Github Einführung

Wir haben Github bereits gestern kennengelernt, als wir den gesamten Inhalt dieses Kurses auf unser Gerät installiert haben. Dieser Ordner soll euch als Referenz gelten. Und von hier werdet ihr die Hausaufgaben herauskopieren. Aber dazu später mehr.

Es handelt sich bei Github um die wohl grösste Ansammlung von offenem Code, den es auf der Welt gibt. Die Plattform ist so etwas wie das Facebook für Programmierer. Nur, dass hier ein System entwickelt wurde, das dafür sorgt, dass ihre Nutzer kreativ sind. Github ist natürlich nicht frei von Viren oder anderem schädlichen Code. Doch weil der ganze Code hier offen ist, ist es für Kriminelle sehr viel schwieriger, schädliche Software hier zu publizieren. Wer mehr dazu lesen will, der findet hier eine gute Einführung.

Github ist nicht die einzige Plattform seiner Art: Bitbucket oder Gitlab sind andere. Github ist allerdings derzeit die Grösste. Ob das so bleibt, ist ungewiss. Vielen missfällt, dass Microsoft Github gekauft hat. Und vorallem Gitlab erfährt zur Zeit viel Wachstum.

Den Kurs-Ordner aktuell halten

- `git clone` haben wir gestern schon kennengelernt. Damit kann man den gesamten Inhalt einer Repo auf den eigenen Computer übertragen.
- Github ermöglicht es dann mit einem komplexen System an Rechten und Verifizierungen Hunderten Entwicklern, an demselben Programm zu arbeiten, ohne des es zu Vandalenakten kommen kann - oder, dass jemand, ohne es zu Wissen, eine Entwicklung verschlimmbessert.
- Wir werden in diesem Kurs nicht mit diesem komplizierten Rechtesystem arbeiten. Das ist für den Anfang zu kompliziert. Ihr werdet den Kurs-Clone mit folgendem Befehl à jour halten. Dafür navigiert ihr im Terminal in den Kurs Order `cd kurs_18_19` und führt dann `git pull` aus.
- Wichtig ist nun, nie, nie, NIE etwas an diesem Ordner zu ändern. Denn, sobald ihr das tut, werdet ihr die Kurs-Files nicht mehr updaten können. Der Computer wird Dir sagen, dass er die Änderungen von der Plattform nicht auf Deinem Gerät abspeichern kann, weil er ja dann Deine Änderungen überschreiben würde.
- Um das Problem zu lösen, würden nun eine komplizierte Folge von Befehl angewandt. Aber das werden wir uns hier nicht zumuten. Wir müssen uns zuerst überhaupt an diese fremde Plattform gewöhnen.
- Ihr ändert also nichts in diesem Ordner. Um mit den Dateien dennoch zu arbeiten, kopiert ihr sie aus diesem Ordner, und in einen neuen. Ihr könnte das im Terminal tun. Oder, wenn es euch gefällt, mit der guten alten Drag-und-Drop-Methode.

Die eigene Github-Repo

- Repo ist die Abkürzung von Repository, was auf Englisch Lager oder Depot heisst. Depot gefällt mir hier ganz gut. Statt einen Zug halten wir Code im Depot. Wir nehmen ihn gelegentlich raus, fahren ihn herum, verbessern ihn und versorgen ihn dann wieder im Depot.
- Wir werden jetzt die erste eigene Repo kreieren. Dazu brauchen wir ein Nutzerkonto auf Github.
- Start a Project.
- Name; Beschreibung; setzen wir die Repo auf öffentlich, wir können auch private Repos einrichten, die man nicht mit der Öffentlichkeit teilen will.
- gitignore "Python" auswählen. Aber eigentlich ist es egal. Wir werden das ohnehin noch ergänzen. Gitignore sorgt dafür, dass die auf dem Computer versteckten Dateien nicht auch noch mitgeliefert werden. Wenn wir die

versteckten Files in einem Ordner sehen wollen, navigieren wir im Terminal auf in diesen Ordner, und geben `ls -a` ein. Nun erscheinen sie alle.

- Lizenz ergänzen. Wählt MIT aus. Alles darf geteilt werden. Und der Code darf auch kommerziell genutzt werden. Die einzige Bedingung ist, dass der Code, der damit funktioniert, ebenfalls frei verfügbar genutzt werden kann. Wer mehr darüber wissen will, kann hier mehr lesen.
- Nun müssen wir unserem Computer die Berechtigung geben, dass er sich mit Github austauschen darf. Wir gehen also zum Terminal und navigieren zum Desktop. Das müssen wir jeweils nur einmal machen.
- `git config --global user.name "Peter Müller"` , derselbe Nutzername, den wir auf Github benutzt haben.
- `git config --global user.email deine@email.com` , dieselbe Email, die ihr auf Github gebraucht habt.
- `git config --global user.password "your password"` , und nun setzt ihr noch das Password. Alle diese Informationen werden nun einem File abgespeichert, den ihr nicht mehr bedienen müsst.
- Mit `git config --list` könnt ihr alles überprüfen.
- Nun clonen wir die eben geschaffene Repo. Dazu holen wir die URL und geben folgendes in den Terminal ein: `git clone DEINEURL` . Auch das müssen wir natürlich nur einmal machen.
- Jetzt sind wir startbereit. Schauen wir, ob alles so funktioniert, wie es sollte. Öffnen wir im Terminal das README.md file mit Atom. Dazu geben wir in das Terminal ein: ```atom README.md```
- Ergänzen wir folgenden Text im Readme-File: "Das ist mein Titel", wir speichern es ab, nicht vergessen. Und nun synchronisieren wir unser Folder mit Github.
- `git add .` Alles in diesem Folder wird ergänzt. Wir könnten auch spefizische Folder ergänzen, statt einem "Punkt" würden wir den Namen des Folders in "eingeben".
- `git commit -m "nachricht"` mit diesem Befehl geben wir der Aktion einen Namen. Das ist für sehr komplexe Projekte hilfreich.
- `git push` Nun wird alles in die Cloud gepusht. Wenn es Unstimmigkeiten gibt, sorgen die ersten beiden Befehl dafür, dass alles erhalten bleibt, nicht wird überschrieben. Aber das sollte jetzt nicht passieren, weil ihr die einzigen seid, die mit diesem Github-Repo arbeitet.
- Üben wir das nun. Macht ein neues File. Nennt es File "01 Dokument". Schreibt ein paar Sätze rein. Dann pusht es auf Github. Nun geht ihr auf Slack. Und teilt eure Repo mit allen.
- Genauso in diesem Prozess werdet ihr Hausaufgaben lösen.