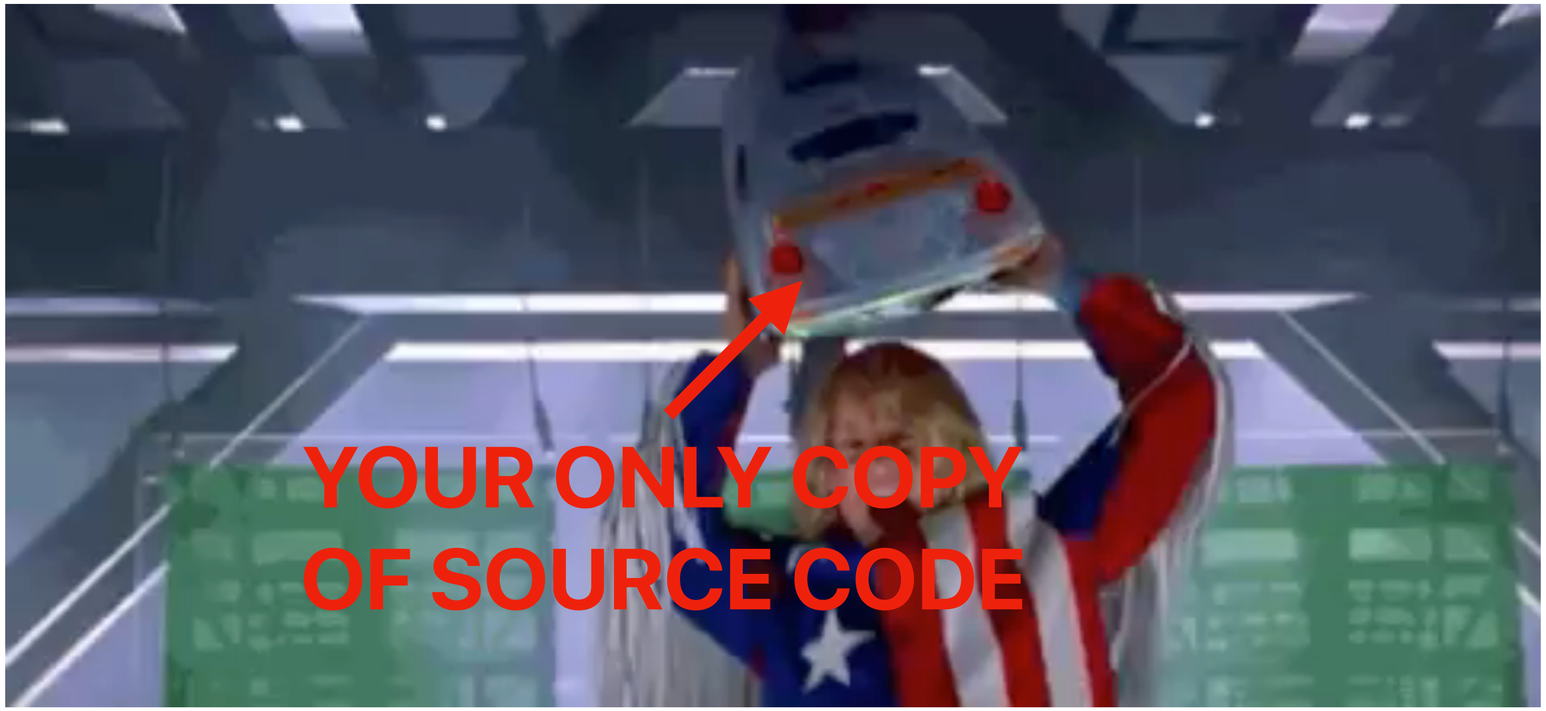
Intro to Git and GitHub

Why version control?



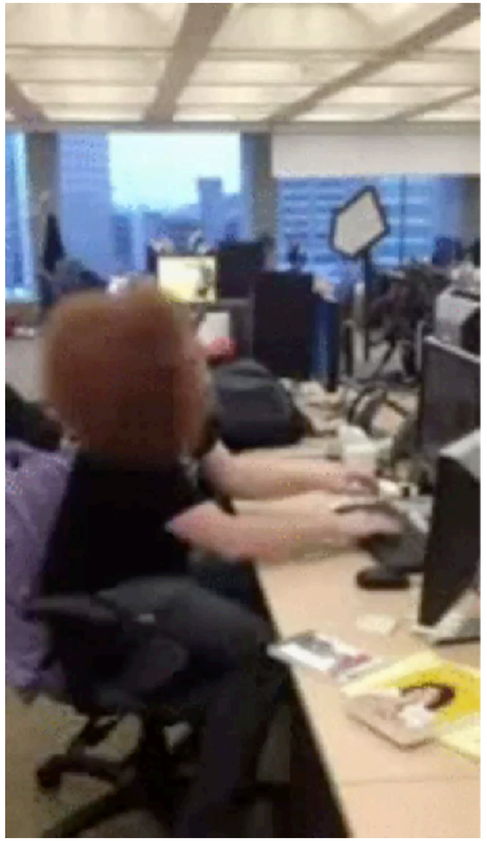
1. **BACKUPS**

Why version control?

2. **UNDO**

Why version control?

+



3. **COLLABORATION**

Why version control?

1. Backups. 💾
    - Hard drive failures, computer crashes, blue screens of death, file corruption.
2. Undo. 💦 🔥
    - Great way of handling new bugs introduced into builds.
3. Collaboration. 👨‍💻+👩‍💻
    - **Branching** - Allows teams to work in parallel, and bring it all together. Prevents the team from overwriting each other’s work.
    - **Tracking** - What changes were introduced while I was sleeping?

What is Git?

