Introduction to git - Part A

Installation & Setup

Contents

1 Goal

This lab is divided into two parts. Part A must be done at home as preparation, while Part B is done together in the lab. The lab will be done independently on your laptop and graded at the end.
In this lab we will learn the basic principles of version control `git` [1], especially the tools Git command line and Sublime Merge, which need to be installed and configured on your machine (see Section 2). Furthermore, accounts are created on the platforms Github and Hevs Gitlab [2].
Finally, we will learn the basics of Markdown in Section 3 to easily write text files.

It is crucial that the installation and configuration is done carefully to avoid wasting time in the lab.

2 Installation

The first step is to install Git as well as Sublimemerge. You can choose whether you want to use the command line or the GUI during the lab. However, both tools should be installed and configured.

2.1 `git`

You can download the latest version from the official website https://git-scm.com/ [1]. Git is available for Linux, Mac, and Windows. This lab requires git ≥ 2.27.

2.1.1 Command line

Start "Git Bash" on Windows or "Terminal" on MacOS. This is a Unix/Linux-like command editor that allows you to run Git commands in console mode.

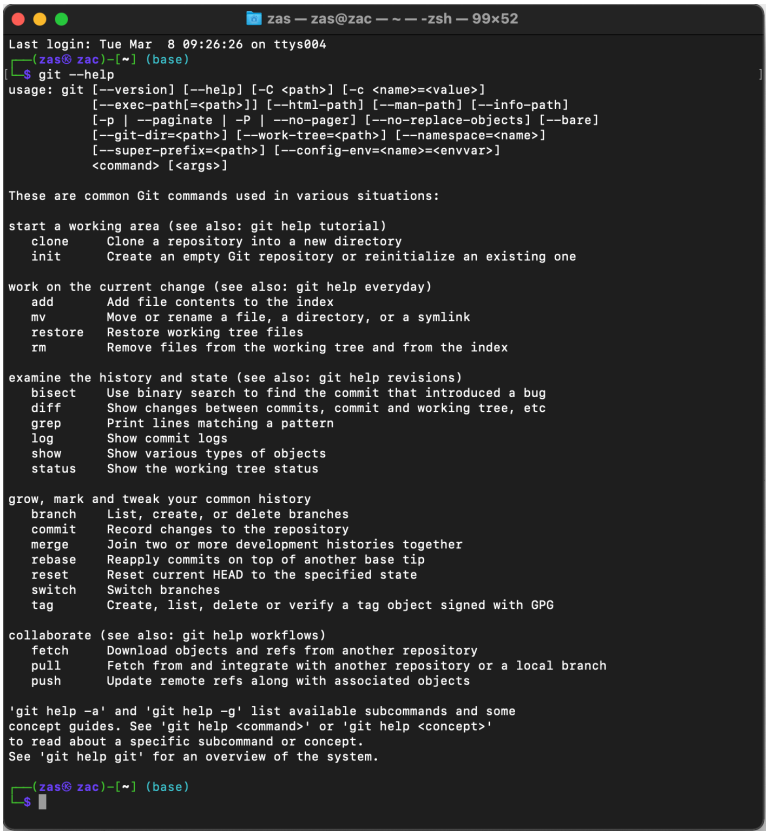

Figure 1: git Terminal

Note that for all commands in Git Bash, you can get help by inserting `--help` after the command.

```
git --help
```

2.1.2 Global configuration

A variety of settings can be configured in Git. It is possible to change the settings globally on your computer (flag `--global`) or only for a specific repository.
We will now perform the minimal configuration. Use the following commands to set your identity in Git globally on the system. Use your name and email address. This information is publicly visible to identify your work (your commits).

```
git config --global user.name "Firstname Lastname"
git config --global user.email first.last@email.ch
```

For example:

```
git config --global user.name "Silvan Zahno"
git config --global user.email silvan.zahno@hevs.ch
```

You can check the configuration with the following command:

```
git config --list
```

You can also check a specific setting:

```
git config user.name
```

2.2 Sublime Merge

Visit the website https://www.sublimemerge.com and download and install the Sublime Merge tool [3].

