Introduction to Git and Github

Sandi Ritter (she/her)
Staff Software Engineer, P&G

HELLO!

Let's get to know each other and set the tone for our time together

Overview

- The basics of git:
    - Background and understanding of Git
    - Labs to set up Git
- The basics of github:
    - Difference between Git and Github
    - How to create a repo
    - Github features
    - Github desktop

“Generally, the best way to learn git is probably to first only do very basic things and not even look at some of the things you can do until you are familiar and confident about the basics.

Linus Torvalds

What is Git?

- Git is a modern version control system that is FREE
- It tracks and logs changes made to your files over time
- It allows for branching of your code, which means you can create independent local branches of the code
- A very large number of software projects rely on Git for version control, ranging in size from very tiny to extremely large

VCS:
Version Control System

Version Control Systems

Local

The simplest form where a database keeps all the changes to files under revision control. Changes are kept as patch sets and you can then add up all the patches needed to re-create a file at any point in time.

Centralized

A single repository contains all history and each user gets their own working copy of the latest snapshot of the code. You need to commit your changes in the repository and others can only see your changes by updating their working copy.

Example: Subversion

Distributed

Multiple repositories where each user has their own repository that contains all history in addition to their own working copy. You must commit and push your changes in order to make them visible on the central repository.

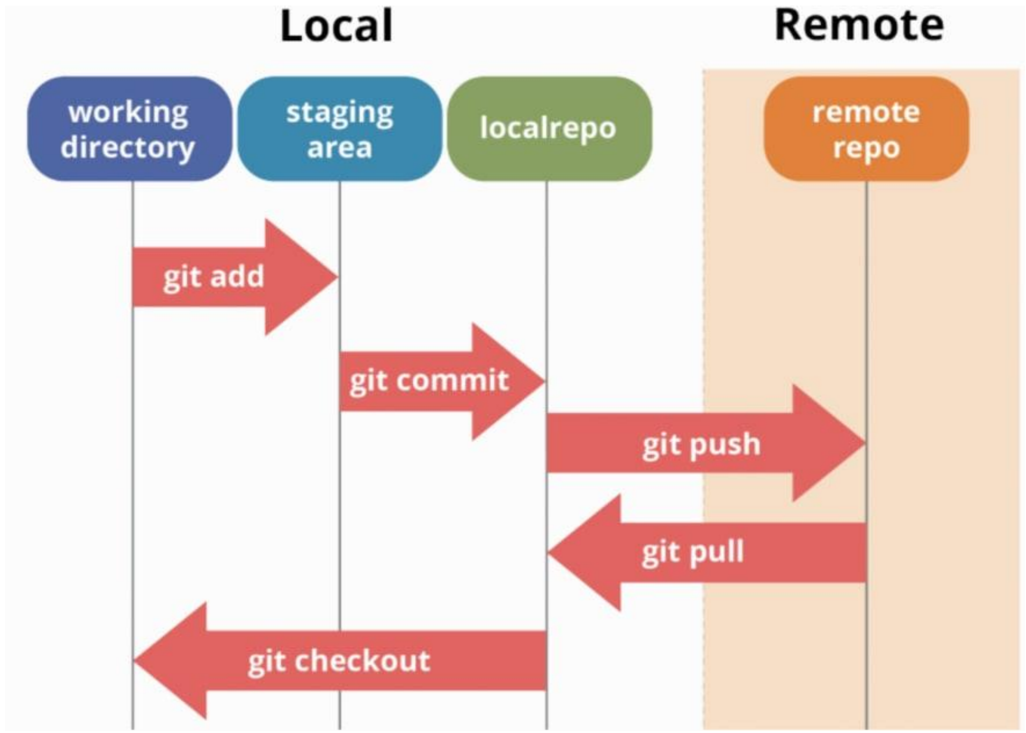
Example: Git

git

Why is a VCS such as Git needed?

- Allows software teams of all sizes to track changes to the code
- It enhances collaboration between team members
- Stores multiple version of the code, and allows any version to be restored
- Provides a complete history of all changes made
- Provides a backup of the code

How Does Git Work?



Some Useful Terms

Repo

A project's folder. A repository contains all of the project files and revision history. It can be either public or private

Checkout

Switches between different versions of the code, and generally is used with branches

Commit

A revision to the repo. Git creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of the the changes

Branch

A parallel version of a repository, generally used to change code in an encapsulated environment

Status

Shows the state of your working directory and helps you see all the files which are untracked by Git, staged or unstaged.

