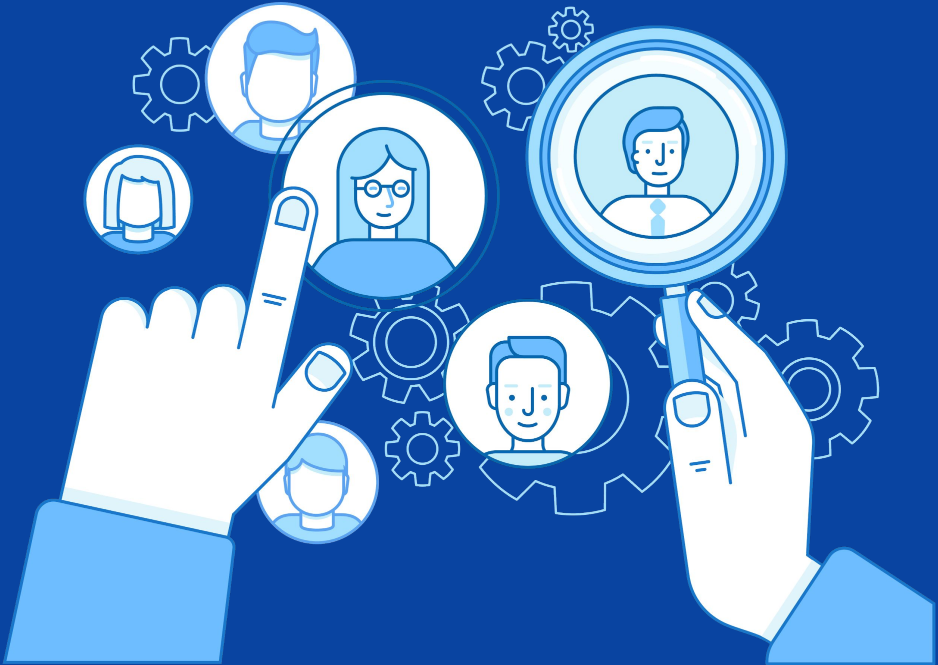
Digital Career Institute

Versioning and Collaboration



Digital Career Institute

Goal of the Module

By the end of this sub module, you will

- Understand common use cases for version control software
- Create local repositories using the terminal
- Create remote repositories on GitHub
- Understand the basic git workflow
- Use commits and branches to create version histories for projects

Digital Career Institute

Topics

- Introduction to Version Control Systems
  - Including a brief history
- The git program
  - Distributed version control
- Git core concepts and commands
  - Local and remote repositories
  - Staging
  - Basic commands and workflow

Digital Career Institute
DCI

Introduction to Version Control Systems

Digital Career Institute

Version Control Systems

Very quickly when developing software you will notice the need for versioning your work. There are many reasons why this is the case.

Many things are very hard to achieve without a formal version control system in place, such as

- Multi-person teams working on one project
- Maintaining old versions and developing new ones
- Fixing problems in multiple versions
- Maintaining multiple versions before release
  - one version for testers
  - one version for developers
  - one for marketing

A system

These are some of the issues solved by a VCS (Version Control System) - also known as

- revision control
- source control
- source code management

These systems are often mainly used for source code, but they can also be used for almost any versioning

- Documents - like markdown files
- Documentation for projects
- Data files - like language translation files
- Configuration files

VCS history

The history of version control goes back to the ‘60s and ‘70s.

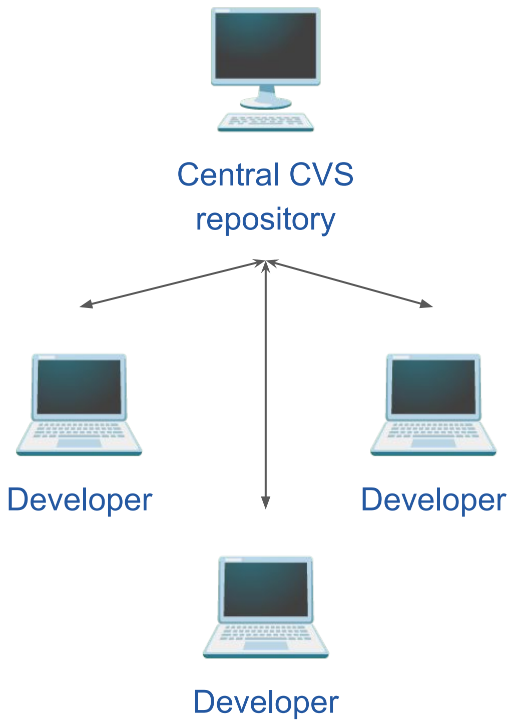
One of the first widespread version control systems was CVS (Concurrent Versions System), released in 1986.

CVS introduced a client-server model that became the standard for version control for years.

In this model, a central server stores the files and history data in a central *repository*.

Programmers acquire or “*check out*” copies of the repository (or parts of it) to their computer (the client).

After working, programmers “*check in*” their changes.



GIT

Digital Career Institute

git

Git was created by **Linus Torvalds in 2005** for development of the Linux kernel.

- Decentralized / Distributed
- Free and open source
- Very fast
- Scales well for **very** large project, like these public projects:
    - Linux 🐧
    - Visual Studio Code
    - React
    - Homebrew
- Rapid branching
    - Making a new branch is easy, fast and lightweight
- Used in practically all modern software development
- Large ecosystem of tools and services
    - Hosting services
    - Visualization tools
    - Integrations to development tools

Git CLI

There are many tools for using git, the main one is still the git CLI (Command Line Interface).

Meaning that a lot of git usage is done purely in the terminal, using the **git** program.

After installing git, you have to know a few central commands and concepts to work with it.

1. Any directory (folder) can be made into a git repository
2. Repositories can also be “cloned” from an existing source
3. You can have repositories inside repositories, but probably should not

Always stay aware of what directory you are working in. Usually it is best to have a system, at a minimalistic approach is to have one directory where you keep all your git repositories.

For example ~/projects/

It’s best to start a system immediately and follow it at all times to avoid confusion ❗

Documentation

Digital Career Institute

Documentation

1. Git Handbook
2. Git Flow
3. Git Glossary
4. Git Documentation
5. Git Developer Beginner
6. Git Config Cheatsheet
7. Git Cheatsheet

THANK YOU

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute
DCI