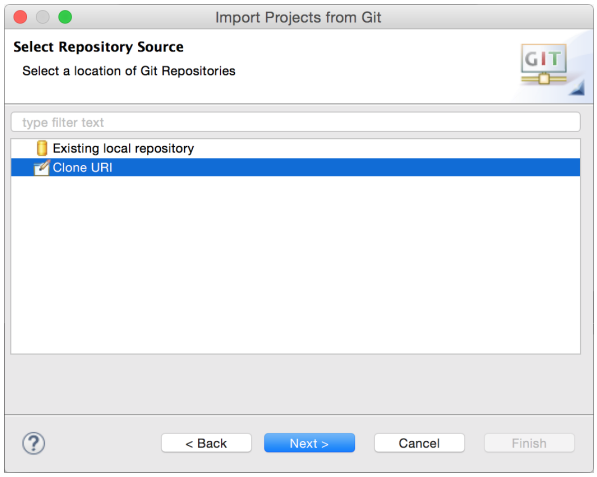
Creating The Eclipse Project

Eclipse uses a concept of *projects* to organize your files. Each project in the course will have its own project. We will create the project by using the template `gitlab` repository. To do this, follow these steps:

(1) Go to `File > Import`

(2) Type "`git`" into the box; continue by choosing "Projects from Git" and "Clone URI":



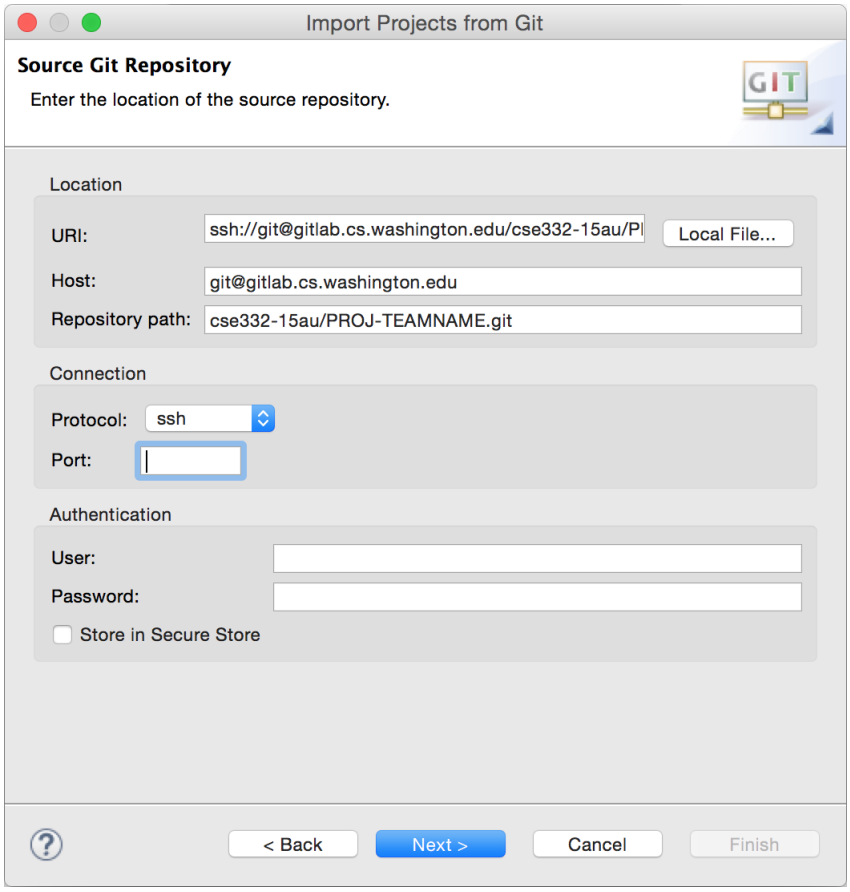


(3) The next dialog will ask you to enter the source of your Git Repository. Ignore the URI field, and fill in the following fields:

- **Host:** `git@gitlab.cs.washington.edu`
- **Repository Path:** `cse332-`**15au/p1-bulbasaur**`.git`
- **Protocol:** `ssh`

A **bold** phrase means you should substitute it with your equivalent.