- Git is the de facto **version control system** (*aka software*).
- It was made by Linus Torvalds in response to his frustration with other version control software as he was managing the building of Linux kernel.
- Git implements a **method** of version control. Git takes **snapshots** of your folder at the stage you *saved* it. This type of save is explicit within Git and is **not** the saving that is done in the editor.
    - These snapshots are stored in case you need to revert to an older version.

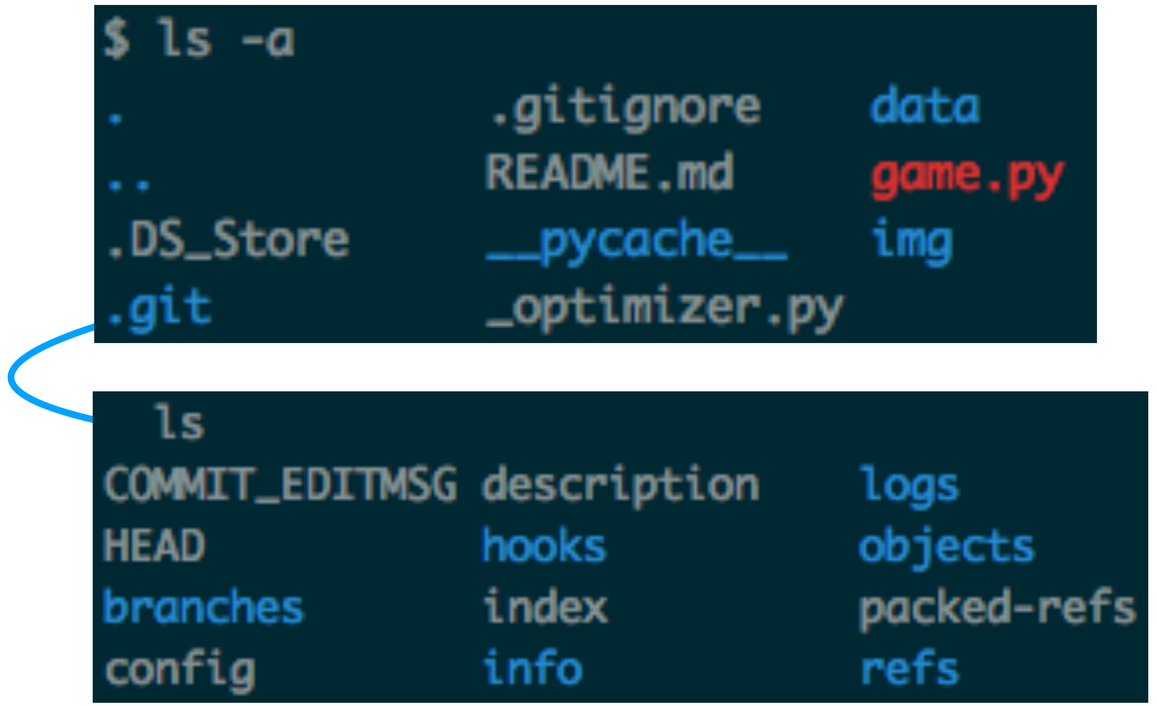
Everything you save using Git is tracked, so say this with me: **I will not save or add sensitive data to Git or Github.**

Questions?

