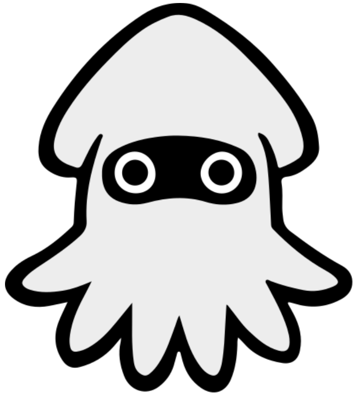
Git pt.2 & GitHub:

~~Squid~~ Game

Octocat

~~Doggo~~ Tax Squid



Announcements

Congrats on finishing the midterm!!

Extratation: Crash Course w/ ScottyLabs
(Registration link at the end of slides)

Review

Git: Version Control System

add, commit, branch, checkout

Let’s Play A Game

You just broke your sugar honeycomb!

Git Ready For...
Undoing Mistakes

Unstaged Changes
(before using git add)

- **Scenario**:
    - you're working on *trainerlab* and accidentally delete the *professor*
    - you haven't staged or committed since pulling the lab
    - you want to fix your sugar honeycomb
- **Use**:
    - `git checkout <file_name>`

Staged Changes
(after git add, before git commit)

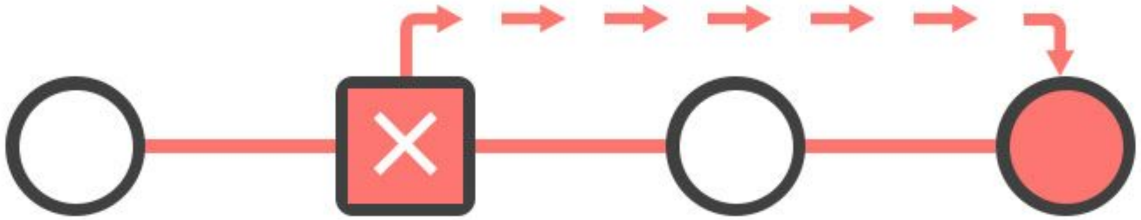
- **Scenario**:
  - you’re working on *sportslab* and accidentally delete a paragraph of *big-league.txt* and :wq
  - you finished other tasks and don’t want to redo them
  - you’ve staged everything

- **Unstage**: `git reset HEAD <file name>`
- **Save for later**: `git stash`
- **Retrieving the stash**:
  - `git stash list`
  - git stash apply stash@{*n*}

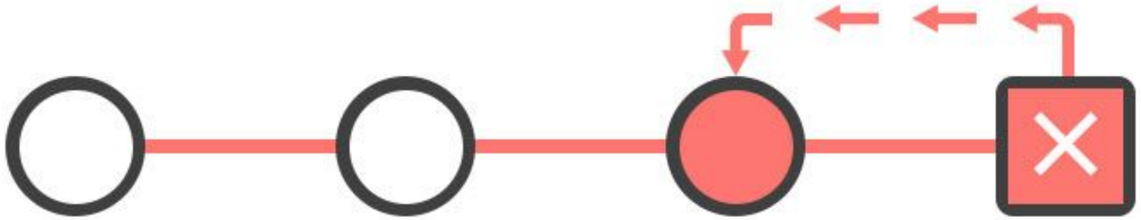
After committing

- **nuke** changes: git reset --hard origin/<branch, commit hash/HEAD~n>
    - *n* is the num of commits you want to go back
- **remove** commits:
    - git reset HEAD~n
    - git revert <commit hash>
- **revert vs reset**
    - striking out vs erasing
    - revert = **new commit** undoing past changes
        - past changes still in log
    - reset removes evidence of old changes

