Tools in our utility belt

- Text Editor - Atom
- Shell - git-bash
- Version Control - Git

Why is version control useful?

- **Backup** – coffee + keyboard = 😖
  - *Undo* – Oops, was that important?
- **Collaborate** – programming is a team sport
  - *Track steps* – What happened while you were sleeping
  - *Branching and merging* – Work in parallel and bring it back together

A Git repository or "repo" is basically a folder that keeps track of changes to its contents.

Imperfect analogy: if Git is a camera, the repo is what's in frame

A little on how Git works

The Git index

Git keeps a local copy of everything you add to it, even if you delete it later.

Imperfect analogy: if Git is a camera, the index is a roll of film

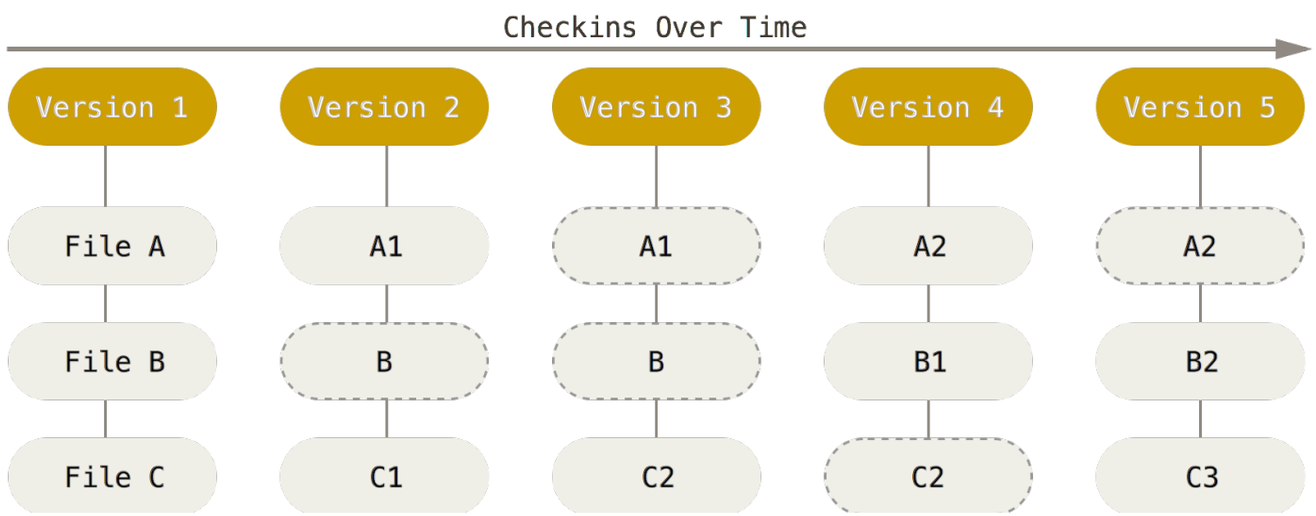
Say it with me.

As a general rule, **don't add sensitive data to Github.**

Commits: snapshots of your files

When you change things in files, or add or remove files, you commit these changes and Git records the current state of everything

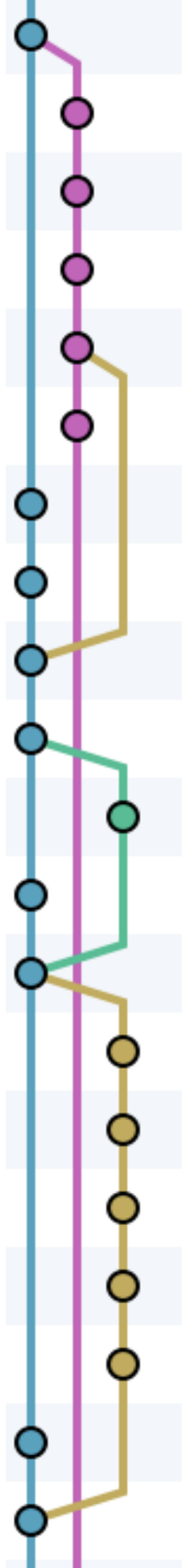
Imperfect analogy: if Git is a camera, commits are pictures



Version history is (kind of) like a family tree

Every commit (except the first one) has a parent (AKA the state of the repo when you started making changes).

If you and someone else start working in parallel from the same starting point, git will try to merge your changes back together later*



Remotes are copies of a repo on another computer

(or on a service like Github).

GitHub is a central place to host a repo

GitHub is a place to store your repo, so others can clone it, push to it, and pull from it.

