Intro to git / github.com

Version control and code sharing

Terence Parr
MSDS program
University of San Francisco

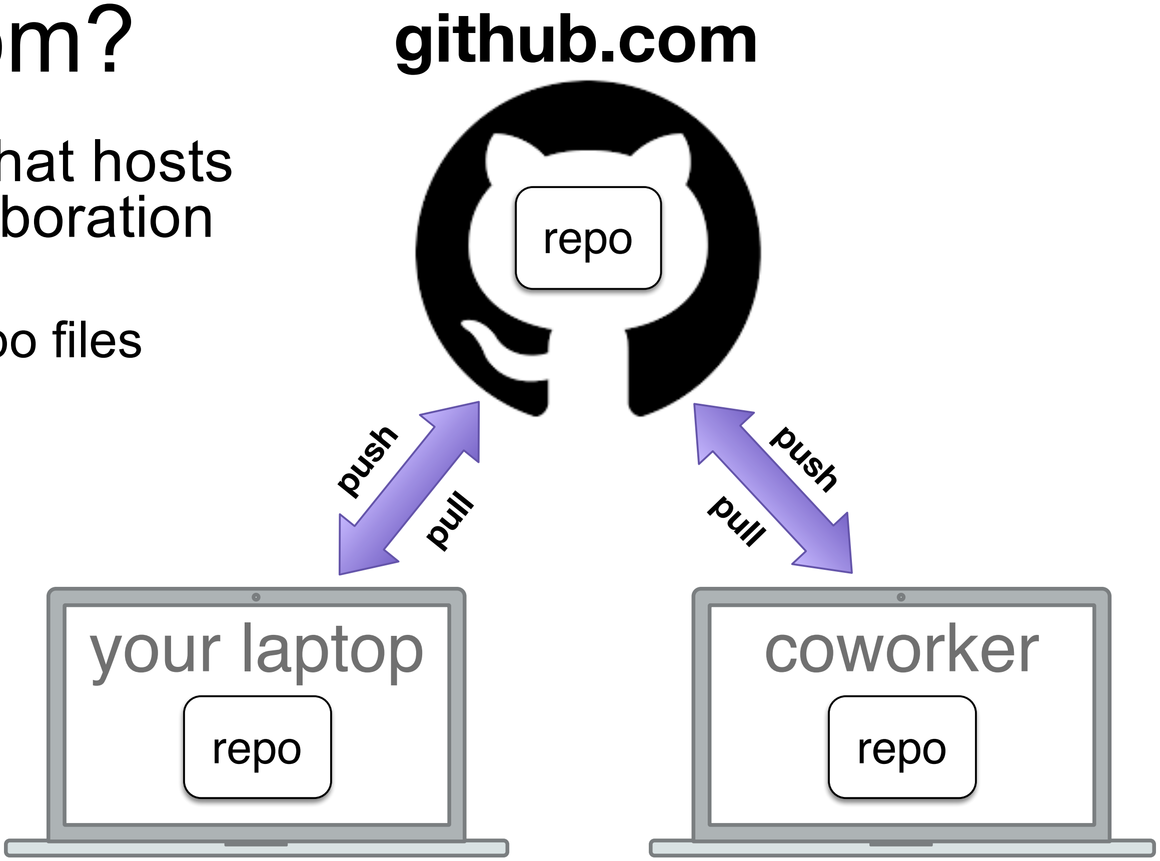
UNIVERSITY OF SAN FRANCISCO

What is git?

- Git is a version control system that tracks changes to files and directories within a repository
- A repository is just a directory subtree containing files and, optionally, directories that we tell git to treat as a repository
- Git allows multiple people to operate on two different copies of the repository without getting confused or losing changes
- Workers push/pull changes from a repo on one machine to a repo on a collaborator’s machine
- Git is a program that runs on your laptop

What is github.com?

- Github.com is a website that hosts repositories, making collaboration much easier
  - A web interface to your repo files
  - A free backup!
- Note: git != github.com
  - git is program
  - github is a web site/server
- For our purposes, we'll ignore the advanced capabilities, such as branching and merging (master/main/...)



github.com

repo

push

pull

push

pull

your laptop

repo

coworker

repo

See [github learning resources](#)

Motivation

- Every commercial developer uses version control at work
- Every company you encounter uses it
- For that reason alone, you need to learn version control to be functional in a commercial setting, such as your practicum
- In this class and future classes, you will also use version control to submit your work

An analogy to backup systems

- If your laptop is stolen, we will be sympathetic but not excuse missing projects
  - github doubles as a backup
  - but I recommend you also get backblaze to keep off-site backups of your disk
- Personally, I also have a local Timemachine OS X backup drive sitting next to my computer that takes a snapshot every hour
- Using this multi-tiered backup strategy is a good way to think about how programmers use version control
  - git is kind of like Time Machine, a local backup (that tracks changes)
  - github.com is kind of like the off-site backblaze cloud-based backup
- A difference between git and a backup system is that we tell git **when** to take a snapshot
- Each snapshot should be a logical chunk of work done to your files

Repositories (Repos)