Reverting



Resetting



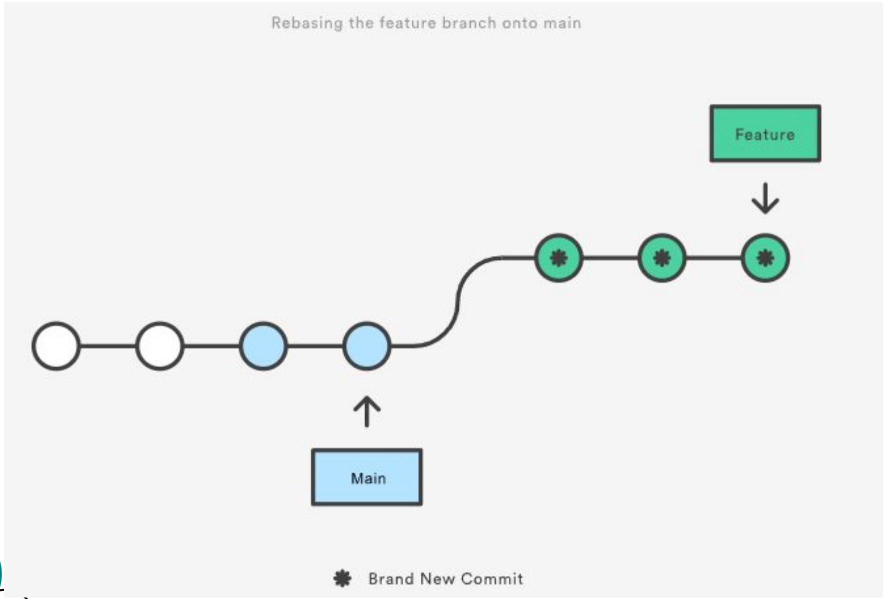
Rebasing

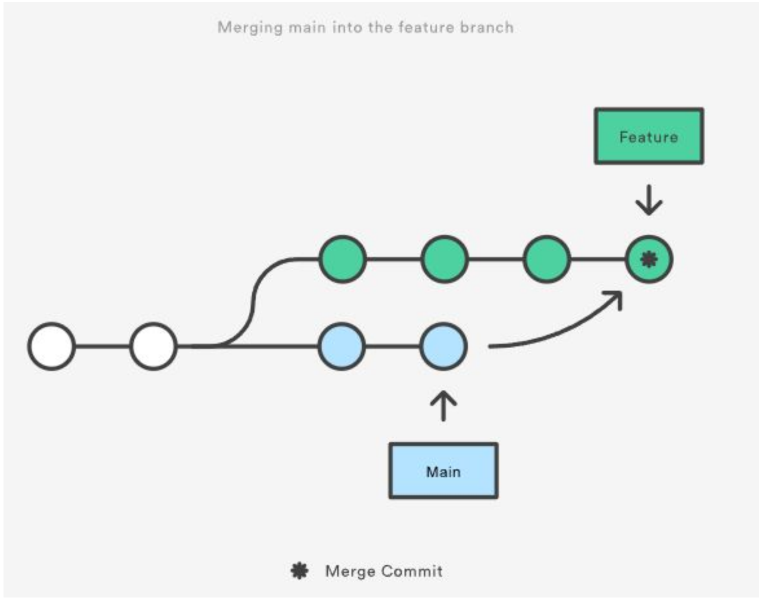
- **Rewrite** history!
    - git rebase <branch_to_base_on>
- Visual Demo time!

Rebase vs. Merge

- Both integrate changes from one branch into another
- git merge:
  - Creates a “merge commit” to tie together the history of the two branches
  - Existing branches are not changed
  - Generally safer!
- git rebase:
  - Moves the entire branch to the tip of current branch
  - Can result in a cleaner git log
  - Do NOT use on public branches!! (more on that with remotes later)

