Agenda

- Opening: Review the command line pre-class exercise (code) - 6:30 15 min
- Slack tour - 6:45 10 min
- Git and GitHub
  - Intro to new material - 6:55 20 min
  - Connecting Git with GitHub – clone 7:15 15min
  - Making local changes then syncing and exercise - 7:30 20 min
    - Exercise (in pairs) - 7:50 10 min
- Break - 8:00 5min
- Git and Github continued
  - Updating a Local Repo with updates from GitHub - 8:05 15 min
    - Exercise - 8:20 10 min
- Intermediate command line – 8:30 30 min
- Closure and exit ticket submittal 9:00 - 15 min
- Q&A – 9:30 15min

Feedback themes

Oct 27

- We spent a goodly amount of time on logistics and initial concepts (a necessary evil?)
- Everyone is eager to get to hands-on and dig into deeper content
- Some uncertainty about project scope and schedule

Git and GitHub, an intro

General Assembly – Data Science

- Introduction
- Exploring GitHub
- Using Git with GitHub
- Updating a Local Repo from GitHub
- Assorted Tips

Introduction

Version control, Git and GitHub

Why learn version control?

- Useful to have a system for tracking and storing versions of your work
- Especially useful when you write code but not restricted to just code
  - an be used with all kinds of documents
- Enables teams to easily collaborate on the same codebase
- Enables you to contribute to open source projects
- Attractive skill for employment

What is Git?

- A version control system that allows you to track files and file changes in a repository ("repo")
- Primarily used by software developers
- Most widely used version control system
  - Alternatives: Mercurial, Subversion, CVS
- Runs from the command line (usually)
- Can be used alone or in a team

We will directly interact with Git on our laptops

- Think of it as the program on your laptop that provides a *local* repository that keeps track of and stores versions our *local files*

What is GitHub?

- A website, not a version control system
- Allows you to put your Git repos online
  - Largest code host in the world
  - Alternative: Bitbucket
- Benefits of GitHub:
  - Backup of your files, revisions and revision history
    - from your local laptop git repository to the web
  - Visual interface for navigating repos
  - Makes repo collaboration easy

Note: Git does not require GitHub

Some challenging when learning Git

- Designed (by programmers) for power and flexibility over simplicity
- Hard initially to know if what you did was right
- Hard to explore functionality since there are no “undue buttons”
  - However since each revision of a file is saved you can typically go back to an earlier version even if it not obvious
- We'll focus on the most important parts we need for our purposes

Exploring GitHub

Navigating a GitHub repo (1 of 2)

- Example repo: github.com/JamesByers/GA-SEA-DAT1
- Account name, repo name, description
- Folder structure
- Viewing files:
  - Rendered view (with syntax highlighting)
  - Raw view
- README.md:
  - Describes a repo
  - Automatically displayed
  - Written in Markdown

Navigating a GitHub repo (2 of 2)

- Commits:
  - One or more changes to one or more files
  - Revision highlighting
  - Commit comments are required
  - Most recent commit comment shown by filename
- Profile page

Creating a repo on GitHub

- Click "Create New" (plus sign):
  - Define name, description, public or private
  - Initialize with README (if you're going to clone)
- Notes:
  - Nothing has happened to your local computer
  - This was done on GitHub, but GitHub used Git to add the README.md file

