Git-Kurs

Für Anfänger und Auffrischer

04.06.2019 Aaron Seggelke, David Kathmann , Florian Marks, Kerstin Dückman, Valerij Stelmach

Technology
Arts Sciences
TH Köln

Agenda

- Motivation
- Vokabular
- Aufgaben

Motivation – Warum Versionierung?

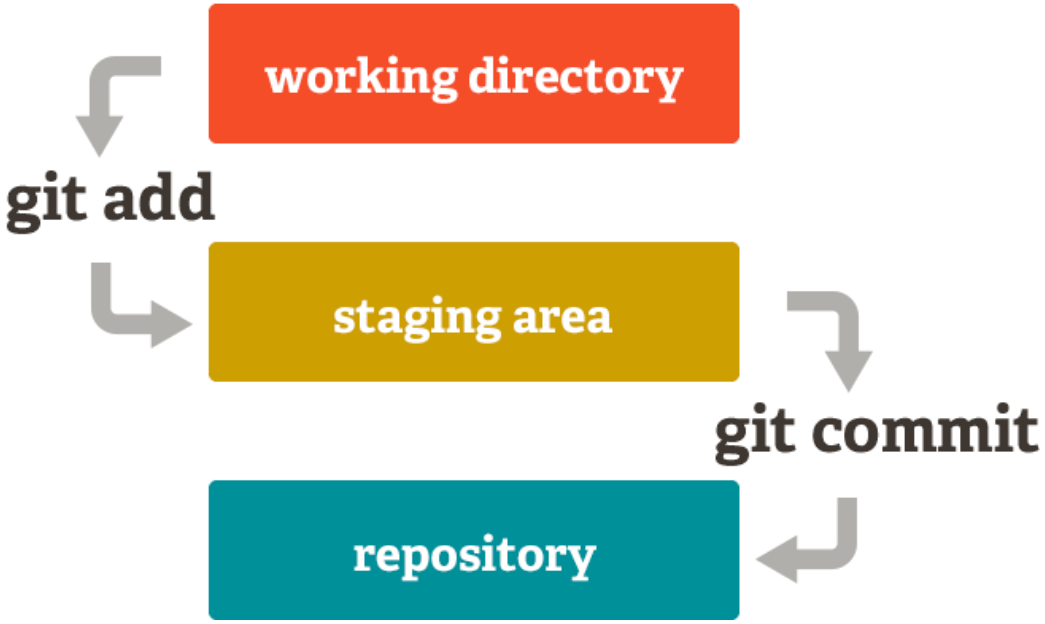
- Verfügbarkeit eines funktionierenden Standes
- Nachvollziehbarkeit
- Möglichkeit, zu vergleichen
- Schritte rückgängig zu machen

Motivation – Warum Git?

- Kostenlos und quelloffen
- Schnell und geringer overhead
- Lokal sowie zentralisiert nutzbar
- Anwendbar auf verschiedenste Entwicklungsparadigmen
- Am weitesten verbreitet (Bspw. Google, Facebook, Microsoft)

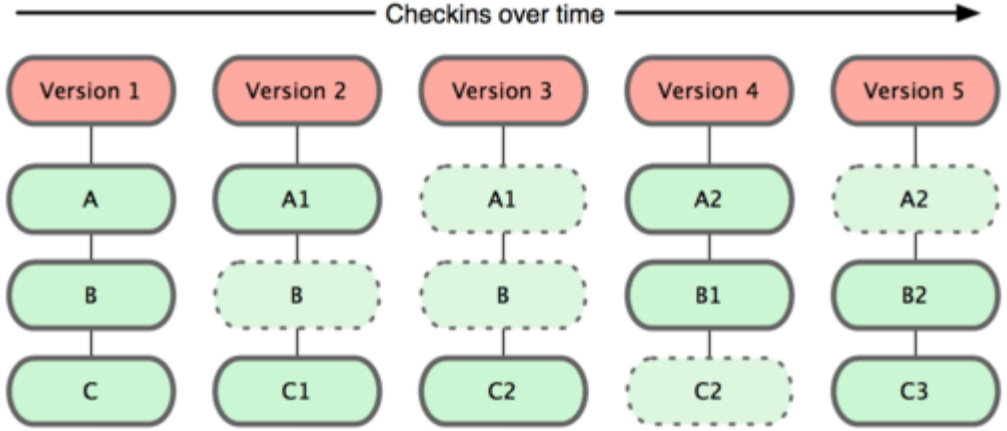
Vokabeln

- Working directory: Ordner/Zustand des Projekts, an dem direkt gearbeitet wird
- Staging Area: Vorbereitungsstufe für commtis
- Repository: „Ort“, wo Projekt in Form von commits existiert und versioniert wird



