Introduction to Git and GitHub

General Assembly – Data Science

Agenda

I. Introduction
II. Exploring GitHub
III. Using Git with GitHub
IV. Updating a Local Repo from GitHub
V. Assorted Tips

I. Introduction

What is Git?

- *Version control system*: tracks 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?

- *Website:* upload your Git repos
  - Largest code host in the world
  - Alternative: Bitbucket
- Benefits of GitHub:
  - Backup of files
  - Visual interface for navigating repos and files
  - Makes repo collaboration easy
- "Social network"
- Note: Git does *not* require GitHub, but GitHub uses Git

Why learn version control?

- Useful to have a formal system for tracking different versions of your work
- Especially useful when you write code
- Enables teams to easily collaborate on the same codebase
- Enables you to contribute to open source projects
- Attractive skill for employment

Git can be challenging to learn

- Designed (by programmers) for power and flexibility over simplicity
- New terminology
- 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 subset of features

Terminology

- *Clone*: copy repo to your computer
- *Fork*: copy someone's repo into your GitHub account
- *Commit*: save changes to history in Git
  - *Commit message* (required)
- *Pull request:* request repo owner to add your changes
- *Push:* upload your Git repo to GitHub
- *Branch:* series of changes to repo
  - *Master branch* (main)
- *Merge:* incorporate changes from one branch into another

II. Exploring GitHub

Navigating a GitHub repo (1 of 2)

- Example repo: github.com/vybstat/dat9
- 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 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

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 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 within 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

Staging and committing changes

- Stage changes for committing:
  - Add a single file: git add <filename>
  - Add all “red” files: git add -A
- 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 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:
  - 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 <filename>, then git status
- git commit -m “message”
- git push origin master
- Refresh your GitHub repo

IV. Updating a Local Repo 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: git pull origin master
- Let’s practice with the DAT9 repo!

Practice pulling from DAT9

- Clone the DAT9 repo (if you haven't already)
  - Store it where you can find it easily
  - Don't store it inside another Git repo
- Navigate to the repo (cd), then list the files (ls)
- git pull origin master
- I'll push a new change to DAT9
- 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

V. 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:
  - 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>