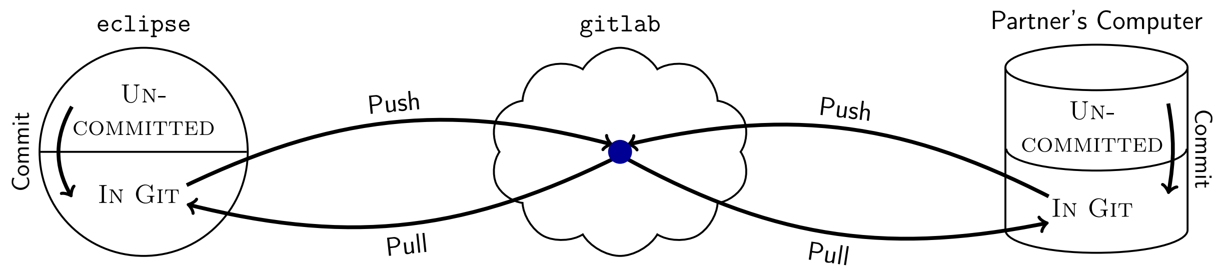
Instructions on using Git in Eclipse

Written by Adam Blank for CSE 332, applicable for CSE 373

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".)

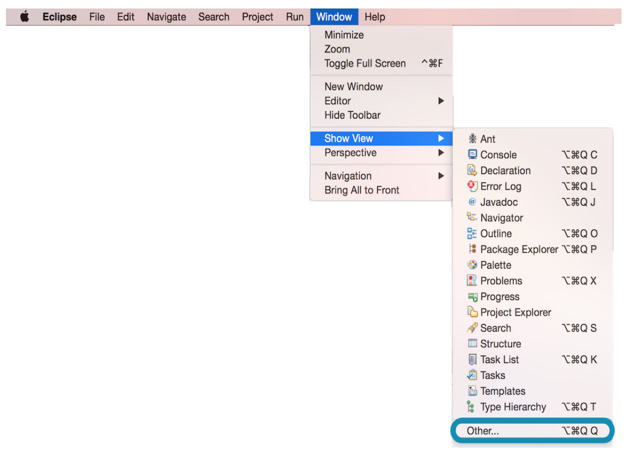
Using Git in Eclipse

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

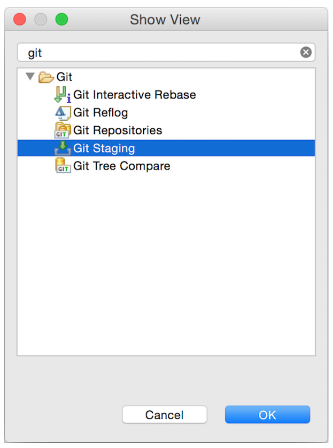
Open Git Staging View

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:

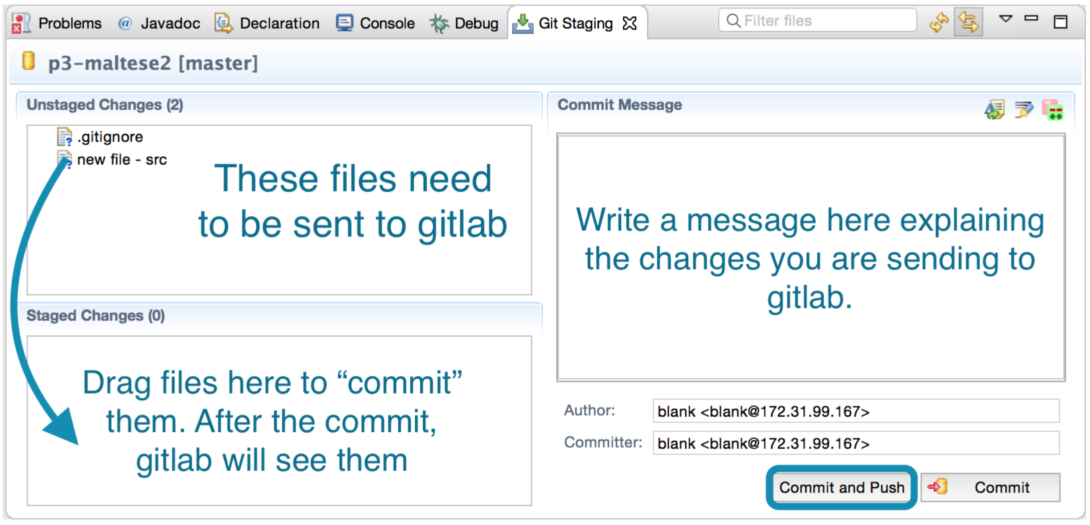
(1) Go to `Window > Show View > Other`



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



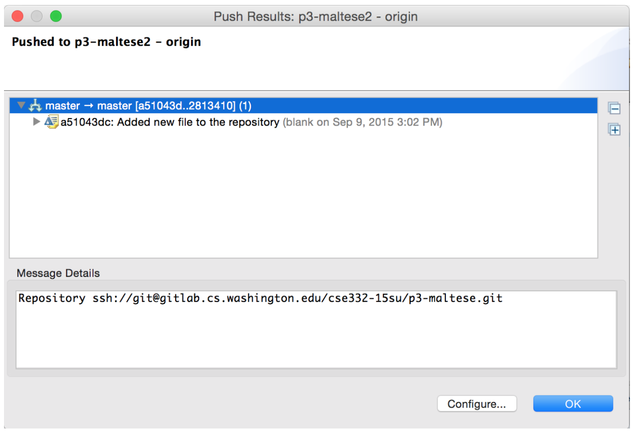
How Staging/Committing Works


Problems
Javadoc
Declaration
Console
Debug
Git Staging
Filter files
p3-maltese2 [master]
Unstaged Changes (2)
.gitignore
new file - src
These files need to be sent to gitlab
Staged Changes (0)
Drag files here to "commit" them. After the commit, gitlab will see them
Commit Message
Write a message here explaining the changes you are sending to gitlab.
Author:
blank <blank@172.31.99.167>
Committer:
blank <blank@172.31.99.167>
Commit and Push
Commit

After Pushing

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

Success!



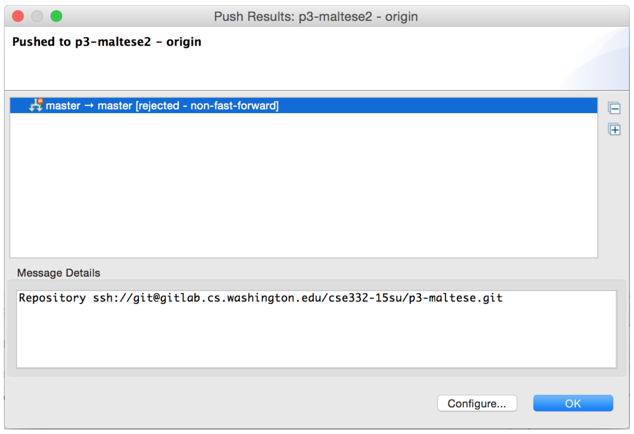
Pushed to p3-maltese2 – origin

master → master [a51043d..2813410] (1)
a51043dc: Added new file to the repository (blank on Sep 9, 2015 3:02 PM)

Message Details

```
Repository ssh://git@gitlab.cs.washington.edu/cse332-15su/p3-maltese.git
```

Rejected!



