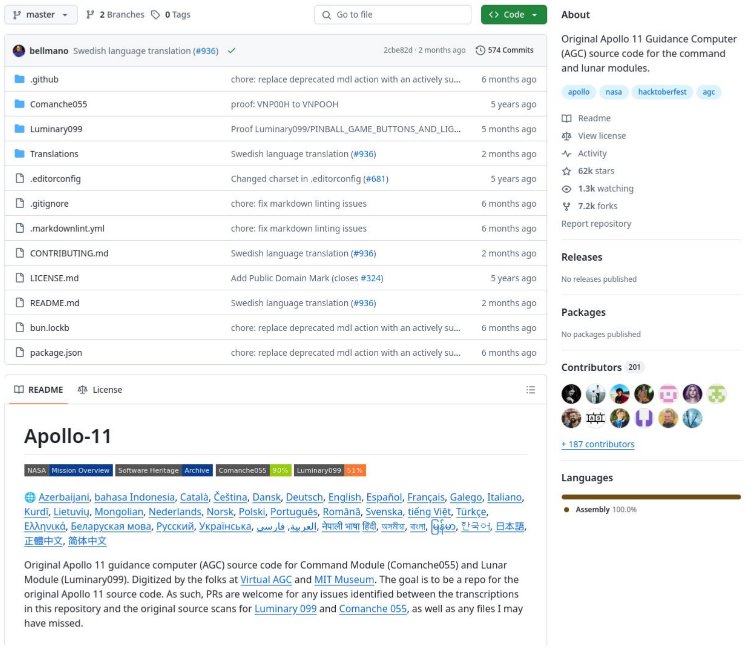
Git and GitHub for TDD and Collaborative Development

99-520

From Source Code to Version Control



master · 2 Branches · 0 Tags · Go to file · Code

bellmano Swedish language translation (#936) ✓ 2cbe82d · 2 months ago · 574 Commits

.github	chore: replace deprecated mdl action with an actively su...	6 months ago
Comanche055	proof: VNP00H to VNPOOH	5 years ago
Luminary099	Proof Luminary099/PINBALL_GAME_BUTTONS_AND_LIG...	5 months ago
Translations	Swedish language translation (#936)	2 months ago
.editorconfig	Changed charset in .editorconfig (#681)	5 years ago
.gitignore	chore: fix markdown linting issues	6 months ago
.markdownlint.yml	chore: fix markdown linting issues	6 months ago
CONTRIBUTING.md	Swedish language translation (#936)	2 months ago
LICENSE.md	Add Public Domain Mark (closes #324)	5 years ago
README.md	Swedish language translation (#936)	2 months ago
bun.lockb	chore: replace deprecated mdl action with an actively su...	6 months ago
package.json	chore: replace deprecated mdl action with an actively su...	6 months ago

README · License

Apollo-11

NASA Mission Overview · Software Heritage Archive · Comanche055 90% · Luminary099 51%

🌐 Azerbaijani, bahasa Indonesia, Català, Čeština, Dansk, Deutsch, English, Español, Français, Galego, Italiano, Kurdî, Lietuvių, Mongolian, Nederlands, Norsk, Polski, Português, Română, Svenska, tiếng Việt, Türkçe, Ελληνικά, Беларуская мова, Русский, Українська, فارسی, العربية, नेपाली भाषा हिंदी, অসমীয়া, বাংলা, မြန်မာ, 한국어, 日本語, 正體中文, 简体中文

Original Apollo 11 guidance computer (AGC) source code for Command Module (Comanche055) and Lunar Module (Luminary099). Digitized by the folks at Virtual AGC and MIT Museum. The goal is to be a repo for the original Apollo 11 source code. As such, PRs are welcome for any issues identified between the transcriptions in this repository and the original source scans for Luminary 099 and Comanche 055, as well as any files I may have missed.

About

Original Apollo 11 Guidance Computer (AGC) source code for the command and lunar modules.

apollo · nasa · hacktoberfest · agc

Readme
View license
Activity
62k stars
1.3k watching
7.2k forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 201

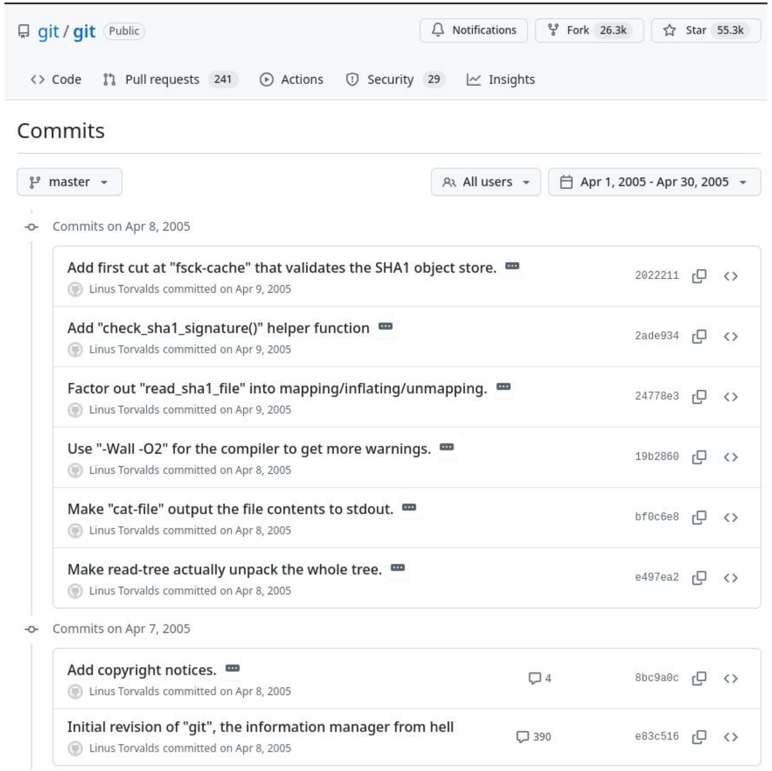
+ 187 contributors

Languages

● Assembly 100.0%

Why Use a Version Control System (VCS)?

