Übungen PdP-Praktikum — Blatt 3

Aufgabe 1:

Als Vorbereitung für Aufgabe 2: Erzeugen Sie ein leeres Verzeichnis namens `projekt`. Schreiben Sie ein Shellskript `neue-datei.sh`, das mit einem Parameter i aufgerufen wird und dann die Datei i.dat erzeugt, deren Inhalt aus drei Zeilen der Form `i i i i i i i i` besteht. Der Aufruf

```
./neue-datei.sh 2
```

erzeugt also die Datei `2.dat`, deren Inhalt so aussieht:

```
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
```

Erzeugen Sie nun damit die drei Dateien `1.dat, 2.dat` und `3.dat`. (Die Lösung darf ruhig etwas unelegant sein, da wir hier keine Schleifen behandeln.)

Aufgabe 2:

Erledigen Sie folgende Aufgaben. Welche Befehle brauchen Sie jeweils?

1. In eine globale git-Konfigurationsdatei `.gitconfig` Ihren Namen und Ihre Emailadresse eintragen,
2. Im Verzeichnis `projekt` aus Aufgabe 1 ein git-*repository* anlegen,
3. Alle Dateien in `projekt` *stagen* und dem *repository* hinzufügen,
4. Eine im lokalen git-Ordner `projekt` geänderte Datei *stagen* und im *repository* ablegen,
5. Eine Datei im lokalen git-Ordner löschen und die Datei aus dem *repository* wiederherstellen,
6. Alle Dateien im lokalen git-Ordner löschen und aus dem *repository* wiederherstellen,
7. Die Datei `neue-datei.sh` aus dem Repository entfernen und unstagen,
8. Eine Datei `.gitignore` so anlegen, dass die Datei `neue-datei.sh` im lokalen git-Ordner ab jetzt von git ignoriert wird (also z.B. von `git add *` nicht mehr dem *repository* hinzugefügt),
9. Die Datei `.gitignore` so ändern, dass ab jetzt alle Dateien der Form `*.sh` ignoriert werden.

Aufgabe 3:

1. Schreiben Sie ein Shellskript namens `wieviele`, das anzeigt, wieviel Unterverzeichnisse und wieviel Dateien das Unterverzeichnis `.git` eines als Argument angegebenen Verzeichnisses (Z.B. `projekt/`) aktuell enthält. Ein Aufruf soll z.B. so aussehen:

```
$ wieviele ~/projekt/
```

Die Ausgabe dann etwa so:

```
Das Verzeichnis ~/projekt/.git enthält 17 Unterverzeichnisse und 7 Dateien.
```

2. Wie können Sie geschickt herausfinden, in welcher Datei in in welchem Unterverzeichnis von `projekt/.git` ihre commit-Kommentare gespeichert werden? *Tipp:* `grep` "`^abc`" *liefert alle Zeilen, die mit abc anfangen.*

Aufgabe 4:

Legen Sie ein `git`-Repository an, und ändern Sie einige der Dateien. Probieren Sie `git status -s` aus. Welche Befehle brauchen Sie für folgende Aufgaben:

1. Die Datei `2.dat` so manipulieren, dass sie von `git status -s` angezeigt wird als `M 2.dat`?
   (Also Leerzeichen-M-Leerzeichen-2.dat).
2. Die Datei `2.dat` so manipulieren, dass sie von `git status -s` angezeigt wird als `M  2.dat`?
   (Also M-Leerzeichen-Leerzeichen-2.dat).
3. Die Datei `2.dat` so manipulieren, dass sie von `git status -s` angezeigt wird als `MM 2.dat`?
4. Die Datei `1.dat` so manipulieren, dass sie von `git status -s` angezeigt wird als `D  1.dat`?
5. Knifflig: Datei `3.dat` so manipulieren, dass sie von `git status -s` angezeigt wird als `?? 3.dat`?

