



Version Control

Pertemuan 1










MK Algoritma Pemrograman II

Ika Qutsiati Utami, S.Kom., M.Sc.

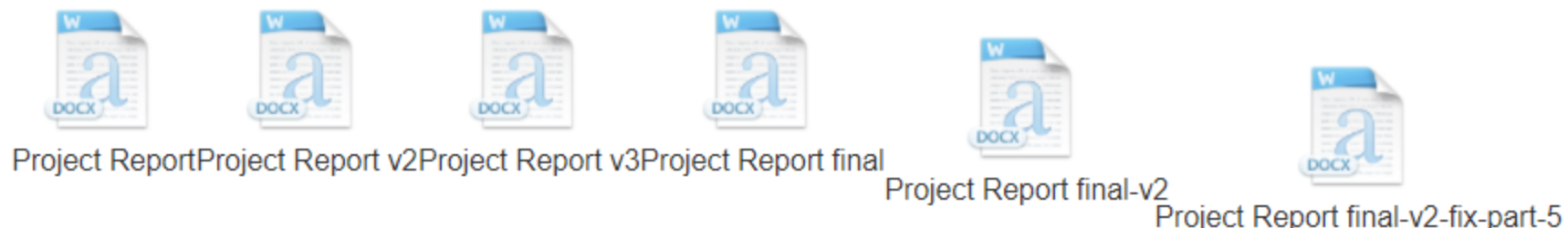
M. N. Fakhruzzaman, S.Kom., M.Sc.

Program Studi S1 Teknologi Sains Data
Fakultas Teknologi Maju dan Multidisiplin
Universitas Airlangga Indonesia

Background

Name	Date modified	Type
 Bismillah1	28/08/2021 9:33	Text Document
 Bismillah2	28/08/2021 9:33	Text Document
 Bismillah3	28/08/2021 9:33	Text Document
 Bismillah4	28/08/2021 9:33	Text Document
 BismillahFix1	28/08/2021 9:33	Text Document
 BismillahFix2	28/08/2021 9:33	Text Document
 BismillahPalingFix1	28/08/2021 9:33	Text Document
 BismillahPalingFix2	28/08/2021 9:33	Text Document
 BismillahPalingFixxxxxxxxxxxx	28/08/2021 9:33	Text Document

Keeping multiple copies of files with version numbers



Background

- VCS is essential tools of the **software engineering** world
- Without version control, coordinating a team of programmers all editing the same project's code will reach **pull-out-your-hair levels of aggravation.**



VCS you've already used

- Dropbox
- Undo/redo buffer
- Keeping multiple copies of files with version numbers
-

Version control terminology

- **Repository:** a local or remote store of the versions in our project
- **Working copy:** a local, editable copy of our project that we can work on
- **File:** a single file in our project
- **Version or revision:** a record of the contents of our project at a point in time
- **Change or diff:** the difference between two versions
- **Head:** the current version

VCS Supports

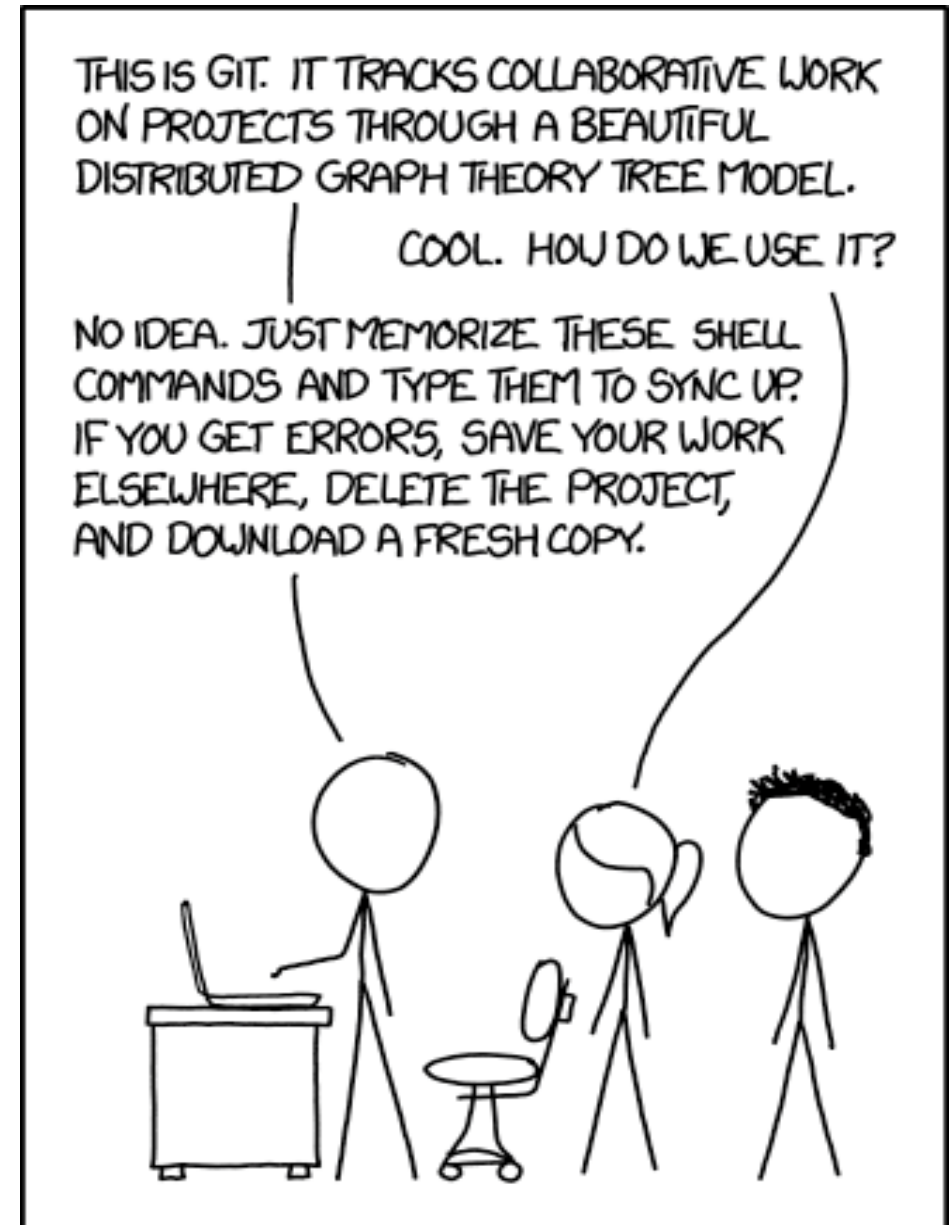
- **Reverting** to a past version
- **Comparing** two different versions
- **Pushing** full version history to another location
- **Pulling** history back from that location
- **Merging** versions that are offshoots of the same earlier version

VCS Features

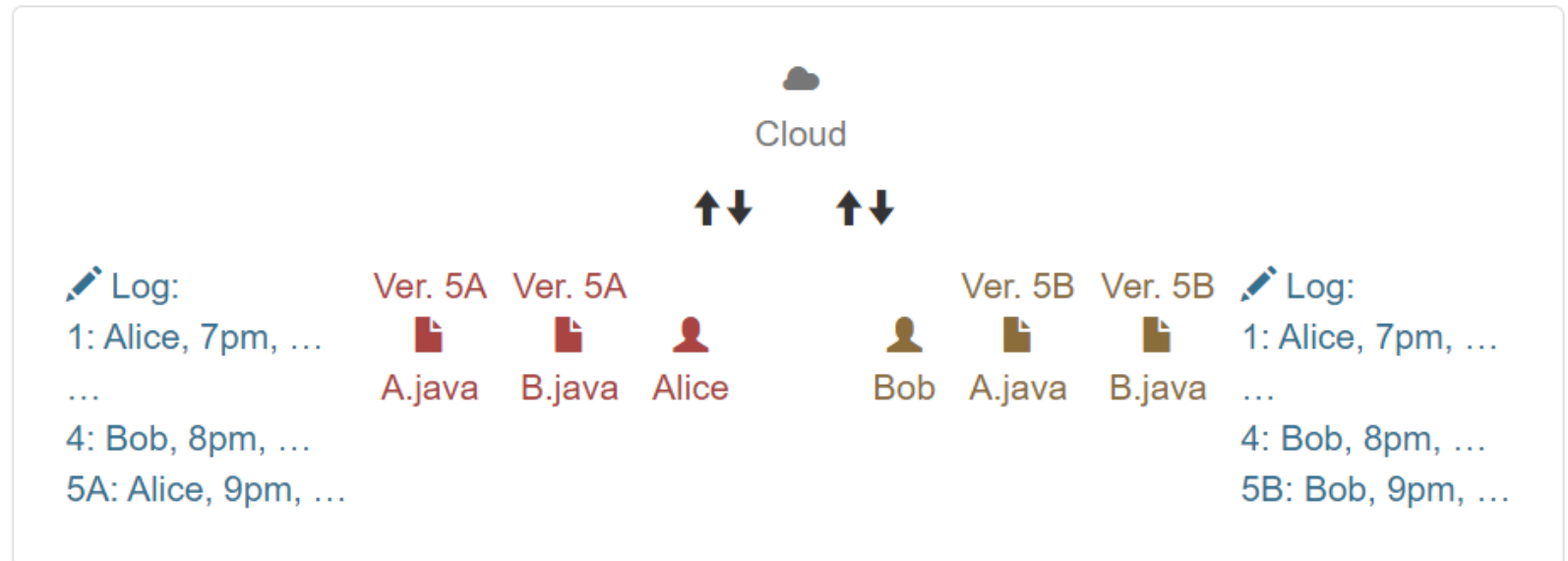
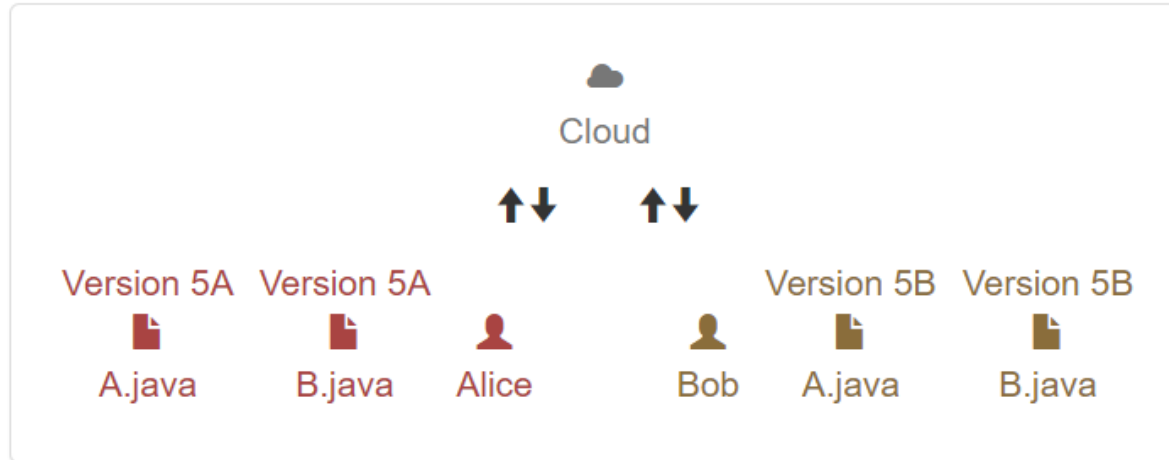
- **Reliable:** keep versions around for as long as we need them, allow backups
- **Multiple files:** track versions of a project, not single files
- **Meaningful versions:** what were the changes, why where they made?
- **Revert:** restore old versions, in whole or in part
- **Compare versions**
- **Review history:** for the whole project or individual files
- **Not just for code:** prose, images, ...
- **Allow multiple people to work together:**
 - **Merge:** combine versions that diverged from a common previous version
 - **Track responsibility:** who made that change, who touched that line of code?
 - **Work in parallel:** allow one programmer to work on their own for a while (without giving up version control)
 - **Work-in-progress:** allow multiple programmers to share unfinished work (without disrupting others, without giving up version control)

VCS Features

- Modern VCSs also let you easily (and often automatically) answer 3 questions like:
 1. **Who wrote this module?**
 2. **When was this particular line of this particular file edited? By whom? Why was it edited?**
 3. **Over the last 1000 revisions, when/why did a particular unit test stop working?**



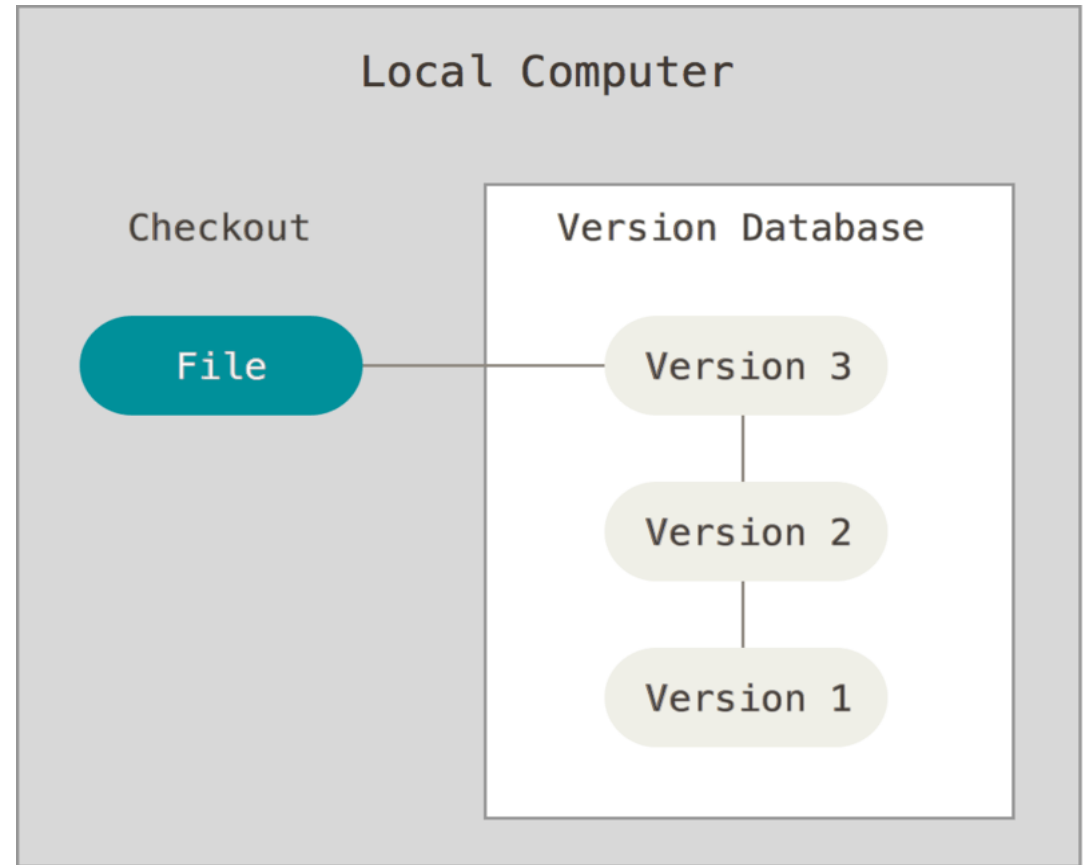
VCS for Multiple Developers



VCS type

1. Local version control systems

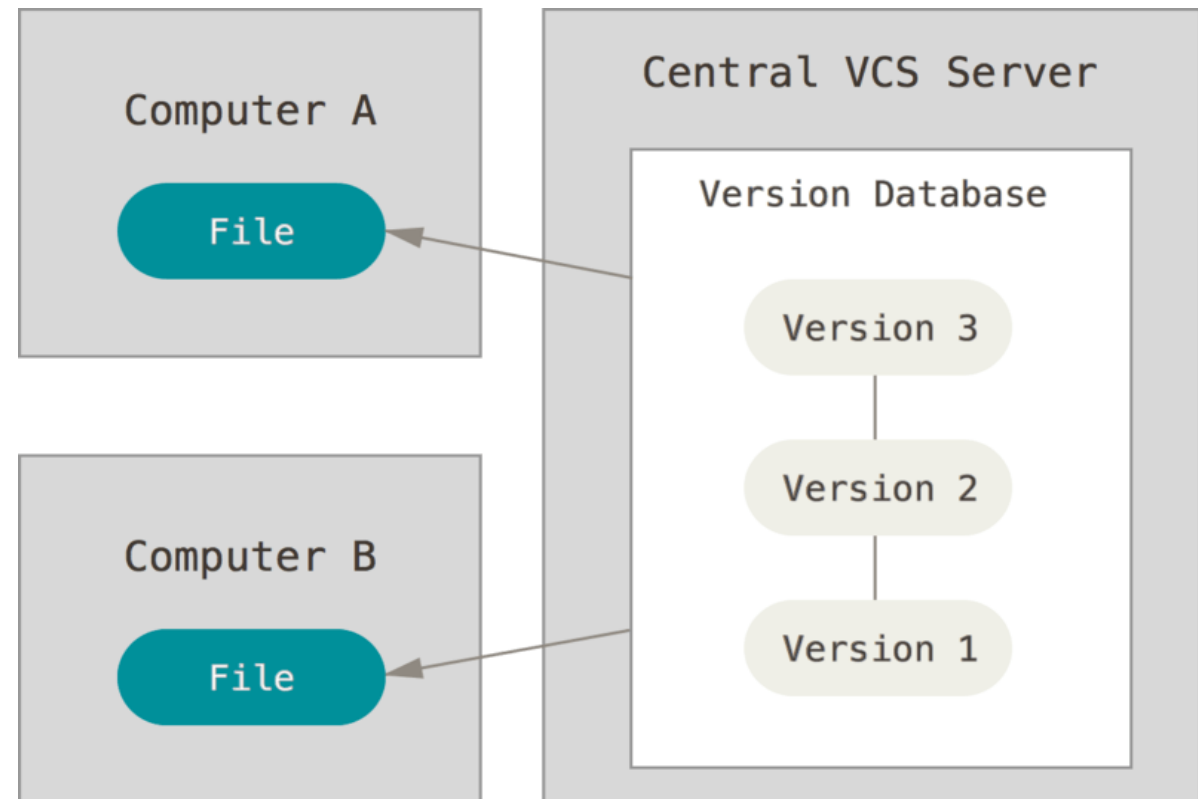
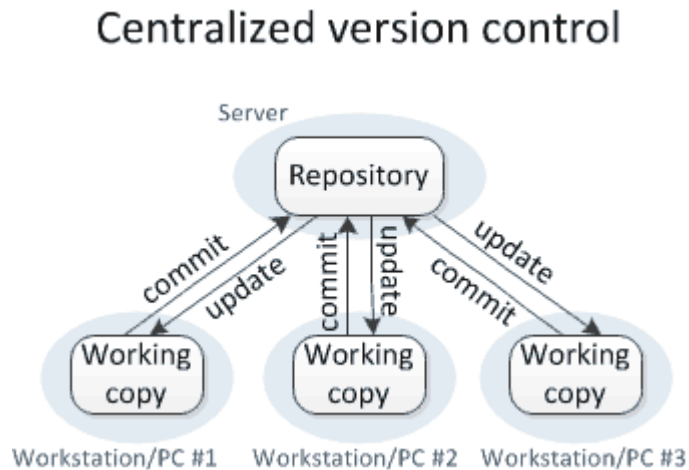
A simple discipline of saving backup files would get the job done.



VCS type

2. Centralized version control systems

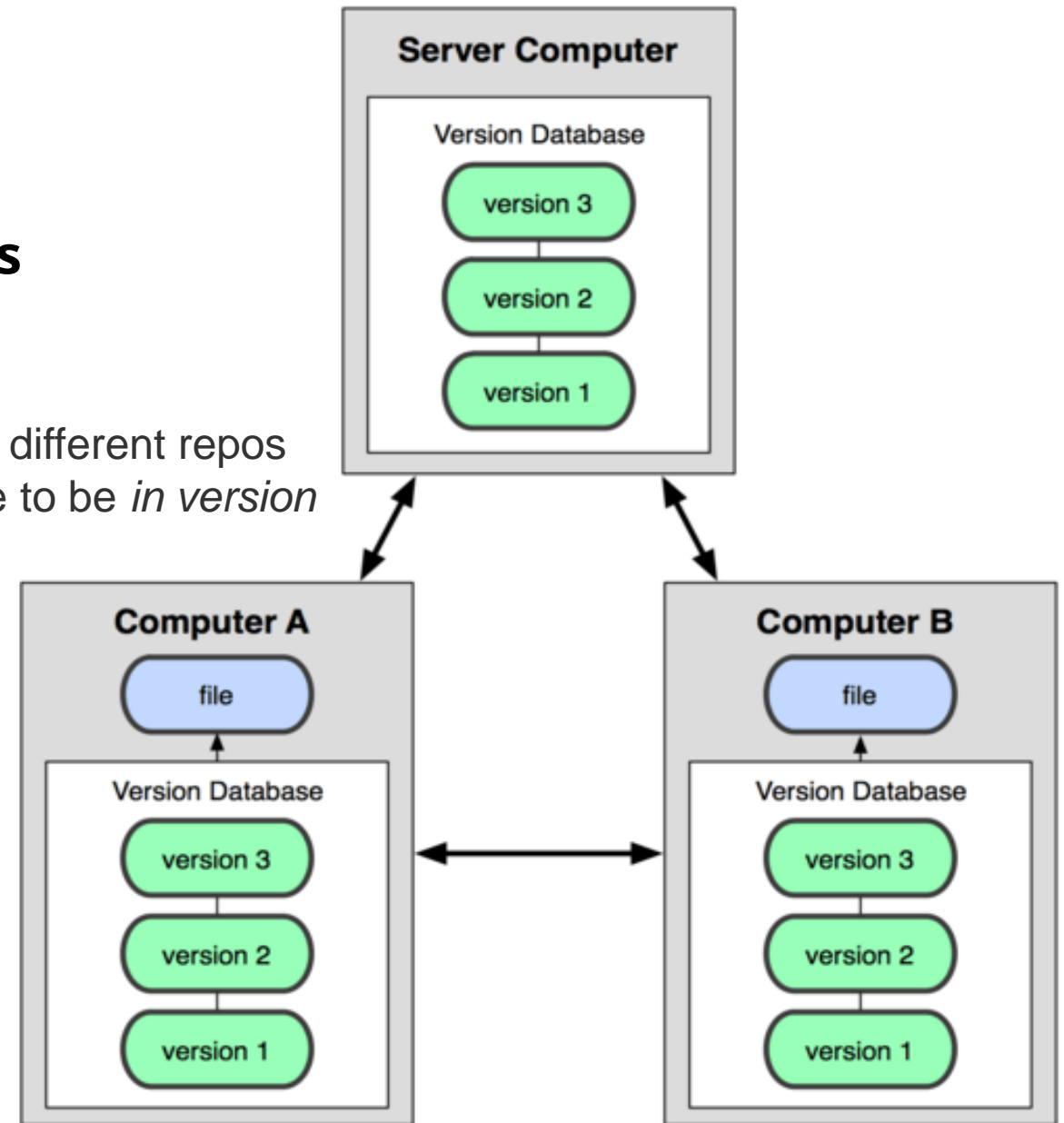
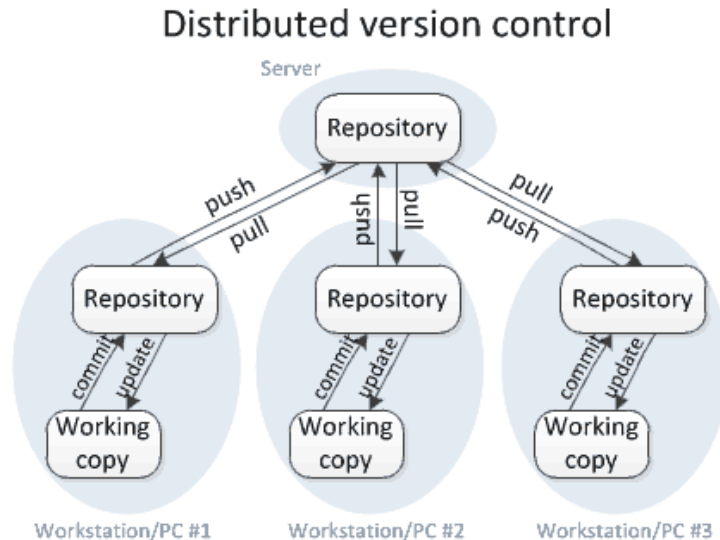
Everyone must share their work to and from the master repository, and a change is only *in version control* if it's *in the master repository*.



VCS type

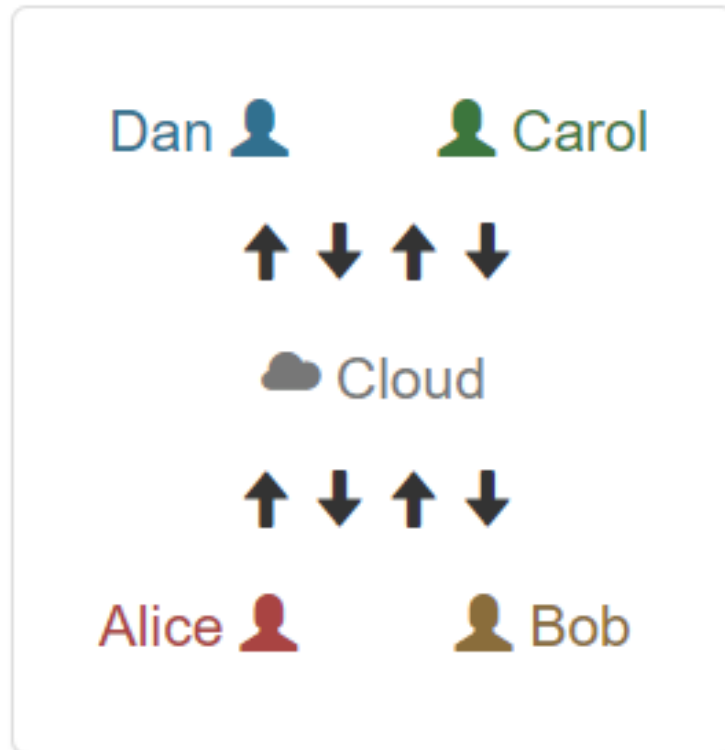
3. Distributed version control systems

- All repositories are created equal
- It's up to users to assign them different roles
- Different users might share their work to and from different repos
- The team must decide what it means for a change to be *in version control*.

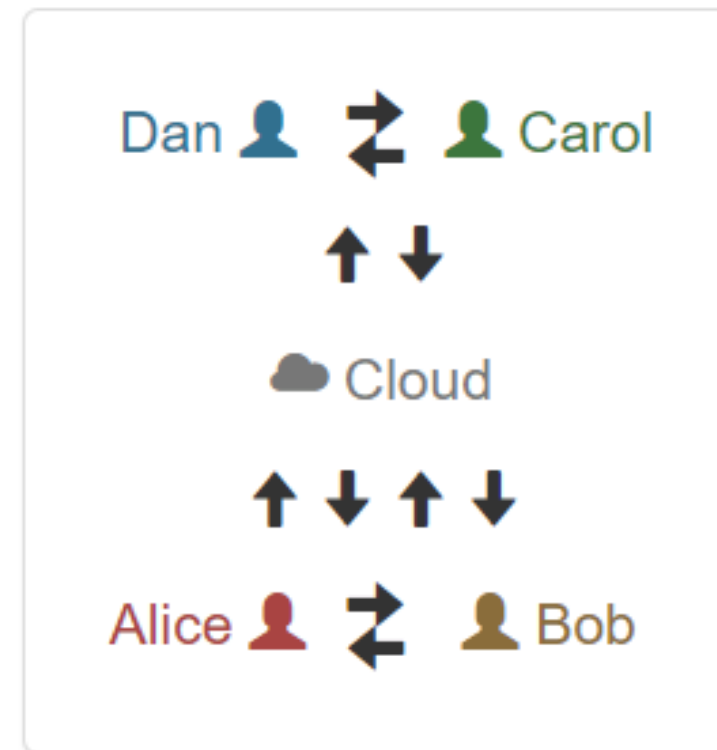


Distributed vs. Centralized

- Centralized



- Distributed



Version Control System

- Visual Source Safe (Microsoft)
- CVS (opensource) -> Subversion
- **GIT**
- Mercurial
- Monotone
- Bazaar
- TFS
- VSTS
- Perforce Helix Core
- IBM Rational ClearCase

Conclusion

- Version control and the big three

Safe from bugs

find when and where something broke

look for other, similar mistakes

gain confidence that code hasn't changed accidentally

Easy to understand

why was a change made?

what else was changed at the same time?

who can I ask about this code?

Ready for change

all about managing and organizing changes

accept and integrate changes from other developers

isolate speculative work on branches

Terima Kasih

Referensi:

[*http://web.mit.edu/6.005/www/fa14/classes/05-version-control/*](http://web.mit.edu/6.005/www/fa14/classes/05-version-control/)

[*https://missing.csail.mit.edu/2020/version-control/*](https://missing.csail.mit.edu/2020/version-control/)

[*https://homes.cs.washington.edu/~mernst/advice/version-control.html*](https://homes.cs.washington.edu/~mernst/advice/version-control.html)