Track Changes Over Time



git / git Public

Notifications | Fork 26.3k | Star 55.3k

Code | Pull requests 241 | Actions | Security 29 | Insights

Commits

master | All users | Apr 1, 2005 - Apr 30, 2005

Commits on Apr 8, 2005

Add first cut at "fsck-cache" that validates the SHA1 object store. Linus Torvalds committed on Apr 9, 2005		2022211
Add "check_sha1_signature()" helper function Linus Torvalds committed on Apr 9, 2005		2ade934
Factor out "read_sha1_file" into mapping/inflating/unmapping. Linus Torvalds committed on Apr 9, 2005		24778e3
Use "-Wall -O2" for the compiler to get more warnings. Linus Torvalds committed on Apr 8, 2005		19b2860
Make "cat-file" output the file contents to stdout. Linus Torvalds committed on Apr 8, 2005		bf0c6e8
Make read-tree actually unpack the whole tree. Linus Torvalds committed on Apr 8, 2005		e497ea2

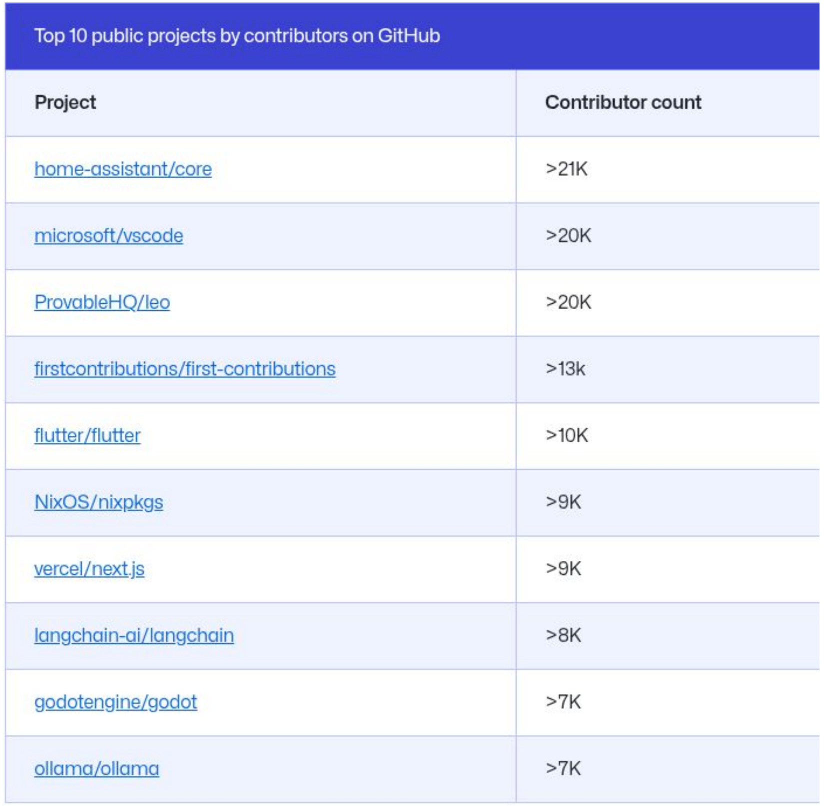
Commits on Apr 7, 2005

Add copyright notices. Linus Torvalds committed on Apr 8, 2005	4	8bc9a0c
Initial revision of "git", the information manager from hell Linus Torvalds committed on Apr 8, 2005	390	e83c516

Why Use a Version Control System (VCS)?

Track Changes Over Time

Collaboration



Top 10 public projects by contributors on GitHub	
Project	**Contributor count**
home-assistant/core	>21K
microsoft/vscode	>20K
ProvableHQ/leo	>20K
firstcontributions/first-contributions	>13k
flutter/flutter	>10K
NixOS/nixpkgs	>9K
vercel/next.js	>9K
langchain-ai/langchain	>8K
godotengine/godot	>7K
ollama/ollama	>7K

https://github.blog/news-insights/octoverse/octoverse-2024/

Why Use a Version Control System (VCS)?

Track Changes Over Time

Collaboration

Other advantages:

- **Branching and Experimentation**
- **Backup and Safety**
- **Transparency**
- **Accountability**

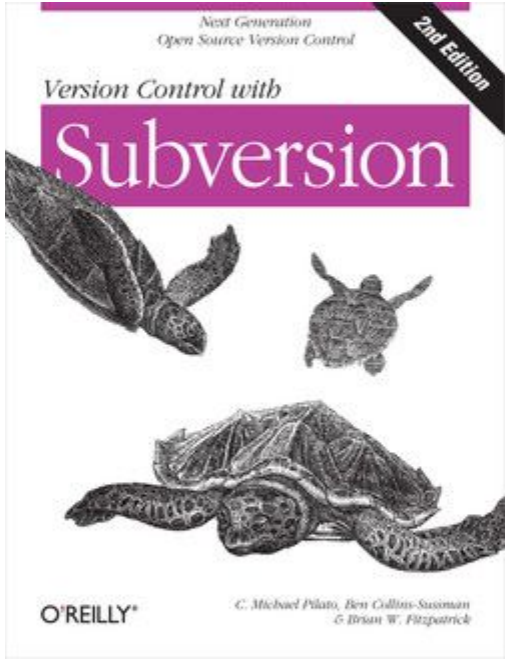
Version Control Systems Power Open Source

“CVS and its semi-chaotic development model have become cornerstones of open-source.”

