



Short Introduction to Git

Group Seminar

András Hartmann, Sascha Zickenrott

LCSB, Computational Biology group University of Luxembourg

June 29, 2016, Luxembourg

Content

1 Introduction

2 Git basics

- Install
- Commands
- Working with remote(s)
- Branches

3 GitLab

4 Git resources

Summary

1 Introduction

2 Git basics

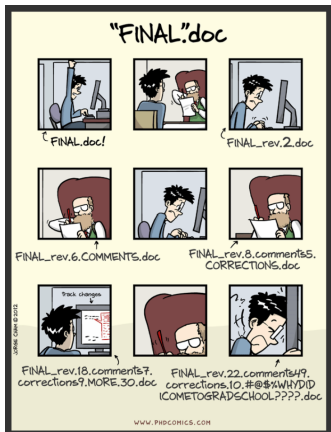
- Install
- Commands
- Working with remote(s)
- Branches

3 GitLab

4 Git resources

Why use Version Control Systems?

- Version Control = Revision Control = Source Control
- Allows you to track your files over time on a standardized (foolproof) way.
- Ever get files like Final_rev.22.comments49 ... doc?



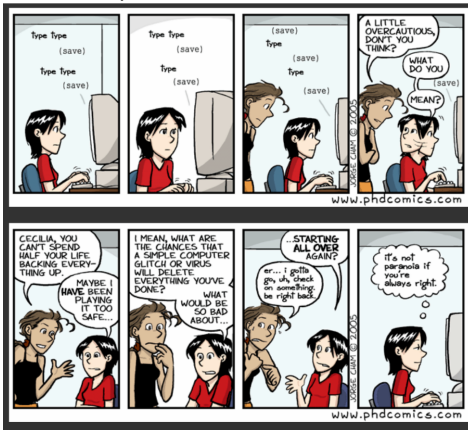
Why use Version Control Systems?

- Collaboration

- Sharing files
- Collaborative editing
- Review changes, trace problems
- Optimal team work-flow

- Backup & History

- Ultimate ctrl-z (undo)
- Remote / local storage
- Logs: comments to all revisions
- No more losing your files



What kind of data?

Version Control Systems

- Mainly **text** files
 - Source code, txt files, \LaTeX source, etc ...
- No sophisticated difference for binary files
 - Word, Excel documents, pictures, pdfs ...
- For \LaTeX collaborative editing you may want to try
 - sharelatex: <https://www.sharelatex.com>

Why git?

- Decentralized / Distributed
- Fast
- Flexible
- Works offline

Companies & Projects Using Git

Google

facebook

Microsoft

twitter

Linked in.

NETFLIX



PostgreSQL




Summary

1 Introduction

2 Git basics

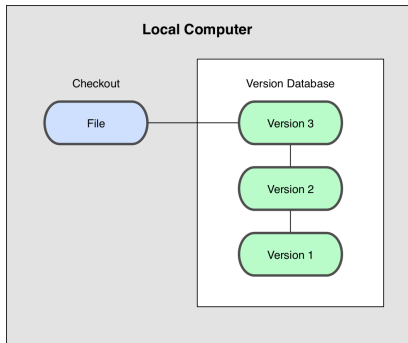
- Install
- Commands
- Working with remote(s)
- Branches

3 GitLab

4 Git resources

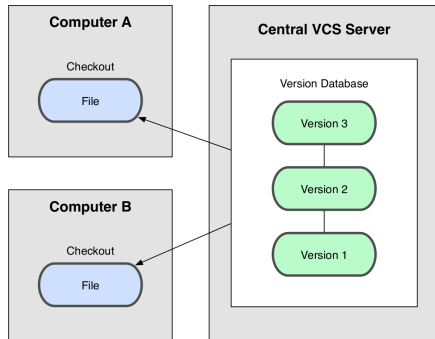
Local vs Centralized VCS

LOCAL



Not shared with anyone
e.g. on your own HDD

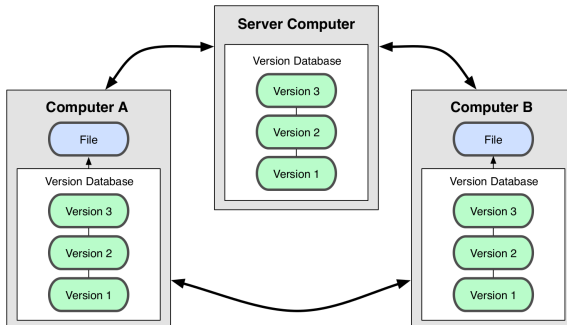
CENTRALIZED



Everybody sees the same version
on the server

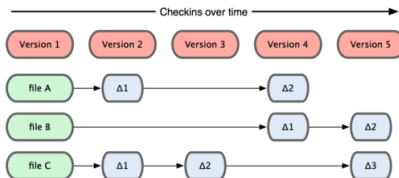
Distributed VCS

Git

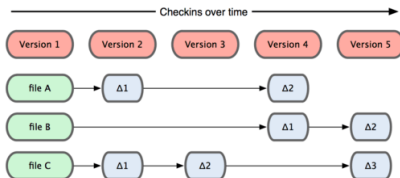


Both local AND global, but most operations are local
Everybody has the full history of commits

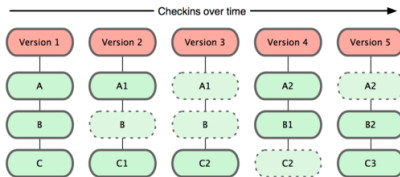
Most VCS



Most VCS



Git



“Git thinks of its data more like a set of snapshots of a mini filesystem.”