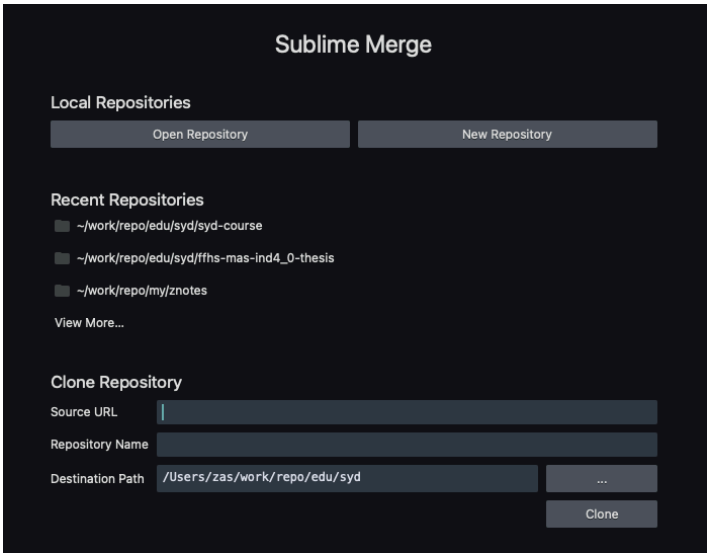

Figure 2: Sublime Merge GUI

2.3 Online accounts

2.3.1 Gitlab

Visit the website https://gitlab.hevs.ch and log in with your school account (SwitchEDU-ID).

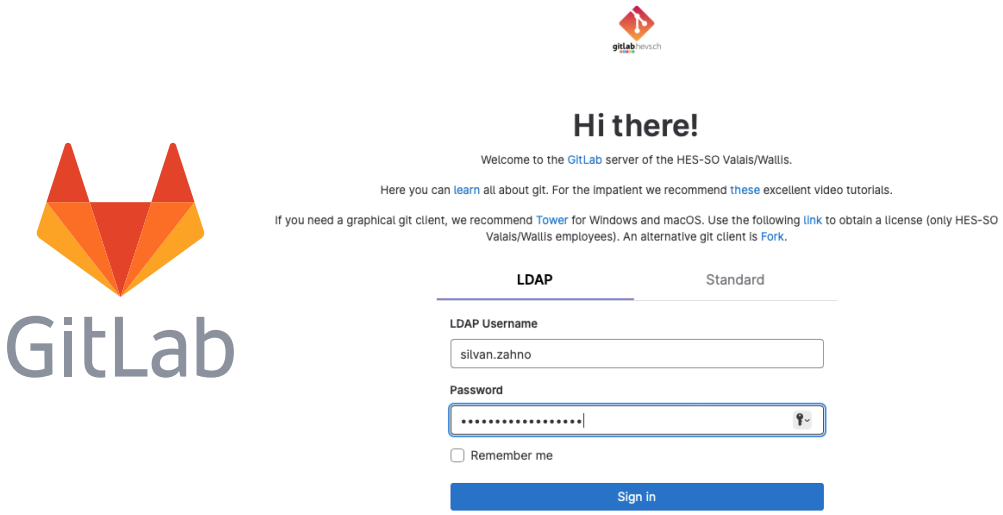

Figure 3: Gitlab Login

2.3.2 Github

Visit the website https://github.com and create an account and log in.

