

# Introduction to Git and GitHub

General Assembly – Data Science

# Agenda

- I. Introduction
- II. Exploring GitHub
- III. Using Git with GitHub
- IV. Contributing on GitHub
- V. Bonus Content

# I. Introduction

# Why learn version control?

- 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

# 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 files
  - Visual interface for navigating repos
  - Makes repo collaboration easy
- “GitHub is just Dropbox for Git”
- Note: Git does not require GitHub

# Git can be challenging to learn

- 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

## II. Exploring GitHub



# GitHub setup

- Create an account at [github.com](https://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)

- Example repo:  
[http://github.com/sinanuozdemir/SF DAT 15](http://github.com/sinanuozdemir/SF_DAT_15)
- 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 profile

- Click on the signup button on the top-right
- Choose a plan (one of them is free)

# Creating a repo on GitHub

- Click “Create New” (plus sign) on your profile:
  - 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

# III. Using Git with GitHub

# Git installation and setup

- Installation: [tiny.cc/installgit](https://tiny.cc/installgit)
- 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): [tiny.cc/gitssh](https://tiny.cc/gitssh)
  - More secure than HTTPS
  - Only necessary if HTTPS doesn't work for you



# Preview of what you're about to do

- 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

# Cloning a 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 you created to be stored: `cd`
- Then, clone the repo: `git clone <URL>`
  - Get HTTPS or SSH 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

# 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

# Committing changes

- 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`

# Pushing 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:
  - Like syncing local file changes to Dropbox
  - `git push <remote> <branch>`
  - Often: `git push origin master`
- Refresh your GitHub repo to check!

# 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**)

# Let's do it again!

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



## IV. Bonus Content

# Two ways to initialize Git

- Initialize on GitHub:
  - Create a repo on GitHub (with README)
  - Clone to your local machine
- Initialize locally:
  - Initialize Git in existing local directory: `git init`
  - Create a repo on GitHub (without README)
  - Add remote: `git remote add origin <URL>`

# 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!

# 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](https://github.com/github/gitignore)

# Gists: lightweight repos

- You have access to Gist: [gist.github.com](https://gist.github.com)
- Add one or more files
- Supports cloning, forking, commenting, committing
- Can be public or secret (not private)
- Useful for snippets, embedding, IPython nbviewer, etc.

# Useful to learn next

- Working with branches
- Rolling back changes
- Resolving merge conflicts
- Fixing LF/CRLF issues