Git and Github

Practical Data Science

@jattenberg

May 2014

What is it?

- Git: a modern library for version control
- Github: a free* website for hosting git repositories.

*for open source code

Huh?

- Track all changes to code (or any other text). Maintain records of work done
- Revert prior versions in the event of a bug
- Remote hosting: resiliency, access from multiple computers, or to multiple users
- Merge changes from multiple, simultaneous collaborators

Huh?

- Share your work with the world
- Host ipython notebooks (to be explained soon)
- Fork many open source projects, tweak and contribute.
- Make gists: a quick and easy way to share notes / error messages / code
- An online portfolio of your work. Great resume component!
- Extremely popular in practice

But...

- Git is extremely complicated with a ton of power features.
- In this class we're just going to be scratching the surface.
- Things can go wrong: at some point, you'll find yourself in "git hell". Keep your cool, move carefully, and google is your friend.

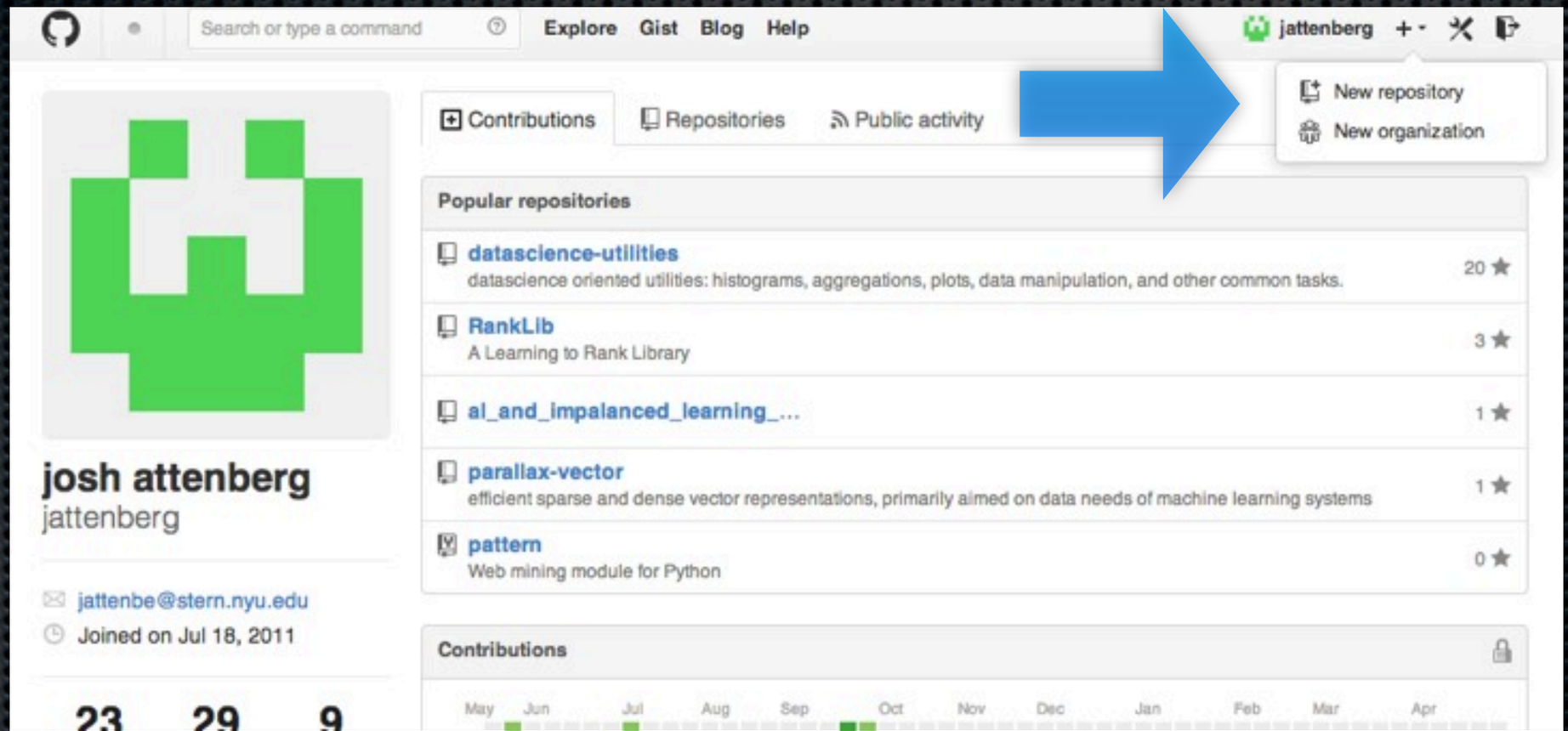
Coding (usually)

- A directory corresponding to a particular project
- A structure of sub-directories for different programming languages, tasks, packages, tests, etc
- Source code: text files
- Other stuff: assets, documentation, configs, notes. Often much of this is text too.

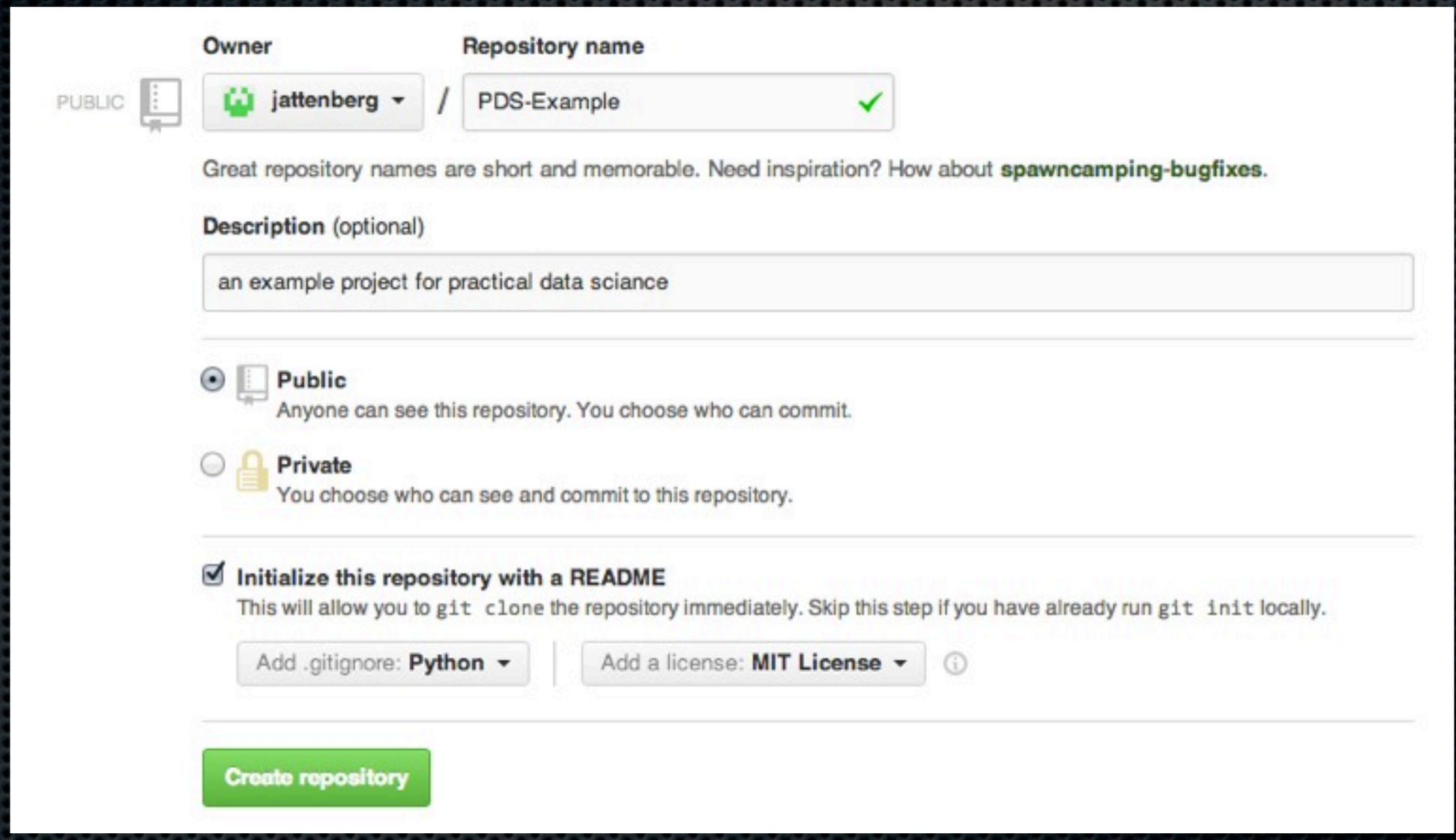
Coding with git (usually)

