

AAP13: GIT

Fabio Plunser
Betreuer : Walter Mueller

June 9, 2020

Inhaltsverzeichnis

1	Verwendet PyCharm und klonst mit dem integrierten GIT Tools ein Projekt von GitHub	1
2	Legt einen eigenen Benutzer bei GitHub an. Legt im neuen Account auch eine Testrepository an.	2
3	Erzeugt mit clone eine Kopie Eures GitHub Repos in Eurer PyCharm Umgebung.	3
3.1	Erstellt in der Umgebung im Projekt weitere Dateien wie z.B. ein personalisiertes HelloWorld.py. . .	3
3.2	Commitet Eure Änderung in Euer lokales Repo	3
4	Pusht Eure Projektänderungen nach GitHub	3
5	Findet einen Partner in der Klasse und berechtigt Euch wechselseitig für Eure Git-Hub-Projekte.	4
6	Klonst das Repo Eures Partners in ein neues lokales PyCharm-Projekt auf Eurem PC. Bringt dort zuerst lokal und dann auch im Remote-Repo bei GitHub Eure Änderungen ein.	5
7	Was ist unklar geblieben, wo braucht ihr Hilfe?	6

Abbildungsverzeichnis

1	MUE-History on Git	1
2	Github Seite	2
3	MeinGit-History	3
4	History änderung nach Git push	3
5	Github Seite nach Push	3
6	Github-Berechtigung	4
7	Git-Page vom Lorenz	4
8	Kopie von Repo von Lorenz	5
9	Meine Änderungen	5
10	Änderung auf Github vom Lorenz	5

1 Verwendet PyCharm und klonet mit dem integrierten GIT Tools ein Projekt von GitHub

Github-Repo

Last das Programm laufen, schaut Euch die Git-Historie an. Macht und commitet Änderungen in Eurem lokalen Repo.

Mit der Extension git-history für Visual Studio Code kann genau nachgeschaut werden wer, wann, was geändert hat.

Wenn viele Personen an einem Projekt arbeiten ist das wirklich unglaublich nützlich.

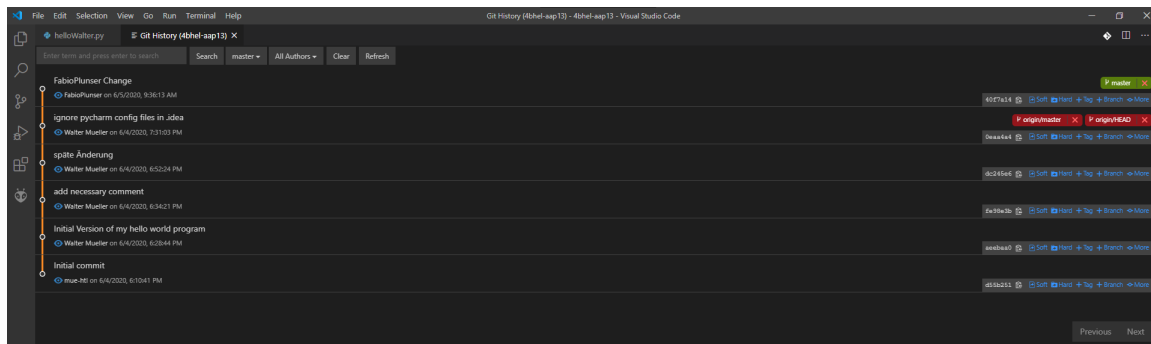


Figure 1: MUE-History on Git

2 Legt einen eigenen Benutzer bei GitHub an. Legt im neuen Account auch eine Testrepository an.

(dokumentiert hier Eure Benutzererkennung sowie die URL Eurer Repositories bei GitHub und macht einen screenshot Eurer Github-Webseite)