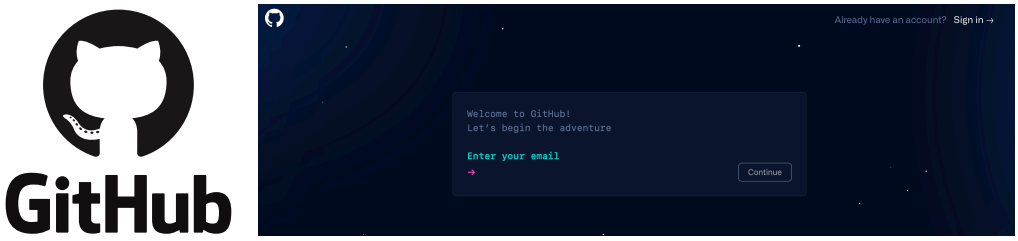

Figure 4: GitHub Login

2.4 Windows Configuration

In order to see also the hidden `.git/` folder as well as file extentions. Configure your Windows File Explorer as follows Figure 5:
File Explorer ⇨ View ⇨ Show ⇨ Activate "**File name extensions**" and "**Hidden items**"

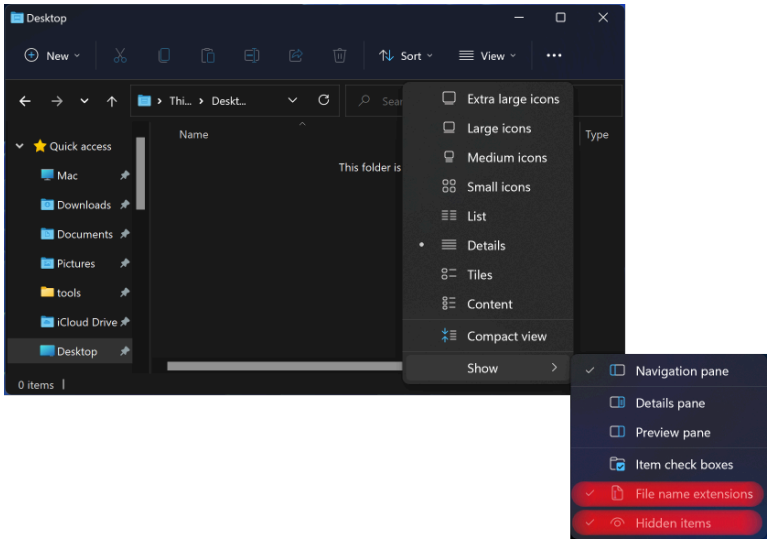

Figure 5: Windows File Explorer Configuration

3 Markdown

Markdown is a lightweight markup language with plain-text formatting syntax. It is designed to be easy to read and write, while also being easily converted into PDF, HTML or other formats. Markdown is commonly used for formatting text on the web, such as in `README.md` files, documentation, forum posts, and messaging.

In order to write Markdown, you need your preferred Text editor or you can install a spezialised Markdown Editor such as Marktext.

For example the intropage of this course is written in Markdown, you can see the source code of the page by clicking on the "Edit this page" button on the top right corner of the page or via the link.

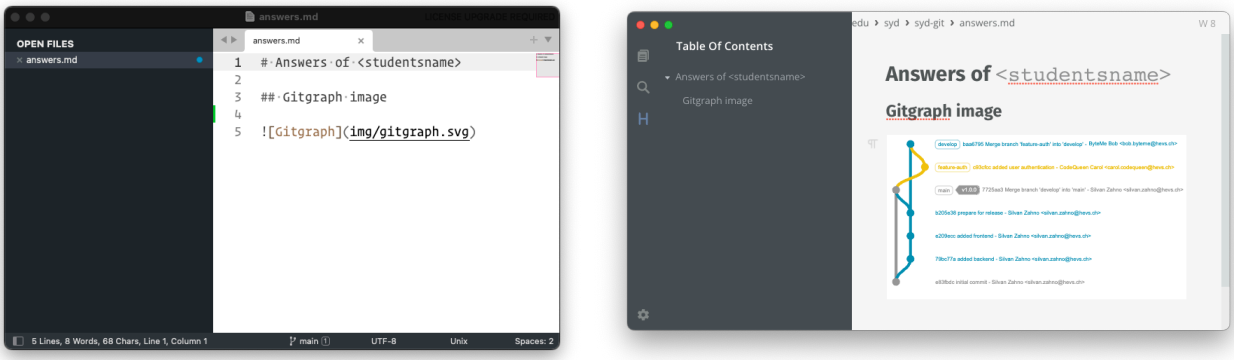

Figure 6: Left: Simple Texteditor (Sublime Text), Right: Marktext

For the lab, you will need to write a documents in Markdown format. Be ready with your editor.

