Git and GitHub

# Git

Git – Open source and free source control management (SCM)

Git allows files to be managed and shows what changes have been made to those files over time.

Git can be accessed with Git Bash or the terminal once Git is installed.

* Git probably was installed from CS 232

Make a git repository:

* Make a folder, name it, and take note of its location in your local system
* On Git Bash, navigate to that file location
  + cd to navigate
  + ls to see what is in your folder
  + ls -al for more details
* use command “git init” to initialize current location as a repository
  + Should see “Initialized empty Git repository in . . . “

To see the status of your repository, use command “git status”

Any files in your repository will be untracked, which can be seen with “git status”

* Untracked files will not be paid attention to by git

To track a file, simply use command “git add <file name.ext>”

Using git status again will show that the file is now being staged, or tracked.

To remove the file from being tracked use command “git rm –chached <file name.ext>”

To have git ignore some files completely:

* Make a text document, delete the name AND extension
* Name the file “.gitignore” – notice the period
  + A popup will ..popup, press yes
* Open the document with a text processor like Notepad
* Use # for comments if wanted
* Type in what files you want ignored
  + \*.txt, for example, would ignore all .txt files
* Useful for ignoring autogenerated files during compilation or something

To track all the files “git add .” or “git add .A” or “git add .all”

Commit – like taking a snapshot of your repository at that point in time

* Like writing an entry in a log and can go back to it

To commit “git commit -m “<commit message here>”

Using git status again at this point will not show anything

* Everything has been committed at this point
* Changes in any of the files in the repository will show as untracked

At this point, any changes can be viewed with “git diff”

* Compares last committed file with changed file

Three git environments:

* Working files – can make edits to files
* Staging – holding
* Commit – item has been added to the log, written in history

To take files out of staging: git restore –staged <file.ext>

You can also skip over staging with the command “git commit -a -m “<commit message>””

Git can also remove files “git rm “<file.ext>””

However, if that file has been committed, using git status will show that in the record  
 deleted: <deleted\_file.ext>

To get the file back: git restore “<deleted\_file.ext>”

Rename the file in the terminal: git mv “<original\_file.ext>” “<new\_file.ext>”

Review all commits: git log

* Shorten log:
  + git log –oneline
* View detailed changes of commits
  + git log -p

Manual for git log: git help log

* Many different things that can be done in log

With commit history in the log, previous commits can be jumped to: git reset <#tag from log>

To modify things in the history book: git rebase -i –root

* Opens an editor with many many options
* To exit this view press **:** then **x** then **enter**

*Setting up different branches*

Another branch is a copy of the main branch

* Has all the same entries
* Changes made on branches have no effect on main branches
* The file then can be merged back to the main branch

To branch off: git branch <BranchName>

To view the branches: git branch

* Asterisk and highlight shows currently active branch

To change to another branch: git switch <BranchName>

* Any files in the repository will look the same, but you are viewing those files in the branch

Anything done while in this branch is the same as if it were the main or master branch

If you switch back to your main (git switch main/master), opening the file in the repository that was committed while in the branch shows no change.

* Files are reflective of the main
* Changes were only made in branch

To merge changes back in: git merge -m “<Merge message>” <branch to merge back into main>

After you are done with a branch, to delete it: git branch -d <BranchName>

What if you branch from main, and when you merge the main branch has changed (my concern)

To create new branch and switch to it: git switch -c <branchName>

To create a merge conflict:

* Branch from main
* Make changes
* Commit these changes (in branch)
* Switch back to main
* *Do not merge*
* Change the same file in main
* Commit these changes in main
* Try merging branch to main
* Error message to say merge conflict
* This will put you in a new, autogenerated branch (main | MERGING or something like that)
* Open the conflicting file in the repository to see a diff-like view of the file
* <<<HEAD and >>>>BranchName indicate the different parts
* Changes made in this intermediate branch can then be committed
  + git commit -a -m “<commit message>”
* The commit will then go through as you indicate in the merging branch

# GitHub

Git is hosted on the local computer, but the repository can be hosted on the cloud

GitHub is the most popular cloud repository

In Git Bash, navigate to your repository, or a new folder and make it a repository

* git init (while in that folder)

To establish connection to the GitHub repository:

git remote add origin <https://github.com/> . . . / . . .

Set target branch to main:

git branch -M main

Push any files in your repository to GitHub:

git push -u origin main

Push branches to GitHub:

git push –all

On GitHub, we can look through the different files in that repository

Clicking on commit messages show changes made

Issues

Enter in feature request, bug information, etc.

Can be assigned to other collaborators

Changes

Editing files can be done directly in GitHub

Can commit directly to the main branch or can create a new branch and start a pull request

* If creating a new branch, name it something that explain changes

Pull request – request to merge back to main

After a pull request is made, we can all discuss the merits of the changes before merging

When reviewing pull requests, the right sidebar has a “Development” section. This allows us to select an Issue that we can tie together with the pull request.

Confirming the request closes the corresponding issue.

Actions

Actions tab at the top can be used to run different tests

Projects

Project management view for different topics

Wiki

Document our code, like Wikipedia for our project

Security

Define our policies

Insights

Traffic, contributions, other stats on project

Settings

Configure various settings

Collaborators – additional people to work on project

Get changes from GitHub to local Git repository:

In Git Bash: git fetch, then git merge

Or git pull (combines fetch and merge)