

GIT UP GIT GIT

---

**GIT DOWN**

(Adapted from a lecture by TA Hall-of-Famer Dylan Lederle-Ensign)

git is "a free and open source distributed version control system"

git-scm.com

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



**Learn Git in your browser for free with Try Git.**



### About

The advantages of Git compared to other source control systems.



### Documentation

Command reference pages, Pro Git book content, videos and other material.



### Downloads

GUI clients and binary releases for all major platforms.



### Community

Get involved! Bug reporting, mailing list, chat, development and more.



**Pro Git** by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

### Companies & Projects Using Git

Google

facebook

Microsoft

twitter

LinkedIn

NETFLIX



ANDROID



RAILS



Qt



GNOME



eclipse



This open sourced site is hosted on GitHub.  
Patches, suggestions and comments are welcome.

Git is

Originally built by Linus Torvalds to manage the development of the Linux kernel.

The scale and complexity of the Linux project requires a globally-distributed, fast, and flexible version control system.



Yoshi's cousin?

# Linux

**Q: WHAT IS VERSION CONTROL?**



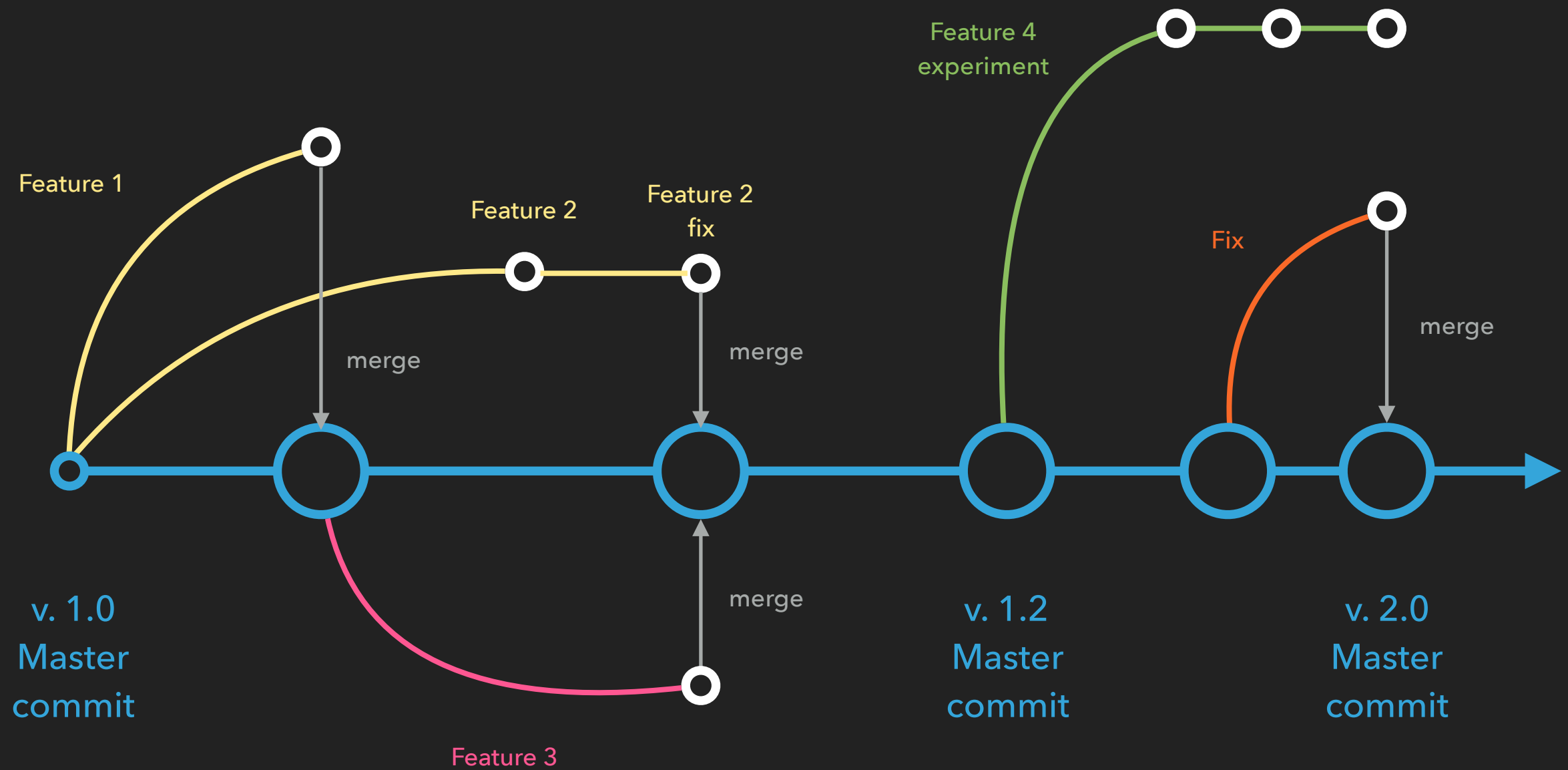
**Version control** tracks and manages your software (and other) project history, including collaboration with team members.

