Git and GitHub

Lead Front End Developer

Version control

What is it?

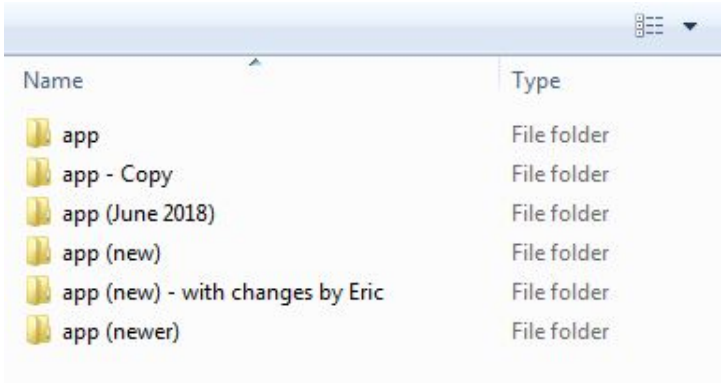
A ***system*** to **record changes** to files **over time** so that you can **recall specific versions later.**

Informal ‘systems’ people use for version control

It’s not long after being a computer user that people start creating their own ‘systems’ for local version control. Common approaches include:

- `Ctrl+Z`
- Creating copies to act as a ‘backup’

This are very common because they are so simple, but are **crude**, **limited** and **incredibly error prone**.



Before looking at formal version control, let's introduce the 'commit' (aka 'patch set', 'change set' etc.)

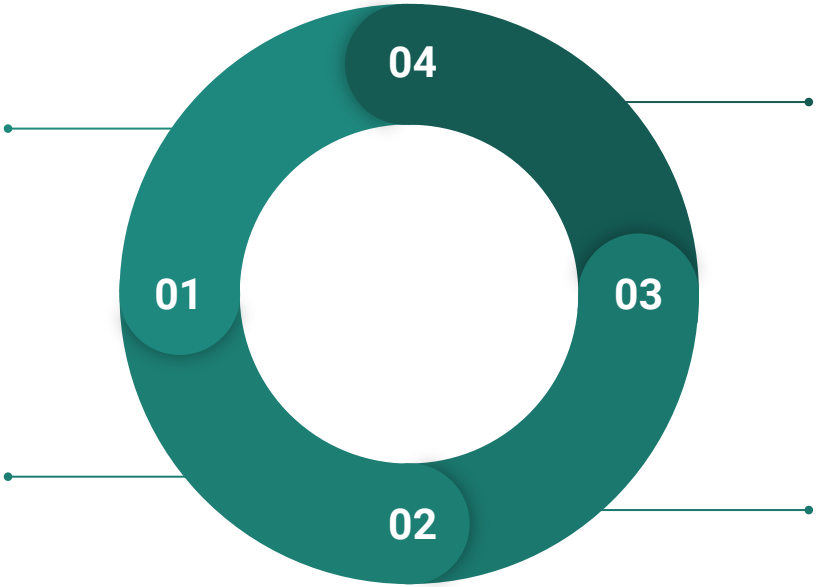
A representation of the commit cycle

Create or change code files to fulfil a specific task

In order to fulfil a task, the developer creates or amends files

Select files to be included in the commit

This will often be all the files that have changed, but it may be that the developer feels there would be benefit if splitting the commit over two or more commits