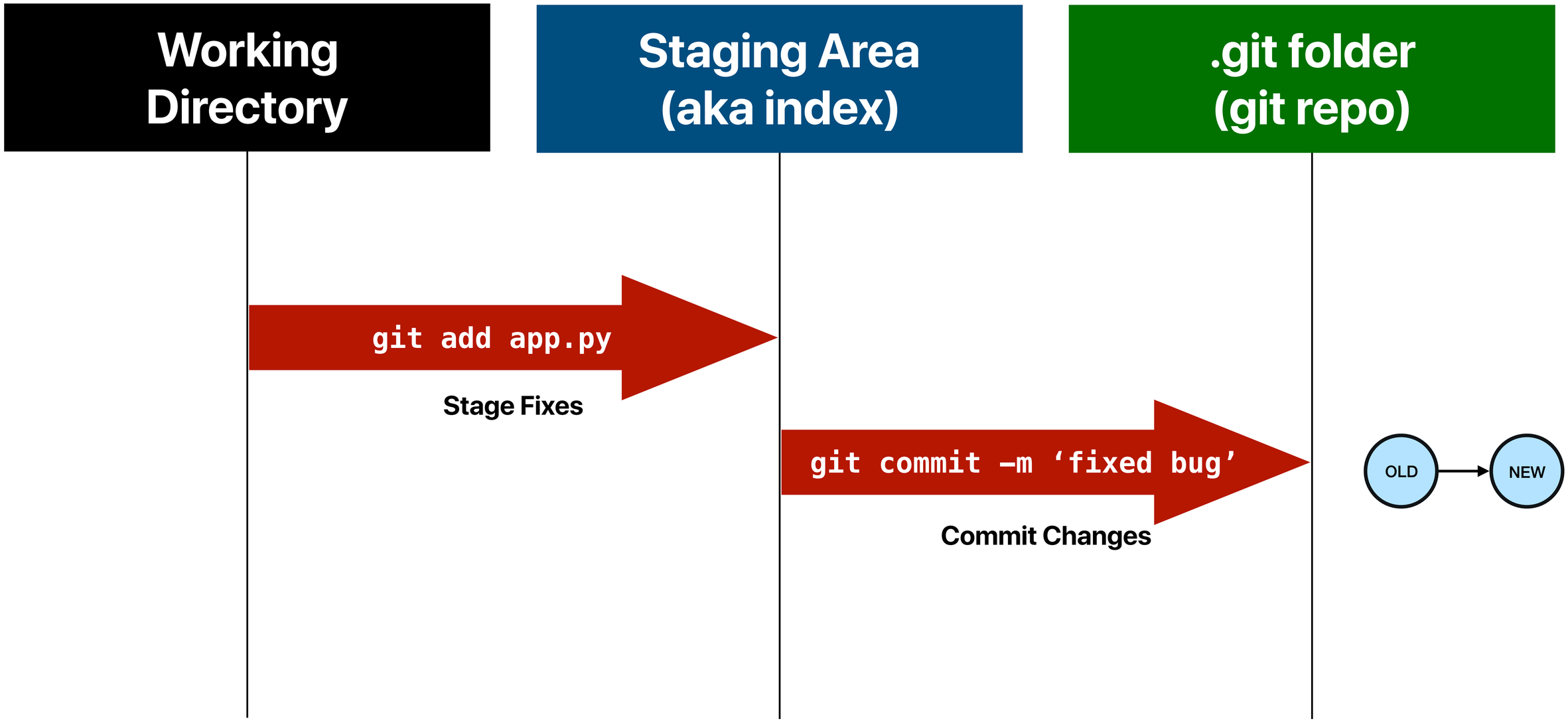

Let’s dive in.

Git Repository

- aka a Git repo.
- This is a local folder that keeps track of changes (it's like a mini-filesystem).
- The Git directory is where Git stores the metadata and object database for your project.
- This is the most important part of Git and what is copied when you clone a repository from another computer.



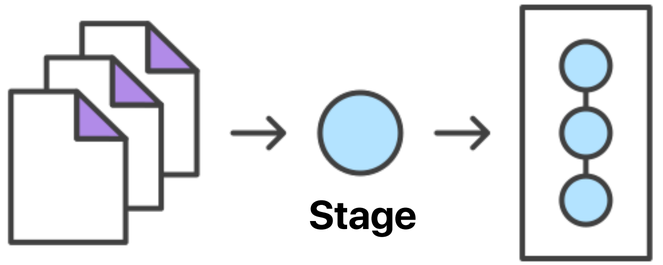
How do we write to the local Git repo?



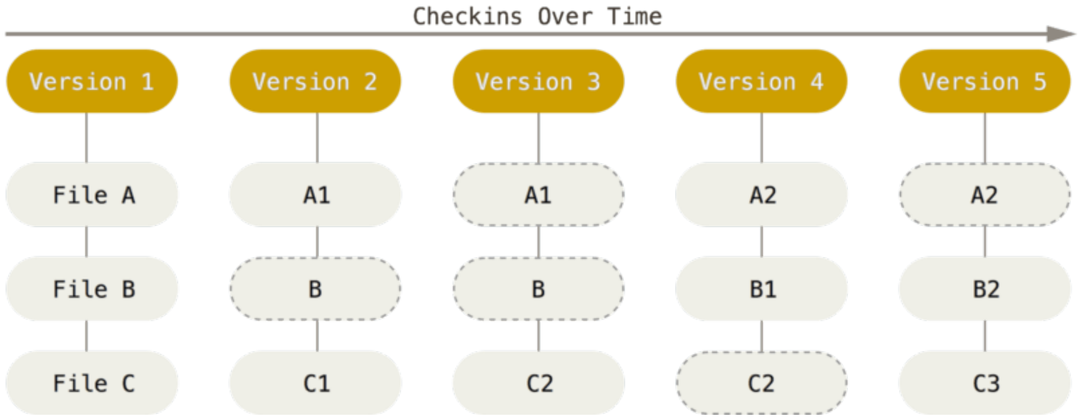

```
/dev/my-awesome-project
   - app.py
   - utils.py
```

