Git für pragmatische Einsteiger

Hands ON!

- Wer will kann parallel mitarbeiten
- Benötigt werden:
- Github / Bitbucket whatever account
- Oder Server mit ssh Zugang
- Installieren:
  - Git
  - git gui → simple Ui für Standartaufgaben
  - Meld installiert → 3way Diff editor
  - Gitk → History viewer

Übersicht

- Git init || git add remote || git clone
- .gitignore
- Git add
- Git rm
- Git push & Git pull
- Git mergetool
- Git reset
- branching
- Was kann git gut, was nicht

UI Tools

- Die meisten sind Katastrophal schrottig
  - Viel sind zudem unheimlich langsam…
  - … und haben feherhafte workflows die den lokalen clone zerstören können
  - … und das Wissen ist oft tool spezifisch und nicht transferierbar
  - Kann ich nicht zum Einstieg empfehlen
- Git gui
  - Bewusste begrenzter Scope, den dafür ok
  - Kernui von git, kann Zeit sparen, zeigt diff direkt
  - Gut für git add, add remotes, git rm

Git init || git add remote || git clone

- Git init → erzeugt ein lokales git repository
- Git remote add, fügt einen remote server hinzu
  - Origin der normale remote name
  - zb. „git remote add origin http://gitwhatever/myrepository“
  - Benötigt, wenn ein schon vorhandenes Projekt hochgeladen werden soll

- Alternativ „git clone“ http://gitwhatever/myExistingRepository“
  - Erstellt einen lokalen klon, in welchen ich arbeiten kann
  - Fügt die URL als remote mit namen origin hinzu
  - Benötigt wenn ein Projekt schon remote existiert und wir dies benutzen wollen

.gitignore

- Magic file für git
- Wird mit ins git gepackt, für alle anderen Benutzer
- Enthält pattern für per default ignorierte Dateien
  - .class || .o || .tmpWhatever
  - bin/ excludiert den bin Ordner und seinen Inhalt
  - .jar ignoriert alle jar Files egal wo die sind
  - Es dürfen multiple Dateien existieren, in jedem Ordner der im git
    - Z.b in resources/ andere gitignore wie in src/
- Generator auf gitignore.io

Git rm

- Löscht eine Datei
  - Aus git und von der Platte
  - Hab ich noch nie von Hand benutzt (git gui)

Git add

- „Git add .gitignore“
  - Teilt git mit, dass es sich um diese Datei kümmern soll
  - Wird auch benutzt um git mitzuteilen, das Probleme in einer Datei behoben wurden
    - Mehr dazu später

Git commit

- Git commit -m „my great commit message“
  - Man sollte vorher status benutzt haben!
- Benutze ich fast nie, da git gui

Git status

- Zeigt was mit dem Repository los ist
- Empfiehlt teilweise was zu tun ist
- Hat im zweifelsfall recht

Branching

- Git branch
  - Zeigt aktuellen Branch
- Git branch yay
  - Erzeugt neuen Branch yay
- Git checkout yay
  - Wechselt zu yay
- Git branch -d yay
  - Löscht yay

Git mergetool (aka ich habe Konflikte)

- Git mergetool -t meld
  - Started Interaktive Konfliktbehebung mit Meld als mergetool
- Statt meld können auch andere Tools benutzt werden, ich mag lediglich meld
- Ich benutze bewusst ein externes dummes Tool zum mergen!
  - Auch wenn es evtl. noch einen amend Commit benötigt zum aufräumen

Git reset

- Aka I fucked up
  - Setzt den Dateizustand auf einen bestimmten Hash oder tag/label
  - z.B. git reset –hard stellt den letzten comitteten stand wieder her
- --hard wird meistens benötigt
- Ist offensichtlich destruktiv gegenüber lokalen Änderungen

Git push & Git pull

- Git push || git push notOrigin
  - Läd alle Änderungen auf origin (oder angegebenen remote namen) hoch
  - Schlägt fehl wenn remote unbekannte nicht integrierte Änderungen sind
  - Git push origin –delete branchnamel
- Git pull || git pull notOrigin
  - Läd alle Änderungen von origin herunter
  - Wird Änderungen vom aktuellen Branch direkt lokal anwenden
  - Benötigt bei Konfliken manuellen Eingriff

- Ändert NIE bereits gepushte commits
  - Ja dies benötigt eine Eigene Seite!
  - Grüße an einen meiner Kollegen an dieser Stelle
  - Auch: Wie macht man sich im Projekt unbeliebt

Was kann git gut:

- Text basierte / simple Dateien
- Kleinere nicht veränderliche Binaries
  - eg. eine 100kb png mit icons
  - Aufpassen, nicht mergebar, darf nur einer ändern
- Viele Leute mehr oder weniger parallel
  - Einsteiger in Verzweiflung treiben :)
    - Oft mit UI tools verstärkt, statt verhindert
- Offline arbeiten
- Backups / Datenverlust
- Viele Köche
- Schlechtes Internet
  - Git hat eine gutes Verhältniss von Übertragungsgrößen zu Sourcegrößen

Was kann git nicht so gut:

- Große Dateien
  - Git benötigt 2-4x ram wie größte Datei auf Server und Client!
  - Also kann ein 32Gb ramdump alles töten
- Binaries
  - Git kann diese nicht mergen
  - Git diff tool kann sehr langsam werden
  - Partielle Lösung git-lfs
- Schlechtes Internet
  - Clone ist atomar, wenn die Verbindung instabil ist darf man von vorne anfangen

Fragen

- Gibt es noch Fragen?