Commit the changes

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

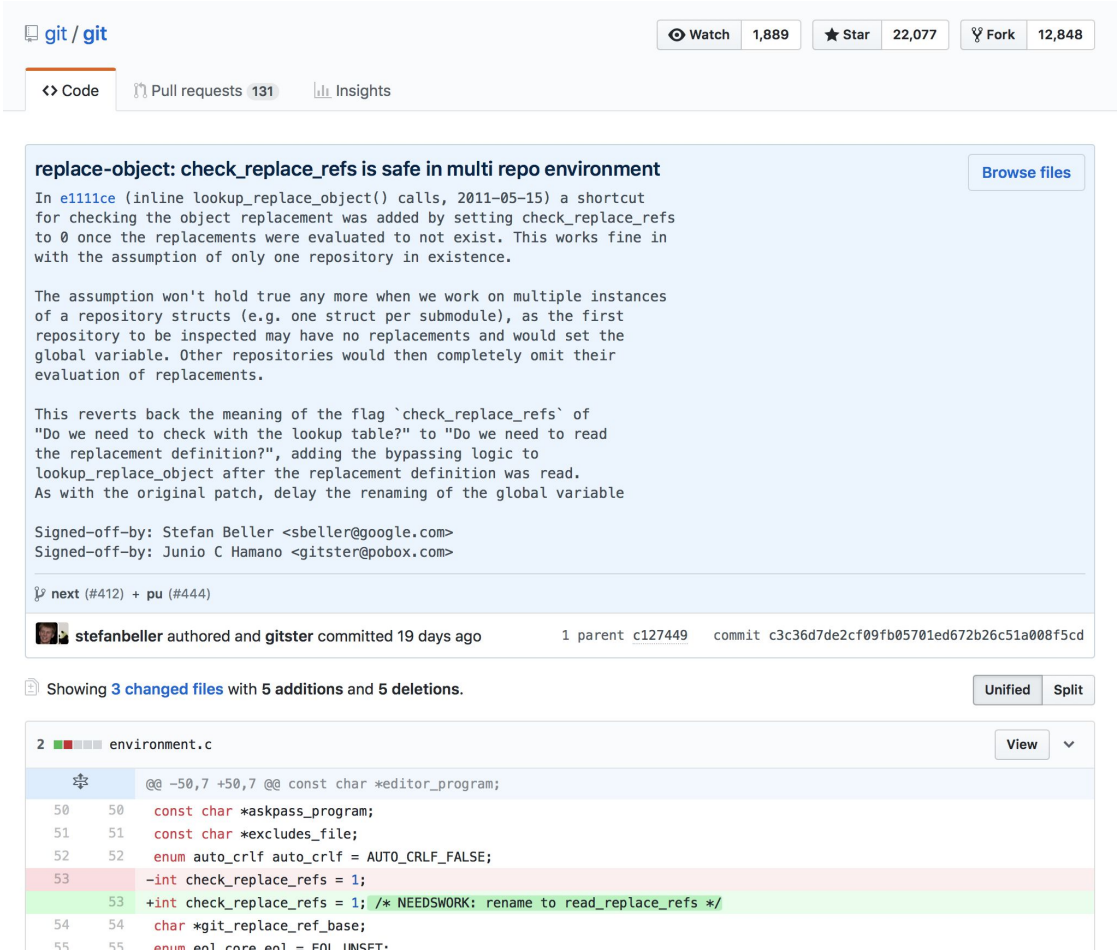
Describe the changes

Provide a message that communicates the **context** of a change to future developers

Commit messages communicate the context of a change

*"The contributors to these repositories know that a well-crafted Git commit message is **the best way to communicate context about a change to fellow developers** (and indeed to their future selves). A diff will tell you **what** changed, but **only the commit message can properly tell you why**."*

Chris Beams. "How to Write a Git Commit Message"