Pushed to p3-maltese2 – origin

master → master [rejected - non-fast-forward]

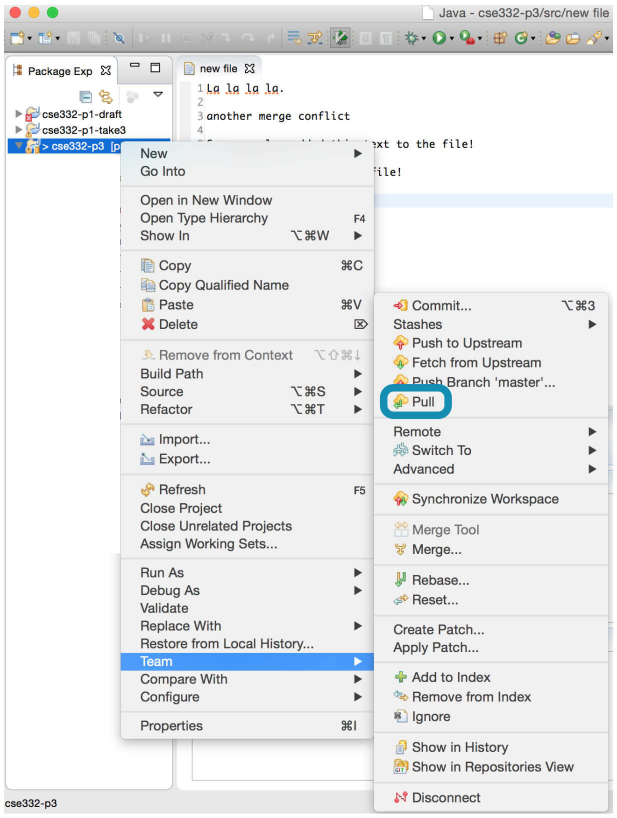
Message Details

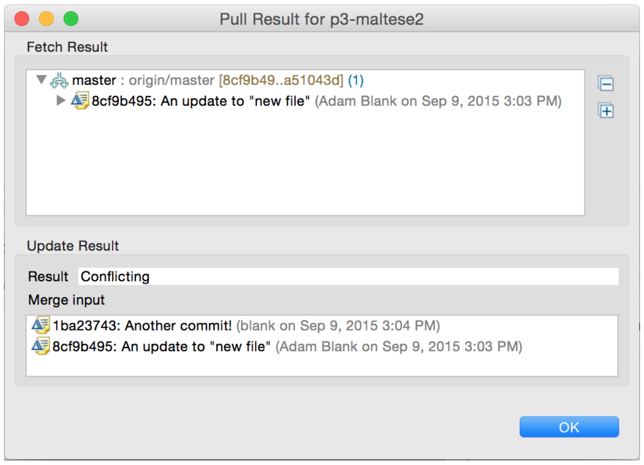
```
Repository ssh://git@gitlab.cs.washington.edu/cse332-15su/p3-maltese.git
```

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:





Pull Result for p3-maltese2

Fetch Result

master : origin/master [8cf9b49..a51043d] (1)
8cf9b495: An update to "new file" (Adam Blank on Sep 9, 2015 3:03 PM)

Update Result

Result Conflicting

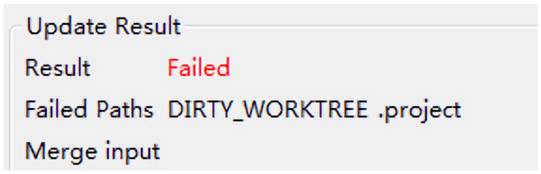
Merge input

1ba23743: Another commit! (blank on Sep 9, 2015 3:04 PM)
8cf9b495: An update to "new file" (Adam Blank on Sep 9, 2015 3:03 PM)

OK

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.

If you see a DIRTY_WORKTREE error when attempting to pull, then you have uncommitted local changes which conflict with changes on GitLab.