Basic Markdown

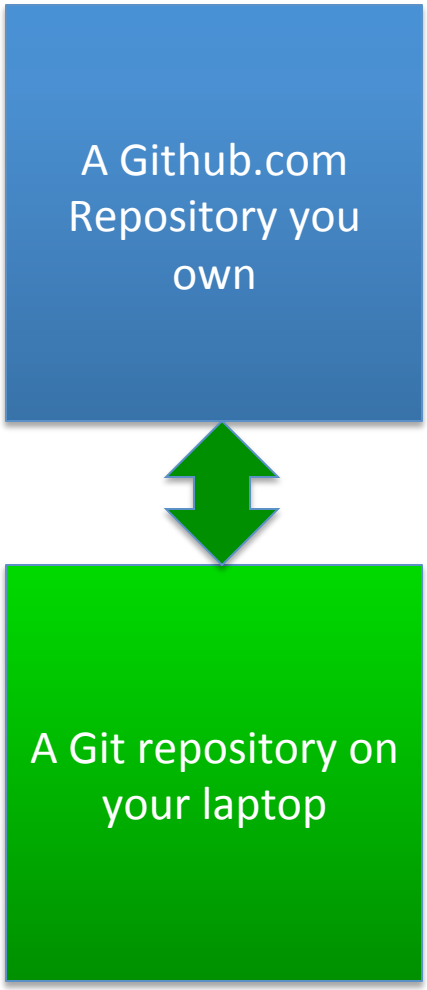
- Easy-to-read, easy-to-write markup language
- Usually (always?) rendered as HTML
- Many implementations (aka "flavors")
- Let's edit README.md using GitHub!
- Common syntax:
  - ## Header size 2
  - *italics* and **bold**
  - [link to GitHub](https://github.com)
  - * bullet
  - `inline code` and ```code blocks```
- Valid HTML can also be used within Markdown

Using Git with GitHub - Clone

In the next few minutes you will:

- Copy your new GitHub repo to your computer
- Make some file changes locally
- Save those changes locally ("commit" them)
- Update your GitHub repo with those changes

How we will connect our local Git to Github



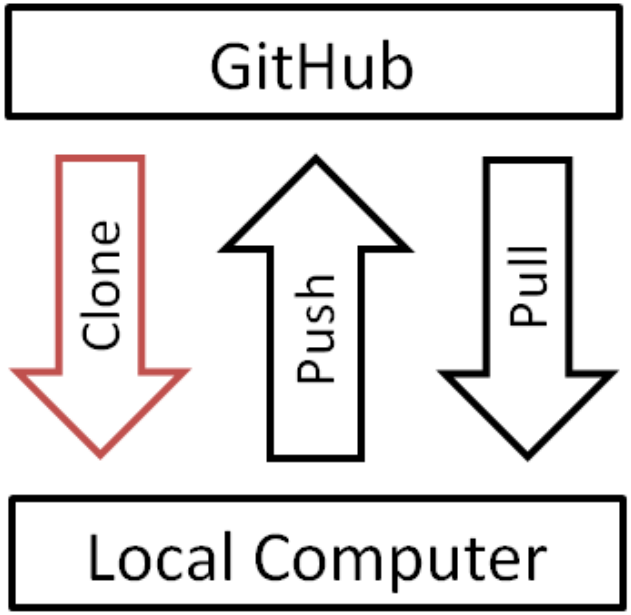
Create a master repository on GitHub that you own

- It will provide:
  - A visual interface for navigating the repo
    - You can move, delete files an edit text files
  - A backup on the web when you manually sync from your laptop
    - GitHub is not automatically synced with your laptop files
    - You must manually initiate the synchs from your laptop

Clone your GitHub repo to a directory on your laptop

- Adds a .git directory into that local folder
  - Contains all the information that the git commands need to properly manage the local repo
  - Contain all the information that git commands need to allow you to manually synch your laptop directory (and subdirectory) files with your GitHub repo
- You will initiate all synchs using Git commands on your laptop

GitHub workflow diagram



- You will use clone only once for each local version of the GitHub repository. But you will push and pull often to synch changes between your local files and GitHub

- Note that there is no Git "system" running on your laptop. Rather all info about each local repo is stored in the .git directory in each local repo. You make things happen using git commands on your laptop

Cloning your new GitHub repo

- Cloning = copying to your local computer
  - Like copying your Dropbox files to a new machine
- First, change your working directory to where you want the repo to be stored locally: cd
- Then, clone the repo: git clone <URL>
  - Get URL from GitHub (ends in .git)
  - Clones to a subdirectory of the working directory
  - No visual feedback when you type your password
- Navigate to the repo (cd), then list the files (ls)