Capital One Land™

GitHub Enterprise

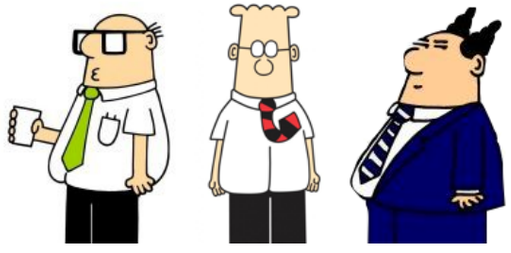
CapitalOne Github
(github.kdc.capitalone.com)

The internet

Public Github
(github.com)

The Proxy

you



The Great
CapitalOne Firewall

plebians/hackers

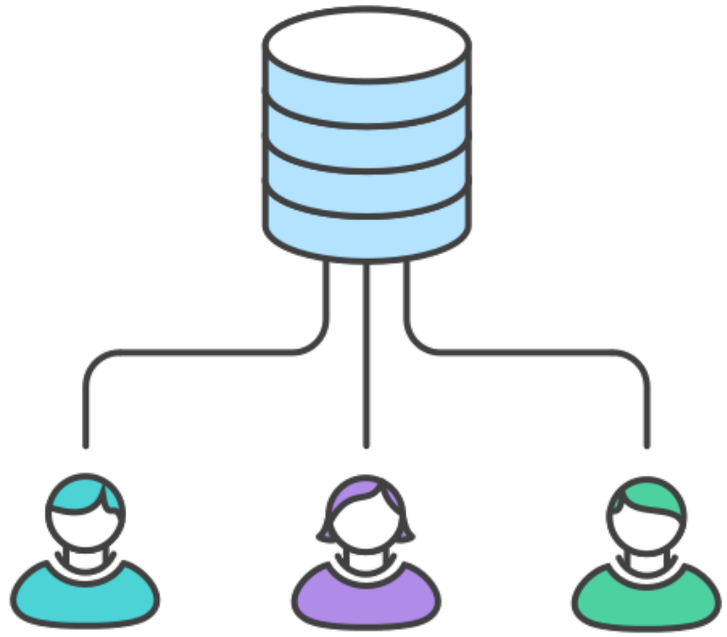
Collaborating using Git

Centralized workflow

Typically used on small teams

Pros:

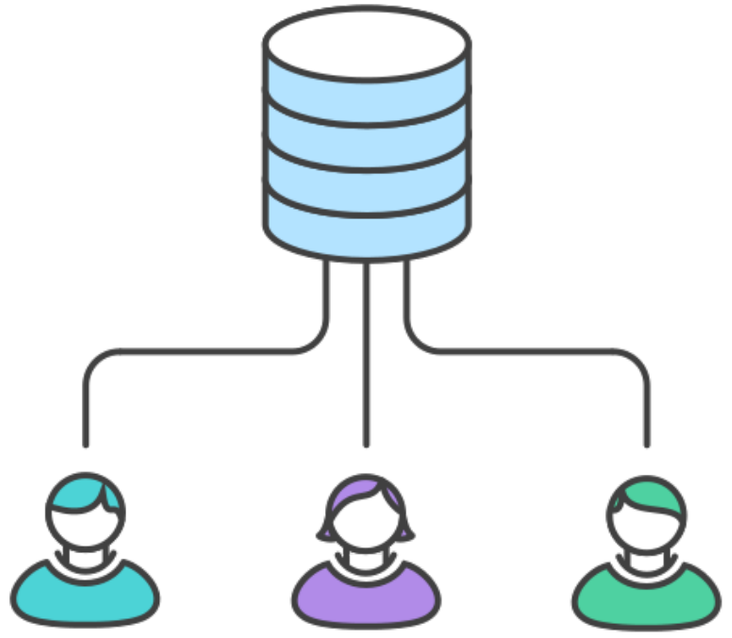
- everyone can do everything, no bottlenecks
- efforts can be organized + coordinated using git branches
- Github not technically required



Centralized workflow

Cons:

- people can make decisions/ruin code arbitrarily

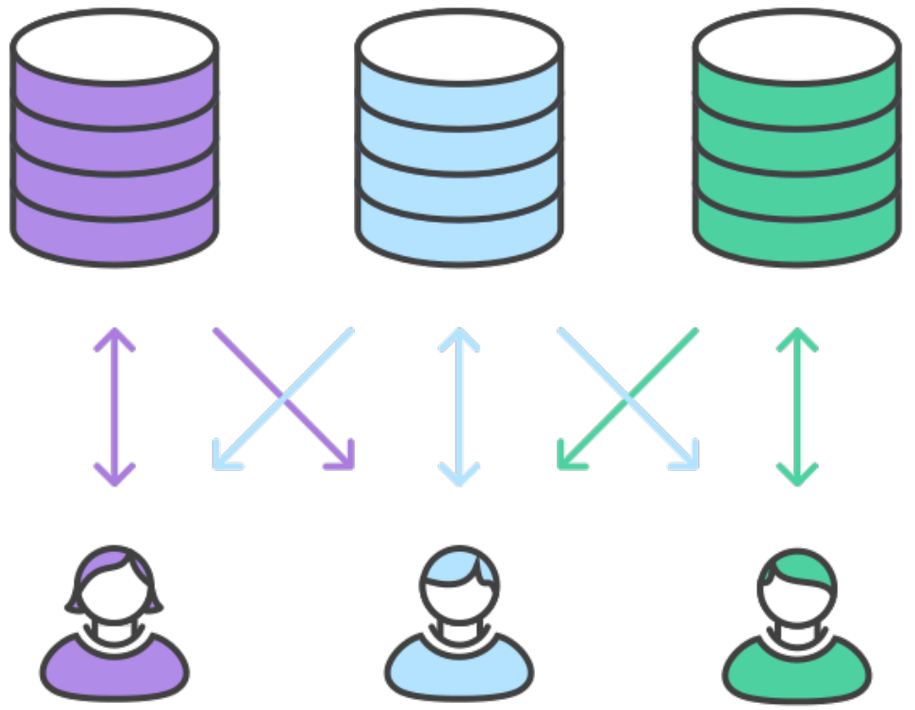


Fork + Pull request workflow

Typically used on larger open-source projects

Pros:

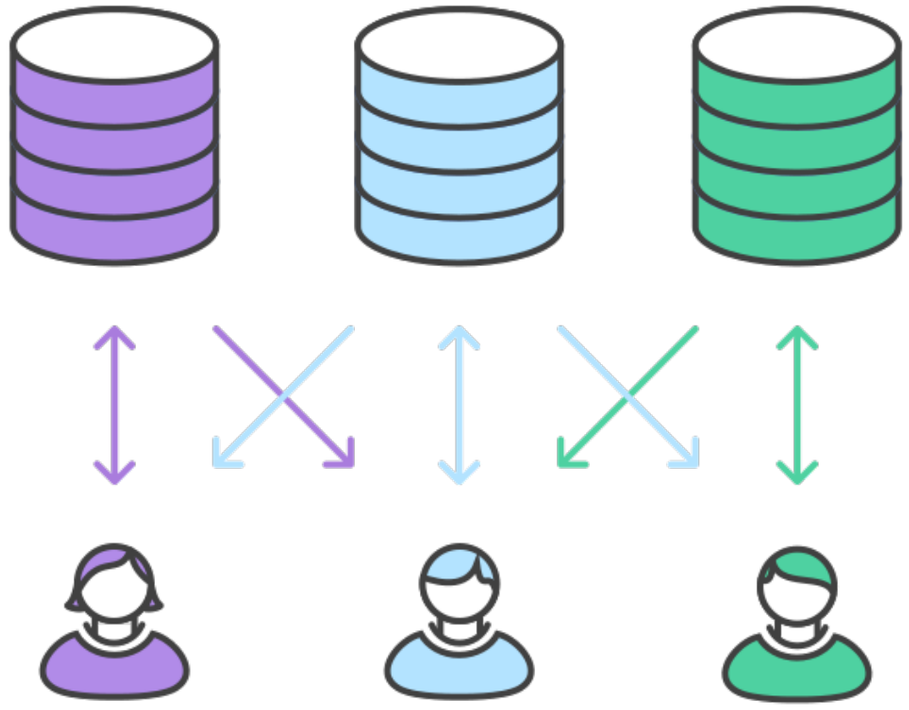
- Pull requests serve as places to review code for quality and discuss architecture
- Forks give people freedom to change the code in a way that suits them

