Software Engineering: Tutorial 1

David Voigt

October 28th, 2022

Organisational matters

General information

- The tutorial starts at c.t., that is, at 12:15 o'clock
- Attendance is not mandatory
- Please bring your laptop if possible
- Relevant material presented during the tutorial will be uploaded to Github

Format of the tutorial

- Discuss common mistakes in your assignments
- Provide additional useful knowledge with respect to the lecture, e.g., basic terminal usage
- Prepare you for your assignments
- Answer your questions regarding the assignments
  - For general questions regarding the lecture, please ask directly in the forum

Basic terminal usage

Poll

1. Who of you has previous experience working with the command line?
2. Who is regularly using the command line during their normal workflow?

For advanced users

1. Create a function `mkcd` that is accessible from the terminal.
   - `mkcd [FOLDER]` : creates a new folder and navigates to it using `cd`
2. Make sense of the following command and decide what the output might be:
   - `cat file.txt | sort  | uniq > out.txt`

Useful commands

command	usage
`ls`	show files in current directory
`cd [DIR]`	changes the current working directory
`pwd`	prints the current working directory
`mv [FILE]* [FILE]`	moves one or more file(s) to another directory
`cp [FILE]* [File]`	copies one or more file(s) to another directory
`touch [FILE]`	creates a new file
`mkdir [-p] [FILE]`	creates a new directory

Useful commands

command	usage
`cat [FILE]`	outputs the content of a given file
`path/to/executable`	executes an executable
`[CMD1] | [CMD1]`	redirects the output of `CMD1` to `CMD2` as input
`[CMD] > out.txt`	writes the output of `CMD` to the file `out.txt`
`[CMD] >> out.txt`	appends the output of `CMD` to the file `out.txt`
`man CMD`	shows the manual/documentation for a given command

Questions

Are there any questions?

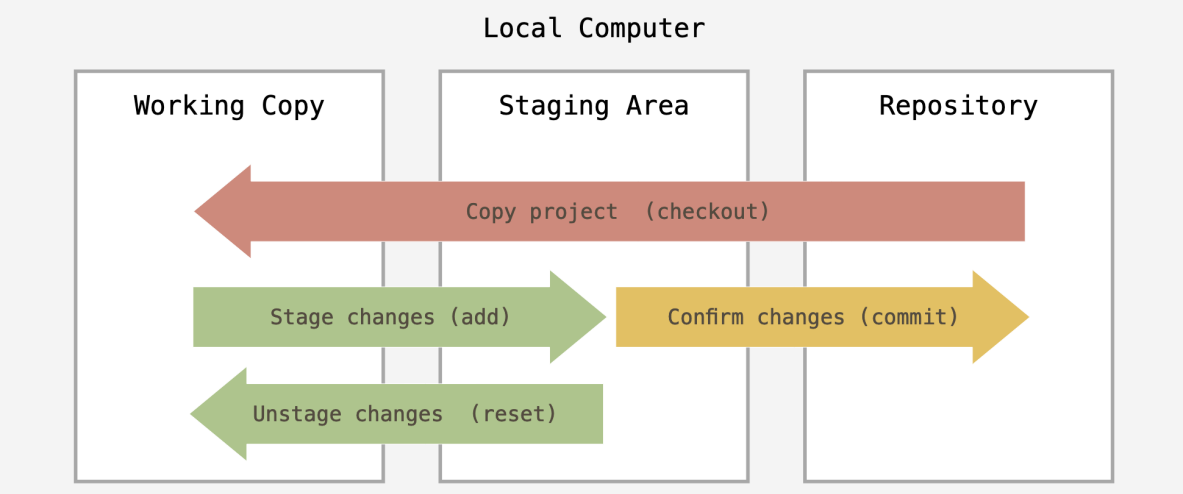
git recap

For advanced users

1. Play around with `git bisect`: here's an example
2. Make sense of `git reflog`
3. Figure out the use of `git stash`

Summary

- `git` is the de-facto standard VCS
- snapshot based
- decentralized



Working Copy vs. Staging Area vs. Repository

Working Copy The **working copy** the project folder that is currently under git version control. The working copy consists of "normal" files outside the `.git`, that can be altered.

Staging Area The **staging area** is like a drafting area. It is also called **index** and contains snapshots of files to be commited.

Repository The **repository** is represented by the `.git` folder. The repository **contains the whole history** of the project, e.g., commits and file snapshots. For example, this is also what is stored on Github.

Common commands

- `git help` *cmd*
- `git add` *file*
- `git checkout` *file*
- `git init`
- `git reset` *file*
- `git commit`
- `git status` *folder*
- `git log`
- `git diff`
- `git show` *commit*

Other useful commands and options

- `git add -p`
  - interactively add files
- `git add -u`
  - only re-add files the index that already have been added previously
- `git mv` *file*
  - move the working copy of a file and reflect the change in the index
- `git rm` *file*
  - remove the working copy of a file and reflect the change in the index

Questions?

Are there any questions?

Useful links

- git visualizer: https://git-school.github.io/visualizing-git/
- git branching tutorial: https://learngitbranching.js.org/
- git cheat sheet: https://training.github.com/downloads/github-git-cheat-sheet.pdf