Version Control Systems

Progress, far from consisting in change, depends on retentiveness. Those who cannot remember the past are condemned to repeat it.
- George Santayana, Life of Reason

You want to cook a new Dish!

Recipe 1

```
recipe.add('secret sauce')
recipe.add('eggs')
```

You want to Improvise Your Dish!

Recipe 1.1

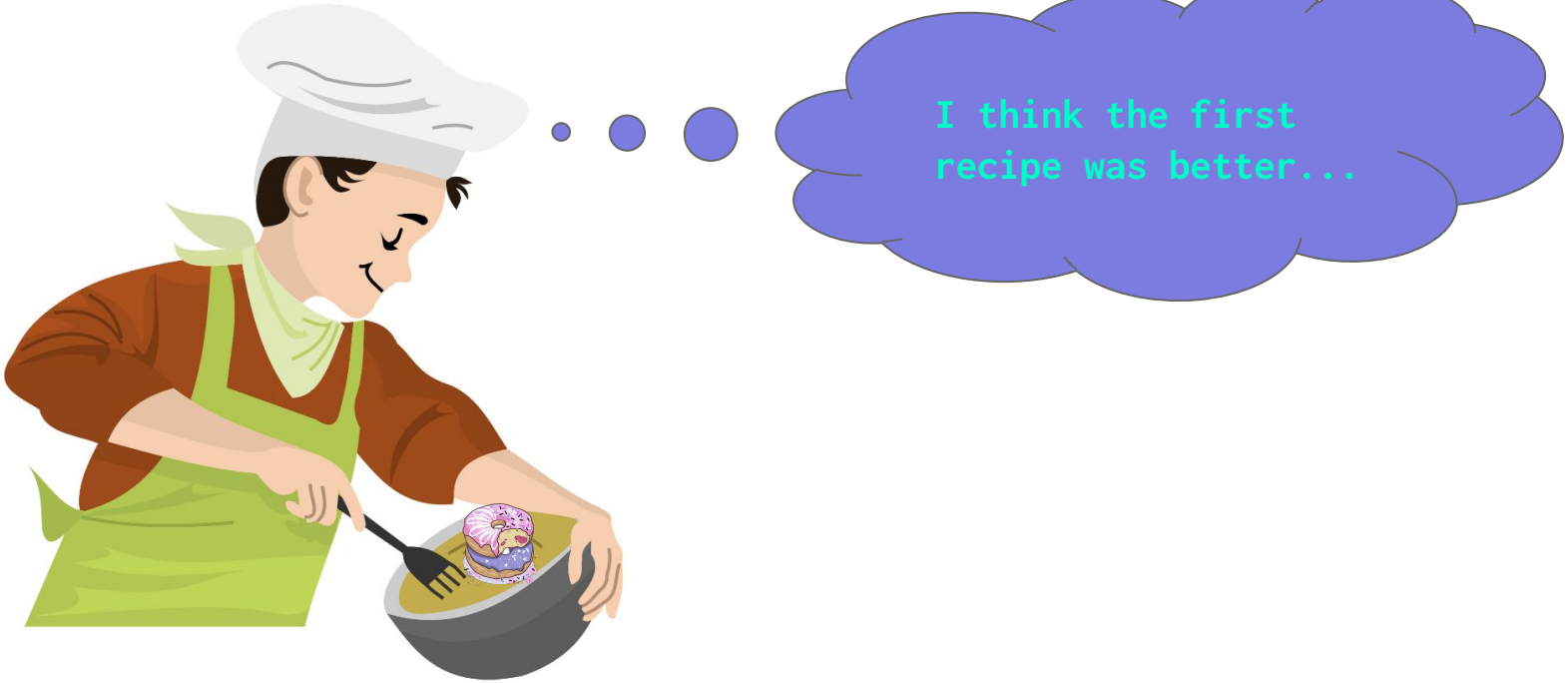
```
recipe.add('secret sauce 2')
```

You Are Still not satisfied :/

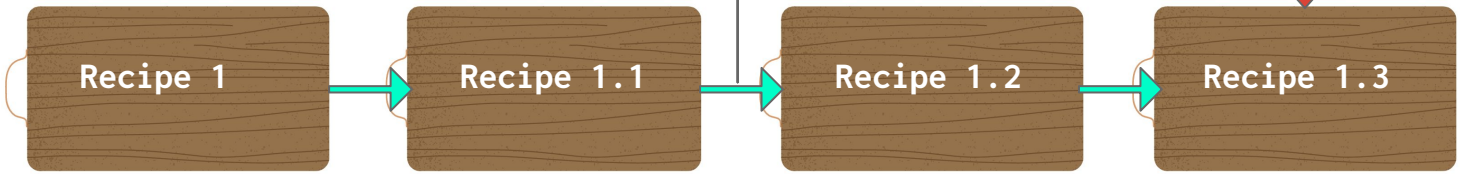
