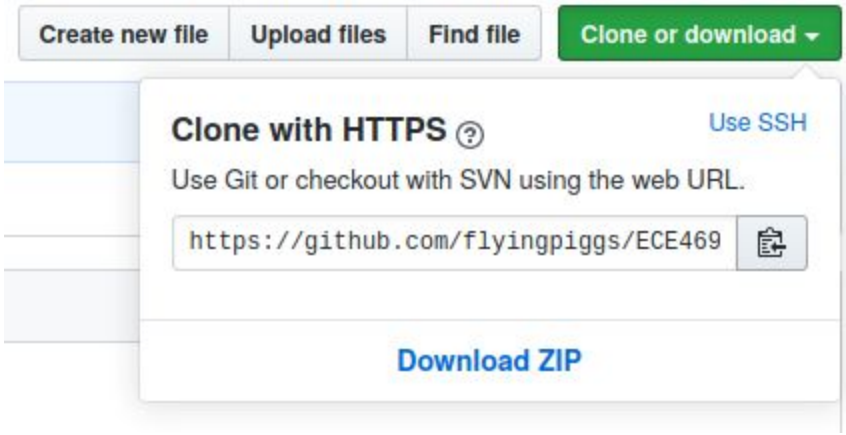
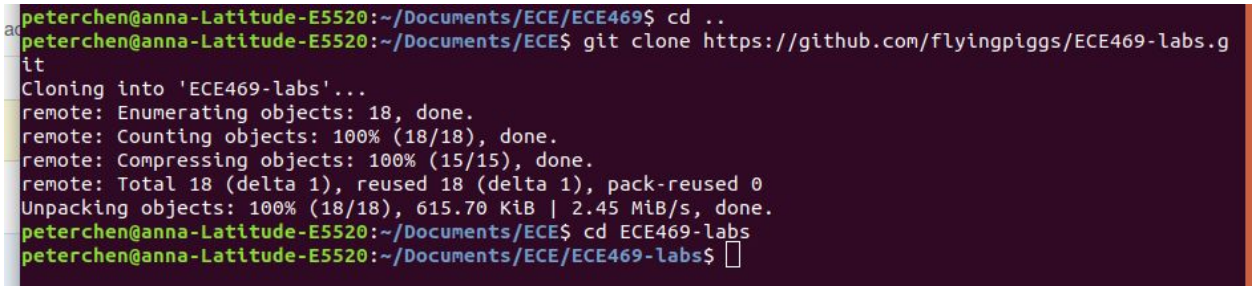
Once again, feel free to use the Git GUI instead of the Git CLI, but since I'm not familiar with the Git GUI, you'll have to figure out the corresponding directions yourself. The keywords/commands are all the same, but how you execute them is different.

I didn't proofread this so let me know if anything is confusing.

