Informatik für Data Science 2

Tutorium - Termin 7 (04.05.2021)

Die Lösung (und weiteres Material) sind in GitHub (Repository: https://github.com/hka-mmv/dscb230-tutorial.git)

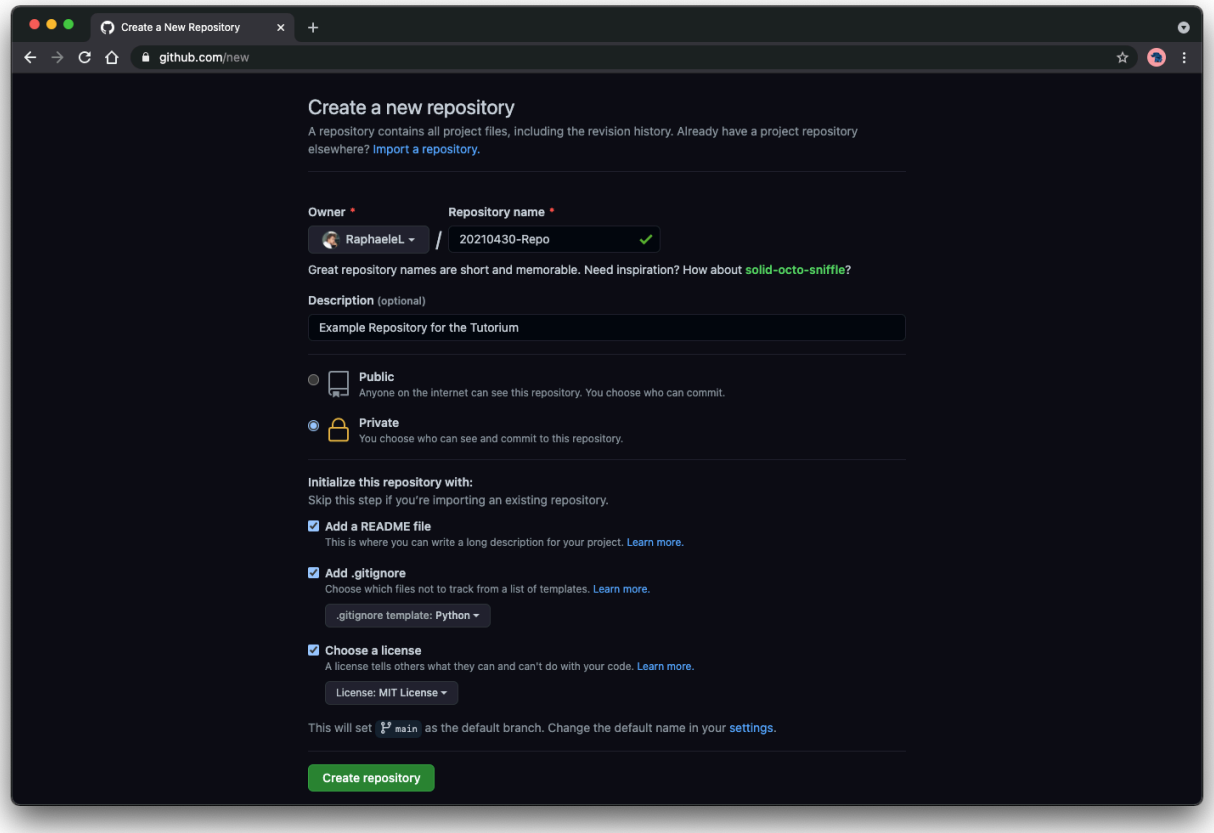
Werkzeug konfigurieren

Konfiguriert eure Benutzerinformationen für alle lokale Repositories.

```
$ git config --global user.name „ABC DEF“
$ git config --global user.email „aaaa0000@hs-karlsruhe.de“
$ git config --global user.password „mypassword_is_01234567890“
```

Repositories anlegen

Erstellt ein privates Github Repository über, welches die `MIT Lizenz` beinhaltet, ein automatisches generiertes `README.md` (Markdown Format) sowohl eine `.gitignore` Datei für Python beinhaltet.



Bitte klont das Repository auf euren Lokalen Rechner.

```
$ git clone https://github.com/RaphaeleL/20210430-Repo.git
```

Ändert das `README.md` so ab, dass jeder Entwickler bei dem Betreten des Repository alle nötigen Informationen sofort ablesen kann. Bitte denkt an eine aussagekräftige Commit Nachricht.

```
Einführung in das Projekt, Inhaltsverzeichnis, Anforderungen,
Voraussetzung, Starten des Projektes, Alternative Umgebungen,
Danksagungen, Kontakt
```

Bitte erstellt einen Ordner `my-secrets` und sorgt dafür, das dieser nicht in das remote Repository geladen wird.

```
$ echo „my-secrets/„ >> .gitignore
```

Änderungen gruppieren

Lasst euch alle vorhandenen Branches anzeigen und erzeugt an einen neuen Branch `Issue-512`. Wechselt in diesen.

```
$ git branch --av
$ git branch Issue-512
$ git checkout Issue-512
```

Erstellt auf diesem Branch eine Python Datei, welche genau „`Hello, World!`“ auf der Konsole ausgibt. Anschließend sollen diese Änderungen auch auf dem Remote Repository vorhanden sein.

```
$ touch hello-world.py
$ echo "print('Hello, World!')" > hello-world.py
$ git add hello-world.py
$ git commit -m „Implementing Feature XYZ“
$ git push --set-upstream origin Issue-512
```

Bitte Vereinigt den default Branch mit dem `Issue-512` Branch. Anschließend soll der `Issue-512` Branch Lokal und Remote gelöscht werden.

```
$ git checkout main
$ git merge Issue-512
$ git branch -d Issue-512
$ git push origin --delete Issue-512
```

Historie überprüfen

Durchsucht und inspiziert die bisherigen Entwicklungsabschnitte eures Projektes.

```
$ git log --graph
```

Nun durchsucht euer Projekt nach einer Spezifischen Datei (beispielsweise eure neue Python Datei).

```
$ git log --follow hello-world.py
```

Macht euch ein Bild über einen nun gefundenen Commit. Was wurde wo und von wem geändert?

```
$ git show 770ab29ca79322765dec0eb5cd255a9263782891
```

Gemeinsames Arbeiten

Stellt euch vor jemand anderes hat eine Änderung an dem Projekt getan. Wie könnt ihr euren bisherigen Veränderungen in das remote Repository laden ohne die Änderungen eures Kollegen zu verlieren?

```
$ git pull
$ git add .
$ git commit -m „Changes for Issue 512“
$ git push
```

Nur Kollege hat einen Fehler gemacht und die Falsche Datei hochgeladen. Bitte löscht die falsche Datei aus dem Remote Projekt. Wenn ihr schon dabei seid könnt ihr gleich das `README.md` aktualisieren und ebenso hochladen.

```
$ vim README.md
$ git pull
$ git add README.md
$ git rm my-secrets.json password-list.lst
$ git commit -m „Remove any Secrets and Updating README.md“
$ git push
```

Tagging

Ihr habt erfolgreich die Version 0.0.1 abgeschlossen. Nun könnt ihr das Repository mit einem entsprechenden Tag versehen und endlich das Release erstellen. Bitte macht dies.

```
$ git tag v.0.0.1
$ git push origin v.0.0.1
```

Leider hat sich der Kollege erneut verschrieben, löscht den Tag.

```
$ git tag -d v.0.0.1
$ git push origin :refs/tags/v.0.0.1
```