AUFGABE 1: QUELLCODEVERWALTUNG MIT GIT

Ziel dieser Aufgabe ist es, ein Gefühl für die dezentrale Versionsverwaltung mit `git` zu bekommen. In dieser Aufgabe erhalten Sie von uns ein bereits mit einer Versionshistorie befülltes `git`-Repository. An dieser Versionsgeschichte werden Sie eine Reihe von Änderungen vornehmen.

Nach Bearbeitung dieser Übungsaufgabe sollten Sie verstanden haben wie man Inhalte zu einem `git`-Repository hinzufügt und entfernt, wie man mit einem entfernten Repository kommuniziert, wie man Geschichte umschreibt, und vor allem was man machen muss, wenn mal etwas schiefgeht.

Notieren Sie sich für die Abgabe, welche Befehle Sie in den einzelnen Teilaufgaben verwendet haben und was Sie durch die Verwendung des Befehls erreichen wollten. Dokumentieren Sie bei jedem Schritt (nach dem initialen Klonen) die Ausgabe von

```
git log -n4 --graph --oneline >> abgabedatei
```

Halten Sie beides in einer dafür vorgesehenen Datei für die Abnahme bereit.

1 Aufgabenstellung

Aufgabe 1 *SSH-Schlüssel*

Da das Vorgabenrepository nur nach Anmeldung zugänglich ist, müssen Sie sich gegenüber Gitlab, beispielsweise mittels eines SSH-Schlüssels, authentifizieren. Erzeugen Sie dazu ein Schlüsselpaar und hinterlegen Sie den öffentlichen Schlüssel in Ihrem Gitlab-Account.

☞ `ssh-keygen`
☞ `.ssh/config`

Aufgabe 2 *Klonen*

Verwenden Sie `git` um die Übungsvorgabe herunterzuladen:

```
git clone git@gitlab.cs.fau.de:ezs/vezs19-vorgabe.git
```

Aufgabe 3 *Aufsetzen des Gruppenrepositories*

Legen Sie, wie in der Tafelübung besprochen, ein Gruppenrepository an. Bitte denken Sie daran, den Nutzer „i4ezs“ als Entwickler hinzuzufügen. Passen Sie die Konfigurationsdatei des Repositories so an, dass das Remote `origin` auf Ihr Gruppenrepository und das Remote `vorgabe` auf unser Vorgaberepository verweist (siehe Übungsfolien).

☞ `git remote rename`
☞ `git remote add`
☞ `.git/config`

Aufgabe 4 *Log*

Verschaffen Sie sich nun mittels `git log` einen Überblick über die Versionsgeschichte. Wenn Sie wollen, können Sie hierfür auch ein graphisches Werkzeug wie `git cola` oder `gitk` verwenden.

Aufgabe 5 *Hinzufügen*

Fügen Sie Ihren Namen und Ihre E-Mail-Adresse zur Datei `README` zum Abschnitt „Credits“ hinzu. Legen Sie anschließend einen Commit mit dieser Änderung an. *Wie sieht die Versionsgeschichte nun aus?* Dokumentieren Sie sie in Ihrer Abgabedatei.

☞ `git add`

☞ `git commit`

Aufgabe 6 *Rückgängig machen*

Verwenden Sie nun `git reset --hard HEAD^` um den letzten Commit rückgängig zu machen. Betrachten Sie die Versionsgeschichte. *Worin liegt der Unterschied zwischen dem Ausgangsrepository und dem aktuellen Zustand?*

Antwort:

Aufgabe 7 *Benennung*

Was beschreibt der Platzhalter `HEAD^`*?*

Antwort:

Aufgabe 8 *Zeitreise*

Betrachten Sie nun die Ausgabe von `git reflog`.
Was sehen Sie hier?

Antwort:

Aufgabe 9 *git reset (I)*

Suchen Sie nach dem Zustand, in dem sich das Repository am Ende der Teilaufgabe 5 „Hinzufügen“ befand, und stellen Sie diesen mit Hilfe von `git reset --hard` wieder her! Lesen Sie die man-Page von `git reset`.
Was wird hier rückgesetzt*?*

☞ `man git-reset`

Antwort:

Aufgabe 10 *git reset (II)*

Wie verändert der Kommandozeilenparameter `--hard` *das Ergebnis des Befehls?*

Antwort:

Aufgabe 11 *Zusammenarbeit*

Ändern Sie in der Datei `modern.astimerc` den Wert des „handThickness“-Eintrags von 70 auf 50 und legen Sie einen Commit für diese Änderung an.

Ziehen Sie nun mit Hilfe von `git pull` die Commits aus dem Repository

```
git@gitlab.cs.fau.de:ezs/vezs-vorgabe-dev.git
```

nach.

Beheben Sie den dabei entstehenden Konflikt. Sie können `git status` verwenden, um in Erfahrung zu bringen, welche Dateien von dem Konflikt betroffen sind. Achten Sie dabei auf den **Erhalt Ihrer Änderung**. Betrachten Sie nun die Historie – *wie Sieht diese aus?* Wie nennt man den Typ des aktuellsten Commits?

☞ `git mergetool`

Antwort:

Aufgabe 12 *git rebase*

Verwenden Sie jetzt `git rebase` um diesen Commit zu beseitigen. Achten Sie bei eventuell auftretenden Konflikten wiederum auf den Erhalt Ihrer Änderungen. *Wie sieht die Historie jetzt im Vergleich zu vorher aus?*

Antwort:

Aufgabe 13 *Geschichte neu schreiben*

Führen Sie einen interaktiven Rebase durch, in dem Sie die beiden Commits, die den Konflikt betreffen, miteinander verschmelzen. *Welchen Commit müssen Sie als Basis für* `git rebase -i` *angeben?*

☞ `git rebase -i`

Antwort:

Aufgabe 14 *Umsortieren und verschmelzen*

Führen Sie nun einen interaktiven Rebase auf dem Commit vom Montag, den 27. Mai 2013 um 18:45:34 Uhr durch und gruppieren Sie diejenigen Commits, die Warnungen des Übersetzers betreffen. Sie erkennen die entsprechenden Commits an dem Wort „warning“ in der Commitnachricht. Verschmelzen Sie diese Commits anschließend miteinander, so dass nur noch ein Commit übrig bleibt, der Übersetzerwarnungen in Ordnung bringt.

Notieren Sie sich jetzt die Ausgabe von `git reflog` *für die Abgabe.*

Aufgabe 15 *Hochladen*

Laden Sie nun Ihre Versionsgeschichte als neuen `master`-Branch in Ihr Gruppenrepository hoch.

Aufgabe 16 *Abschließende Frage*

Angenommen die beiden Repositories, aus denen Sie gezogen haben, wären öffentlich zugänglich – wieso wären die Rebase-Schritte, die Sie durchgeführt haben, dann eine schlechte Idee?

Antwort:

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 16.05.2019
- Fragen bitte an `i4ezs@lists.cs.fau.de`