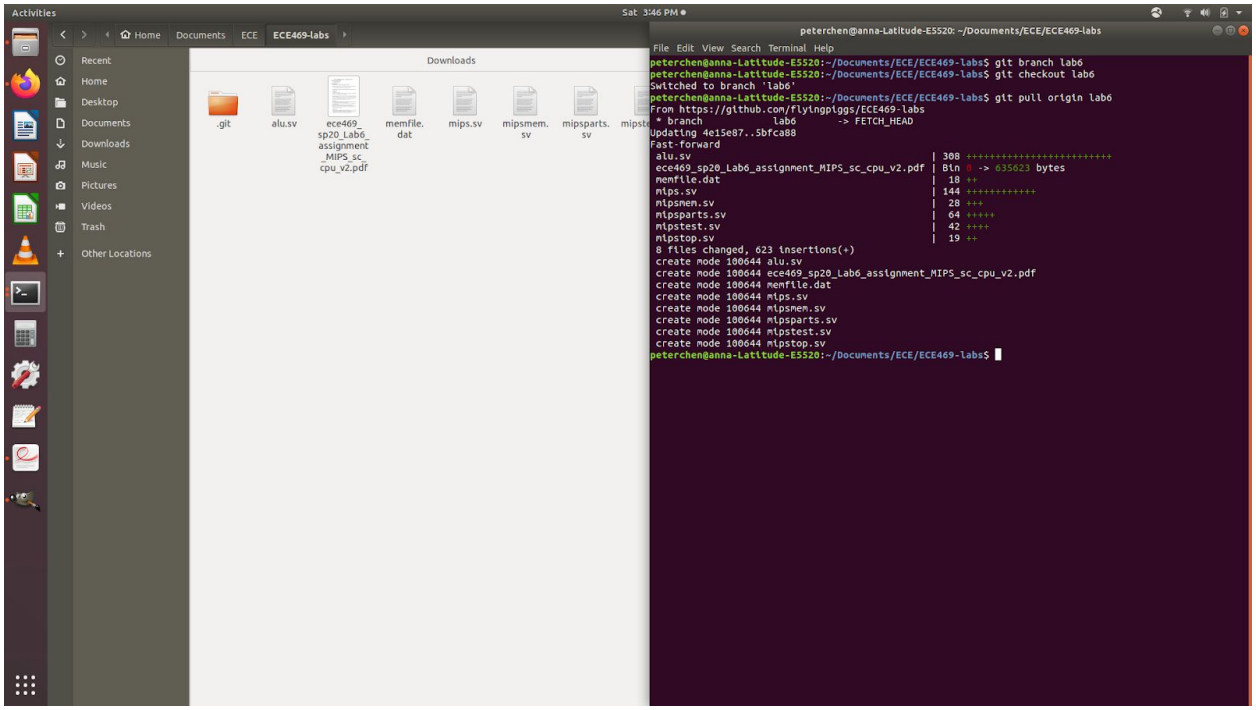


1. Use this icon to get the https link to our Github repo. It's actually just the same as the Github repo's link, but this button gives you another button that will directly copy the link for you (it's the clipboard icon).



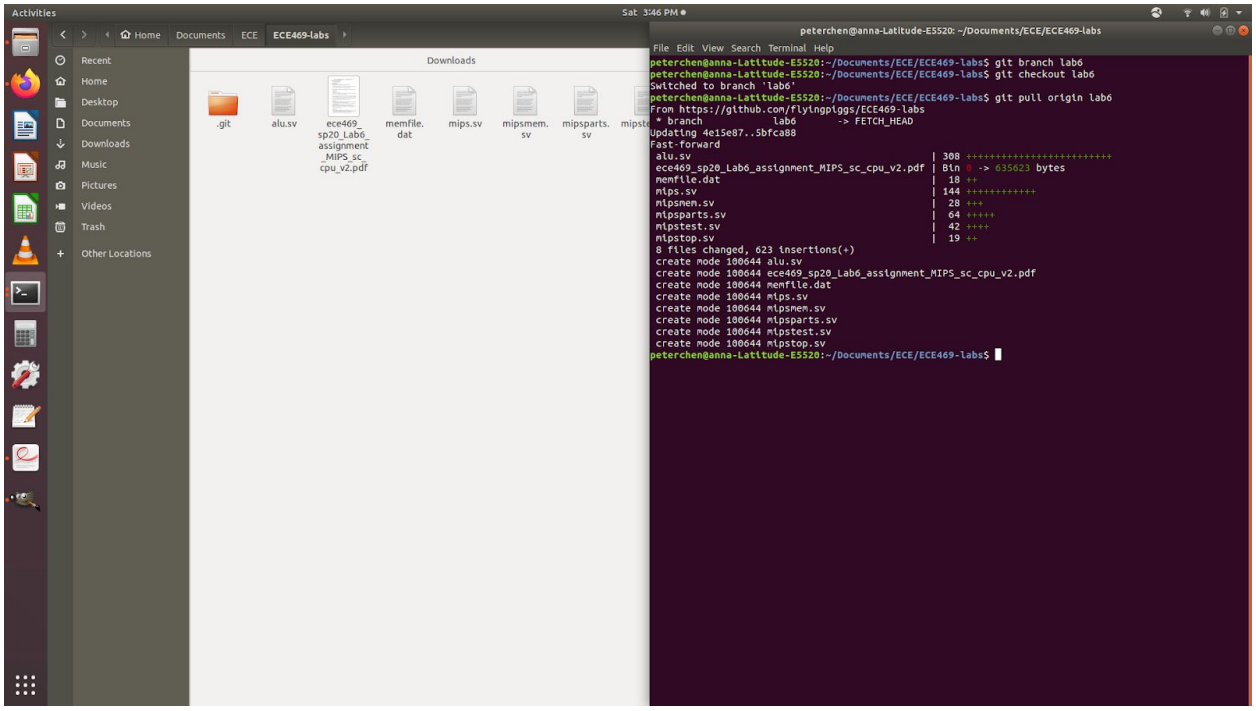
2. Open up the terminal and move your working directory to wherever you want to create the local git repo.
3. Execute the command "git clone https://github.com/flyingpiggs/ECE469-labs.git" like I did, and it'll automatically create a new folder/directory that will be the git repo.
4. Change your working directory to the newly created git repo.
5. Move the machine code file into the git repo, and type in "git status"
    - You should see the file show up as an untracked file
6. You'll most likely be in the master branch, and you can check which branch you're on by typing in "git branch".
    - You'll only see what branches the local git repo has, and the working branch should be highlighted in green and have an asterisk next to it.
    - This is different from all the branches that are on the Github repo along with any other remote repos, like forks, that are associated with this local git repo.

- This is why I've been making a distinction between the git repo and Github repo because the git repo is local on your computer while the Github repo is remote and on Github's servers.
- You can see all the branches, including the Github repo's branches by typing in "git branch -a"



7. You'll need to create a local "lab6" branch.
    - Create the branch by typing in "git branch lab6". For future labs, the command syntax is the same, but just replace "lab6" with the names of the future labs.
        - It should be noted that whenever you create a new branch, it'll use your current working branch to base the new branch off of unless you specify which branch the new branch is supposed to be based off of.
            - You can figure out how to do this by looking at "git branch"'s code documentation if you want.
        - This means that for future labs, you should be on the master branch before creating a new branch if you want to be on a blank slate.
8. Now we change the working branch to the lab6 branch.
    - Change your working branch to the new branch by typing in "git checkout lab6", and for future labs, you'd replace "lab6" with whatever the new branch is.
        - "git checkout" can also be used for version control, provided that you haven't committed any changes, but we'll discuss this at a future date.
9. This new, local branch will not have the files that are on the remote branch, so we need to pull in the remote branch's files.
    - First, type in "git fetch origin", just to make sure you have the most recent git log if I end up making changes to the repo before you actually get through this document.

    - Look up the documentation on “git fetch” to understand what it does, and why it’s necessary.
- Afterwards, type in “git pull origin lab6”.
- The first parameter indicates which remote we’re pulling from and the 2nd parameter indicates which branch of the specified remote we’re pulling from.
- You will now see some messages and the files should show up in the folder now as seen in the picture below



10. Now you’ll need to stage the machine code so that it can be committed.
    - Stage all the untracked local files/changes by typing in “git add .” or specific files by replacing the dot with the specific file name (the dot indicates current working directory, you should look up command line shortcuts and wild cards if you want more information).
    - The file will show up as a staged file now if you type in “git status”

11. Now commit the changes by using “git commit”
    - Some sort of text editor should pop up, which editor that pops up is chosen when you install git. I think it’s Vim by default, but you’ll most likely want to use something else since Vim is weird. Look up directions online if it is set to Vim about how to change it to something else like VS Code
    - You can also bypass the text editor by using *git commit -m “Your message here”*
        - I switched to italics here indicate that you need to include the quotation marks when you type this into the command line
        - Your commit messages should be succinct and not include too much details, In this case, it should be something like, “Added the machine code to test PartB”
12. Now push these changes onto the remote Github repo so that I’ll have access to it.
    - If you type in “git push”, you should get a message that you haven’t set a default branch to push to, and it’ll tell you what to type to set this.
        - If you set this up, then in the future, everytime you type in “git push” without anything else, it’ll automatically push to whatever it was set to.
        - I think the command is something like “git push --set-upstream-to origin lab6”, but I’m fairly certain I’m messing up the parameters with the dashes so just follow what git tells you to do.
    - If you don’t want to see the message above, you can type in “git push origin lab6” instead.