Micah's TODO list intro to git tutorial:

**Part 1 – The Basics**

1) In a new folder, create a new git repository. Once the repository is created, observe what the 'git status' command tells you.

*mkdir test_proj*
*cd test_proj*
*git init*
*git status*

2) Create a file named TODO with a list of ~3 made up tasks (not school assignments). After creating the file, observe what the 'git status' command tells you. Add that file to the index (aka staging area). After adding the file, observe what the 'git status' command tells you. Do you see where git is telling you what command to use to remove the file from the index?

*vim TODO*
*git status*
*git add TODO*
*git status*

3) Now that the file is in the index, commit it to the repository. Afterwards, observe what the 'git status' command tells you.

*git commit*
*git status*

4) Look at and observe the output of 'git log'.

**Part 2 – Branching**

1) Create a honey_do branch. In this branch add another 2 things that your "honey" has asked you to do and commit the changes to the repository.

*git branch honey_do*
*git checkout honey_do*
*vim TODO*
*git add TODO*
*git commit*

2) Look at and observe the output of 'git log' and 'git lola'.

3) Add another 2 things to the TODO list inside the honey_do branch and commit changes.

*vim TODO*
*git add TODO*

*git commit*

4) Look at and observe the output of 'git log' and 'git lola'. Take a few minutes to tinker with the git log command with different combinations of the flags used in git lola (git lola = git log --graph --decorate --pretty=oneline --abbrev-commit –all).

5) Have git show you the difference between your TODO file in the master branch and your TODO file in the honey_do branch.

*git diff master honey_do*

**Part 3 – Merging**

1) On the master branch (which should point to the very first commit), add a date to the top of the file and commit the change.

*git checkout master*
*vim TODO*
*git add TODO*
*git commit*

2) Look at the output of 'git log' and 'git lola'. You now have divergent branches.

3) Use 'git merge' to bring in the changes from honey_do to master.

*git merge honey_do*

4) Look at and observe the output of 'git lola'.

5) Look at the TODO file. Is it what you expect it should be after merging the two branches together?

6) Use 'git reset --hard HEAD^1' to move the master branch back to the previous commit. (Afterwards, notice what happens to the special merge commit that was made. Why do you think this happened?)

*(The merge commit that was automatically made when the merge command was given is deleted because the commit is no longer a part of the history of any branches.)*

7) Use 'git merge --squash' to bring in the changes from honey_do into master. Remove 2 of the 4 items from your "honey" before committig. Also, after changing TODO but before adding it, look at the output of 'git status'.

*git merge --squash honey_do*
*vim TODO*
*git status*
*git add TODO*
*git commit*

8) Look at and observe the output of 'git lola'. Do you see how 'git merge' and 'git merge --squash' are

different?

**Part 4 – Git over the network**

(Note: For simplicity we will practice with files on our own hard drives rather than over a network, but the commands will be the same when you do use a server)

1) Initialize a new git repository (not inside of the one we've already made).

*cd ..*
*mkdir test_proj_2*
*cd test_proj_2*
*git init*

2) Add a remote to this new repository named origin with a url that references our first repository. HINT: At the top directory of any repo is a .git directory, the url will be the path to this directory. Something like /home/micah/test_repo/.git

*git remote add origin <path to .git directory>*

3) Look at what 'git remote -v' tells us.

4) Retrieve data from our first repository.

*git fetch origin*

5) Look at and observe the output of 'git lola'.

6) Checkout the master branch from the origin remote. HINT: When dealing with branches on a remote, the remote name is part of the branch name (origin/<branch_name>)

*git checkout origin/master*

7) Create a new local branch, based on origin/master, called assignments. Checkout that new branch.

*git branch assignments*
*git checkout assignments*

8) In this new branch, add a new file called SCHOOL_TODO. Inside this file, start creating a list of homework items to complete (I don't care how long this list is). Once the file is saved, commit it to your assignments branch.

*vim SCHOOL_TODO*
*git add SCHOOL_TODO*
*git commit SCHOOL_TODO*

9) Look at and observe the output of 'git lola'.

10) Put the assignments branch on the remote repository (the original repository that we made).

*git push origin assignments*

11) Go back to the original repository that we developed in parts 1 through 3. Run 'git lola' to confirm that the assignments branch is present. Now merge the assignments branch into your master branch (you can use merge with or without the --squash flag, your choice).

*(navigate back to the original repository)*
*git lola*
*git checkout master*
*git merge assignments*

12) Look at the contents of your project. Observe the output of the 'git lola' command. If you used the git merge command without the --squash option, is there something different about this merge than the first time?

*(This time, there is no special git commit created. Because the assignment branch was only one commit directly ahead of the master branch, the merge operation performs a "fast-forward" and moves the master branch up to the assignments branch rather than creating a special commmit with two parent commits)*