# Basics:

1. Version Control System (VCS):

Help us getting rid of “final\_no\_more\_edits\_or\_I’ll\_chop\_off\_my\_hand.doc”

1. Distributed Version Control System (DVCS):
2. Local or centralized VCS
3. Benefits:

* Flexible, support remote collaboration, different uncentralized workflow
* Robust against server malfunction (retrieve cloned user repository)
* Less dependent on Network/VPN (access cloned repository offline)
* Other systems: Mercurial, Bazaar or Darcs. Not sure whey Git is so popular. Maybe because it is free?

# Configuration and Authentication:

1. Configuration
2. Authentication:
3. Git on the server:

Create a “intermediate repository” that all collaborators have access to

1. Server protocols:

* Local: not good with multi-site work
* SSH and HTTP: use to be equivalent, until HTTP started asking for personal token
* Git: Not as secure as the previous ones

Books/tutorials tend not to talk about this. But it is usually the first problem Git users run into when they try to work remotely.

1. SSH keys: stored in ~/.ssh
2. Generate new key: remember the passphrase
3. If Git asks for passphrase every time you pull or push:

Open SSH agent: eval $(ssh-agent)

Add personal SSH key: ssh-add

# Collaboration:

Pull = fetch+merge

# View commit history:

* git log, with a lot of options
* But mostly can be done on GitHub, so omitted

# Undo changes:

* Add to last commit
* Unstage changes: restore
* Remove from working tree: reverse changes on local hard drive

Git checkout: Any local changes you made to that file are gone

# Branches

1. Create branches
2. Merge branches and conflicts

* Happens 99% of the time
* Need to solve by hand