

Einleitung zum Program



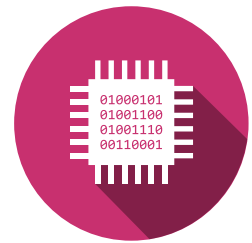
Inhaltsverzeichnis

1 Ziel	1
2 Einleitung	2
2.1 Gruppenaustausch	2
3 Persönliches GIT Repository	3
3.1 Erstellen des Persönlichen Repositories	3
3.2 Klonen des Repositories auf U:\	4
3.2.1 Via Kommandozeile	4
3.2.2 Via Sublime Merge GUI (optional)	5
3.2.3 Manuelles Kopieren (optional)	6
3.3 Speichert des Projektes nach jedem Labor auf Github	6
3.3.1 Via Kommandozeile	7
3.3.2 Via Sublime Merge GUI	7
4 GIT Befehle	10
4.1 Änderungen überprüfen und eine Commit-Transaktion anfertigen	10
4.2 Änderungen synchronisieren	10
4.3 Meistgebrauchten Git Befehle	10
4.3.1 Start a working area	10
4.3.2 Work on the current change	10
4.3.3 Examine the history and state	11
4.3.4 Grow, mark and tweak your common history	11
4.3.5 Collaborate	11

1 Ziel

Dieses Dokument dient als Vorbereitung zum Labor. Es wird benötigt um die Labor Dateien zu erhalten und zu verwalten.

Das selbe Prozedere kann auch für die Projekte angewendet werden.



2 Einleitung

Die Dateien für das Labor werden mithilfe des Github Classroom Tools verteilt. Jeder Student muss sein eigenes Git Repository haben und verwalten.

Git ist ein Werkzeug um Code Dateien verwalten zu können. Es erlaubt diesen sicher auf Github Servern zu speichern und verschiedene Revisionsstände zu speichern und zu vergleichen.

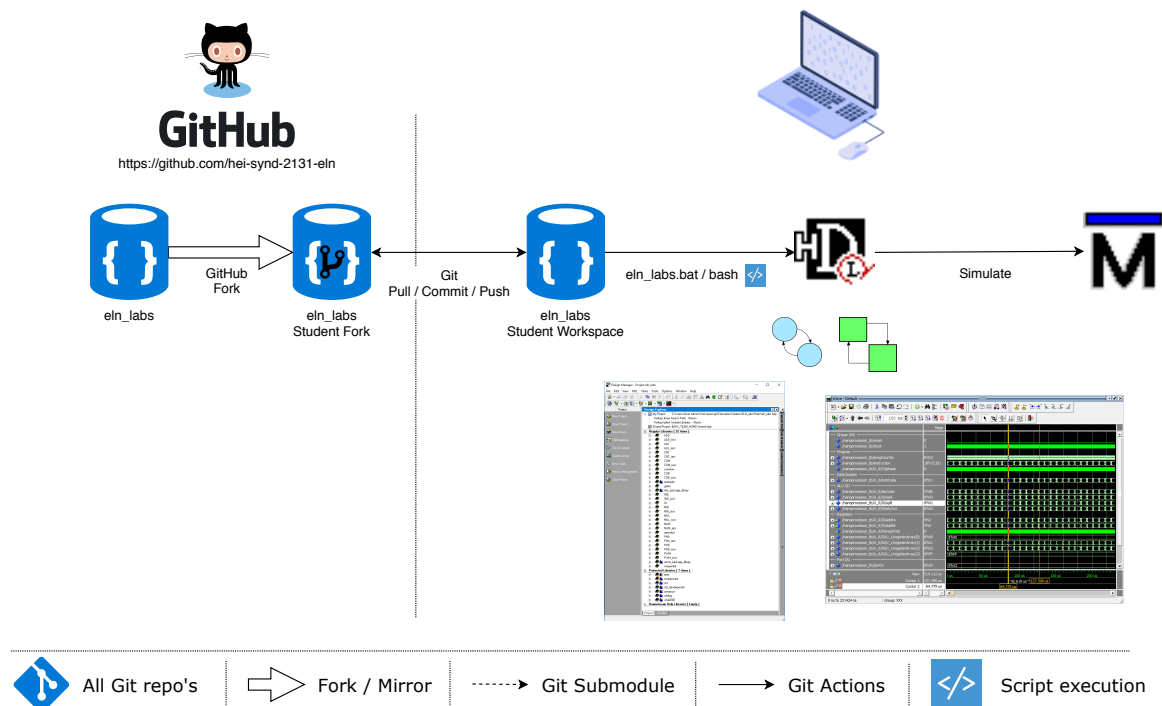
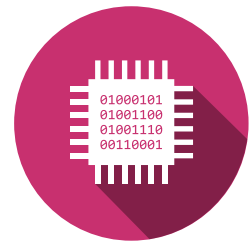