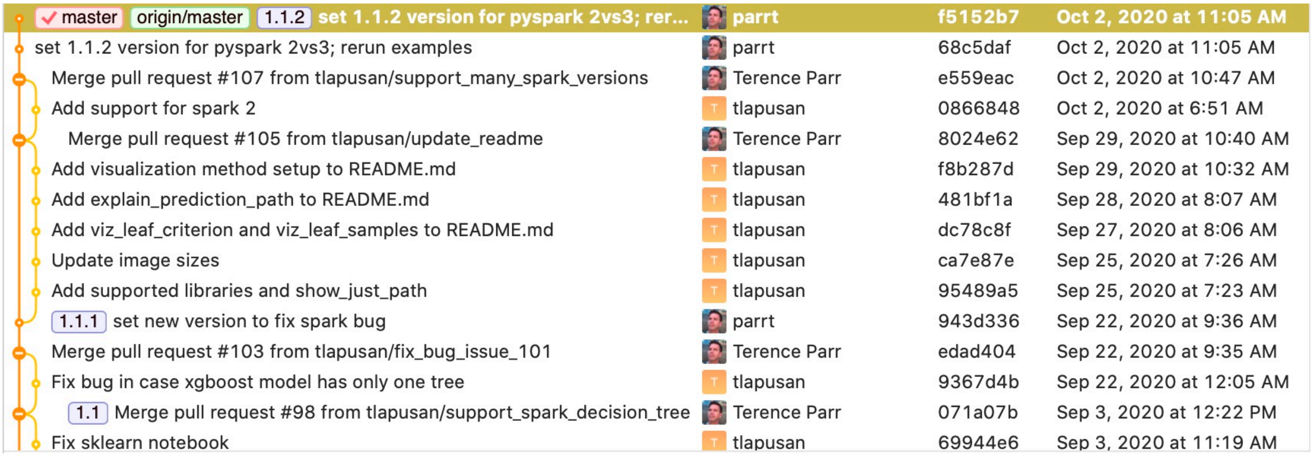
- Not only do we have to tell git **when** to take a snapshot, we also tell it **which** files to pay attention to (in the repo directory)
- The set of files to track is called a *repository* and at any given time, my computer has lots and lots of these repositories
- All files associated with a repo sit somewhere in or below a directory
- Each project you work on will be in a separate directory/repo
- A git repository instance is just a directory but it also has a **.git** (hidden) subdirectory, with a database of all changes
- To remove a repo, just **rm** the whole repo directory; there is no central server to notify (this would not delete repo from github.com)

Committing changes

- As with the Time Machine backup, git tracks snapshots as the difference from the last time you requested a snapshot
- Each snapshot is called a *commit* (and programmers think of these commits as *transactions*)
- Perform a commit to lock in a logical chunk of work, such as the addition of a feature or fixing of a bug
- **Warning**: always use the "-a" option on the git commit command

Commit log (history)

- Having a complete list of changes is extremely useful
- We can revert those change sets later
- We can discover who created or when a bug was introduced
- Can temporarily reset your repository to a moment in time

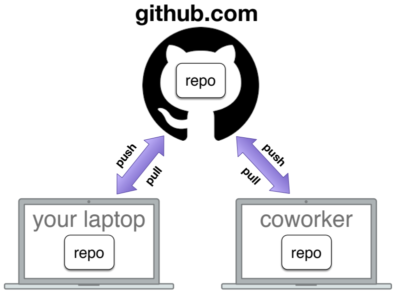


✓ master origin/master 1.1.2 **set 1.1.2 version for pyspark 2vs3; rer...**	**parrt**	**f5152b7**	**Oct 2, 2020 at 11:05 AM**
set 1.1.2 version for pyspark 2vs3; rerun examples	parrt	68c5daf	Oct 2, 2020 at 11:05 AM
Merge pull request #107 from tlapusan/support_many_spark_versions	Terence Parr	e559eac	Oct 2, 2020 at 10:47 AM
Add support for spark 2	tlapusan	0866848	Oct 2, 2020 at 6:51 AM
Merge pull request #105 from tlapusan/update_readme	Terence Parr	8024e62	Sep 29, 2020 at 10:40 AM
Add visualization method setup to README.md	tlapusan	f8b287d	Sep 29, 2020 at 10:32 AM
Add explain_prediction_path to README.md	tlapusan	481bf1a	Sep 28, 2020 at 8:07 AM
Add viz_leaf_criterion and viz_leaf_samples to README.md	tlapusan	dc78c8f	Sep 27, 2020 at 8:06 AM
Update image sizes	tlapusan	ca7e87e	Sep 25, 2020 at 7:26 AM
Add supported libraries and show_just_path	tlapusan	95489a5	Sep 25, 2020 at 7:23 AM
1.1.1 set new version to fix spark bug	parrt	943d336	Sep 22, 2020 at 9:36 AM
Merge pull request #103 from tlapusan/fix_bug_issue_101	Terence Parr	edad404	Sep 22, 2020 at 9:35 AM
Fix bug in case xgboost model has only one tree	tlapusan	9367d4b	Sep 22, 2020 at 12:05 AM
1.1 Merge pull request #98 from tlapusan/support_spark_decision_tree	Terence Parr	071a07b	Sep 3, 2020 at 12:22 PM
Fix sklearn notebook	tlapusan	69944e6	Sep 3, 2020 at 11:19 AM