Ben Collins-Sussman

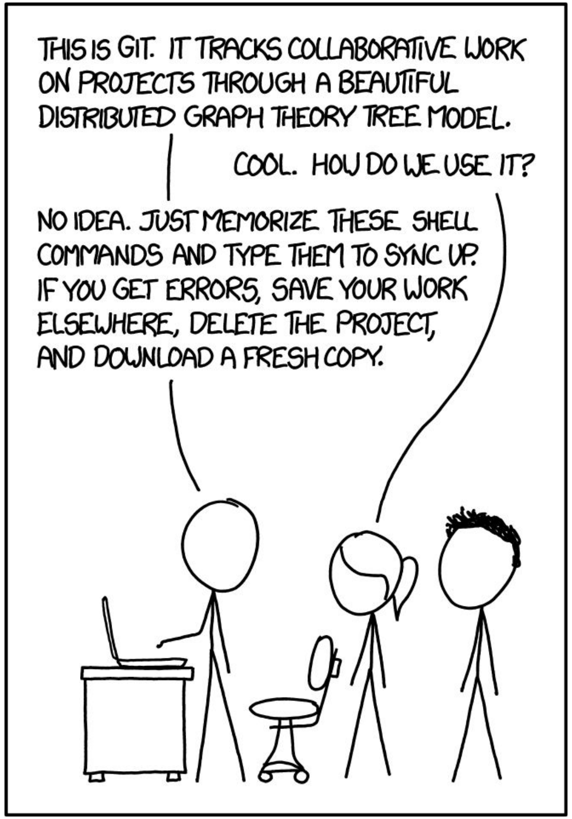
Sourceforge used CVS

Hosted 100,000 FOSS projects



Git

A (very very) brief introduction


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Linus Torvalds on *Why did you create Git?*

*"I really never wanted to do source control management at all ... But then BitKeeper came along and really changed the way I viewed source control. BK got most things right and having a local copy of the repository and distributed merging was a big deal. The big thing about **distributed source control** is that it makes one of the main issues with SCM's go away – the politics around "who can make changes." BK showed that you can avoid that by just **giving everybody their own source repository**. But BK had its own problems, too; … but the biggest downside was the fact that since **it wasn't open source**"*

Linus Torvalds
torvalds

git

Git is a widely used **distributed version control system**, known for its speed, flexibility, and strong support for collaboration through branching and merging.

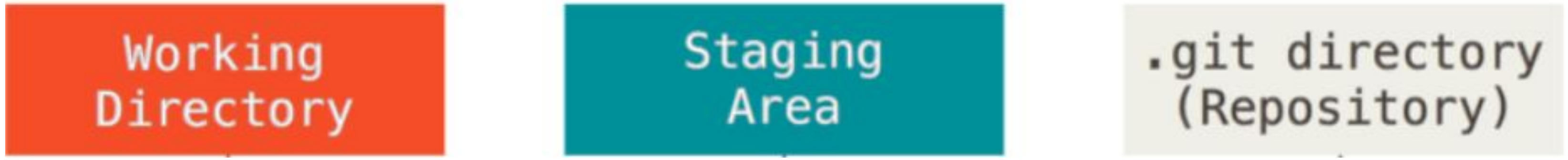
Terminology & Concepts

- Repository (Repo): A project folder tracked by Git.
- Commit: A snapshot of your code at a point in time.
- Clone: Create a local copy of a remote repository.
- Remote: A version of the repo hosted online (e.g., GitHub).
- Branch: A separate line of development.
- Merge: Combine changes from one branch into another.

Getting a Git Repository

- Create one locally
  - `$ git init`
- Or, get a copy from a remote server
  - Example:
    - `$ git clone https://github.com/EduardoFF/git-simple-demo.git`