<https://github.com/FabioPlunser/FSST4BHEL>

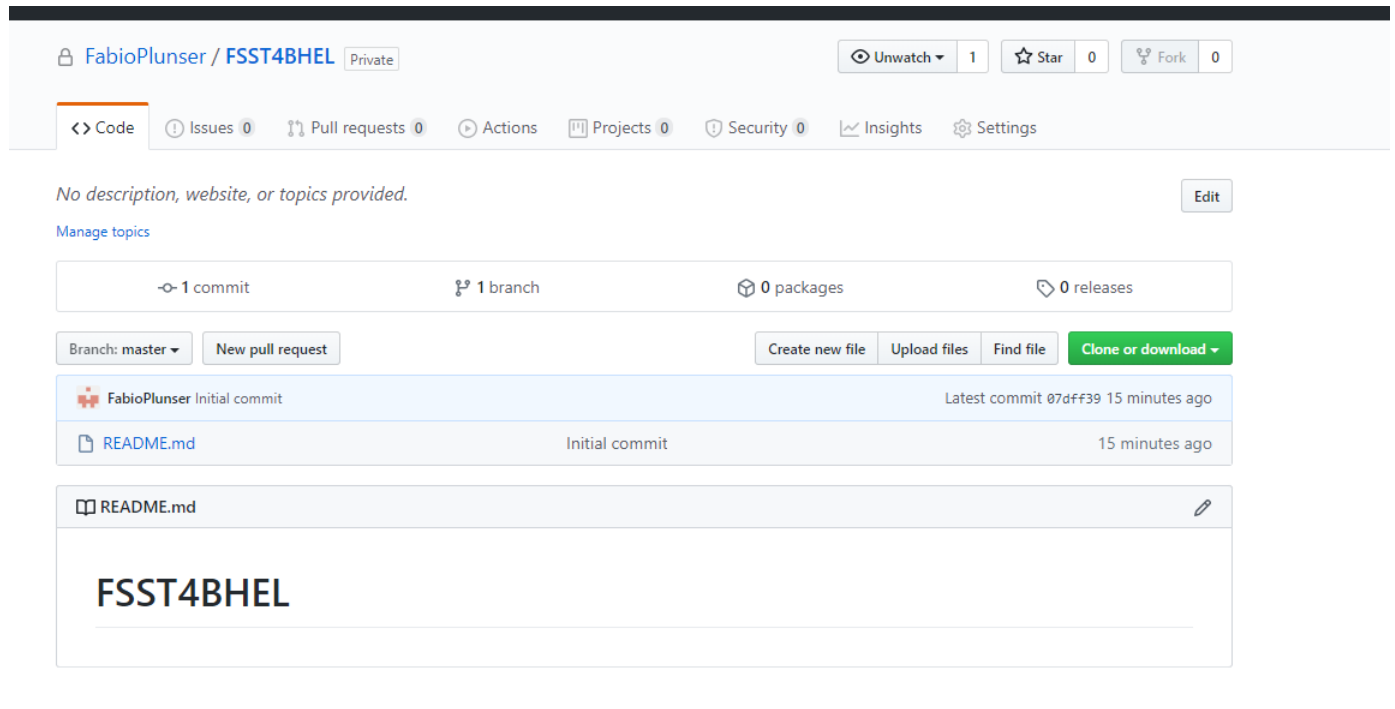


Figure 2: Github Seite

3 Erzeugt mit clone eine Kopie Eures GitHub Repos in Eurer Py-Charm Umgebung.

3.1 Erstellt in der Umgebung im Projekt weitere Dateien wie z.B. ein perosnalisiertes HelloWorld.py.

3.2 Commitet Euro Änderung in Euer lokales Repo

Vorsicht: Beachtet dabei Eure Privacy - ihr werdet das Projekt ja bei GitHub publizieren.

In meinem Fall wird Visual Studio Code verwendet mit der GIT-History extension.
Man sieht zwar nicht direkt die DateiSturktur es reicht aber vollkommen aus, auch wenn man die Commands weiß benötigt man nur die weboberfläche.

