Einführung in Git - Teil A

Installation und Einrichtung

Contents

1 Ziel

Dieses Labor ist in zwei Teile unterteilt. Teil A muss als Vorbereitung zu Hause durchgeführt werden, während Teil B gemeinsam im Labor bearbeitet wird. Das Labor wird eigenständig auf Ihrem Laptop durchgeführt und am Ende bewertet.
In diesem Labor werden wir die Grundprinzipien der Versionskontrolle `git` kennenlernen [1], insbesondere die Tools Git-Komandozeile und Sublime Merge, die auf Ihrem Rechner installiert und konfiguriert werden müssen (siehe Abschnitt 2). Des Weiteren werden Konten auf den Plattformen Github und Hevs Gitlab [2] erstellt.
Schlussendlich lernen wir die Grundlagen von Markdown in Abschnitt 3, um einfach Textdateien zu schreiben.

Es ist entscheidend, dass die Installation und Konfiguration sorgfältig durchgeführt wird, um Zeitverlust während Teil B des Labors zu vermeiden.

2 Installation

Der erste Schritt ist die Installation von Git sowie Sublimemerge. Sie können selber auswählen ob sie die Kommandozeile oder das GUI benutzen möchten während des Labors. Es sollten aber beide Tools installiert und konfiguriert werden.

2.1 `git`

Du kannst die neueste Version über die offizielle Website https://git-scm.com/ [1] herunterladen. Git ist für Linux, Mac und Windows verfügbar. Für dieses Labor wird git ≥ 2.27 benötigt.

2.1.1 Kommandozeile

Starte "Git Bash" auf Windows der „Terminal" auf MacOS. Dies ist ein Unix/Linux-ähnlicher Befehlseditor, der es ermöglicht, Git-Befehle im Konsolenmodus auszuführen.

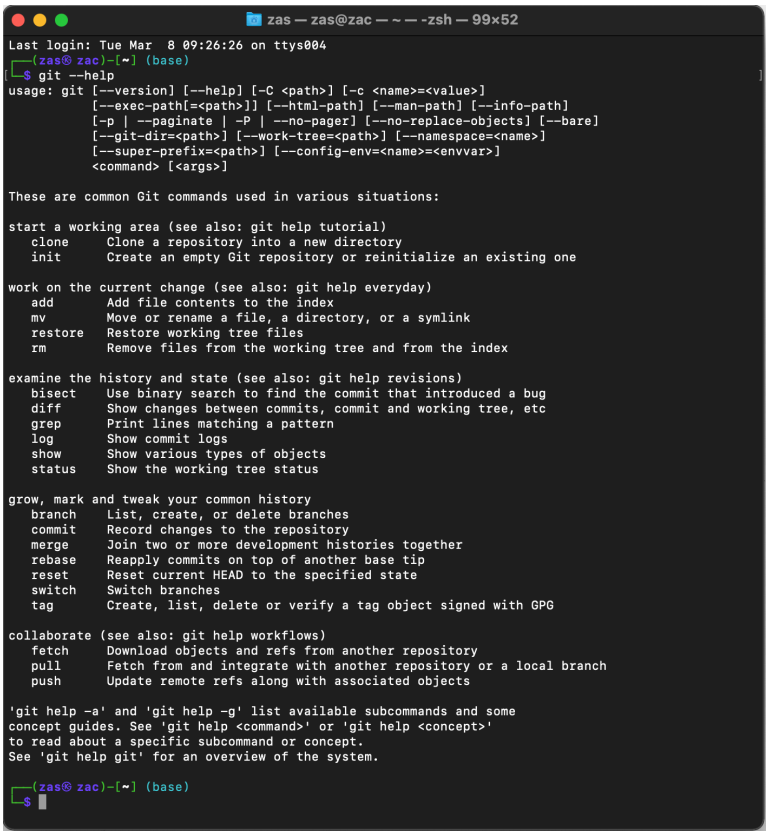

Abbildung 1: git Terminal

Beachten Sie, dass Sie für alle Befehle in Git Bash Hilfe erhalten können, indem Sie `--help` nach dem Befehl einfügen.

```
git --help
```

2.1.2 Globale Konfiguration

Eine Vielzahl von Einstellungen kann in Git konfiguriert werden. Es ist möglich, die Einstellungen global auf deinem Computer (Flag `--global`) oder nur für ein bestimmtes Repository zu ändern. Wir werden nun die Minimalkonfiguration durchführen. Verwende die folgenden Befehle, um deine Identität in Git `global` auf dem System einzustellen. Verwende deinen Namen und deine E-Mail-Adresse. Diese Informationen sind öffentlich sichtbar, um deine Arbeit (deine Commits) zu identifizieren.

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

Zum Beispiel:

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

Sie können die Konfiguration mit dem folgenden Befehl überprüfen:

```
git config --list
```

Sie können auch eine bestimmte Einstellung überprüfen:

```
git config user.name
```

2.2 Sublime Merge

Besuchen Sie die Webseite https://www.sublimemerge.com and laden sowie installieren Sie das Tool Sublime Merge herunter [3].

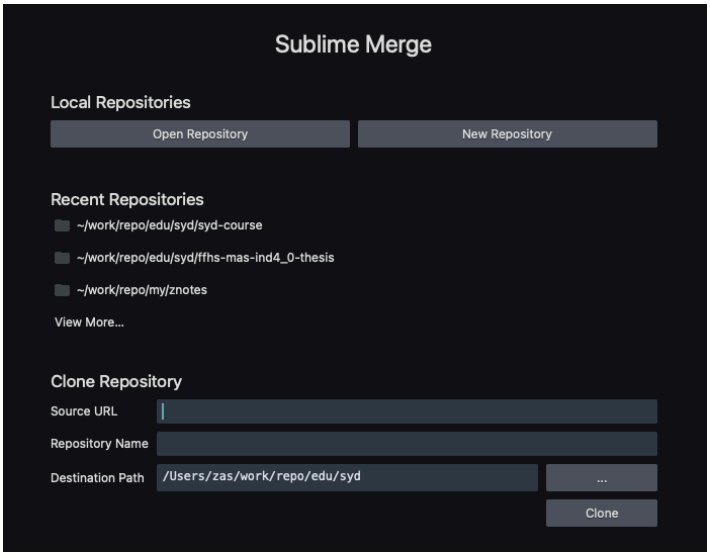

Abbildung 2: Sublime Merge GUI

2.3 Online-Konten

2.3.1 Gitlab

Besuchen Sie die Webseite https://gitlab.hevs.ch und loggen Sie sich mit Ihrem Schulkonto ein (SwitchEDU-ID).

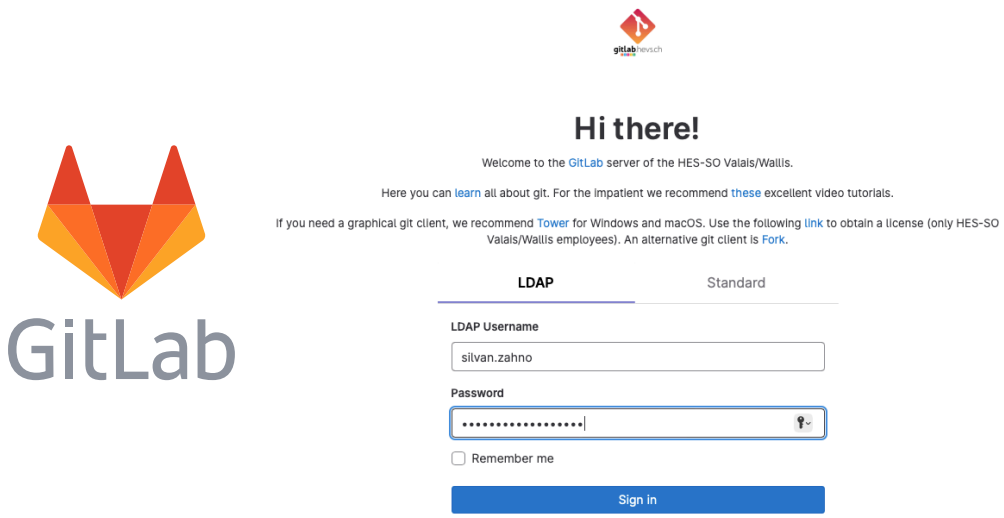

Abbildung 3: Gitlab Anmeldung

2.3.2 Github

Besuchen Sie die Webseite https://github.com und erstellen Sie ein Konto und loggen Sie sich ein.

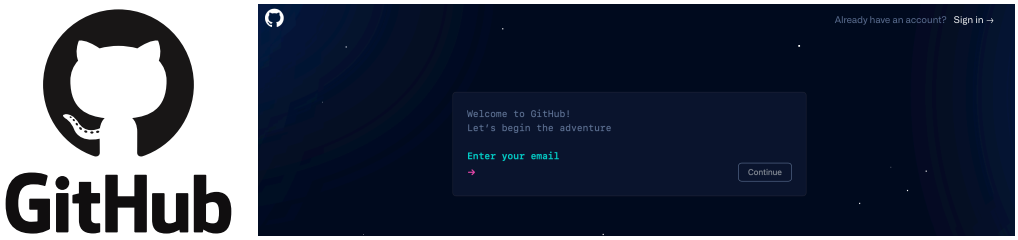

Abbildung 4: GitHub Anmeldung

2.4 Windows-Konfiguration

Um auch den versteckten `.git/`-Ordner sowie die Dateierweiterungen sehen zu können. Konfigurieren Sie Ihren Windows Datei Explorer wie folgt:
Datei-Explorer ⇨ Ansicht ⇨ Anzeigen ⇨ Aktivieren Sie „**Dateinamenerweiterungen**“ und „**Versteckte Elemente**“

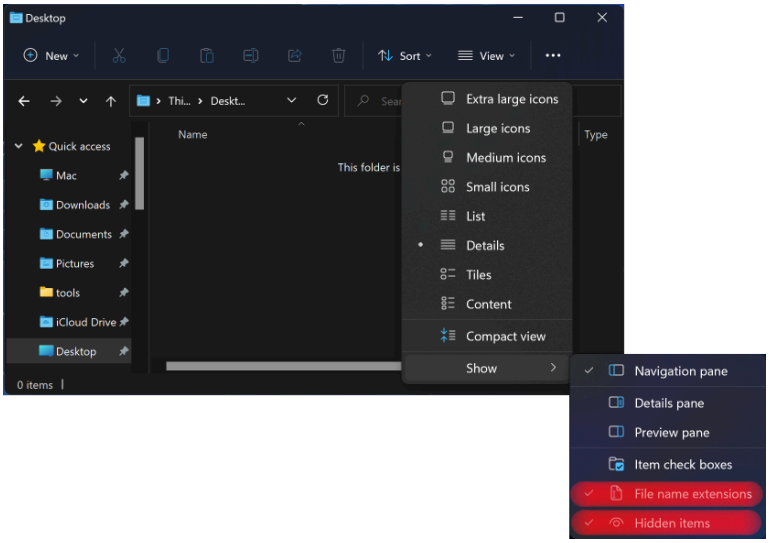

Abbildung 5: Windows Datei Explorer Konfiguration

3 Markdown

Markdown ist eine leichte Beschreibungssprache mit einer Syntax zur Formatierung von Klartext. Sie ist so konzipiert, dass sie leicht zu lesen und zu schreiben ist und gleichzeitig leicht in PDF, HTML oder andere Formate konvertiert werden kann. Markdown wird häufig für die Formatierung von Text im Web verwendet, z. B. in `README.md`-Dateien, Dokumentationen, Forenbeiträgen und Nachrichten.

Um Markdown zu schreiben, benötigen Sie Ihren bevorzugten Texteditor oder Sie können einen spezialisierten Markdown-Editor wie Marktext installieren.

Zum Beispiel ist die Einführungsseite dieses Kurses in Markdown geschrieben. Sie können den Quellcode der Seite sehen, indem Sie auf die Schaltfläche „Diese Seite bearbeiten“ in der oberen rechten Ecke der Seite oder über den link klicken.