Update Result

Result Failed

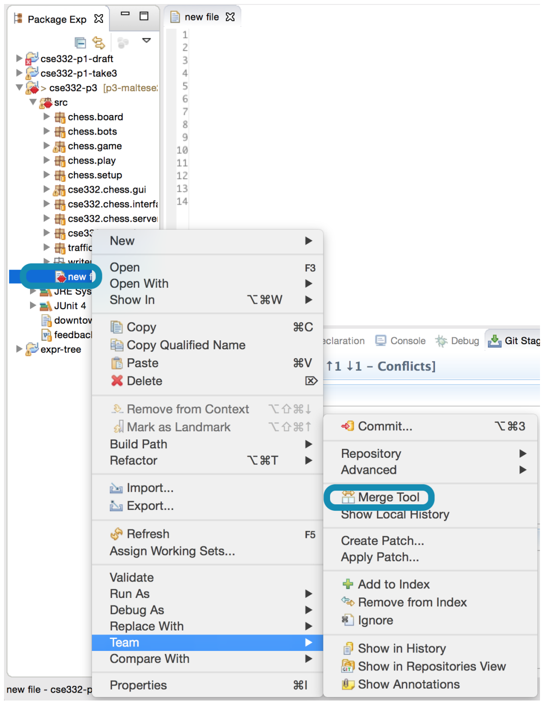
Failed Paths DIRTY_WORKTREE .project

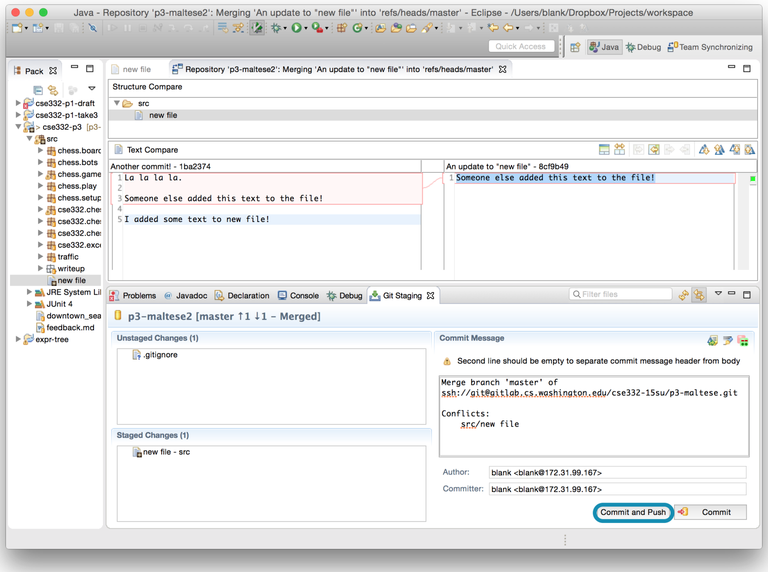
Merge input

To resolve this error, you should first **commit (but do not push)** all local changes, then **pull**, and *then* **merge any conflicts** using the procedure described on the following page.

Merging a Conflict

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




Java - Repository 'p3-maltese2': Merging 'An update to "new file"' into 'refs/heads/master' - Eclipse - /Users/blank/Dropbox/Projects/workspace
Quick Access
Java
Debug
Team Synchronizing
Pack
new file
Repository 'p3-maltese2': Merging 'An update to "new file"' into 'refs/heads/master'
Structure Compare
src
new file
Text Compare
Problems Javadoc Declaration Console Debug Git Staging
Filter files
p3-maltese2 [master ↑1 ↓1 – Merged]
Unstaged Changes (1)
.gitignore
Staged Changes (1)
new file - src
Commit Message
Commit and Push
Commit

Another commit! - 1ba2374

```
La la la la.

Someone else added this text to the file!

I added some text to new file!
```

An update to "new file" - 8cf9b49

```
Someone else added this text to the file!
```

Second line should be empty to separate commit message header from body

```
Merge branch 'master' of
ssh://git@gitlab.cs.washington.edu/cse332-15su/p3-maltese.git

Conflicts:
    src/new file
```

Author: blank <blank@172.31.99.167>

Committer: blank <blank@172.31.99.167>

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.