Abbildung 1: Github System

2.1 Gruppenaustausch

Während des Labors werdet Ihr in Gruppen arbeiten. Es ist wichtig das Ihr am Ende jedes Labors die Dateien unter den Gruppenmitgliedern verteilt. Hierzu genügt es den Ordner des jeweiligen Labors zu kopieren.



3 Persönliches GIT Repository

Ein Git Repository ist der Name einer beschriebenen Code Datenbank. Folgend werden 3 Methoden erläutern das bereitgestellte Repository auf das personelle Laufwerk U:\ zu transferieren.

3.1 Erstellen des Persönlichen Repositories

Erstellt einen Github Account mit eurem HEVS Email

- Besucht die Webseite: <https://github.com/join>
- Username: <Vorname><Nachname>
- Email: <Vorname><Nachname>@students.hevs.ch

Abbildung 2: Join Github page

Folgt den Einladungslink welcher Ihr per Email erhalten habt.

Er hat die Form <https://classroom.github.com/<X>/<YYYYYYYY>>

Abbildung 3: Github Classroom invitation

Akzeptiert die Einladung mit eurem soeben erstellen Github Account

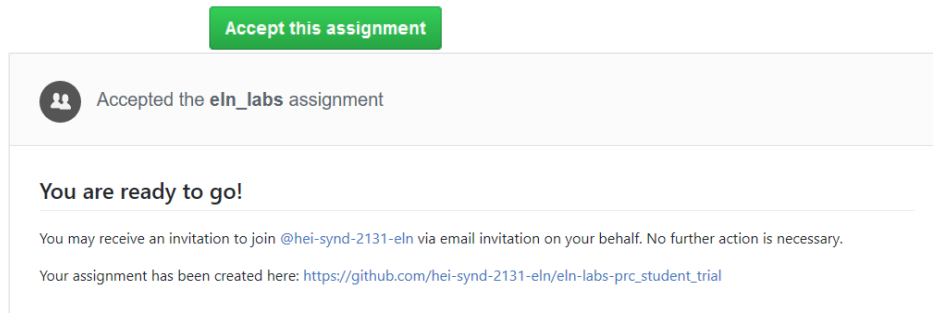
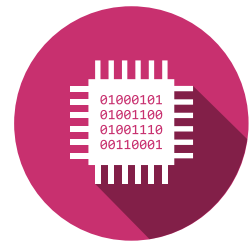


Abbildung 4: Github Classroom Assignment

Folgt dem Link um auf euer eigenes eln-labs Repository zu gelangen. Dort kann auch der Link zum Klonen gefunden werden.