Rebase vs. Merge

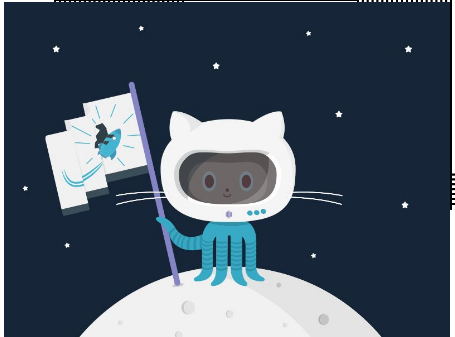




Remotes and Github

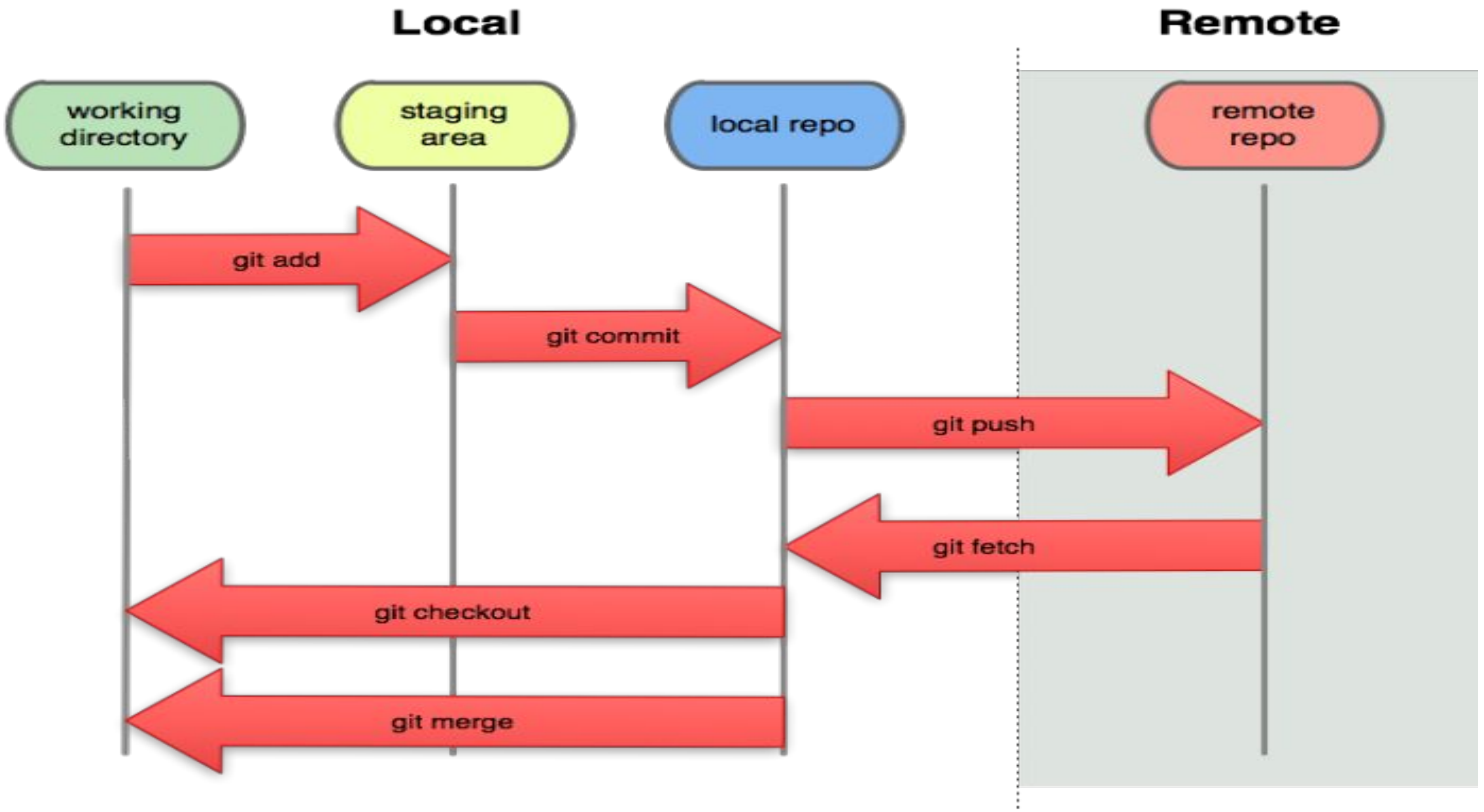
What is GitHub?

