Version Control

Dario Campagna

Agenda

Version control

Introduction to Git

Using Git and GitHub

Other version control systems

Version control is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information.

Why version control?

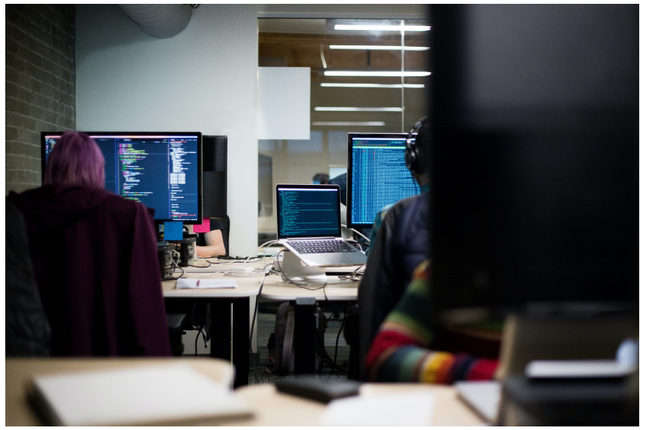
It is common for multiple versions of the same software to be deployed in different sites.

- Bugs or features are often only present in certain versions.
- Need to retrieve and run different versions of the software.

Why version control?

It is common for developers to be working simultaneously on updates.

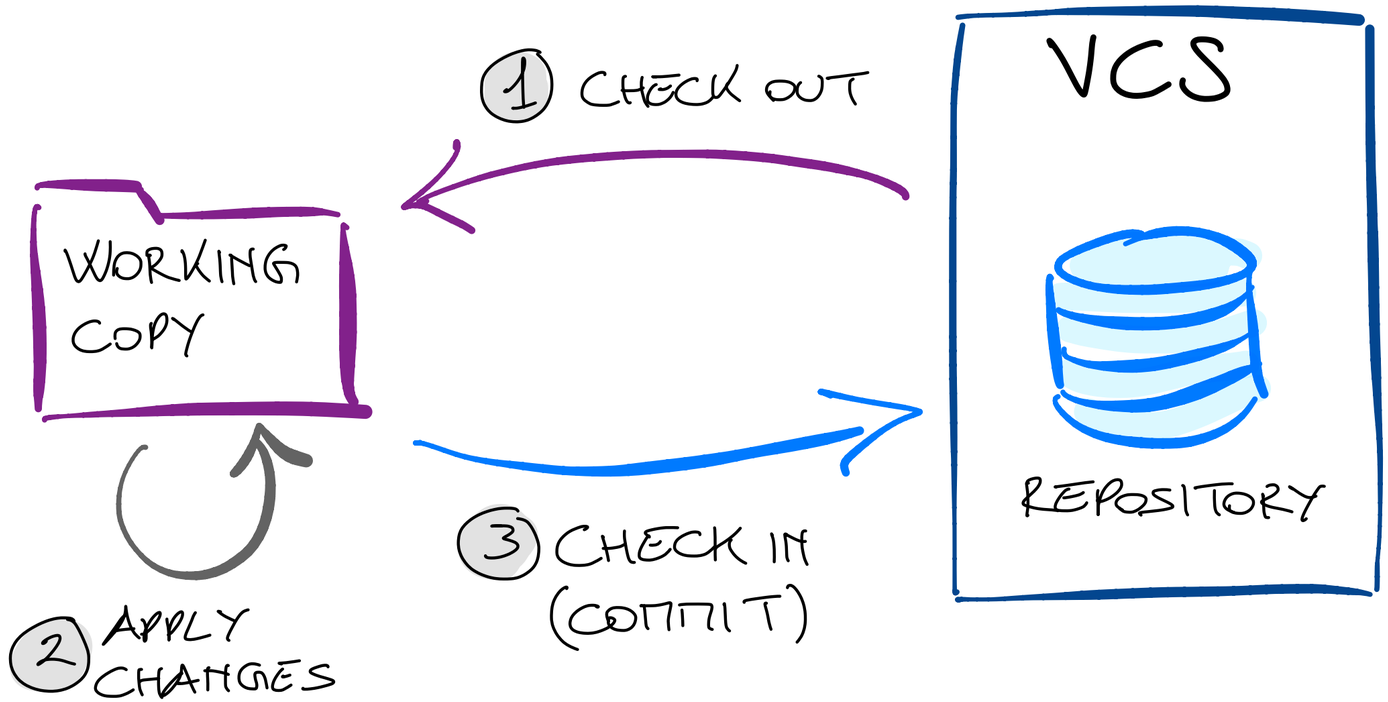
- We need to integrate the changes into the software.
- Integrating the work of different developers is a crucial task.



