

Versioning with Git



Teacher

Sergei Kudinov

Developer and Big Data engineer at Adaltas

sergei@adaltas.com

GitHub - github.com/sergkudinov

We will learn Git basics

Plan:

- What is versioning
- Git introduction
- Lab: practicing Git

What is Versioning?

Also known as:

- Version Control or Version Control System (VCS)
- Source Control Management (SCM)

What is Versioning?

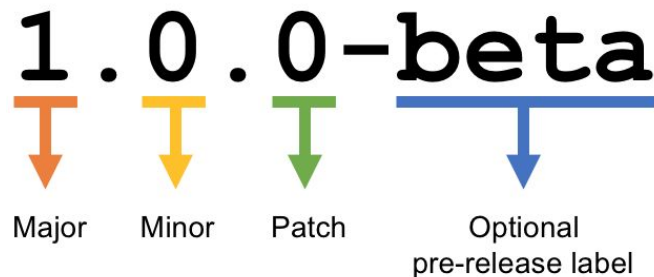
It is the practice of tracking and managing changes to software code.

SCM (or VCS) - tools that help you keep track of your code with a complete history of changes.

What do they manage?

- Code (or any **text-based** documents)
- Project versions:
 - Global project version (**tags**, like ``1.2.4-beta``)
 - Each modification is a «version»
- Change requests

Semantic Versioning (tag names)



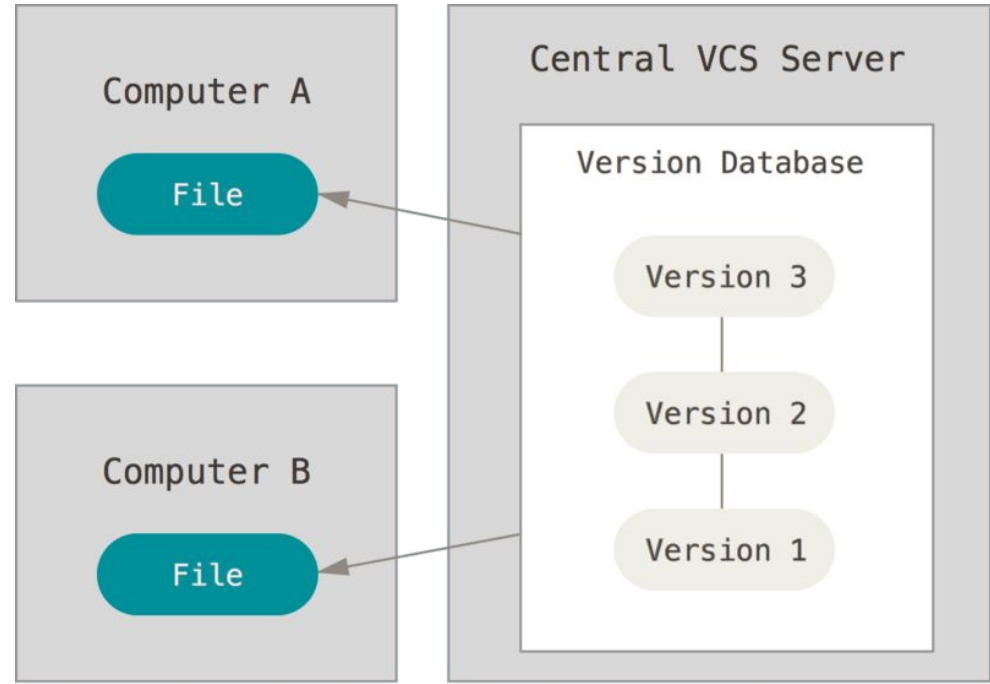
- **MAJOR** - when incompatible API changes
- **MINOR** - when new functionality
- **PATCH** - when bug fixes
- **LABEL** - optional, for pre-releases

What types of VCS?

