S01 Software Development Tools

IDEs
Revision Control
gitlab and git

Integrated Development Environment (IDE)

- An editor to do more than just *write* code.
  - Syntax highlighting, completion, continuous compilation, testing, debugging, packaging, version control integration
  - Code analysis and refactoring capabilities
- Examples: Eclipse, IntelliJ, VisualStudio, Xcode
  - We will use IntelliJ

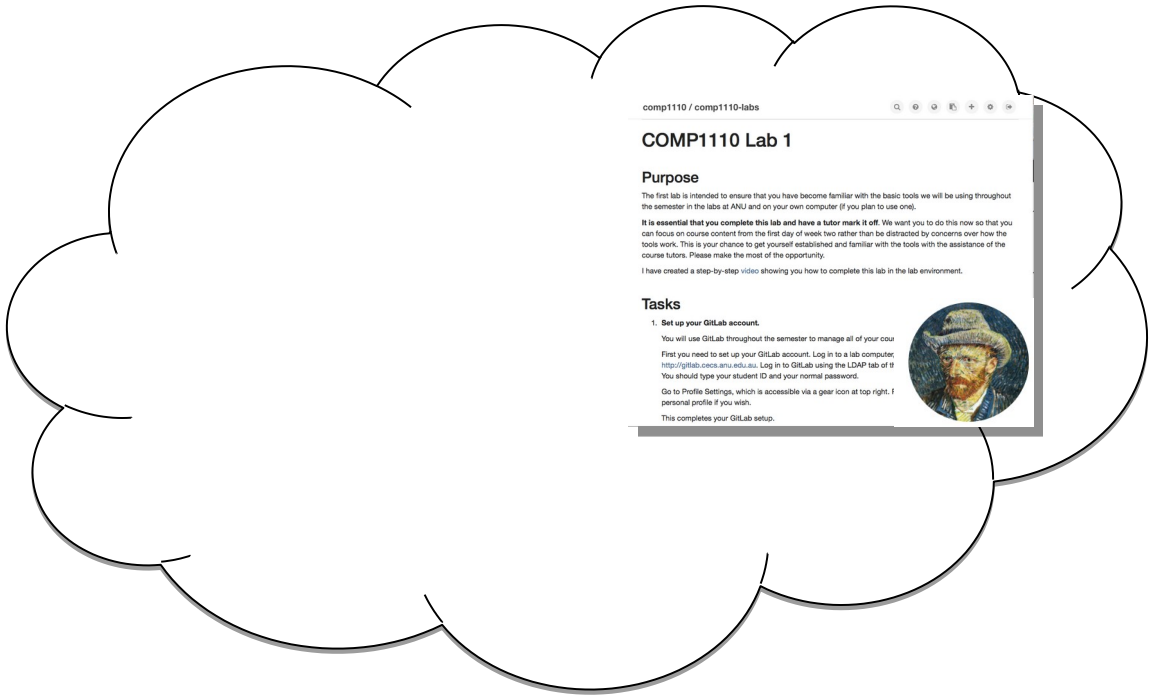
Version Control (VCS, RCS, SCM)

- Indispensable software engineering tool
- Solitary work
  - Personal audit trail and time machine
  - Establish when bug was introduced
  - Fearlessly explore new ideas (roll back if no good)
- Teamwork
  - Concurrently develop
  - Share work coherently