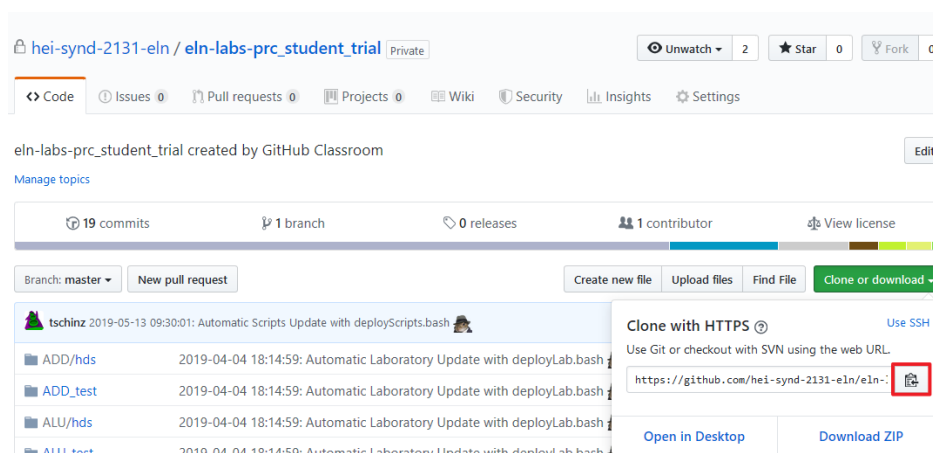


Abbildung 5: Github Student Repository

3.2 Klonen des Repositories auf u:\

3.2.1 Via Kommandozeile

Öffnen einer Kommandozeile

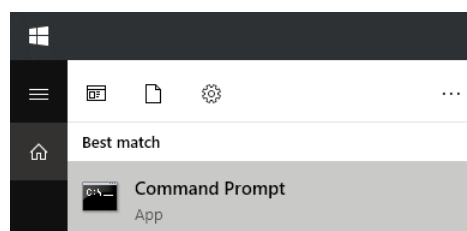
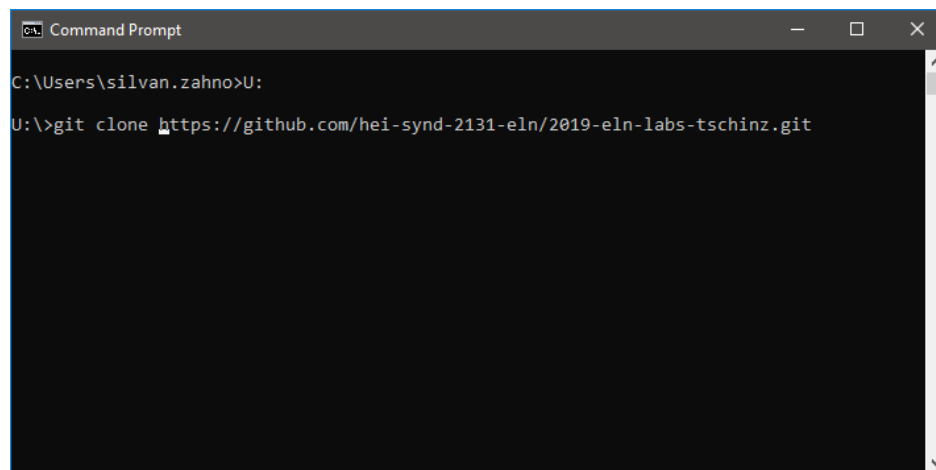
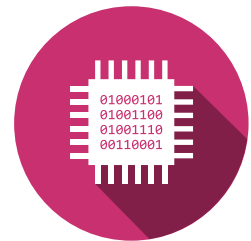


Abbildung 6: Starten Kommandozeile

Ausführen des `git clone` Befehles.



```
Command Prompt
C:\Users\silvan.zahno>U:
U:\>git clone https://github.com/hei-synd-2131-elN/2019-elN-labs-tschinz.git
```

