INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Introduction to the version control system GIT

Great thanks to Christophe Gravier, Telecom St Etienne
his git course has largely inspired parts of this course

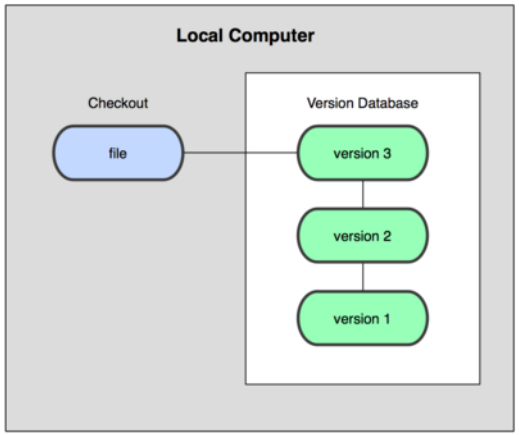
Version control system

- A version control system stores the changes of a set of files so that one can come back to previous versions
- It applies to any kind of document
- It is enormously useful for the management of code files

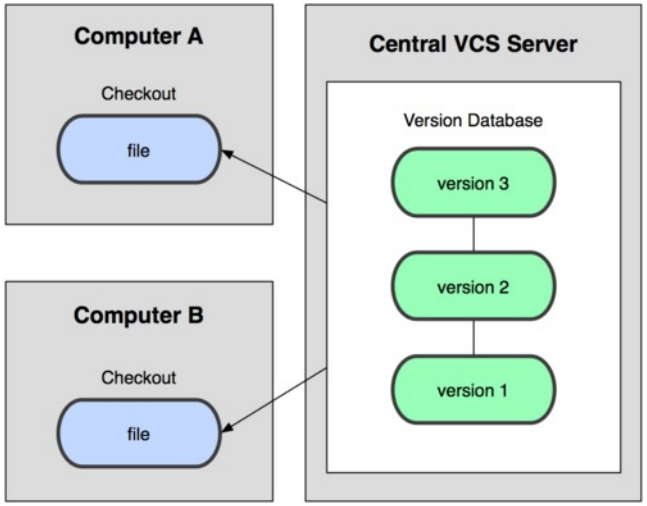
Version control advantages

- Save and restore in case of problem
- Synchronisation and sharing
  - Version control systems are often on shared servers
  - Sending code by email or social network does not scale
- Sandbox
  - if the last updates broke everything
  - When you keep a working version while doing a trial on a new idea/novel functionality
- Follow changes with short text
  - For each version, provide a text describing it

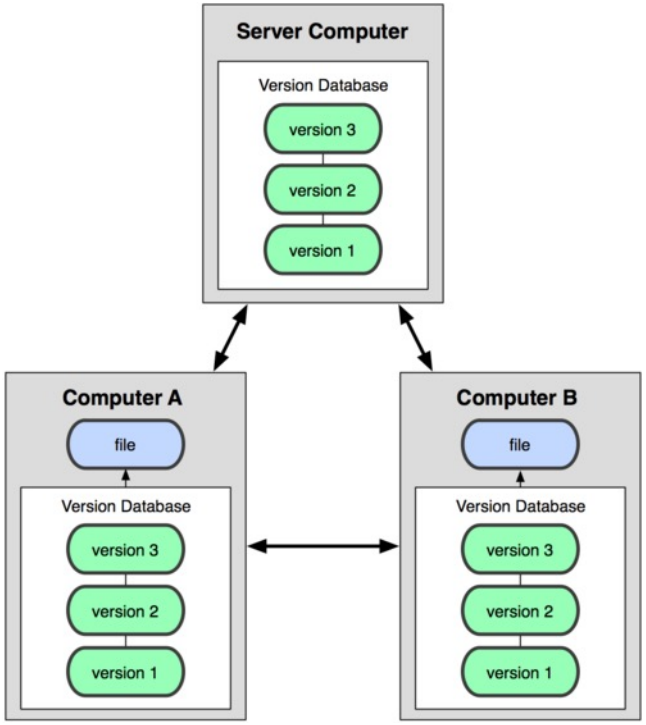
3 families of version control systems



Local



Centralised



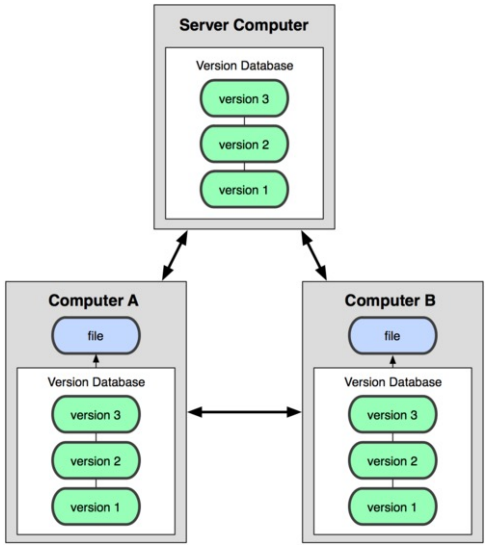
Distributed

From http://git-scm.com/book/en/Getting-Started-About-Version-Control

Git [gít]

- Created in 2005 to manage the Linux kernel
  - BitKeeper stopped being free
- Original characteristics
  - Rapid
  - Fully distributed
  - Simple
    - Still true for basic instructions
    - Many commands and options, not so easy to master all options
      - http://etnbrd.github.io/git-cheatsheet/
      - https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf

Git is fully distributed



- A project has a root directory
  - Root directory in the server (repo)
  - A local copy of the root directory on each user computer (*clone*)
- Local copies are independant…
  - Each copy lives its own changes
  - Changes are validated locally (*commit*)
- …and synchronize with the server
  - Validated local changes can be « published » to the server (*push*)
  - Changes sent by other local copies can be « downloaded » from the server (*pull*)

Git : almost everything happens locally

- A .git directory locally manages the different versions of my project files
  - The git commands make additions in the .git directory to « remember » what happened on the project files
  - The git commands cannot damage the .git directory: they only add information in a history log

Create a new versioned project with Git

- Create a directory with the name of the project (MyProject), move to the MyProject directory
- `git init`
  - It creates the .git directory under MyProject
  - .git is called the « local Repo »
  - .git is a database, git commands manage its data
  - Everything is ready!

Git : nothing is automatized

- Warning
  - It is not enough for everything to be tracked!
- Explicitly ask git to store a new version
  - No automatic saving
- Tell Git which files to track for the next version

Locally : prepare a version

- Create the index of files to track
  - `git add fic1.java fic2.java`

    (`git rm` removes from the index)
  - `git ls-files` lists the files in the index

  - One can also work conversely, i.e. :
    (a) list the files to ignore in file .gitignore
    (b) `git add .` Inserts all other files in the index

  - Example .gitignore for Java : https://tinyurl.com/gitignoreJava

Locally : save a new version

- `git commit`
- This command first asks for a text to describe the new version
  - Important : pay attention to your descriptions
- Each version receives an ID
  - Find a version ID: `git log`

Locally: Oops – backtrack on one file

- I made trash in my file and cannot manage
- I want to restore the file in the state of the previous version state
  - `git checkout <IDversion> <nomFic>`

Locally : Oops – full backtrack

- I want to go back to a version just before the last one
  - `git reset HEAD~1`
  - (HEAD is the last one)
- Go back to IDversion
  - `git reset <Idversion>`

Locally: branch

- To make trials on the side, independently from the main version (called `master` branch)
- `git branch nameMyBranch` creates the branch
- `git checkout nameMyBranch`
  - I will work on my branch without changing master
  - My commits concern the branch
- `git checkout master`
  - I go back to the master branch