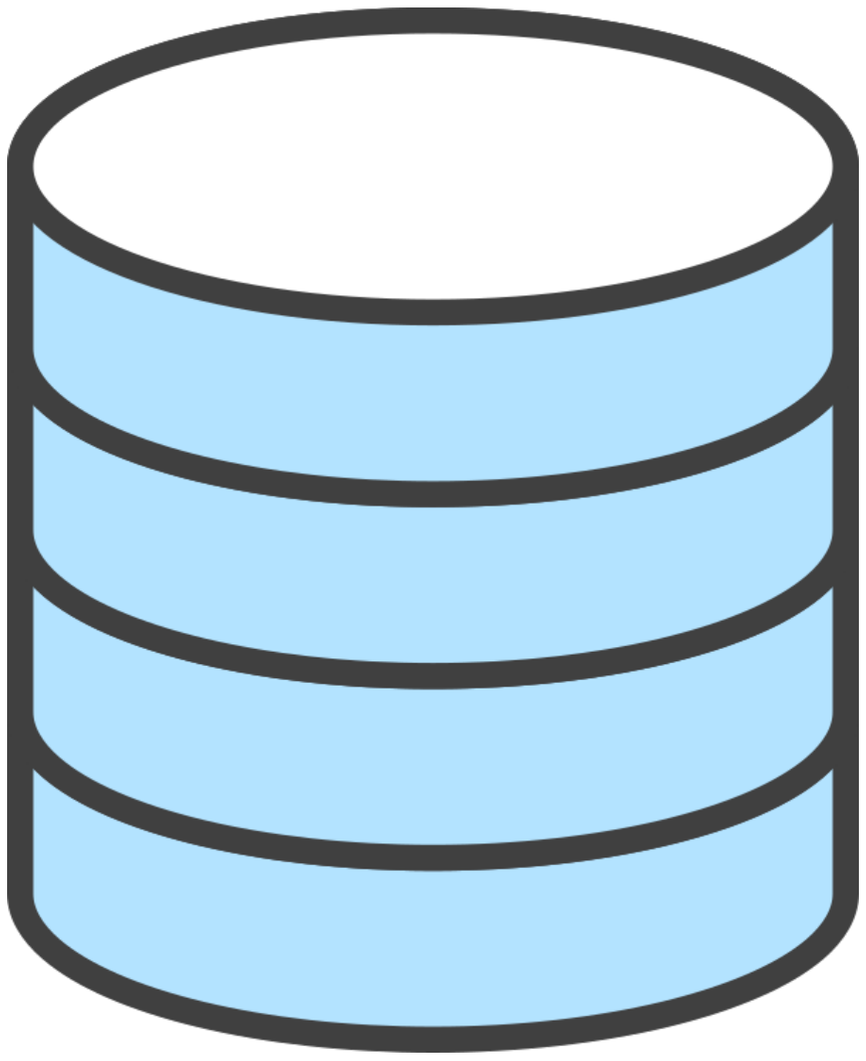


Fork + Pull request workflow

Cons:

- Changes must be approved by one of the *anointed ones*
- Harder to do without something like Github