Abbildung 7: Windows Kommandozeile

U:
git clone <Repository_URL>

3.2.2 Via Sublime Merge GUI (optional)

Öffne das Program Sublime Merge

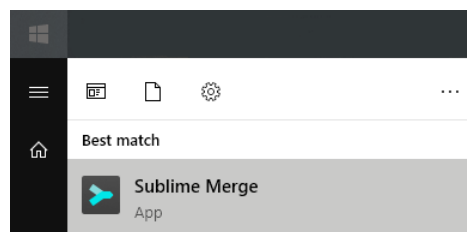


Abbildung 8: Starten Sublimemerge

File → Clone Repository

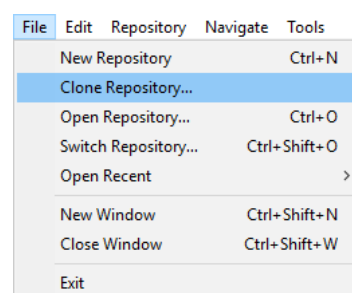


Abbildung 9: Sublime Merge starten klonen

Füllt das Formular Clone Repository aus

Source URL: von Github

Repository Name: leer lassen

Destination Path: wählt euer persönliches U:\ Verzeichnis aus

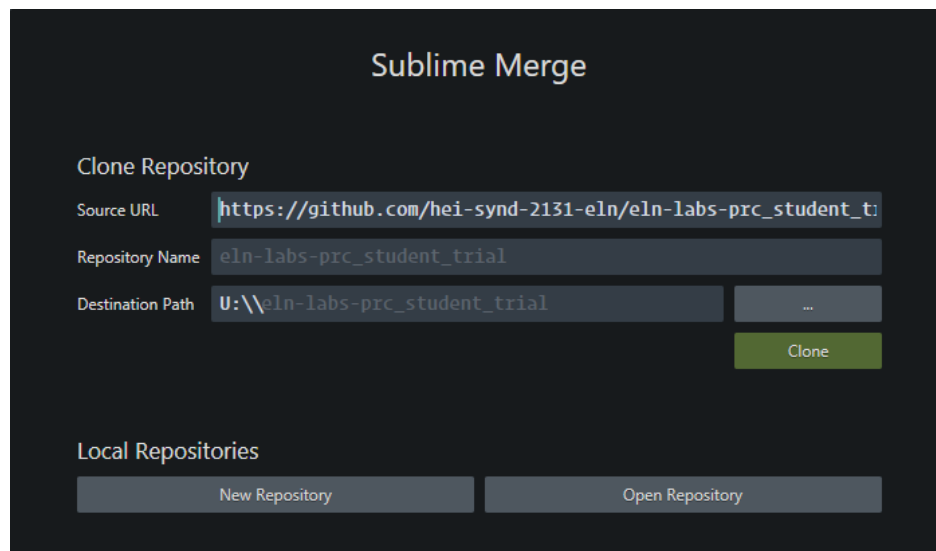
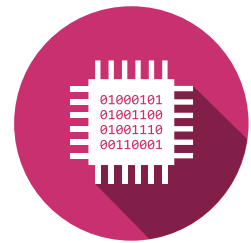


Abbildung 10: Sublime Merge klonen

Klickt auf den grünen Knopf Clone

3.2.3 Manuelles Kopieren (optional)

Als Alternative kann das Repository auch manuell als Zip heruntergeladen und im Verzeichnis U:\\ entpackt werden.

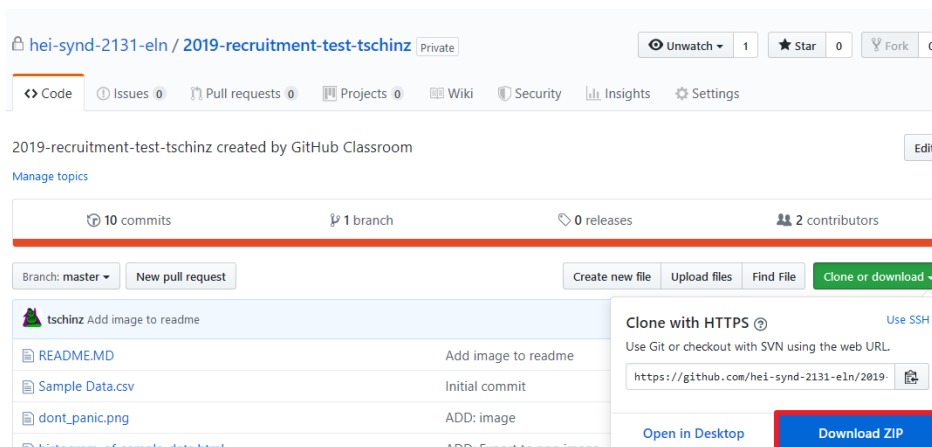


Abbildung 11: Manuelles herunterladen

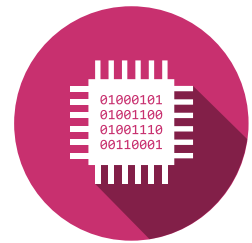
3.3 Speichert des Projektes nach jedem Labor auf Github



Da jeder Student sein eigenes git Repository hat, müssen nach jedem Labor die Gruppenmitglieder den Ordner mit dem jeweiligen Labornamen untereinander austauschen.