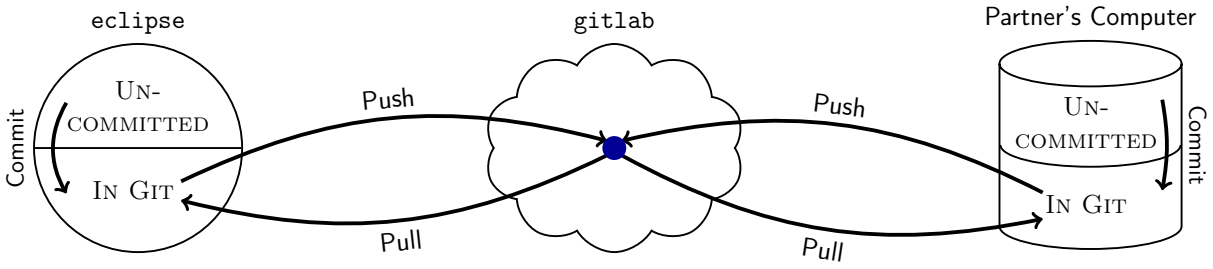
Do *not forget the* `.git` at the end!



(4) Finish out the wizard, and you should have a new project in Eclipse!

Committing, Pushing, and Pulling

`git` is the *version control system* (VCS) underlying `gitlab`. Most real projects are kept in a VCS to avoid losing data and the ability to go back to older versions of code. Another major reason VCS's are important is that they allow you to effectively work together with other people. They allow you to combine (called "merge") several different versions of your codebase together.



As shown in the diagram, there are several major actions you can do with respect to your `git` repository:

- **Commit:** A "commit" is a set of changes that go together. By "committing" a file, you are asking `git` to "mark" that it has changed. `git` requires you give a message indicating the high-level idea behind the changes. An example might be "Adds error handling for the empty queue case in dequeue".
- **Push:** A "push" sends your commits to another version of the repository (in our case, this will almost always be `gitlab`). If you do not push your commits, nobody else can see them!
- **Pull:** A "pull" gets non-local commits and updates your version of the repository with them. If you and someone else both edited a file, the histories of the file diverged, and `git` asks you to explain how to merge the changes together. (This is called resolving a "merge conflict".)

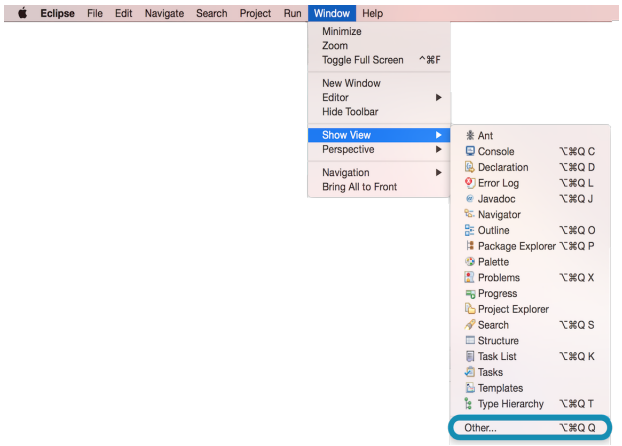
Using Git in Eclipse

Eclipse provides GUIs for all of the `git` operations. We now explain how to handle a `git` workflow.

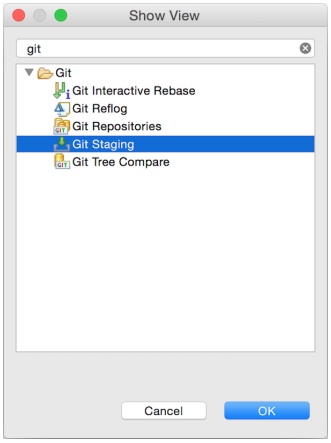
Open Git Staging View

Your main tool to use `git` in `eclipse` is called the "staging view". This view allows you to see changed files, make commits, and write commit messages. To open the Staging View, follow these steps:

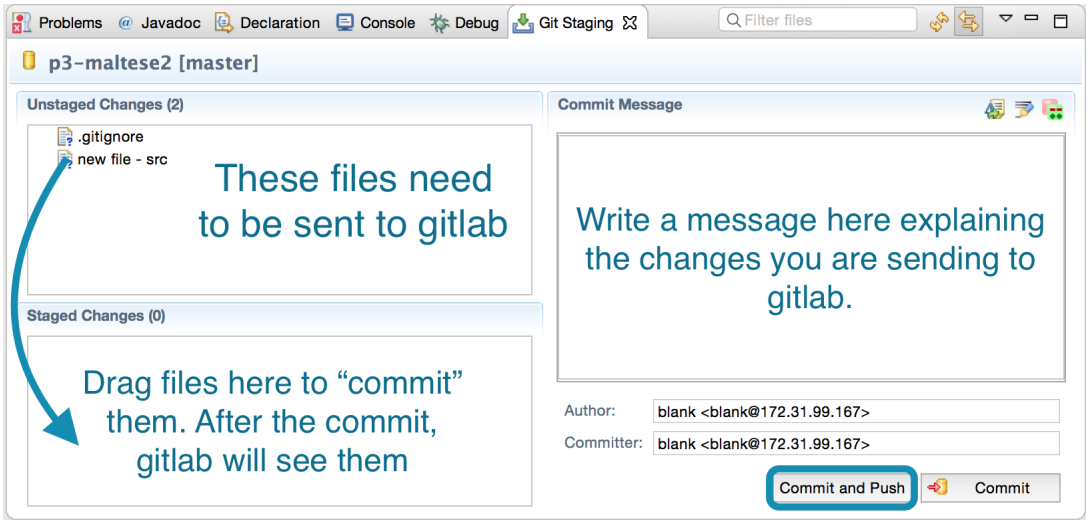
(1) Go to `Window > Show View > Other`



(2) Type "`git`"; choose "Git Staging"



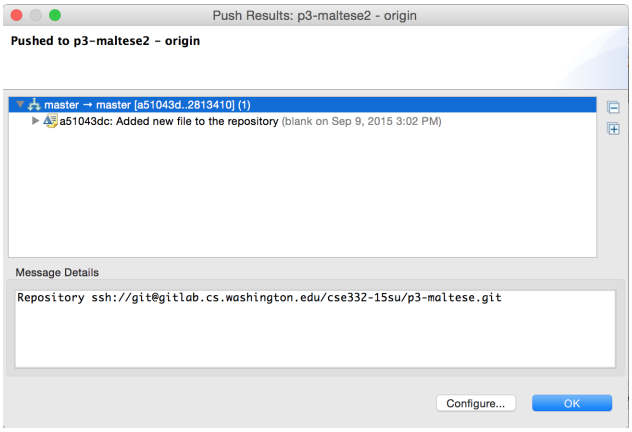
How Staging/Committing Works



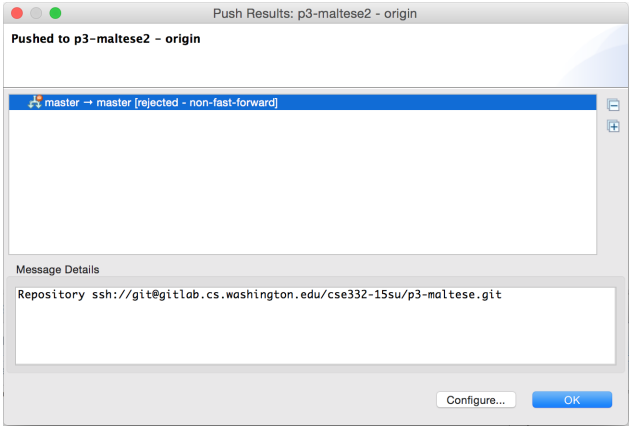
After Pushing

After you "Commit and Push", you will get one of several dialog messages:

Success!



Rejected!

