Versioning with git

Day 2

Severin Reiz

20.03.2019

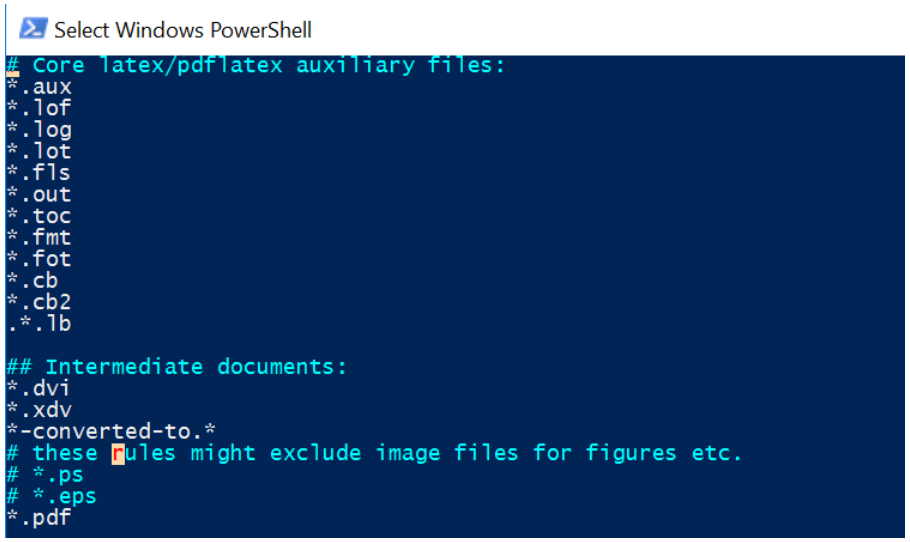
Git/Bash/Python-Course for MPE

Agenda

- Overview of yesterday:
  - The general concept of git: version control
  - git checkout, git branch
  - git add, git commit, git push
  - Remote repositories
- The most important commands
  - git fetch, git pull
  - git reset
  - git checkout, git merge, git diff
- Some references

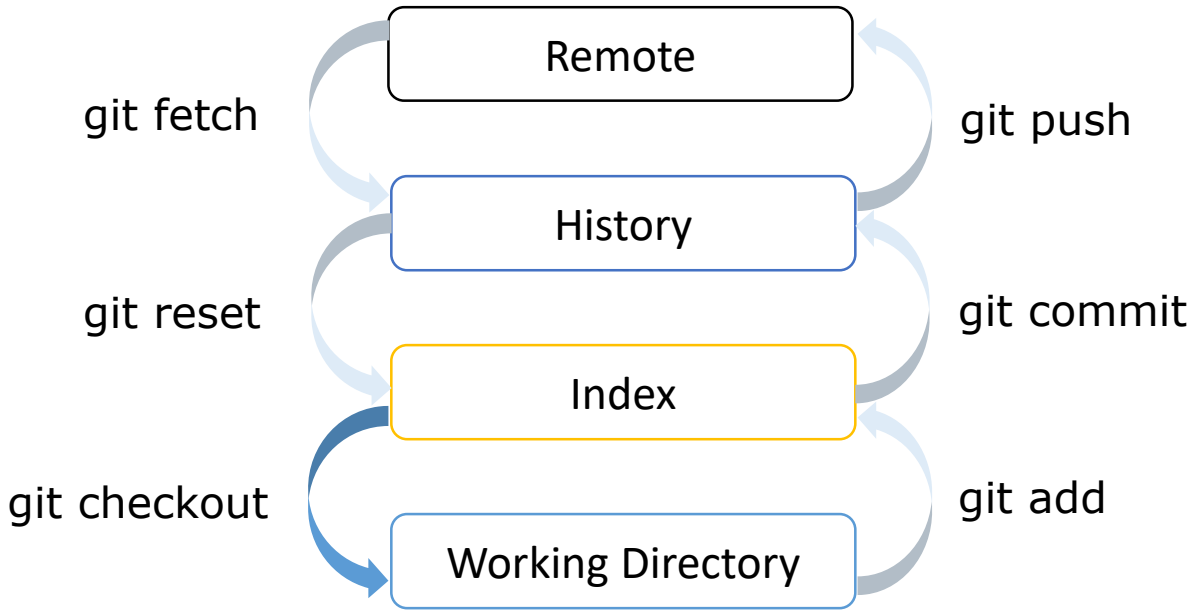
Ignoring Stuff: .gitignore

- List of files and paths to be ignored by git.
- Accepts ”*” as wildcard.
- This file should be uploaded as any other.
- Useful for output or IDE files, data directories, etc..
- Do NOT put data, word/ppt, (or pdf?) in your git repository!