git & gitlab

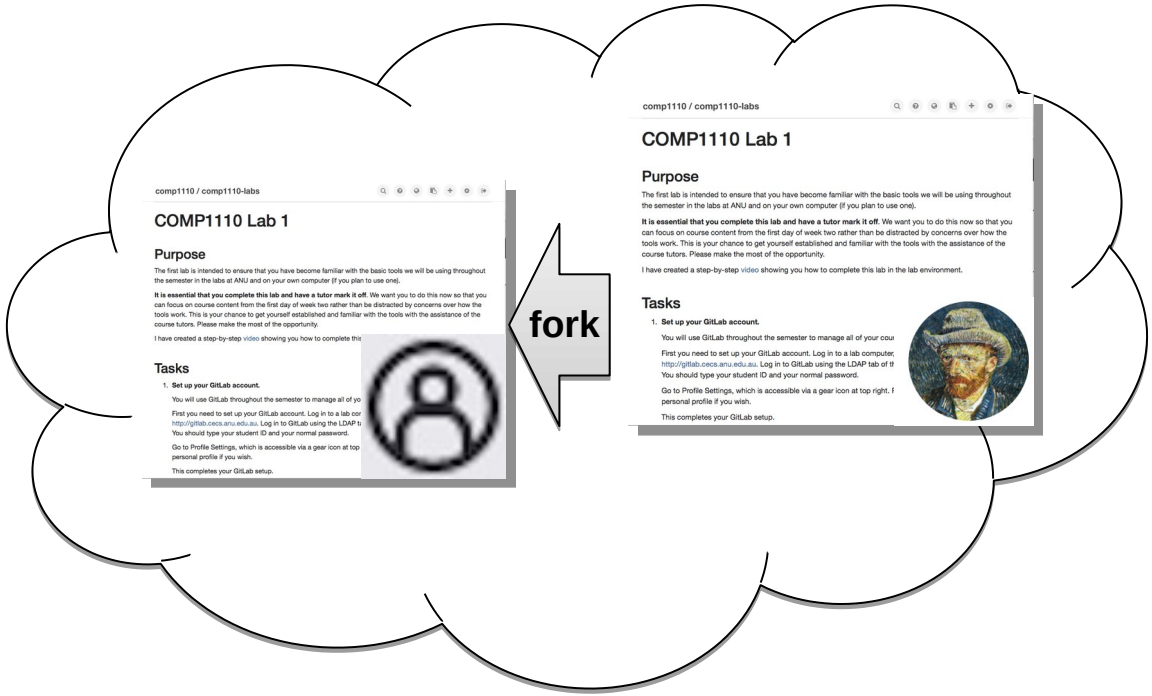
- We will use a distributed version control system – git – and a server – the ANU teaching gitlab – for sharing and submitting course work.
  - Distributed: git keeps entire project and its history in every copy.
  - The copy on the gitlab server typically serves as a shared copy (between you and us, or you and your team), and as backup.

git & gitlab



(master) labs repo
(owned by comp1110)