# LIFE BEFORE VERSION CONTROL

game-backup.js  
game.js  
gameNEW.js  
gameNEWv2.js  
gameNEWv2.1.js  
gameRealNew.js  
game-fixes-v2.js  
game-final.js  
game-final-fantasy-xv.js  
game-final-dev.js  
game-release.js  
game-release-actual.js  
game-FINAL-deathAwaits.js

# WHY USE GIT?

1. I make you use it
2. It makes collaborative coding easier
3. It makes experimental coding easier
4. It makes open source contributions easier
5. "GitHub is my resume"



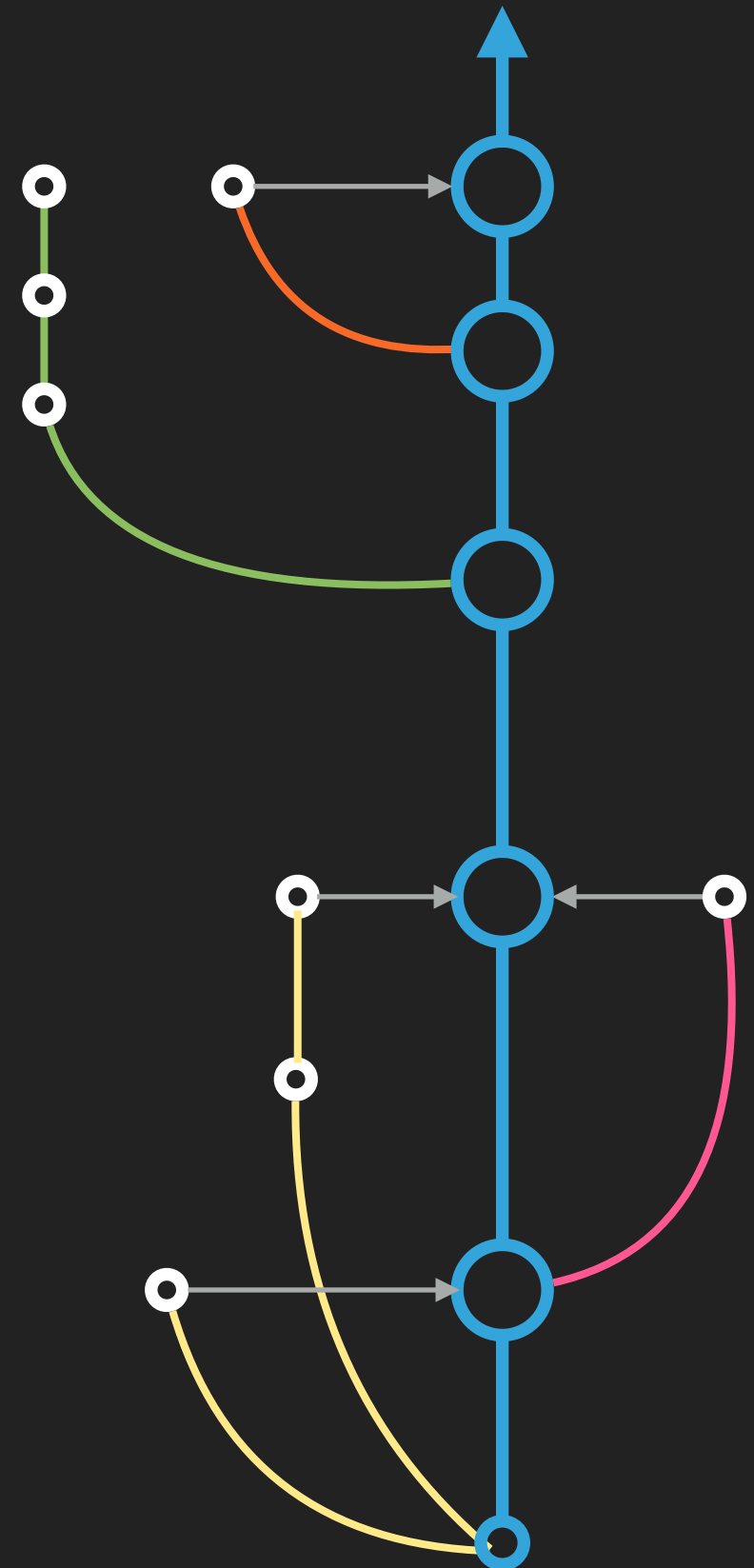
git's "beads on a string" model



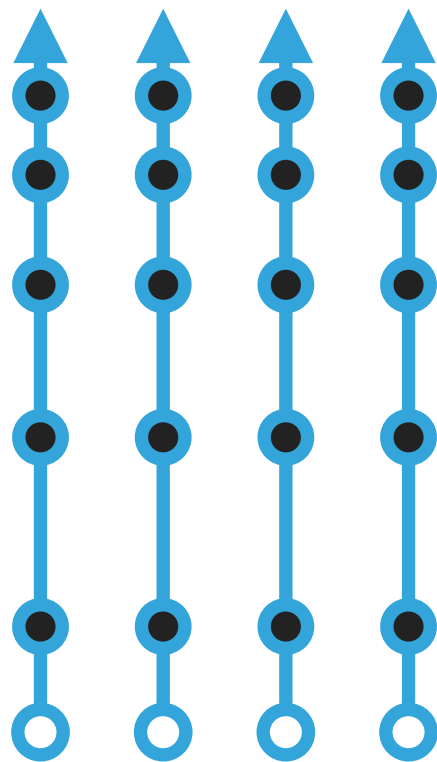
**commit**: add new content, a  
"snapshot in time"

**branch**: new segment of  
development history

**merge**: bring changes from one  
branch into another



# GIT AND PROJECT ORGANIZATION



Git is **distributed**, **non-hierarchical**, and **flexible**.

Every repository contains **all** of the development history that has been committed to it.

