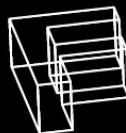


Git & Github:

Our Coding Safety Net



bit.ly/tf-git-github

Introductions

- What's your name?
- What brought you here today?
- What is your programming experience?

About Thinkful

We train developers and
data scientists through
1x1 mentorship and
project-based learning.

Guaranteed.



Why Learn Git?

- Ever made a mistake and wish you could take it back?
 - ◆ Git lets you do that
- If suddenly broken, allows us to roll back to when our project was working
- Allows for easier collaboration with other developers
- Can be intimidating and complex at first, but a little knowledge will go a long way

Getting Set Up - we will work on this together

→ Please create a GitHub account.

GitHub: <https://github.com/join>

→ Then, create a Gitpod account using your GitHub login.

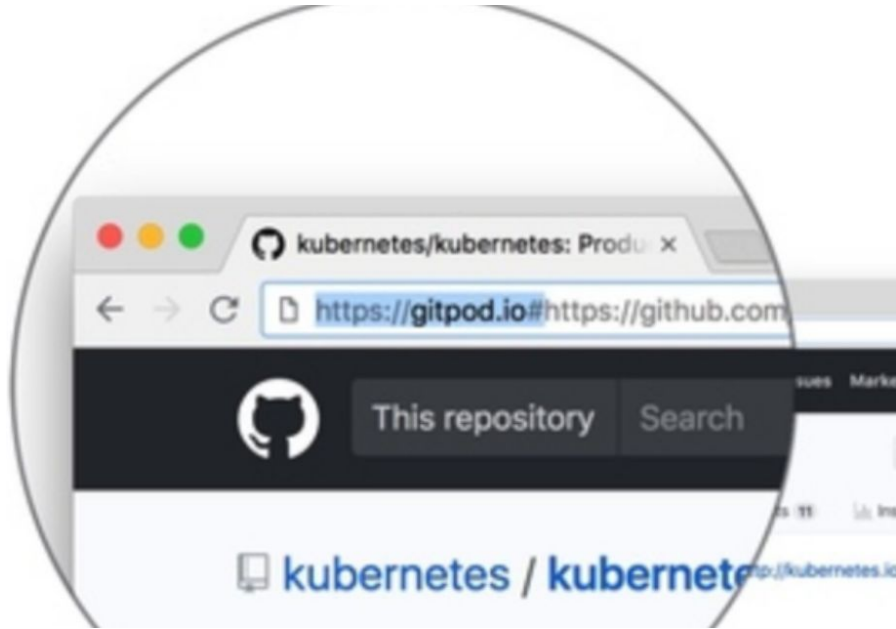
Gitpod: <https://www.gitpod.io/>

→ We will FORK a repo together. (If you don't know what that means, no worries, we will talk about it!)

Git Workshop repo:

<https://github.com/chrisbridges/thankful-git-workshop>

Getting Set Up - we will work on this together



→ Link gitpod and this repo by adding (copy and paste) [https://gitpod.io#](https://gitpod.io#https://github.com/kubernetes/kubernetes: Production) to the beginning of the url, as shown below.

Example URL:

`https://gitpod.io#https://github.com/{username}/thankful-git-ws/tree/master/git`

What Is Git?



- A version-control system that allows us to track changes made to files, and document when / where changes occurred, as well as who made the changes
- Uses a branching system that allows us to modify code without fear of breaking existing functionality

