



Introduction to git - Part A

Installation & Setup



Contents

1	Goal	2
2	Installation	3
2.1	Sublime Merge	3
2.2	git - command line	5
2.3	Online accounts	6
2.4	Windows Configuration	6
3	Markdown	7
3.1	Markdown syntax	8
4	Outro	9
	Bibliography	10
5	Appendix	11
A	GIT commands	11
AA	Review changes and make a commit transaction.	11
AB	Synchronize changes	11
B	Most used Git commands	13
BA	Start a working area	13
BB	Work on the current change	13
BC	Examine the history and state	13
BD	Grow, mark and tweak your common history	13
BE	Collaborate	13



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], in particular the tool [Sublime Merge](#) and optionally the [Git command line tool](#), which have to be installed and configured on your computer (see Section 2). In addition, an account is created on the [Github](#) platform. 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 during part B of the laboratory.



2 | Installation

The first step is to install Git and/or Sublime Merge. You can choose whether you want to use the command line or the GUI during the lab.

2.1 Sublime Merge

Visit the website <https://www.sublimemerge.com> and download and install the Sublime Merge tool [2].



Figure 1 - Sublime Merge GUI

2.1.1 Configuration

When cloning a repository, Sublime Merge will automatically ask for your identity and prompt you to log in to your Github account.



2.1.2 Overview

The interface of Sublime Merge is presented in the Figure 2:

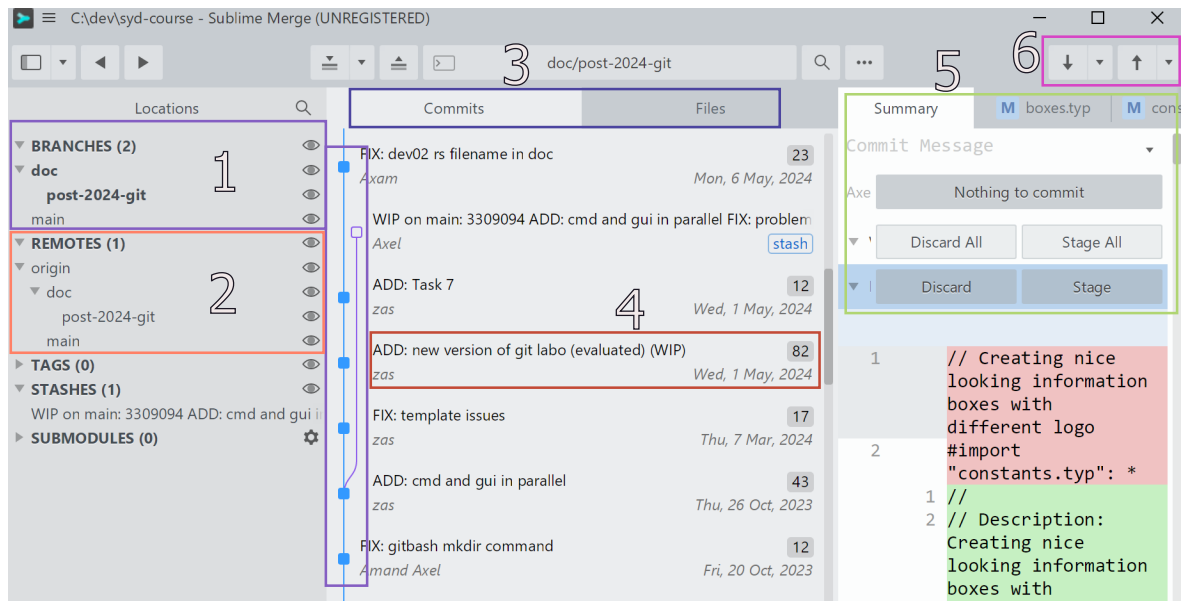


Figure 2 - Sublime Merge GUI

1. **Branches:** list of branches of the repository. You can create, delete and rename branches. They are also displayed in their timeline under the **Commits** tab (point 3).
2. **Remotes:** list of remote repositories, i.e. the servers on which the code is stored and to which you can **push**.
3. **Commits/Files:** tabs to switch between the chronological view of commits and the view of changes for the selected commit.
4. **Commit description:** a commit consists of a message, an author and a date.
5. **Current changes:** the files that have been modified compared to the last commit are listed here. You can select the files you want to **commit** by selecting them and clicking on the **Stage** button. As long as the files have not been **committed**, you can also remove a file from staging by clicking on the **Unstage** button or completely delete a file's changes by pressing **Discard** (⚠ permanent deletion ⚠).
6. **Pull / Push:** the two buttons allow you to **pull** - take the latest changes from the remote repository - and **push** - send local changes to the remote repository. It is also possible to **fetch** by clicking on the small arrow next to the **Pull** button, which allows you to see the new commits without modifying the local repository.



2.2 git - command line

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 \geq 2.27.