Pull Request

Proposed changes to a repo submitted by a user. It can be accepted or rejected. You are asking to have your changes “pulled into” the repo

More Useful Terms

Add

Adds the change(s) in the working directory to the staging area. However the changes are only recorded once they are committed.

Fetch

Retrieves the latest meta-data info from the original but doesn't do any file transferring. Basically checking to see if there are any changes available

Merge

Taking the changes from one branch (in the same repository or from a fork), and applying them into another

Push

To send your committed changes to a remote repository on github. You "push" them to the repo

Pull

Fetch and download content from a remote repository and update the local repository to match. It's a combination of two other commands, fetch followed by merge

Where do we start??

- Git will need to be installed on any device used to modify your code
- Git can be configured and personalized
- We will use the command line to accomplish this
- We will also need an IDE to make code changes

Git Setup

1. Start by downloading git
   a. If using windows, once the installer has been downloaded, you should see the Git Setup wizard screen. Follow the prompts to complete the installation
   b. If using a mac, the easiest method is via homebrew. Open a terminal window and use the command:

      brew install git

Git Configuration

Using the terminal or git bash, let's set basic config variables:

a. *git config --global user.name "My Name"*
b. *git config --global user.email me@example.com*
c. This can be verified by using the command

   git config --list

IDE Installation

For this course, you can use any IDE you have installed. However, if you need to install one, let's use VScode

a. Download from here : https://code.visualstudio.com/download
b. For mac, double click the downloaded file and the installation will complete
c. For windows, once downloaded, run the installer and follow the prompts

CONGRATS!!

You have successfully installed git, an IDE and can now successfully manage all your code!

git vs GitHub

- Software
- Focused on version control and sharing
- Installed locally on your machine

- Service
- Focused on the hosting of source code
- Hosted in the cloud

Github - Let's Create an account!

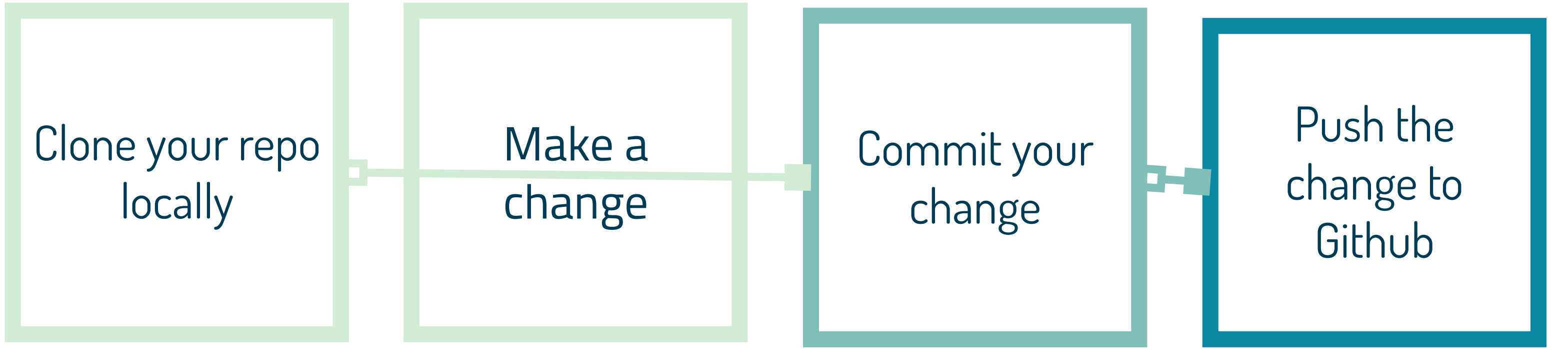
1. Navigate to https://github.com
2. Create an account (if needed)
   a. Complete all fields
   b. Verify email

If you have an account, just sign in instead

Github - Create a Repo

1. Click the "Create Repository" button on the left side of the screen
2. In the Repository Name field, enter *yourusername*.github.io
3. Leave the public setting enabled
4. Click Create Repository button
5. Repo created!