Cloning from, pushing to github

- Continuing with the analogy now, github.com is like the off-site cloud-based backup
- Each repo you mirror at github is like a free backup
- We'll likely create a repo using a web interface at github then **clone** that repo to an (initially empty) directory on our laptops
- As with committing changes, we also have to specifically **push** changes made to the local repository back to github
- Every push ensures that the complete file set and git change database (in **.git** subdirectory) is mirrored at github

Collaboration



- I can access your repos mirrored on github, whereas I have no access to your laptop drive
- To grade projects, I will **clone** your repository onto my disk
- If you make changes, I can **pull** those in after you **commit**/**push**
- I can make comments and then push back to your github repo, which you can then **pull** down to your laptop
- This is how multiple programmers communicate, and how I share work between my USF and home machines

Key commands summary

- I recommend using a git GUI like fork in practice, but we'll use the command line to learn the actual operations and sequence
- **git clone** *github_url*
- **git add** *file_or_dir*
- **git commit -a -m** '*commit message*'
- **git status**
- **git push origin main** (**main** could be called master)
- **git pull origin main**
- **git rm** *filename*
- **git mv** *from_filename to_filename*
- **git reset --hard HEAD**
- **git checkout –** *filename*

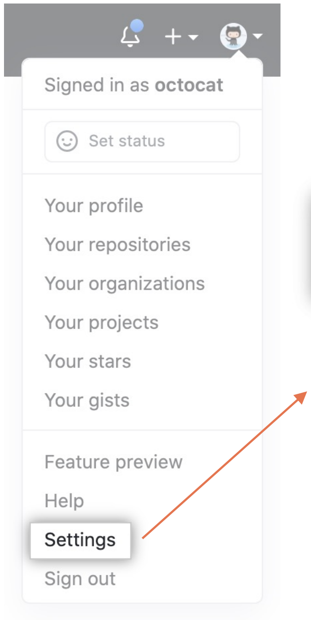
Connecting to github via SSH

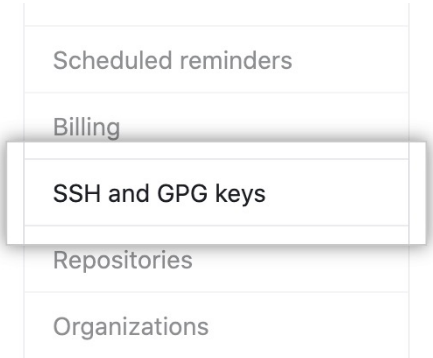
- Naturally you use the website with a username and password
- But to remotely access github repositories from the command line or tools such as fork, we need a secure mechanism to identify ourselves to github
- See the link below; most of you will not have SSH keys yet so follow "*Generating a new SSH key and adding it to the ssh-agent*"
  `ssh-keygen -t ed25519 -C your_email@example.com`
- Press "ENTER" to get all of the defaults; it creates files like: **id_rsa** and **id_rsa.pub** (or **id_ed25519.pub**) in **~/.ssh** dir

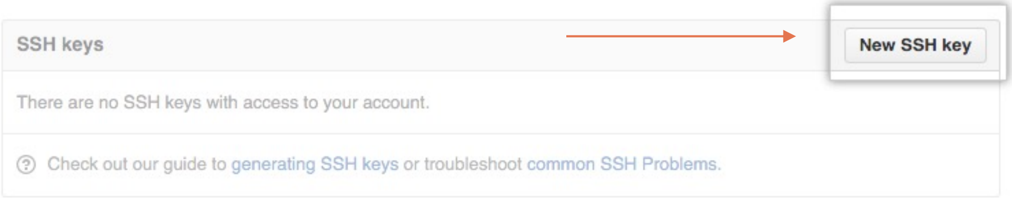
https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh

Adding your key to github.com

- Copy (cmd-C) contents of file id_rsa.pub (or similar); then at github navigate here:

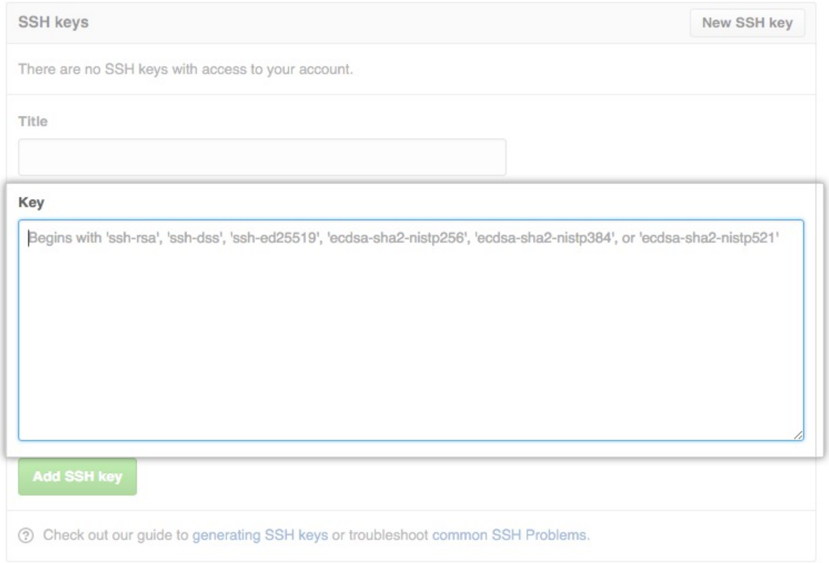






There are no SSH keys with access to your account.

