Gotta Get Git

Understanding Git & GitHub
Presented by Trent Hand

What’s the difference between Git and GitHub?

*Git is a local version control software that allows users to take snapshots of their code (stored as versions), branch off of a master copy to test/write new code, return to a previous version in case of irreparable damage, and share their code with other developers’ machines.

*GitHub is an online storage facility for depositing your code in repositories, which can then be shared with pretty much anyone in the world(assuming they know how to use Git & GitHub).

*So, in a nutshell, Git is how you version control on your machine, and GitHub is how you share those versions with the world.

A deeper explanation: https://jahya.net/blog/git-vs-github/

Why bother with Git(or version control in general)?

*It’s better to try implementing a new feature in a safe environment

*You will very rarely be the only person working on a project.

*Your code WILL BREAK!!!

Why bother with GitHub?

* Having a central location where everyone can pull down the latest version of a project is crucial for collaborative dev environments.

* GitHub is the most popular repository site in the world (It's like Facebook, but without your Grandmother, unless she's REALLY awesome!), so chances are, wherever you go, GitHub will be used.

*There are a lot of great existing projects and code you can pull down to your local machine to work on or just enjoy the benefits of those who've coded before you.

Why is this so hard?

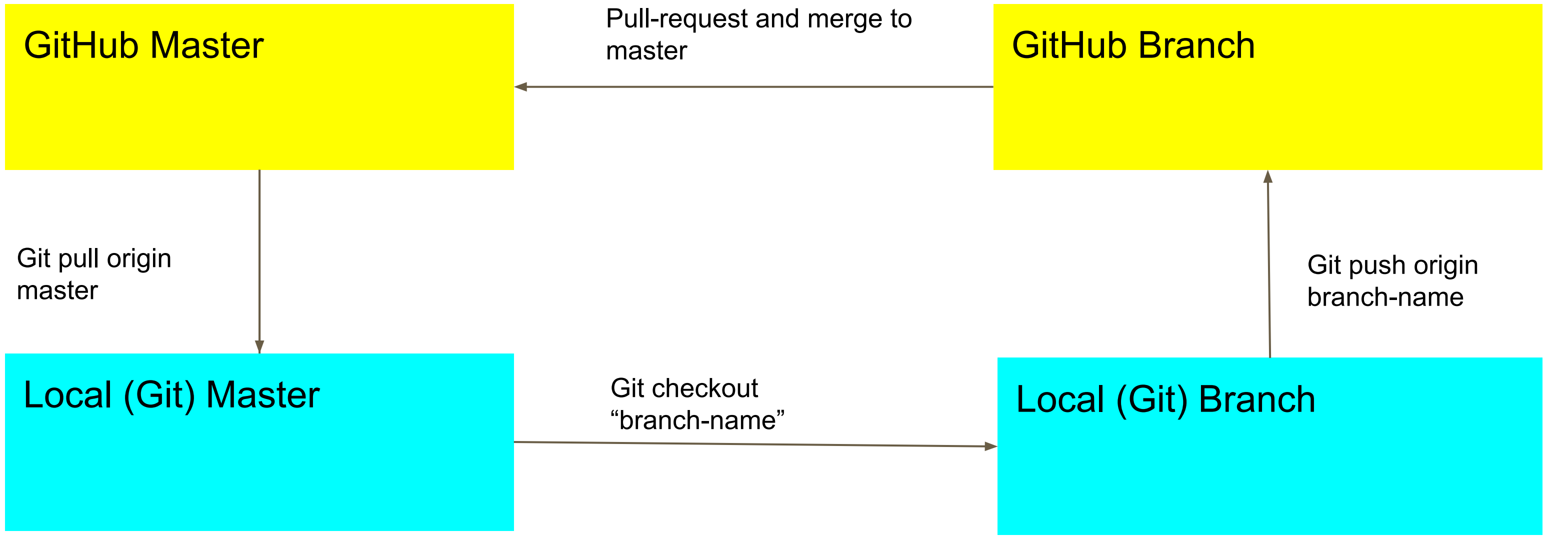
*Ever tried to skateboard? People doing it for years say it's easy. I beg to differ?

*The truth is, neither Git, GitHub, or skateboarding are inherently hard, they just all take practice (and a certain threshold of pain tolerance).

Creating a new project with GitHub

1) On GitHub, click the + button in the top left and create a new repository
2) Name the repository and hit the create button
3) In your terminal, go to the directory you want to start your project in and mkdir to create a new directory. Name the directory to match the name of the GitHub repo
4) Follow the **…or create a new repository on the command line** instructions
5) You are now ready to follow the Git/GitHub workflow instructions for the duration of the project :)

The Git/GitHub workflow pt. 1


GitHub Master
Pull-request and merge to master
GitHub Branch
Git pull origin master
Git push origin branch-name
Local (Git) Master
Git checkout “branch-name”
Local (Git) Branch

The Git/GitHub Workflow pt. 2

PROCESS FOR GIT/GITHUB for a new branch

1) Make sure you’re in the master branch
2) $ git pull origin master(updates your local master to match what’s on GitHub)
3) $ git checkout -b “branch_name”(creates a new branch and moves you to it)
4) Open your branch in your text editor and do some work(create .gitignore if there isn’t one)
5) $ git status(shows which files have been changed, added, or deleted)
6) $ git add . (adds all the files shown on git status)
7) $ git commit -m “message goes here”(this saves this commit or version)
8) $ git push origin BRANCH_NAME(pushes your code up to GitHub, creates branch there)
9) Pull-Request on GitHub(Someone will need to approve this for you in a group project)
10) Merge the branch to master on GitHub, then delete the branch on GitHub
11) $ git checkout master(brings you to master)
12) Repeat step 2, then $ git branch -D BRANCH_NAME(deletes the old branch)
13) Repeat from step 3

Git/GitHub pt. 3

PROCESS FOR GIT/GITHUB for a currently active branch

1) $ git stash (takes a snapshot of your current work, think like “cut”)
2) $ git checkout master(takes you back to master)
3) $ git pull origin master(updates your local master to match what’s on GitHub)
4) $ git checkout BRANCH_NAME(takes you back to your branch)
5) $ git merge master (updates your branch to match master)
6) $ git stash pop(paste your code back on the branch, may cause merge conflicts)
7) Resolve any merge conflicts, then work on your code
8) When done, repeat steps 5-13 on the previous slide

How to properly test a Pull Request

1) In you local terminal, $ git checkout master
2) $ git fetch NAME_OF_PR_BRANCH(this will pull in the branch and create it on your local drive)
3) $ git checkout NAME_OF_PR_BRANCH(to look at the code)
4) Test however is necessary(good PR's will have test instructions)
5) If the code isn't working or you spot a bug, leave a comment on GitHub detailing where you find these in the code. Test when the PR is updated.
6) If the code is good, approve the PR by whatever conventions your team uses
7) The PR is now ready to be merged to master

How do I use GitHub to access an existing project?

*There are really two ways to do this: Cloning or Fork&Clone

*Cloning simply means you are pulling the code into your local system(think of it like copy/paste)

*Forking is when you make a copy of the repository on your GitHub and then you can pull to your local drive.

*The difference is when you push back up to a forked repository, you won't affect the original GitHub repository at all.

For practice, let's go to this repository:
https://github.com/eorstrom/eorstrom.github.io