Levels zur Veröffentlichung

[1] <https://git-scm.com/book/en/v2/Git-Tools-Interactive-Staging> 29.10.2018

[2] <https://dev.to/neshaz/how-to-git-stash-your-work-the-correct-way-cna> 29.10.2019

[3] <https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning> 29.10.2019

Stage

Ist die lokale Arbeitskopie vorhanden, können nun Änderungen und neue Elemente hinzugefügt werden. Mit dem Befehl ***git status*** werden die derzeitigen Änderungen angezeigt, zu sehen in ABBILDUNG. Unter dem derzeit ausgecheckten Pfad findet man Dateien, welche sich zum letzten Stand des Pfades verändert haben, sowie neue Dateien, welche noch nicht versionsverwaltet sind. Diese Änderungen können nun gestaged werden, dazu wird der Befehl ***git add*** benutzt. Hier können explizit Dateien ausgewählt werden, welche für den nächsten Commit vorgemerkt werden. Will man alle Veränderungen übernehmen reicht ein einfaches ***git add \**** . Der Benutzer kann mit dem Stage Level gezielt auswählen, welche Teile versionsverwaltet werden sollen. Dadurch ermöglicht man kleine und übersichtliche Commits, welche nur Änderungen beinhalten, welche im Repository oder im speziellen Commit benötigt werden. Man stelle sich eine größere Änderung vor, welche mehrere kleinere Module, unabhängig voneinander, beinhaltet. Anstatt die Änderungen als Gesamtpaket zu comitten, kann man die Module einzeln Stagen und committen. Um differenzierter, nicht nur Dateien sondern auch einzelne Abschnitte oder Zeilen zu stagen kann das *add* Kommando noch erweitert werden. Eine Möglichkeit ist ***git add –interactive*** . Hier werden nun mehrere Commandos abgefragt, welche einzeln, nacheinander auf Dateien angewendet werden können. Mit dem Kommando *patch* kann nun eine Datei weiter aufgesplittet werden, um einzelne Zeilen der Stage Area hinzuzufügen [1]. ABBILDUNG???

In GitKraken kann im File View über die GUI einzelne Abschnitte oder Zeilen einer Datei gestaged werden.

ABBILDUNG??

Für dauerhafte oder wiederkehrende nicht versionsverwaltete Dateien in der Arbeitskopie ist es hilfreich eine ***.gitignore*** Datei anzulegen.

.Gitignore

Benutzt man Dateien, welche explizit nicht in die Versionsverwaltung inkludiert werden sollen, kann ***.gitignore*** verwenden. Dateien und Ordner, welche in *.gitignore* zeilenweise vorkommen, werden automatisch in ***git status*** ausgeblendet und somit nicht gestaged bzw. dem nächsten Commit hinzugefügt. In ABBILDUNG zu sehen, werden *object-Dateien* sowie das gebaute Programm ausgeblendet. wird unter anderem häufig für Dateien benötigt, welche im Buildprozess verwendet oder erstellt werden, da diese jeder Entwickler lokal erstellen kann, somit die Versionsverwaltung nicht unnötig vergrößert wird und mehr Übersichtlichkeit erhält. In GitKraken kann eine Datei oder Ordner iin der Änderungsansicht durch ein Rechtsklick ignoriert werden, beziehungsweise *.gitignore* hinzugefügt werden, siehe ABBILDUNG.

Stash

Der Stash stell eine Art Zwischenspeicher da. Da die Arbeitskopie bei einem Pfadwechsel überschreiben wird, sollten alle Änderung committet werden, da sie sonst überschreiben werden. Falls man nun den Pfad wechseln will, jedoch Änderungen besitzt, welche erst zu einem späteren Zeitpunkt fertig werden, kann man diese im Stash bis zur Rückkehr zum Pfad ablegen. Auch können darin Ideen oder Ansätze gespeichert werden, um sie nicht zu verlieren. Da ein Stash auf jeglichen Pfad angewendet werden kann, wäre eine weitere mögliche Anwendung darin Änderungen abzulegen, welche das Projekt schnell und temporär verändern, zum Beispiel ein spezielles Debugging Feature. Der einfachste Grund ist die Arbeitskopie zu reinigen, vorheriges aber nicht zwangsweise zu verlieren, falls man darauf zurückkommen sollte. Der Befehl ***git stash*** speichert alle Veränderungen zum Stand des letzten Commits des ausgecheckten Pfads. Mit der Option *–patch* kann wiederum nur ein Teil zwischengespeichert werden, mit *-u* werden nicht versionsverwaltete Dateien mit einbezogen. Um eine Übersicht der Zwischenspeicherungen zu erhalten, listet der Befehl ***git stash list*** alle Gespeicherten auf. Um den Überblick zu behalten kann dem Stash eine Nachricht hinzugefügt werden, dafür wird *save <message>* benötigt. [2] In ABBILDUNG wird die lokale Arbeitskopie gereinigt und alle Änderungen, inklusive der nicht verfolgten Dateien gestashed mit dem Namen "Feature B on master".   
Um den Stash wieder auf die aktuelle Arbeitsmappe anzuwenden wird der Befehl ***git stash pop*** oder ***git stash apply*** verwendet. Im ersten Fall wird der Stash nach der Anwendung gelöscht, bei *apply* bleibt er erhalten. Mit ***git stash show*** werden die Änderungen zum letzten Stash angezeigt. Um einen Stash zu löschen kann der Befehl ***git stash drop <stash-id>*** , wobei die ID zum Beispiel *stash@{2] ist.*

*[3]*

Commit

Push