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.

```

Last login: Tue Mar  8 09:26:26 on tty006
[~] zsh zsc@zsc ~ -- ssh -- 90x52
[~] $ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path<path>] [--html-path] [--man-path] [--info-path]
      [-p | --paginate] [-P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir<path>] [--work-tree<path>] [--namespace<name>]
      [--super-prefix<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common git commands used in various situations:

start a working area (see also: git help tutorial)
  clone Clone a repository into a new directory
  init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add Add file contents to the index
  mv Move or rename a file, a directory, or a symlink
  restore Restore working tree files
  rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect Use binary search to find the commit that introduced a bug
  diff Show changes between commits, commit and working tree, etc
  grep Print lines matching a pattern
  log Show commit logs
  show Show various types of objects
  status Show the working tree status

grow, mark and tweak your common history
  branch List, create, or delete branches
  commit Record changes to the repository
  merge Join two or more development histories together
  rebase Reapply commits on top of another base tip
  reset Reset current HEAD to the specified state
  switch Switch branches
  tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  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

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

[~] $
  
```

Figure 3 - git Terminal



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

```
git --help
```

2.2.1 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
```

2.3 Online accounts

2.3.1 Github

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

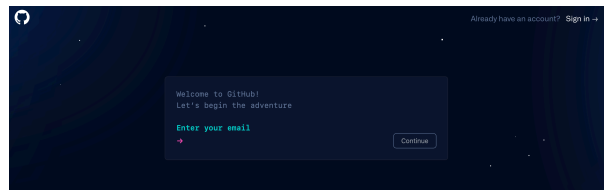


Figure 4 - GitHub Login

2.4 Windows Configuration

In order to see also the hidden `.git/` folder as well as file extensions. Configure your Windows File Explorer as follows Figure 5:

File Explorer ⇒ View ⇒ Show ⇒ Activate “**File name extensions**” and “**Hidden items**”

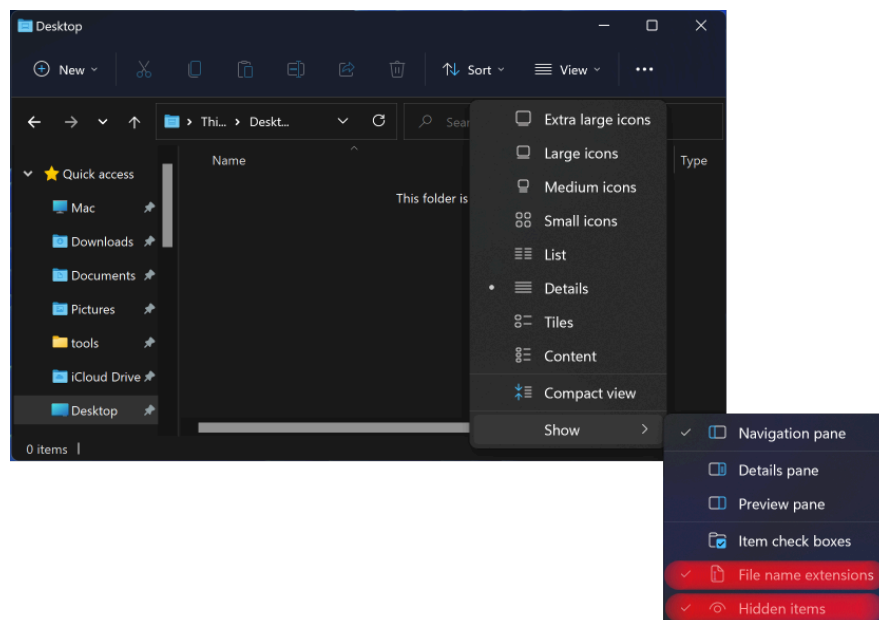


Figure 5 - Windows File Explorer Configuration



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 specialised 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](#).

```

1 # Answers of <students-firstname> <students-lastname> <github-username>
2
3 ## Basics
4
5 ### Task 1
6
7 ### Task 2
8
9 ### Task 3
10
11 ### Task 4
12
13 ### Task 5
14
15 ### Task 6
16
17 ## Gitgraph
18
19 ### Task 7
20
21 ![[Gitgraph]](img/gitgraph.svg)
  
```

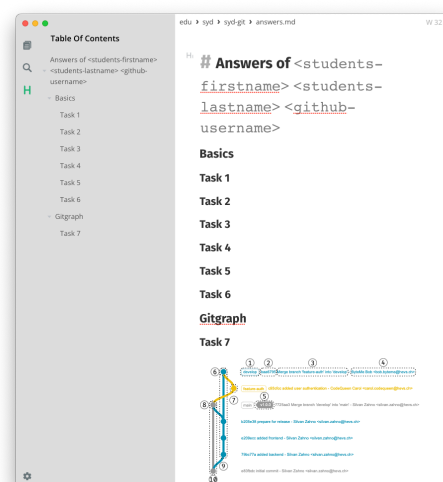


Table 1 - Left: Simple Texteditor (Sublime Text), Right: Marktext



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 https://wiki.zahno.dev/multimedia/writing/md/md_github.html.

```
# Title 1

## Title 2

### Title 3

Some simple Text italic bold
Strikethrough `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 rust 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
```

urces › git › markdown-cheatsheet.md W 90

Title 1

Title 2

Title 3


Some simple Text *italic* **bold** ~~Strikethrough~~ `monospaced`

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

- List Item 1
- List Item 2

- Numbered List Item 1
- Numbered List Item 2

[Link name](#)



```
// A rust 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. You should have:

- ☐ Installed Git and Sublime Merge
- ☐ Configured git with your name and email
- ☐ Created a GitHub account
- ☐ Familiarized yourself with the Sublime Merge graphical user interface (GUI)
- ☐ Familiarized yourself with the Git theory and commands
- ☐ Familiarized yourself with Markdown and its syntax



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



Bibliography

- [1] T. Linus, “Git.” Accessed: Apr. 25, 2023. [Online]. Available: <https://git-scm.com/>
- [2] “Sublime Merge - Git Client from the Makers of Sublime Text.” Accessed: Apr. 25, 2023. [Online]. Available: <https://www.sublimemerge.com/>
- [3] gitlab, “Git Cheatsheet.” 2023.
- [4] “GitHub Git Spickzettel.” Accessed: Apr. 25, 2023. [Online]. Available: <https://training.github.com/downloads/de/github-git-cheat-sheet/>



5 | Appendix

A | GIT commands

[Github git cheatsheet](#) [3], [4]

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