## SETUP FÜR ALIAS-CHEATSHEET

```
$ source <(curl -s https://raw.githubusercontent.com/tklepzig/git-init/master/setup.sh) -u
```

Führt das Setup-Script aus:

> **Ohne** den Parameter **-u** wird die grundlegende Einrichtung von Git und Anlegen der Aliase (alle Einstellungen können natürlich später auch geändert werden) vorgenommen.
>
> **Mit** dem Parameter **-u** wird das Setup-Script im Update-Modus ausgeführt, sodass NUR die Aliase aktualisiert werden.

## LOKALE ÄNDERUNGEN

#### Status anzeigen

```
$ git s
```

Ausstehende Änderungen kurz und knapp anzeigen

#### Änderungen anzeigen

```
$ git dt
```

Öffnet das konfigurierte Diff Tool und zeigt alle Änderungen der Dateien an, die sich **nicht** in der Staging Area befinden

```
$ git dts
```

Öffnet das konfigurierte Diff Tool und zeigt alle Änderungen der Dateien an, die sich in der Staging Area befinden

```
$ git d
```

Zeigt alle Änderungen der Dateien an, die sich **nicht** in der Staging Area befinden

```
$ git ds
```

Zeigt alle Änderungen der Dateien an, die sich in der Staging Area befinden

#### Änderungen hinzufügen

```
$ git a [*Pfad*]
```

Alle Dateien oder nur die unterhalb des angegeben Pfades (optional) stagen (gesamtes Repository)

```
$ git ap [*Pfad*]
```

Alle Dateien oder nur die unterhalb des angegeben Pfades (optional) stagen, jedoch die Teile (Hunks) pro Datei separat wählen (gesamtes Repository)

```
$ git cm *Commit message*
```

Alle in der Staging Area befindlichen Dateien committen

```
$ git cma *Commit message*
```

Letzten Commit bearbeiten

```
$ git acm *Commit message*
```

Alle Dateien stagen und sofort committen (gesamtes Repository)

```
$ git acmp *Commit message*
```

Alle Dateien stagen, committen und pushen (gesamtes Repository)

#### Dateien aus dem nächsten Commit nehmen (Unstaging)

```
$ git r  [*Pfad*]
```

Alle Dateien oder nur die unterhalb des angegeben Pfades (optional) aus der Staging Area nehmen

#### History anzeigen

```
$ git l
```

Übersichtlich die komplette History anzeigen

```
$ git ls
```

Wie git l, zeigt aber nur die letzten 10 Commits an

```
$ git lm
```

Wie git l, listet aber nur Merge-Commits auf

#### Änderungen rückgängig machen

```
$ git rh [*Pfad*]
```

Alle Dateien oder nur die unterhalb des angegeben Pfades (optional) auf den Stand des letzten Commits zurücksetzen (betrifft keine neu erstellten Dateien)

```
$ git undo
```

Alle Änderungen (betrifft auch neu erstellte Dateien) auf den Stand des letzten Commits zurücksetzen

## UPDATE & PUBLISH

#### Änderungen von einem Remote Repository holen

```
$ git f
```

Alle Änderungen des Remote Repository in das lokale Repository laden, dabei wird aber das Arbeitsverzeichnis nicht geändert

```
$ git frb
```

Alle Änderungen des Remote Repository in das Arbeitsverzeichnis laden (unter Verwendung eines Rebase)

```
$ git p origin:*Branchname*
```

Löscht den angegebenen *remote* Branch *Branchname*

## BRANCHING & MERGING

#### Branch-Verwaltung

```
$ git b *Branchname*
```

Branch mit dem Namen *Branchname* anlegen

```
$ git c *Branchname*
```

Zum Branch *Branchname* wechseln

```
$ git c -
```

Zwischen den zuletzt genutzten Branches hin und her wechseln

```
$ git b -d *Branchname*
```

Aktuellen bzw. übergebenen Branch *Branchname* löschen (nur lokal)

```
$ git b -a
```

Listet alle lokalen und remote Branches auf

#### Merging

```
$ git m *Branch-Name*
```

Angegebenen Branch in den aktuellen Branch mergen

```
$ git mff *Branch-Name*
```

Angegebenen Branch in den aktuellen Branch mergen, wenn Fast-Forward möglich ist

```
$ git mr *Branch-Name*
```

Angegebenen Branch in den aktuellen Branch mergen, dabei immer einen Merge-Commit erzeugen

```
$ git mt
```

Öffnet das konfigurierte Standard Mergetool, um die Merge Konflikte aufzulösen.

#### Rebase

```
$ git rb
```

Commit History neu schreiben, v.a. hilfreich, wenn beim fetch & merge eines Remote Branches kein Fast Forward möglich ist

```
$ git rbc
```

Bspw. nach dem Auflösen von Konflikten mit den Rebase Prozess fortfahren