Directory Map



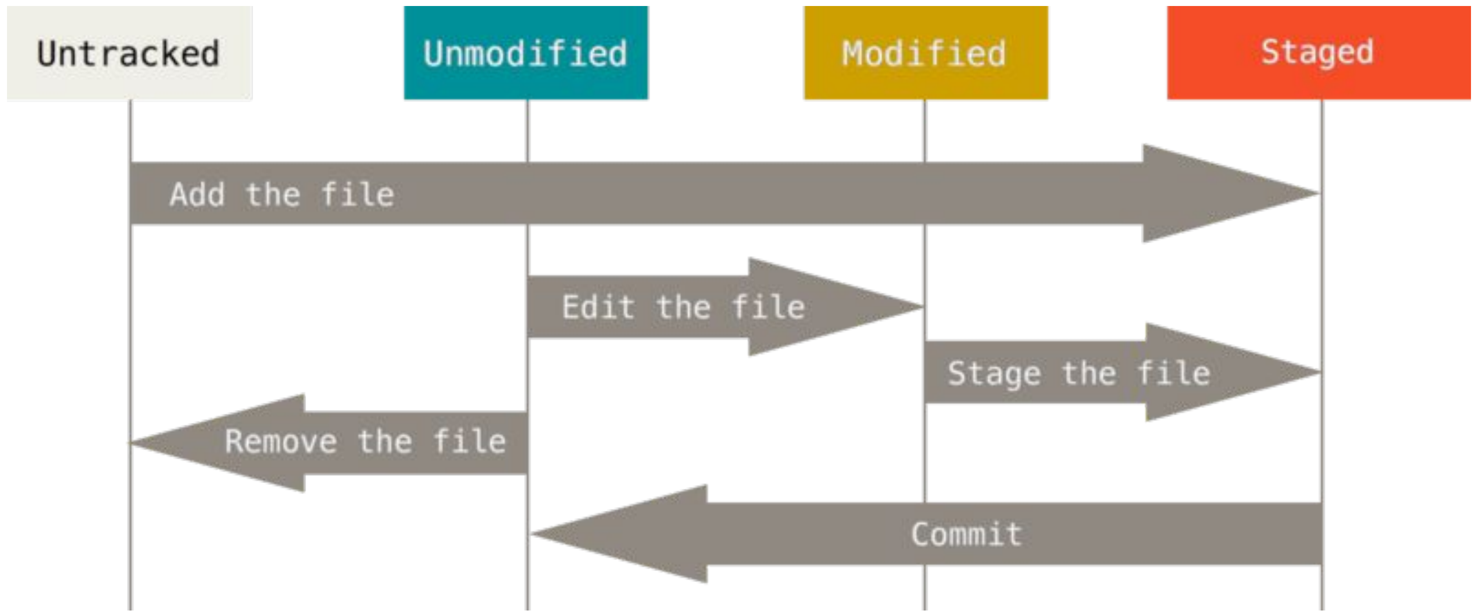
What you see in your file manager

“Virtual” directory

A “database”

Lifecycle of a File

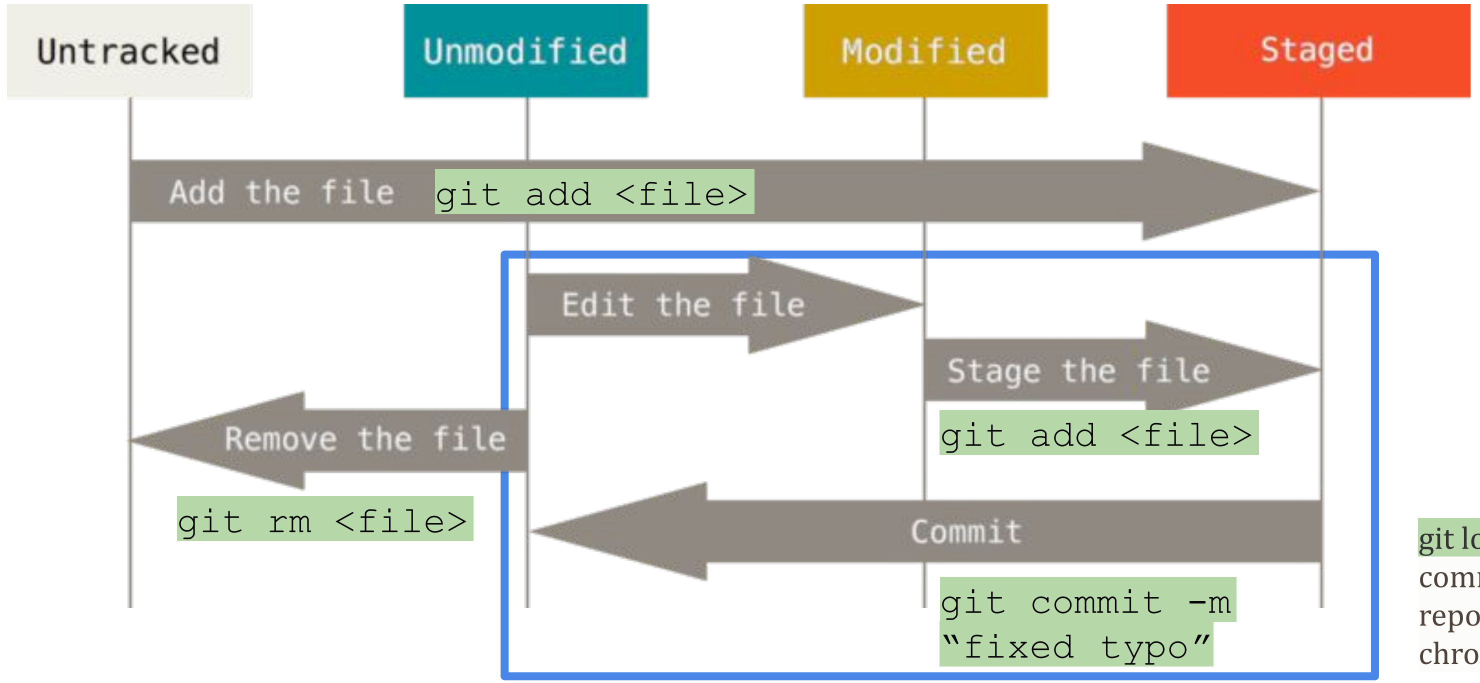
A temporary version of a file before it goes into the tracking history



The main tool you use to determine which files are in which state is the `git status` command.

if the git status command is too vague for you — you want to know exactly what you changed, not just which files were changed — you can use the git diff command.