Maintaining multiple versions of the software and integrating the work of different developers in a **manual way** is **inefficient** and **error prone**.

What's a Version Control System?

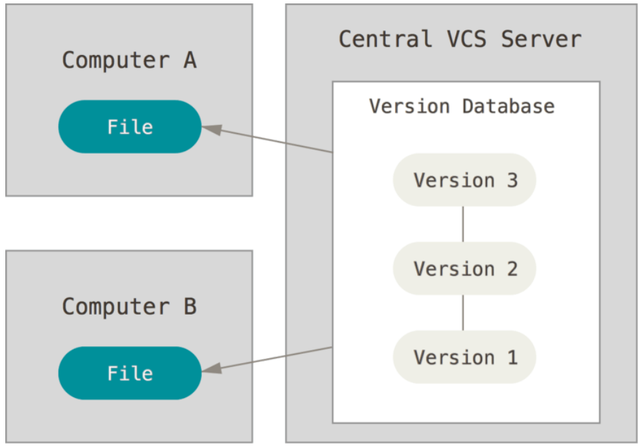
A Version Control System (VCS) is a software that manages changes to a set of data over time.



Centralized VCS

A single server contains all the versioned files, a number of clients check out files from that central place.

- Everyone knows what everyone else on the project is doing.
- Administrators have fine-grained control over who can do what.
- The centralized server is a single point of failure.

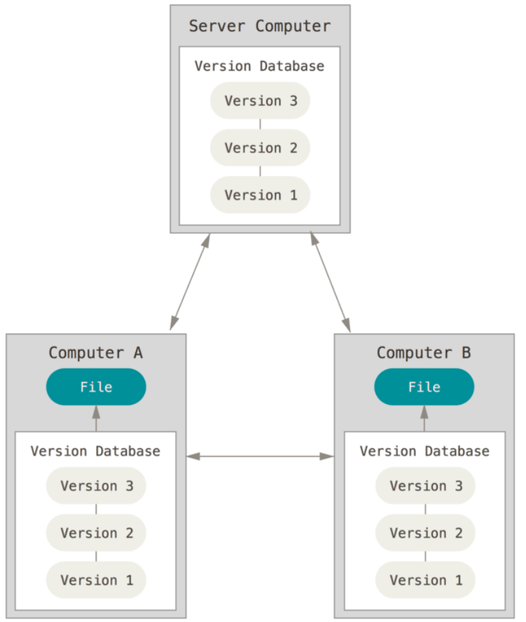


Picture from Pro Git

Distributed VCS

Clients don’t just check out the latest snapshot of the files.
They fully mirror the repository, including its full history.

- No canonical, reference copy of the codebase exists by default.
- Common operations are fast.
- Communication only when pushing/pulling to/from other peers.
- Working copies function as remote backups.



Picture from Pro Git

Git

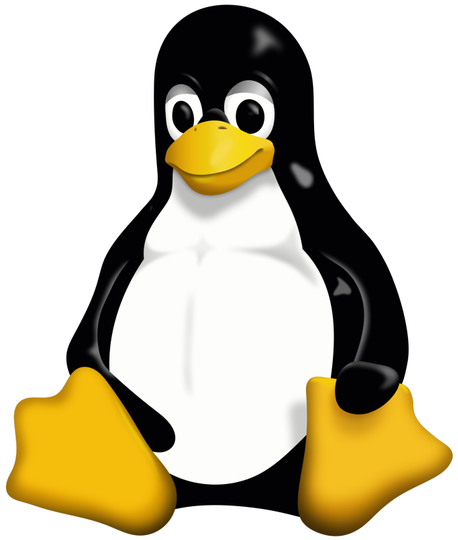
Free and open source distributed VCS designed to handle everything from small to very large projects with speed and efficiency.

- Built to work on the Linux kernel.
- Speed and performance has been a primary design goal.

A bit of history of Git

Git development began in 2005 in response to changes around the Linux kernel.

- 1991-2002: patches and archived files.
- 2002: BitKeeper, a proprietary distributed VCS.
- 2005: relationship with BitKeeper company broke down, development of Git starts.