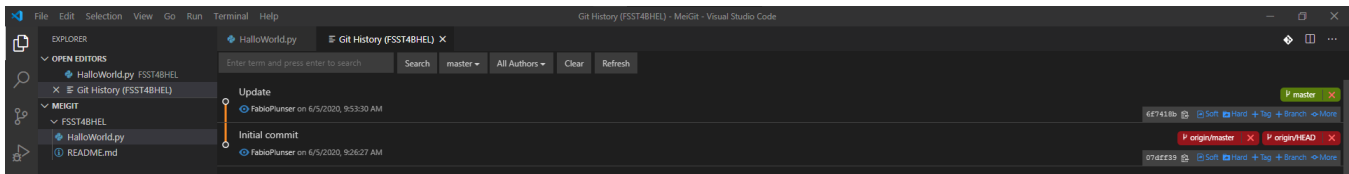


Figure 3: MeinGit-History

4 Pusht Eure Projektänderungen nach GitHub

(screenshots der Webansicht Eures Projekts)

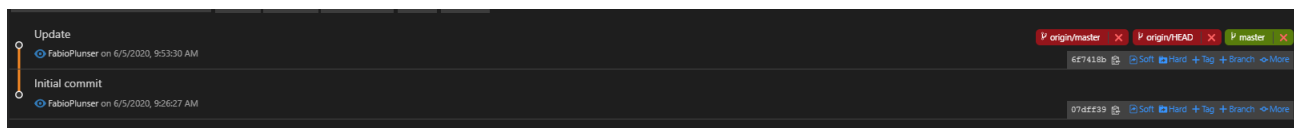


Figure 4: History änderung nach Git push

FabioPlunser Update		Latest commit 6f7418b 16 minutes ago
HalloWorld.py	Update	16 minutes ago
README.md	Initial commit	43 minutes ago

Figure 5: Github Seite nach Push

5 Findet einen Partner in der Klasse und berechtigt Euch wechselseitig für Eure Git-Hub-Projekte.

Partner Mitterhuber Lorenz

Habe noch nicht ganz verstanden wic ich einstellen kann was der Lorenz machen darf und was nicht bzw. habe es nirgends gefunden