GitHub

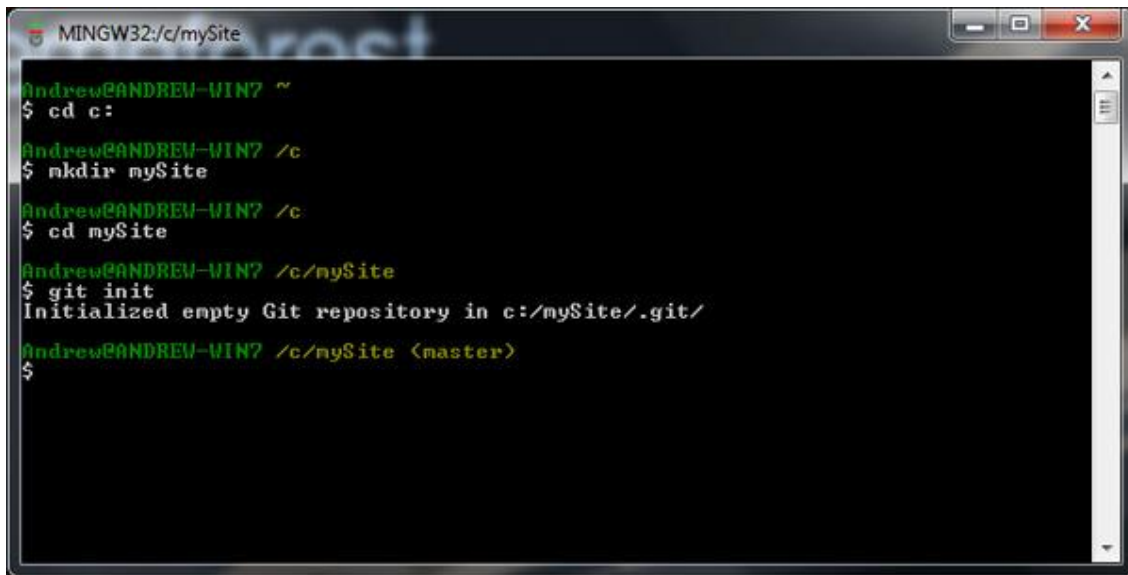


- The most popular site for remotely saving our code
- Allows for easier collaboration and communication of projects involving multiple developers
- GitHub \neq Git

Local / Remote Branches

- Local Branch
 - ◆ Branch that is stored on your personal machine
 - ◆ Can be altered without fear of editing remote branches
- Remote Branch
 - ◆ Branch stored in a separate repository, like GitHub, BitBucket, GitLab, etc

Git Init



```
MINGW32/c:/mySite
Andrew@ANDREW-WIN7 ~
$ cd c:
Andrew@ANDREW-WIN7 /c
$ mkdir mySite
Andrew@ANDREW-WIN7 /c
$ cd mySite
Andrew@ANDREW-WIN7 /c/mySite
$ git init
Initialized empty Git repository in c:/mySite/.git/
Andrew@ANDREW-WIN7 /c/mySite <master>
$
```

- Initializes our current directory as a Git repo
- Allows Git to start tracking changes to your files

Git

Status

```
no changes added to commit (use "git add" and/or "git commit -a")
[vacas: ~/learnLua (waqas)]$ git status
On branch waqas
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/garbage.lua

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test/
        test1/

no changes added to commit (use "git add" and/or "git commit -a")
```

- One of the most important and helpful Git commands
- Tells us
 - ◆ Which files have been altered
 - ◆ Which files are currently being tracked
 - ◆ Which branch we are currently on
 - ◆ Where we are in relation to our remote branch (how many commits ahead / behind)

Git

Add

```
~/emacs.d(master) $> git init
Reinitialized existing Git repository in /home/nick/.emacs.d/.git/

~/emacs.d(master) $> git add -A

~/emacs.d(master) $> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitignore
        modified:   auto-save-list/.save-8600-nick-ThinkPad-X200~
        modified:   projectile.cache

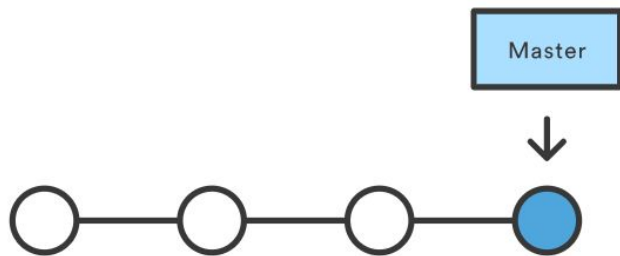
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)

        modified:   el-get/ace-jump-mode (untracked content)
        modified:   el-get/ace-window (untracked content)
        modified:   el-get/dash (untracked content)
        modified:   el-get/f (untracked content)
        modified:   el-get/flycheck (untracked content)
        modified:   el-get/fuzzy (untracked content)
        modified:   el-get/s (untracked content)

~/emacs.d(master) $> █
```

