Introduction to Version Control

Git & Github

Key Study Skill: organisation

You will have to keep track of the following:

- Files prepared by staff which you downloaded (slides, codio, lab sheets etc.);
- Files you created (code in codio, solutions to exercises, additional tasks, different versions of all the above);
- Written notes from labs (things the lecturer said not written on slide, explanations about why code worked the way it did).

Version Control

A system of management for changes to a collection of information: This could be documents, web sites, or even movie scripts. Most commonly: code.

Basic idea:

- Call initial set of files "Version 1".
- The results after someone edits them is called “Version 2” and so on.

The system saves the old revisions and also metadata (e.g. who made each revision and when). Together all of this forms the **repository**.

Revisions can be compared, restored or even merged.

Also known as **Revision Control**.

Version Control for individuals

You are working on a program. You want to try out some changes but are not sure they will work.

- Option 1: Copy all files to a new directory and implement changes. If changes fail delete new directory and go back to old.

Version Control for individuals

You are working on a program. You want to try out some changes but are not sure they will work.

- Option 1: Copy all files to a new directory and implement changes. If changes fail delete new directory and go back to old.

Ok for a one-off but cannot do this too many times: cumbersome, at risk of human error, inefficient (especially if copying many unchanged files).

- Option 2: Use Version Control.

Version Control for groups

Version control is even more useful when working in a group. It has systems to avoid the problem of two people editing a file at the same time. E.g.

- When adding your changes you are informed of any others that have been made while working. Either merge them together or overwrite.
- A person working on a file *locks* it so others cannot edit.

Aids project management: everyone can see who is making what changes.

Why use Version Control?

- Systems usually easy to set up and use
- Full tracking of your project's progress
- Aids group programming
- Limits loss due to taking the wrong turn

Why would you not use it?

Why use Version Control?

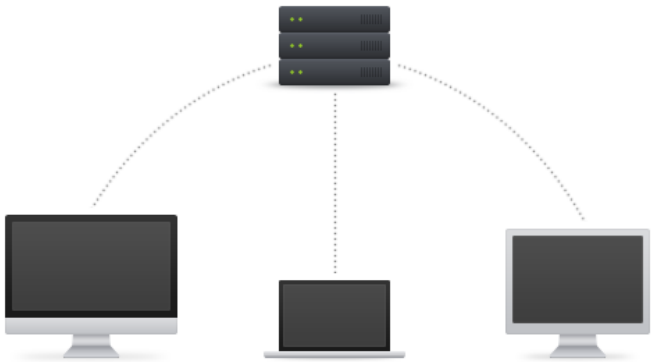
- Systems usually easy to set up and use
- Full tracking of your project's progress
- Aids group programming
- Limits loss due to taking the wrong turn

Why would you not use it?

All software development companies will use some form of version control. So **better to learn now** (this is an example of *professional practice*).

Types of Version Control 1

1. Centralised (client-server): There is one master copy of the files and their history somewhere (usually a server).

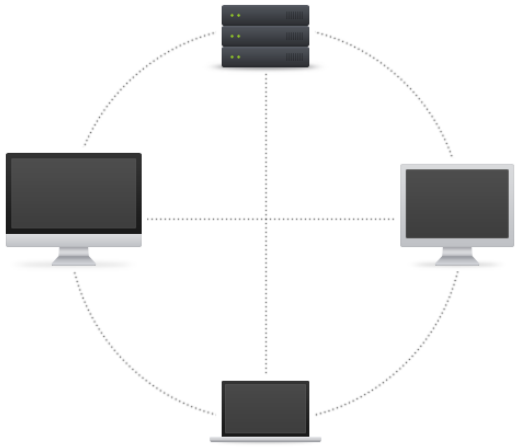


Developers download the files, edit them, and submit changes to the master copy.

Examples: Subversion (free open-source VC system); Perforce (proprietary software with advanced GUI).

Types of Version Control 2

2. Distributed: Each user has their own copy of entire repository (files and history).
   There may also be a master copy, but not necessary.



Developers use VC to record their changes on their own local repository. Whole repositories can then be copied and merged.

Examples: Git (free open-source VC system); Mercurial (actually written in Python)).

Centralised vs Distributed

Advantages of Distributed:

- VC for individual work is done locally: saves time in server communication; do not need internet connection.
- Can VC all your steps but only upload to master copy and show to others when you are all finished.
- Can share changes with individuals to get feedback before uploading to master copy.

Centralised vs Distributed

Advantages of Distributed:

- VC for individual work is done locally: saves time in server communication; do not need internet connection.
- Can VC all your steps but only upload to master copy and show to others when you are all finished.
- Can share changes with individuals to get feedback before uploading to master copy.

Disadvantages of Distributed:

- If files are very large it is not efficient to store whole repository many times. (N/A for programming code).
- If project has very long history (> 50,000 revisions) then storing it multiple times is inefficient. (N/A for most projects).

We will be using **Git**

- Distributed Version Control System
- Free to download and open source
- Very widely used:
  - Certainly the most popular DVC
  - Subversion the most popular CVC.
  - Currently which is the most popular overall is debatable - but Git is still growing very fast.

Git History

- Created to replace the proprietary VC system used for Linux kernel development.
- First released in 2005.
- Rapid growth partly due to the success of GitHub.

Git creator Linus Torvalds (also founder of the operating system Linux)

Difference between Git and GitHub

- Git is free open source software (with command line interface).
- GitHub is a company
  - Operates website to host Git repositories;
  - Includes social-network functions (feeds, followers).

GitHub founder and CEO Tom Preston-Werner

- There are alternatives to GitHub, e.g. **GitLab**.

Key Git Commands

Most important commands when using git in terminal:

- `git init`
  To create a new repository. Or
- `git clone <url>`
  To download an existing repository. E.g.
  git clone https://github.com/Matthew-England/121COMWeek2Ex7.git
- `git add <filename>`
  To add a file for the next commit.
- `git commit –m “message”`
  To locally version control changes on your computer.
- `git push` `git pull`
  To globally sync with the master repo (e.g on GitHub).

Return to Zero

EEWeb.com