Why Version Control?

What is not in Git does not exist

Dmytro Velychko

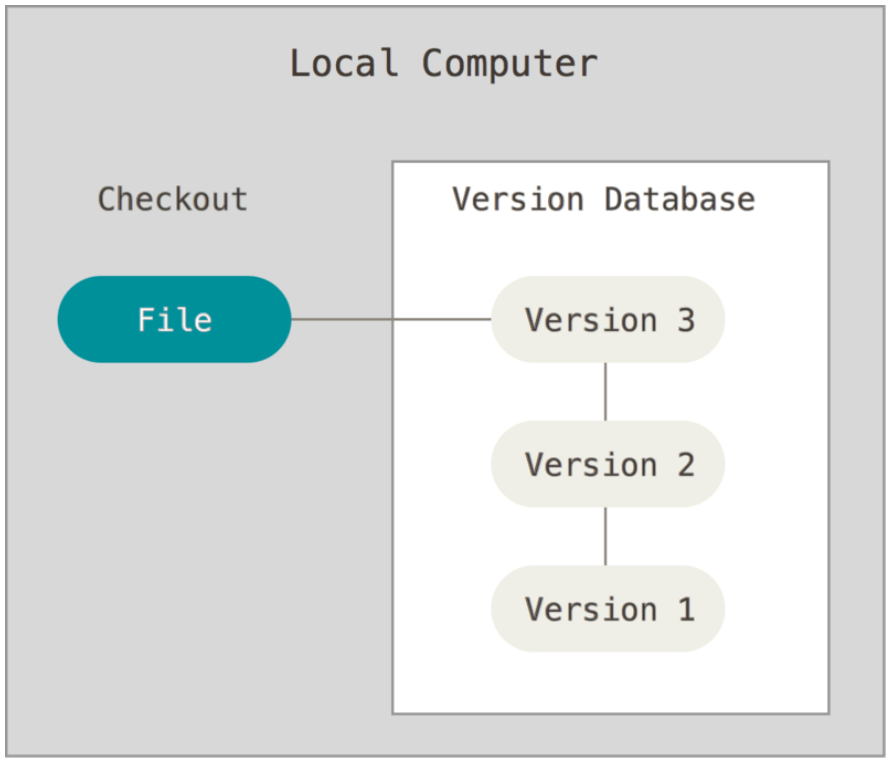
Theoretical Neuroscience lab
Philipps-Universität Marburg
IRTG-1901 "BrainAct"

Last edited: April 5, 2017

Outline

1. Why do you need a version control system?
2. Why Git
3. Git workflow
4. FAQ
5. Hands-on-keyboard examples

Why do you need a version control system?



- Safety. It is really hard to lose you work once it is in a repository.
- Activity track. You get the history of your work automatically.

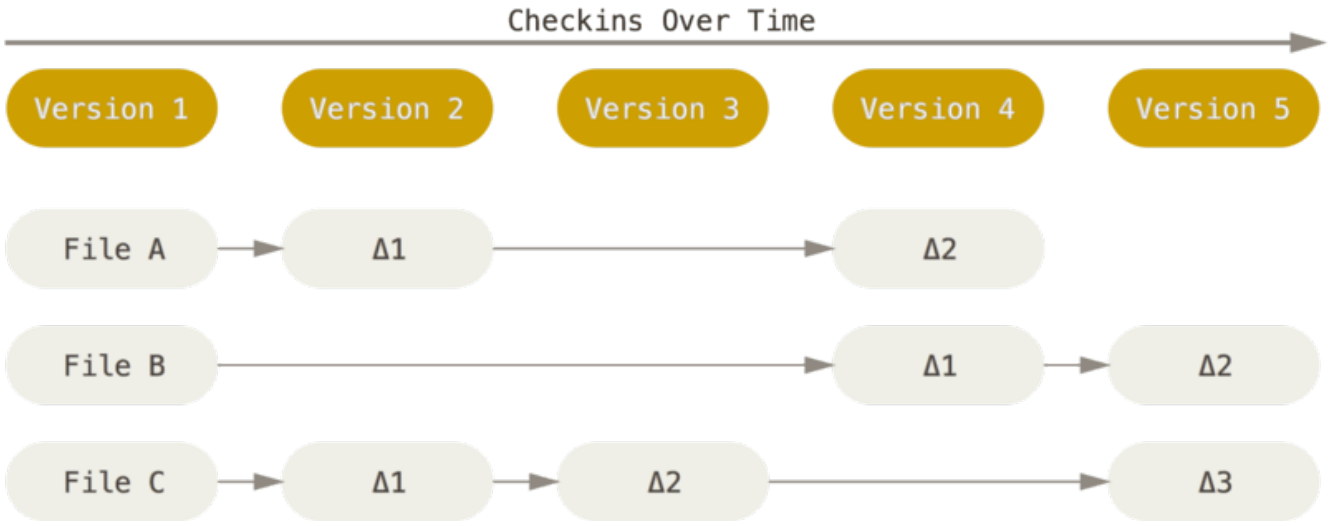
Why do you need a version control system?

- Self-discipline. Once you start following the version control guidelines, make contributions logical and granular, think about your work ahead...
- Collaborative work, sharing, merging.
- Versioning! You can always roll-back to an old stable version.
- Branching. If you have to make a huge change that could take many days, work on it in a separate branch, then merge into the main branch once the work is done. Thus, you don't interfere with your collaborators, the main branch stays always up-and-running.
- "Blaming". You may find the author of every byte in your repository.

Is it a back-up?

No! **Version control is not a backup!** It just tracks the changes in your project (plus much more).

What to version?



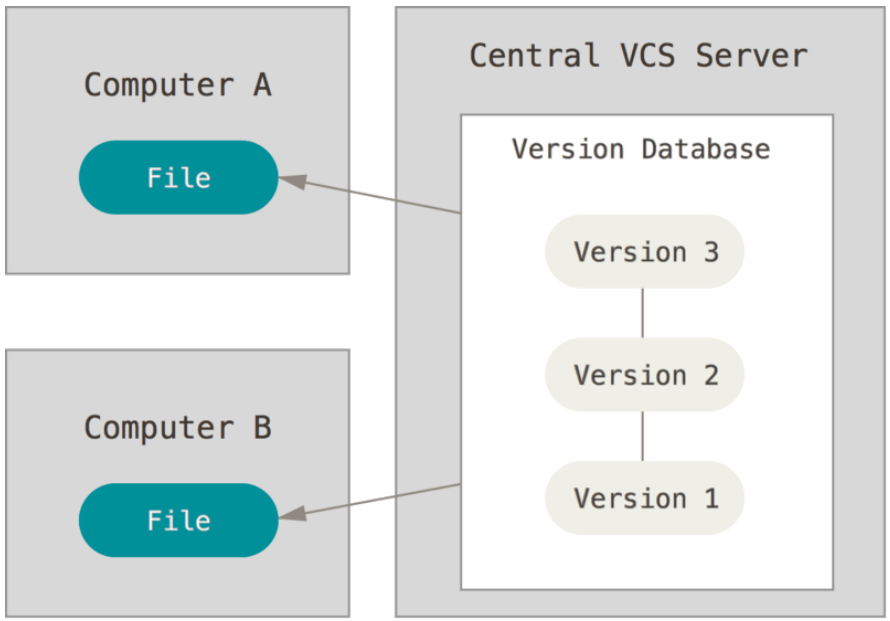
- Text (code, write-ups (LaTeX e.g.))
- Binary objects (raw experimental data, maybe post-processed and cleaned data, binary write-ups, like office documents, drawings)

Make sure you know how your VCS handles changes in large binary objects!

Version Control Systems

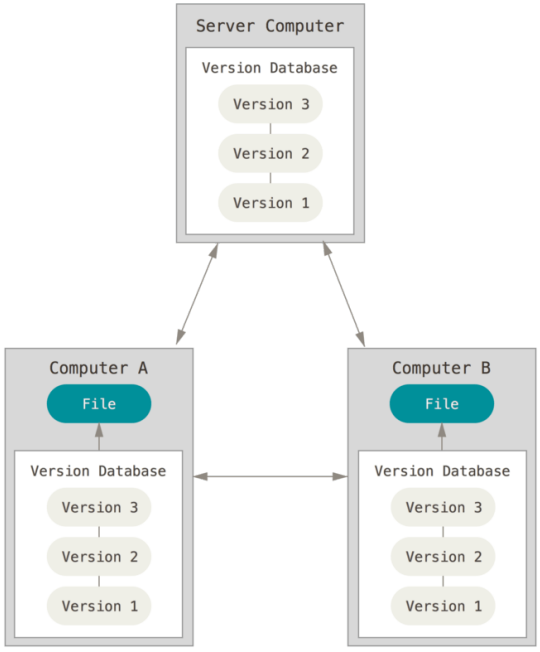
- Team Foundation Server, Team Services (Microsoft, commercial, proprietary, centralized)
- Subversion (Apache Software Foundation, free, open-source, centralized)
- Perforce Helix (Perforce Software, commercial, proprietary, centralized)
- Git (free, open-source, distributed, initially developed to maintain the Linux Kernel project)

Centralized vs Distributed. Centralized



- Client-server
- Requires connection to the central server to work

Centralized vs Distributed. Distributed



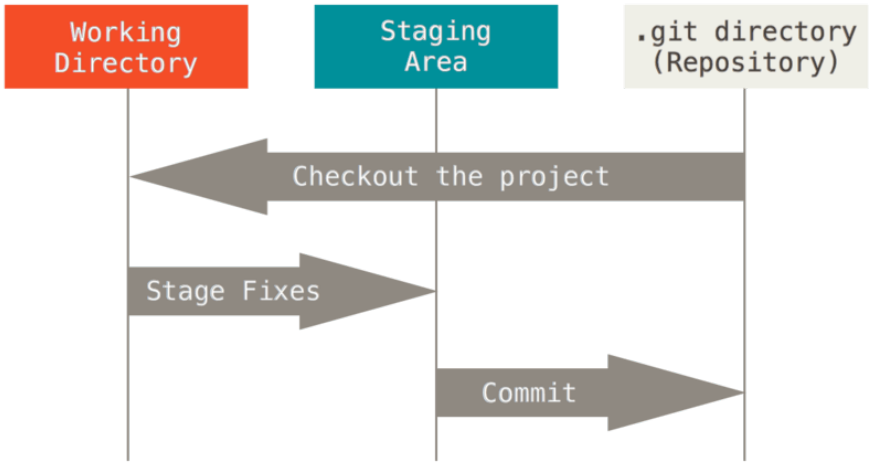
- Distributed. Peer-to-peer synchronization
- You can work off-line

Distributed doesn't mean there is no central repository!

Why Git?

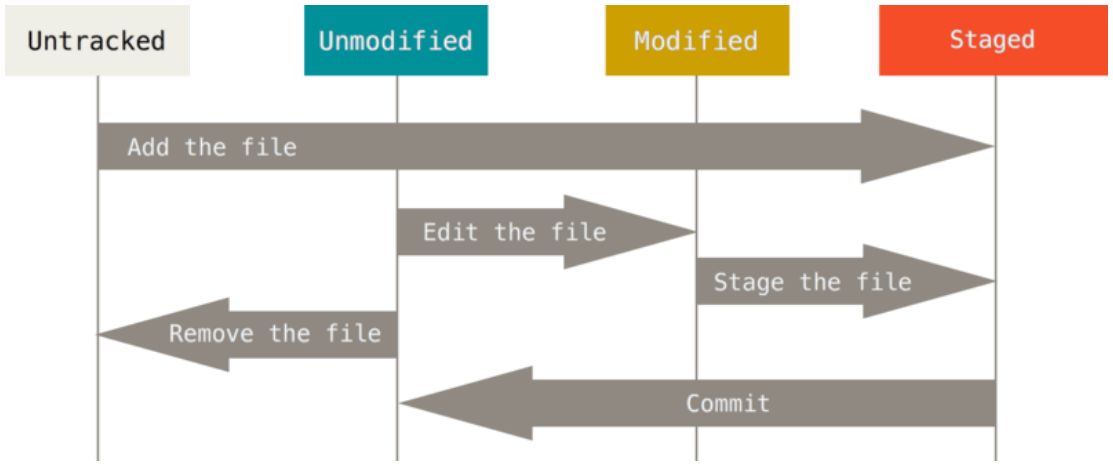
- Free, open-source, distributed, initially developed to maintain the Linux Kernel project
- Cheap version branching
- https://git-scm.com/ for the installation instructions, documentation, tutorials. (The official tutorial may be more comprehensible than this short presentation!)
- Some like to share it on GitHub (GitHub, Inc.), some use Atlassian BitBucket....

Git main concepts



- repository, origin, version, head
- fetch, checkout
- stage, commit, revert, push, pull
- branch, merge, rebase

Git workflow

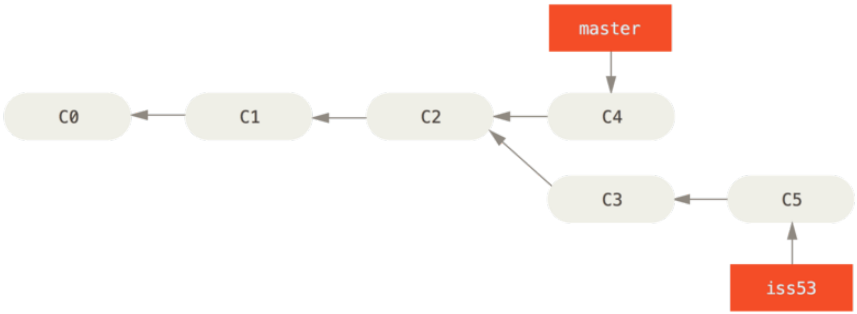


- create a new repository or pull an existing one
- add/remove files, make changes
- "stage" the files/changes you want to preserve
- make the commit with a descriptive message
- synchronize with remote repositories when needed

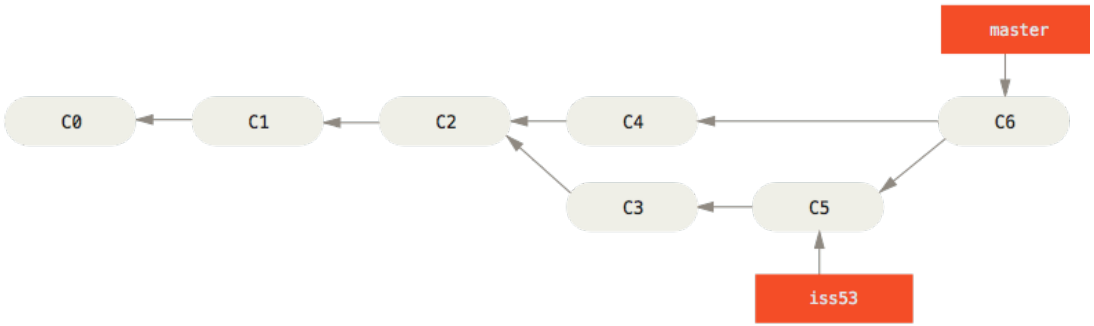
Branching, merging and rebasing

Let's say you want to make a huge change to you repository, like refactoring the codebase, extending your model, adding a new analysis method, updating your lecture slides. All these tasks may take days and many commits, but you wouldn't like to break the existing main repository. That's why one needs branches!

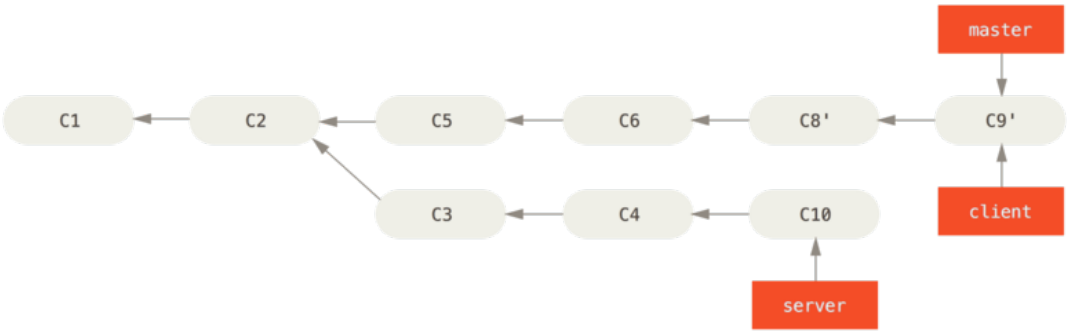
Branching, merging and rebasing



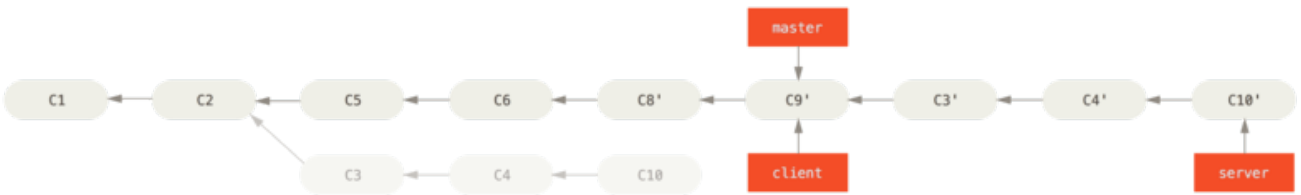
Merging joints-in one branch into another. Commits are merged in. Lattice-like history



Branching, merging and rebasing



Rebase cuts off the branch and reconnects the branch changes to another one. Linear history



A possible software development branching workflow

- master - should maintain production-ready state
- develop - integration branch. When reaches a stable state - merge into master
- feature/fX - every feature is implemented in a separate branch, then merged into develop
- release/vX - branched out from the master to release the product, then may include only fixes

A possible research branching workflow

- master - should be relatively stable, all work usually happens here
- feature/fX - implementation of large extensions, just to keep the master branch stable.

FAQ

- Q: How to start using Git?
- A: Go through the tutorials at the official web site, make sure you understand the basic concepts, play around.
- Q: Do I have to remember all those commands?
- A: No, there are graphical user interfaces. But knowing some of them doesn't hurt.
- Q: What is the best way of using Git for me?
- A: Git is a very flexible and powerful tool. You have to agree on some basic rules in your team. Be consistent and logical, respect the person who will use your project in the future. But it is up to you!

Hands-on-the-keyboard examples