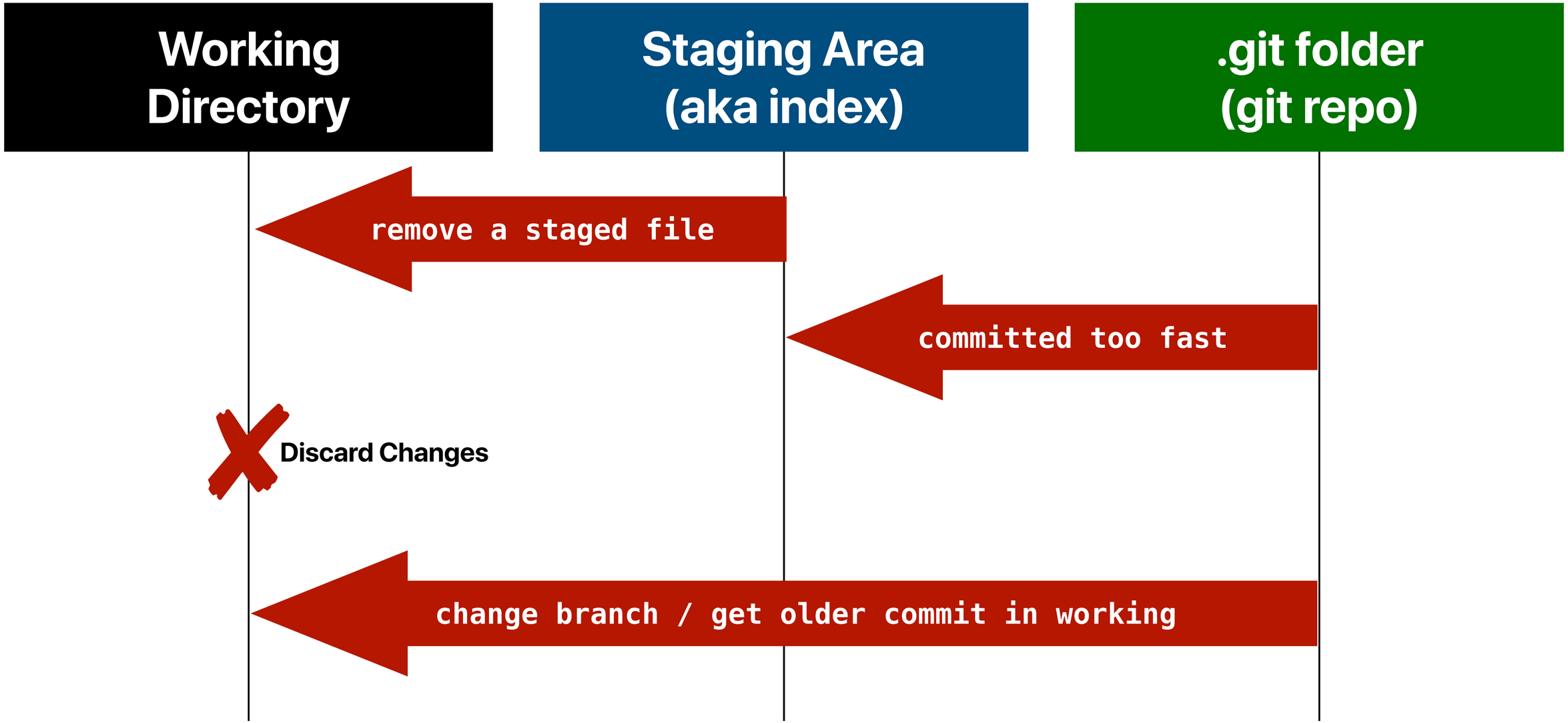
Commits

- These are snapshots of your files where the contents are stored in full.



- When you change things in files, or add or remove files then commit these changes, Git will record the current state of everything.





We'll see some of these later

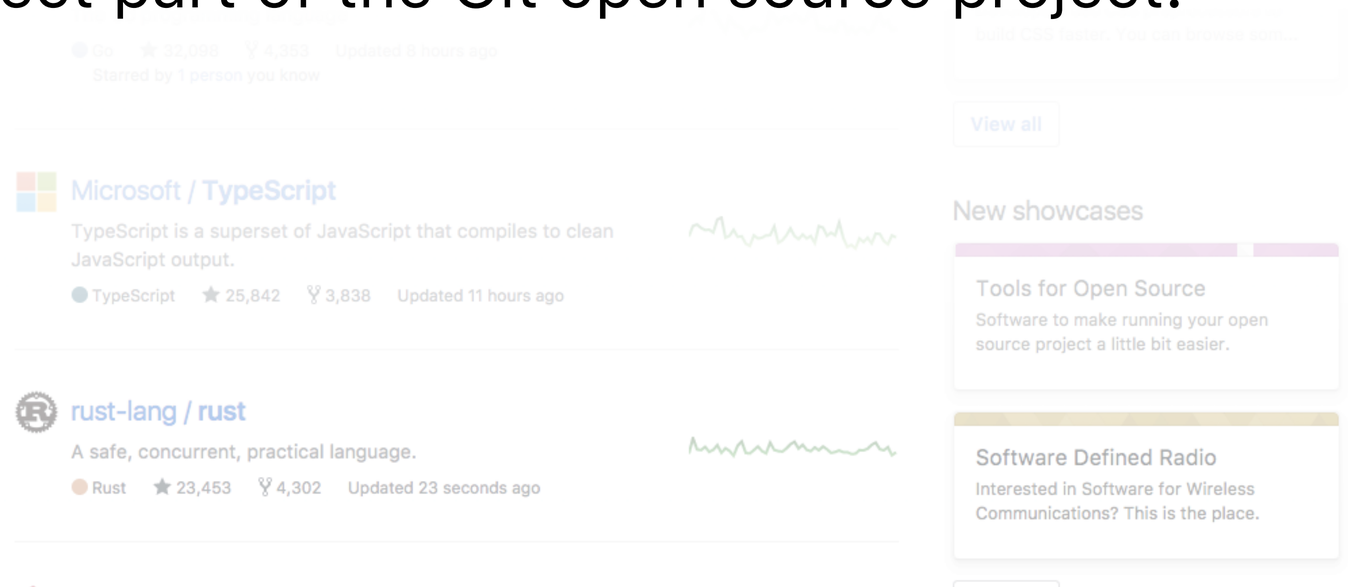
Questions?

GitHub

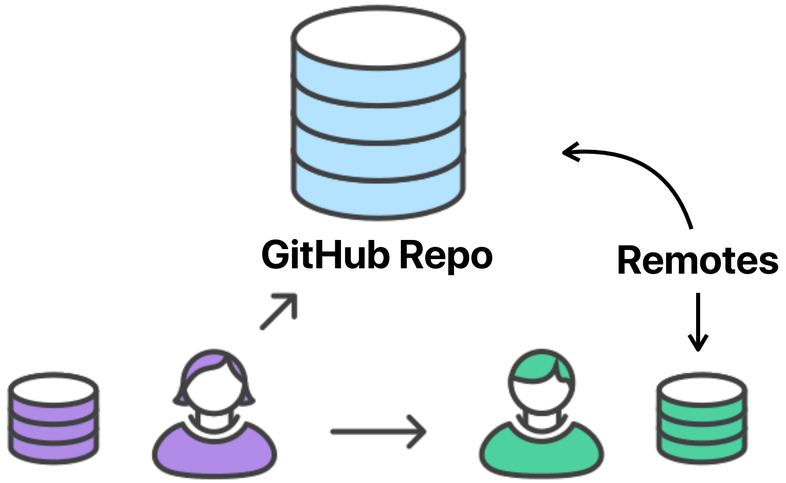
(octocat)

What is GitHub?

- GitHub is the single largest host for Git repositories, and is the central point of collaboration for millions of developers and projects.
- Many open-source projects use it for Git hosting, issue tracking, code review, and other things.
- It’s not a direct part of the Git open source project!



Remotes are copies of a git repo on another computer (or on a service like GitHub).