Zusätzlich können die veränderten Dateien auf Github hochgeladen werden. Die geschieht in 3 Schritten:

1. Stage - Auswahl der Dateien welche gespeichert werden sollen



2. Commit – Speichert die Dateien im lokalen Repository auf dem PC
3. Push – Synchronisiert das Repository auf Github mit dem lokalen Repository

3.3.1 Via Kommandozeile

Führe die folgenden Befehle in der Kommandozeile aus.

```
git add -all
git commit -m "Commit message ex. Labo Num"
git push
```

3.3.2 Via Sublime Merge GUI

Öffne das Program Sublime Merge

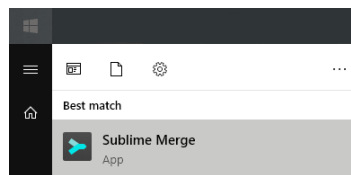


Abbildung 12: Starten von Sublime Merge

Dateien stagen bzw. zum speichern auswählen.

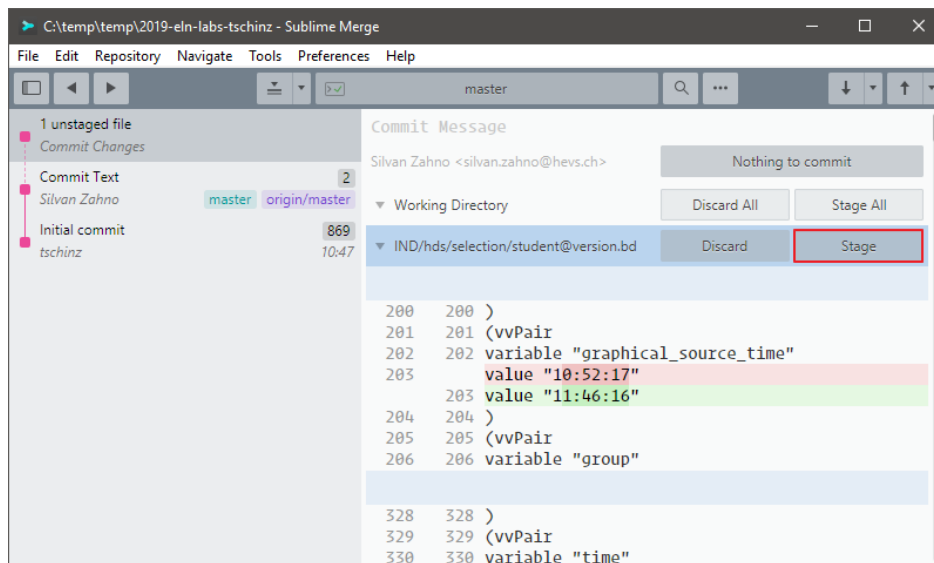


Abbildung 13: Auswählen von Dateien zum stage

Commit Nachricht schreiben: Was haben Sie geändert?