Checking your remotes

- A "remote alias" is a reference to a repo not on your local computer
  - Like a connection to your Dropbox account
- View remotes: git remote -v
- "origin" remote was set up by "git clone"
- Note: Remotes are repo-specific
  - If you are in another local repo the same commands are only working on that repo and using the information in that repo's .git directory

Using Git with GitHub
Making local changes then syncing

Making changes, checking your status

- Making changes:
  - Modify README.md in any text editor
  - Create a new file: touch <filename>
- Check your status:
  - git status
- File statuses (possibly color-coded):
  - Untracked (red)
  - Tracked and modified (red)
  - Staged for committing (green)
  - Committed

Staging and committing changes

- Stage changes for committing:
  - Add a single file: git add <filename>
  - Add all "red" files: git add –A
    - “git add .” also works
- Check your status:
  - git status
  - Red files have turned green
- Commit changes:
  - git commit -m "message about commit"
- Check your status again!
- Check the log: git log

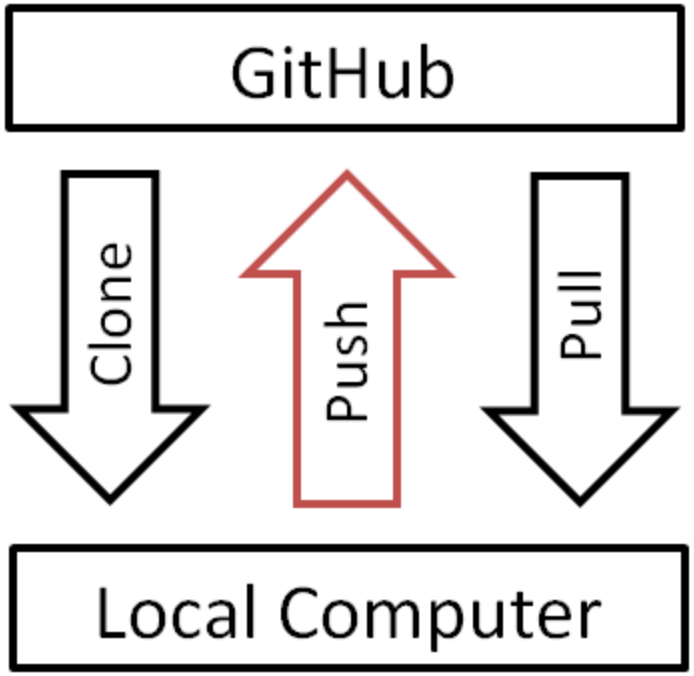
Exiting Vim

- Always include a message with your commit: git commit -m "message"
- You will be taken to a text editor if you try to commit without a message: git commit
- Default text editor is often Vim
- How to exit Vim:
  - Hit Esc key (to enter command mode)
  - Type :q! (to quit without saving)
  - Hit Enter key

Pushing changes to GitHub

- Everything you've done to your cloned repo (so far) has been local
- You've been working in the "master" branch
- Push committed changes to GitHub:
  - First time: git push <remote> <branch>
    - Typically: git push origin master
  - Later you will just need to git push since <master> and <branch> got stored in files in the repo's local .git files
- Refresh your GitHub repo to check!

GitHub workflow diagram



Quick recap of what you've done

- Created a repo on GitHub
- Cloned repo to your local computer (git clone)
  - Automatically sets up your "origin" remote
- Made two file changes
- Staged changes for committing (git add)
- Committed changes (git commit)
- Pushed changes to GitHub (git push)
- Inspected along the way (git remote, git status, git log)

Exercise: Let's do it again!

