Herzlich willkommen zu unserer Präsentation, How to GitHub. Ich werde heute den theoretischen Teil übernehmen und werde versuchen euch zu erklären, was GitHub ist, wieso is des sinnvoll in der heutigen Zeit und dann werde ich euch ein paar Begriffe näher bringen. Danach wird mein Kollege euch praktisch ein paar wichtige Funktionen von GitHub zeigen und abschließend schauen wir uns ein kurzes Video an.

Git und GitHub

Fangen wir mit ein paar Daten an.

Der Firmensitz von GitHub ist in San Francisco und gehört seit 2018 zu Microsoft. Die Website hat ca. 40 Millionen User.

**Git**

Sinn und Funktion von GitHub stecken im zweiteiligen Namen. Git ist nämlich der Name einer Software zur Versionsverwaltung von Projekten. Nun, Und wos is des schon wieder?

An einem Software-Projekt arbeiten heutzutage oftmals mehrere, teilweise sogar Hunderte oder Tausende Entwickler mit. Jeder widmet sich seinem Fachgebiet und arbeitet an einem anderen Teil des Programms, und deren Arbeitsergebnisse müssen irgendwo zusammengeführt werden. Um den Code nicht immer an eine zentrale Person zu schicken, die den Code dann immer wieder aktualisiert, wird die Software Git verwendet. Diese automatisiert das Ganze.

Linus Torwald hat übrigens die Git Software erfunden. Und eins der erfolgreichsten Open-Source Projekte ist Linux

GitHub stellt Kommandozeilen-Tools zur Verfügung mit denen Entwickler ihre Änderungen an einem Projekt einreichen und abspeichern können. Diese Änderungen werden wiederum ausführlich auf der Website dargestellt. Git speichert außerdem jede Version des Software-Projekts – egal, wie groß oder klein die Änderung ist. Mit Git kann man immer auf die vorherige Version zugreifen.

* **Repository**: Ein Repository (oder kurz Repo) kann bei GitHub einfach als Projekt verstanden werden. Die Dateien für ein Software-Projekt werden in einem Repository abgelegt. Der Begriff stammt aus der Linux-Welt.
* **Branch**: Innerhalb eines Repositorys kann es mehrere Versionen einer Software geben. Zum Beispiel eine experimentelle Beta-Version und eine stabile Version für den Produktiv-Einsatz. Jede Version stellt dabei einen Branch („Ast“) des Repositorys dar.
* **Commit**: Commit ist ein Vorgang, wenn eine neue Version eines Branches eingereicht wird. Das heißt, nachdem der Entwickler einen Vorgang an einer Software abgeschlossen hat, „committed“ er die Änderungen.
* **Pull beziehungsweise Pull-Request**: Hat ein Entwickler einen Bug gefixt oder eine neue Funktion eingebaut, will er, dass seine Änderung in das ursprüngliche Projekt einfließt. Deswegen stellt er einen Pull-Request an den Administrator des jeweiligen Projekts. Dieser kann sich die Änderungen dann ansehen und entscheiden, ob er den Pull durchführt oder nicht.
* **Fork**: Da alle öffentlichen Git-Projekte unter Open-Source-Lizenz stehen, kann auch jeder einen eigenen Ableger eines Repositorys, einen sogenannten Fork, starten. Dort kann jeder privat vor sich hin entwickeln und seine Version am Ende wieder dem ursprünglichen Projekt zuführen (Pull-Request stellen) – oder aber einfach eine eigenständige Variante verbreiten.

**Hub**

Im zweiten Teil des Namens – Hub – steckt dann noch der Hinweis auf die Web-Fähigkeit von GitHub. GitHub konnte man auch komplett ohne ein Web-Interface verwenden. Heute kann man dank grafischer Darstellung im Browser den Entwicklungsprozess von Software-Projekten anschaulich darstellen.

Nun übergebe ich das Wort an meinen Kollegen, der euch nun praktisch ein paar wichtige Dinge zeigt.