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

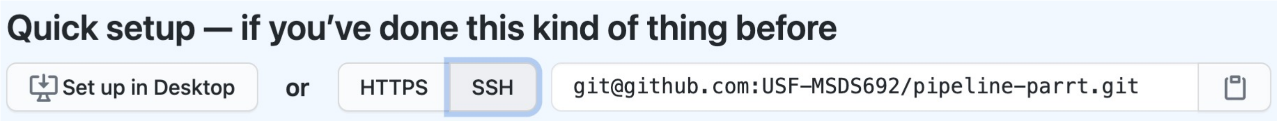


There are no SSH keys with access to your account.

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

Paste (cmd-V) here



Quick setup — if you've done this kind of thing before

git@github.com:USF-MSDS692/pipeline-parrt.git

Typical startup sequence

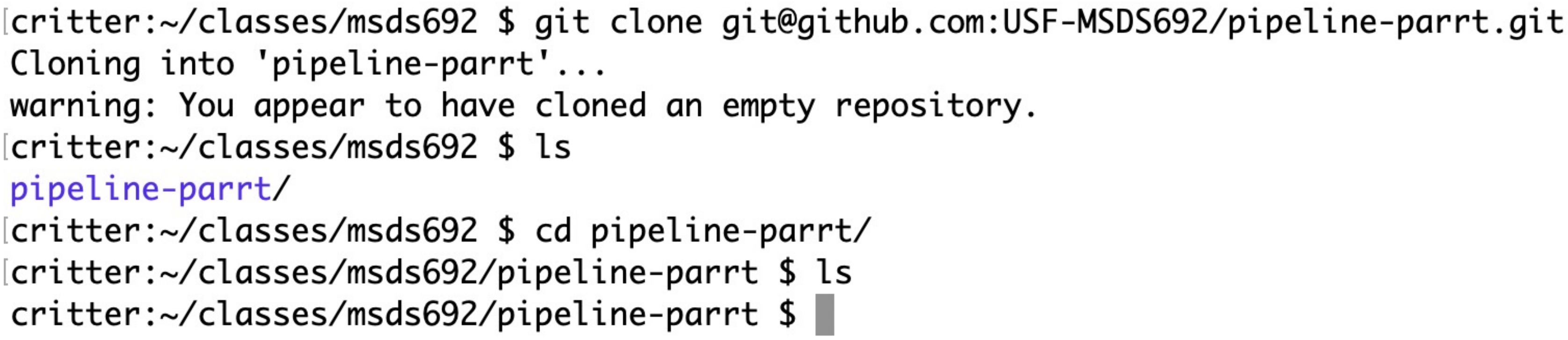
- Click on the invitation URL sent to you by instructor to create a repository, which creates repo at github:

  `https://github.com/USF-MSDS692/pipeline-parrt`

- Get the repo spec from github, which looks similar to repo's github web page URL:

  `git@github.com:USF-MSDS692/pipeline-parrt.git`

- Clone that (empty) repo onto your laptop from command line:

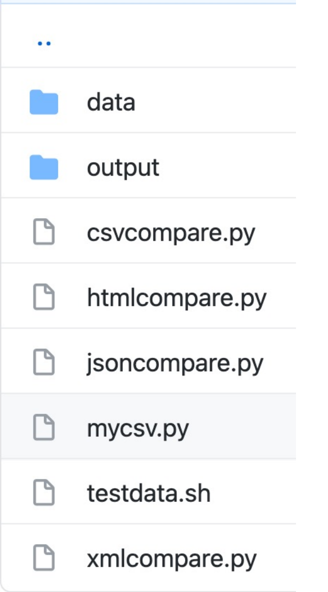


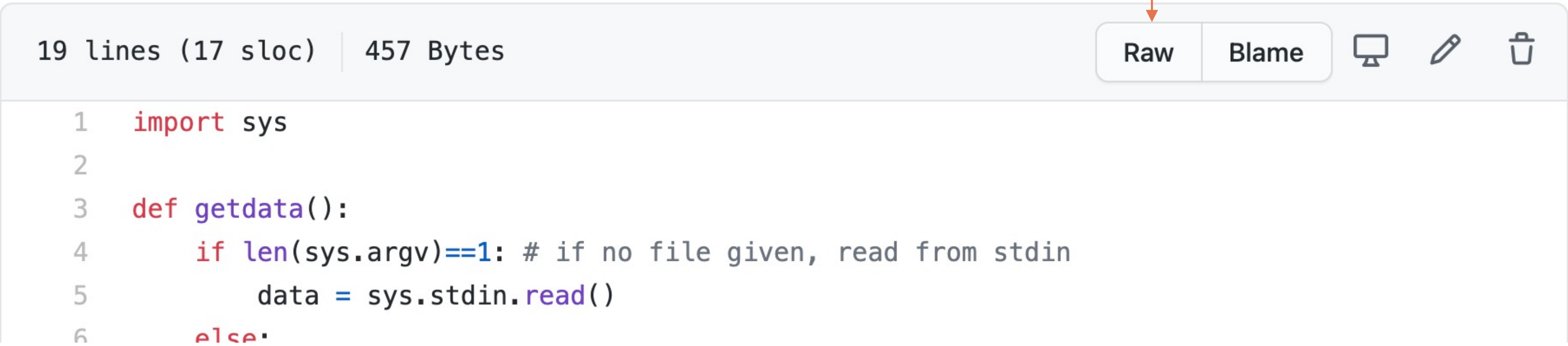
```
[critter:~/classes/msds692 $ git clone git@github.com:USF-MSDS692/pipeline-parrt.git
Cloning into 'pipeline-parrt'...
warning: You appear to have cloned an empty repository.
[critter:~/classes/msds692 $ ls
pipeline-parrt/
[critter:~/classes/msds692 $ cd pipeline-parrt/
[critter:~/classes/msds692/pipeline-parrt $ ls
critter:~/classes/msds692/pipeline-parrt $ 
```

