Git

... eine Einführung

Timeline

1. Einführung

2. Commits und Referenzen

Was ist Version Control?

Bietet:

- Protokoll der Änderungen an den Daten
- Ermöglicht dadurch auf beliebige alte Versionen zuzugreifen
- Unterschiedliche Entwicklungszweige
- Geteilter Zugriff auf die Daten

Umsetzung in Git

- Git ist dezentral, es benötigt *keinen* Git-Server, insbesondere keinen Host wie GitHub, GitLab, ...
- Die Protokollierung der Änderungen geschieht durch sog. **commits**
- **commits** sind identfizierbar über ihren Hash und haben *eindeutige* Eltern!
- Die Struktur ist ein „gerichteter azyklischer Graph“

Logbeispiel

- Zeiger auf den aktuellen commit: **HEAD**
- Man alle Vorgängercommits die zum aktuellen Status der Daten führen erreichen!
- Andere Entwicklungszweige sind einfach weiterer solcher Zeiger, sog. **branch**es.
- „master“ ist ein typischer Name der Hauptbranch *aber nicht vorgegeben!*

Tags

Also für jedes Release / jeden Zustand den man „festhalten“ möchte eine neue Branch? – Nein, es gibt **tags**.

- Referenz auf einen bestimmten **commit**
- Hat einen Namen (bspw. Versionsnummer `v3.14`)
- Es gibt „lightweight“ Tags und „annotated“ Tags

Tags II

Lightweight

- Besteht rein aus Referenz und Name
- Keinerlei weitere Informationen wie Name des Autors und Zeitpunkt

Annotated

- Speichert zusätlich Metainformationen, bspw. auch Beschreibung
- Sinnvoll für größere Releases, bspw. um relevante Änderungen zu listen

Remotes: Arbeiten mit Online-Repositories

Bisher: Entwickler arbeitet alleine auf seinem Rechner, Code bleibt lokal.

Möchten: Code Teilen: **remote**s

Remotes: Arbeiten mit Online-Repositories

Bisher: Entwickler arbeitet alleine auf seinem Rechner, Code bleibt lokal.

Möchten: Code Teilen: **remote**s

Remote:

- Benötigt Server (bspw. `git.imp.fu-berlin.de`, GitHub, GitLab)
- Es können mehrere remotes eingetragen werden
- **pull**/**push**: Änderungen holen/veröffentlichen

Divergente Entwicklung: Mergen

Szenario:

1. Mehrere Entwickler ändern Datei **A** in ihren lokalen Repos **l1** und **l2**

Divergente Entwicklung: Mergen

Szenario:

1. Mehrere Entwickler ändern Datei **A** in ihren lokalen Repos **l1** und **l2**
2. Entwickler 1 veröffentlicht Änderung auf gemeinsamer Remote **r**

Divergente Entwicklung: Mergen

Szenario:

1. Mehrere Entwickler ändern Datei **A** in ihren lokalen Repos **l1** und **l2**
2. Entwickler 1 veröffentlicht Änderung auf gemeinsamer Remote **r**
3. Entwickler 2 versucht ebenfalls zu **push**en: Schlägt fehl, da „Geschichte“ divergiert ist / seine lokale Version nicht mehr „aktuell“ ist

Divergente Entwicklung: Mergen

Szenario:

1. Mehrere Entwickler ändern Datei **A** in ihren lokalen Repos **l1** und **l2**
2. Entwickler 1 veröffentlicht Änderung auf gemeinsamer Remote **r**
3. Entwickler 2 versucht ebenfalls zu **push**en: Schlägt fehl, da „Geschichte“ divergiert ist / seine lokale Version nicht mehr „aktuell“ ist
4. Muss vorher **pull**en: Die Änderungen auf dem Server werden in die lokale Kopie integriert

Divergente Entwicklung: Mergen

Szenario:

1. Mehrere Entwickler ändern Datei **A** in ihren lokalen Repos **l1** und **l2**
2. Entwickler 1 veröffentlicht Änderung auf gemeinsamer Remote **r**
3. Entwickler 2 versucht ebenfalls zu **push**en: Schlägt fehl, da „Geschichte“ divergiert ist / seine lokale Version nicht mehr „aktuell“ ist
4. Muss vorher **pull**en: Die Änderungen auf dem Server werden in die lokale Kopie integriert

! *WICHTIG*: Die Reihenfolge der Änderungen auf der Remote sind fest und dürfen nicht mehr geändert werden

Divergente Entwicklung: Mergen

Szenario:

1. Mehrere Entwickler ändern Datei **A** in ihren lokalen Repos **l1** und **l2**
2. Entwickler 1 veröffentlicht Änderung auf gemeinsamer Remote **r**
3. Entwickler 2 versucht ebenfalls zu **push**en: Schlägt fehl, da „Geschichte“ divergiert ist / seine lokale Version nicht mehr „aktuell“ ist
4. Muss vorher **pull**en: Die Änderungen auf dem Server werden in die lokale Kopie integriert

! *WICHTIG*: Die Reihenfolge der Änderungen auf der Remote sind fest und dürfen nicht mehr geändert werden

$\Rightarrow$ Lokale Änderungen werden (automatisch) angepasst und „angehängt“

Mergeconflicts: Überschneidende Änderungen

Problem: Gleiche Datei wurde geändert.

Original:

```
1    the quick brown
2    fox jumps ovver
3    the lazy dog
4
```

Entwickler 1:

```
1    The quick brown
2    fox jumps ovver
3    the lazy dog
4
```

Entwickler 2:

```
1    the quick brown
2    fox jumps over
3    the lazy dog
4
```

- Textdatei & Änderung in verschiedenen Zeilen

$\Rightarrow$ Git kann idR. automatisch Änderungen zusammenführen

Referenzen I

- Git project.
  Git Referenz
  https://git-scm.com/docs
- Scott Chacon, Ben Straub.
  Pro Git E-Book
  https://git-scm.com/book/en/v2
- Wikipedia Autoren.
  Artikel der englischen Wikipedia
  https://en.wikipedia.org/wiki/Git