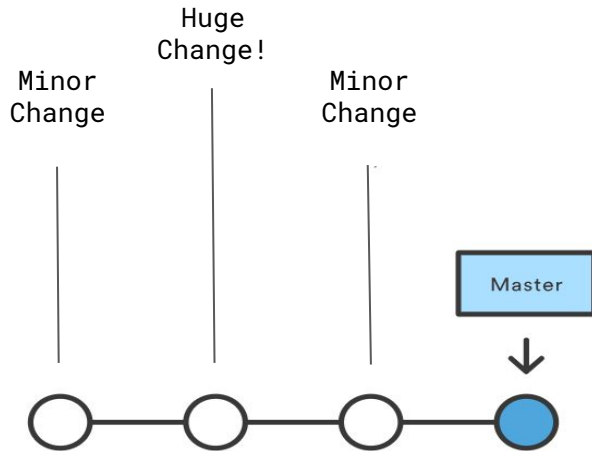
- Tells Git which files to add to the staging area, to prepare for commit
- We can add specific files or all files that have been altered

Git Commit



-
- Each commit is a snapshot of our project at that given time
 - Commits our code to our branch

Commit Often!



- Good Git habits will save you from yourself
- If we make monumental changes to our code base between commits, it's harder to know which code is breaking our project / we may lose code that is actually functional
- Git allows us to roll back to any of our previous commits

Git Clone & Forking

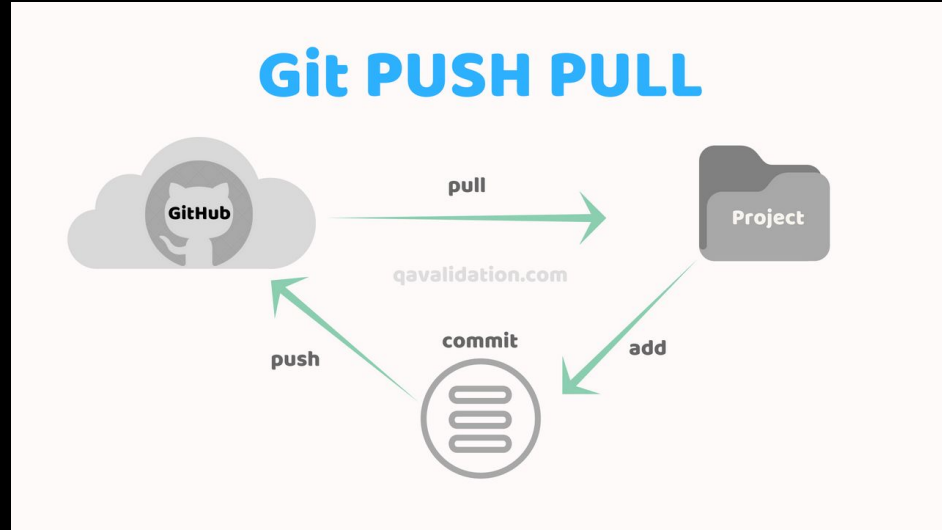
→ Git Clone

- ◆ Copying a project from a remote repo and downloading it to your local machine
- ◆ Will need access to push your changes to the remote