- Super useful tool for development!
- Lets you host "remotes" in the cloud
    - What's a remote? Next slide lol
- Development features:
    - Issues, code review tools, an ice vault in the Arctic Circle to save your code in the event of an apocalypse, etc.
- Great way to host and share open source projects
- Other ways to host remotes:
    - bitbucket (competitor to github)
    - host your own on your own servers



Remotes

- Remotes are “copies” of your repository stored in the cloud
  - Specifically versions of the git graph that have the same initial commit
  - **DEFAULT REMOTE NAME IS ORIGIN**
- ✅ Goal: use these copies to backup and store code, enable collaboration, deploy and manage code better
- ❌ Problem: maintaining consistency across these different versions

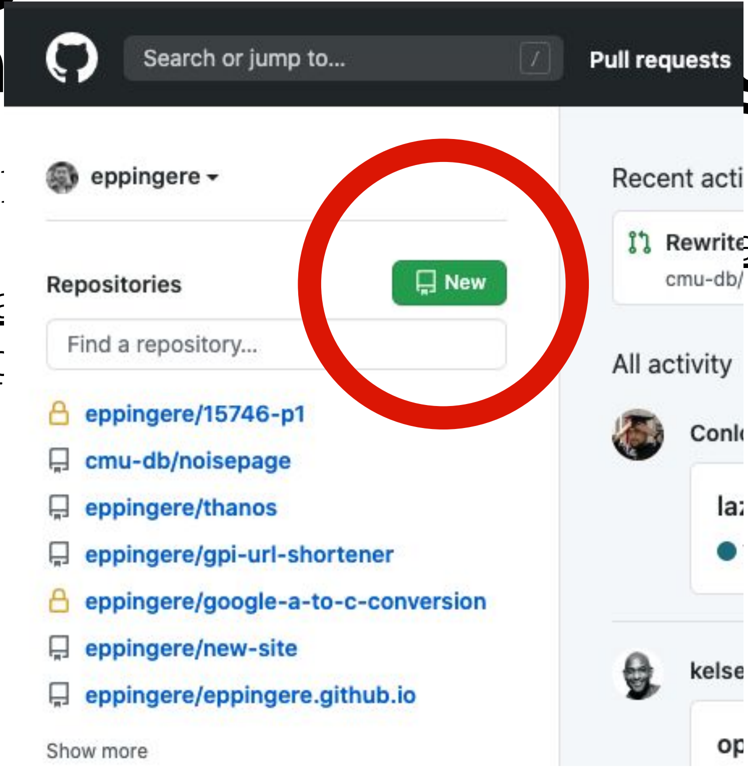

Local
Remote
working directory
staging area
local repo
remote repo
git add
git commit
git push
git fetch
git checkout
git merge

Lets get started with a Github Repository

- Step 0: make a GitHub account
  - While you're there, sign-up for the education program and git a tone of free stuff
- Make a repository using the gui (super easy)

Lets get started with a Gith y

- Step 0: ma ation
  progr
- Make a rep



eppingere

Repositories

New

Find a repository...

eppingere/15746-p1
cmu-db/noisepage
eppingere/thanos
eppingere/gpi-url-shortener
eppingere/google-a-to-c-conversion
eppingere/new-site
eppingere/eppingere.github.io