Clones, branches, forks

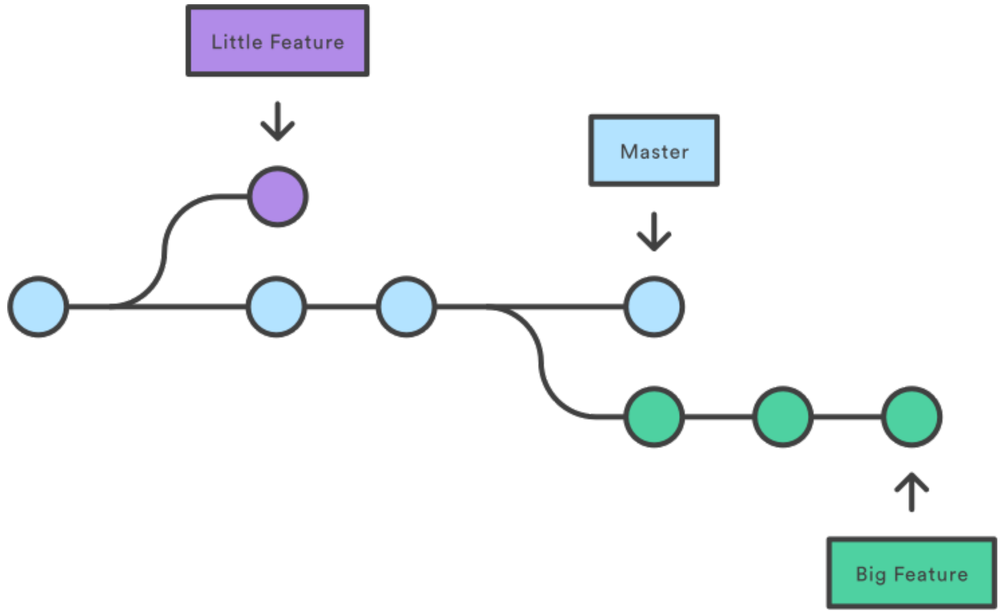
Clone

- A **clone** is simply a copy of a repository.

- No one but you sees any changes that are made to this copy, and it doesn’t automatically refresh when the original version refreshes unless you tell it to.

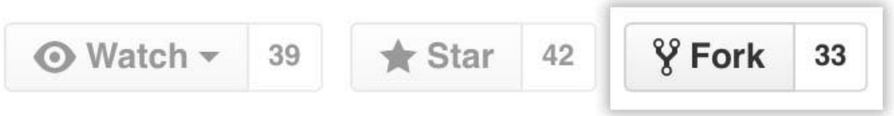
Branch

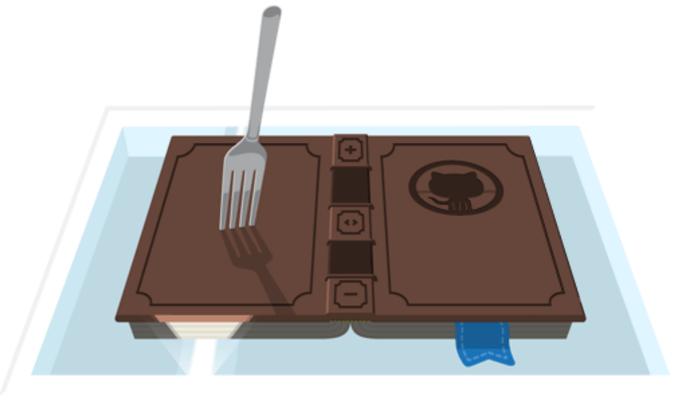
- A **branch** is a seperate thread of development within a repo.
  - Branches are usually temporary, and are intended to work on some small aspect of your main project.
  - Any commits you make will stay on that branch until you merge them with the main branch.
  - Branches can only be made by people authorized in the repository.



Fork

- A **fork** is a public way of making a branch when you don’t have push access to the project.
  - Forking allows anybody make a local copy in their own git repository account.
  - They can then make changes and when finished send a pull request to get the code accepted.





Questions?

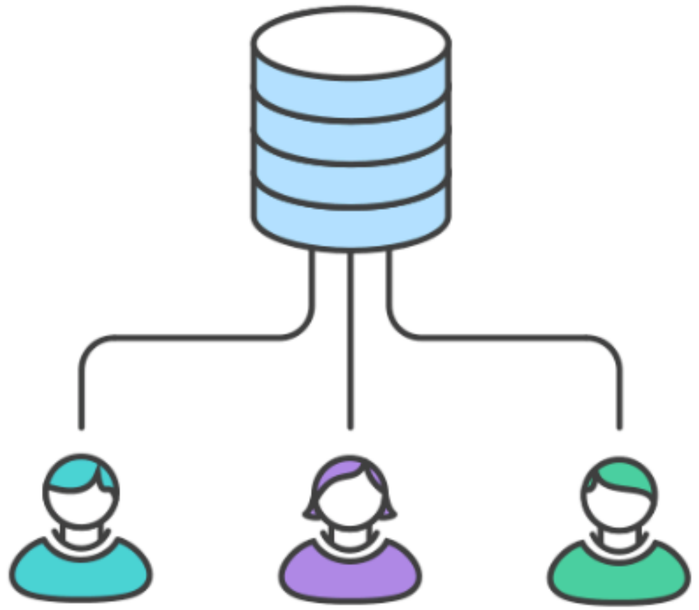
Collaboration with Git

Centralized Workflow

Typically used on small teams

Pros:

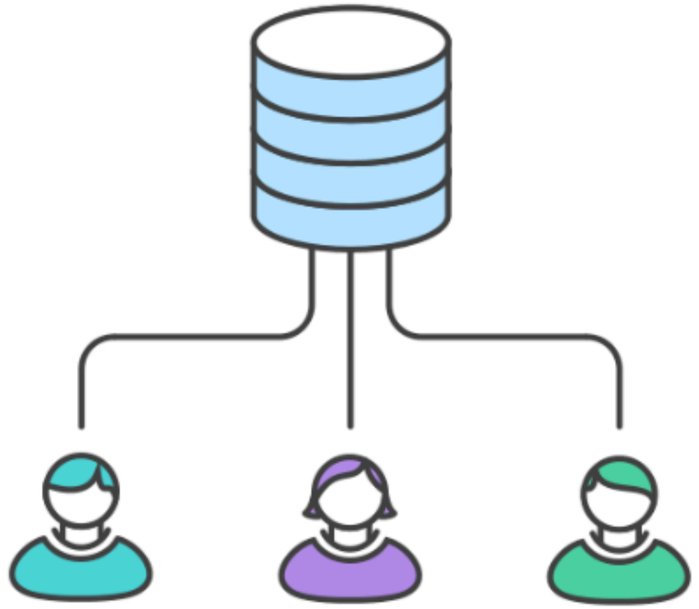
- Everyone can do everything, no bottlenecks.
- Efforts can be organized and coordinated using git branches.
- Github not technically required.



Centralized Workflow

Cons:

- People can make arbitrary decisions and possibly ruin hard work and code quickly.



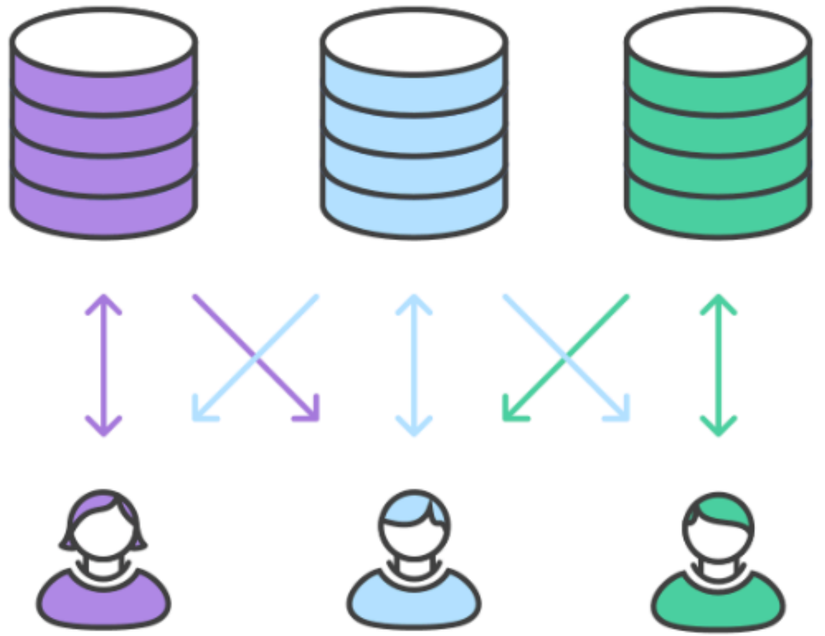
Forking Workflow

Typically used on larger open-source projects

Pros:

- Pull requests serve as places to review code for quality and discuss architecture.

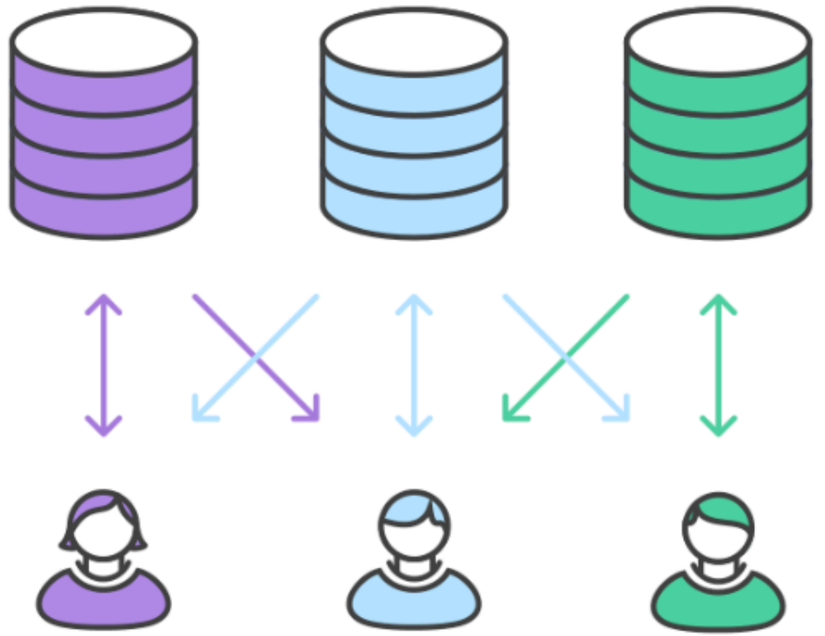
- Forks give people freedom to change the code in a way that suits them.



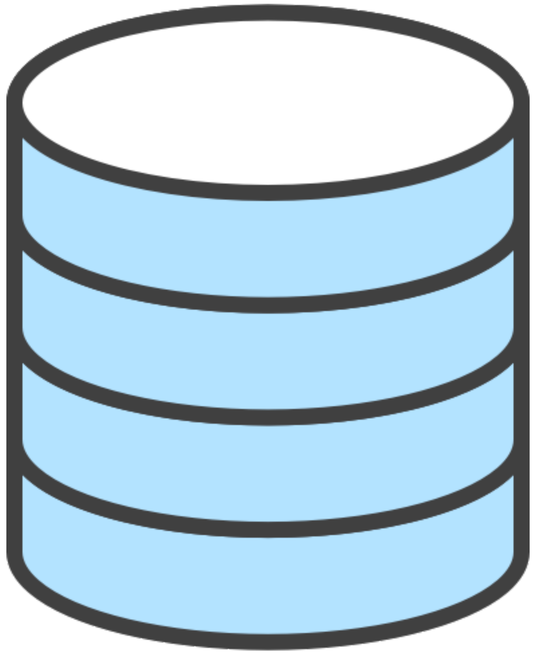
Forking Workflow

Cons:

- Changes must be approved by key contributors.
- Harder to do without services like GitHub.

