Git Submodule Cheatsheet

Ein git-submodule ist, unterm Strich, ein Verweis auf ein anderes git-repository. Es ist ein hervorragender Weg, Code eines anderen Herstellers oder auch aus einem eigenen bestehenden Respository (wie plugins oder themes) einzubinden.

Dieser Beitrag enthält einige Beispiele zur effizienten Verwendung von git Submodulen.

Hinzufügen eines Submodules

Sie müssen die Url des Remote-Git-Repositorys kennen und wissen, wo Sie sie in Ihrem Repository platzieren möchten. Unter Umständen kann z.B. ein Unterordner wie `submodules` sinnvoll sein

```
git submodule add https://example.com/submodule-repo.git path/to/submodule
git commit -m "adds submodule path/to/submodule"
```

Branchs eines Submodules

Will man einen speziellen Branch eines Submodules hinzufügen, kann man diesen direkt im Command angeben:

```
git submodule add -b branch_name https://example.com/submodule-repo.git
path/to/submodule
```

Will man den Branch im Nachhinein ändern, hat man Extra-Schritte. Zuerst muss man die Datei `.gitmodules` in seinem Repo bearbeiten und das Submodule suchen. Der Eintrag sieht in etwa wie folgt aus:

```
[submodule "path/to/submodule"]
    path = path/to/submodule
    url = https://example.com/submodule-repo.git
```

Diesen kann man jetzt abändern zu:

```
[submodule "path/to/submodule"]
    path = path/to/submodule
    url = https://example.com/submodule-repo.git
    branch = branch_name
```

Man kann die Änderung auch über einen Command ausführen:

```
git config -f .gitmodules submodule.path/to/submodule.branch stable
```

Anschließend muss man die Änderungen synchronisieren:

```
git submodule sync
git submodule update --init
```

Zu guter Letzt muss man die Submodul-Änderungen noch übermitteln:

```
git add path/to/submodule
git add .gitmodules
git commit -m "change path/to/submodule branch to branch_name"
git push
```

Klonen eines Projektes mit Submodulen

Wenn man ein Projekt mit Submodulen klonen will, hat man einen zusätzlichen Schritt auszuführen:

```
git clone http://example.com/repo.git repo
cd repo
git submodule update --init
```

Wenn man Submodule inklusive deren Submodulen benötigt, kann man diese auch Rekursiv auschecken:

```
git clone --recurse-submodules http://example.com/repo.git repo
```

Aktualisieren der Submodule

Remote Aktualisierung der Submodule

Wenn man einfach den Master-Zweig für das Submodul nachverfolgt, kann man mit einem einfachen Abrufen und Zusammenführen auskommen:

```
cd path/to/submodule
git fetch
git merge origin/master
```

Etwas schneller kann man das auch durchführen, solange man weiß, dass es keine lokalen Änderungen im Submodule gibt:

```
cd repo
git submodule update --remote
```

Hat man die Submodule rekursiv abgerufen, kann man sie auch rekursiv aktualisieren:

```
git submodule update --remote --recursive
```

Diese Methode ist aber nicht immer empfehlenswert, weil die rekursiven Sub-Submodule alle auf die neueste Version ausgecheckt werden, nicht unbedingt auf die Version des Submodule Repos. Will man diese behalten, ist der sicherere Weg die Submodule einzeln zu aktualisieren:

```
git submodule update --remote
cd path/to/submodule
git submodule update
```

Anschließend muss man die aktualisierten Submodule noch in seinem eigenen Repo übermitteln, damit andere auch die neue Version nutzen:

```
git add path/to/submodule
git commit -m "update path/to/submodule"
git push
```

Anschließend können andere wie im nachfolgenden Abschnitt gezeigt, die Änderungen einfach übernehmen:

Aktualisieren der Submodule auf eine neuere Repo-Version

Will man seine lokale Version auf die Version des eigenen Repos aktualisieren, kann man ein einfaches update ausführen:

```
cd repo
git submodule update
```

Damit wird die neue Version der Submodule geholt. Das geht natürlich auch rekursiv:

```
cd repo
git submodule update --recursive
```

In diesem Fall gibt es keine Änderungen zu übermitteln, weil man ausschließlich auf die neuere Submodule-Version des eigenen Repos aktualisiert.

Remote Host des Submodules verändern

Man hat zwei Möglichkeiten. In der ersten bearbeitet man die Datei `.gitmodules` im Repo und ändert dort die URL's des Submodules ab:

```
[submodule "lib/module-a"]
    path = lib/module-a
    url = git@host.de:path/to/module-a.git #voriger host
    branch = dev
```

```
[submodule "lib/module-a"]
    path = lib/module-a
    url = git@host.com:path/to/module-a.git # neuer host
    branch = dev
```

Anschließend muss man die Submodule in git synchronisieren:

```
git submodule sync
```

Möglichkeit 2 läuft komplett über die Kommandozeile. Gehen wir, wie im vorigen Beispiel, von folgender Konfiguration aus:

```
[submodule "lib/module-a"]
    path = lib/module-a
    url = git@host.de:path/to/module-a.git #voriger host
    branch = dev
```

```
git config --file=.gitmodules submodule.lib/module-a.url
git@host.com:path/to/module-a.git # neuer host
git config --file=.gitmodules submodule.lib/module-a.branch Development # Optional
den Branch wechseln
git submodule sync # (re-)sync
git submodule update --init --recursive --remote # (re)init und Aktualisierung
```

Will man sich die Pfade/Bezeichnungen der vorhandenen Submodule anschauen, kann man das über

```
git config --file=.gitmodules -l
```

Der Command erzeugt eine Ausgabe wie

```
submodule.lib/module-a.path=lib/module-a
submodule.lib/module-a.url=git@host.de:path/to/module-a.git
submodule.lib/module-a.branch=dev
submodule.lib/module-b.path=lib/module-b
submodule.lib/module-b.url=git@host.de:path/to/module-b.git
submodule.lib/module-b.branch=master
submodule.lib/module-c.path=lib/module-c
submodule.lib/module-c.url=git@host.de:path/to/module-c.git
submodule.lib/module-c.branch=master
```

Entfernen eines submodules

Das Entfernen einens Submodules erfolgt in zwei Schritten:

1. man entfernt die Referenz
2. man entfernt die Lokale Version im Cache

```
git submodule deinit path/to/submodule
git rm path/to/submodule
git commit -m "remove submodule path/to/submodule"
```

```
rm -rf .git/modules/path/to/submodule
```

Den Submodule Status in `git status` anzeigen

Man kann git so konfigurieren, dass eine Submodul-Zusammenfassung angezeigt wird, wenn man einen Git-Status ausstellt. Es gibt hier einen kleinen Leistungskompromiss, aber es könnte auch nützlich sein.

```
git config status.submodulesummary 1
```