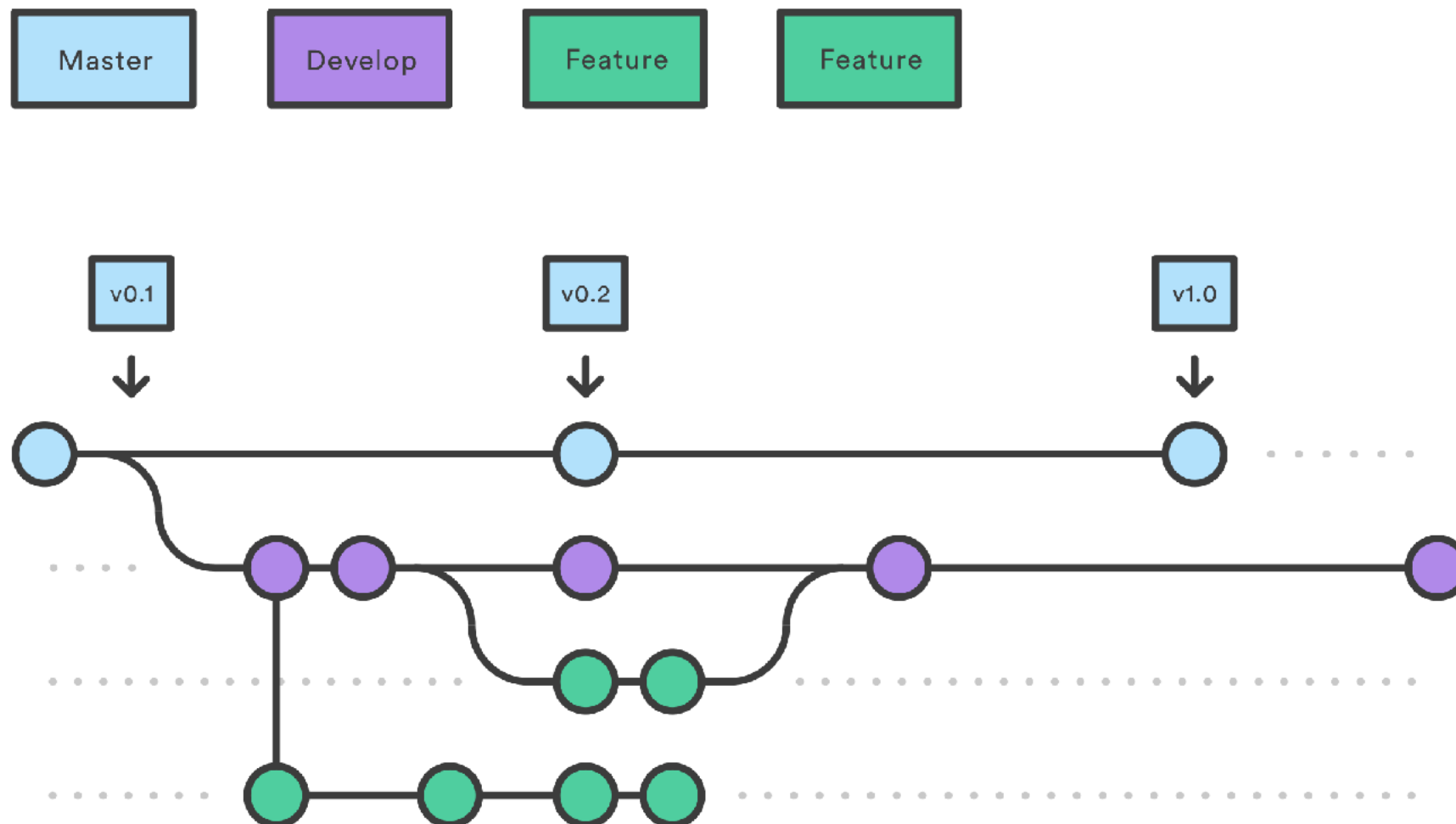
Git does not impose a branching model on your project structure.

However...

# ...WE WILL USE A PATTERN CALLED GITFLOW

Basic structure:

- **Master** is always demoable
- **Develop** is functional but untested
- **Feature-*n*** is a specific task



[ Check out this [atlassian tutorial](#) ]

# GOOD PRACTICES FOR FEATURE BRANCHES

Yep:

`feature-add_scrolling`

`feature-implement_physics`

`feature-level_data_input`

`feature-enemy_prefabs`

`feature-menu_state`

Nope:

`todds-rad-branch`

`feature-things`

`tmpbranch`

`gitSux`

`adlfkadflkasdjflaksdj`

Pretty much everything you need to know about git for class is in this nice tutorial :)

[git-scm.com/docs/gittutorial](https://git-scm.com/docs/gittutorial)

## About

### Documentation

Reference  
Book  
Videos  
External Links

## Blog

## Downloads

## Community

Version 2.13.1 ▾ gittutorial last updated in 2.13.1

## NAME

gittutorial - A tutorial introduction to Git

## SYNOPSIS

```
git *
```

## DESCRIPTION

This tutorial explains how to import a new project into Git, make changes to it, and share changes with other developers.

If you are instead primarily interested in using Git to fetch a project, for example, to test the latest version, you may prefer to start with the first two chapters of [The Git User's Manual](#).

First, note that you can get documentation for a command such as `git log --graph` with

```
$ man git-log
```

or:

```
$ git help log
```

With the latter, you can use the manual viewer of your choice; see [git-help\[1\]](#) for more information.

