Introduction to Git and GitHub

By Kevin Markham

Agenda

- What is Git? What is GitHub?
- Why are we learning this?
- Reflections on learning Git
- Getting set up
- Practical exercises (many!)
- What we didn’t cover
- Further resources

What is Git?

- System for version control
- Primarily used by programmers
- Runs from the command line
- Allows you to track files and file changes in a repository (aka “repo”)
- Can be used alone or in a team
  - Team members can work independently on the same files and merge changes together

What is GitHub?

- A website, not a version control system
- Allows you to put your Git repos online
- Benefits of GitHub:
  - Backup of files
  - Visual interface for navigating repos
  - Others can navigate your repos
  - Easy to collaborate on repos
  - “GitHub is just Dropbox for Git”
- Note: Git does not require GitHub

Why are we learning this?

- Data scientists write code
- Version control is useful when you write code
- Allows you collaborate more effectively with colleagues
- Allows you to contribute to open source projects
- Attractive skill for employment

Git can be hard for beginners

- Designed (by programmers) for power and flexibility over ease of use
- Many ways to accomplish the same task
- Hard to know if what you did was right
- Most actions are permanent (hard to explore)
- Hard for others to troubleshoot
- Most reference materials are not written for beginners

Beginner mistakes (that we will avoid!)

- Misunderstanding the difference between Git and GitHub
- Typing commands without knowing what they do (can be a good thing in other languages)
- Leaving out optional arguments to commands
- Lacking an understanding of the workflow

Don’t sweat it!

- We’re going to focus on the most important 10% of Git
- You’re going to be doing the same tasks over and over
- Being slow to learn Git will not hold you back in the rest of the course
- We can help you to troubleshoot
- We’re all in this together!

Installation

- GitHub:
  - Create an account at github.com
  - There’s nothing to install
  - Note: “GitHub for Windows” & “GitHub for Mac” are GUI clients (alternatives to command line)
- Git:
  - Download from git-scm.com/downloads
  - Install

Setup

- 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: tiny.cc/gitssh
  - More secure that HTTPS

Navigating a GitHub repo (1 of 2)

- Example repo: git.io/ggplot2
- Account name, repo name, description
- Folder structure
- Viewing files
  - Rendered vs raw
  - Change history
- README.md
  - Side note: What is Markdown?

Navigating a GitHub repo (2 of 2)

- Commits:
  - One or more changes to one or more files
  - Commit comments
  - Revision highlighting
- Most recent commit: comment and date
- Issues, pull requests, branches, contributors
- Profile page, repo list

Creating a repo on GitHub

- Creating a repo:
  - Name, description, public/private
  - Initialize with README (if you’re going to clone)
  - Do it!
- Notes:
  - Nothing has happened to your local computer
  - This was done on GitHub, but GitHub used Git to add the README.md file

Cloning a GitHub repo

- Cloning a repo:
  - Copies it to your local computer
- Commands:
  - Change your working directory: cd
  - Clone: git clone <URL>
- Notes:
  - Copy SSH or HTTPS URL from GitHub (ends in .git)
  - No visual feedback when you type your password

Examining your cloned repo

- Cloned repo is a subdirectory of your working directory (with the same name as the repo)
  - Navigate to the repo (cd) then list files (ls)
- Branch name is shown:
  - Folder is being tracked by Git
  - Working on master branch
  - No need to initialize Git
- Everything is managed by ".git" folder

Check your remotes

- A “remote alias” is a reference to a repo not on your local computer
- “origin” remote was set up by “git clone”
- View remotes: git remote -v
- Add remote: git remote add <alias> <URL>

Making changes, checking your status

- Making changes
  - Modify README.md
  - Create new file with “touch” (not a Git command)
- Check your status
  - git status
- Possible file statuses:
  - Untracked (red)
  - Tracked and modified (red)
  - Staged for committing (green)
  - Committed

Committing changes

- Stage changes for committing:
  - Add 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

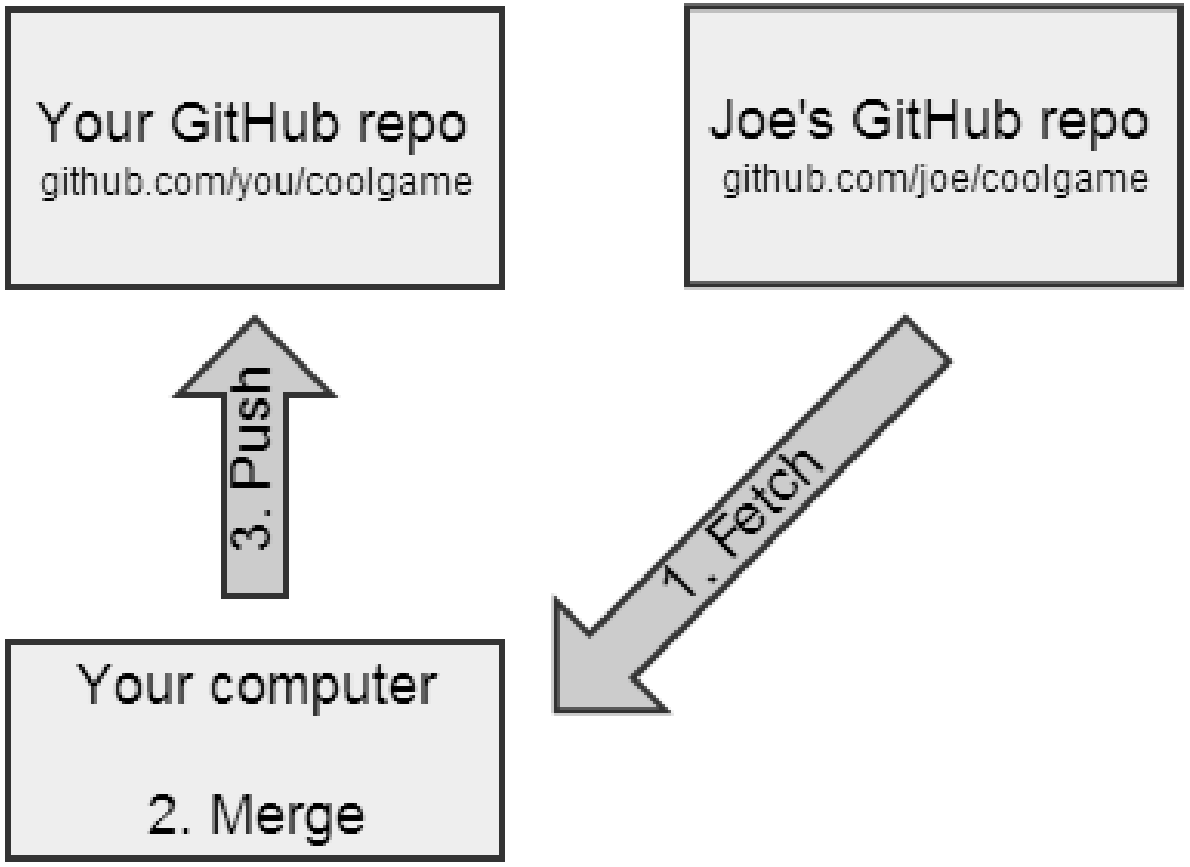
Pushing to GitHub

- Everything you've done to your cloned repo (so far) has been local
- Push committed changes to GitHub:
  - git push <remote> <branch>
  - Usually: git push origin master
- Refresh your GitHub repo to check!

Forking a repo on GitHub

- What is forking?
  - Copy a repo to your account (includes everything!)
  - Links to the “upstream” (but does not stay in sync)
  - Do it! git.io/gadsdc2
  - Note: This is a GitHub operation!
- Why fork?
  - You want a copy of the files on GitHub
  - You want to contribute
- Now clone this repo: git clone <your URL>

GitHub flow for syncing a fork



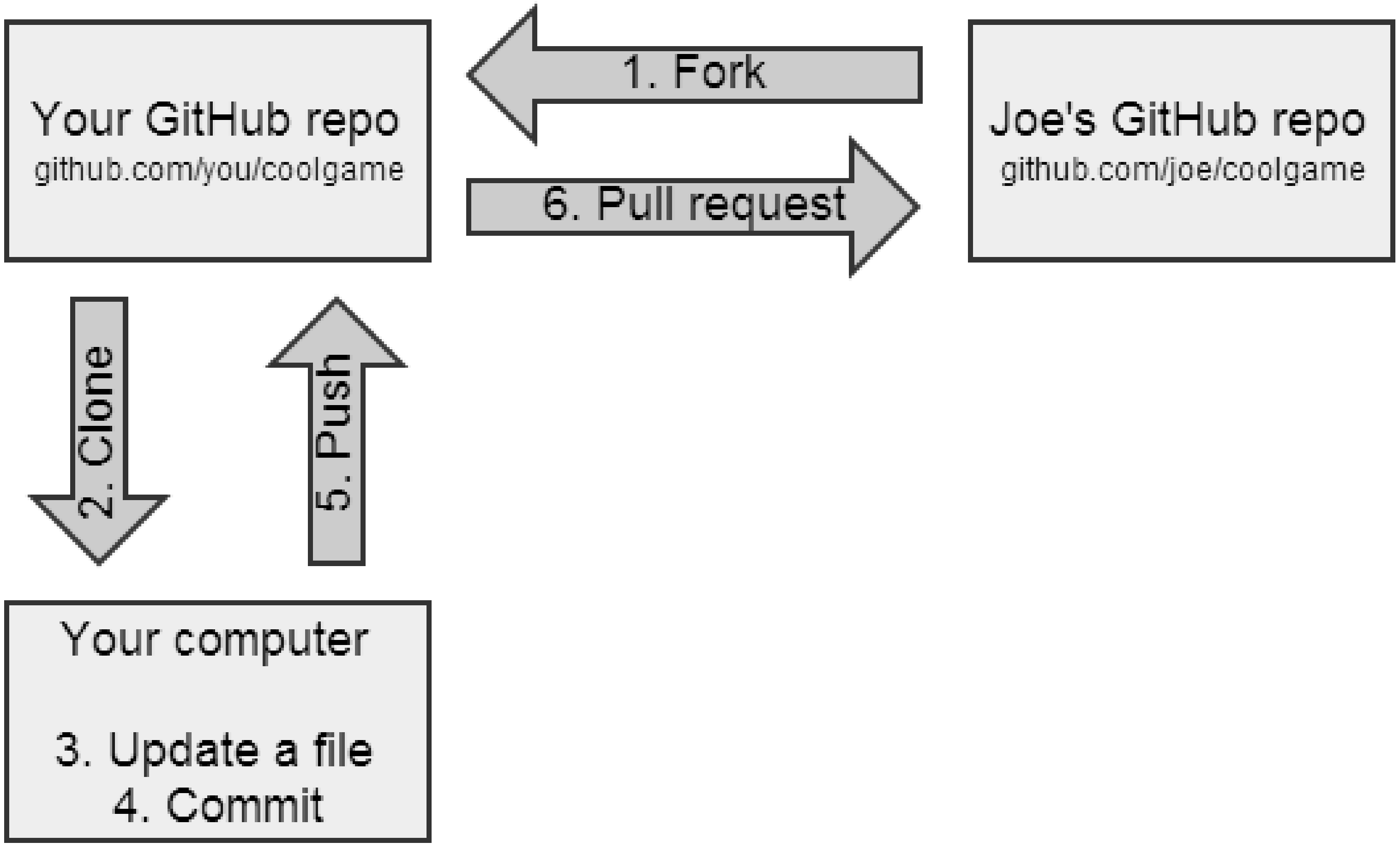
Sync your gadsdc2 fork!

- We've added a new file to gadsdc2
- Add an "upstream" remote (one-time operation):
  - git remote add upstream <Aaron's URL>
  - Check that it worked: git remote -v
- Pull the changes from the upstream:
  - git pull upstream master
  - (Does the same thing as "git fetch upstream master" + "git merge upstream/master")
- Push the changes up to GitHub (optional):
  - git push origin master

Working with branches

- A branch is a “context” for your work
- Branches control your files:
  - Create new branch and switch to it: git checkout -b newbranch
  - Add a file (touch), check that you added it (ls), add it, commit it
  - Switch to master branch: git checkout master
    - ls: the file is gone!
    - git log: the commit is gone!
  - Switch to other branch: git checkout newbranch
    - Phew, it’s still there
- Deleting a branch
  - Switch back to master: git checkout master
  - git branch -d newbranch (generates error)
  - git branch -D newbranch (force the deletion)

GitHub flow for contributing



Recipe for submitting assignments

- git checkout master
- git pull upstream master
- git checkout -b <branchname>
- # do your assignment
- git add <filename>
- git commit -m “message”
- git push origin <branchname>
- GitHub: switch to <branchname>, submit PR

Not covered (but useful to learn!)

- Initializing a repo locally (git init), then later pushing it to GitHub
- Deleting or moving a repo locally
- Deleting a repo on GitHub (easy)
- Using .gitignore so that Git ignores files
- Viewing diffs using Git
- Rolling back or unstaging changes
- Resolving merge conflicts
- Fixing LF/CRLF issues

Further resources

- Pro Git (book): git-scm.com/book
- Git Reference: gitref.org
- My reference guide: tiny.cc/gitref
  - Common sets of commands explained
  - Links to my video series (watch most of this presentation again!)