

Übungsaufgaben: Git und die Wolke

Motivationshilfe:

- Wer Aufgabe I fertig gemacht hat kriegt einen Octocat-Sticker.
- Wer sich bei Aufgabe II hervortut bekommt auch eine kleine Überraschung.

Voraussetzungen:

Ihr habt aufgepasst, sonst keine. Umgang mit der Shell ist aber hilfreich.

I - Git Lokal

Ein wenig lokales Warm-Up mit `git`. (Dauer ca. 15 min.)

Folgende Schritte sind mit Hilfe des Cheatsheets durchzuführen:

1. Lege einen Ordner an und Initialisiere ein neues Git Repository
2. Schreibe eine neue `README.txt` Datei und füge sie dem `stage` hinzu.
3. Committe deine Änderungen und schaue vorher und nachher nach mit `git status` nach was sich ändert hat.
4. Prüfe dein Vorgehen in eine Visualisierungstool deiner Wahl. (z.B. `git log`)
5. Lege ein `.gitignore` Datei an und exkludiere darin alle Files mit der Endung `.txt`. Stell sicher dass es funktioniert hat. Wird `README.txt` ignoriert?
6. Branching:
 1. Lege einen neuen Branch `readme-improve` an.
 2. Mach einen neuen commit, und verändere in diesem `README.txt`
 3. Wechsel zum `master` branch.
 4. Mache auch dort einen Commit in dem du `README.txt` veränderst.
 5. Merge `master` mit `readme-improve`! Fortgeschrittene können hier auch `git rebase` nutzen.



II - Collaboration Game

Bei `git` geht es um Zusammenarbeit. Deshalb wollen wir einen Workflow gemeinsam mal probieren.

1. Bildet Gruppen von 2 - 3 Mann. Maximal 8 Gruppen.
2. Legt euch einen GitHub-Account an. Ihr könnt ihn später auch wieder löschen.
3. Teile `git` mit wer ihr seid (Tipp: Cheatsheet.)
4. Hier findet ihr ein Python-Projekt das noch nicht ganz fehlerfrei ist:

<https://github.com/studentkittens/git-python-project.git>

Forkt dieses Projekt!

5. Clont das geforkte Projekt in eure VM:

```
$ git clone https://github.com/<euer_user>/git-python-project.git
$ cd git-python-project
```

6. Wer hiermit fertig kriegt von uns einen Task (meldet euch!).

Jeder Task besteht aus einer fehlerhaften Python Funktion. Eure Aufgabe ist es nun diese entweder durch Überlegung zu reparieren, oder unter Anwendung der vorgestellten Git-Tools. Weitere Hinweise findet sich auch im Quelltext.

Auf den Zettel den eine Gruppe bekommt steht der Name des Directories das ihr bearbeitet. Darin findet sich auch immer nur eine `.py` Datei mit der Aufgabe. Editiert diese.

7. Wenn ihr fertig seid prüft hiermit nach ob der Test durchläuft:

```
$ make test_<task_name>
```

8. Falls ja: Pusht euren Code zu eurem Fork.
9. Macht ein Pull Request auf das Ursprungs Repository.
10. Sollte alles gut gehen sollte die LED vorne von Rot nach Grün wandern.

Tipps und Hinweise:

1. Der erste interessante Commit ist mit *"initial"* getaggt. D.h.: Ihr könnt folgendes machen:

```
$ git bisect start HEAD initial
```

2. Nach einem `git clone` ist nur der master branch vorhanden. Andere Branches nur als sog. *"Remote Tracking Branches"*. Um daraus einen benutzbaren Branch zu machen müsst ihr einen *"Local tracking branch"* anlegen:

```
$ git checkout -b <name> origin/<name>
```

3. Die Aufgabennamen und die dazu empfohlenen git Kommandos:

| Git Kommando | Aufgaben |
|------------------------------------|---------------------------|
| <code>git bisect / show</code> | monte_carlo_pi, fibonacci |
| <code>git branch / checkout</code> | int2hex, fac |
| <code>git grep / blame</code> | deduplicate, euler |
| <code>git log / git show</code> | reverse, count_even |

III. Gource

Keine Panik, Ihr müsst nichts machen.

Zum Abschluss visualisieren wir dann eure Arbeit mit gource und gitstats.