

# 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

- Example repo: [https://github.com/misrab/SG\\_DAT1](https://github.com/misrab/SG_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

# 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 (“clone”) 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 (“push”)

# 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

# Lab -- Before you leave

- Install [Git](#) on your machine
- Make a [Github](#) Profile on the web
- Create your own repo, call it “SG\_DAT\_1” and clone it to your machine
- Clone the [class 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)