The GIT

Programming Concepts in Scientific Computing

EPFL, Master class

September 20, 2023

Source repository

What would you demand to a tool that will hold your program sources ?

Source repository

What would you demand to a tool that will hold your program sources ?

- ▶ Manage history (evolution in time)

Source repository

What would you demand to a tool that will hold your program sources ?

- ▶ Manage history (evolution in time)
- ▶ Rewind time

Source repository

What would you demand to a tool that will hold your program sources ?

- ▶ Manage history (evolution in time)
- ▶ Rewind time
- ▶ Transport/Backup through network

Source repository

What would you demand to a tool that will hold your program sources ?

- ▶ Manage history (evolution in time)
- ▶ Rewind time
- ▶ Transport/Backup through network
- ▶ Team/Concurrent working

Source repository

What would you demand to a tool that will hold your program sources ?

- ▶ Manage history (evolution in time)
- ▶ Rewind time
- ▶ Transport/Backup through network
- ▶ Team/Concurrent working

This is the standard of most **Version control systems** such as **GIT** or **SVN**.

GIT

- Git is a free distributed version control system (DVCS), used for source code management (SCM)

- Git operates on a decentralized architecture, so every git working directory has the complete history

- Git was initially designed and created by Linus Torvalds for Linux kernel development

- EPFL has GIT repository services (`http://c4science.ch` and `http://gitlab.epfl.ch`)
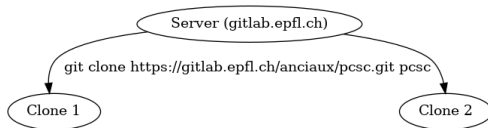
GIT - Cloning

```
git clone https://gitlab.epfl.ch/anciaux/pcsc.git
```
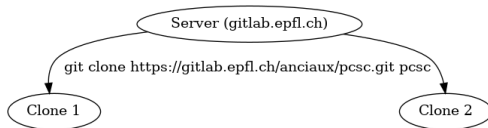
GIT - Cloning

```
git clone https://gitlab.epfl.ch/anciaux/pcsc.git
```


- ▶ The *working copy* is the state (can be modified) of a selected branch (definition comes later)
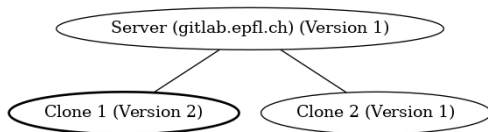
- ▶ To know the status of the working copy:

```
git status
```

- ▶ See the log

```
git log
```

GIT - Commit your modifications

```
git commit -m "I made an interesting modification" file.cc
```

GIT - Branches

- Branching means you diverge from the main line of development and continue without perturbing the code

- Branches can evolve independently

- The main branch in GIT is *usually* called *master*

- GIT doc on branches

GIT - Branches

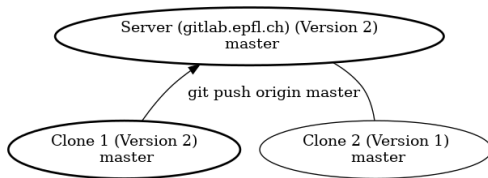- See/create branches:

  ```
  git branch
  ```

- Change the working copy to another branch.

  ```
  git checkout stable-branch
  ```
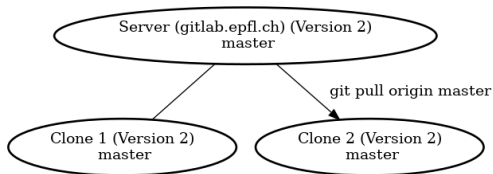
GIT - Push your modifications

```
git push origin master
```

This operation sends the current branch and merges it into the remote branch
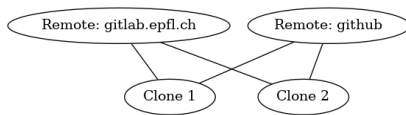
GIT - Pull modifications

```
git pull origin master
```


This operation actually fetches the remote branch and merges into current branch

GIT - remotes

- ▶ You can pull/push from/to more than a single distant server (remote)

- ▶ list the declared remotes:

```
git remote -v
```

- ▶ add/remove remotes

```
git remote add/remove
```

GIT - commands

```
git log
```

```
git checkout
```

```
git add file.cc
```

```
git rm file.cc
```

```
git mv file.cc
```

```
git commit -m nice message" file.cc
```

```
git push remote branch_name
git push origin master
```

```
git pull remote branch_name
git pull origin master
```

```
git diff
git diff revision_hash
```

```
git help whatever_command
```

GIT - resources

- Cheat Sheet: http://ndpsoftware.com/git-cheatsheet.html
- Simple guide: http://rogerdudler.github.io/git-guide/
- Nice tutorial: http://learngitbranching.js.org/

c4science.ch and gitlab.epfl.ch

What are c4science and gitlab.epfl.ch ?

C4Science and gitlab.epfl.ch are co-creation, curation and code sharing platforms. They include:

- ▶ Version management system
- ▶ Common authentication to all Swiss universities to local + external collaborators
- ▶ Social dimension (wikis, bug tracking, ...)
- ▶ Code test system (continuous integration)
- ▶ Swiss alternative to github and gitlab.com

Connect to c4science/gitlab.epfl.ch

The recommended way to connect to these servers server (and actually any distant linux machine) is through the SSH protocol:
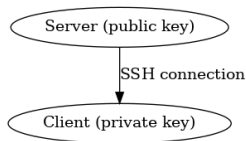
- ▶ You need a pair of keys: one public and one private
- ▶ They are stored in the directory *.ssh* in your home directory
- ▶ The public can be distributed, the private should stay **secret**
- ▶ A good habit is to generate one key-pair per client and never transport the private key