Timo Kötzing, Aleksander Morgensterns und Tyron Franzke Sommer 2024

Knobelaufgaben Tag 11

Eine Übersicht über unsere Themen findest du hier:

https://hpi.de/friedrich/docs/scripts/24_Vorkurs/index.html

Es lohnt sich, diese Seite beim Bearbeiten der Aufgaben offen zu haben. Für unsere Freunde das analogen Aufgabenblattes gibt es am Ende noch einen QR-Code.

Prolog

In der Welt des Codes, so tief und weit,
Da gibt es ein Tool, das stets bereit.
Es heißt "Git", ein Hüter von Versionen,
In der Entwickler-Welt, voller Emotionen.
- ChatGPT

Aufgabe 1: Gitlab/-hub Account erstellen

Jetzt geht's ans Eingemachte! Für die heutigen Übungen und das Zusammenarbeiten darüber hinaus, benötigt ihr einen Account auf einem Repository-Hoster, wie Github oder Gitlab. Wir empfehlen an dieser Stelle, einen Github-Account zu erstellen.

(a) Gehe auf https://github.com/. Klicke dort auf *Sign up* und folge den Anweisungen.

(b) Gehe auf https://about.gitlab.com/. Klicke dort auf *Pricing* und klicke dort unter *Free* auf *Get started*. Alternativ kannst du auch einfach auf *Get free Trial* klicken.

Aufgabe 2: Git klonen

Lassen wir nun unsere Beine ein wenig im Wasser baumeln. Klone dir das folgende Repository: https://github.com/tyronfranzke/hpi-vorkurs-git.

Aufgabe 3: Git Repo erstellen

Jetzt wo ihr alle Datein lokal habt, kopiert sie in einen anderen Ordner und erstellt ein eigenes Repo mit diesen Dateien.

Aufgabe 4: Git ignore

Als Hausaufgabe solltet ihr ein Programm schreiben, um euch automatisch auf einer Seite anzumelden. Kompilier das C-Programm manuell und teste, ob es funktioniert. Fixe gegebenfalls Fehler.
Jetzt willst du ja nicht die Executable oder (insbesondere!) deine geheimen Login-Daten auf deinem öffentlichen Repo rumliegen haben. Schreibe eine `.gitignore`-Datei, die alle Executables und die `secret data.txt` Datei beim commiten ignoriert. Lösche die Textdatei vom Repo mit `git rm --cached {Dateiname}` (die –cashed Flag macht, dass die Datei nicht lokal gelöscht wird). Commite und pushe dann.

Aufgabe 5: Switcheroo

So, ihr habt jetzt genug alleine gearbeitet. Git ist schließlich zum Kollaborieren da! Wählt euch einen Übungspartner (oder eine Dreiergruppe) und ladet euch gegenseitig zu eurem Repo ein. Klont dann das Repo des jeweils anderen.

Aufgabe 6: Branching

In der Datei `super-wichtige-hausaufgabe.txt` befinden sich einige Rechtschreibfehler. Ihr wollt ja nicht, dass euer Übungspartner durchfällt! Fixt die Rechtschreibfehler, erstellt eine neue Branch und commitet in diese mit einem netten Kommentar. Gebt dann `git status` ein und prüft, ob ihr in der korrekten Branch seid.

Aufgabe 7: Merging

Wartet darauf, dass euer Partner fertig ist. Geht dann auf euer Repo und kommentiert den Commit ebenso nett. Pulled und merged dann die von eurem Partner erstellte Branch in die `main`-Branch. Mit `git log` könnt ihr die lange Geschichte eueres Repos bestaunen.

Aufgabe 8: Commit-Messages verbessern

So geht das nicht! Du hast nun einige Commits gemacht, und deine Commit-Nachrichten waren schlampig und nicht besonders aussagekräftig. Ändere eine deiner letzten Commit-Nachrichten, damit sie klarer und präziser wird, sodass ich nicht auf den letzten Metern von dir enttäuscht werde. Nutze dazu den Befehl `git commit --amend`, um die Nachricht deines letzten Commits zu bearbeiten.
Wenn du eine ältere Commit-Nachricht ändern möchtest, kannst du `git rebase -i` verwenden. Achte darauf, die Historie des Projekts übersichtlich und sinnvoll zu gestalten!