CISCO

Getting an initial file into your repo

- In the directory created during cloning, you will create and edit files associated with the repository
- Let's download a starter kit file for this project; Click on **mycsv.py** and then right click Raw and "Save as…" into your repo directory **pipeline-parrt**





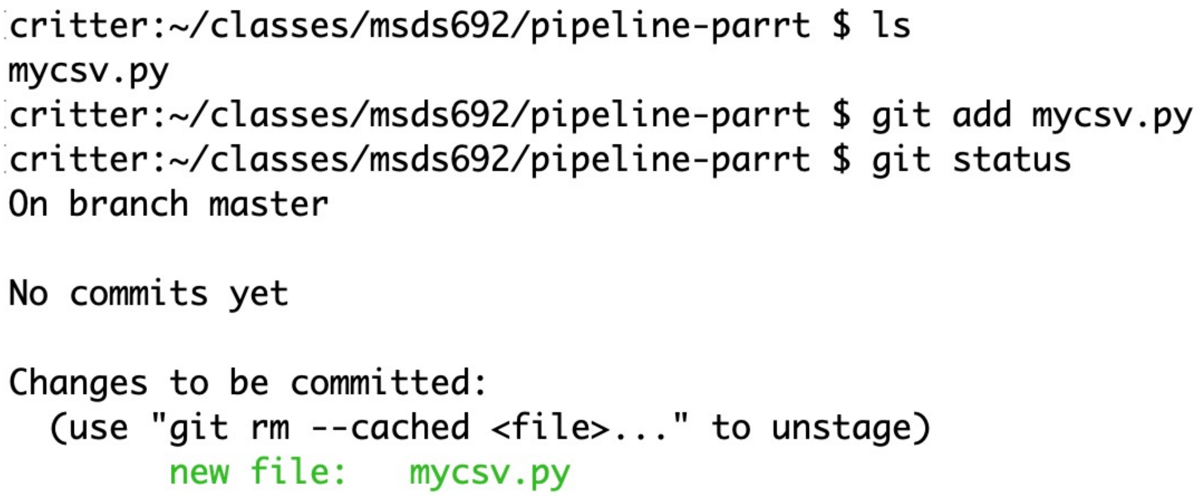
```
import sys

def getdata():
    if len(sys.argv)==1: # if no file given, read from stdin
        data = sys.stdin.read()
    else:
```

Starter kit: https://github.com/parrt/msds692/tree/master/hw/code/pipeline

Adding files to the repo

- git ignores files unless we tell ask it to pay attention; it's not enough just to put files into the repository directory
- "**git add**" the files of interest so git knows to manage them
- Check status; git now sees files



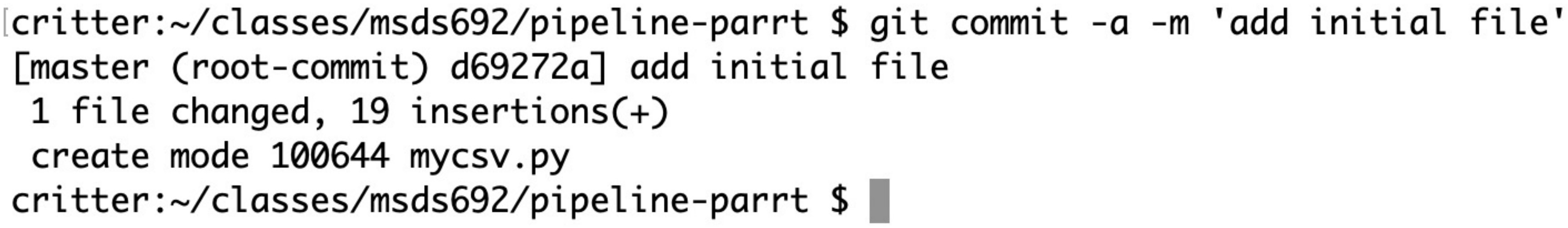
```
critter:~/classes/msds692/pipeline-parrt $ ls
mycsv.py
critter:~/classes/msds692/pipeline-parrt $ git add mycsv.py
critter:~/classes/msds692/pipeline-parrt $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   mycsv.py
```

Commit a transaction

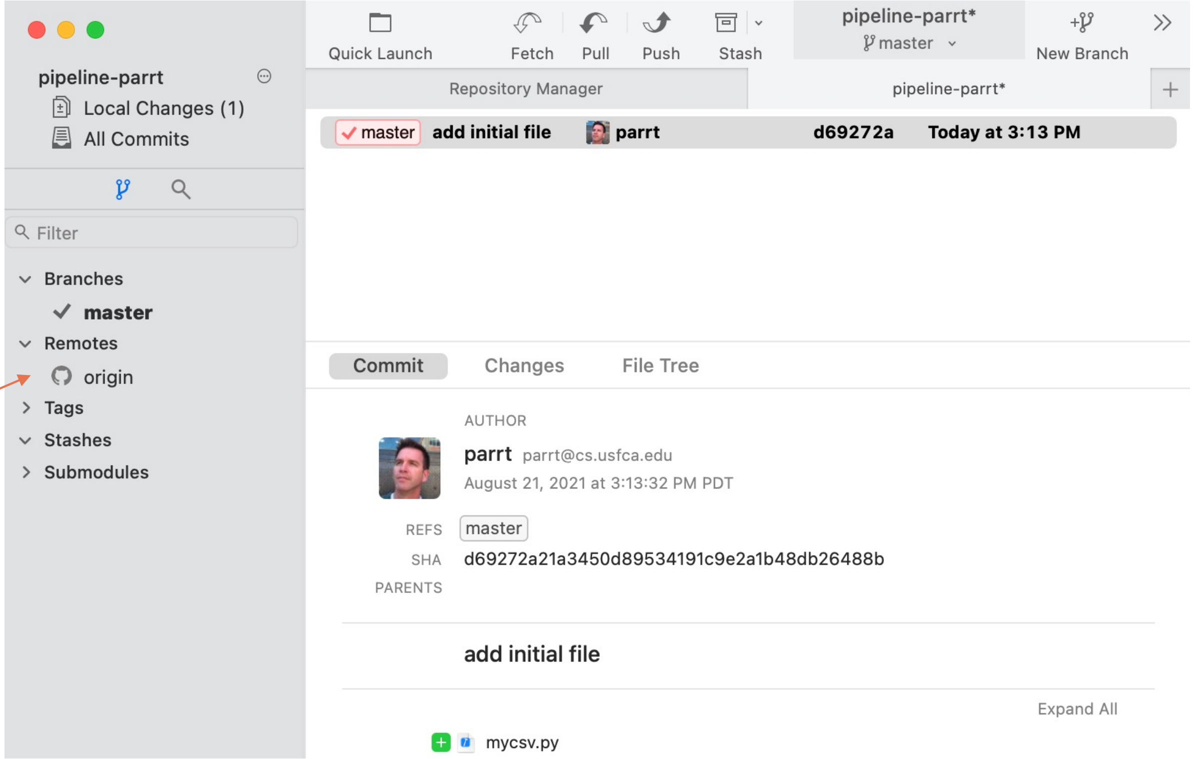
- Commit tells git to take a snapshot and record it in its log of changes
- Additions, deletions, renamings are all considered (reversible) changes
- Use a decent commit message and **don't forget** the "**-a**" argument which means "*do what this command should do by default*"



```
[critter:~/classes/msds692/pipeline-parrt $ git commit -a -m 'add initial file'
[master (root-commit) d69272a] add initial file
 1 file changed, 19 insertions(+)
 create mode 100644 mycsv.py
critter:~/classes/msds692/pipeline-parrt $ 
```

The fork GUI view

- There is only one commit but you can see the commit message and the files involved in the transaction
- You can also see the **origin** remote repository is connected because it's listed in the left gutter



master add initial file parrt d69272a Today at 3:13 PM

AUTHOR
parrt parrt@cs.usfca.edu
August 21, 2021 at 3:13:32 PM PDT

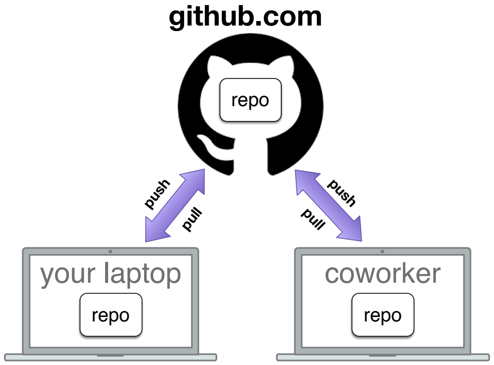
REFS master
SHA d69272a21a3450d89534191c9e2a1b48db26488b
PARENTS

