Einführung in Git

Praktikumsaufgaben

Im Praktikum erfolgt eine beispielhafte Einführung in die Versionsverwaltung mit Git. Über einfache Änderungen an einem demonstrativen Projekt wird dabei ein erster Einblick in die Möglichkeiten von Git gegeben.

- Git – offizielle Webseite (http://git-scm.com/)
- Git Documentation Resource – offizielle Dokumentation (http://git-scm.com/documentation)
- Pro GIT – Ebook von Scott Chacon und Ben Straub (https://git-scm.com/book/)

Die Bearbeitung der Aufgaben erfolgt zum Kennenlernen der Befehle in der Konsole unter Linux. Grafische Clients und IDEs mit Git-Integration erleichtern den Umgang mit den Befehlen entsprechend.

Konfiguration

1. Identität festlegen

   Die Konfigurationswerte werden in der Datei `~/.gitconfig` abgelegt und gelten global für alle eigenen Git Repositorys.

   ```
   $ git config --global user.name "<vorname> <nachname>"
   $ git config --global user.email <s00000>@informatik.htw-dresden.de
   ```

   Repository spezifische Konfigurationen befinden sich in der Datei `.git/config` im Arbeitsverzeichnis. Hierfür bei `git config` die Option `--global` weglassen. Darin enthaltene Konfigurationswerte **überschreiben** die eigenen globalen Werte in der Datei `~/.gitconfig`.

2. Anzeigen der Konfiguration

   ```
   $ git config --list
   $ cat ~/.gitconfig
   ```

Aufgabe 1 (geführt)

1. Ein neues Git Repository anlegen

   ```
   $ mkdir demo-hello-world && cd $_

   $ git init
   Initialized empty Git repository in ...
   $ ls -a
   .git
   ```

   - [Arbeitsverz] <—> [Index] <—> [HEAD]

2. Eine C Programm *hello_world.c* im Arbeitsverzeichnis erstellen und testen

   ```
   $ vi hello_world.c
   #include <stdio.h>

   int main(void)
   {
   printf("Hello, World\n");

   return 0;
   }
   ```

   Test:

   ```
   $ clang hello_world.c
   $ ./a.out
   Hello, World
   ```

3. Relevante Änderungen im Arbeitsverzeichnis in den Index aufnehmen und unerwünschte Dateien ausschließen

   Ändrungen im Index aufnehmen:

   ```
   $ git add .
   $ git status
   ```

   Änderungen aus Index entfernen:

   ```
   $ git rm --cached a.out
   ($ git reset a.out)
   $ git status
   ```

   unerwünschte Dateien ausschließen:

```
$ vi .gitignore
*~
a.out

$ git status
```

Ändrungen im Index aufnehmen:

```
$ git add .gitignore
($ git add .)
$ git status
```

4. Änderungen im Index in das Repository (HEAD) aufnehmen

```
$ git commit -m "add hello_world.c and .gitignore"
```

Commit History anzeigen:

```
$ git log
$ git log --oneline
```

5. Änderung im Arbeitsverzeichnis vornehmen, untersuchen und in den Index und das Repository (HEAD) aufnehmen

Quelltext anpassen:

```
$ vi hello_world.c
#include <stdio.h>

int main(void)
{
printf("Hello, World\n");
printf("Hallo\n");

return 0;
}
```

Änderung untersuchen:

```
$ git status
$ git diff hello_world.c
```

Änderung im Index aufnehmen:

```
$ git add .
$ git status
```

Quelltext erneuten anpassen:

```
$ vi hello_world.c
#include <stdio.h>

int main(void)
{
printf("Hello, World\n");
printf("Hallo Welt!\n");

return 0;
}
```

Änderung untersuchen (3 Zustände: Arbeitsverzeichnis, Index, Repsository (HEAD)):

```
$ git status

$ git diff hello_world.c // Arbeitsverzeichnis gegenüber Index
$ git diff --cached hello_world.c // Index gegenüber HEAD
$ git diff HEAD hello_world.c // Arbeitsverzeichnis gegenüber HEAD

$ cat hello_world.c // Version im Arbeitsverzeichnis
$ git show :hello_world.c // Version im Index
$ git show HEAD:hello_world.c // Version im HEAD
```

Änderung im Index und Repsository (HEAD) aufnehmen:

```
$ git add .
$ git commit -m "add german text"
```

6. Änderung im Arbeitsverzeichnis vornehmen und verwerfen

   Quelltext anpassen:

```
$ vi hello_world.c
#include <stdio.h>

int main(void)
{
printf("Hallo Welt!\n");

return 0;
}

$ git diff hello_world.c
```