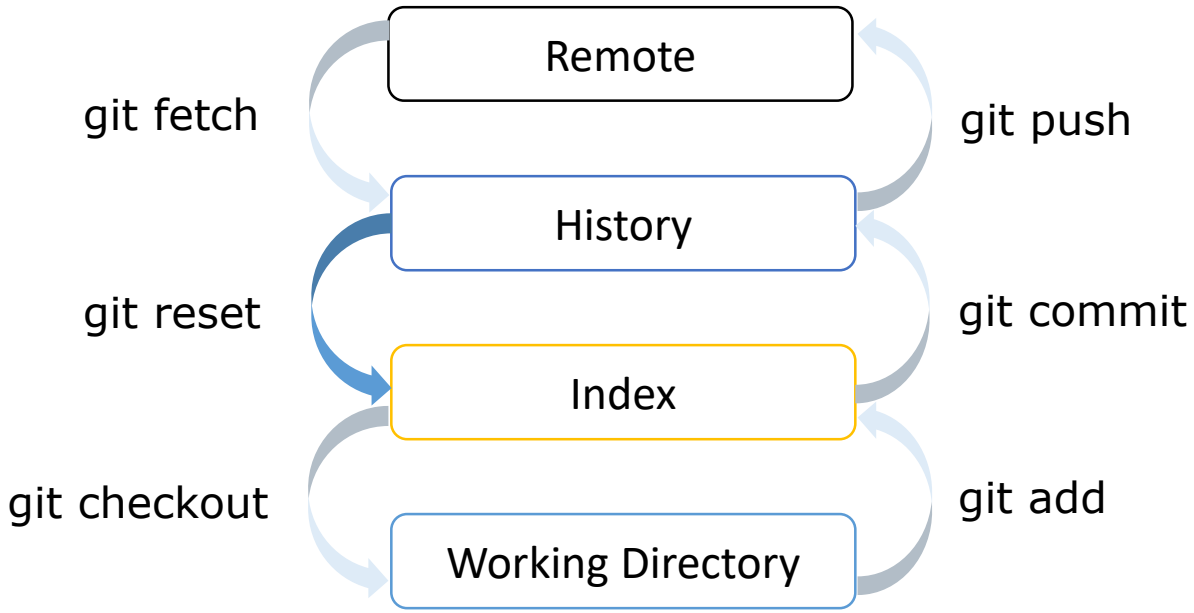
The most important commands:
git checkout

- git checkout copies files from the index or a commit to the wd
- git checkout <files> updates files in the wd from the stage
- Can also be used to switch to/create a branch