- An extra hidden directory (.git) in the project root directory.
- This contains version info for (almost) everything in the project
- Occasionally, work is backed up to a remote git repository (eg, github)
- a .gitignore file tells git about any files it shouldn't version control

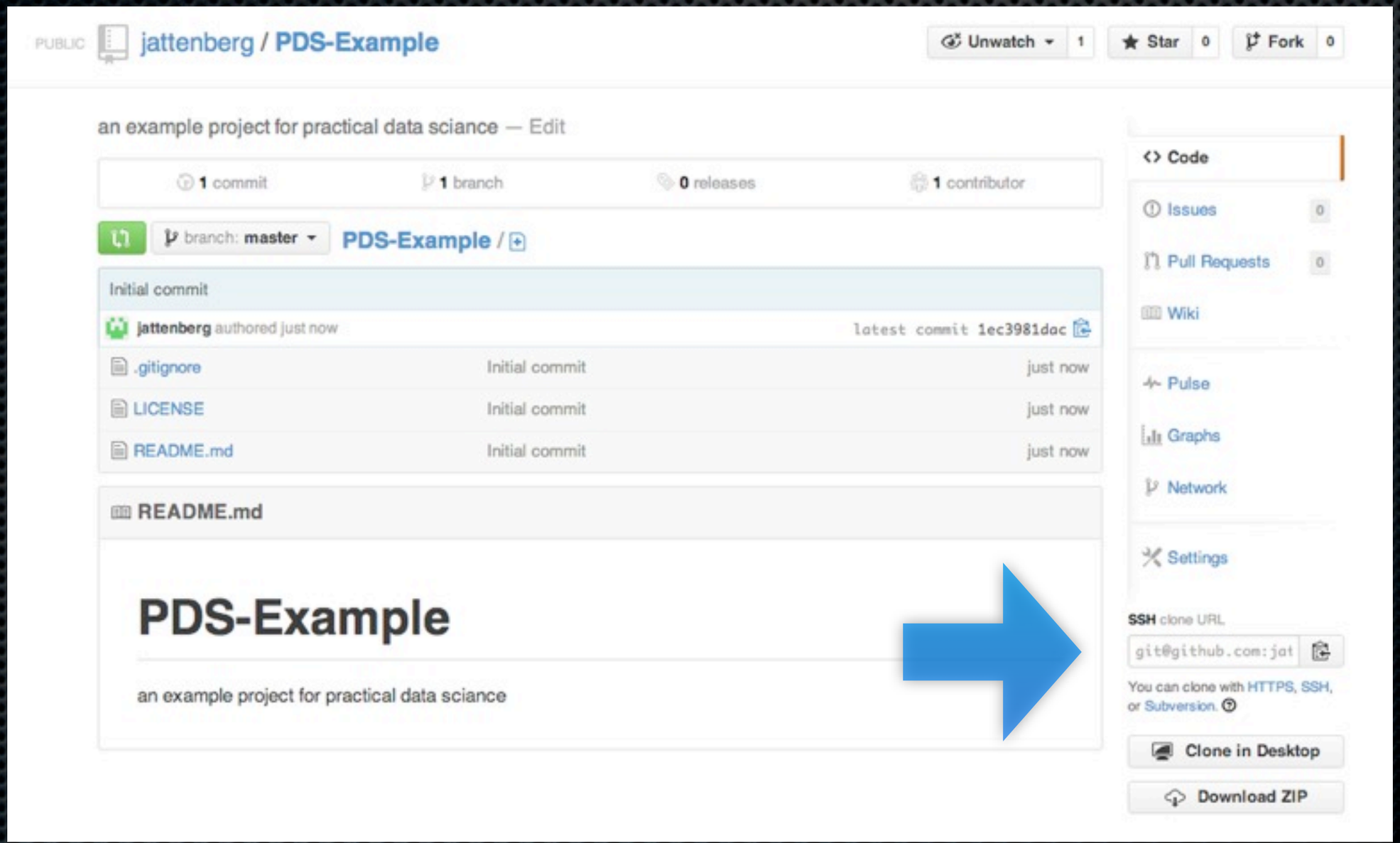
Starting a new project



Starting a new project



Starting a new project



Starting a new project

- In your terminal, navigate to where you want the project to reside, then:
- git clone [ssh clone URL from github]
- This creates a new folder for the project, you can cd into this and get to work