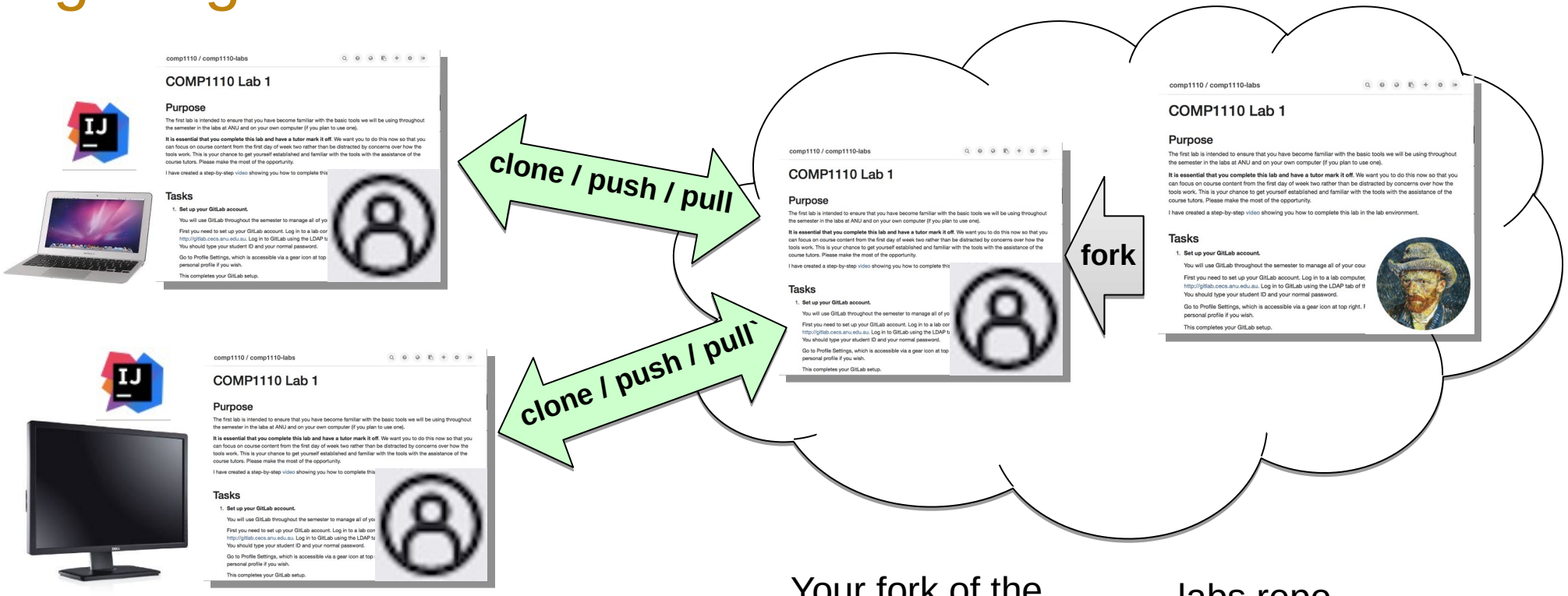
git & gitlab



Your fork of the labs repo (owned by you)

labs repo (owned by comp1110)

git & gitlab



Clone(s) of your fork of the labs repo
(owned by local user)

Your fork of the
labs repo
(owned by you)

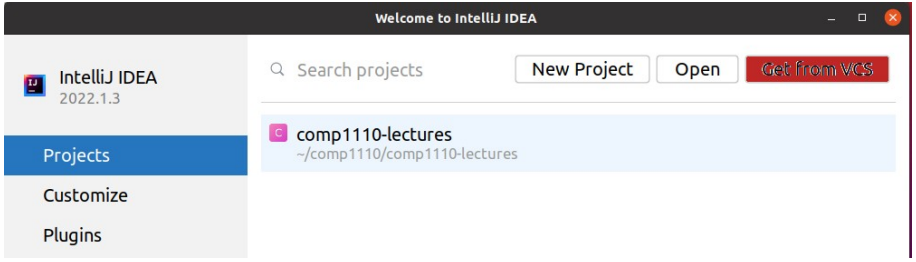
labs repo
(owned by comp1110)

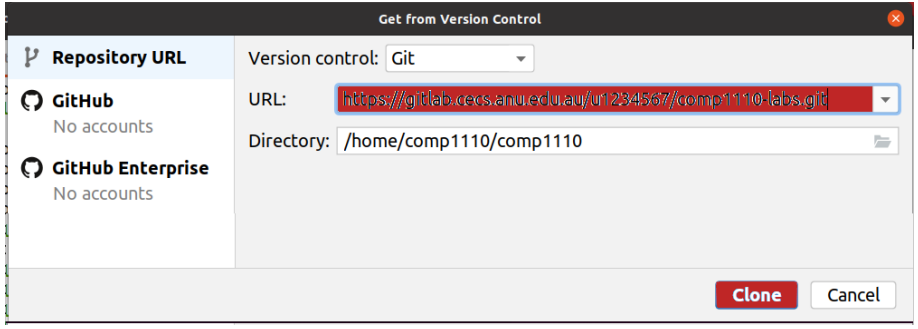
Recap

- **Repository** (“repo”): A copy of a project and its history.
- **Gitlab**: A server (remote) that stores repos
  - ANU teaching gitlab: https://gitlab.cecs.anu.edu.au
- **Clone**: A working (local) copy of a repo.
- **Pull**: Fetch updates from a remote to a working copy.
  - **Upstream pull**: Fetch updates from the master repo (that you forked)
  - https://comp.anu.edu.au/courses/comp1110/help/upstream-pull/
- **Push**: Send updates from a working copy to a remote.
- **Commit**: An update to a repo.

IntelliJ git integration

- Clone an existing repository:





- Other git operations: