Kochbuch GitHub

Glen Langer

Published with GitBook

Inhaltsverzeichnis

Kochbuch für GitHub

Das ist (m)ein Kochbuch für GitHub, gesammelte Anleitungen für das eine oder andere Problem.

Die neuste Version ist unter docs.contao.ninja zu finden.

Inhaltsverzeichnis

Licence

Mehrere GitHub-Accounts mit unterschiedlichen SSH-Keys benutzen

Hat man mehrere GitHub-Accounts, so bekommt jeder einen eigenen public SSH-Key. Anders erlaubt es GitHub nicht.

Problem

Beim Befehl `git clone git@github.com:username/repo.git` wird eine SSH Verbindung zu git@github aufgebaut und dabei der erste SSH-Key verwendet den SSH findet und von GitHub akzeptiert wird.

Folgt nun ein Push von Git, kann es zum 'Access Denied' kommen, da nicht der passende SSH-Key verwendet wurde.

Hinweis:

Hier ist von der Arbeit mit der Kommandozeile die Rede. Einige GUIs gestatten es die SSH-Keys je Account zu definieren, sofern diese Multiaccount fähig sind. GitKraken ist das beispielsweise nur in der Pro Version.

Lösung 1

Speziell wenn es einen Hauptaccount gibt und den/die weiteren nur selten verwendet werden bzw. wesendlich weniger Projekte hat.

Ab Git Version 2.10

Eventuell muss zuvor Git über den Service "Personal Package Archive" (kurz PPA) aktualisiert werden, je nach Linux Version. Beispielsweise bringt Ubuntu 16.04 selbst nur Git 2.7.4 mit.

```
sudo add-apt-repository ppa:git-core/ppa
sudo apt-get update
sudo apt-get install git
```

Hauptaccount

Hier wird ein **globaler** SSH-Key `id_rsa_github` definiert für den Hauptaccount:

```
git config --global core.sshCommand 'ssh -i ~/.ssh/id_rsa_github -F /dev/null'
```

Dadurch wird ein passender Eintrag geschrieben in der Datei `~/.gitconfig`. Den SSH-Key Dateiname entsprechend anpassen.

Jetzt kann man damit normal arbeiten.

```
git clone git@github.com:username/repo.git
```

Dieser SSH-Key wird nun für jedes GitHub-Repository verwendet, sofern nicht lokal dieser überstimmt wird, wie im nächsten Kapitel beschrieben.

Weitere Accounts

Hier wird in der **lokalen** Konfiguration jedes GitHub-Repository eines weiteren Accounts ein eigener SSH-Key definiert. Dazu in das lokale Repository wechseln, dann:

```
git config core.sshCommand 'ssh -i ~/.ssh/id_rsa_github_second -F /dev/null'
```

Dadurch wird ein passender Eintrag geschrieben in der Datei `.git/config`. Den SSH-Key Dateiname entsprechend anpassen.

Nun wird speziell für dieses GitHub-Repository der zweite SSH-Key verwendet. Jetzt kann man damit normal arbeiten.

```
git clone git@github.com:username_second/repo.git
```

Mit Git Version <2.10

Hauptaccount

Für den Hauptaccount wird eine Default Einstellung für die SSH Verbindung erstellt, welcher SSH-Key verwendet werden soll beim Aufbau einer Verbindung zu github.com.

In `~/.ssh/config` einfügen:

```
host github.com
 HostName github.com
 IdentityFile ~/.ssh/id_rsa_github
 User git
```

Key-Dateiname entsprechend anpassen. Jetzt kann man damit normal arbeiten.

```
git clone git@github.com:username/repo.git
```

Weitere Accounts

Für jedes GitHub-Repository eines weiteren Accounts wird hier über eine Umgebungsvariable der aktuelle SSH-Key definiert:

```
export GIT_SSH_COMMAND="ssh -i ~/.ssh/id_rsa_github_second -F /dev/null"
```

Key-Dateiname entsprechend anpassen. Jetzt kann man damit normal arbeiten.

```
git clone git@github.com:username_second/repo.git
```

Aber Achtung: Die Umgebungsvariable muss wieder gelöscht werden, falls danach wieder mit dem Hauptaccount gearbeitet werden soll.

Lösung 2

Sind hier zu finden: www.bennyn.de oder hier in englisch www.keybits.net

Hier wird nur mit der `~/.ssh/config` gearbeitet, jedoch mit "Dummy-Hostnamen" und die Repositories müssen nach dem `git clone` noch angepasst werden.