Recipe 1.2

```
recipe.add('donuts')
```

Everything is messed up

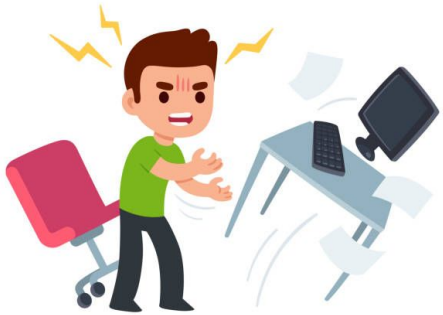


This is where Git (our Savior) comes into Play



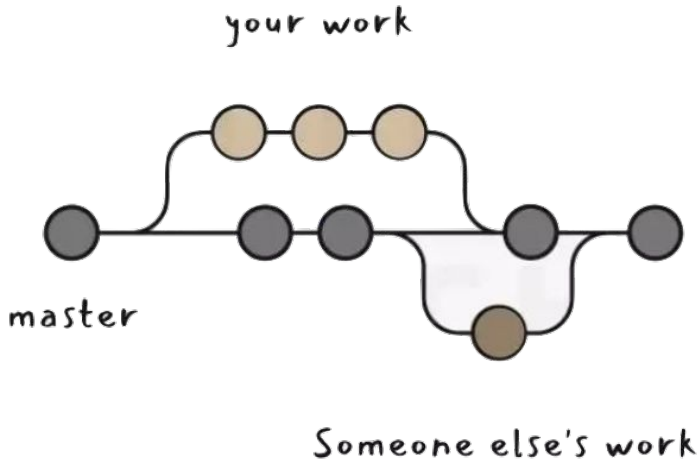
Git is a distributed version control system, that tracks the changes you made, so that you have a record of what has been done and you can revert to **specific versions.** Collaboration becomes easier and multiple people can contribute to the same project.