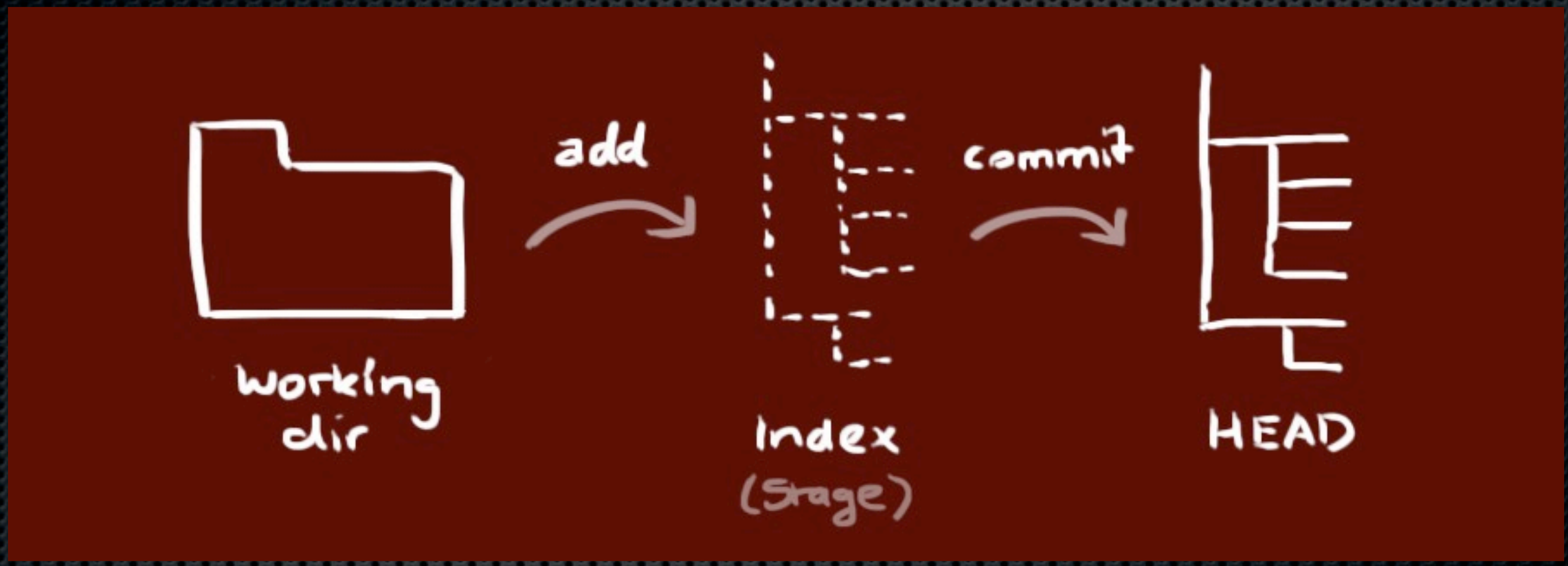
Git Basics

- Three “trees” of files maintained by git
  - working dir: the actual files as they are on your file system
  - index: staging area: files that are “known” by git
  - HEAD: versioned work: the last commit you’ve made

Git Basics

- To add to the working dir, just create files or folders.
- To move to index: git add [files to be added]
- To move to HEAD from index: git commit [files] -m “committing”

Git Basics



Git Basics

- To send changes from the HEAD of your local repository to the remote repository (usually on github):
  - git push origin master
  - origin: the name of the server you're pushing to
  - master: the name of the branch you're pushing to (more on this soon)

Git Basics

- Git is a great tool for collaborative coding. In this setting, multiple developers are pushing to master.
- Git tries to ensure that the remote master version is consistent- you need to merge your changes the most recent version of the code
- If others have made changes to master, pull them down and apply your work: git pull --rebase (git rpull)

Git Basics

- If there are any conflicts (eg, when you and someone else have touched the same part of the source), the pull and rebase will fail, notifying you there's been a conflict
- Edit the files to make sure the file is in the desired state.
- then: git rebase --continue
- Once you've got the most recent changes on your HEAD, you can push to master

