**VERSIONIERUNG, VERWALTUNG UND VERSIONSKONTROLLE**

Wir verwenden GitHub zur Versionierung und Versionskontrolle für unser Software-Projekt. GitHub ist eine zentrale Anlaufstelle für alle Änderungen eines solchen Projektes, welches durch Automatisierung jede Änderung in das fertige Programm implementiert. Das große Vorteil an GitHub ist, dass es jede neue Änderung abspeichert. Durch diese Versionierung ist es möglich auf eine ältere Version zuzugreifen, im Falle eines aufgetretenen Fehlers. GitHub wird ebenfalls von vielen anderen Unternehmen verwendet, wie zum Beispiel Airbnb, reddit, Starbucks oder Yahoo.

**Wichtige Befehle in Github**

Die wichtigsten Befehl in GitHub, um diese Software grundlegend zu verstehen sind Reprository, Branch, Commit, Push Origin, Pull origin, Fetch origin.

*Reprository:* Dies ist ein verwaltetes Verzeichnis zur Speicherung von Code und anderen Objekten in einem Projekt in unserem Fall heißt dieses Reprository „Move“ (Abbildung 5.1)

*Branch:* Innerhalb eines Reprository kann es verschiedene Branches geben, man bezeichnet diese als seperate Arbeitszweige. Diese Branches können verschieden weit entwickelt sein. (Abbildung 5.2)

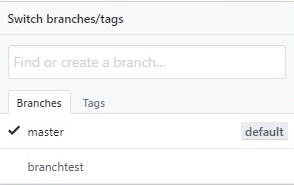
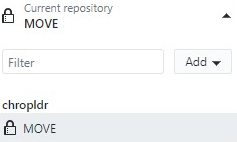


Abbildung 5.1 Abbildung 5.2

*Commit:* Mit dem Commit bestätigt man seine Änderungen, welche im Reprository gespeichert werden. Dieser Änderungen haben einen Autor, Datum und Uhrzeit, und eine Nachricht, der sogenannten „commit message“. (Abbildung 5.3).

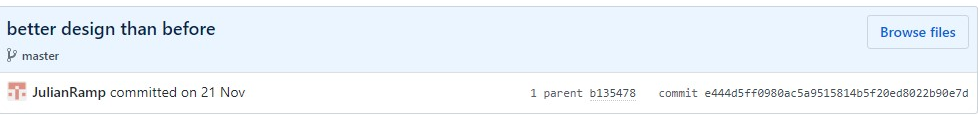


Abbildung 5.3

*Push origin:* Mit Push origin werden die lokalen Änderungen auf den Server übertragen.

*Pull origin:* Mit Pull origin aktualisiert man sein lokales GitHub-Verzeichnis und lädt sich alle Änderungen herunter.

*Fetch origin:* Durch Fetch origin wird es dem Benutzer ermöglicht, zu sehen ob Änderungen am GitHub Server vorzufinden wären.

**Benutzung von Github**

Um Github verwenden zu können muss man sich zunächst auf <https://github.com/> anmelden. Danach kann man entweder auf der Webseite selbst ein Reprository erstellen oder in der GitHub Desktop Applikation, welche wir verwendet haben (Abbildung 5.4). Man benötigt einen Namen des Reprository, optional eine Beschreibung und einen lokalen Pfad, auf dem der ganze Inhalt heruntergeladen wird bzw. „geclont“ wird. Von diesem Ordner aus greift GitHub zu und ist daher für alle zukünftigen Prozesse in GitHub wichtig.



Abbildung 5.4

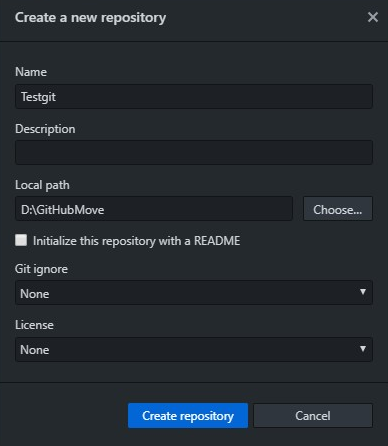


Abbildung 5.5

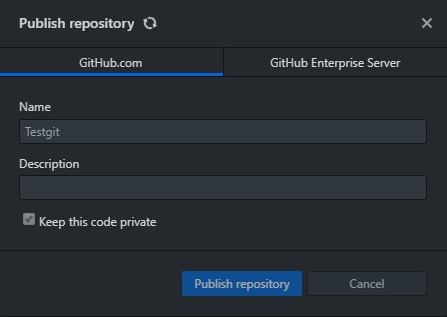
Im weiteren Verlauf muss man noch das neue Reprository veröffentlichen, damit es auch öffentlich sichtbar ist (Abbildung 5.5).

Abbildung 5.5

Falls man etwas Neues in dieses Reprository hochladen möchte, muss man die Dateien in das lokale Verzeichnis des Computers verschieben, in meinem Fall ist es der Ordner „Testgit“, welcher auf der D: Festplatte hinterlegt ist (Abbildung 5.6).



Abbildung 5.6

Nachdem man dies erledigt hat, hat man die Möglichkeit die lokalen Änderungen zu speichern. Ebenfalls besteht die Möglichkeit noch seine Änderungen zu kommentieren, damit sich Teammitglieder oder man sich selbst besser auskennt. Dateien, welche man neu hinzugefügt hat sind mit einem grünen „+“ gekennzeichnet (Abbildung 5.7). Anschließend „commited“ man seine Änderungen (Abbildung 5.8).

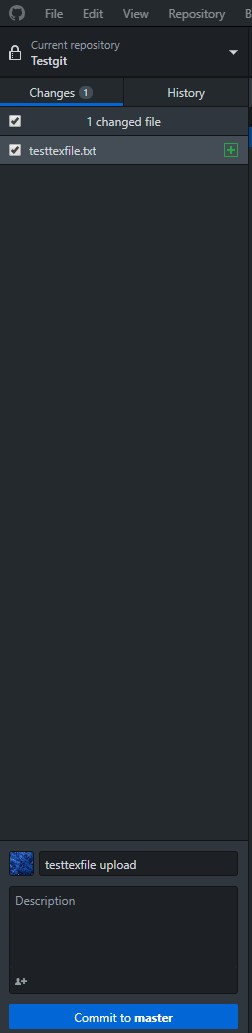
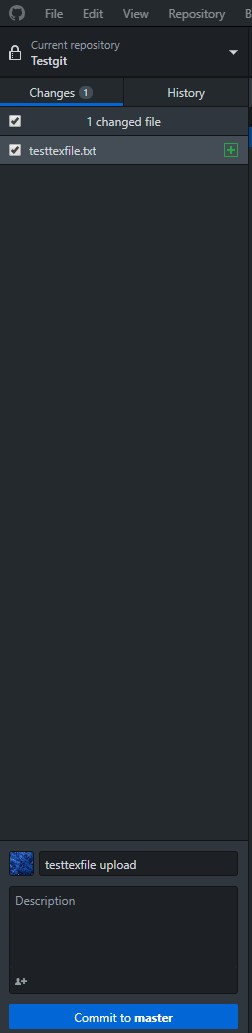


Abbildung 5.7

Abbildung 5.8

Nun sind die Änderungen „commited“, jedoch müssen wir diese noch auf den GitHub Server sichern, dies kann man mit „push origin“ erledigen (Abbildung 5.9).

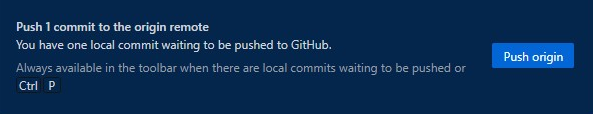


Abbildung 5.9

Bei Änderungen von anderen Teammitgliedern, gibt es die Option „Fetch origin“ (Abbildung 6.0), um potenzielle Aktualisierung anzuzeigen. Falls welche vorhanden sind kann man durch den Button „Pull origin“ die neuersten Änderungen herunterladen auf sein lokales Reprository (Abbildung 6.1).

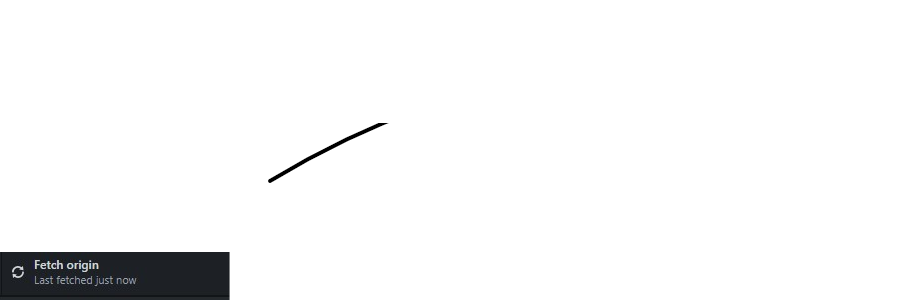


Abbildung 6.0



Abbildung 6.1

Im Falle eines Fehlers, besteht die Möglichkeit auf eine ältere Version der Datei oder Ordners zurückzugreifen, indem man in der „History“ des jeweiligen Reprository nachschaut und die Änderung auswählt, auf welche man zurückspringen will. Mit dem Befehl „Revert this commit“ ist dies sicherzustellen (Abbildung 6.2).