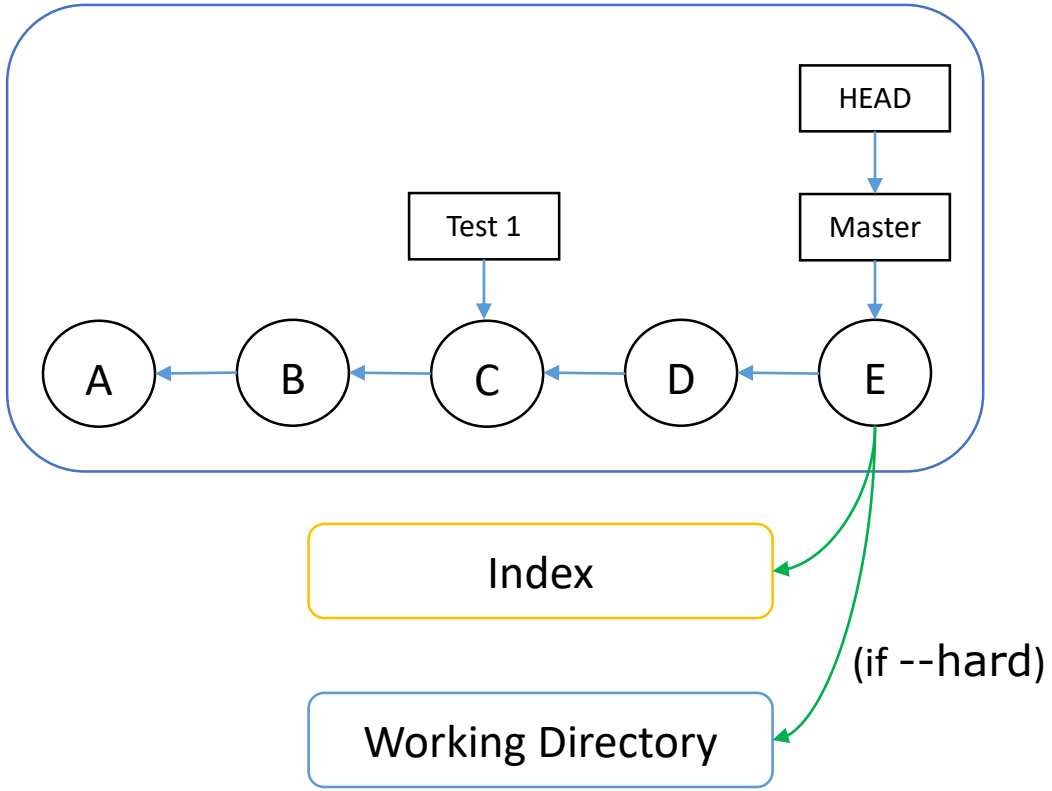
The most important commands:
git reset

- git reset <commit> <files> resets <files> in index from <commit>
- git reset <commit> resets current branch to <commit>
- Working directory can also be updated