- Centralized
- Decentralized

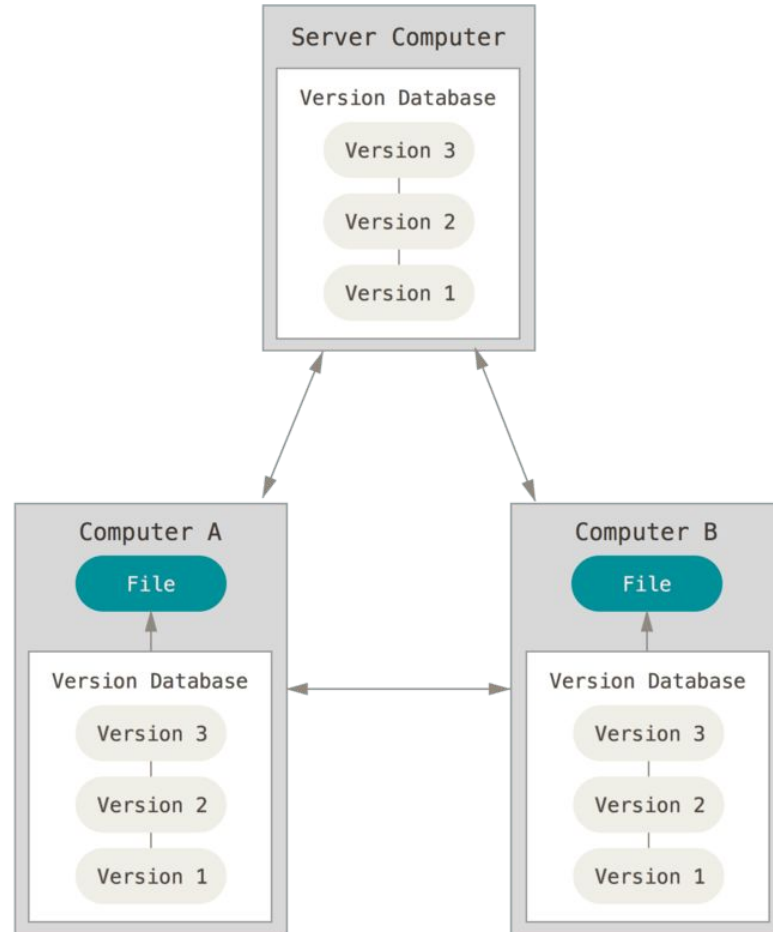
Centralized VCS

- Needs network
- Single Point of Failure



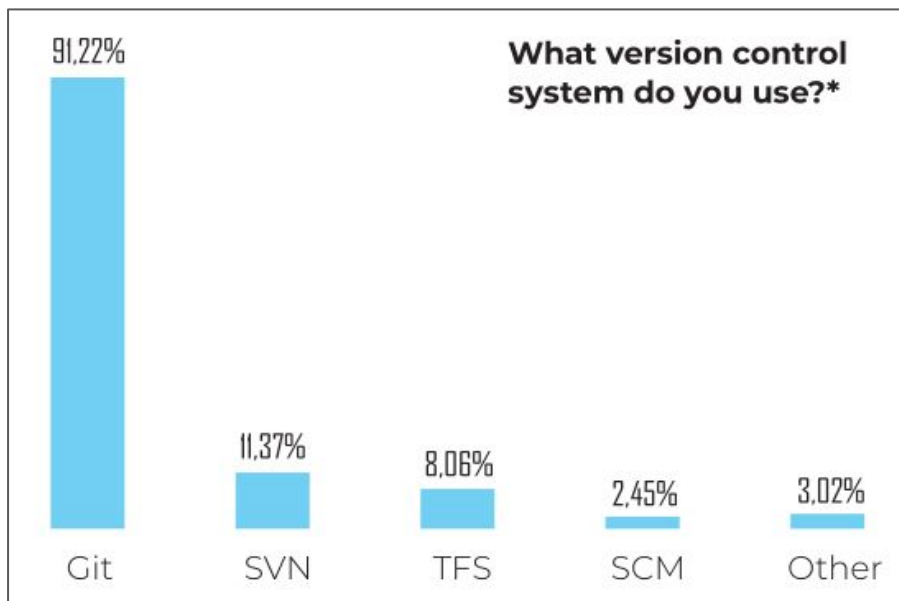
Decentralized VCS

- + Local work is possible
- Workflow is more complex



Why learning Git?

- Alternative to **SVN** (Subversion, created in 2000)



From [State of software development in 2019](https://codingsans.com/state-of-software-development-in-2019/),
codingsans.com

What is Git?

- **The most popular VCS**
- **Decentralized**
- **Open Source and free**
- Created in **2005** by Linus Torvalds ----->