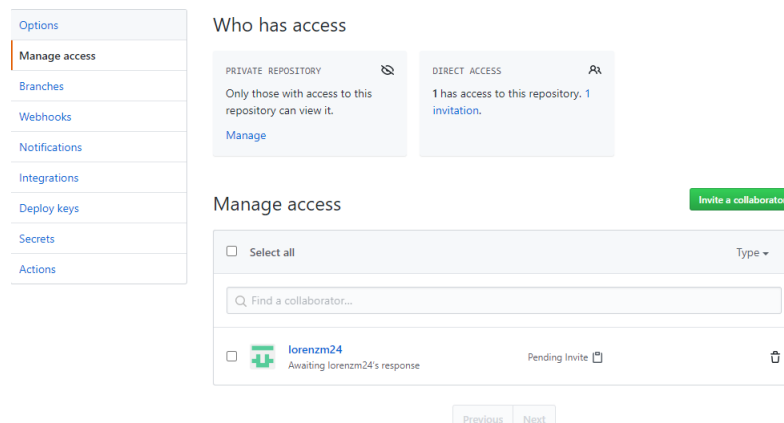


Figure 6: Github-Berechtigung

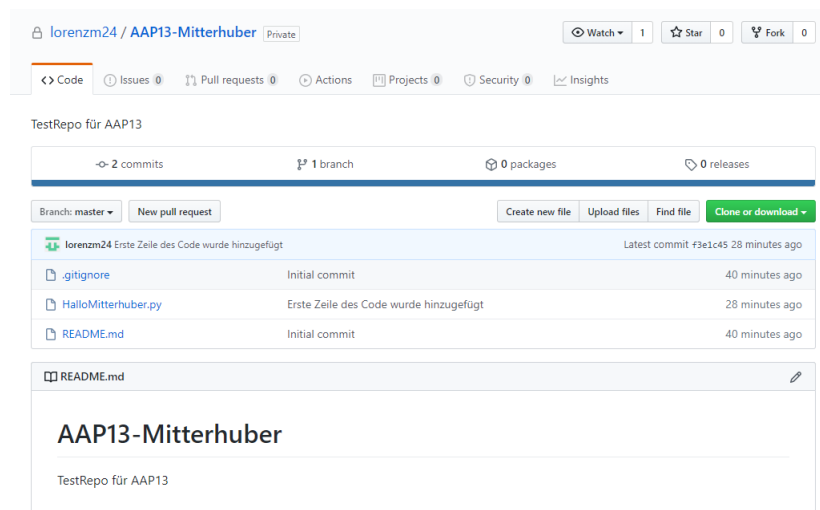


Figure 7: Git-Page vom Lorenz

- 6 Klont das Repo Eures Partners in ein neues lokales PyChram-Projekt auf Eurem PC. Bringt dort zuerst lokal und dann auch im Remote-Repo bei GitHub Eure Änderungen ein.

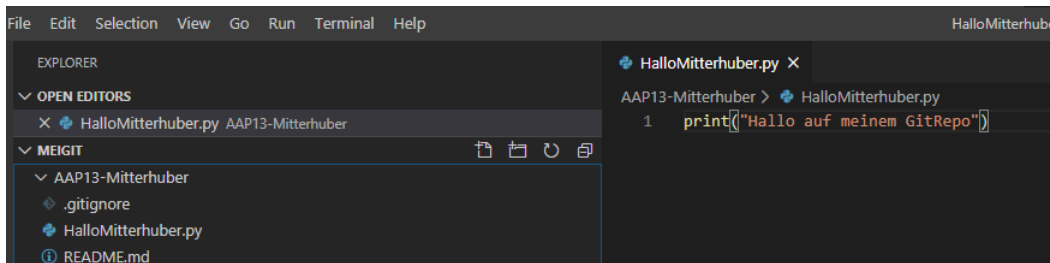


Figure 8: Kopie von Repo von Lorenz

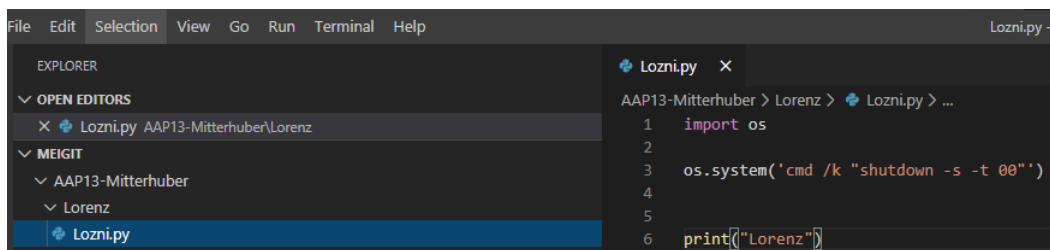


Figure 9: Meine Änderungen

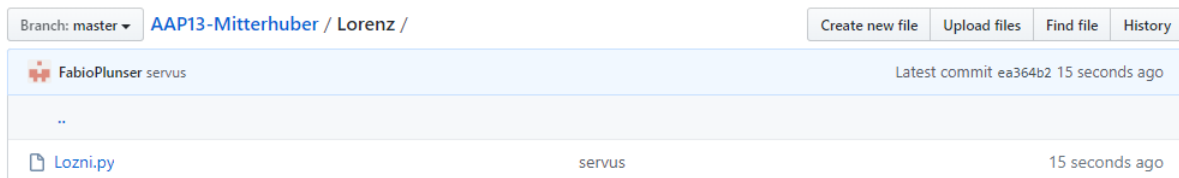


Figure 10: Änderung auf Github vom Lorenz

In der File History kann genau nachgeschaut werden, wer, wann was geändert hat

7 Was ist unklar geblieben, wo braucht ihr Hilfe?

Was meinen sie mit der Benutzerkennung?

Wie kann man die Berechtigungen genauer bestimmen habe das nicht rausgefunden.