Show more

Recent acti

Rewrite
cmu-db/

All activity

Conl

la

kelse

op

Lets get started with a Github Repository

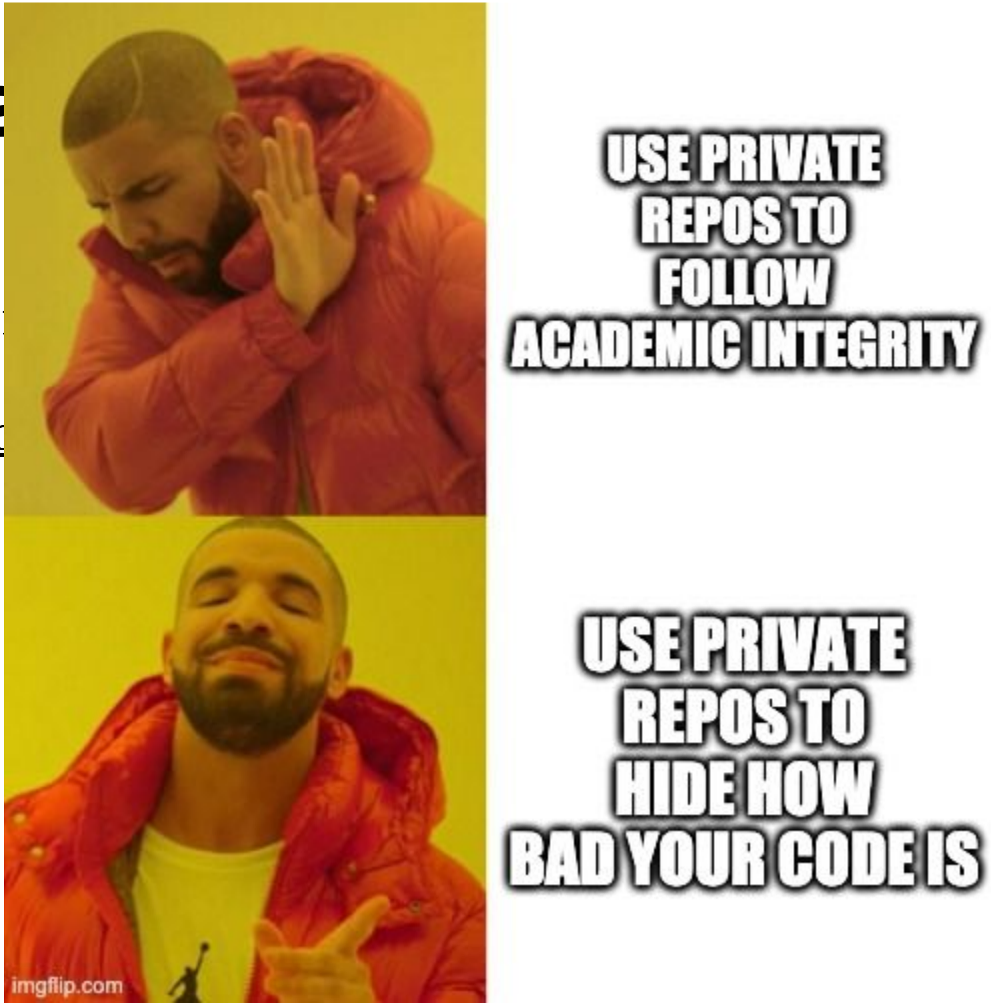
- Step 0: make a GitHub account
  - While you’re there, sign-up for the education program and git a tone of free stuff
- Make a repository using the gui (super easy)
  - Things to know about making repos
    - Public vs Private
      - Public to show of and flex on them recruiters
      - Private to be sneky and follow academic integrity

Lets
a Gi

- Step 0:

  pro

- Make a


USE PRIVATE REPOS TO FOLLOW ACADEMIC INTEGRITY
USE PRIVATE REPOS TO HIDE HOW BAD YOUR CODE IS
imgflip.com

Lets get started with a Github Repository

- More things to know about making your first repo
  - README.md
    - write-up about your code, instructions, things for collaborators to know
    - Written in markdown
  - .gitignore
    - Remember those? Github provides you with some starters

GitHub Licenses Explained

- If your code is public, what rights people have who use your code
- Common Licenses:
  - MIT License: very open and gives rights to everyone while protecting you from being sued if your code breaks something
  - Apache License (2.0): also very open, explicitly protects your code's intellectual property, gives you the right to any code someone contributes to your project in any form
  - GPL: notoriously restrictive license, copyrights the code in it and explicitly restricts how you are allowed to use the code

Ok what now?

- You now have a remote of your repo
- You want to have a local version of your repo
- Simply “clone the repository”
    - Click the “clone” button on your repo’s GitHub page
    - Copy link and run:
        - `$ git clone <clone url here>`

Wait, can u have multiple remotes tho?