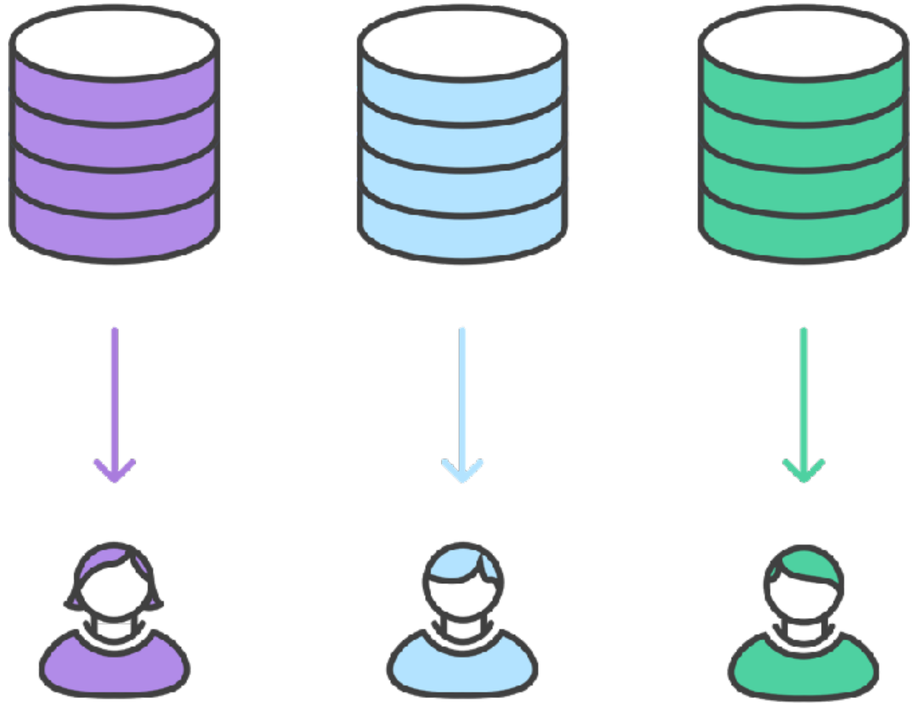


Let’s take a deeper look into the forking workflow.

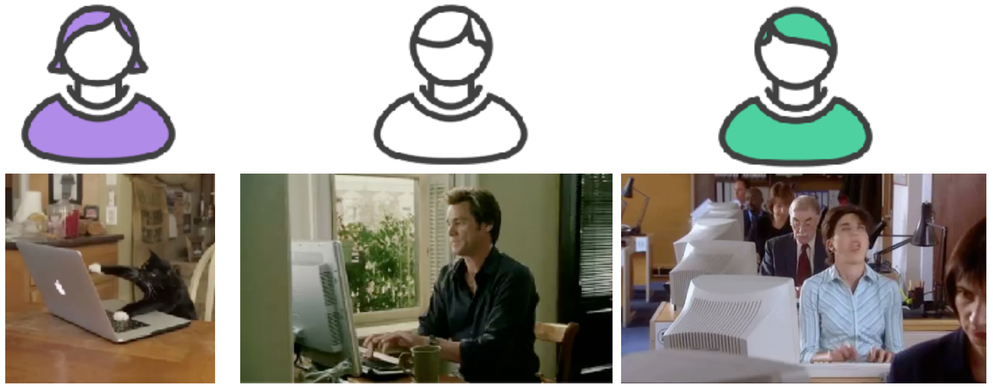


First, we build a repo on GitHub, or our local machine and upload it to GitHub.

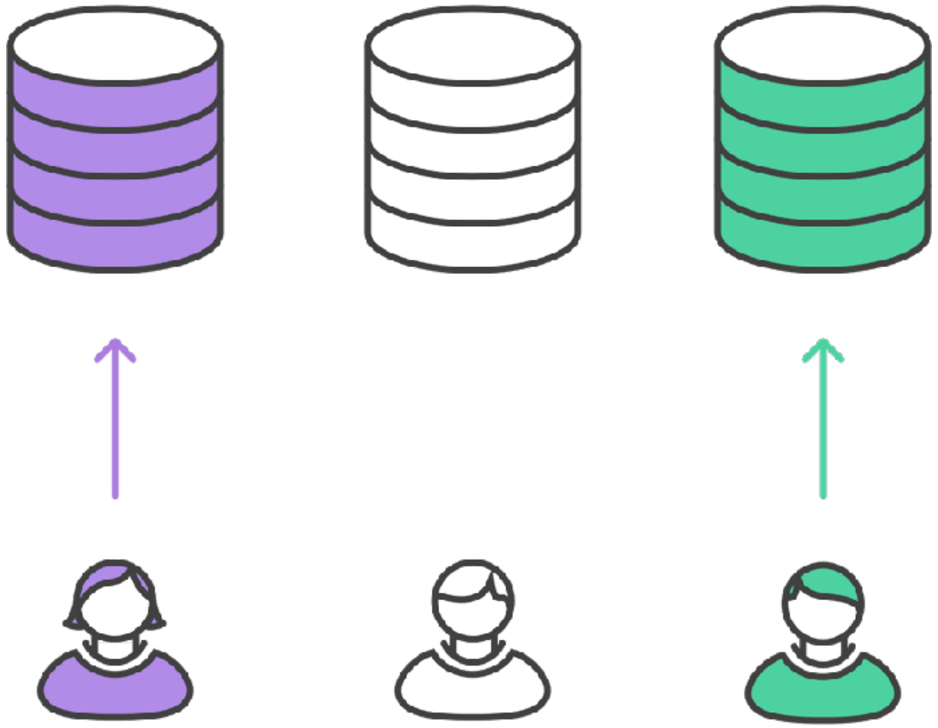
Then we have other contributors make forks of the repo.