→ Forking

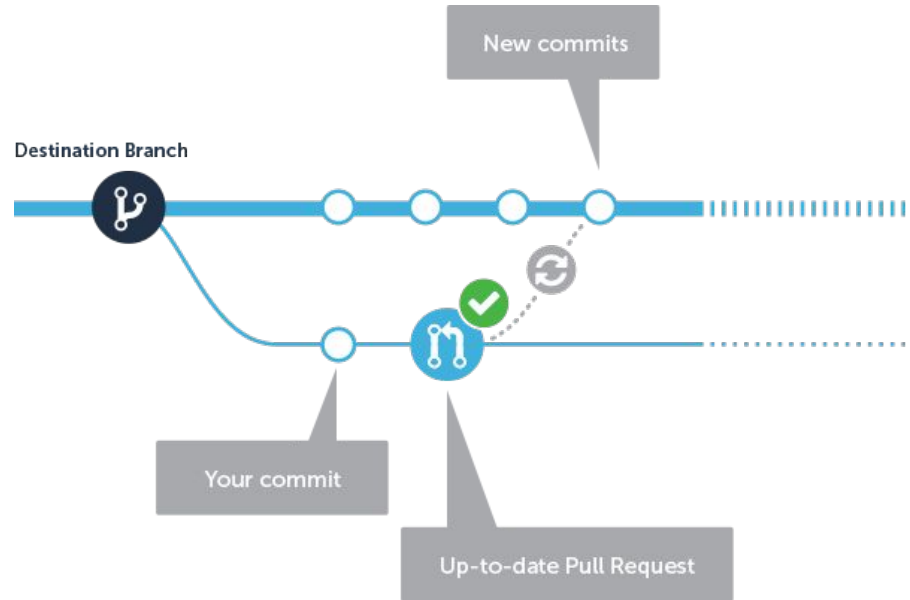
- ◆ Similar to clone, but creates your own branch of the project
- ◆ Allows you to hack away without fear of altering repo you forked from
- ◆ Your changes will need to go through a pull request process to be merged (to be discussed next)

Git Push & Pull



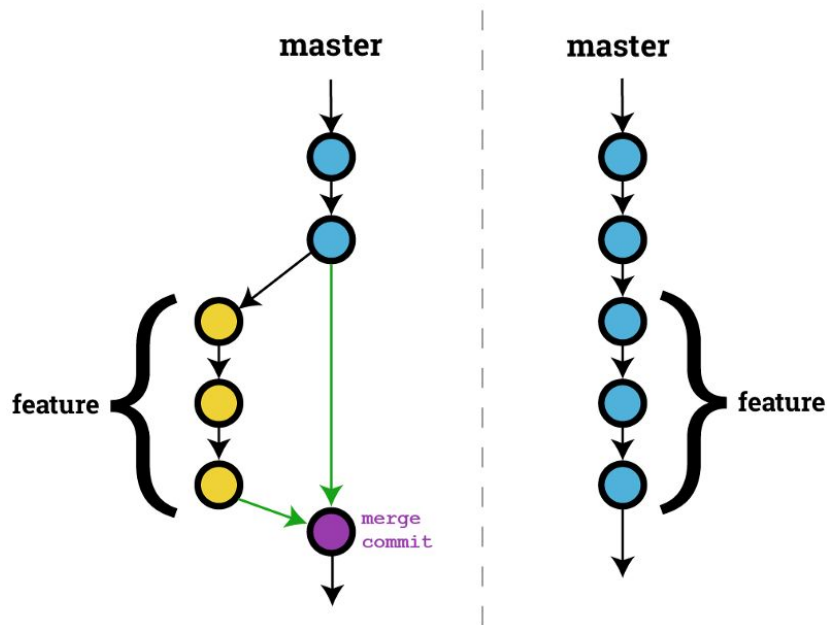
- Pushes our local branch up to our remote branch
- Pulls files and changes down from our remote branch