Lifecycle of a File



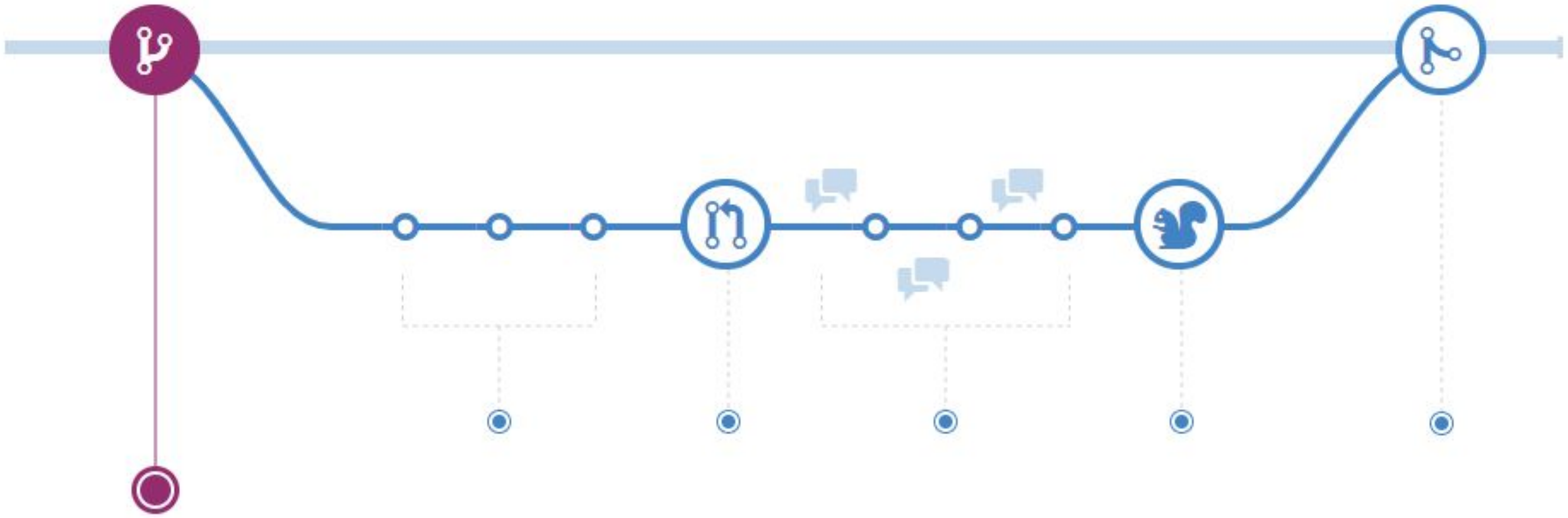
git log lists the commits made in that repository in reverse chronological order

The main tool you use to determine which files are in which state is the `git status` command.

if the git status command is too vague for you — you want to know exactly what you changed, not just which files were changed — you can use the git diff command.

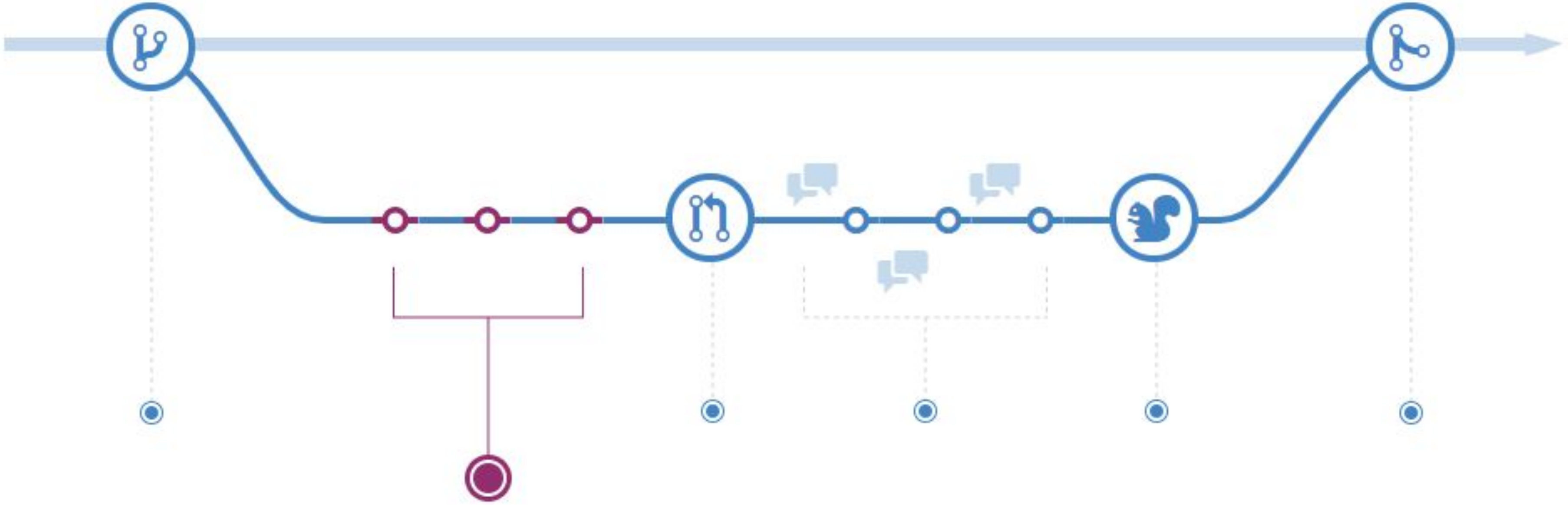
Git Flow

Taken from: Understanding the GitHub Flow
https://guides.github.com/introduction/flow/