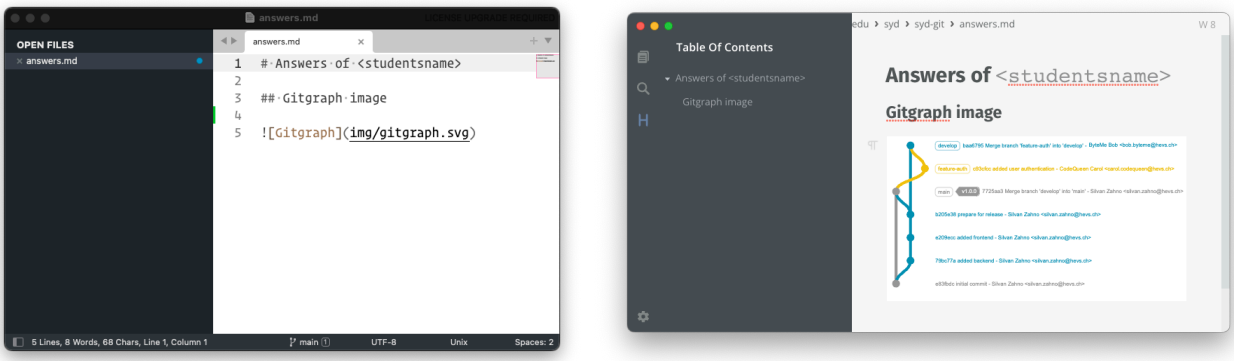

Abbildung 6: Links: Einfacher Texteditor (Sublime Text), Rechts: Marktext

Für das Labor müssen Sie ein Dokument im Markdown-Format schreiben. Halten Sie Ihren Editor bereit.

3.1 Markdown-Syntax

Nachfolgend ein kurzer Überblick darüber, wie eine Markdown-Datei aufgebaut ist. Die Syntax ist einfach und leicht zu erlernen. Die Datei muss mit der Endung `.md` gespeichert werden. Eine vollständigere Syntaxliste finden Sie unter hier.

````
# 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 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 Schlusswort

Glückwunsch Sie haben nun alles benötigte Installiert und konfiguriert um mit Git zu arbeiten. Bereiten Sie sich auf das nächste Labor vor:

- Studieren Sie die Theorie
- Familiarisieren Sie sich mit den Git Kommandos
- Familiarisieren Sie sich mit dem Grafischen Tool Sublimemerge
- Üben Sie das schreiben von Dokumenten mit Markdown

Im Anhang Abschnitt A und Abschnitt B finden Sie eine Zusammenfasssung der wichtigsten Git Befehle.

A GIT Befehle

Github git cheatsheet [4], [5]

AA Änderungen überprüfen und eine Commit-Transaktion anfertigen

```
git status
```

Listet alle zum Commit bereiten neuen oder geänderten Dateien auf.

```
git diff
```

Zeigt noch nicht indizierte Dateiänderungen an.

```
git add [file]
```

Indiziert den derzeitigen Stand der Datei für die Versionierung.

```
git diff --staged
```

Zeigt die Unterschiede zwischen dem Index (“staging area”) und der aktuellen Dateiversion.

```
git reset [file]
```

Nimmt die Datei vom Index, erhält jedoch ihren Inhalt.

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

Nimmt alle derzeit indizierten Dateien permanent in die Versionshistorie auf.

AB Änderungen synchronisieren

Registrieren eines externen Repositories (URL) und Tauschen der Repository-Historie.

```
git fetch [remote]
```

Lädt die gesamte Historie eines externen Repositories herunter.

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

Integriert den externen Branch in den aktuell lokal ausgecheckten Branch.

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

Pusht alle Commits auf dem lokalen Branch zu GitHub.

```
git pull
```

Pullt die Historie vom externen Repository und integriert die Änderungen.

B Meistgebrauchten Git Befehle

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

Bibliographie

[1] T. Linus, „Git“. Zugegriffen: 25. April 2023. [Online]. Verfügbar unter: https://git-scm.com/

[2] „GitLab Hevs“. Zugegriffen: 25. April 2023. [Online]. Verfügbar unter: https://gitlab.hevs.ch/

[3] „Sublime Merge - Git Client from the Makers of Sublime Text“. Zugegriffen: 25. April 2023. [Online]. Verfügbar unter: https://www.sublimemerge.com/

[4] gitlab, „Git Cheatsheet“. 2023.

[5] „GitHub Git Spickzettel“. Zugegriffen: 25. April 2023. [Online]. Verfügbar unter: https://training.github.com/downloads/de/github-git-cheat-sheet/