CS383 Git Manual

**Setup**

1. Explain Git
   1. Version management software
   2. Record changes in your project
   3. Combine with software repositories
      1. GitHub, GitLab, BitBucket
   4. Allows storing project and history of recorded changes in a centralized, remote location, facilitating parallel remote development
   5. We will be going over project and repository setup using GitHub
2. Show GitHub setup
   1. Login or Create account
      1. Walk class through logging in
   2. Navigate to new repository
   3. Name and description
   4. Explain differences between privacy settings
      1. Public repositories can be accessed by everyone.
         1. Often used as webpages for software
      2. Private must be explicitly shared to be viewed
      3. Both require explicit permission to collaborate
   5. Show readme.md, how it works, what markdown is.
      1. Includes file at repository root
      2. Works as homepage info for your project [show GitHub page]
      3. Markdown editors like StackEdit
      4. Markdown guide included in our guide
   6. Show .gitignore, .gitignore templates, how making .gitignore files works
      1. Ignores files per spec
      2. Allows changes to your local directory without affecting repository
      3. GitHub template dropdown (ex. Unity)
      4. Make your own using Git documentation (link included)
   7. GitHub interface
      1. Show navigating GitHub directory structure
      2. Show editing files on GitHub
      3. Show adding collaborators
3. GitHub Desktop Setup
   1. Show Windows/Mac download, mention Linux downloads
   2. Show cloning a repo
   3. Show setting up a Unity project with GitHub Desktop
      1. Issues with creating a project over an existing repo, issues with cloning a repo into an already existing project, show getting around this by creating a project and with the same name and moving it into the repository.
4. Setting up with Git GLI
   1. Show cloning a repository
   2. Show adding a to the repository and failing authentication
   3. Show authentication setup
      1. Navigate to GitHub Profile Settings > Developer settings > personal access tokens > Generate new token
      2. This allows you granular control of what machines have access to what aspects of your GitHub account through git
      3. git config --global credential.helper store
   4. show adding to the repository and entering login info
   5. show repeating and not needing login info
5. [Workflow]
6. [Disaster Recovery]
7. [Q&A]

* **Git Workflow:**

Before starting make sure that you are in the root directory of the project in terminal or have the projected loaded on a gui application.

1. **Branches:** Each branch is essentially a pointer to a snapshot of changes to the repo and any commit will only affect the selected branch on git. To list all branches, use ‘git branch’. To change the current working branch use ‘git switch <branch name>’ For this project we will only use the ‘main’ branch.
2. **Pull:** Use ‘git pull’ to download any changes from the online repository. This should be the first thing you do anytime you start coding. You will want your code to be up to date with your team since you will be using some of their functions. Also if changes are made to the same line of code then there will be a merge conflict which will have to be resolved before you can push your code to the repo. In this class though we shouldn’t have to worry about that sense no one should be working on the same file.
3. **Stagging:** Each change that you want to commit first needs to be stagged. Use ‘git add’ to stage any changes or added files to the current branch. You can add single files, directories, every change in the branch, or do it interactively by hunks. Note that adding every change at once is fast and easy it also makes it easy to add unwanted changes without realizing it.

* ‘git add <path>’ stages a single file or directory
* ‘git add .’ a period means to add every change in the repository
* ‘git add –p' interactively select chunks of code to stage. Note can’t be used to add new files

1. **Commit:** Use ‘git commit’ to save any staged changes to the current local branch. To stage a change us ‘git add <file name>.

* ‘git commit –a' commits all changes in the working directory. Not only staged changes.
* ‘git commit –m ”commit message”' adds a message to the commit. Every commit should have a message and it should be a brief description of what the changes do.
* ‘git commit –am “commit message”’ Commits all changes with a message.

1. **Push:** ‘git push’ will upload all the commits from your local branch into the online git repository. Do this when your code is at a point where it is ready to be shared with the group. You should generally not push code that does not compile or is broken. Also make sure all your changes are committed before pushing.

Here is a more in depth guide for using git <https://github.com/git-guides/>

**Using git website/Desktop:**

**1.) Setup:**

-First navigate to your teams main repository, Tl1 should send you a link to their github repository. Star this repo, or save it in any way you choose. This is where you will be submitting assignments.

**2.) Repository:** What is it? Like a project folder, or C disk drive. It stores all your current working projects.

**3.) Branches:** What is it? Similar to a file system. Similar to a root folder with sub folders in them. In a nut shell branches work the same way. Branches are like sub files branching off from your root folder, the repository.

**4.) Commit:** What is it? Committing is saving changes on YOUR local machine only if using git desktop or command line. This does not change or upload anything into the main repository. If you are using git website, commit will save changes to the main repository DIRECTLY. No pushing is needed.

How to commit: In the git website select the upload files, or the create new files option. Fill out the name, and options as needed, then press "Commit". In github desktop, drop files into your github folder, then press “commit”, on the github desktop.

**5.) Push:** What is it? Pushing is like uploading files to the main repository. In a sense Git will take the files off your local machine and put them into the main repository.

How to Push: If using the github webiste, there is no local repository on your computer to push from. We can commit directly to the main repository without pushing. If using github desktop select “Push to origin”, after committing.

**7.) Pull:** What is it? Pulling is like pulling files from the main repository, and putting them onto your local machine. You are pulling the files.

