GENERAL ASSEMBLY

DATA SCIENCE
11 WEEK PART TIME COURSE

Week 1 Lab - Git

- Introduction
- Exploring GitHub
- Using Git with GitHub
- Contributing on GitHub
- Bonus Content

- Version control is useful when you write code, and data scientists write code
- Enables teams to easily collaborate on the same codebase
- Enables you to contribute to open source projects
- Attractive skill for employment

WHAT IS GIT?

- 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

- Allows you to put your Git repos online
- Largest code host in the world
- Alternative: Bitbucket
- Benefits of GitHub:
  - Backup of files
  - Visual interface for navigating repos
  - Makes repo collaboration easy

Git does not require GitHub

- Designed (by programmers) for power and flexibility over simplicity
- Hard to know if what you did was right
- Hard to explore since most actions are “permanent” (in a sense) and can have serious consequences
- We’ll focus on the most important 10% of Git

- Create an account at github.com
- There's nothing to install

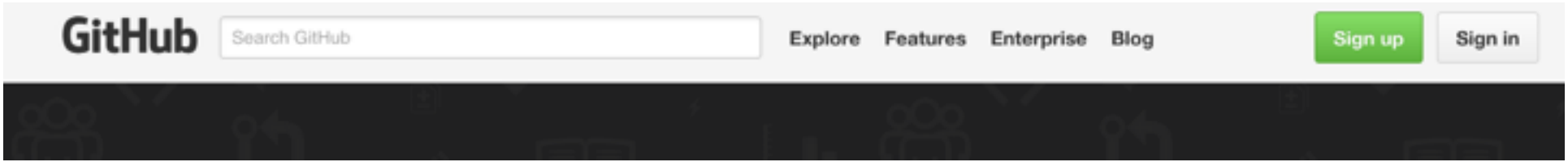
  "GitHub for Windows" & "GitHub for Mac" are GUI clients

  (alternatives to command line)

NAVIGATING A GITHUB REPO (1 of 2) 8

- Example repo: https://github.com/ihansel/SYD_DAT_3
- 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

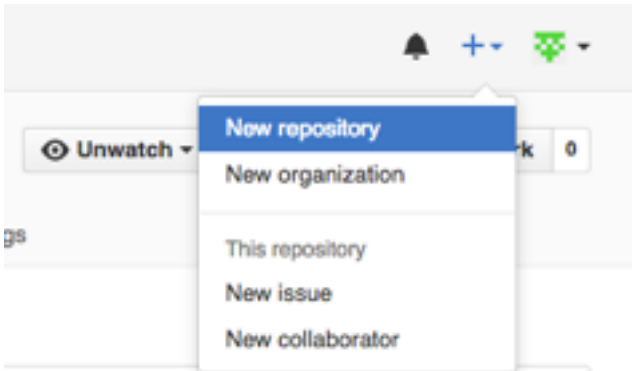
- Commits:
  - One or more changes to one or more files
  - Revision highlighting
  - Commit comments are required
  - Most recent commit comment shown by filename



- Click on the signup button on the top-right
- Choose a plan (one of them is free)
- Remember your email and password!!!!

- Click “Create New” (plus sign) on your profile:
  - Define name, description, public or private
  - Initialise 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



- Easy-to-read, easy-to-write markup language
- Valid HTML can also be used within Markdown
- 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```

- Installation: goo.gl/MJXSXp
- Open Git Bash (Windows) or Terminal (Mac/Linux):

```
git config --global user.name “YOUR FULL NAME”
git config --global user.email “YOUR EMAIL”
```

- Use the same email address you used with your GitHub account
- Generate SSH keys (optional): goo.gl/xtH0jJ
- More secure that HTTPS
- Only necessary if HTTPS doesn’t work for you

- Copy your new GitHub repo to your computer - **`clone`**
- Make some file changes locally
- Save those changes locally - **`commit`**
- Update your GitHub repo with those changes - **`push`**

- 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 you created to be stored: cd
- Then, clone the repo: **`git clone <URL>`**
- Get HTTPS or SSH URL from your 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)

- 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

- 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

- Stage changes for committing:
  - Add a single file: **`git add <filename>`**
  - Add all “red” files: **`git add .`**
- Check your status:
- Red files have turned green
- Commit changes:

```
git commit –m “message about commit”
```

- Check your status again!
- Check the log: git log

- 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`**