

Let's git this together



Let's git the commands



- Git init
 - Git add .
 - Git commit -m "Your Message"
 - Git push
-
- Git Branch
 - Git Checkout
 - Git Merge
 - Git flow
 - Git Status
-

Git init

Starting fresh and telling git that we want it to play with us and track our files.




```
git init
```

Git add

Tell git what files to track for us.

PS: You can either use filenames or . (dot means all files and folders)



```
git add .
```

```
git add filename
```

Git Commit

You are committing to it and save the state of all files. This is like a save for a game.

Committing also needs a message hence why we use -m

```
git commit -m "Init Commit 🚀"
```

Think about others



Git and a team



Even though you work on your own, think like you are working in a team.

- Commit messages should be nice and readable.
- As long as a tweet

```
git commit -m "fixed divs 🐛"
```

```
git commit -m "Removed unnecessary extra divs 👍"
```

Git push

Telling git that we want to push it to the cloud

Note that this can be github, gitlab, heroku anything that has git installed.



```
git push origin master
```




Let's ignore this...

Git Ignore

Because sometimes we don't want to keep track of things.



- Creating the file should be the first thing you do.
- It's just a text file
- With folders remember trailing /
- With files type the filename

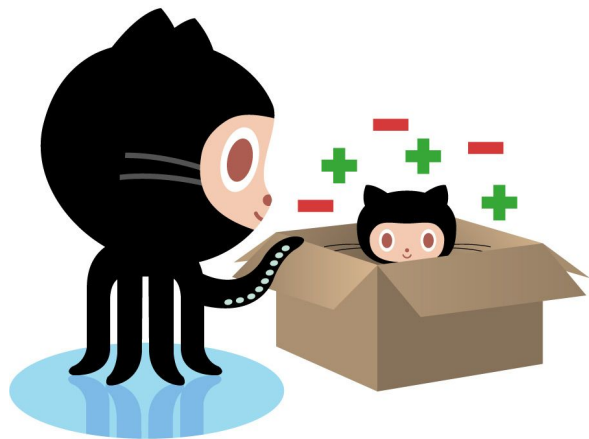
```
git rm -r --cached FILE/FOLDERNAME
# This will remove the .c9 folder from been tracked
# Only use this if you have a file or folder that you have committed you want removed
```



**Let's play with
history**

Merge

Because copy paste is old school. ✂️ 📄



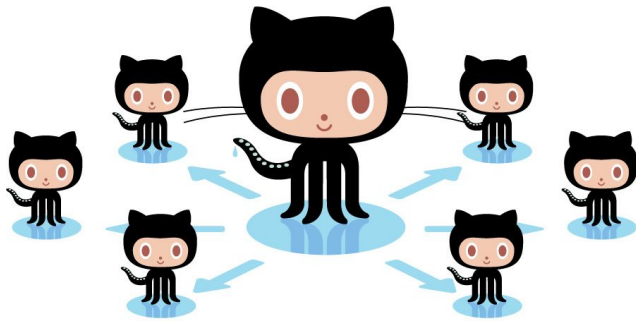
- No need for cut and paste
- Move code between timelines
- Keep old commits.
- History is preserved
- You merge things INTO the branch you are on.

```
$ git checkout master  
Switched to branch 'master'  
  
$ git merge <commit>  
$ git merge <branch>  
$ git merge my-feature
```

💡 🙌 This merges all commits from feature branch into master branch

Branches

What are these branches and why use them? 🌲



- Alternative Timelines
- Independent of others
- Great for testing new features

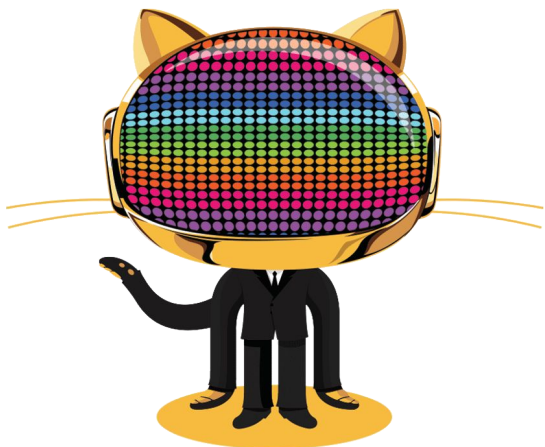
```
$ git branch
* master

$ git branch my-feature

* my-feature
  master
```

Checkout

Have you checked out now?



- Moving between timelines
- Peak into the feature or the past.
- Git is a time machine



```
$ git checkout <branch>  
  
$ git checkout my-feature  
  
Switched to branch 'my-feature'
```

**Keep calm, and remember
Git got this.**



Merge Conflicts

FATAL ERRORS are fixable



- Don't be alarmed
- Git is just saying you have to fix it
- You took changes that are pretty similar and git don't know what to do
- Fixing it is as simple as open the file

```
$ git status
$ git add filename
$ git commit -m "Fixed Merge Conflicts 🐛⚡"
```


Git Reset

Git can save you from certain death.



```
$ git reset --hard HASHCODE
```

```
$ git reset --soft HASHCODE
```

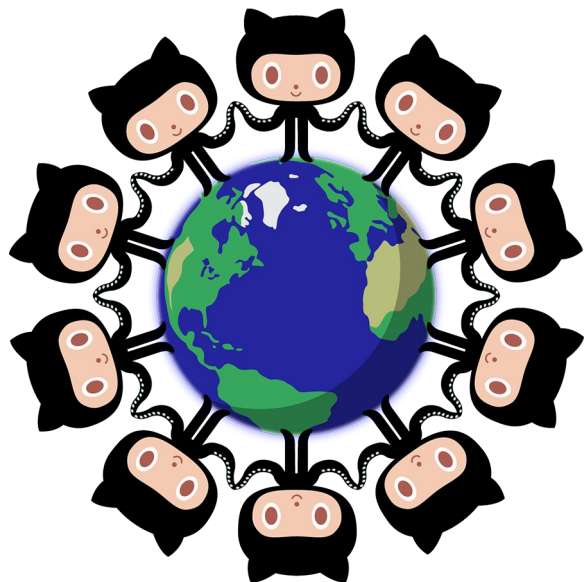
```
$ git reset HEAD
```

git to da choppah!
git into the flow



Git Flow?

Follow the flow for better development



- If you stick to the flow you will speed things up
- Focus on one feature
- Commit often
- Merge things and redo it.

```
$ git checkout -b feature/my-feature  
$ git add .  
$ git commit -m "My Awesome new feature ✨"  
$ git checkout master  
$ git merge feature/my-feature  
$ git push
```

Thank you



Now you should git it too