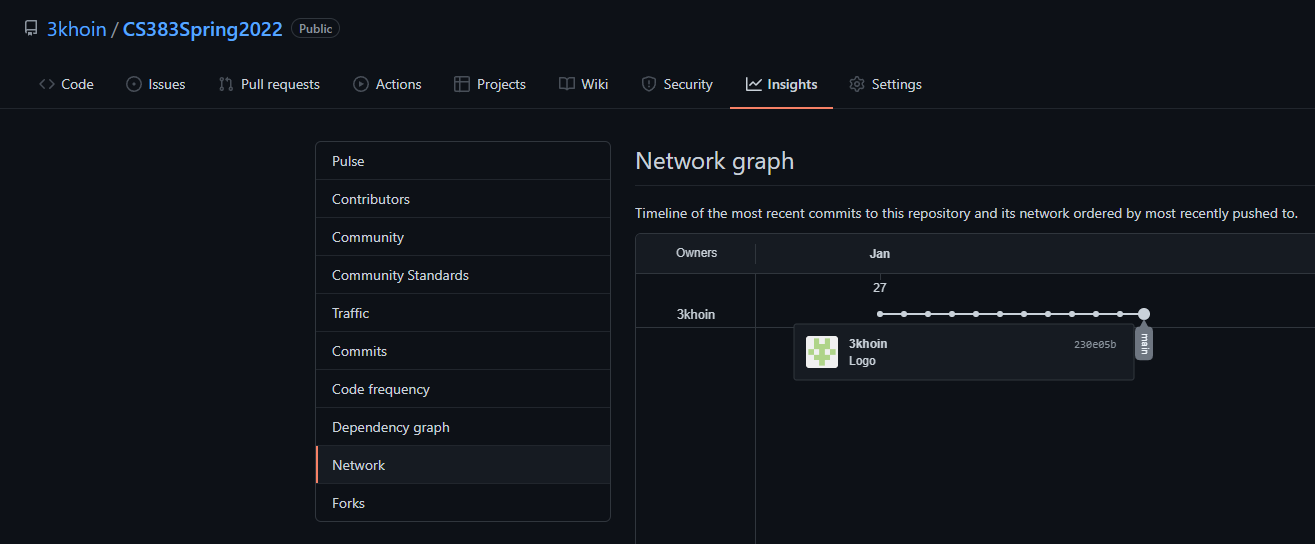
How to pull: In the github website if there are any changes made in the main repository, you can select the pull from origin option, or fetch origin to pull and refresh the git files on your desktop.

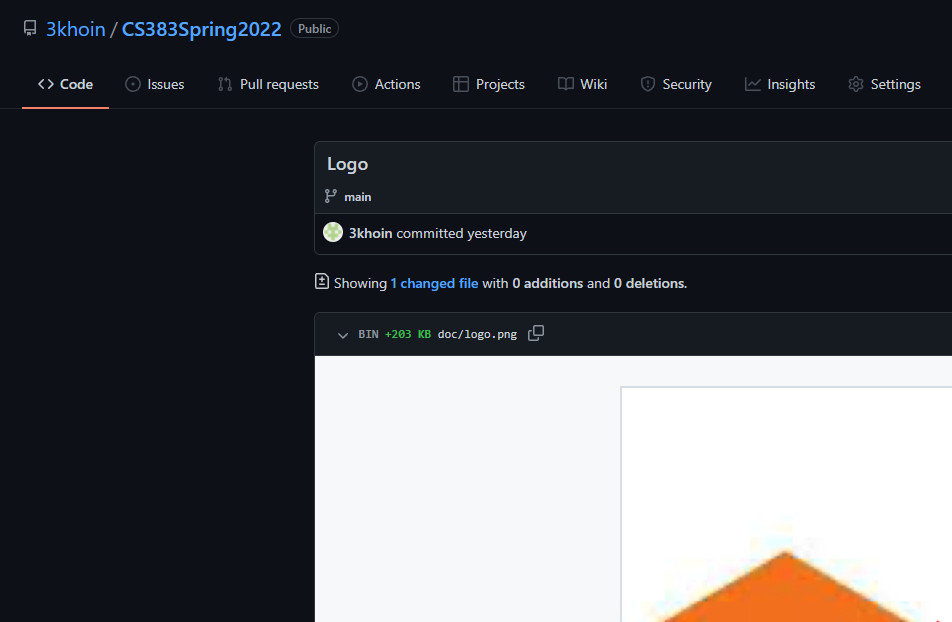
Version Control:

You may find that, having made too many unfamiliar changes to a branch at once before testing, its contents have ceased to function. Alternatively, you accidentally committed or pushed unwanted changes by accident. When something like one of these cases happens, and you either wish to troubleshoot the problem or just ensure that you have a functional product to present, you will want to revert to an earlier version that you know works.

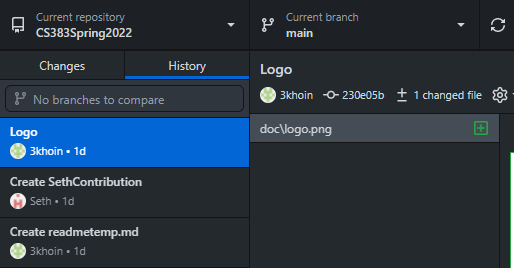
To begin, you will want to have the full version history of your branch so that you can identify which version you want to revert to.

Website: Browse to your Git branch -> Insights -> Network -> Click on commit





Desktop: Open your Git repository -> History -> Click on commit



Once you click on a specific commit, you will be able to identify which files have been changed since the previous version of the branch was pushed, as well as specific changes within the files.

For your later convenience, you should have a meaningful commit message for each commit as well as possibly a quick status update indicating whether the previous version was functional at the time of commit. That way, if you encounter any dysfunctionality in a branch in the future, you will be able to quickly recognize which commits for that branch you can safely revert to.

Now that we have identified the specific version to which we want to revert to, we must then transition to the GitHub Desktop app or the terminal.