3.1 Markdown syntax

Hereafter a short overview about how a markdown file is structured. The syntax is simple and easy to learn. The file has to be saved with the extension `.md`. A more complete syntax list can be found at here.

```
# Title 1

## Title 2

### Title 3

Some simple Text _italic_ **bold**
~~Strikethough~~ `monospaced`

Fomulas $S = \sum_{i=1}^{n} x_{i}^{2}$

- List Item 1
- List Item 2
1. Numbered List Item 1
2. Numbered List Item 2

[Link name](https://hevs.ch/synd)

![logo](logo.svg)

```rust
// A python code bloc
fn main(){
  println!("Hello World");
}
```

Tables	Are	Cool
col 1	left-align	f_{clk}
col 2	centered	$12
col 3	right-align	1024

```

4 Outro

Congratulations You have now installed and configured everything you need to work with Git. Prepare for the next lab:

- Study the theory
- Familiarize yourself with the Git commands
- Familiarize yourself with the graphical tool Sublimemerge
- Practice writing documents with Markdown

See the appendix Section A and Section B for a summary of the most important Git commands.

A GIT commands

Github git cheatsheet [4], [5]

AA Review changes and make a commit transaction.

```
git status
```

Lists all new or changed files ready for commit.

```
git diff
```

Displays file changes that have not yet been indexed.

```
git add [file]
```

Indexes the current state of the file for versioning.

```
git diff --staged
```

Shows the differences between the index (“staging area”) and the current file version.

```
git reset [file]
```

Takes the file from the index, but preserves its contents.

```
git commit -m "[descriptive message]"
```

Adds all currently indexed files permanently to the version history.

AB Synchronize changes

Register an external repository (URL) and swap the repository history.

```
git fetch [remote]
```

Downloads the entire history of an external repository.

```
git merge [remote]/[branch]
```

Integrates the external branch with the current locally checked out branch.

```
git push [remote] [branch]
```

Pushes all commits on the local branch to GitHub.

```
git pull
```

Pulls the history from the external repository and integrates the changes.

B Most used Git commands

BA Start a working area

- `clone` - Clone a repository into a new directory
- `init` - Create an empty Git repository or reinitialize an existing one

BB Work on the current change

- `add` - Add file contents to the index
- `mv` - Move or rename a file, a directory, or a symlink
- `reset` - Reset current HEAD to the specified state
- `rm` - Remove files from the working tree and from the index

BC Examine the history and state

- `log` - Show commit logs
- `show` - Show various types of objects
- `status` - Show the working tree status

BD Grow, mark and tweak your common history

- `branch` - List, create, or delete branches
- `checkout` - Switch branches or restore working tree files
- `commit` - Record changes to the repository
- `diff` - Show changes between commits, commit and working tree, etc
- `merge` - Join two or more development histories together
- `rebase` - Reapply commits on top of another base tip
- `tag` - Create, list, delete or verify a tag object signed with GPG

BE Collaborate

- `fetch` - Download objects and refs from another repository
- `pull` - Fetch from and integrate with another repository or a local branch
- `push` - Update remote refs along with associated objects

Bibliography

[1] T. Linus, “Git.” Accessed: Apr. 25, 2023. [Online]. Available: https://git-scm.com/

[2] “GitLab Hevs.” Accessed: Apr. 25, 2023. [Online]. Available: https://gitlab.hevs.ch/

[3] “Sublime Merge - Git Client from the Makers of Sublime Text.” Accessed: Apr. 25, 2023. [Online]. Available: https://www.sublimemerge.com/

[4] gitlab, “Git Cheatsheet.” 2023.

[5] “GitHub Git Spickzettel.” Accessed: Apr. 25, 2023. [Online]. Available: https://training.github.com/downloads/de/github-git-cheat-sheet/