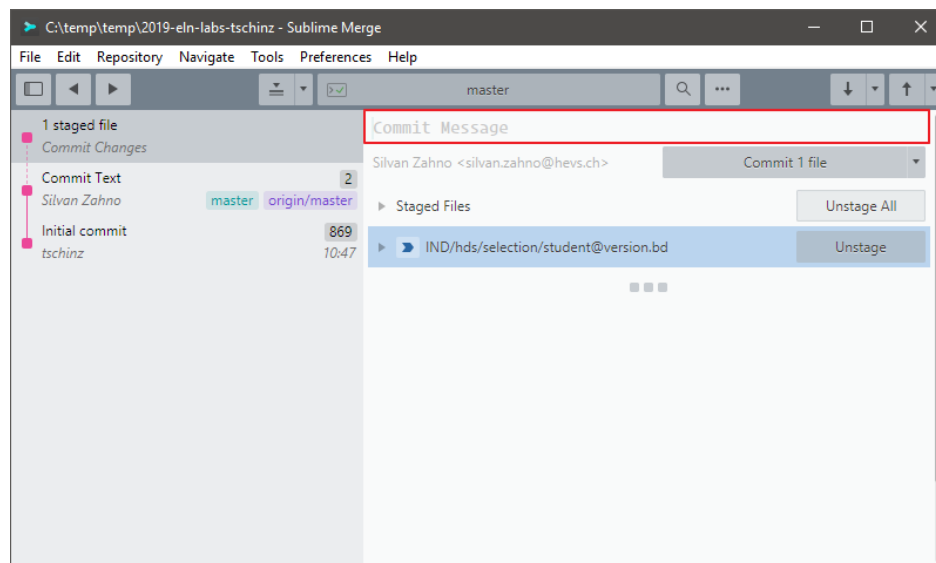
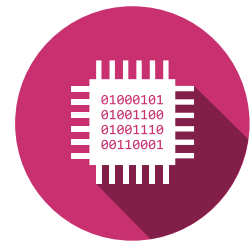


Abbildung 14: Eingeben der Commit Nachricht

Änderungen commiten bzw. im lokalen Repository speichern.

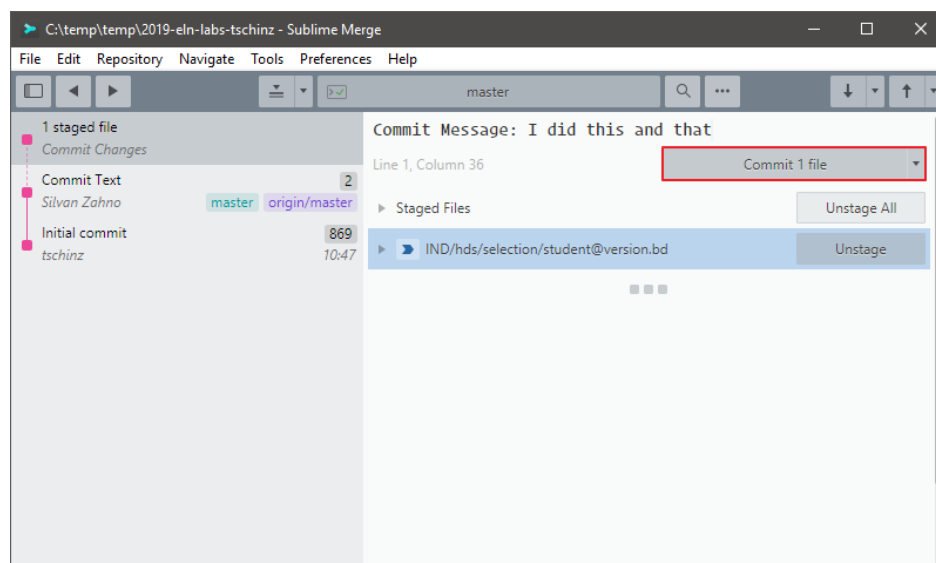


Abbildung 15: Commit der Änderungen

Änderungen pushen bzw mit dem Github repository synchronisieren.