It is a good idea to introduce yourself to Git with your name and public email address before doing any operation. The easiest way to do so is:

```
$ git config --global user.name "Your Name Comes Here"
$ git config --global user.email you@yourdomain.example.com
```

## Importing a new project

Assume you have a tarball project.tar.gz with your initial work. You can place it under Git revision control as follows.

```
$ tar xzf project.tar.gz
$ cd project
$ git init
```

Git will reply

```
Initialized empty Git repository in .git/
```

# FUN TIMES ON THE COMMAND LINE

\$ git init = initialize new repository (a hidden directory called .git)

\$ git add . = add everything in directory, i.e. 'staging' (note the period!)

\$ git commit -m "NDA - initial master commit" = commit with message

\$ git log = view the history of all changes made to repository

\$ git branch movement-system = create new branch

\$ git branch = display a list of existing branches

\$ git checkout -b movement-system = "check out," i.e., switch to, a branch

\$ git merge movement-system = merge branch into master

\$ git status = where we're at, what's going on

```
git-test — -bash — 80x24
[NateBook-Pro:git-test nathanaltice$ touch main.js index.html ]
[NateBook-Pro:git-test nathanaltice$ git init ]
Initialized empty Git repository in /Users/nathanaltice/Dropbox/CM120/git-test/.
git/
[NateBook-Pro:git-test nathanaltice$ git add . ]
[NateBook-Pro:git-test nathanaltice$ git commit -m "Initial commit" ]
[master (root-commit) 29585e1] Initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
 create mode 100644 main.js
NateBook-Pro:git-test nathanaltice$
```

[ Let's do a git demo ]

# GIT $\neq$ GITHUB

**git** is an open source, command line version control system.

**GitHub** is VC-funded startup that provides git repository hosting along with a social web interface for managing projects.

## IT MATTERS WHO OWNS THE TOOLS



Three key GitHub features:

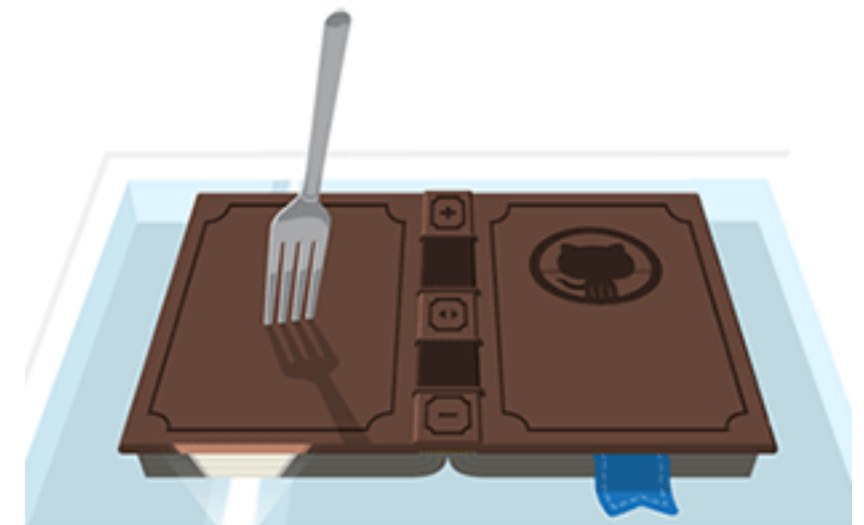
- Forking
- Pull requests
- Octocat?



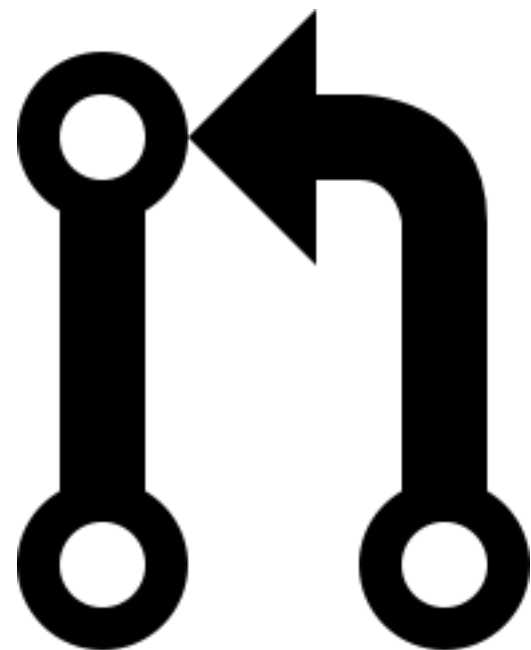
"A **fork** is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea."

GitHub Help



VERY LITERAL PUN



"Pull requests let you tell others about changes you've pushed to a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before the changes are merged into the repository."

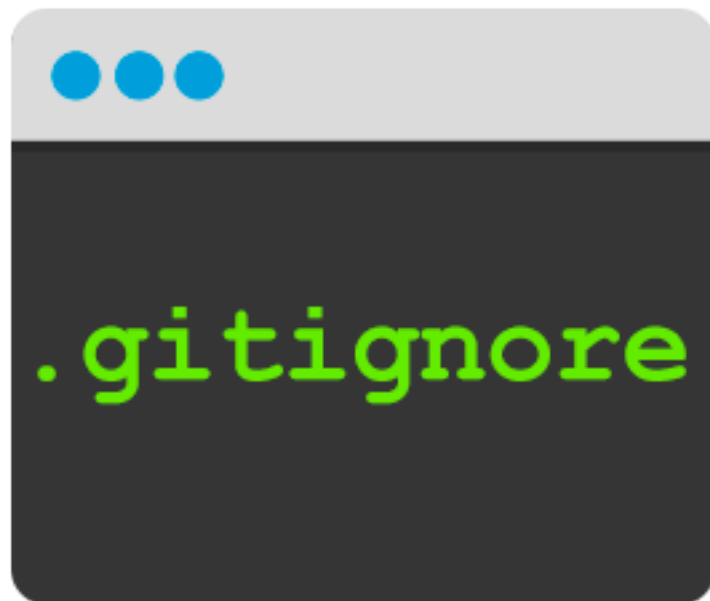
GitHub Help

For class:

Forking ensures that no one team member is the "central repo."

Pull requests ensure that your team has built-in code review.





**Important:** git is great at version control for text, but not so great for binary files. In general, we want track changes to our source, not our assets. To do so, we use a `.gitignore` file. This file is a set of rules that tells git which files to ignore before a commit. **Be sure to do this first!**



octocat / .gitignore

Created 3 years ago

★ Star

1,506

🍴 Fork

476



&lt;&gt; Code

🔗 Revisions 1

★ Stars 1506

🍴 Forks 476

Embed ▾

&lt;script src="https://gist."



Download ZIP

Some common .gitignore configurations

&lt;&gt; .gitignore

Raw

```
1  # Compiled source #
2  #####
3  *.com
4  *.class
5  *.dll
6  *.exe
7  *.o
8  *.so
9
10 # Packages #
11 #####
12 # it's better to unpack these files and commit the raw source
13 # git has its own built in compression methods
14 *.7z
15 *.dmg
16 *.gz
17 *.iso
18 *.jar
19 *.rar
20 *.tar
21 *.zip
22
23 # Logs and databases #
24 #####
25 *.log
26 *.sql
27 *.sqlite
28
29 # OS generated files #
30 #####
31 .DS_Store
32 .DS_Store?
33 ._*
34 .Spotlight-V100
35 .Trashes
36 ehthumbs.db
37 Thumbs.db
```

[ GitHub's sample .gitignore ]



# Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#)[Start a project](#)

[dlederle](#) forked [iscarabaid/CMPPM120](#) to [dlederle/CMPPM120](#) on May 10

★ [calebAvaldez](#) starred [iscarabaid/CMPPM120](#) on May 8

★ [ShadowDai](#) starred [iscarabaid/CMPPM120](#) on May 7

★ [cyreb7](#) starred [iscarabaid/CMPPM120](#) on May 7

💡 **ProTip!** Edit your feed by updating the users you [follow](#) and repositories you [watch](#).

[Subscribe to your news feed](#)



**GitHub Universe**  
October 10-12 in San Francisco



Your repositories **1**

[New repository](#)

[All](#) [Public](#) [Private](#) [Sources](#) [Forks](#)

[CMPPM120](#)

