Berner Fachhochschule - Technik und Informatik

Object-Oriented Programming 2

Topic 0: Version Control Systems

Rolf Haenni & Andres Scheidegger

FS 2018

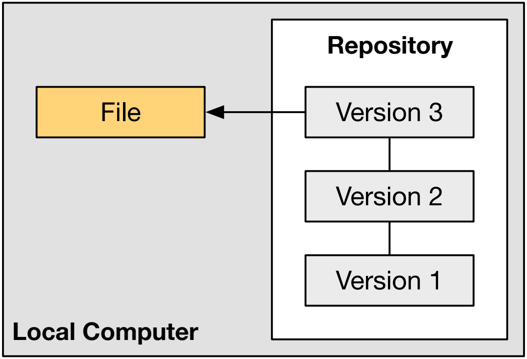
Version Control Systems

- Version control systems (VCS) are widely used in today's software engineering practice
- Benefits of using a VCS in software engineering:
  - → Software development in teams
  - → Remote collaboration
  - → Keep control over changes to source code
  - → Management of versions and updates
  - → Automatic backup
- Most popular VCS
  - → RCS (since 1982 almost not used anymore)
  - → CVS (since 1986, almost not used anymore)
  - → Subversion (since 2000, still in use today)
  - → Git (since 2005, the most widely used VCS today)

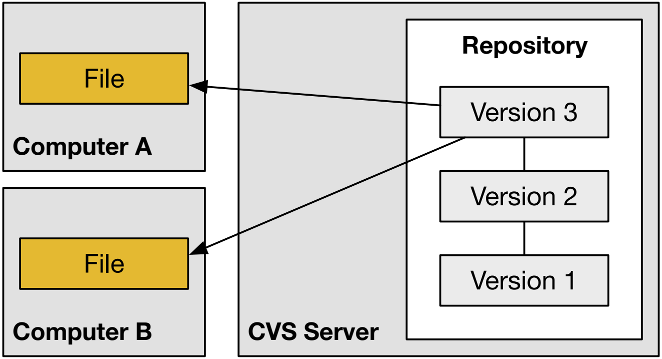
Git and Github

- Git is a free open-source VCS designed by Linus Torvalds
  - → Current version: 2.12.2 (March 2017)
  - → Web page: https://git-scm.com
  - → Tutorial: "Become a Git Guru"
- Git is a distributed VCS, which means that the whole version database (called repository) is stored everywhere
- A distributed VSC combines the advantages of a local (such as RVC) a centralized VCS (such as Subversion)
- Dedicated Git server software helps to add access control, manage multiple repositories, and display them on the web
- Two of the most popular Git servers are GitHub and GitLab

