Git Cheatsheet

Git-Grundlagen

`git init` – *Erstellt ein leeres Repository im aktuellen Verzeichnis*

`git clone <url> [<path>]` – *Klont ein entferntes Repository ins aktuelle oder definierte Verzeichnis*

`git status` – *Zeigt, welche Dateien hinzugefügt oder committet werden müssen*

`git log [-n <limit>]` – *Zeigt die Commit-History an (Optionale Begrenzung der anzuzeigenden Commits durch –n <limit>*

`git config [--global] user.name <name>` – *Legt den Benutzernamen fest*

`git config [--global] user.email <email>` – *Legt die E-Mail Adresse des Benutzers fest*

Dateioperationen

`git add <file/path>` – *Fügt die ausgewählte Datei oder das ausgewählte Verzeichnis zur Staging Area hinzu*

`git commit` – *Öffnet einen Editor zur Eingabe der Commit-Message, mit der alle Dateien aus der Staging Area comittet werden*

`git commit –m „<message>“` – *Commitet gestagte Dateien ohne Öffnung des Texteditor und verwendet <message> als Commit-Message*

`git checkout <commit> <file>` – *Stellt eine Datei aus einem vergangenen Commit im Arbeitsverzeichnis wieder her und fügt sie zur Staging Area hinzu*

`git checkout <commit>` – *Stellt einen kompletten Commit wieder her*

`git clean –f [<path>]` – *Löscht alle modifizierten ungestagten Dateien (aus einem Verzeichnis)*

`git revert <commit>` – *Stellt einen vergangenen Commit wieder her, indem er als neuer Commit zur Versionsgeschichte hinzugefügt wird*

`git reset` – *Setzt Staging Area auf den Stand des aktuellsten Commits, ohne das Arbeitsverzeichnis zu verändern*

Branches

`git branch` – *Listet alle Branches im Repository auf*

`git branch <branch>` – *Kopiert den aktuellen Branch und speichert ihn als <branch>*

`git branch –d <branch>` – *Löscht <branch>, sofern er keine ungesicherten Änderungen enthält*

`git branch –m <branch>` – *Benennt den aktuellen Branch um zu <branch>*

`git checkout <branch>` – *Wechselt zum bereits existierenden Branch <branch>*

`git checkout –b <branch>` – *Kopiert den aktuellen Branch, speichert ihn als „<branch>“ und wechselt zu ihm*

`git merge <branch>` – *<branch> wird mit dem aktuellen Branch zusammengeführt*

Remote-Repositorys

`git fetch <remote> [<branch>]` – *Lädt eine Kopie von einem Remote-Repository herunter, allerdings ohne sie selbstständig zu mergen*

`git pull <remote>` – *Lädt eine Kopie von einem Remote-Repository herunter und mergt sie mit der lokalen Kopie*

`git push <remote> <branch>` – *Lädt den angegebenen Branch auf das Remote-Repository hoch*

Notiz: `<arg>` ist eine notwendige, `[<arg>]` eine optionale Variable.