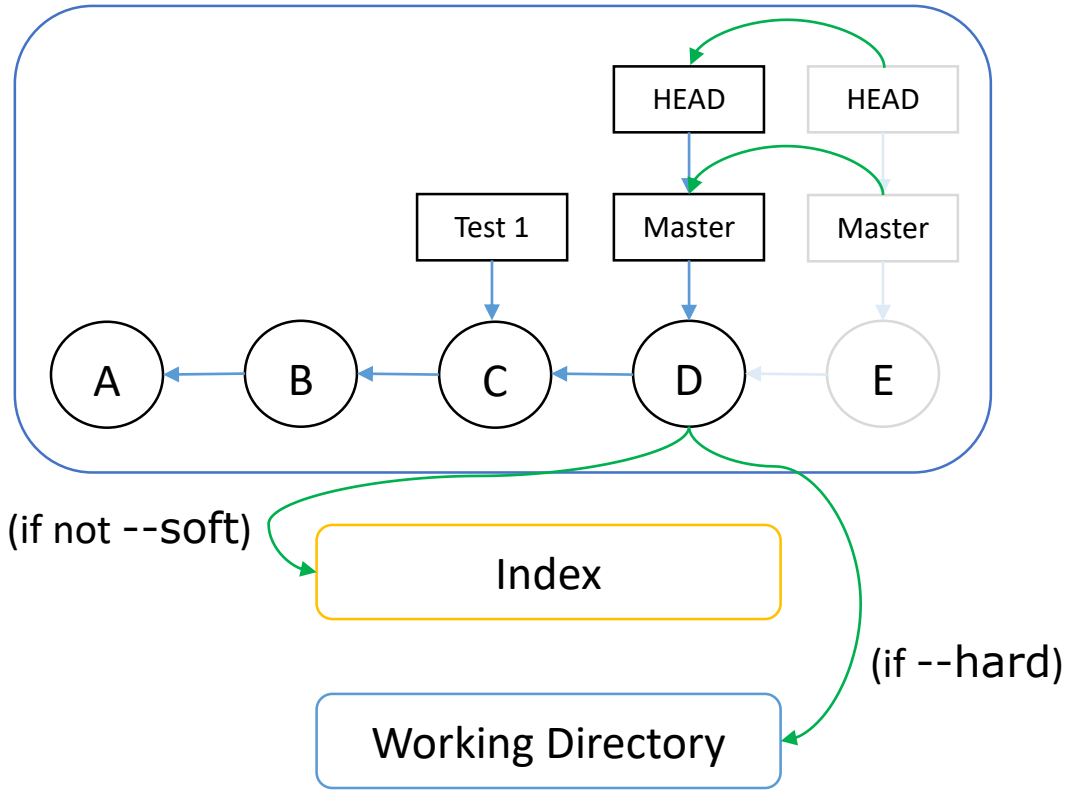
The most important commands:
git reset

- git reset <commit> <files> [--hard] copies from commit to index



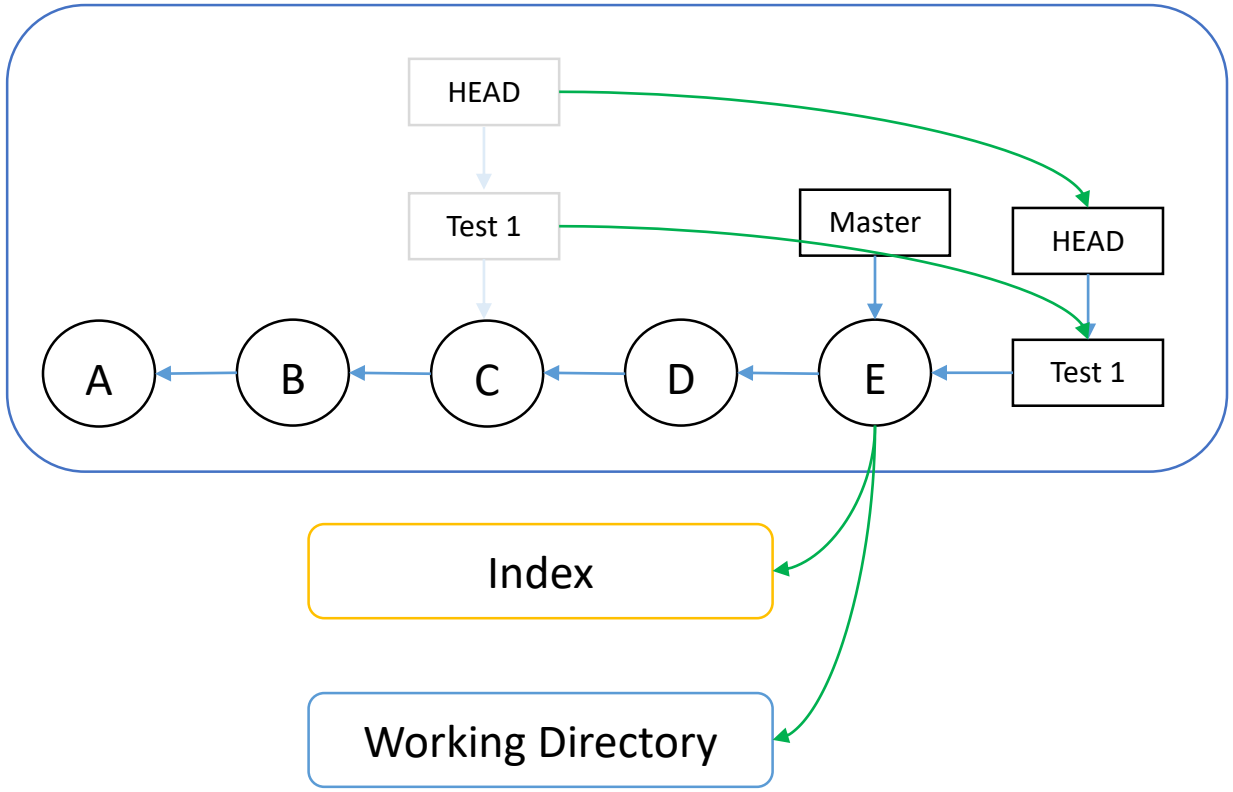
The most important commands:
git reset

- git reset [--hard/--soft] <commit> sets the branch to previous commit
- If not reachable anymore following commits are deleted