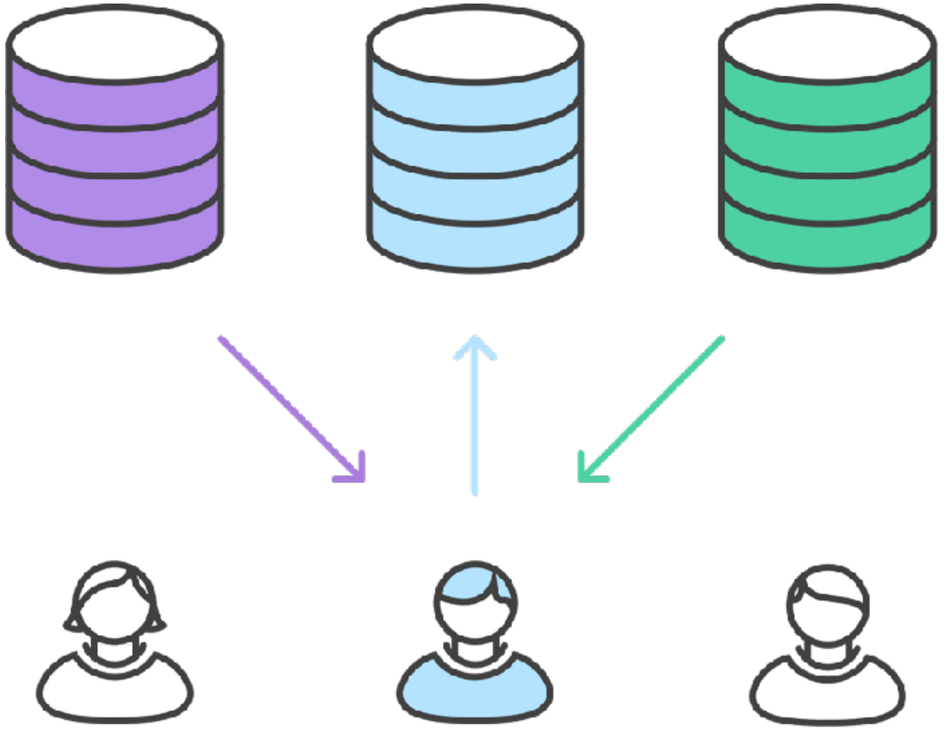
The contributors now clone the forked repos.



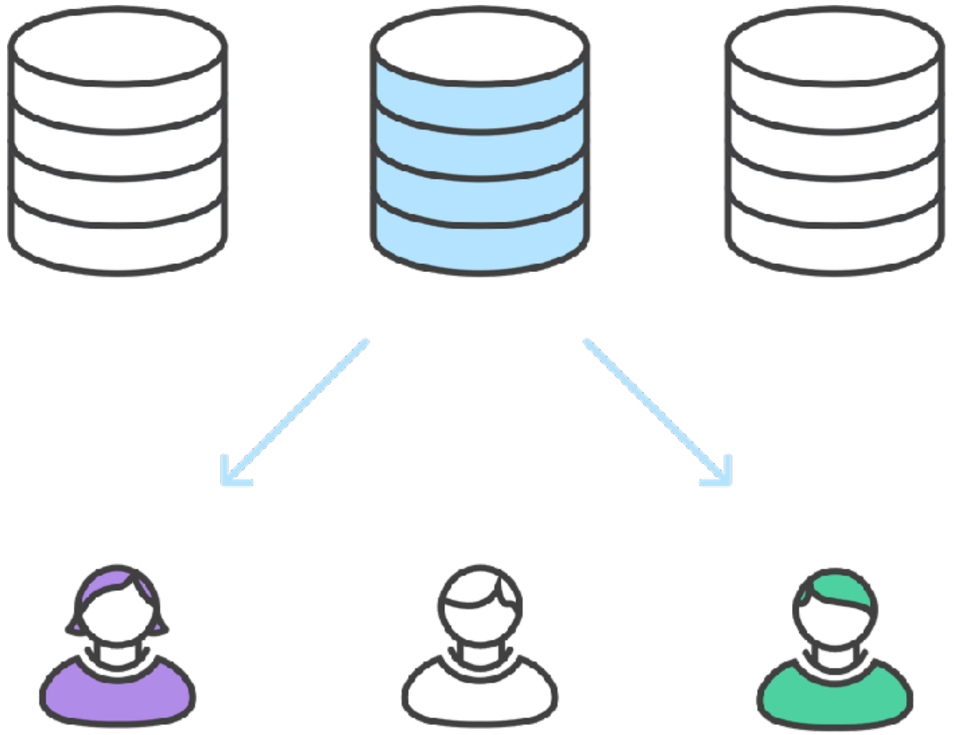
The contributors begin working on their features.



Then push those changes back to their repos.



Developers submit pull requests and now the project maintainer can accept or deny these features, and integrate it back into the main repo.



Once the features have been added, the contributors can now sync up with the latest branch.

Summary

1. Project maintainer creates repo.
2. Contributors fork repo.
3. Contributors clone their forked repo.
4. Everyone builds code and contributes as they wish.
5. Contributors submit a pull request.
6. Project maintainer accepts or denies the pull requests.
7. Project maintainer merges the pull request with the core repo.
8. The contributors can now sync with the latest repo.

Questions?

Let’s put this knowledge to work.