Pull Requests



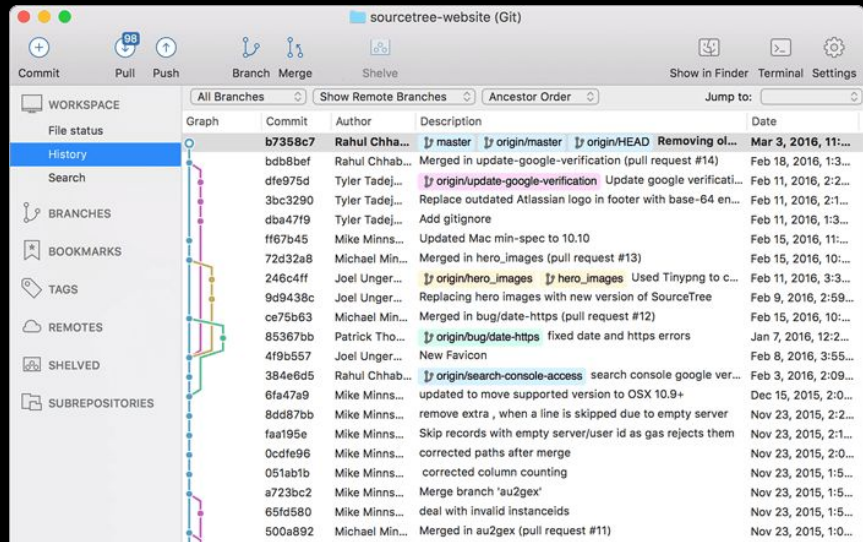
- Your code will typically be reviewed by one or more parties, and then, if accepted, merged into the branch

Git Merge



- Similar to pull requests, but typically performed on branches on your local machine
- Good practice: every new feature you add, create a new branch, and then merge
 - ◆ Prevents breaking functionality on your master

GUIs



→ Graphical user interfaces can make visualizing our git history much simpler

Summary

- Local & Remote
- Init
- Status
- Add
- Commit
- Clone
- Forking
- Push
- Pull
- Pull Requests
- Merge
- GUI

Assignments for Tonight

bit.ly/tf-learn-git

Learn by reading

Git Handbook

Git, GitHub, DVCS, oh my! Learn all the lingo and the basics of Git.

Cheat Sheets

Keep these handy! Reference sheets covering Git commands, features, SVN migrations, and bash. Available in a multiple languages.

Learn by doing

Learn Git branching

Try Git commands right from your web browser. Featuring some of your soon-to-be favorites: branch, add, commit, merge, revert, cherry-pick, rebase!

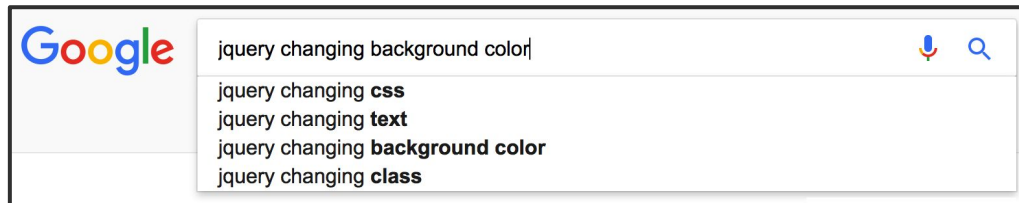
Visualizing Git

Look under the hood! Explore how Git commands affect the structure of a repository within your web browser with a free explore mode, and some constructed scenarios.

Git-It

You've downloaded Git, now what? Download Git-It to your machine and you'll get a hands-on tutorial that teaches you to use Git right from your local environment, using commands on real repositories.

Real
Developers
Use Google...
A Lot.



Ways to Learn Code

codecademy

meetup

w3schools.com

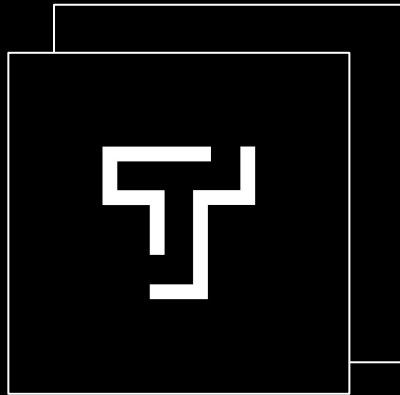
Google

stackoverflow

freeCodeCamp(🔥)

YouTube

Thinkful Two-Week Free Trial



- Free trial of Full Stack Flex online program
- Start with HTML, CSS and JavaScript
- Personal Program Manager
- Unlimited Q&A Sessions
- Student Slack Community
- bit.ly/freetrial-webdev