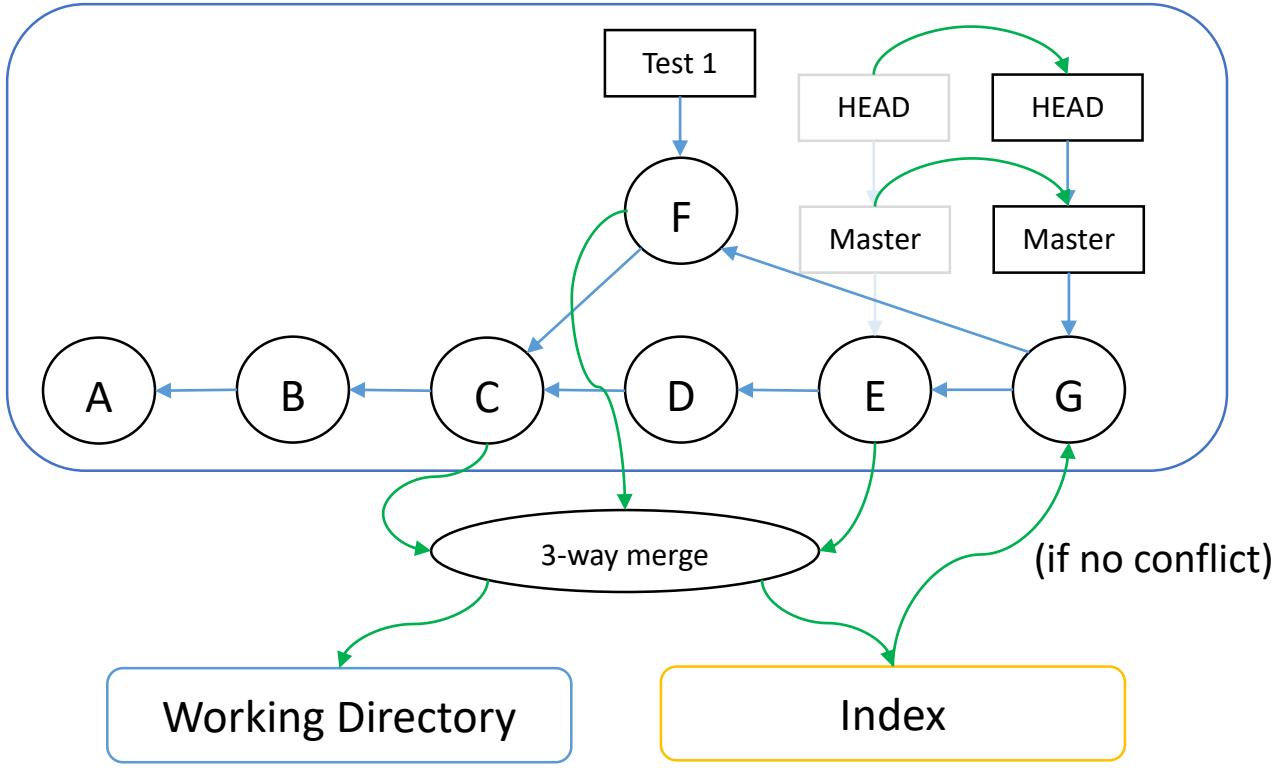
The most important commands:
git merge

- git merge <branch> merges the current branch with <branch>
- The easiest case is if HEAD is an ancestor of <branch>
- This case is called a fast-forward merge



The most important commands:
git merge

- git merge <branch> merges the current branch with <branch>
- Merging two arbitrary branches is a bit more involved



The most important commands: git merge

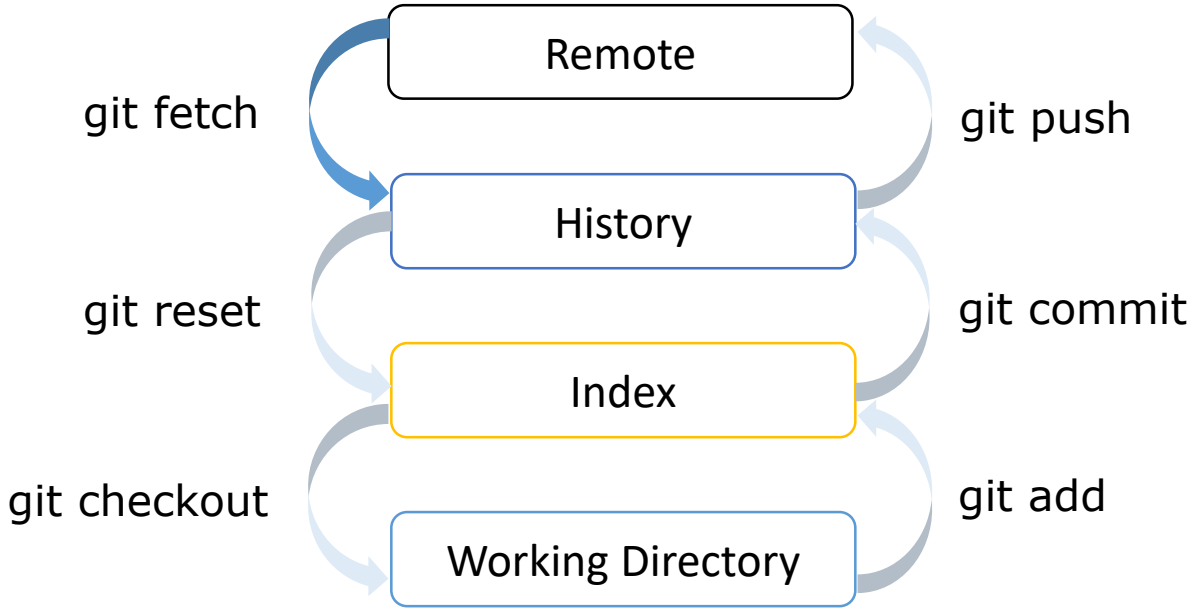
- A conflict occurs if git can not decide automatically what to keep
  - Multiple overlapping modifications of a file
- Occurences are marked in the respective files
- Format:

```
<<<<<<<<<<<<
<Own version>
============
<Other version>
>>>>>>>>>>>>
```