Before Git - a programmer's worst nightmare



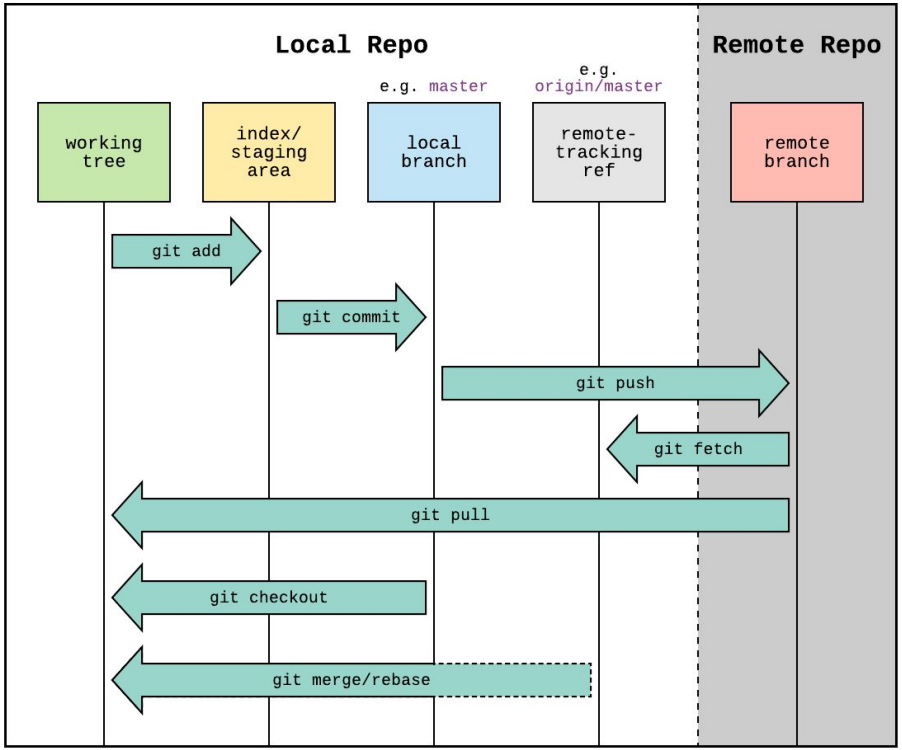
- Before version control systems were created, people relied on manually backing up previous versions of projects. They were copying modified files by hand in order to incorporate the work of multiple developers on the same project.
- It cost a lot of time, hard drive space, and money.
- Git has market share of 89.74% in version-control market.

Branching



The **Master Branch** is usually the first branch committed to a repository.A **Branch** is basically an independent version of the project code within a library. Git can automatically merge the changes so two people can even work on the same features in a project and later merge those changes without losing each other’s work.

Overview of Git Functionality



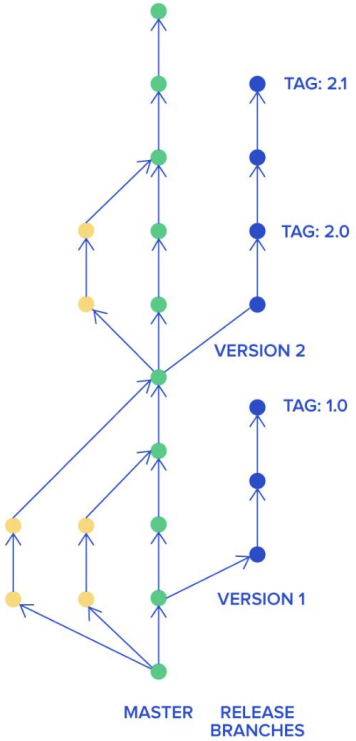
Git Cheat Sheet

https://git-school.github.io/visualizing-git/

Workplace Development Styles - Git Flow

- One main development branch with strict access to it. It's often called the **develop** branch.
- Developers create **feature** branches from the develop branch and then they create **pull requests.**
- When the main branch has reached enough maturity to be released, a separate branch (**version**) is created to prepare the final version.
- The application gets **tested** and **bug fixes** are applied, and the final product is merged to master with the release version (e.g. **TAG: 2.0**)
- Advantages: Strict control, Application optimization
- Disadvantages: MVP, iterate slowly, Micromanagement
- Use Cases: Open Source, Established products, lot of junior devs.

Workplace Development Styles - Trunk Based Development



- Developers work on the **master** branch with open access to it. **Feature branches** are created and merged back to master once it is compiled and passes all tests.
- Works well with a team of seasoned developers.
- Development is continuous and prevents developers from creating merge conflicts that are difficult to resolve.
- Advantages: Development speed, MVP
- Disadvantages: no strict control, large teams
- Use Cases: startups, small teams, senior devs

Resources

- https://learngitbranching.js.org/ - Guided exercises with visualization
- https://github.com/eficode-academy/git-katas - Clone repo and do exercises (no visualization)
- https://education.github.com/git-cheat-sheet-education.pdf - git cheat sheet
- https://git-school.github.io/visualizing-git/ - visualizing git
- https://www.growingwiththeweb.com/2014/02/a-gentle-introduction-to-git.html - great introduction doc