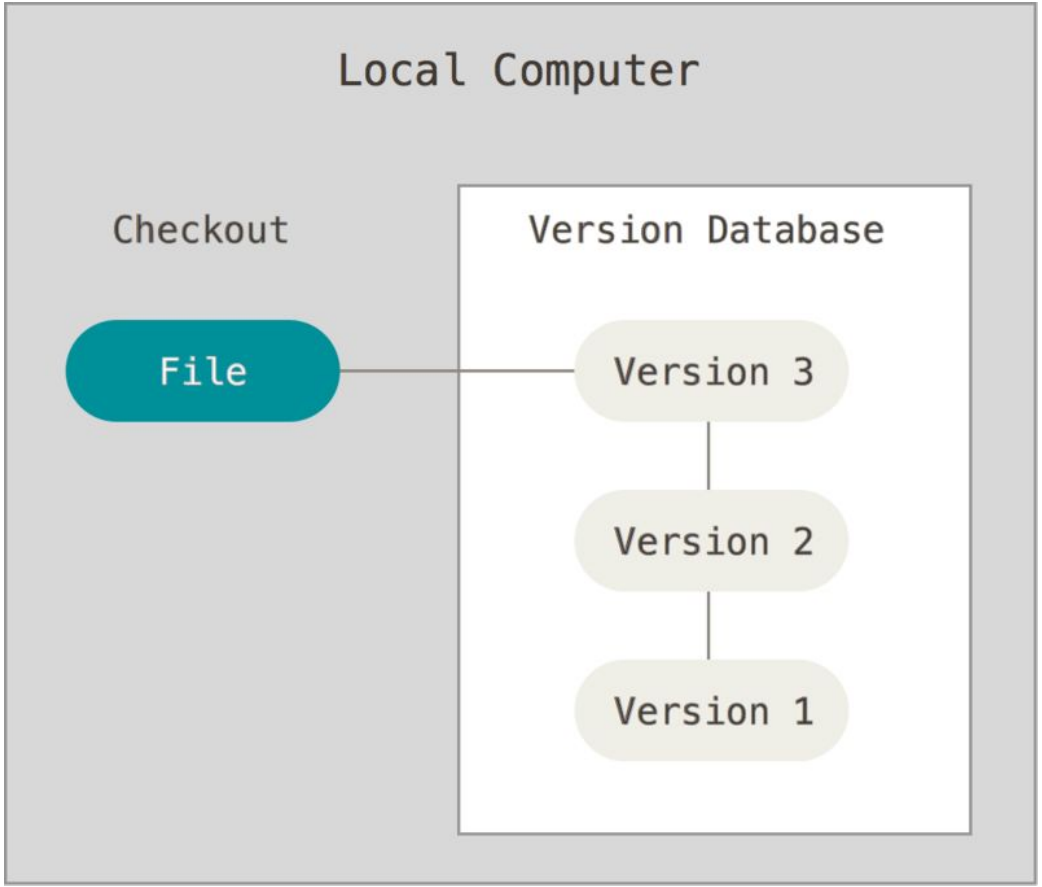
Evolution of formal Version Control Systems (VCSs)

Local Version Control

A long time ago programmers introduced local version control systems which stored ‘patch sets’ (i.e. the difference between files at different points in time).

Primary benefit:

- This allowed a programmer to recreate a specific state of a file at any given point by applying or removing specific patches