- Modify or add a file, then git status
- git add <filename>, then git status
- git commit -m "message"
- git push origin master
- Refresh your GitHub repo

Updating a Local Repo with updates from GitHub

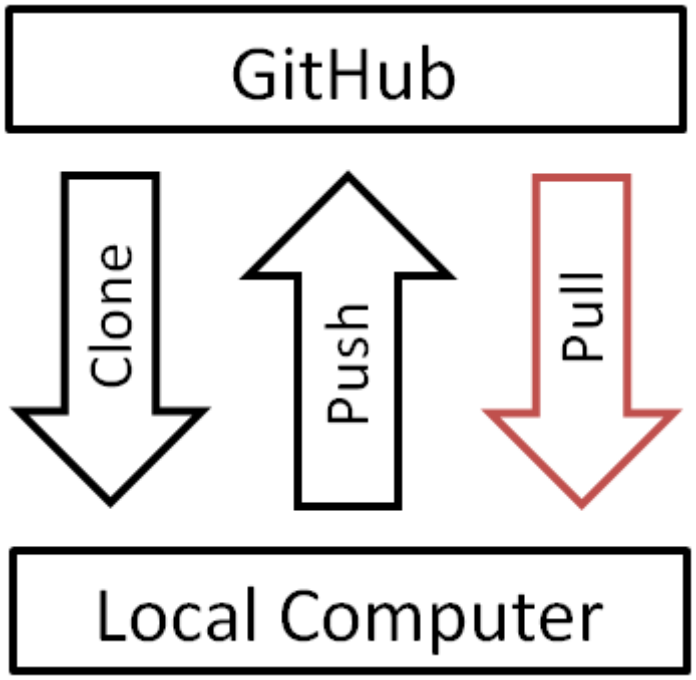
Changes being made remotely

- So far, repo changes have been made on your local machine and then pushed to GitHub
- What if you clone someone else's GitHub repo, and then they make changes to it?
- Git does not automatically update your local repository with remote changes

Pulling changes from GitHub

- Git allows you to manually "pull" changes from remote locations
  - Like syncing your local files from Dropbox
  - git pull <remote> <branch>
    - Often the first pull will be: git pull origin master

GitHub workflow diagram



Practice pulling from GA-SEA-DAT1

- Navigate to the directory where you want the local repo created
  - Choose a directory location where you can easily find it
  - Don't store it inside another Git repo
- Clone the GA-SEA-DAT1 repo:
  - git clone https://github.com/JamesByers/GA-SEA-DAT1.git
  - You cannot affect the GA-SEA-DAT1 repo because I own the master and you do not know my login password. You automatically control your GitHub repository the same way
- Navigate to the repo (cd), then list the files (ls)
- git pull origin master
- I'll push a new change to GA-SEA-DAT1
- Again: git pull origin master

When is pulling necessary?

- Pulling is only necessary when changes have been made remotely but not locally
- Most common scenario: repo is owned by someone else
- Also common: you make changes to the same repo from multiple computers
- Good habit to pull before you start working
  - No harm is done by pulling from a repo that hasn't changed

Merge conflicts

- The most common problem when pulling is a "merge conflict": there is a conflict between the changes being merged and changes you have made locally
- How to avoid merge conflicts:
  - If you want to edit GA-SEA-DAT1 files, make copies and edit the copies instead
- How to resolve a merge conflict:
  - Discard your changes: git checkout -- <filename>
  - Then try pulling again

Assorted Tips

Deleting or moving a repo

- Deleting a GitHub repo:
  - Settings, then Delete
- Deleting a local repo:
  - Just delete the folder!
- Moving a local repo:
  - Just move the folder!

Gists: lightweight repos

- You have access to Gist: gist.github.com
- Can include one or more files
- Useful for snippets, homework submissions
- Can be public or secret (not private)
- Let's create one right now!
- Sharing the correct URL for a Gist
- Supports online editing, cloning, committing, commenting, embedding, etc.

Excluding files from a repo

- Create a ".gitignore" file in your repo:
  touch .gitignore
- Specify exclusions, one per line:
  - Single files: pip-log.txt
  - All files with a matching extension: *.pyc
  - Directories: env/
- Templates: github.com/github/gitignore

Two ways to initialize Git

- Initialize on GitHub: (we did it this way!)
  - Create a repo on GitHub (with a README)
  - Clone to your local machine
  - This is what we did today (and what I recommend)
- Initialize locally:
  - Initialize Git in an existing local directory: git init
  - Create a repo on GitHub (without a README)
  - Add remote: git remote add origin <URL>