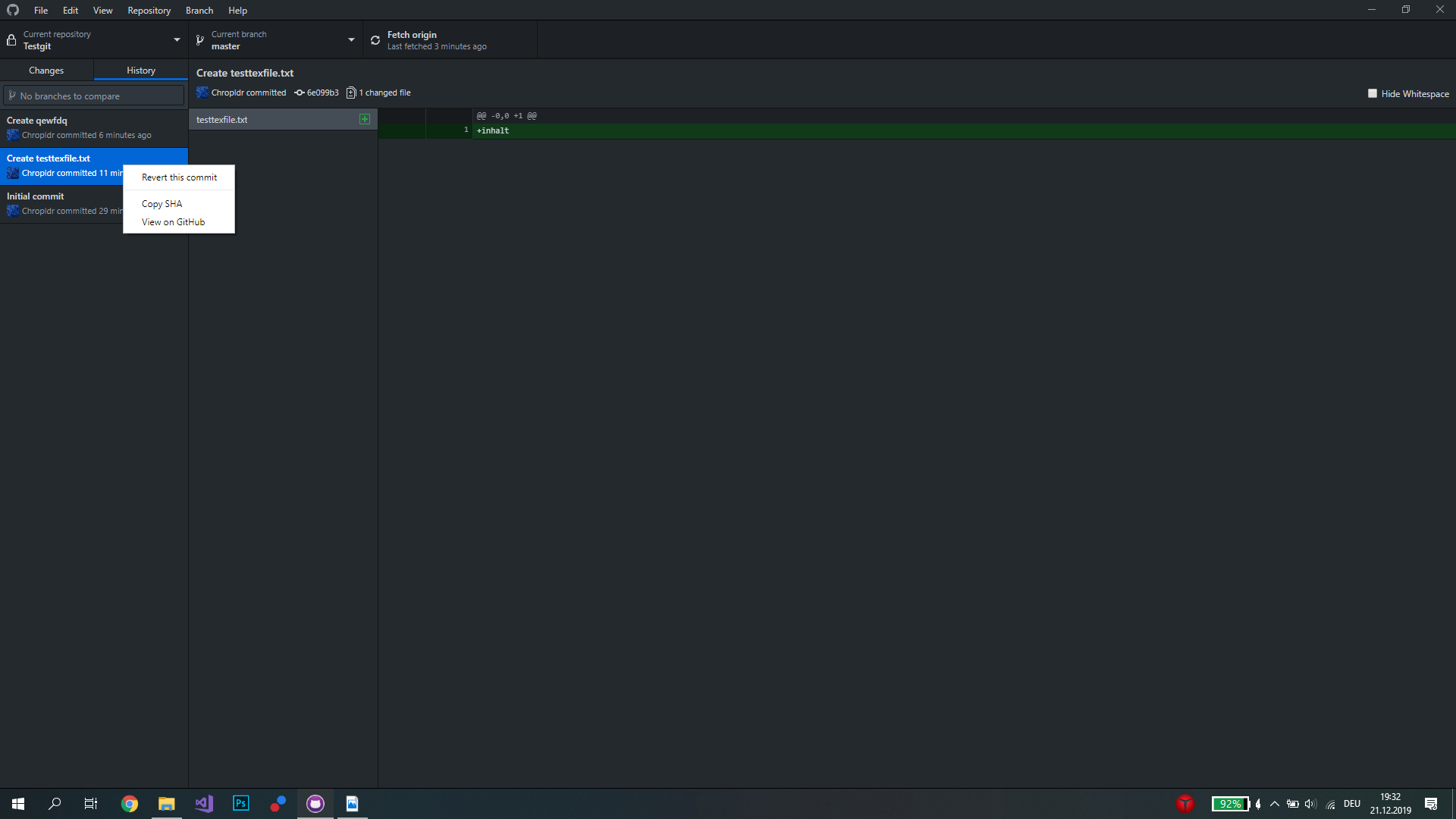


Abbildung 6.2

**Alternativen zu GitHub**

Es gibt sehr viel Alternativen zu GitHub zum Beispiel SourceForge oder SVN, welche bei uns auch in die engere Auswahl kamen. Jedoch haben wir uns für GitHub entschieden, da es unserer Meinung leichter verständlich und einfach zu bedienen ist und momentan von immer mehr Unternehmen benutzt wird, dessen Trend steigend ist.



*Abbildung 6.3*  *Abbildung 6.4*

Sourceforge ist ein Filhosting-Dienst Svn ist eine freie Software zur

für Softwareprojekte. zur zentralen Versionsverwaltung

von Verzeichnissen und Dateien.