GIT, GITLAB, GITHUB, ETC.

ABTEILUNG TGM

Um während der Programmierung gegen Datenverlust gefeit zu sein, und um mit mehreren Programmierern an einem gemeinsamen Projekt zu kollaborieren, verwenden wir die Versionskontrollsoftware Git. Git kann auf https://git-scm.com/ heruntergeladen werden. Bei der Installation muss auf Windows ein Texteditor angegeben werden. Im Zweifel ist dort Notepad++ eine gute Wahl, und muss dann von https://notepad-plus-plus.org/ heruntergeladen und installiert werden.

In der Abteilung Technik und Gesundheit für Menschen wurde ein zentraler Git-Server unter https://tgm-git.jade-hs.de/ eingerichtet. Alternativ kann auch Github oder Gitlab verwendet werden. Mit einem solchen Git-Server kann Programmcode online veröffentlicht werden, und unter den Projektteilnehmern während der Projektarbeit konfliktarm ausgetauscht werden.

GitLab-Server der Abteilung

Die GitLab-Instanz der Abteilung TGM ist unter folgender URL aufrufbar. Der Login erfolgt (wie im Wiki) via Shibboleth. Bei der ersten Anmeldung wird automatisch ein Benutzer-Account in GitLab erzeugt.

 tgm-git.jade-hs.de

Nutzungsbedingungen

Zusätzlich zu den folgenden Informationen gelten die rechtlichen Nutzungsbedingungen zum GitLab denen jede_r Nutzer_in vor der Nutzung der Webseite zustimmen muss. Sie sind jederzeit unter https://tgm-git.jade-hs.de/-/users/terms einsehbar und können dort auch nachträglich noch abgelehnt werden, woraufhin eine automatische Abmeldung (Log-Out) erfolgt und die Frist zur Account-Löschung in 60 Tagen beginnt.

Grundsätzlich darf der Account im GitLab sowohl für studentische als auch private Projekte genutzt werden. Entscheidend ist die Gesamtgröße eines Repositorys (darf 1 GiB nicht überschreiten) und, dass du nur auf Anfrage und in Rücksprache mit einer verantwortlichen Person der Lehre ein Repository auch öffentlich verfügbar machen kannst. Das Standardlevel der Sichtbarkeit ist „Internal“, das bedeutet: jeder mit einem GitLab-Login kann das Repository sehen.

~~Nach Exmatrikulation steht dir der GitLab-Account noch bis zu einem Jahr lang zu Verfügung, um bestehende Repositorys über den SSH-Zugang zu clonen, etc.~~ Der Zugriff auf GitLab steht dir zu Verfügung, bis das Rechenzentrum deinen HRZ-Login abschaltet. Über SSH kannst du dann noch

maximal 75 Tage lang bestehende Projekte clonen, um sie zu sichern. Nach 75 Tagen ohne Anmeldung über das Web-Interface wird schließlich auch dein GitLab-Account gelöscht und Repositorys, die du (für dich allein) angelegt hast, werden unwiederbringlich gelöscht. Projekte in GitLab-Gruppen sind davon nicht betroffen – unabhängig davon, ob du sie erstellt oder Commits beigetragen hast. Mit der Nutzung des GitLab erklärst du dich damit einverstanden, dass Commits zu nichtprivaten Projekten funktionsbedingt nicht wieder entfernt werden können und du auf unbestimmte Zeit mit deinen Commits, deinem Namen und der E-Mail-Adresse des Commit-Authors in der Versionshistorie zu finden sein wirst.

HRZ-E-Mail-Adresse erforderlich

Um sicherzugehen, dass du im vorgenannten Fall nicht deine private E-Mail-Adresse preisgibst, ist es erforderlich, beim Commit-Author die vom HRZ vergebene E-Mail-Adresse zu verwenden. Damit ist Einstellung gemeint, die gleich zu Beginn deiner „Git-Karriere“ vorgenommen hast; wahrscheinlich mittels dieser Befehle:

```
$ git config --global user.name "Max Mustardstudi"
$ git config --global user.email "max.mustardstudi@student.jade-hs.de"
```

wobei `user.email` die E-Mail-Adresse des Commit-Authors ist – beide Angaben sind für jeden Commit für alle sichtbar, die Zugriff auf ein Git-Repository haben und lassen sich im Nachhinein nur durch Neuschreiben der Versionshistorie ändern.

Das Bereinigen der Commit-History – auch nur eines einzigen Commits – wirkt sich auf **alle** Commits aus, die seit dem ersten bereinigten Commit erstellt wurden und bereinigt **nicht** die Clones bei anderen Nutzer_innen. Mehr zum Entfernen sensibler Informationen aus Git.

Um die E-Mail-Adresse des Commit-Authors nur für zukünftige Commits anzupassen, verwende einfach den obigen Befehl erneut. Er überschreibt die bestehende Einstellung. Alternative kannst du das `–global`-Flag weglassen und kannst somit die Einstellung das aktuelle Git-Repository (in dem der Befehl ausgeführt wird) ändern:

```
$ git config user.email "max.mustardstudi@student.jade-hs.de"
```

Für vergangene Commits und eine Repository-spezifische Konfiguration gibt es verschiedene Wege, die hinterlegte E-Mail-Adresse auf Repository- oder globale Ebene anzupassen. Solltest du explizit die Verwendung einer privaten E-Mail-Adresse bevorzugen, kannst du diese deinem GitLab-Profil hinzufügen. Dies macht sie auch für Push-Vorgänge verfügbar.