Aufgabe 5:

Vorbereitung zu Aufgabe 6. (Auf einem Techfak-Rechner, oder per Fernzugang auf compute:) Wechseln Sie in das Verzeichnis `/vol/lehre/Linux`. Erzeugen Sie ein neues Verzeichnis git-XXX (wobei XXX für eine von Ihnen gewählte Zeichenkette steht). Wechseln Sie in das neue Verzeichnis und legen dort mit `git init --bare` ein neues, leeres Repository an. Dieses wird im Folgenden als *globales* Repository bezeichnet.

Um eine Warnung zu unterdrücken, die bei mir auftauchte, müssen Sie evtl. jetzt noch

```
git config receive.denyCurrentBranch ignore
```

eingeben. (Ich habe die Warnung nicht verstanden, aber stackexchange sagte mir, dass die so weggeht.)

Damit alle Nutzer dort Schreibrechte bekommen, machen Sie nun noch Folgendes: Wechseln Sie wieder in das Verzeichnis `/vol/lehre/Linux`. Geben Sie dort `chmod -R go+w git-XXX` ein.

(Nun können alle in allen Unterverzeichnissen — das macht das -R, für rekursiv — Dateien anlegen, ändern und löschen.)

Aufgabe 6:

Welche Befehle brauchen Sie für folgende Aufgaben:

1. Das globale git-repo aus Aufgabe 5 in Ihr Home-Verzeichnis klonen (Beim ersten Mal gibt es eine Warnung: "leeres Verzeichnis geklont" oder so. Egal, das legt sich, sobald im Verzeichnis was liegt, also ab Punkt 2)
2. Eine neu erstellte Datei `dat.txt` in Ihrem lokalen Verzeichnis git-XXX dem globalen Repository in `/vol/lehre/Linux` hinzufügen
3. Die aktuelle Version einer lokal geänderten Datei `dat.txt` in das globale Repository kopieren
4. Eine lokal geänderte Datei `dat.txt` wieder auf den Zustand im globalen Repository zurücksetzen
5. Eine Datei `dat.txt` aus dem lokalen und globalen Repository entfernen

Experimentieraufgabe:

Erzeugen Sie einen Versionskonflikt: Arbeiten Sie mit jemandem zusammen an demselben (nichtleeren) Repository. Pullen Sie beide, bearbeiten beide dieselbe Datei `xxx.txt`, und pushen Sie diese in etwa gleichzeitig.

Für den push, der *nicht* durchgeführt wurde, pullen Sie und reparieren Sie die Datei `xxx.txt` sinnvoll in ihrem lokalen Verzeichnis. Pushen Sie anschließend wieder.

Probieren Sie verschieden Möglichkeiten aus: Einmal ändern Sie nur verschiedene Zeilen (Person 1 Zeile 1, Person 2 Zeile 3), einmal ändern Sie beide dieselbe Stelle (etwa Zeile 1), einmal fügen Sie Zeilen hinzu (z.B. Person 1 ändert Zeile 1 und fügt dahinter eine Zeile hinzu, Person 2 ändert Zeile 4), einmal fügen Sie skrupellos viele Zeilen hinzu (z.B. Person 1 ändert Zeile 1 und fügt dahinter 10 Zeilen hinzu, Person 2 ändert Zeile 4).

Zusatzaufgabe:

Starten sie ein neues git-Projekt, dass Dateien `eins.dat`, `zwei.dat` und `drei.dat` enthält. Erzeugen Sie branches `testen` und `testen2` und manipulieren Sie sie so, dass die Ausgabe von `git log --oneline --decorate --graph --all` folgendes Bild ergibt:

```
*   d6f79c8 (HEAD -> master) Merge branch testen2 -> master
|\
| * 83a8321 (testen2) zwei.dat weiter optimiert
| * 80ee055 zwei.dat optimiert
* |   5340274 erster merge testen -> master
|\ \
| * | 86a2a66 (testen) eins.dat erweitert
| |/
| * 53864bc eins.dat verbessert
* | e169af4 drei.dat optimiert
* | 60254ac drei.dat verbessert
|/
* cba0022 eins.dat gefixt
* 009357a Neues Projekt
```