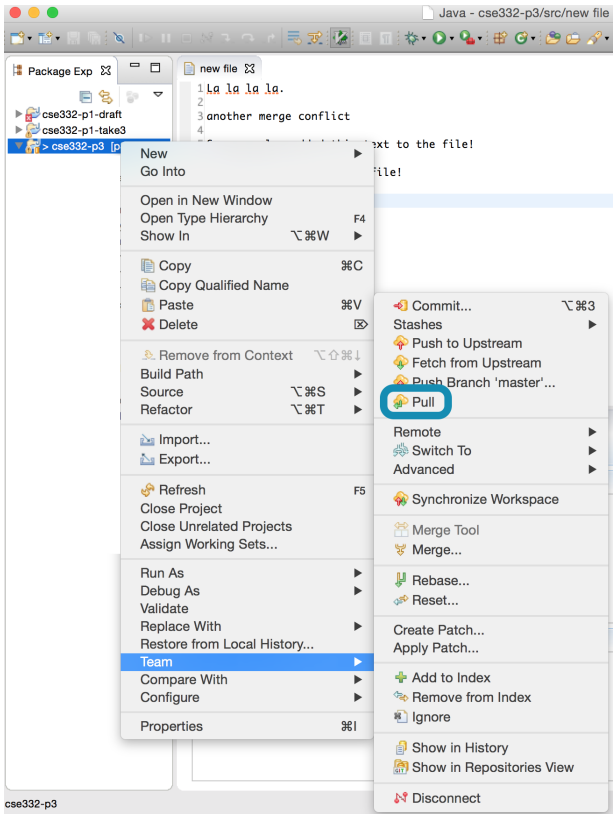


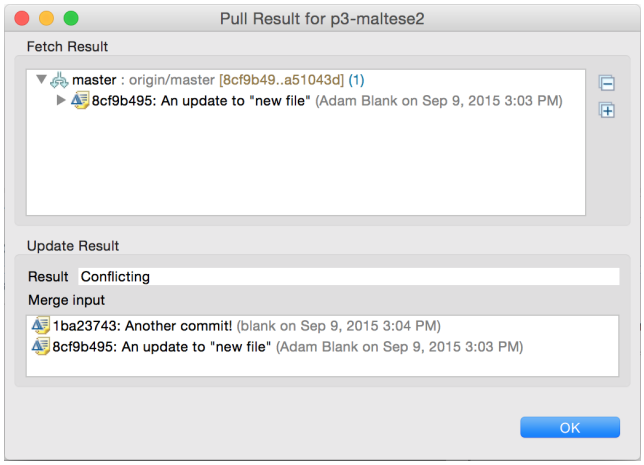
If you get a "success" message, then your code has been pushed, and you're good to go. If you get a "rejected" message, it means your partner pushed code; so, you will need to *pull* before pushing will work. Note that when you pull, you *might* have a merge conflict which you will have to fix before pushing.

Pulling

There are two reasons to pull: (1) you want to get the changes your partner made, and (2) you want to push your changes, but they were rejected because of a conflict.

To pull in Eclipse, click the following menu option:



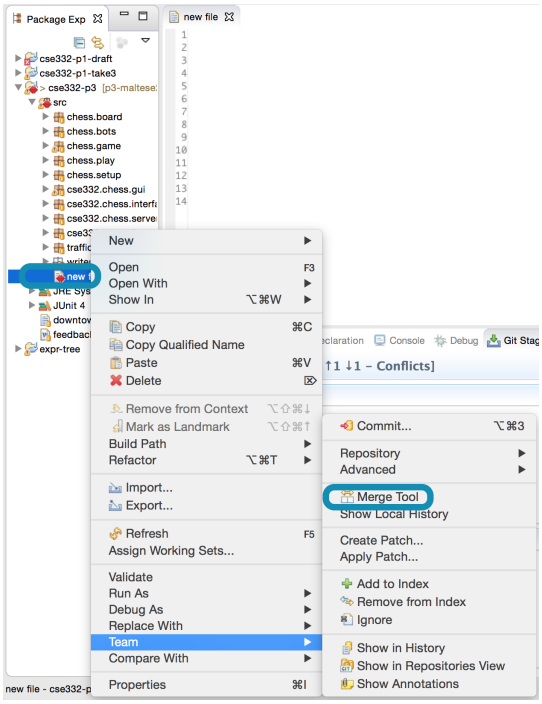


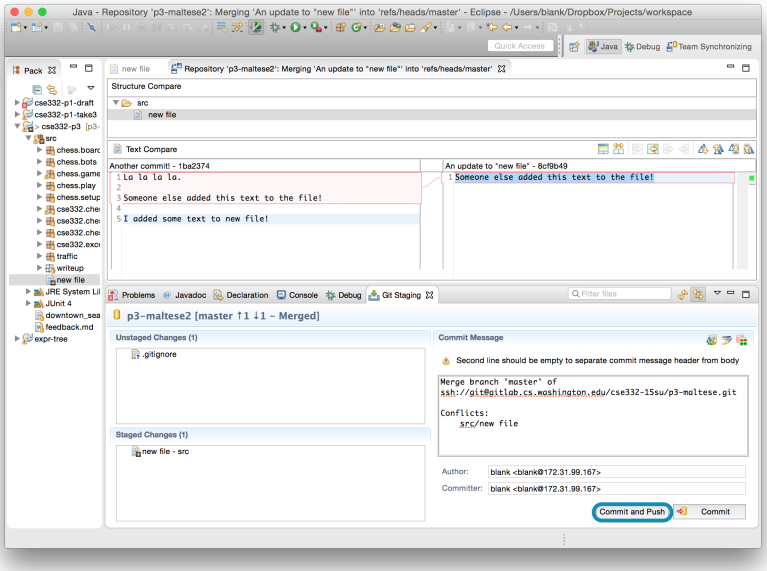
The result will either indicate that you pulled cleanly or that there is a conflict.

The best way to tell if you currently have a conflict is to look for the red diamond icon in the list of files.

Merging a Conflict

If you have any items that have a conflict (red diamond icon), you can fix them using the *merge tool*:





The merge tool allows you to see your changes (on the left) and other peoples' changes (on the right). Your job is to make the file on the left the result you actually want. Once you've done this, drag it to the "staged changes" panel like usual. You will notice that there is an auto-filled Commit Message about a "merge". Go ahead and "Commit and Push". Now, you've pushed your commits and other people can see them!

Submitting Your Final Version

In some courses, you are asked to "tag" your final commit. We will not ask you to do this. The last commit you make within the deadline will be the one we will grade.