Let’s Practice using Git



Github – Clone your Repo

1. Click the copy button next to the repo name
2. Open your terminal
3. Navigate to a location where you prefer to store your code, or create a new folder
4. Type *git clone theRepoNameJustCopied* and enter
5. Change directory to the new repo

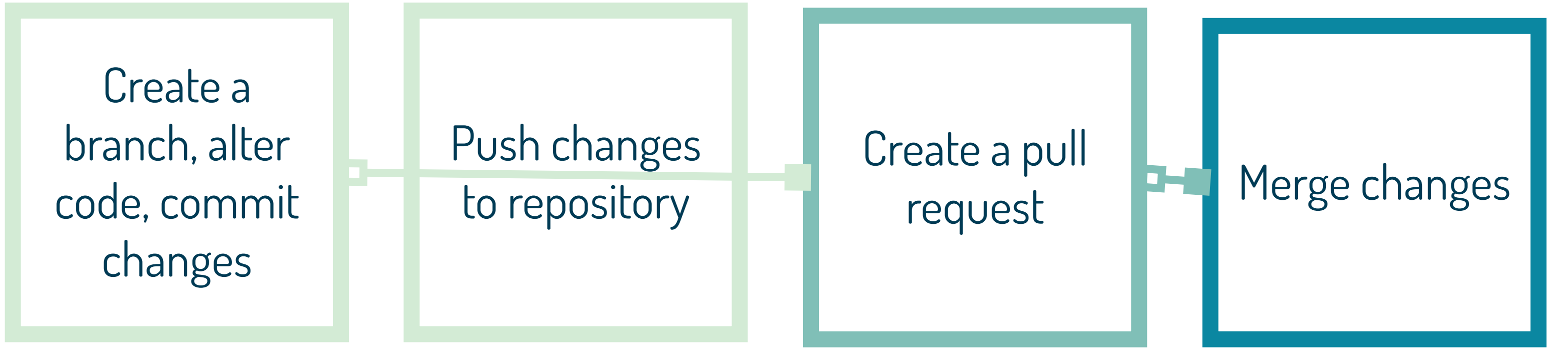
Github – Update your Repo

1. Create a basic html file:
   a. echo "Hello World" > index.html
2. Type *git status*
   a. notice the new untracked file
3. Type *git add .*
4. Type *git status*
   a. notice the file is now labeled as a new file

