



Einführung in Git - Teil A

Installation und Einrichtung



Inhalt

1	Ziel	2
2	Installation	3
2.1	Sublime Merge	3
2.2	<code>git</code> - Kommandozeile	5
2.3	Online-Konten	6
2.4	Windows-Konfiguration	6
3	Markdown	7
3.1	Markdown-Syntax	8
4	Schlusswort	9
	Literatur	10
5	Anhang	11
A	GIT Befehle	11
AA	Änderungen überprüfen und eine Commit-Transaktion anfertigen	11
AB	Änderungen synchronisieren	11
B	Meistgebrauchten Git Befehle	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 | 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 \[1\]](#) lernen, insbesondere die Tool [Sublime Merge](#) und optional das [Git-Befehlszeilentool](#), die auf Ihrem Computer installiert und konfiguriert werden müssen (siehe [Abschnitt 2](#)). Darüber hinaus wird ein Konto auf der Plattform [Github](#) erstellt.

Schließlich lernen wir die Grundlagen von Markdown in [Abschnitt 3](#), um Textdateien einfach 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 und/oder Sublimemerge. Sie können selber auswählen ob sie die Kommandozeile oder das GUI benutzen möchten während des Labors.

2.1 Sublime Merge

Besuchen Sie die Webseite <https://www.sublimemerge.com> and laden sowie installieren Sie das Tool Sublime Merge herunter [2].



Abbildung 1 - Sublime Merge GUI

2.1.1 Konfiguration

Beim Klonen eines Repositories wird Sublime Merge automatisch nach deiner Identität fragen und dich auffordern, dich bei deinem Github-Konto anzumelden.



2.1.2 Übersicht

Die Oberfläche von Sublime Merge ist in der [Abbildung 2](#) dargestellt:

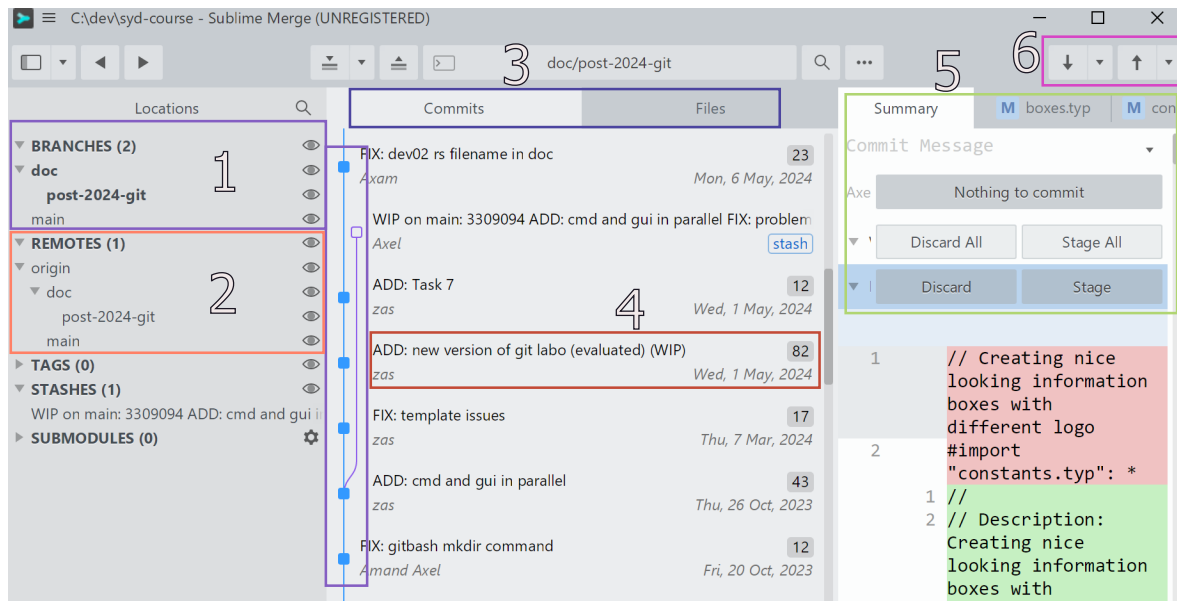


Abbildung 2 - Sublime Merge GUI

1. **Branches:** Liste der Branches des Repositories. Du kannst Branches erstellen, löschen und umbenennen. Sie werden auch in ihrer Chronologie unter dem Tab **Commits** (Punkt 3) angezeigt.
2. **Remotes:** Liste der Remote-Repositories, d.h. die Server, auf denen der Code gespeichert ist und auf die du **pushen** kannst.
3. **Commits/Files:** Tabs, um zwischen der chronologischen Ansicht der Commits und der Ansicht der Änderungen für den ausgewählten Commit zu wechseln.
4. **Commit-Beschreibung:** Ein Commit besteht aus einer Nachricht, einem Autor und einem Datum.
5. **Aktuelle Änderungen:** Die Dateien, die im Vergleich zum letzten Commit geändert wurden, werden hier aufgelistet. Du kannst die Dateien auswählen, die du **committen** möchtest, indem du sie auswählst und auf die Schaltfläche **Stage** klickst. Solange die Dateien nicht **committed** sind, kannst du eine Datei auch wieder aus dem Staging entfernen, indem du auf die Schaltfläche **Unstage** klickst oder die Änderungen einer Datei vollständig löschen, indem du auf **Discard** drückst (⚠️ endgültige Löschung ⚠️).
6. **Pull / Push:** Die beiden Schaltflächen ermöglichen es, **pull** - die neuesten Änderungen vom Remote-Repository zu übernehmen - und **push** - die lokalen Änderungen an das Remote-Repository zu senden. Es ist auch möglich, **fetch** zu verwenden, indem du auf den kleinen Pfeil neben der Schaltfläche **Pull** klickst, was es dir ermöglicht, die neuen Commits zu sehen, ohne das lokale Repository zu ändern.



2.2 git - Kommandozeile

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.

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 3 - git Terminal



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

```
git --help
```

2.2.1 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
```

2.3 Online-Konten

2.3.1 Github

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

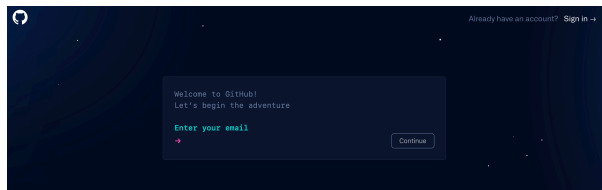


Abbildung 4 - GitHub Anmeldung

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“

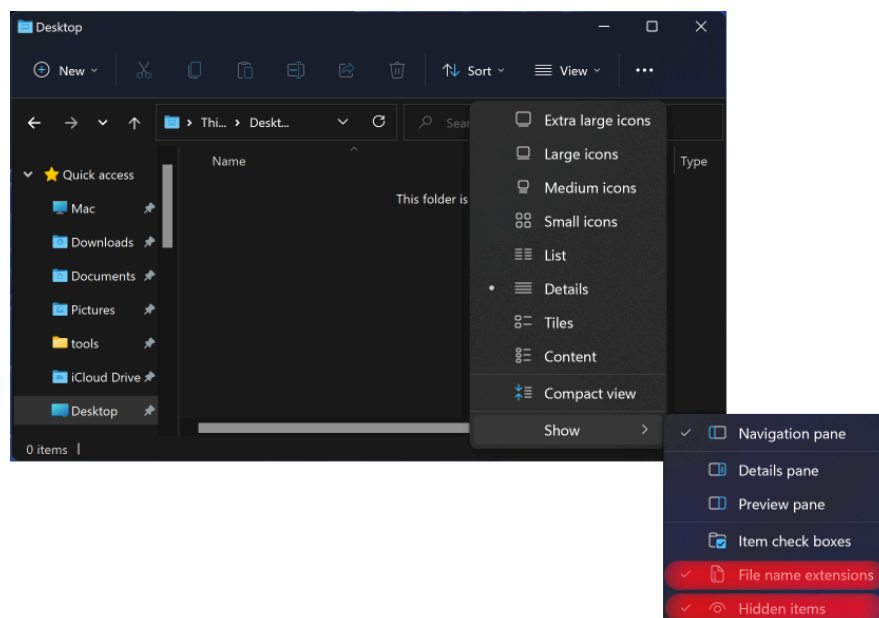


Abbildung 5 - Windows Datei Explorer Konfiguration



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 mit Unterstützung für das Format:

- [Sublimetext](#) mit dem [MarkdownLivePreview](#) Plugin
- [Zed](#) : öffnen Sie die Markdown-Datei und verwenden Sie `Ctrl/Cmd + Shift + V`, um die Vorschau zu öffnen
- [VSCode](#) : öffnen Sie die Markdown-Datei und verwenden Sie `Ctrl/Cmd + Shift + P => Markdown: Open Preview to the Side`
- [Markdown Live Preview](#) : Online Markdown Editor
- ...

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.

```

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)
  
```

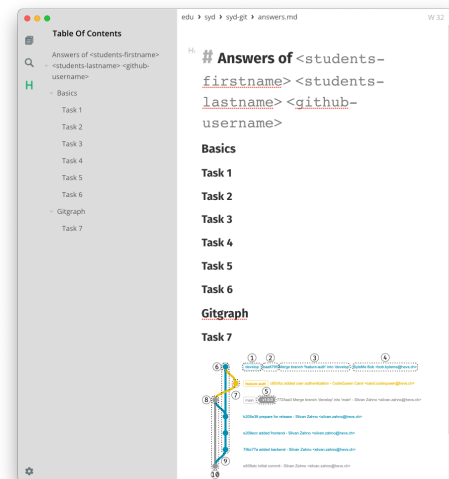


Tabelle 1 - Links: Einfacher Texteditor (Sublime Text), Rechts: Marktext



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 vollständige Syntaxliste finden Sie unter https://wiki.zahno.dev/multimedia/writing/md/md_github.html.

```
# Title 1

## Title 2

### Title 3

Some simple Text _italic_ **bold**
Strikethrough `monospaced`

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

- 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`

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

- 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 | Schlusswort

Herzlichen Glückwunsch! Sie haben nun alles installiert und konfiguriert, was Sie benötigen, um mit Git zu arbeiten. Sie sollten Folgendes getan haben:

- ☐ Git und Sublime Merge installiert
- ☐ Git mit Ihrem Namen und Ihrer E-Mail-Adresse konfiguriert
- ☐ Ein GitHub-Konto erstellt
- ☐ Mit der grafischen Benutzeroberfläche (GUI) von Sublime Merge vertraut gemacht
- ☐ Mit der Theorie und den Befehlen von Git vertraut gemacht
- ☐ Mit Markdown und seiner Syntax vertraut gemacht



Im Anhang [Abschnitt A](#) und [Abschnitt B](#) finden Sie eine Zusammenfassung der wichtigsten Git Befehle.



Literatur

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



5 | Anhang

A | GIT Befehle

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

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