   Änderung im Arbeitsverzeichnis verwerfen:

```
$ git checkout -- hello_world.c
```

7. Änderung im Arbeitsverzeichnis vornehmen, in den Index und das Repository (HEAD) aufnehmen und danach verwerfen

```
$ vi hello_world.c
// C - Hallo Welt Beispiel
#include <stdio.h>

int main(void)
{
printf("Hello, World\n");
printf("Hallo Welt!\n");

return 0;
}

$ git add .
$ git commit -m "add german comment"
$ git log --oneline
```

   Letzten Commit verwerfen:

```
$ git reset --hard HEAD~1
// oder:
$ git reset HEAD~1
$ git checkout -- hello_world.c
```

8. Änderung im Arbeitsverzeichnis in das Repository (HEAD) aufnehmen

```
$ vi hello_world.c
// C - hello world example
#include <stdio.h>

int main(void)
{
printf("Hello, World\n");
printf("Hallo Welt!\n");

return 0;
}

$ git add .
$ git commit -m "add comment"
```

9. Ältere Änderung zurücknehmen, aber im Repository behalten

```
$ git log --oneline

$ git revert HEAD~1

$ vi hello_world.c
$ git log --oneline
$ git diff <id aus 7.> hello_world.c
```

Aufgabe 2 (geführt)

1. In Gitlab einloggen und im Praktikum freischalten lassen
2. SSH-Schlüssel für Gitlab
    1. Key generieren:

        ```
        $ ssh-keygen -t rsa -C "<s00000>@informatik.htw-dresden.de"
        ```

        - Standard Dateipfad übernehmen
        - (optional) Passwort vergeben (Das Passwort würde für jede SSH-Key Kommunikation mit Gitlab benötigt)

    2. öffentlichen Key anzeigen lassen und in die Zwischenablage kopieren

        ```
        $ cat ~/.ssh/id_rsa.pub
        ```

    3. Profile Settings -> SSH Keys

        - Key: hier kopierten public key einfügen
        - Title: hier einen Namen vergeben (zur besseren Unterscheidung bei mehreren SSH Keys)

    4. Von Gitlab wird eine Informationsmail zu dem neuen SSH Key verschickt
3. Neues Projekt demo-hello-world anlegen
4. Das existiernde Repository **demo-hello-world** aus Aufgabe 1 mit dem gleichnamigen Gitlab Projekt verknüpfen und hochladen

```
$ (cd demo-hello-world)
$ (git init)
$ git remote add origin git@iglu.informatik.htw-dresden.de:<username>/demo-
hello-world.git
$ (git add .)
$ (git commit)
```

SSH-Key muss in Gitlab hinterlegt sein. Falls bei 1. ein Passwort für den SSH-Key vergeben wurde, muss dies bei `gut push -u origin ...` eingegeben werden.

```
$ git push -u origin master
```

5. Ansicht in Gitlab unter Projekten
    - Files, Commits, …
6. *README.md* (Markdown) anlegen

```
$ vi README.md
# Demo Hello World

Repository aus dem SE Praktikum zur Einführung in die Versionsverwaltung
mit Git.

$ git add .
$ git commit -m "add README.md"
$ git push -u origin master
```

7. Projekt in Gitlab (Files, Commits, …) und lokal `git log --oneline` anschauen
8. Das existierendes Projekt aus Gitlab als neues weiteres Arbeitsverzeichnis herunterladen, Ändern, hochladen und im anderen Arbeitsverzeichnis die Änderung herunterladen
    1. neue Arbeitskopie 2 herunterladen

    ```
    $ cd ..
    $ git clone git@iglu.informatik.htw-dresden.de:zirkelba/demo-hello-
    world.git demo-hello-world-2
    $ cd demo-hello-world-2
    ```

    Es wird ein Verzeichnis mit dem Repositorynamen angelegt. Optional kann ein anderer Name des Verzeichnisses angegeben werden.

    2. Änderung vornehmen und ins das lokale Repository aufnehmen

```
$ vi hello-world.c
// C - hello world example
#include <stdio.h>

int main(int argc, char* argv[])
{
printf("Hello, World\n");

return 0;
}

$ git add .
$ git commit -m "change main function parameters"
```

3. Änderung an das Remote-Repository (Gitlab) übertragen

```
$ git push -u origin master
```

4. Arbeitsverzeichnis 1 aktualisieren

```
$ cd .. && cd demo-hello-world
$ git pull
```