Git repository

- **Is a set of versioned files**

With entire history of changes

- **Copied to a local folder**

“.git” folder at the root of a project

- **The users choose the files to version**

Common scenario

1. Checking out remote changes

```
git clone or git pull
```

2. Editing a file

eg: add a new function “attack()” to the “player.js” file

3. Adding a file to “staging area” for the next commit

```
git add player.js
```

Common scenario

4. Commit the modification to the local repository:

```
git commit -m "Add player attack"
```

5. Send local modifications to the **remote** repository

```
git push
```

6. Resolution of any conflicts:

Code modification + commit

Conflict resolution

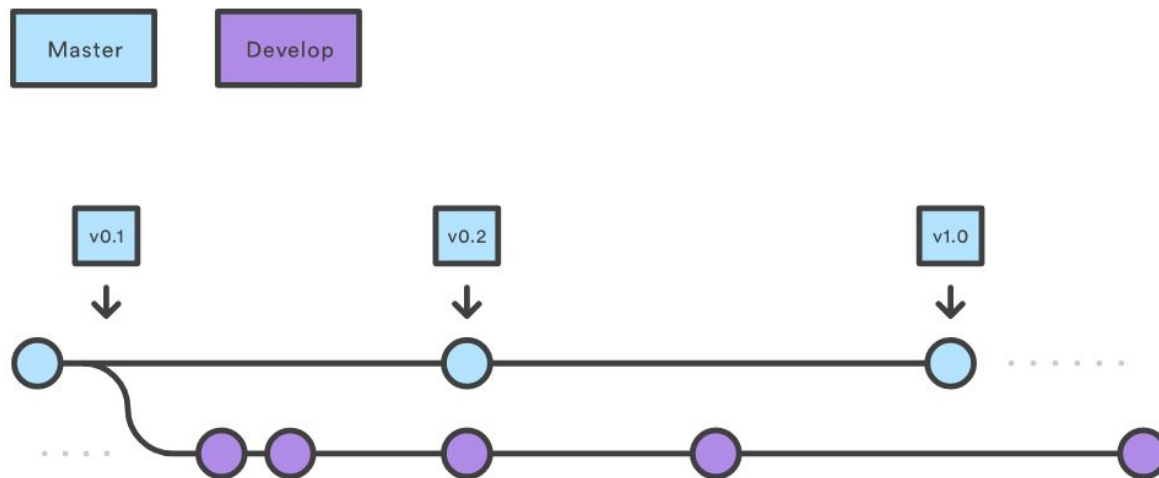
Conflicts - are modifications made on the same line

Must be resolved manually:

- edit files with a resolver tool (or text editor)
- commit the files
- push to the **remote branch**

Branches

- 1 branch = 1 separate versioning space → **isolation of changes**



Managing a git repository

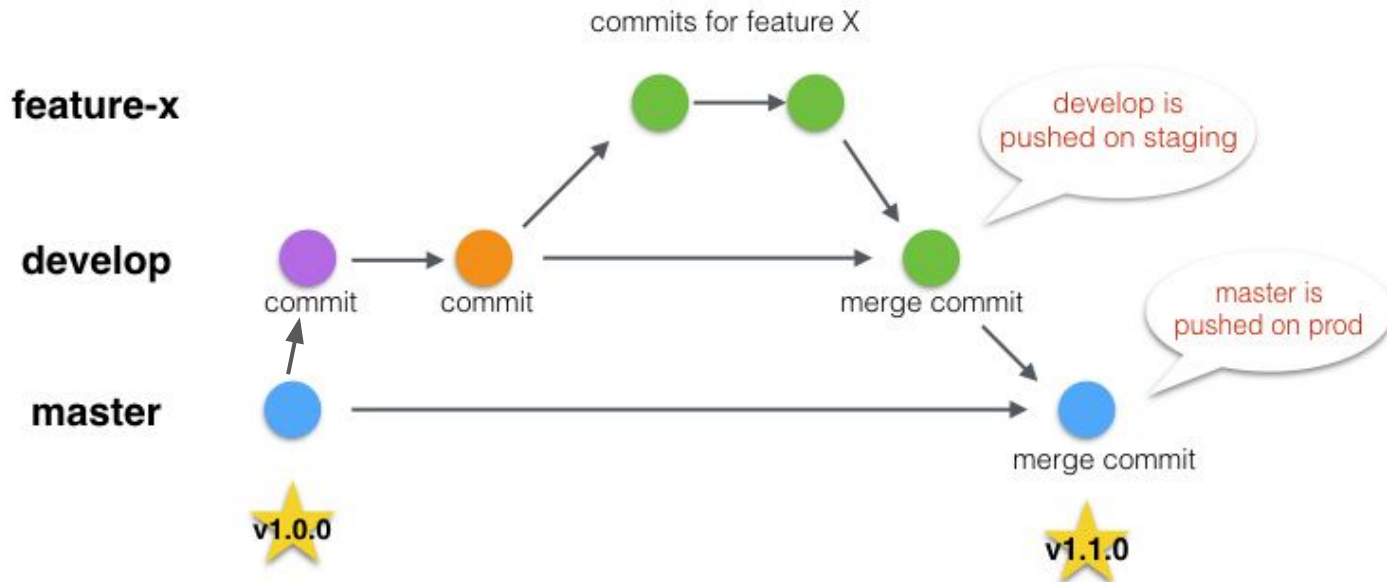
- Git commands:
 - `pull`, `push`, `merge` ...
 - through a graphical interface: GitHub Desktop, Fork, GitKraken...
- Git remote repositories:
 - on GitHub, GitLab, self-hosted...

Git ignore

- The “.gitignore” file contains the files to ignore
- Works with wildcards
 - `.*` : ignore all files starting with a “.”
- Useful for ignoring files specific to your environment:
 - codeblocks or VScode files
 - “node_modules” folder in Node.js

Git Flow

Typical organization of versioning:



Where can you use Git?

- Software development
- Writing (books, articles, theses)
- Whatever requires tracking the history...

Go further

1. Learn the difference between **merge** and **rebase**:
<https://dzone.com/articles/merging-vs-rebasing>
2. Learn **Conventional Commits** - a specification for writing commit messages:
<https://www.conventionalcommits.org/en/v1.0.0-beta.2/>