Git Flow

Bookmark this Git Cheatsheet: https://www.git-tower.com/blog/git-cheat-sheet

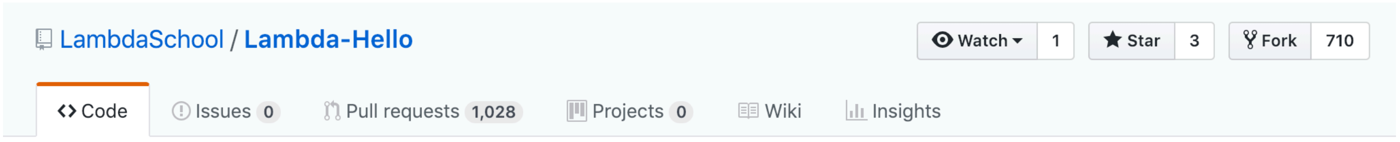
STEP 1: IN GITHUB: FORK IT & ADD PM AS COLLABORATOR
STEP 2: IN TERMINAL: GIT CLONE
STEP 3: CREATE BRANCH <firstname-lastname>
STEP 4: WORK ON PROJECT: MAKE, ADD & COMMIT CHANGES
STEP 5: PUSH LOCAL REPO to REMOTE REPO aka GITHUB
STEP 6: ON GITHUB, CREATE PULL REQUEST

GIT VS GITHUB? Git is a revision control system, a tool to manage your source code history. **GitHub** is a hosting service for Git repositories. **Git** is the tool, **GitHub** is the service for projects that use Git.

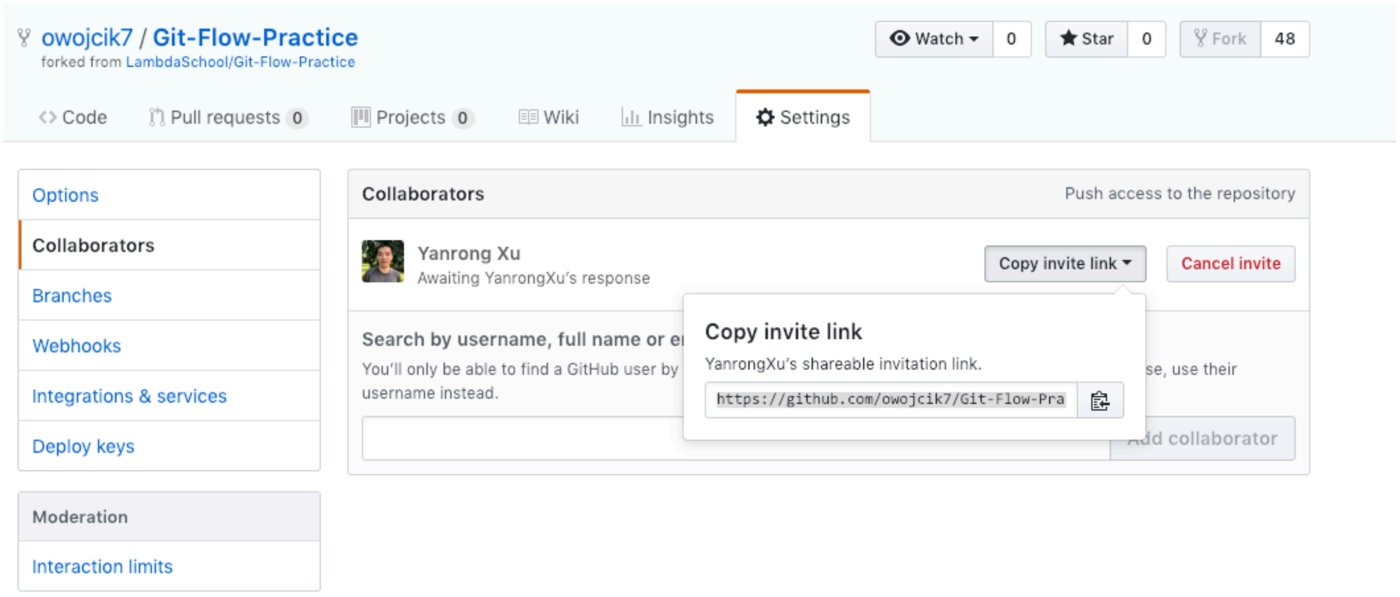
STEP 1: IN GITHUB: FORK IT & ADD PM AS COLLABORATOR

A **repository** (repo) is used to organize a single project. It consists of folders, files, datasets, etc -anything your project needs.

By **forking** someone else's repository (Lambda School's), we create a copy of it on our own Github account and work on it ourselves. **We must fork any Lambda Schoolproject before we start working on it.**



In github repository settings -> add PM as collaborator

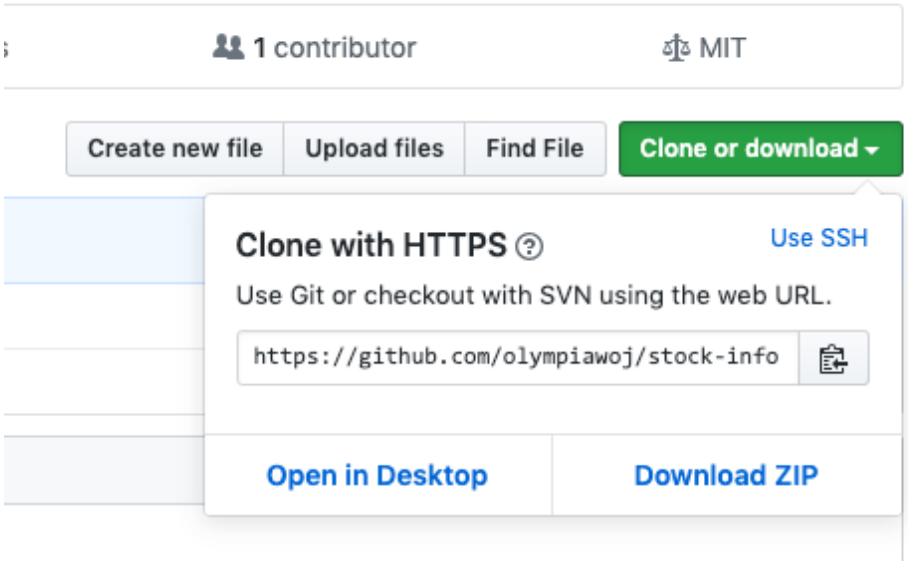


```
https://github.com/owojcik7/Git-Flow-Pra
```

STEP 2: IN TERMINAL: GIT CLONE

Cloning is the process of telling git to go and get all the files from github and do behind the scenes thing for us.

- **Double check fork:** Double-check that we've forked it on our own page. Look for your user name and forked from.
- **Clone or download:** Click the green `Clone or download` button and `Ctrl+C` or click the copy board.
- **Terminal:** in your terminal, *in the git folder you want to copy this to* run `` `git clone <url>` ``

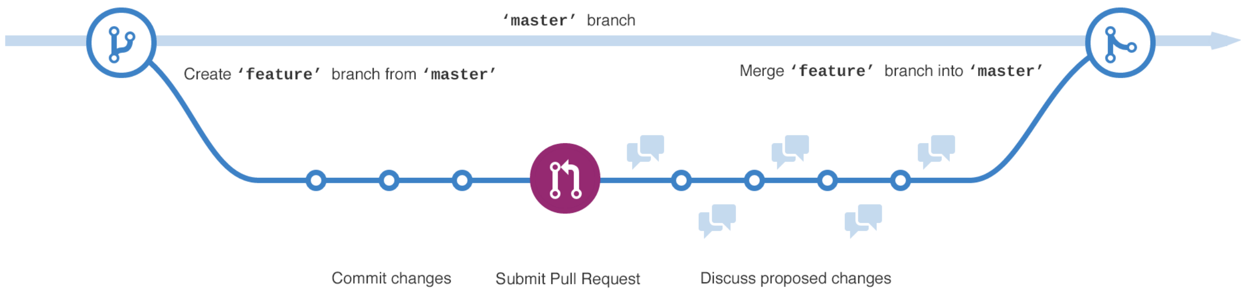


```
https://github.com/olympiawoj/stock-info
```

STEP 3: CREATE BRANCH <firstname-lastname>

Branching is a way to work on different versions of a repository at one time. The master branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master upon completion.

We use branches to experiment and make edits before committing them to master. When you create a branch off the master branch, you're creating a copy or snapshot of master as it was at that point in time.


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

A **branches** in Git are simply pointers to a commit. They are saved different versions of the repo.

At GitHub, developers use branches for keeping bug fixes and feature work separate from our master (production) branch. When a change is ready, they merge their branch into master.

In terminal in cloned file create a branch -> `` `git checkout -b "firstname-lastname"` ``

Branch commands:

- `git branch -av` to list all existing branches
- `git checkout -b <branchname>` creates a new branch AND switches you to that branch. It's a combination of:
  - `git checkout <branch>` switches branches
  - `git branch <branchname>` creates a new branch

<u>STEP 4: WORK ON PROJECT: MAKE, ADD & COMMIT CHANGES</u>

<u>*Make some changes*</u>

- `` `git status` `` to tell us the current status of our git project
  - Tells us what branch we are on
  - Tells us if there have been any changes. **Working tree clean means we haven't changed anything in our project file yet.**
- Make changes in your code, add new files, write lines of code
- `` `git status` `` to see that there is an untracked file, which means that git has no knowledge of this file.

<u>*Add them to the staging area*</u>

Git add command adds a change in the working directly to the staging area, telling Git you want to include updates to a particular file in the next commit. However, it does NOT affect the repository in any way- no changes are recorded until you run git commit.

- `git add newFile.js-` we are going to tell it to add the new file.

  - `git add -A` --if you have a LOT of files
  - `git add .` --these shortcuts save
  - `git add all` --ALL of the files!
- `` `git status` `` to tell us the current status of our git project

Commit/Save

Git Commit saves a snapshot of a repository at different points in time, meant to record different versions of a project.

- `` `git commit` `` -m "added new file, newfile.js”

STEP 5: PUSH LOCAL REPO to REMOTE REPO aka GITHUB

Right these changes are saved on our computer, NOT on our github account yet. To add to our github account, we use the command called **push** which pushes your local repository to a remote repository

Use **git push** to push commits made on your local branch to a remote repositor. Git push takes two arguments:

- **a remote name**- ex: origin
- **a branch name**- ex: master

`` `git push origin <branchname>` `` or for our projects —> `git push origin <firstname-lastname>`

STEP 6: ON GITHUB, CREATE PULL REQUEST

- Make sure your on your firstname-lastname branch and then click **New Pull Request**
- **Click base fork: dropdown -> select** `firstname-lastname/project-name`
- **Click reviewers button, add your PM**
  - NOTE: If your PM doesn't pop up when you type it in for reviewer it's because they haven't accepted your collaborator invite. DM them at that point because they might miss you sometimes. Once they've accepted move to next step.
- Click **create pull request** with message “MVP complete"
- **DO NOT CLICK MERGE PULL REQUEST**. Your PM will be the only person do this after they’ve confirmed you’ve met MVP.
- Check your e-mail to see when request is merged.