Goals of Git

The Linux development community developed Git based on some of the lessons learned while using BitKeeper.

- Speed.
- Simple design.
- Strong support for non-linear development.
- Fully distributed.
- Efficient handling of large projects.

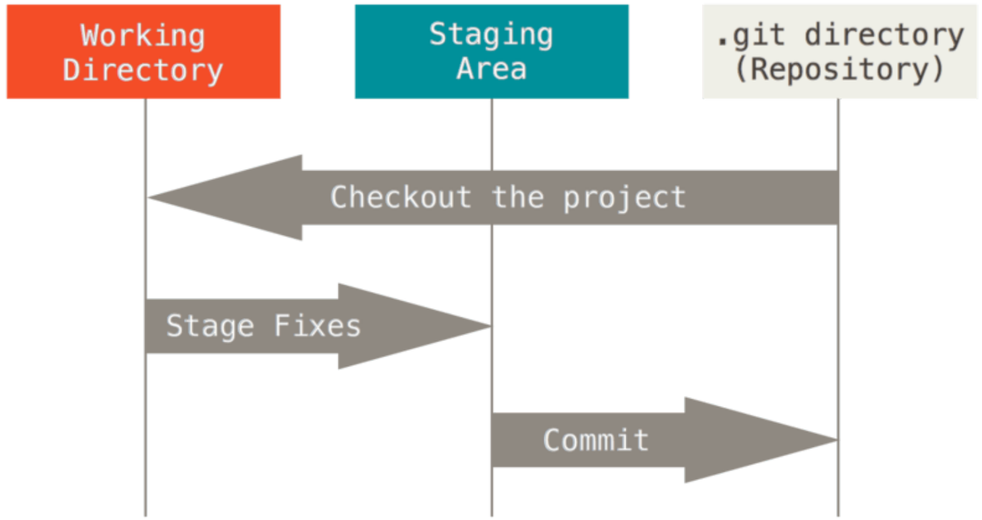
Let’s try Git

Initialize a Git repository for our hello-name application.

- Use Git from the command line.
- Look at some of the basic Git commands.

Main sections of a Git project

The working directory, the staging area and the Git directory.



Basic Git workflow

1. You modify files in your working directory.
2. You stage the changes you want to be part of your next commit.
3. You do a commit.

The three states

Git has three main states that your files can reside in.

Modified

You have changed the file but have not committed it.

Staged

You have marked a modified file in its current version to go into your next commit snapshot.

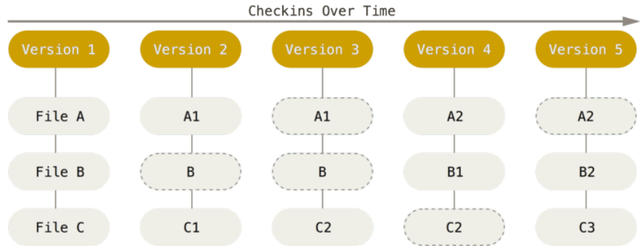
Committed

The data is safely stored in your local database.

Snapshots

Git thinks of its data like a series of snapshots of a miniature filesystem.

- When you commit, Git takes a picture of what all your files look like at that moment and stores a reference to that snapshot.
- If files have not changed, Git doesn't store the file again.



Picture from Pro Git

Nearly every operation is local

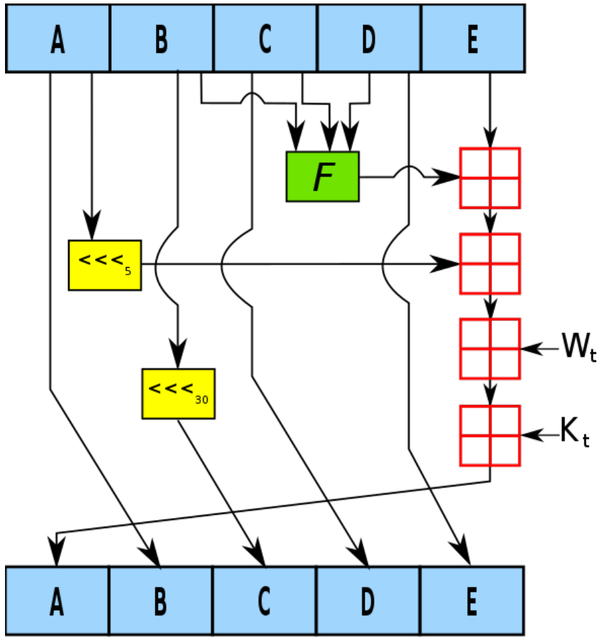
Most operations in Git need only local files and resources to operate.