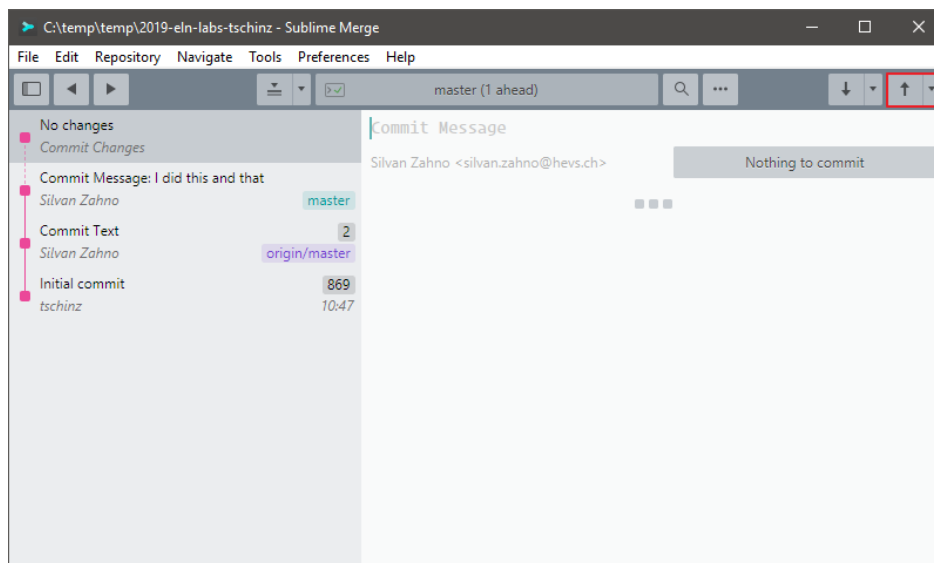
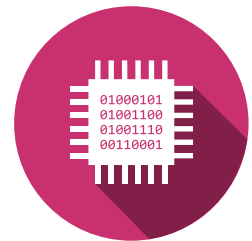


Abbildung 16: Push der Änderungen auf Github

Danach sehen Sie Ihre Änderungen lokal master und github origin/master auf dem gleichen Stand.

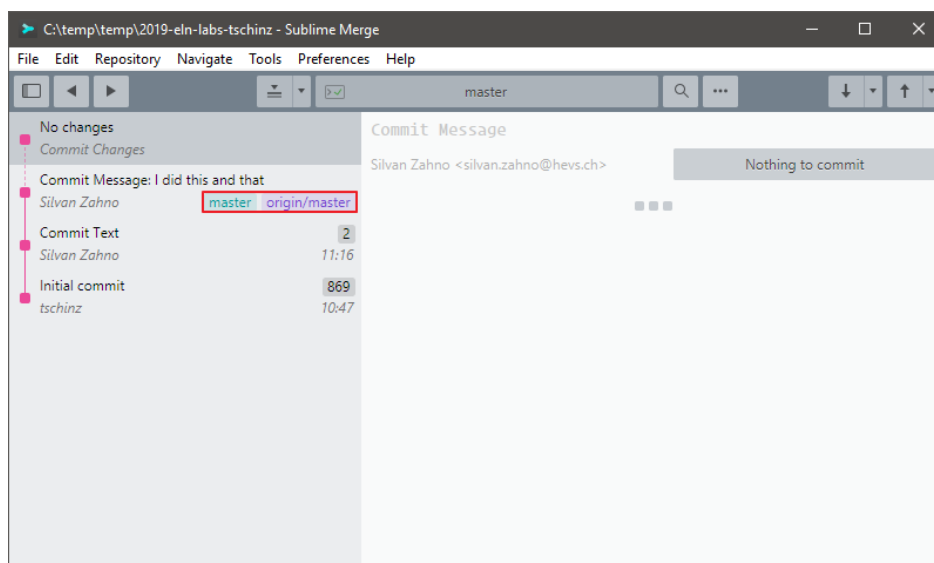
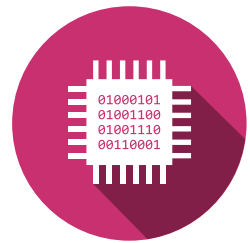


Abbildung 17: Änderungen befinden sich nun auch auf Github



4 GIT Befehle

[Github git cheatsheet](#)

4.1 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.

4.2 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.

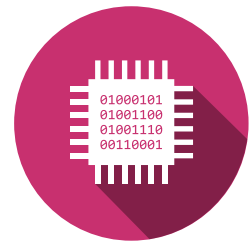
4.3 Meistgebrauchten Git Befehle

4.3.1 Start a working area

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

4.3.2 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

4.3.3 Examine the history and state

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

4.3.4 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

4.3.5 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