add initial file

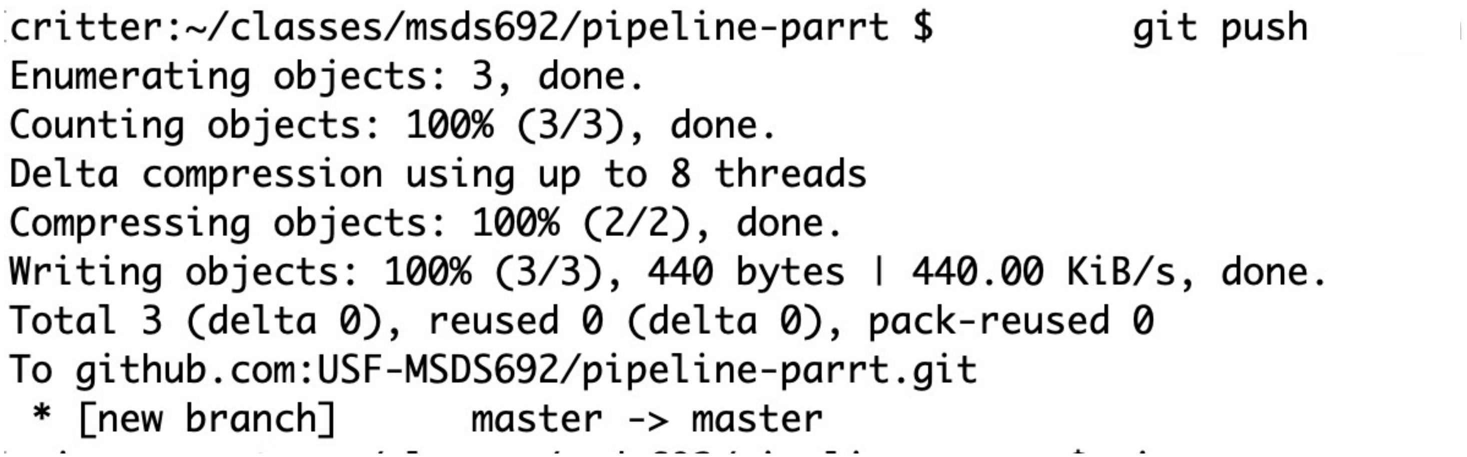
mycsv.py

Download fork here: https://git-fork.com/

Push to github to mirror repo



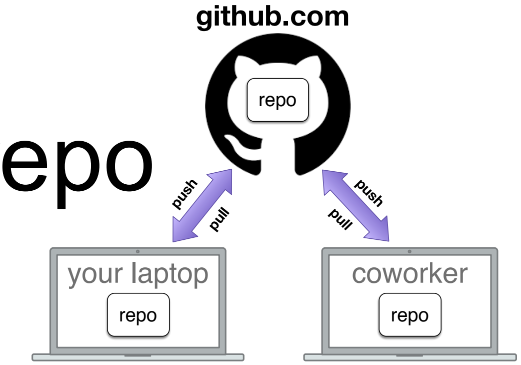
- Github does not know about your changes unless you explicitly push after committing
  - We're ignoring branches but we need to know what the main branch is called; it's either master or main (**master** is the legacy name)

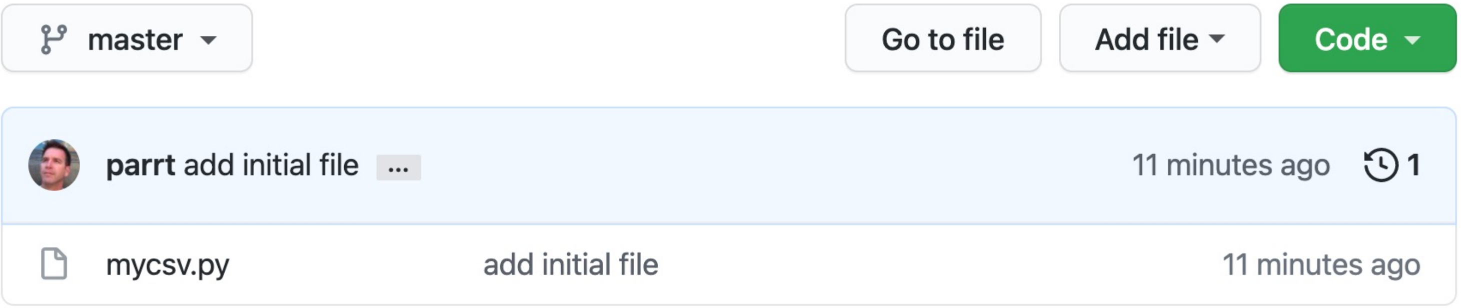


```
critter:~/classes/msds692/pipeline-parrt $         git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 440 bytes | 440.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:USF-MSDS692/pipeline-parrt.git
 * [new branch]      master -> master
```

Check github webpage for your repo

- Github repo page should show your new file





parrt add initial file ... 11 minutes ago 1

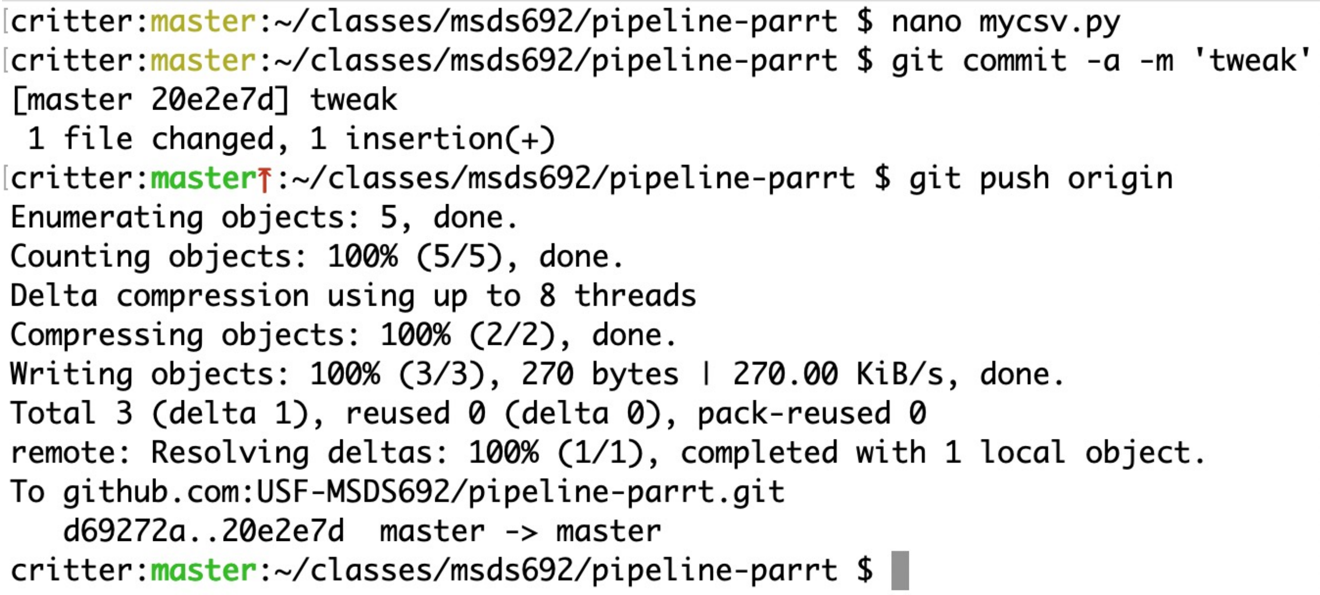
mycsv.py add initial file 11 minutes ago