Aufgabe 9: Git Blame

Manchmal möchte man gerne wissen, wer so b!*d war und in euren schön geschriebenen Code einen Bug eingebaut hat. Nutze den Befehl `git blame`, um herauszufinden, wer Dateien geändert hat und auf wen die Schuld zu schieben ist. Seit einiger Zeit funktioniert `comprog/submission-final-trust-2.py` nicht mehr wie gewünscht, sie gibt aus, dass 3 keine Primzahl ist. Wer hat hier einen Bug eingebaut?

Aufgabe 10: Merge-Konflikt auflösen

Viele Leute posten auf Social Media tagtäglich ihren Lebensalltag, mal mehr und mal weniger gewinnbringend. Wir machen das jetzt auch, bloß teilen wir das nur mit unserem Übungspartner.

Schritt 1: Teilt euch auf! Dein Partner erstellt eine neue Branch namens `ferien-erlebnis`, und du bleibst vorerst auf der `main`-Branch. Ihr beide öffnet die Datei `super-wichtige-hausaufgabe.txt` und schreibt dort jeweils euer **schönstes Ferienerlebnis** zwischen die Anführungszeichen.

Schritt 2: Zeit für das große Merge Dein Partner committet die Änderungen auf der `ferien-erlebnis`-Branch und merged die Branch in die `main`-Branch, während du deine Änderungen direkt auf der `main`-Branch committest.
Nun versucht ihr beide, die `main`-Branch zu aktualisieren. Du führst ein `git pull` aus, um die Änderungen von deinem Partner zu holen. Oh nein! Git warnt euch vor einem Konflikt!

Schritt 3: Konflikt lösen!
Keine Angst, wir sind hier alle pazifistisch (also meistens). Git zeigt dir in der Datei die Stellen an, die zu einem Konflikt geführt haben. Besprecht gemeinsam, welches Ferienerlebnis besser ist, oder kombiniert eure Änderungen. Sobald ihr die Datei editiert habt, speichert und committet die endgültige Version mit einer passenden Commit-Nachricht.

Aufgabe 11: Git Stash

Angenommen, du arbeitest an einer neuen Funktion, aber plötzlich kommt etwas Dringenderes auf, das du zuerst erledigen musst. Du möchtest deine bisherigen Änderungen nicht verlieren. Nutze den Befehl `git stash`, um deine Arbeit vorübergehend zu speichern. Wechsle dann in einen anderen Branch, erledige dort deine Aufgabe und hole die gespeicherten Änderungen mit `git stash pop` zurück.
Mit `git stash list` kannst du dir jederzeit ansehen, welche Änderungen du zwischengespeichert hast.

Aufgabe 12: Git Config

Du hast jetzt alle wichtigen Features gelernt, die git zu bieten hat und die ihr im Laufe des Studiums brauchen werdet. Als letzten Schritt könnt ihr noch mit `git config` die E-Mail des Repo auf eure HPI-Mail ändert (ihr habt sie offiziel noch nicht, aber das Format ist vorname.nachname@student.hpi.uni-potsdam.de).

Aufgabe 13: It's meming time

Es ist Zeit für die ultimative Challenge! Erstellt Memes zum Vorkurs 2024. Ihr könnt dabei Tipps aus `memes.pdf` befolgen (findet ihr im Repo). Das besten Memes (demokratisch gewählt) werden mit Preisen überschüttet, also - Mitmachen lohnt sich!

BONUS - Conventional Commits

Du magst Ordnung? Du findest es ansprechend, wenn deine Git-Historie schön linear ist und einem konsistenten Schema folgt? Dann bist du hier genau richtig. Lerne jetzt wie du vernünftig Commits benennst: https://www.conventionalcommits.org/en/v1.0.0/

Rätselzeit

Fertig! Frage deine Tutorin oder deinen Tutor, ob er von einem Rätsel schwärmen kann.

Skript - Git