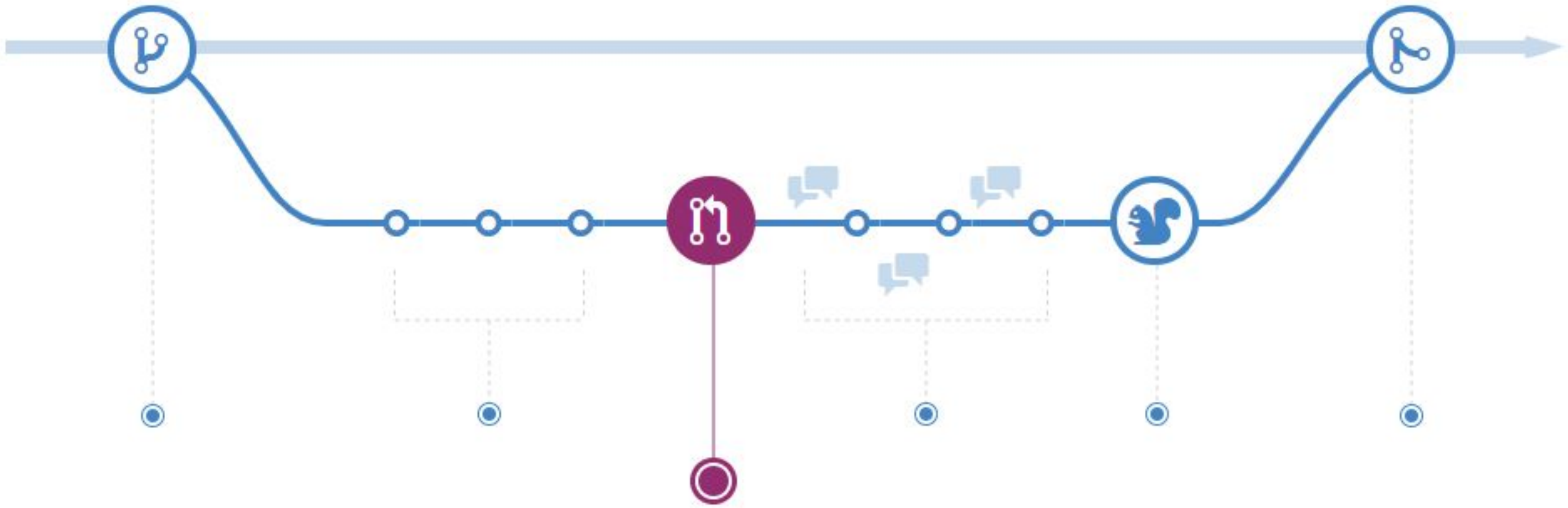
Branching

- Develop features on a separate branch from main
- Branch name should be meaningful "`fix-typo-in-word`"
- `git checkout -b fix-typo-in-word`



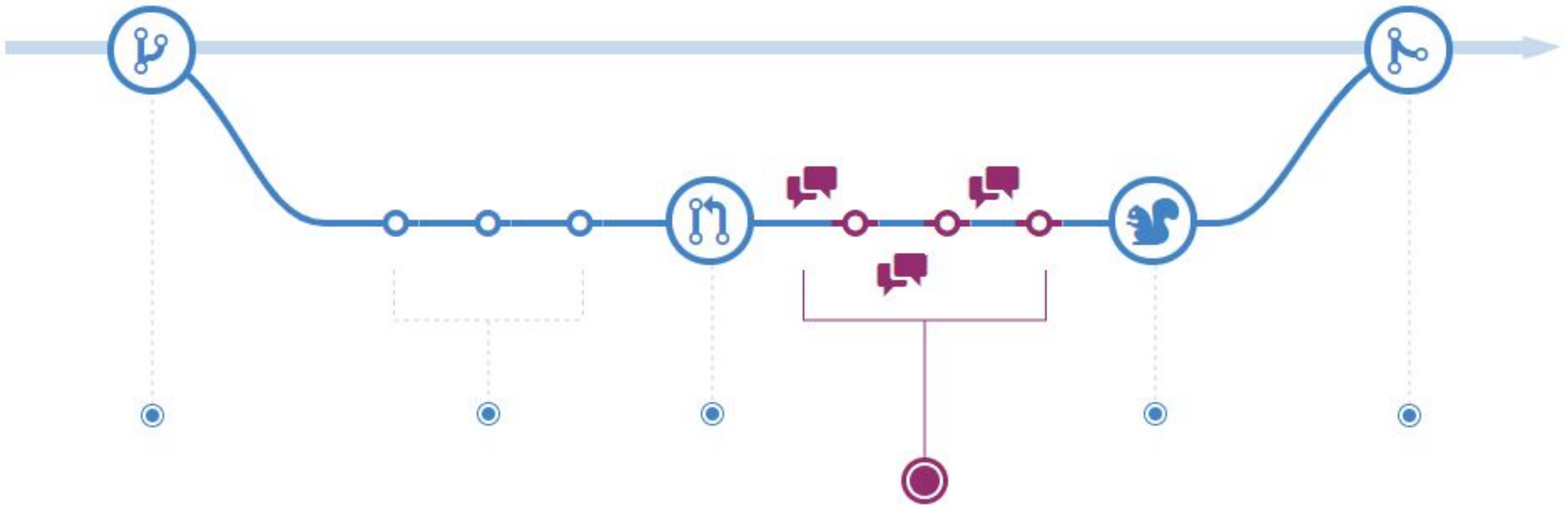
Add Commits

- Commits to create change history
- Commits should be Verb then something. KISS
- Commits can be used to close issues as well ([On GitHub]())