- Most operations seem almost instantaneous.
- There is very little you can’t do if you’re offline or off VPN.

Integrity

Everything in Git is checksummed before it is stored and is then referred to by that checksum.

- Git uses a SHA-1 hash for the checksumming.
- Git stores everything in its database by hash value.



Picture from Wikipedia

Generally only adds data

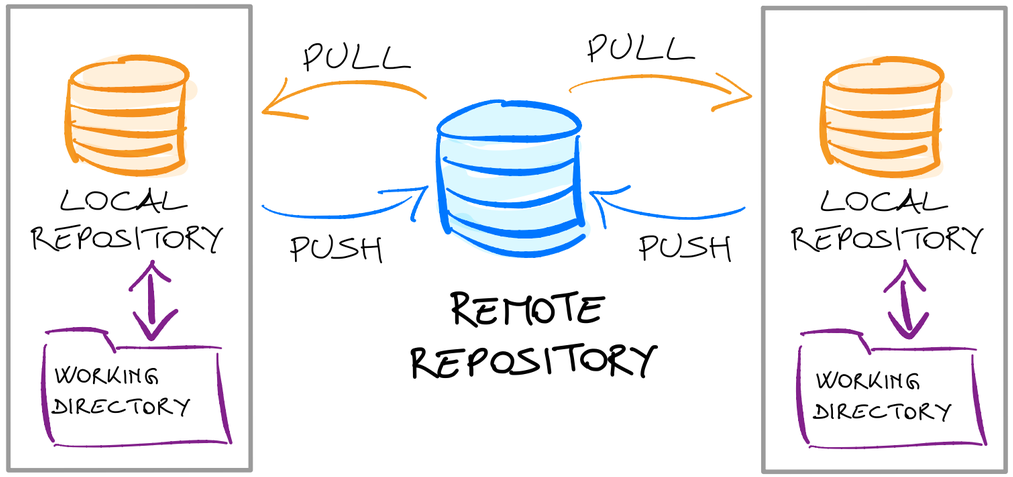
When you do actions in Git, nearly all of them only add data to the Git database.

- After you commit a snapshot into Git, it is very difficult to lose.
- You can experiment without the danger of screwing things up.

Remote Git repositories

In order to do any collaboration in Git, you'll need to have a remote Git repository.

- Remote repositories are versions of your project that are hosted on the Internet or network somewhere.
- The preferred method for collaborating with someone is to set up an intermediate repository.



GitHub

GitHub is the single largest host for Git repositories.

- A large percentage of all Git repositories are hosted on GitHub.
- Many open-source projects use it for Git hosting, issue tracking, code review, and other things.

Let’s try GitHub

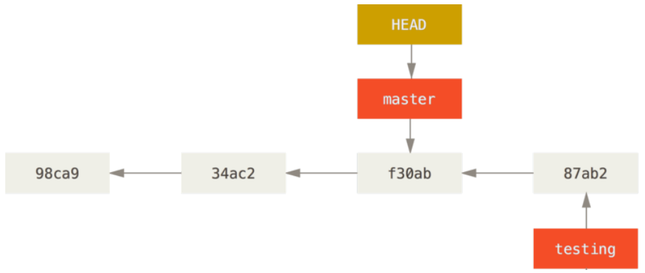
Create a remote Git repository for our hello-name application.

- Set up a remote repository on GitHub.
- Look at Git commands to work with remote repositories.

Branches

With branches you can diverge from the main line of development and continue to do work without messing with that main line.

- A branch is simply a lightweight movable pointer to one commit
- Branching operations are nearly instantaneous
- Git encourages workflows that branch and merge often
- Have a look at https://learngitbranching.js.org



Picture from Pro Git

Alternative tools

IDE

IntelliJ IDEA

NetBeans

Visual Studio Code

…

Git GUI

SourceTree

GitHub Desktop

TortoiseGit

…

Hosting sites

GitLab

Bitbucket

Launchpad

…

Other version control systems

Git is only one of the available free and open source VCS.

- CVS
- Subversion
- Bazaar
- Mercurial
- …

References

Pro Git

https://git-scm.com/book/en/v2

Git reference manual

https://git-scm.com/docs

Go version!

esteco.com