

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

II. Exploring GitHub

GitHub setup

- 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)

- Example repo: github.com/sinanuozdemir/DAT6
- 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

Basic Markdown

- Easy-to-read, easy-to-write markup language
- Usually (always?) rendered as HTML
- Many implementations (aka “flavors”)
- 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
- 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
 - More secure than HTTPS
 - Only necessary if HTTPS doesn't work for you

Some basic terminal movement

- Think of these like using the Finder/Explorer with words
- `cd “//”` (change directory) move into a folder
- `cd ..` will move up one directory
- `pwd` (print working directory) will show you where you are

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 status

- 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 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 changes

- Push changes that have been committed:
 - Add a single file: `git push origin master`
 - Check on github to see if they have been changed!

IV. Contributing on GitHub

Forking a repo on GitHub

- What is forking?
 - Copy a repo to your account (including history)
 - Does not stay in sync with the “upstream”
 - Do it! github.com/sinanuozdemir/DAT6-students
- Why fork?
 - You want to make modifications
 - You want to contribute to the upstream
- Clone your fork: **git clone <your URL>**
 - Don't clone inside your other local repo



Unwatch ▾

1

★ Star

0

Fork

0

Website

Website for this repository (optional)

Save

or [Cancel](#)

0 releases

1 contributor



latest commit e5df30e39e

commit

5 days ago

Code

Issues

0

Pull Requests

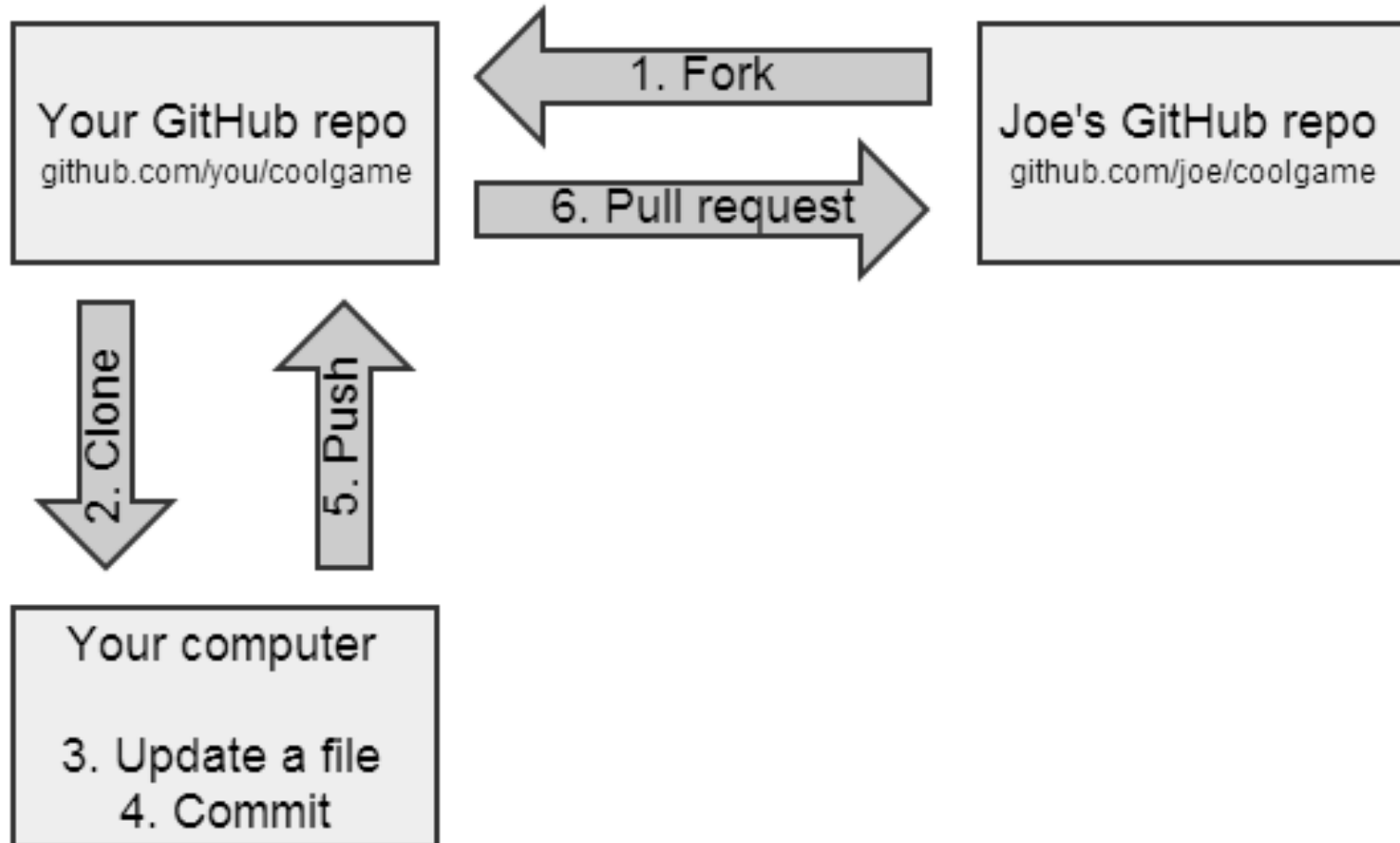
0

Wiki

Pulse

Graphs

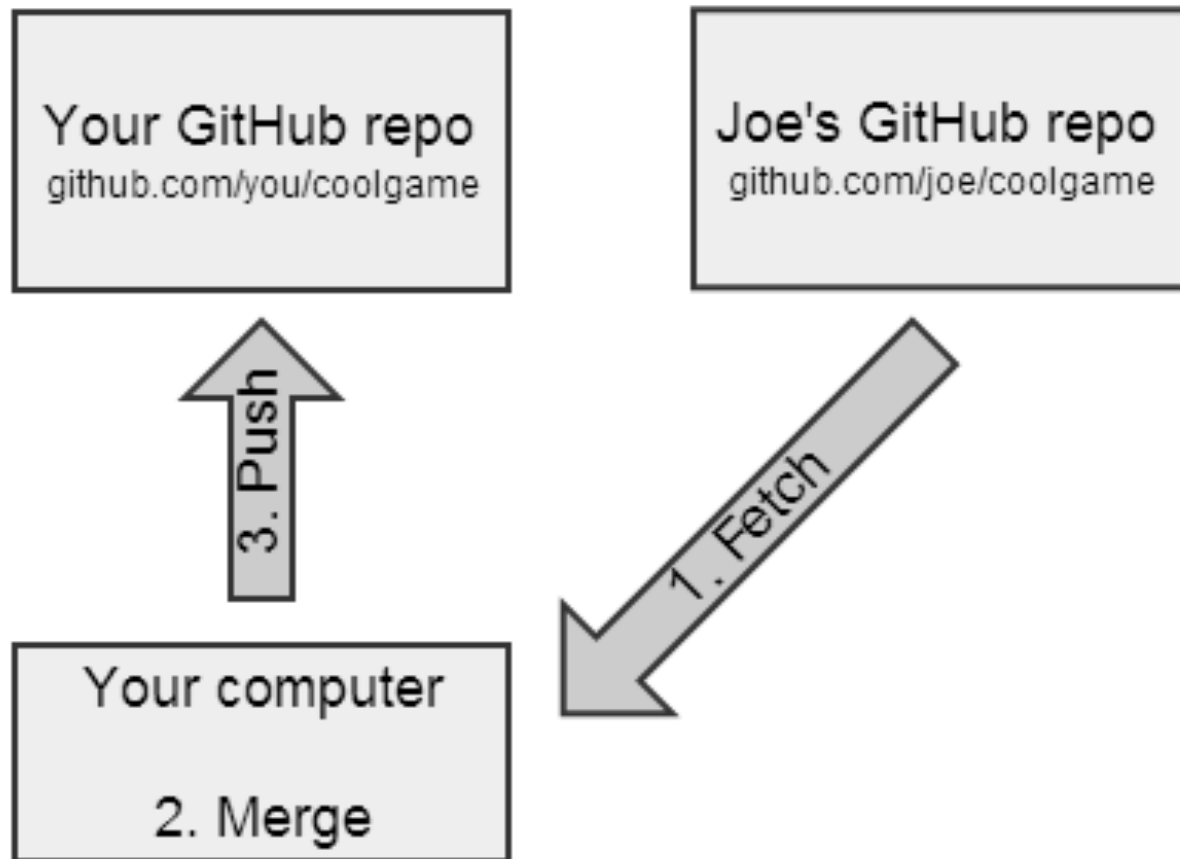
GitHub flow for contributing



Contribute to DAT6-students

- Create a subdirectory in DAT6-students with your first name (lowercase): `mkdir <name>`
- Change to that subdirectory: `cd <name>`
- Create a file named “about.md”
- Write something using Markdown (optional)
- Using Git: add, commit, push
- Create a pull request: “I request that you pull changes from my repo into your repo”

GitHub flow for syncing a fork



Sync your “DAT6-students” fork

- Files have been added to DAT6-students
- Add an “upstream” remote (one-time operation):
 - `git remote add upstream <Sinan’s URL>`
 - Check that it worked: `git remote -v`
- Pull the changes from the upstream:
 - Like updating your local files from Dropbox
 - `git pull upstream master`
 - Pull = fetch + merge (basically)
- Push the changes up to GitHub (optional):
 - `git push origin master`

Recipe for submitting homework

1. Navigate to DAT4-students (locally)
2. `git pull upstream master`
3. Copy your homework file(s) to your folder
4. `git add .` or `git add <filename>`
5. `git status`
6. `git commit -m "message"`
7. `git push origin master`
8. Create GitHub pull request (from your fork)

V. 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

Gists: lightweight repos

- You have access to Gist: 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