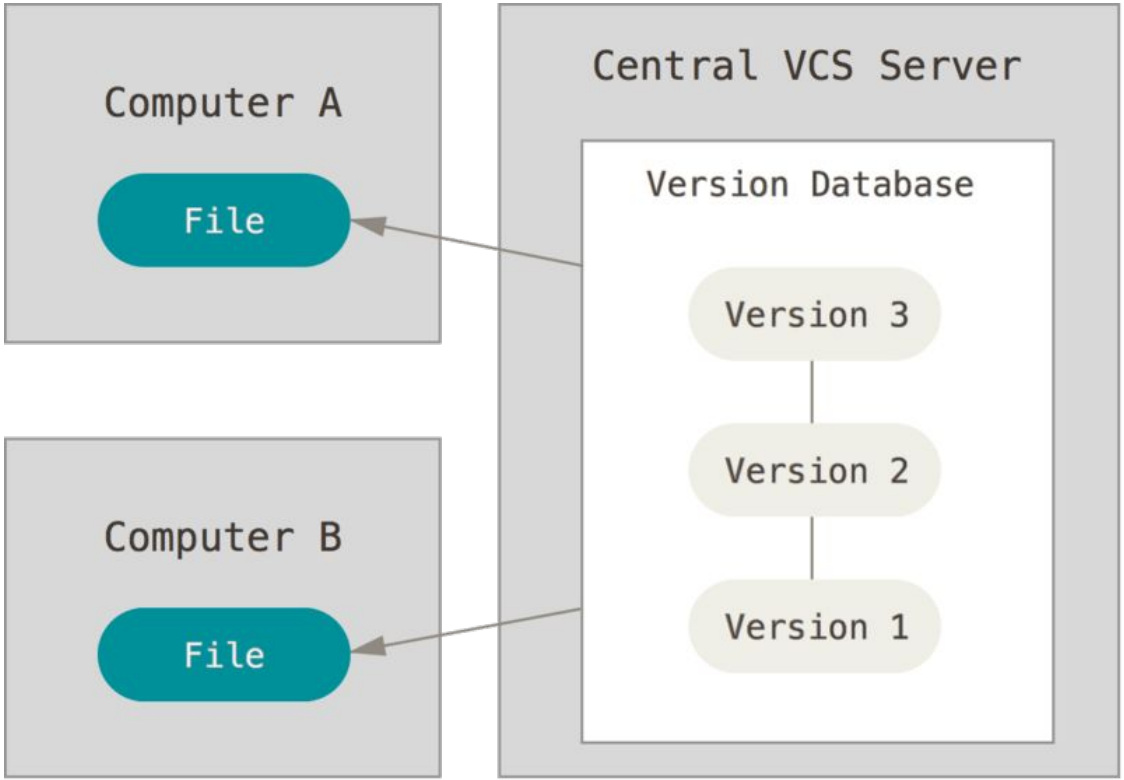


Centralized Version Control Systems (CVCSs)

A single server that contains all the versioned files and programmers ‘check out’ files from that single place

Primary benefit: Allowed programmers to collaborate with others

Drawbacks: huge SPOF

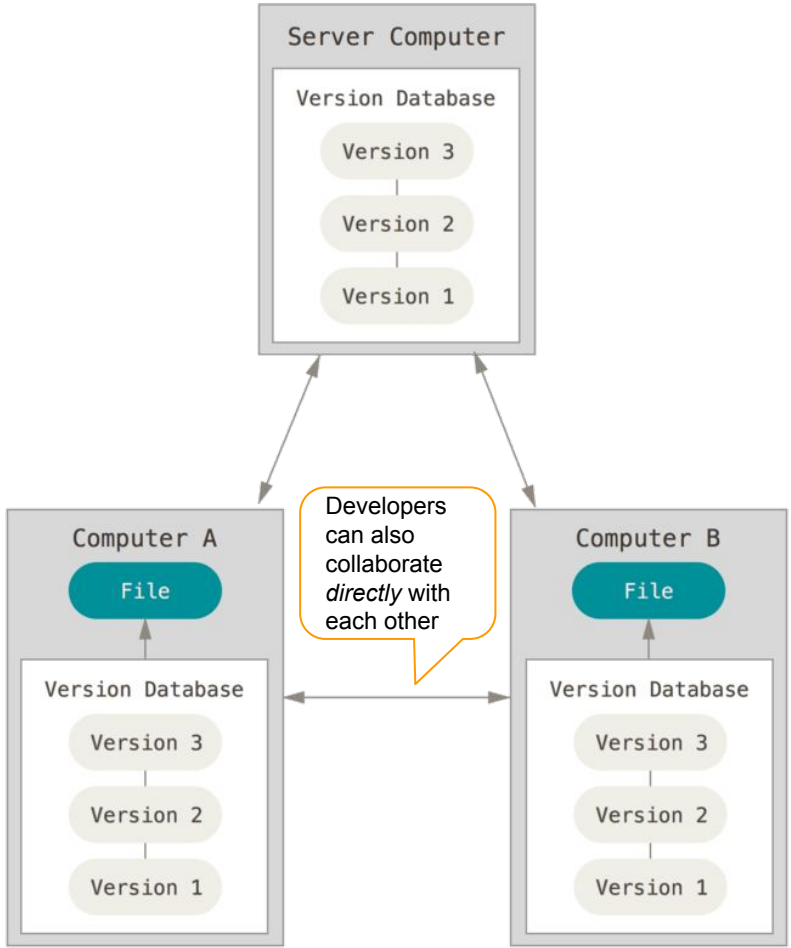


Distributed Version Control Systems (DVCSs)

In a DVCS programmers don't just check out the latest snapshot of the files; they fully mirror the repository - including its history.

Primary benefit: every clone is a full backup of the code and its history.