Initial add/commit sequence summary

- Clone repo from github to a directory with same name on laptop
- Copy or create files in repository directory
- Add those files
- Commit those changes (add/edit/delete are all changes)
- Push back to the origin (github.com)

Making edits, mirroring on github

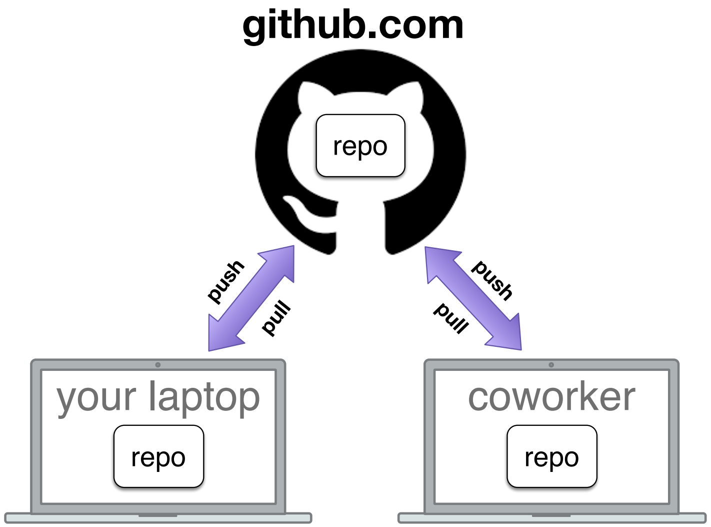
- During the normal course of software development, you will edit files and then commit these changes, pushing to github
- Here, I'm editing an Python file



```
[critter:master:~/classes/msds692/pipeline-parrt $ nano mycsv.py
[critter:master:~/classes/msds692/pipeline-parrt $ git commit -a -m 'tweak'
[master 20e2e7d] tweak
 1 file changed, 1 insertion(+)
[critter:master↑:~/classes/msds692/pipeline-parrt $ git push origin
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 bytes | 270.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:USF-MSDS692/pipeline-parrt.git
   d69272a..20e2e7d  master -> master
critter:master:~/classes/msds692/pipeline-parrt $ 
```

Pull in changes from github

- If there are changes pushed to github that you do not have in your laptop copy, you must pull in those changes with: **git pull origin main** (or just **git pull**)
- This happens when I have cloned and added grading results to your repository and pushed them back, or you are working with a partner on a project; both of you push/pull via same github repo



Miscellaneous but useful commands

- **git rm** *filename*
  Remove a file from the directory and from git repo tracking
- **git mv** *from_filename to_filename*
  Rename a file or directory managed by git
- **git reset --hard HEAD**
  Wipe out any changes you've made to managed files, resetting the repository to the most recent commit
- **git checkout --** *filename*
  Undo changes made to a single file managed by git, resetting to the state of that file at the most recent commit

Configuring username/email the first time

- The first time you try to push something back to github, I think the use of **git** from the command line will ask you to configure your name and email address using the "**git config**" command with a bunch of options
- Or, it looks like you can do this from github's webpages; see https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-email-preferences/setting-your-commit-email-address
- That might be easier because the command line will ask you to use **vi** or some other editor you are unfamiliar with

Warnings and recommendations

- **git** is ridiculously complicated and has a terrible interface in my opinion so proceed with caution, but it is the most commonly used!
- I recommend sticking with a few commands: clone/add/commit/push/pull/rm/mv
- Do NOT do branching/merging until you are much more comfortable with git and version control systems
- Anything beyond these simple commands, I avoid or use very carefully after reading the manual

Pitfalls

- Repo should be

  https://github.com/**USF-MSDS692**/pipeline-youruser

  NOT

  https://github.com/**youruser**/pipeline-youruser

- Don’t change your github username and expect me to notice

- Github is not a homework submission mechanism; you should be using it from the start of the project:
  - Clone and add your initial files to get started
  - As you finish a logical chunk of work, commit and push to github

Even more pitfalls

- Don’t use the github website to add/change, at least until you have more experience; too easy to get out of sync with your laptop
- Make sure that the github website reflects the contents of your laptop repository at the project due date
- Don’t put your project X code into the repository for Y; each project has its own repository
- Don’t create random subdirectories in your repository; for our purposes, you will be creating all files in the root of your repository directory