Install git

Linux / UNIX

- **OS X**

\$ brew install git

- **Ubuntu**

\$ aptitude install git

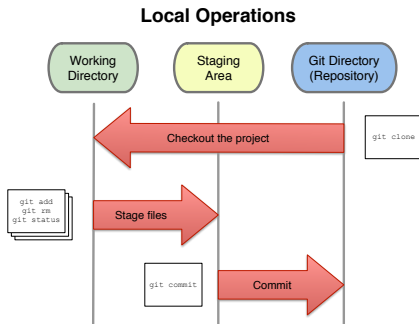
- **Arch**

\$ pacman -S git

Windows

- <https://git-for-windows.github.io/>

The three (local) Stages



- The **local repository** lives in the `.git` directory.
- The **staging area** tracks what will go into the next commit, kind of index

First steps

```
$ git init
```

Initializes a new git repository

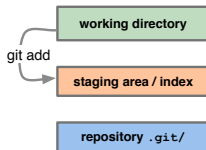
```
$ git clone [--recursive] <url> [<path>]
```

Clones a remote repository to access locally
url can be local / git / git+ssh / http(s), etc ...

Adding new files

```
$ git add [-f] [<pathspec> ...]
```

Adds new file(s) to the index

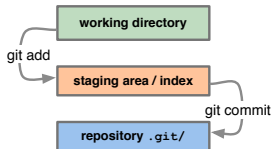


Committing changes

```
$ git commit [-a] [-m "msg"] [<paths> ...]
```

Commits the changes

-a: commits all, -m: write a message

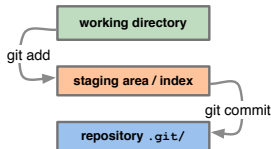


Committing changes

```
$ git commit [-a] [-m "msg"] [<paths> ...]
```

Commits the changes

-a: commits all, -m: write a message



IMPORTANT:

- Always write a descriptive message to your commit!
- In the commit description list all files you changed, and describe why
- In general, try to commit often, commits are save points
- Do not commit code that does not run

Moving / deleting files

```
$ git rm [-rf] [-- cached] [<pathspec> ...]
```

–cached : removes from Staging area
default: from index **and** file system

```
$ git mv <source> <destination>
```

Moves source to destination

```
$ git status
```

Show the working tree status: differences between the index file and current HEAD commit

```
$ git diff [-cached] [<ref>][<pathspec> ...]
```

Check un-staged changes (line by line)

– cached: check staged changes

Can be relative to a revision, like 1776b2, or HEAD

In general it is a good practice to check status and diff **before** committing

```
$ git log [-since="date"] [<pathspec> ...]
```

time-filtering: `-since=2.weeks` or
`-since="2 years 1 day 3 minutes ago"`

```
$ git blame <file>
```

Who was it?

```
$ git commit —amend
```

Changes the last commit

```
$ git unstage <file>
```

```
$ git reset HEAD <file>
```

Unstage staged file

```
$ git checkout <file>
```

Restores file to last commit. DANGER: all the changes lost!

```
$ git revert <commit>
```

Reverts a <commit>: makes a new commit that undoes **all** the changes made in <commit>

.gitignore

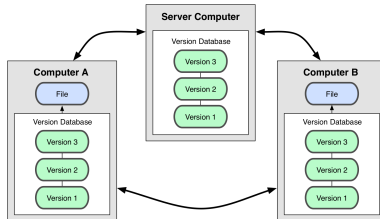
If present in a folder, tells git to ignore files

.gitignore example

```
*.pyc  
*.swp  
/build/  
/doc/[abc]*.txt  
.pyirc  
*.egg-info
```

- Blank lines or lines starting with # are ignored (comments)
- Standard glob patterns work (wildcards)
- End pattern with slash (/) to specify a directory
- Negate pattern with exclamation point (!)

Remotes



- Other clones of the same repository
- Can be local (another checkout) or remote (coworker, central server)
- There are default remotes for push and pull

```
$ git remote -v
```

```
origin git://github.com/schacon/ticgit.git (fetch)
```

```
origin git://github.com/schacon/ticgit.git (push)
```


Remotes

```
$ git pull <remote> <rbranch>
```

Or simply:

```
$ git pull
```

Pulls the commits from the remote

```
$ git push -u <remote> <rbranch>
```

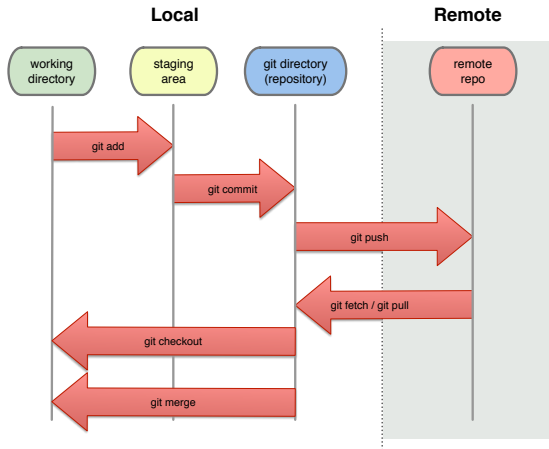
Using defaults:

```
$ git push -u origin master
```

Pushes all the commits into the remote

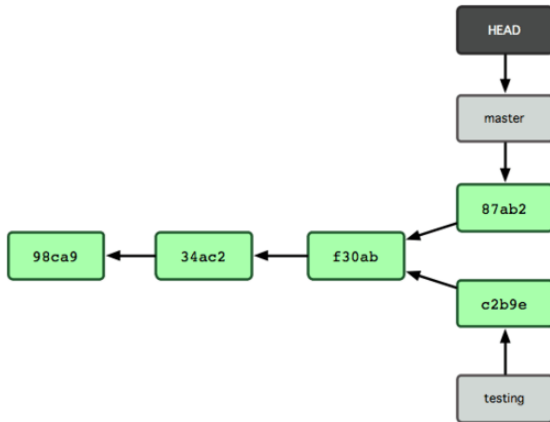
IMPORTANT: do not simply push to someone else's repository
create a pull request instead

Local vs Remote



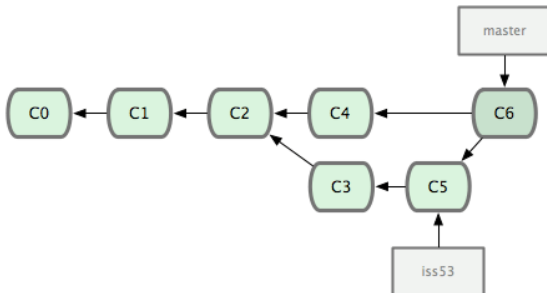
Branches

- Branches are “Pointers” to commits
- Branches can diverge during development



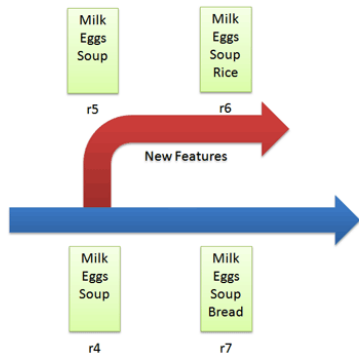
Merging

- Merge: “joining branches”: usually painless
- Conflicts the same line has changed
 - Have to be resolved (manually / automatically)



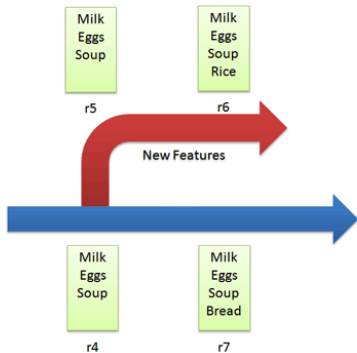
Typical VCS Workflows

Branching

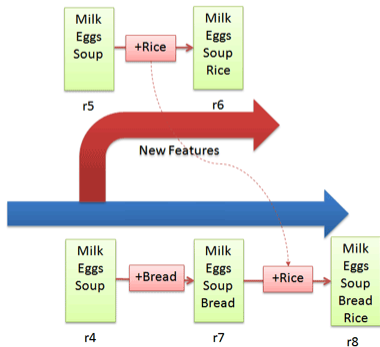


Typical VCS Workflows

Branching



Merging



Summary

1 Introduction

2 Git basics

- Install
- Commands
- Working with remote(s)
- Branches

3 GitLab

4 Git resources

- GitLab is an online git interface (like github)
- Available at <https://git-r3lab.uni.lu>

LCSB R3 GitLab



Manages git repositories hosted at LCSB and powered by the Bioinformatics Core Group.

[Uni.lu](https://uni.lu) | [LCSB](https://lcsb.lu)



Summary

1 Introduction

2 Git basics

- Install
- Commands
- Working with remote(s)
- Branches

3 GitLab

4 Git resources

Further reading

- "Pro Git" Book by ...
→ <http://git-scm.com/book>
- Git reference
→ <http://git-scm.com/docs>
- Git cheatsheet
→ <https://www.git-tower.com/blog/git-cheat-sheet>
- This presentation is on github
\$ git clone <https://github.com/AndrasHartmann/gitpres.git>
- You can always check:
\$ help git [<command>]

Learning by doing

- Tutorials
→ <https://www.atlassian.com/git/tutorials>
- Learn Git on codecademy - Strongly recommended!
→ <https://www.codecademy.com/learn/learn-git>
- Most important:

