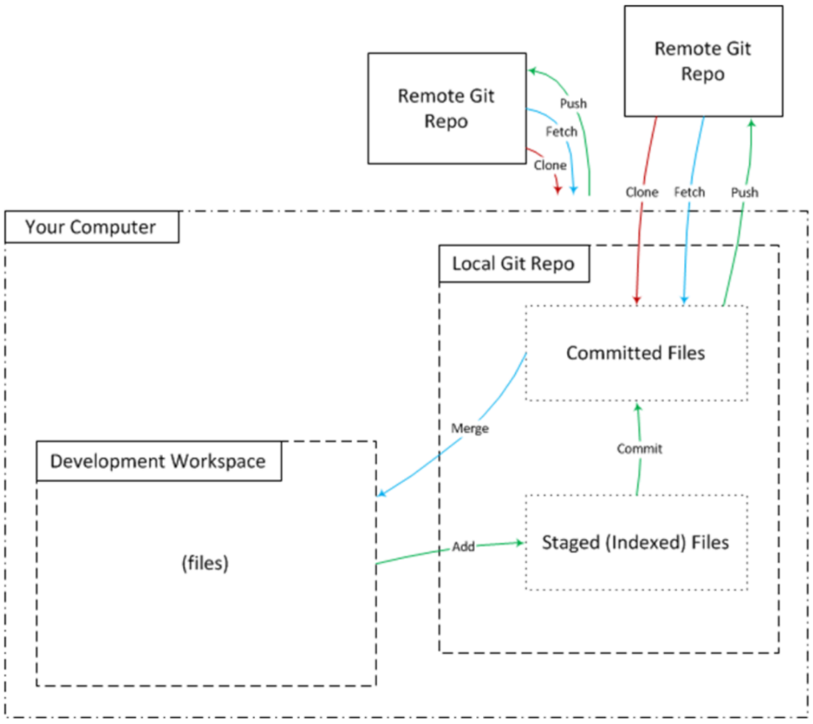
What is Git?

Git is a version control system for tracking changes in files. The distinguishing feature of git from popular version control systems that came before is its distributed nature. The bundle of files that makes up a git repository can be replicated in multiple places, and so syncing them together presents most of the complexity in day-to-day management with git.

Concept of Git

Explaining concept of git in a day in the life story:

At the beginning of the CSE 414, Ryan has come up with a brilliant idea for assigning and collecting homework. Instead of having his TAs drive around and visit everyone's house to deliver the assignment and collect the homework, Ryan sets up a locker for every student in the CSE building. Ryan places the assignment material in the lockers and students can collect them any time they want. Students can then work on their homework anywhere they want, but before the due date, they must submit their final version of homework to their personal locker. When the student finishes the homework, they mark the final version of their homework in the locker "complete," and that's the version that Ryan grades.



Remote repo: This is your locker at the CSE building. We will update them when an assignment is released.

Local repo: This is a box at your home where you keep your homework.

Pull: Fetching the assignment material from your locker (remote repo) to your house

Add, tag, and commit: you can do your homework anywhere in your house, but when you finally complete your final version, you will want to put them in the box (local repo) so that you do not mix them up with your scratch papers.

Push: Submitting your final version of homework from the box in your house (local repo) to the locker in the CSE building (remote repo)

How are we going to use Git in this class?

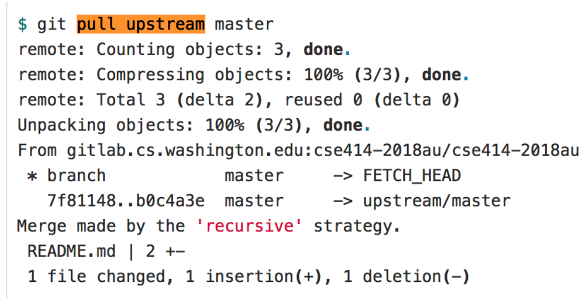
1. Every time a new assignment released, your remote repository in GitLab will be updated with the content. You will use git to pull the content to your local repository and work on the questions.
2. When you are ready to submit your local homework files, you push these files to your remote repository in Gitlab with proper tags
3. Graders will them grade your submission in your remote GitLab repository

Setting up your local Git repo

The procedure on cloning your repo, adding an upstream remote are available in the spec for HW1. They are used only at the very first time you are using git for this class. If everything is set up correctly, you won't need to repeat throughout the course.

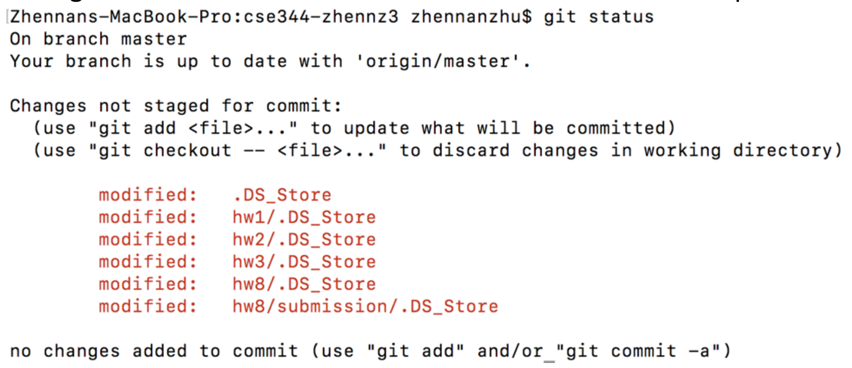
Procedure on submitting your assignment to git

1. Always make sure you have a working internet connection!!!
2. After an assignment is released, use git pull upstream master to first download new assignment starting code from remote repo to local



```
$ git pull upstream master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From gitlab.cs.washington.edu:cse414-2018au/cse414-2018au
 * branch            master     -> FETCH_HEAD
   7f81148..b0c4a3e  master     -> upstream/master
Merge made by the 'recursive' strategy.
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

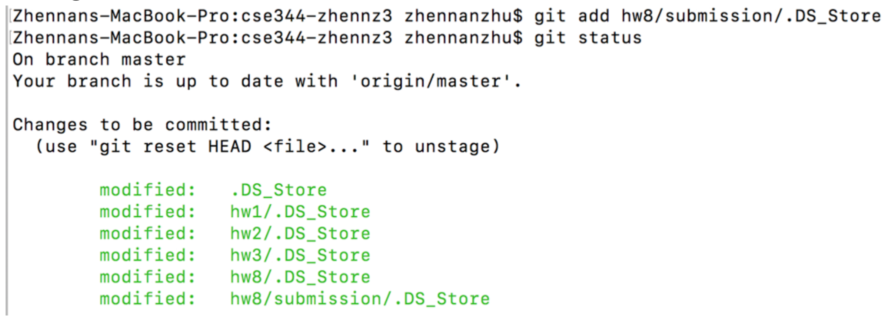
3. Use “git status” to first check for files not added to remote repo or uncommitted changes



```
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .DS_Store
        modified:   hw1/.DS_Store
        modified:   hw2/.DS_Store
        modified:   hw3/.DS_Store
        modified:   hw8/.DS_Store
        modified:   hw8/submission/.DS_Store

no changes added to commit (use "git add" and/or "git commit -a")
```

(in this case, I won’t be able to successfully push my homework because there are new files I created that are not added to the remote git repo, and I have not commited these changes)

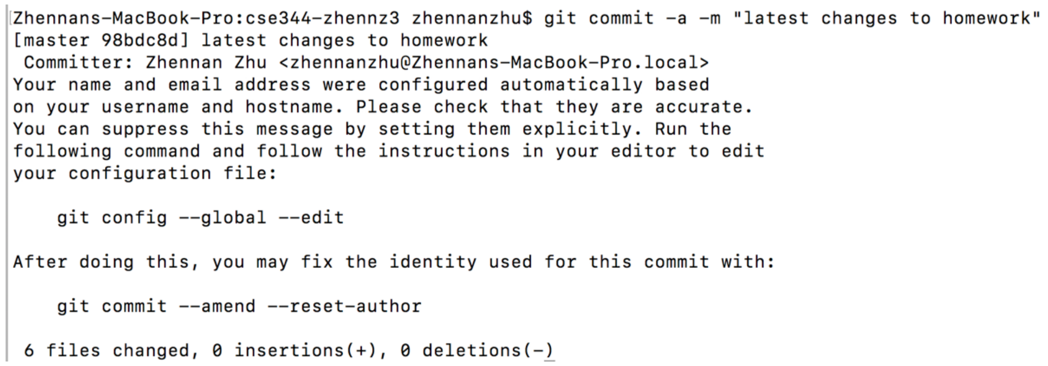
4. Use “git add ” to add files which will be committed later



```
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ git add hw8/submission/.DS_Store
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .DS_Store
        modified:   hw1/.DS_Store
        modified:   hw2/.DS_Store
        modified:   hw3/.DS_Store
        modified:   hw8/.DS_Store
        modified:   hw8/submission/.DS_Store
```

(now I am ready to commit these changes)

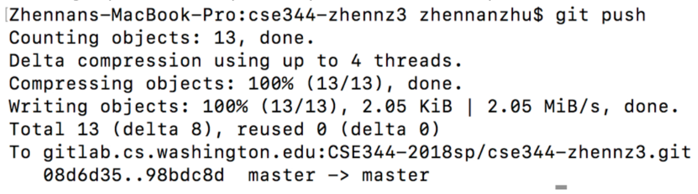
5. Use “git commit” with appropriate commit message to commit changes to remote repo



```
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ git commit -a -m "latest changes to homework"
[master 98bdc8d] latest changes to homework
 Committer: Zhennan Zhu <zhennanzhu@Zhennans-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 6 files changed, 0 insertions(+), 0 deletions(-)
```

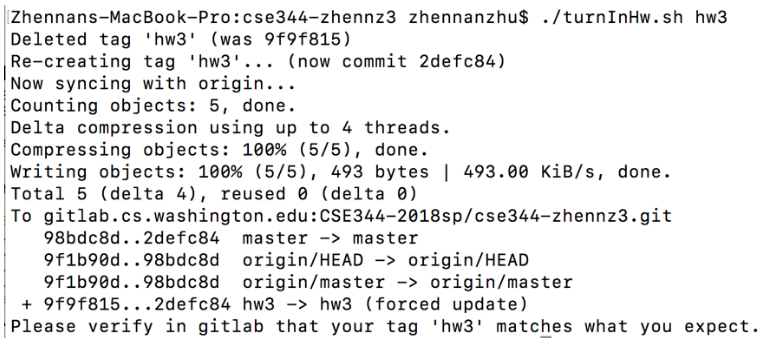
(This message indicates I have successfully committed my changes and shows how many files are changed)

6. Use “git push” to update your remote repo with the latest commit



```
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ git push
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 2.05 KiB | 2.05 MiB/s, done.
Total 13 (delta 8), reused 0 (delta 0)
To gitlab.cs.washington.edu:CSE344-2018sp/cse344-zhennz3.git
   08d6d35..98bdc8d  master -> master
```

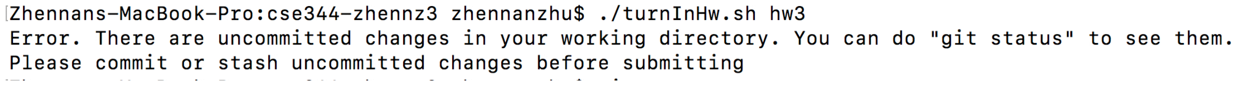
7. Finally, use the script given to you to turn in your homework with the correct tag



```
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ ./turnInHw.sh hw3
Deleted tag 'hw3' (was 9f9f815)
Re-creating tag 'hw3'... (now commit 2defc84)
Now syncing with origin...
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 493 bytes | 493.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
To gitlab.cs.washington.edu:CSE344-2018sp/cse344-zhennz3.git
   98bdc8d..2defc84  master -> master
   9f1b90d..98bdc8d  origin/HEAD -> origin/HEAD
   9f1b90d..98bdc8d  origin/master -> origin/master
 + 9f9f815...2defc84 hw3 -> hw3 (forced update)
Please verify in gitlab that your tag 'hw3' matches what you expect.
```

Common problems encountered when using git

1. Uncommitted changes



```
Zhennans-MacBook-Pro:cse344-zhennz3 zhennanzhu$ ./turnInHw.sh hw3
Error. There are uncommitted changes in your working directory. You can do "git status" to see them.
Please commit or stash uncommitted changes before submitting
```

When you see this message, it means that you are pushing your changes before first committing them. Run through **steps 2 – 4** to first commit your changes first!

2. Interrupted pull

When we are using git pull but the process is terminated before completed, we sometimes may get this error message when we try to pull again:

```
You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you can merge.
```

Do not panic, we can resolve this problem with the following steps:

1. Use **git status** to check the state of our repo. Often times all we need to do is to add files and commit changes. Then a git pull will be successful.
2. Alternatively, we can git clone your Gitlab directory in a separate directory, then copy your homework files you want to submit into the new directory and submit them there.

Optional: you can find a detailed description of how to handle merge conflicts from this link: https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts

3. Git opens up a text editor to request for a message file for the commit to be completed

> "please enter a commit message to explain why this merge is necessary, especially if it merges an updated upstream into a topic branch."

This is not a git error message, it's the text editor as git uses your default editor.

This can be resolved by writing a message and exit the text editor

If you are using a Linux system and has Vim as your default editor (which is normally the setting if you are using a UW VM), then perform the following steps:

1. Press ‘i’ to change in to editing mode.
2. Write a message
3. Press ‘esc’ to exit the editing mode
4. Type ‘:wq’ and press enter to exit vim

You can also change the default editor to others like nano. The instructions for this are easy to find with a web search.

A word of advice

There will be all kinds of problems when we are using git if you do steps in the wrong order or omit certain steps. This is perfectly normal and this is not the end of the world. Whenever you encounter a problem, the first thing to do is to **make a backup of your homework files**.

Most of the problems we will encounter can be solved by git add, commit and push. However, if you run into rare cases, post on Piazza and we are more than happy to help you. Meanwhile, make good use of the internet resources. It’s very likely that whatever issue you run into, many others have had to figure out before. Google the error message and you can see how people with similar problems solve them. After learning git, it will be a very powerful tool for you in the future when working on group projects. So don’t be afraid of errors, it’s all a learning experience.