Yes!!

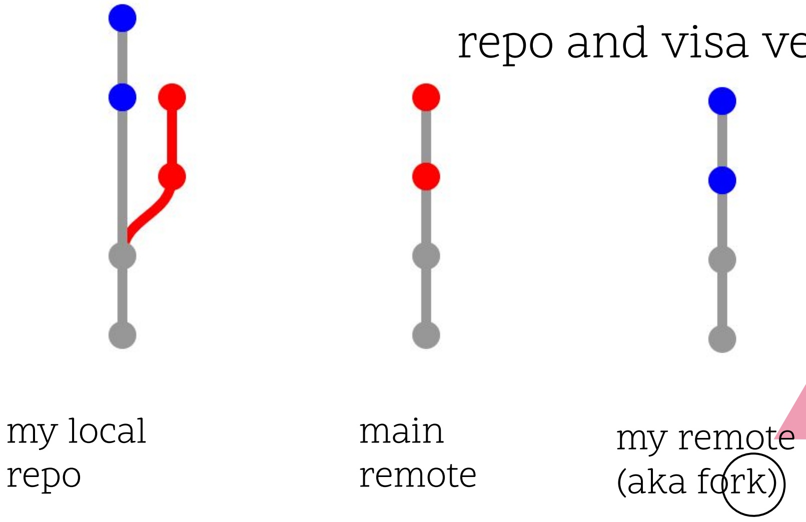
- Your local repo can have multiple remotes.
- To check all remotes, do: `$ git remote`
- When u clone, the default name of remote is origin
- Use `-v` flag if u want to see the URLs linked to the remotes
- Can also add a remote by:
  `$ git remote add <shortname> <url>`

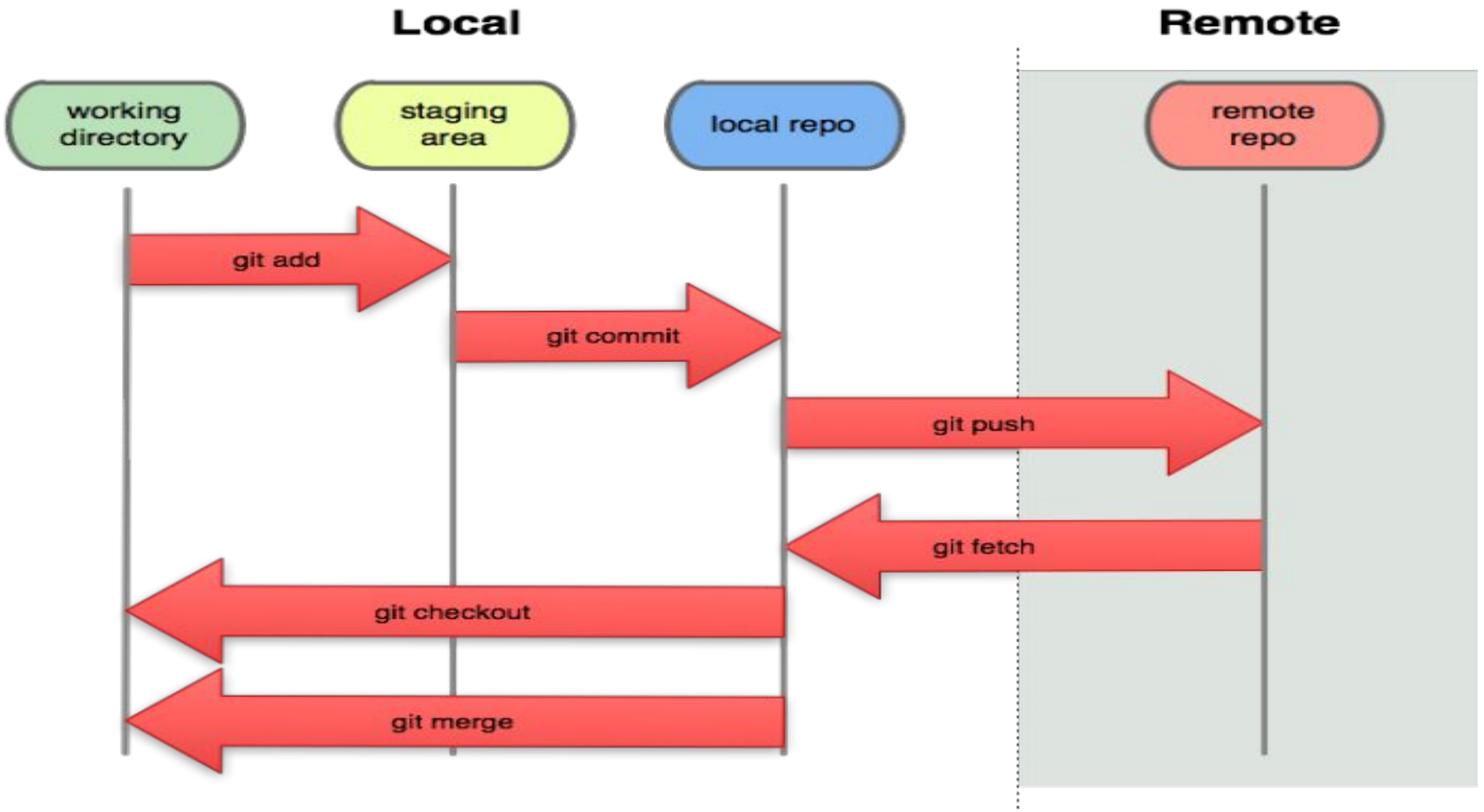
Ok enough riff raff let's Do this!!