Git is a DVCS, bit it is not the only one



Each builds upon the capabilities of its predecessor

1

Distributed Version Control Systems

Fixes the SPOF that was introduced by CVCs

2

Centralised Version Control Systems

Allowed programmers to collaborate

3

Local Version Control Systems

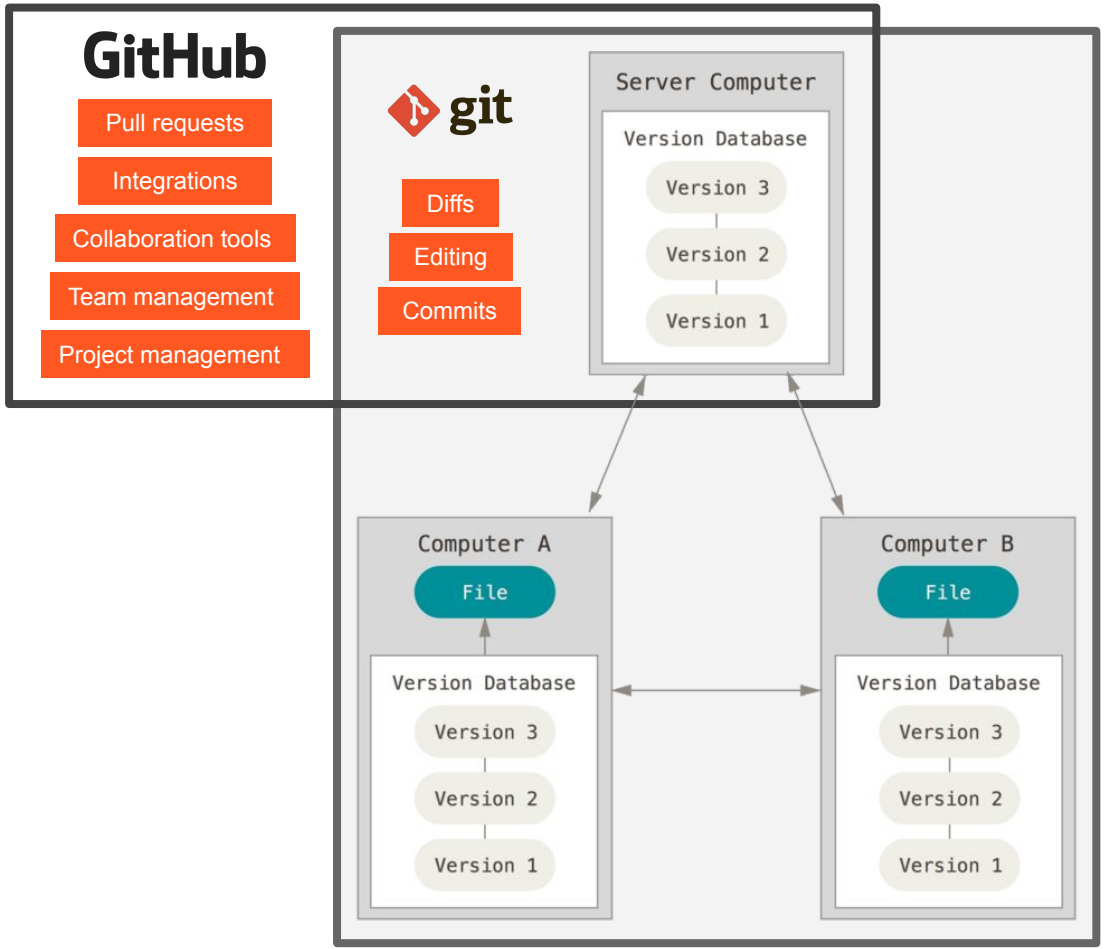
Allowed programmers to recreate the state of a file at any given point in time

Think of GitHub as a place for collaboration *with* Git

GitHub

A hosting service for version control using Git. In addition to being somewhere that developers can ‘pull’ and ‘push’ code, it provides several other features, including:

- Integrations
- Code review (Pull requests)
- Team management
- Project management



So, what is a Pull Request?

Here's a good explanation