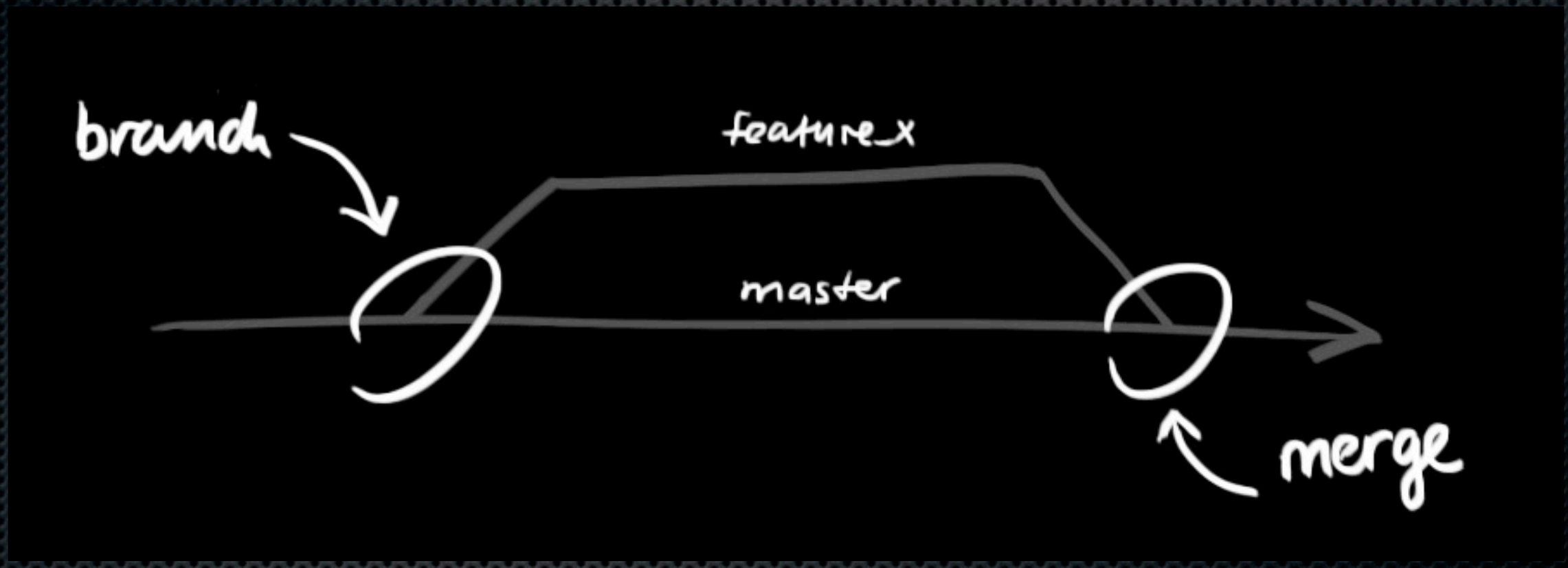
Git Basics

- To discard a local changes: git checkout [file with the change]
- To unstage a commit to HEAD: git reset HEAD^

Git Basics

- Some times a feature requires a lot of work that should be isolated from master until it's ready, but remote storage, versioning, and collaboration are still desired.
- This is what git branches are for.
- Branches are used to develop features in isolation from each other. Master is the default branch. Create and use other branches for development, then merge them back to master if you're ready.

Git Branches



Git Branches

- Create a new branch named “test_idea” and switch to it locally: git checkout -b test_idea
- switch back to master: git checkout master
- delete a branch: git branch -d test_idea
- push a branch to origin (make it available to others): git push origin test_idea
- start working on a branch that someone else has developed: (after rpull) git checkout their_branch

Git Branches

- To merge another branch into the branch that's active: git merge test_idea (how to move your changes into master) resolve conflicts as when rpulling.

- Look at differences between local branch index and HEAD: git diff
- Look at differences between local branch and origin master: git diff origin/master
- Look at differences between branches: git diff test_idea other_branch

Moving an existing project to git

- navigate to project folder in the terminal
- git init
- git add *
- git commit -m “first commit”
- git remote add origin git@github.com:[username]/[project-name].git
- git push -u origin master

More on git:

- http://rogerdudler.github.io/git-guide/
- http://www.dataschool.io/git-and-github-videos-for-beginners/