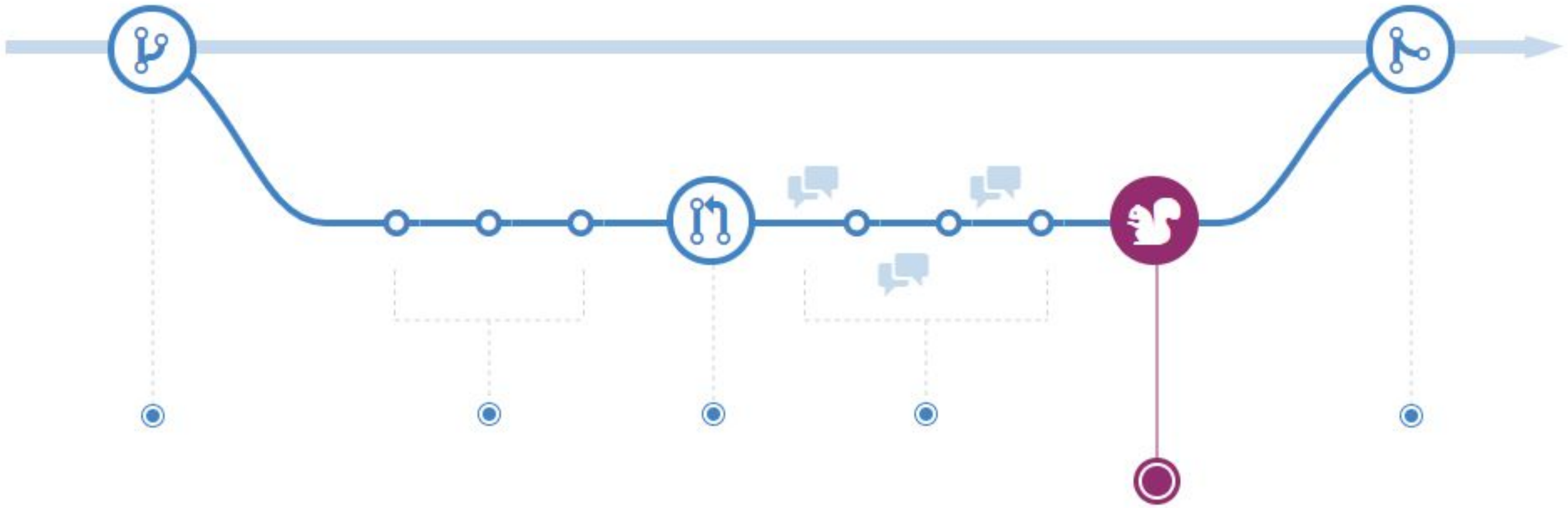
Creating Pull Requests

- Push branch up to prepare for merging with main
- Can be done anytime when writing code
- Title & Description should be meaningful, explain what issue / ticket is fixed by your changes



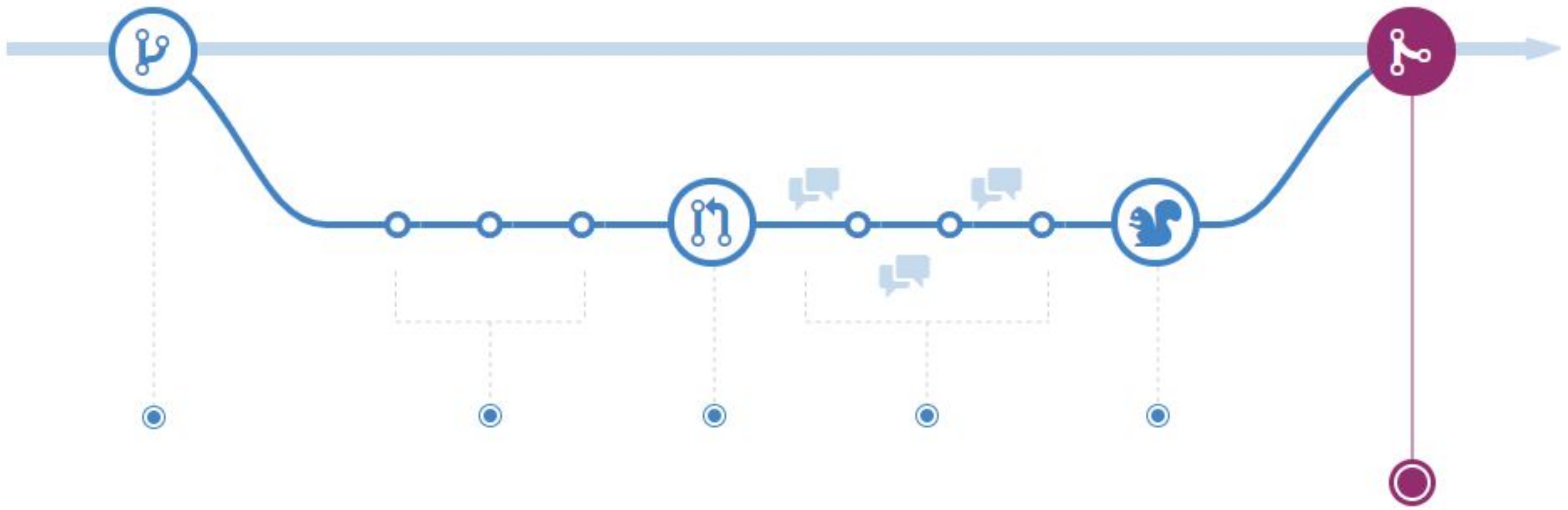
Code Review

- Done when pull request review is requested
- Back and forth process until reviewer approves