Locally: merge branches

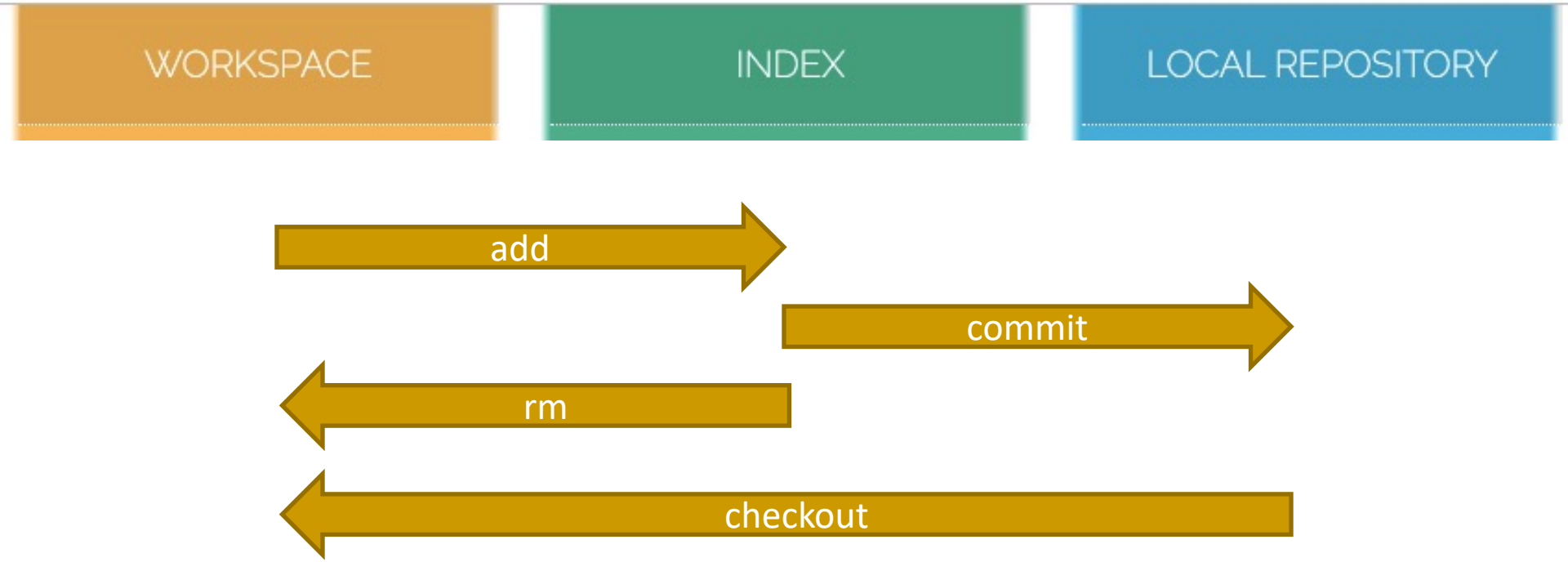
- Merge grabs all that was done in a branch to insert it in another
- `git checkout master`

  (go to master branch)
- `git merge nameMyBranch`

  (`master` « swallows » `nameMyBranch`)

Locally: Commands and files status

- `git status` shows the status of files
  - untracked / committed / modified / staged



And with the server?

- A repo on a Git server
  - E.g. github.com and gitlab.com are git servers on the Web that can be used for free (with restrictions)

EU servers

Create locally a copy of the repo on server

- `Git clone repoUrl`

Publish my work on the server: good practice

- **First** get publications from my teammates, **then** commit locally, **finally** publish all local commits to the server
- `git pull`
- (`git status` **and** `git diff`) **see later**
- `git commit`
- `git push origin master`

Oops, we worked in // on the same lines

- `Git status`
  - Identifies the lines with conflicts
- `Git diff`
  - provides a file that contains the duplicated and modified lines
  - By hand and for each line, choose the version to keep

Git in IDE and git GUIs

- Git is a set of shell commands
- Graphical user interfaces
  - « encapsulation » of commands
  - https://git-scm.com/downloads
- Modern IDEs propose Git
  - Dedicated plugins
  - Menus to make git commands

To learn more

- http://gitready.com/
- http://www-cs-students.stanford.edu/~blynn/gitmagic/
- http://git-scm.com/book
- http://gitimmersion.com/
- http://learn.github.com/
- https://www.kernel.org/pub/software/scm/git/docs/everyday.html

Training

- http://try.github.io/levels/1/challenges/1
- http://pcottle.github.io/learnGitBranching/