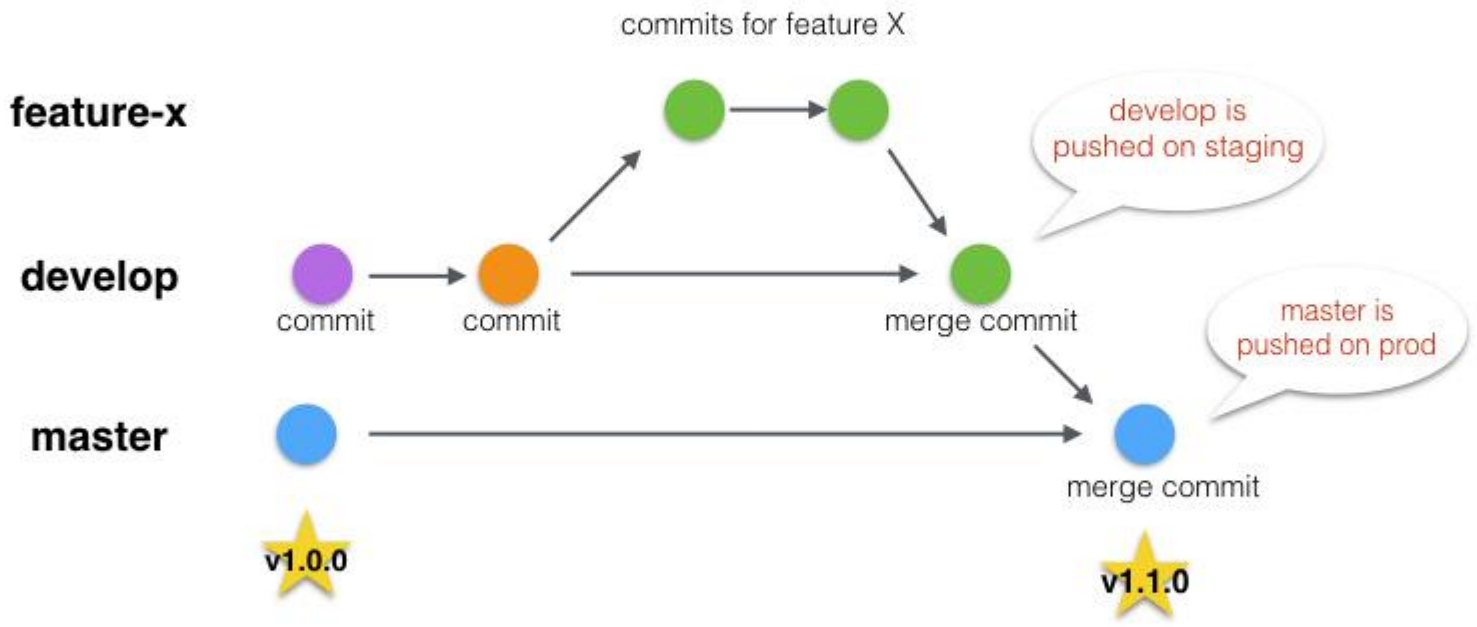
Github - Commit to Repo

1. Type *git commit -m "initial commit"*
2. Type *git status*
   a. nothing to commit, working tree clean
3. Type *git push -u origin main*
4. Your new file has now been pushed to your repo in github and branch main set up to track remote branch main from origin'
5. Navigate to *https://yourusername.github.io.*

Let's Practice Branching


Create a branch, alter code, commit changes
Push changes to repository
Create a pull request
Merge changes

Git Flow


commits for feature X
feature-x
develop is pushed on staging
develop
commit
commit
merge commit
master is pushed on prod
master
merge commit
v1.0.0
v1.1.0

Create a Branch

- Navigate to your source code directory
- Type *git branch* to verify you are on the master branch
- Type *git branch myNewBranchName*
- Type *git checkout myNewBranchName*

This is the long way to do this, there is a short cut!

Make Code Changes and Commit

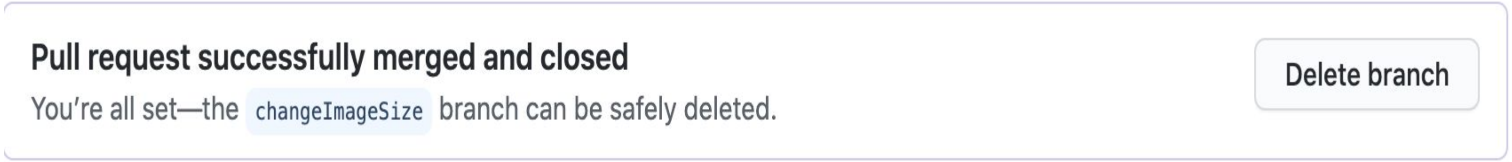
- Open the code in your favorite editor
- Make some code changes to your branch – add a photo, change the words, make it reflect yourself!
- When done, in your terminal type *git status*
- Then add the file changes with *git add .*
- Then verify with *git status*
- Then commit with *git commit -m "my commit message"*

Push Changes and Create a Pull Request

- Type *git push origin myNewBranchName*
- Navigate to your repo in Github
- Click on 'Pull Requests'
- Click on 'New Pull Request'
- Using the compare dropdown, select your branch
- Verify the files are as you expect
- Click 'Create Pull Request'

Merge Pull Request

- In your repo in Github,click on ‘Pull Requests’
- Locate the pull request you just created and select it
- Click “merge pull request” button, and “confirm merge”
- You should see a message indicating that your changes have been successfully merged


Pull request successfully merged and closed

You’re all set—the `changeImageSize` branch can be safely deleted.

This octocat represents my love of the command line! However, it is not for everyone and many people prefer to use a GUI

Github Desktop

This can be installed via download from https://desktop.github.com/

It provides a more visual representation of the code flow and many find this (and similar options) more user friendly.

It is very basic, as it is free, and there are many other options such as fork and gitkraken

Github Desktop vs Command Line

Let's follow the same basic steps we did previously via the command line

- Create a branch
- Make a change on the new branch
- Commit change
- Create pull request
- Merge pull request

Resources

- https://training.github.com/downloads/github-git-cheat-sheet.pdf
- https://git-scm.com/docs/git-config
- https://lab.github.com/
- https://dangitgit.com/en
- https://desktop.github.com/
- https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop

THANKS!

Any questions?

You can find me at

sandiritter68@gmail.com