- `git mergetool` can be used to resolve conflicts
- GUIs normally also have a merge tool
- When all conflicts are resolved, the result can be committed

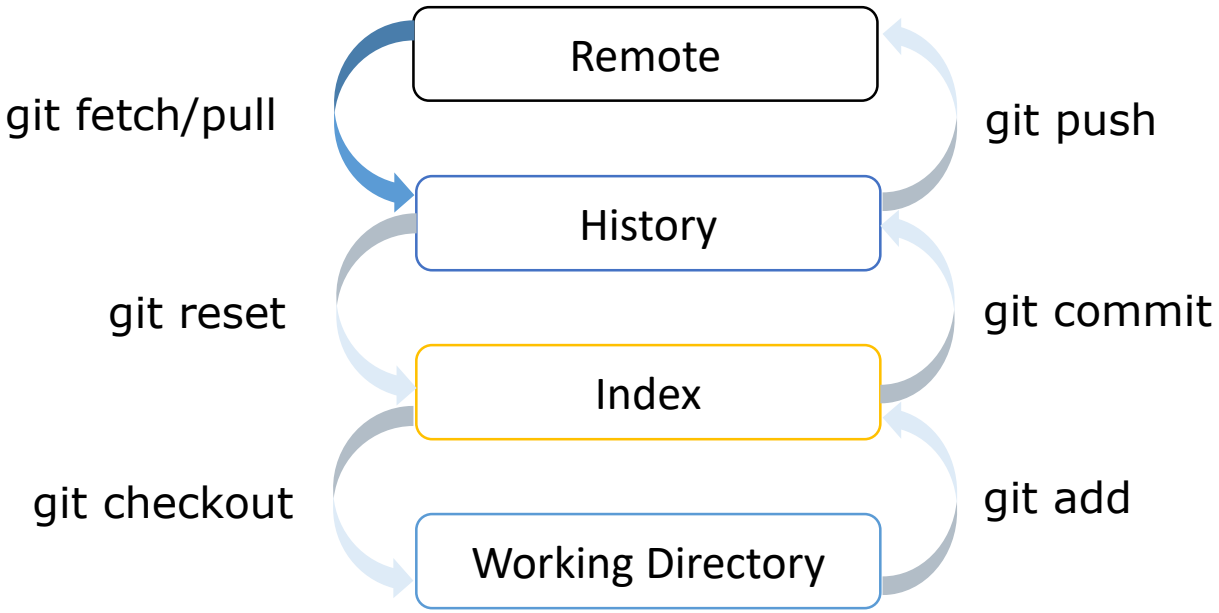
The most important commands:
git fetch

- git fetch <remote> updates all the locally stored remote branches
- Stores the required data and updates references
- git merge can be used to merge remote branches with local ones
- Remote branches are named like <remote>/<branch>



The most important commands:
git pull

- git pull <remote> is simply git fetch + git merge
- Convenient shortcut for standard updates
- Costs some flexibility as user has no way to examine merged content



The most important commands

- This was only a brief overview, many more things exist
  - Cherry picking
  - Rebasing
  - …
- Some more useful commands:
  - `git diff` to show differences between working tree and commits etc.
  - `git status` to show the working tree status
  - `git log [--graph]` to show history of the repository
  - `git show` to have a look at individual objects
- Git also has a very nice reference and lots of tutorials (see literature)