

GIT Workshop

Randolph Maassen (randolph.maassen@alumni.fh-aachen.de)

23. Oktober 2014

GIT Installieren

```
$ sudo apt-get install git
$ git config user.name "Dein Name"
$ git config user.email deineMailadresse@hoster.com
```

Git lokal verwenden

Um Git lediglich lokal zu verwenden kann einfach mittels

```
$ git init reponame
```

ein neues Repository namens *reponame* erzeugt werden.

Git mit einem Server verwenden

Meistens bekommt man jedoch eine git URL wenn man an einem Projekt mitarbeiten möchte. Man kann sich auch ein Repository bei einem der folgenden Anbieter holen:

- Github.com
- BitBucket.org

Um an einem Projekt mitzuarbeiten klonst man das Repo wie folgt:

```
$ git clone [proto://][user]@[server]:[pfad]
```

Bei einigen URLs wird noch ein Protokoll angegeben, z.B. https:// oder git://. Wird https verwendet ist das Repo nur lesbar geklont, es kann nichts hochgeladen werden.

SSH-Key

Wer schon mal mit SSH gearbeitet, oder den Workshop besucht hat weiß es gibt SSH-Keys, die eine sicherere Verbindung ermöglichen, und auch den User besser Identifizieren als ein Login mittels Passwort. In den meisten Fällen arbeitet git über SSH, daher kann man auch bei git diesen SSH-Key benutzen.

Dabei kann man einen bestehenden Key benutzen oder sich wie folgt einen neuen generieren:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/[yourUser]/.ssh/id_rsa):
/home/[yourUser]/.ssh/git
```

Den Inhalt der dabei erzeugten Datei `/home/[yourUser]/.ssh/git.pub` kann man nun bei dem Hoster hochladen oder dem Administrator des git per mail zukommen lassen.

Man kann SSH (und damit auch git) noch mitteilen, welchen Key es verwenden soll, falls nicht der Standardkey verwendet wird. Dafür ergänzt du die Datei `~/.ssh/config` um folgendes:

```
Host [server]
    HostName [server]
    User [user]
    IdentityFile ~/.ssh/git
```

wobei du server und user aus der URL entnimmst und die IdentityFile mit dem ssh-keygen abgleichst.

Die wichtigsten GIT Befehle

```
$ git commit [-a -m "Sinnvoller commit Kommentar"]
```

Daten von dem Dateisystem in das Repository schreiben

```
$ git pull
```

Daten vom Server anfordern

```
$ git push
```

Repository auf den Server übertragen

```
$ git add FILE
```

Datei/Ordner zu einem Repository hinzufügen

```
$ git mv FROM TO
```

Datei im Dateisystem und im Repository verschieben

```
$ git rm FILE
```

Datei im Dateisystem und im Repository löschen

```
$ git status
```

Gibt den aktuellen Status des Repositories im Vergleich zum Dateisystem aus

```
$ git log
```

Gibt eine Liste aller "Commits" aus

```
$ git diff
```

Zeigt Unterschiede seit dem letzten Commit an

```
$ git reset TO-COMMIT-HASH
```

Setzt das Repository auf eine frühere Version zurück

```
$ git --help
```

```
$ git [kommando] --help
```

Gibt weitere Infos

Dateien ignorieren

Es ist zudem möglich Dateien zu ignorieren indem man folgende Datei anlegt: *.gitignore*

```
# Compiled source #  
#####  
*.com  
*.class  
*.dll  
*.exe  
*.o  
*.so
```

Es können beispielsweise Wildcards verwendet werden um, wie im Beispiel, alle Dateien mit der Endung ".class" zu ignorieren. Zeilen die mit einem Hashtag beginnen sind lediglich Kommentare die von GIT ignoriert werden. Diese Datei sollte mit in das Repository gelegt werden und gilt für alle Unterverzeichnisse.

Literatur

[1] <http://de.wikipedia.org/wiki/Git>

[2] <http://git-scm.com/book>