- Two main actions to think about:
  - “push” changes from your local repository to the remote repository
  - “pull” or “fetch” changes from the remote to your local repository

Wait what???

- Remotes are just different versions of the git tree
- We want to move commits from remotes to our local repo and visa versa




Local
Remote
working directory
staging area
local repo
remote repo
git add
git commit
git push
git fetch
git checkout
git merge

Pushing Example

- I have some commits locally that I want to make sure are saved on GitHub. Must need write-access tho
  - run command:
    - $ git push <remote name> <remote branch>
  - Sometimes your local branch isn't on the remote:
    - $ git push --set-upstream <remote name> <branch name>
  - But you usually want to push your current branch to the remote's version of this branch
    - You can just run:
      - $ git push

Pulling Example

- I have some commits in the remote that I want locally
  - $ git pull <remote name> <local branch>
- But usually you can just run for default remote and current branch:
  - $ git pull

Git Fetch

- Allows you to see the changes in remote repo since your last pull.
- $ git fetch <remote>
- Useful if you’re not sure of pulling just yet from the remote and want to review
- Unlike Git Pull, Fetch doesn’t merge the remote repo with your local repo
- Git Pull = Git Fetch + Merge

It's time for spaghetti

- Git forks are duplicate remotes of another remote
- Fork allows you to have your OWN copy
- Why do we want forks?
  - You don't have write access to the og remote
  - You want one just for you to use and the main one is for your group
    - Everyone has their remotes and no one gets in each other's way

Lets be good internet citizens

- You now know everything to contribute to open source projects
- There are a ton of great projects on github
  - linux android the go programming language noise page vscode the GPI website so many more
- Simply fork the project, clone, do your thing
- Submit a pull request to the main project

Pull requests on the dL

- You want to add your changes to the og remote
- How?
- Submit a pull request (PR)
- Push your changes, go to og remote’s page, click “submit a pull request”
- The person who runs the repository can give you feedback and hopefully get your code merged into a really cool project

Announcements

- **Please give feedback :)**
  http://tinyurl.com/f21-gpi-feedback
- **Instructions for the GitHub part of lab:**
  - https://www.cs.cmu.edu/~07131/f21/topics/readings/week-8/

- **Extratation : Crash Course w/ ScottyLabs**
  Registration:
  https://docs.google.com/forms/d/e/1FAIpQLScTH_5m0qdmBUKNfF97Cgleu_KGS87CUlWErQQe2zYtub_Pwg/viewform