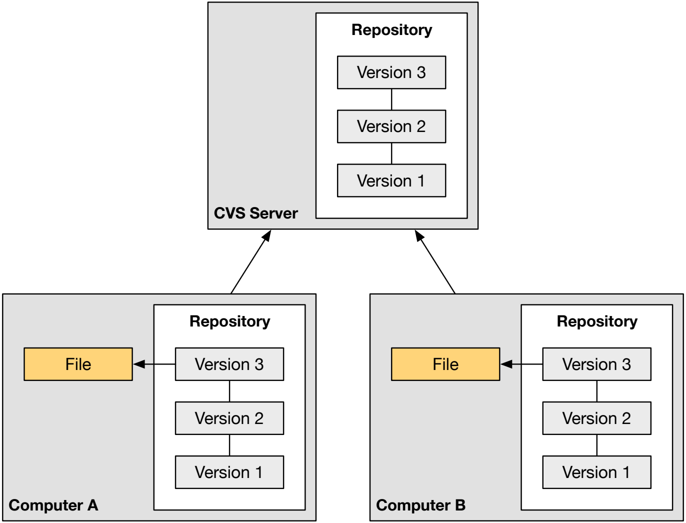
Local VCS


Repository
File
Version 3
Version 2
Version 1
Local Computer

Centralised VCS



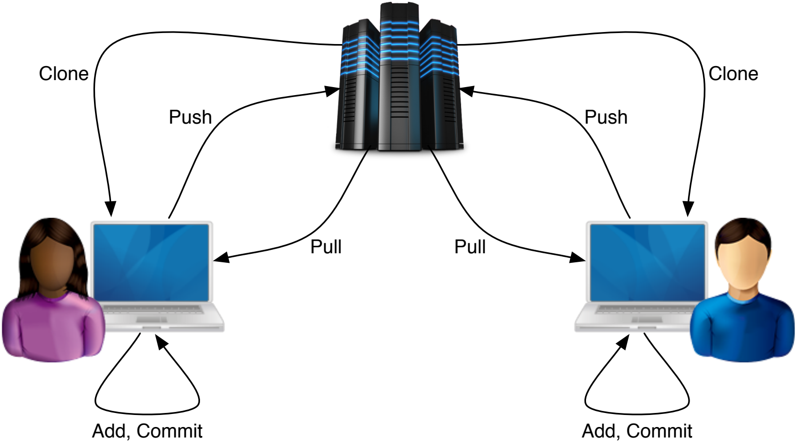
Distributed VCS



Git Commands

- `clone`: Makes a repository copy from a remote source
- `add`: Adds files changed locally to stage (ready to commit)
- `commit`: Includes staged files into local repository (the changes are described in the commit message)
- `push`: Sends the all changed files from local repository to remote repository
- `pull`: Fetches all changed files from remote repository and merges them with the local one (`fetch` followed by `merge`)
- Calling `pull` followed by `push` is called synchronization, but `sync` is not an official Git command

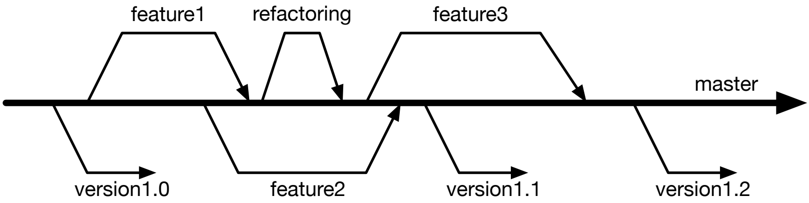
Git Workflow



Branching and Merging

- Git repositories can have multiple branches
- Initially, a Git repository contains a single master branch
- Creating branches keeps multiple streams of work independent from each other (e.g., new features)
- If independent work in a branch has stabilized, it can be merged into the master branch
- The release of a new software version (and its updates) can be defined as a separate branch
- Each team decides about the optimal workflow for its projects

Branching and Merging



Conflict Solving

- Collaboration in teams can lead to code conflicts
- Most of the potential conflicts are resolved automatically by Git during the merge operation
- A conflict appears when two developers edit the same line of code simultaneously
- In such a case, both lines of code appear in the merged document
- After solving the conflict (someone has to decide which code will be kept), `commit` and `push` operations are needed to update the merged files in the repository and the server

Conflict Solving

- Collaboration in teams can lead to code conflicts
- Most of the potential conflicts are resolved automatically by Git during the merge operation
- A conflict appears when two developers edit the same line of code simultaneously
- In such a case, both lines of code appear in the merged document
- After solving the conflict (someone has to decide which code will be kept), `commit` and `push` operations are needed to update the merged files in the repository and the server

Using Git in Practice

- Git is a command line tool, but there are various graphical Git tools to simplify its use
- For repositories hosted on GitHub, the tool GitHub Desktop is recommended
  - → History of commits
  - → Selection of current branch
  - → List of uncommitted changes
  - → Menus and buttons for basic Git commands (clone, commit, pull, push, etc.)
- Git is also directly accessible from IDEs such as Eclipse
  - → Install the EGit plug-in
  - → It is good practice to place any Git repositories outside the Eclipse workspace