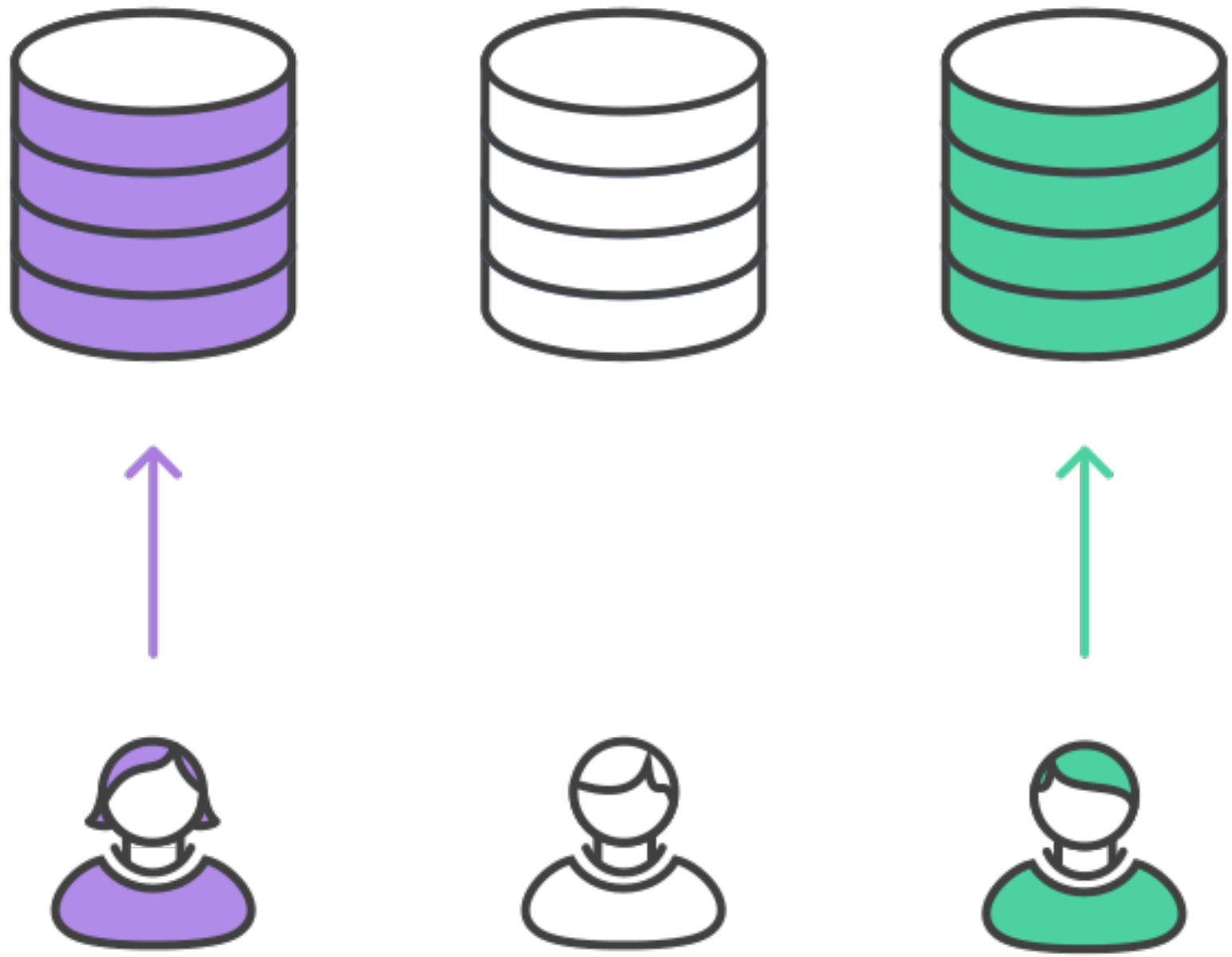


In the beginning, there is a repo.

Only a special few people have privileges to push to it.

People make their own copies of the repo

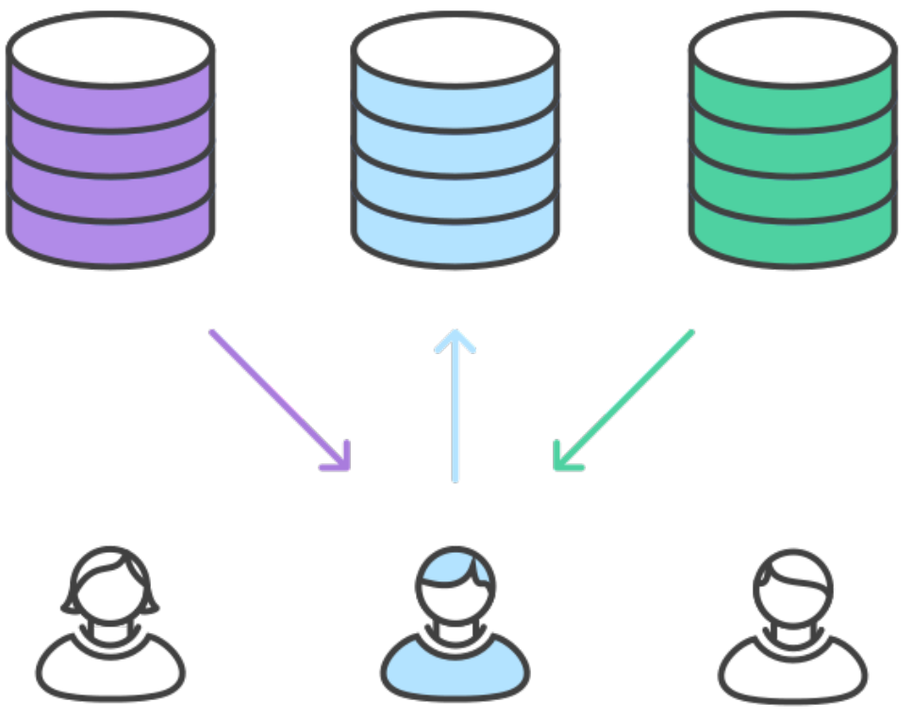
(Github makes this easy with the Fork button)



People make changes and can push to their own copies

The change makers notify the owner of the main repo of their changes. (Github makes this easy with the Pull Request feature).

The main repo owner can choose to accept their changes and merge them in, or provide feedback.



When updates happen to the main repo, people working on forks can pull them to their copies to stay up to date.

Let’s jump in!