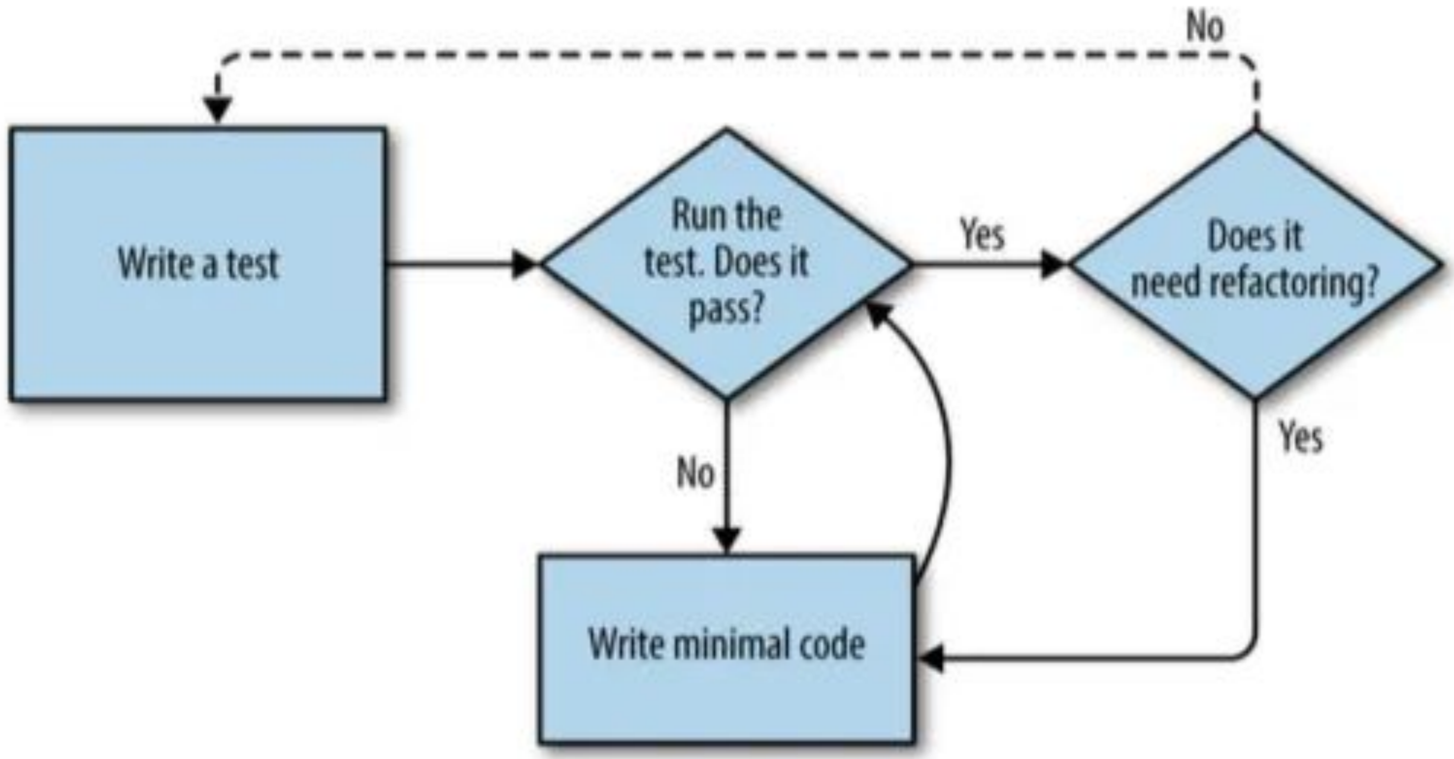
Automated Testing/Deploy

- Most DevOps tools will allow you to set up automated testing
- Run automated testing before merging



Merge to Main!

- Merge to main when your PR is approved and your automated testing passes!


No
Write a test
Run the test. Does it pass?
Yes
Does it need refactoring?
No
Yes
Write minimal code

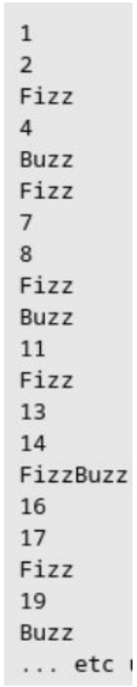
The Rules of TDD

1. Write a failing automated test **before you write any code.**
2. Write (and refactor) the code to pass the tests.

TDD isn’t something that comes naturally. It’s a discipline.

FizzBuzz

Write a function **`fizzbuzz(n)`** that takes an integer n and returns the number converted to string. For multiples of three return **`Fizz`** instead of the number and for the multiples of five return **`Buzz`**. For numbers which are multiples of both three and five return **`FizzBuzz`**.



```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
... etc
```

Tomorrow

We have a team activity (small competition) about TDD and Gitflow

Each of you needs to have:

- A GitHub account
- Git installed on your computer
- Git configured and ready to push changes to GitHub

Make sure everything is set up before class so your team can hit the ground running!

Try now with your team: github.com/EduardoFF/fizzbuzz-tdd-game

New Feature

- Number is also Fizz if it contains a 3 as digit
- Number is also Buzz if it contains a 5 as digit