Cheatsheets

Gerade in den ersten Wochen mit Git fällt es mitunter schwer, alle nötigen Befehle im Kopf zu behalten. Manche wechseln spätestens jetzt zu einer grafischen Git-Oberfläche (GUI), verhindern

damit aber, dass die zugrundeliegenden Befehle und Paradigmen verinnerlicht werden – vieles davon wird von den GUIs hinter Vereinfachungen versteckt. Lass deshalb besser erstmal die Finger von einer GUI und versuche, die Konzepte von Git mit den dazugehörigen Befehlen zu verknüpfen.

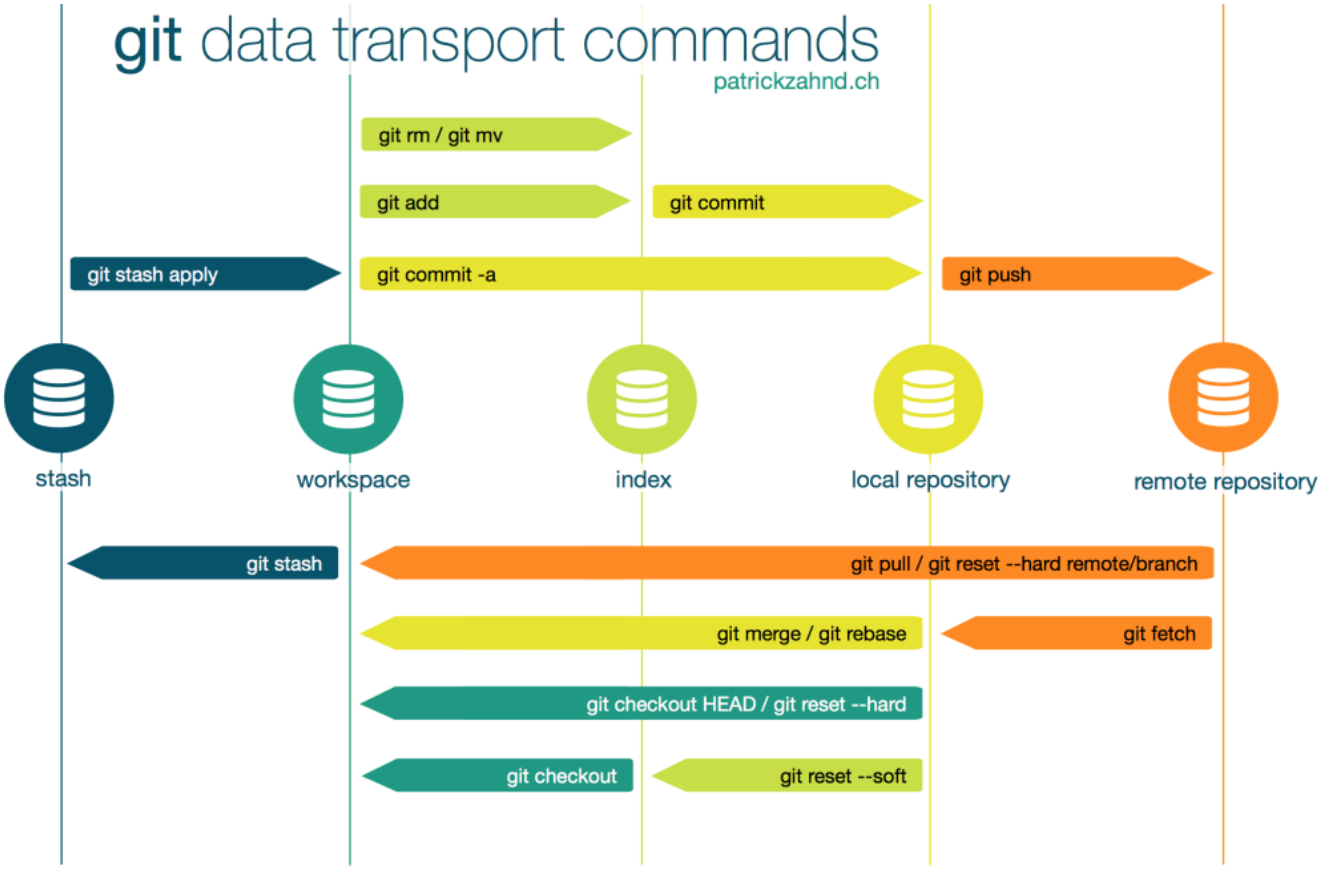
Wenn du doch mal ein Kommando vergessen hast, hilft vielleicht eines der folgenden Cheatsheets weiter.

- Cheatsheet (englisch) von Jan Krüger
- Cheatsheet (deutsch) von Git Tower
- Cheatsheet (englisch) nach Zack Rusin

Git visualisieren

Zusammenhänge von "Arealen" von Git-Daten

Die folgende Abbildung zeigt die möglichen „Arealen“ (Areas) die Daten innerhalb von Git innehaben können. Obwohl sie in der Abbildung eine Art Zeitstrahl bilden, gibt es zunächst keinen zeitlichen Bezug. Die Pfeile zeigen auf, welche Befehle ausgeführt werden können um Daten zwischen den Areas zu bewegen.



–Quelle: Patrick Zahnds Git data transport commands

Ähnlich aber umfangreicher ist diese interaktive Visualisierung. Durch Klicks in die Areale lassen sich die möglichen Befehle einblenden.

Experimente mit den Commit Graph

Dank der Git School von Github gibt es ein wunderbares Webtool, um den Commit Graph zu visualiseren. In der virtuellen Konsole links lassen sich viele Git-Befehle ausführen und ihre Auswirkungen auf den Commit Graph rechts beobachten.

Visualizing Git: https://git-school.github.io/visualizing-git/