Vokabeln

- Commit: „Schnappschuss“ des Projekts
  - Veränderte Dateien werden kopiert
  - Unveränderte Dateien werden referenziert
  - Identifiziert durch hash, der aus commit berechnet wird (Veränderungen, Autor, commit message usw.)



Aufgabe 1 – Git lokal: config

1. `git config --list`
2. `git config --global user.name "John Doe"`
3. `git config --global user.email johndoe@example.com`

Aufgabe 1 – Git lokal: Initialisieren

1. `pwd`
2. `cd`
3. `mkdir repos`
4. `cd repos`
5. `mkdir git-kurs-aufgabe-1`
6. `cd git-kurs-aufgabe-1`
7. `git init`
8. `git config --list`
9. `git status`

Aufgabe 1 – Git lokal: `.gitignore`

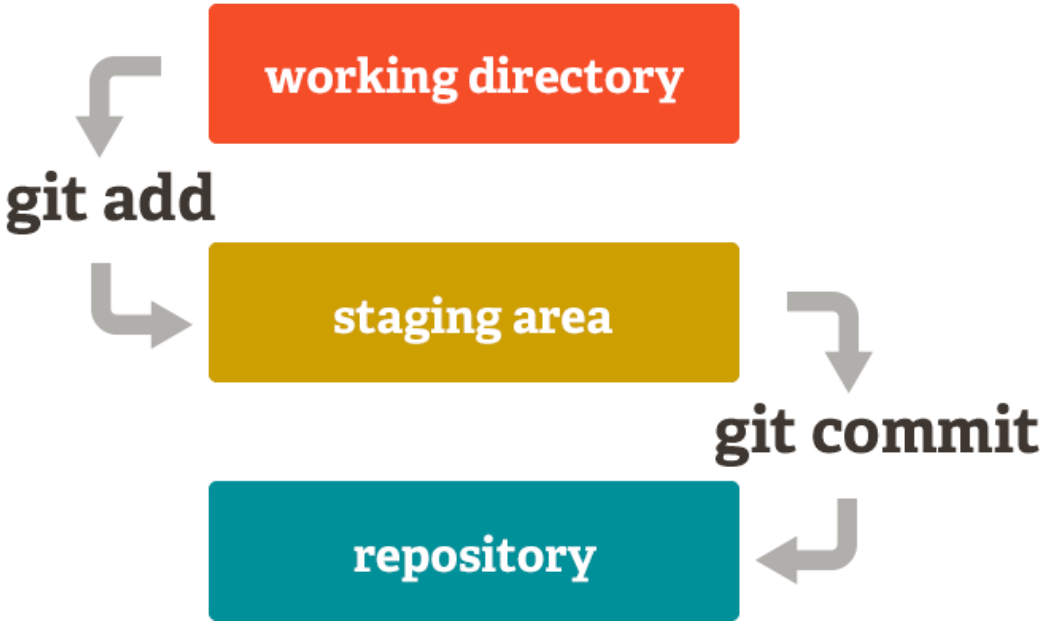
Definiert Dateien und Verzeichnisse, die von Versionierung ausgeschlossen werden sollen

- Bsp.: IDE-spezifische Daten und Einstellungen, node-modules usw.

```
1. touch .gitignore
2. echo "no-tracking.md" >> .gitignore
3. cat .gitignore
4. git status
5. touch no-tracking.md
6. git status
```

Aufgabe 1 – Git lokal: `commit`

1. `git add .gitignore`
2. `git status`
3. `git commit -m "Fuegt .gitignore hinzu"`
4. `git status`
5. `touch readme.md`
6. `git status`
7. commit für readme.md erstellen

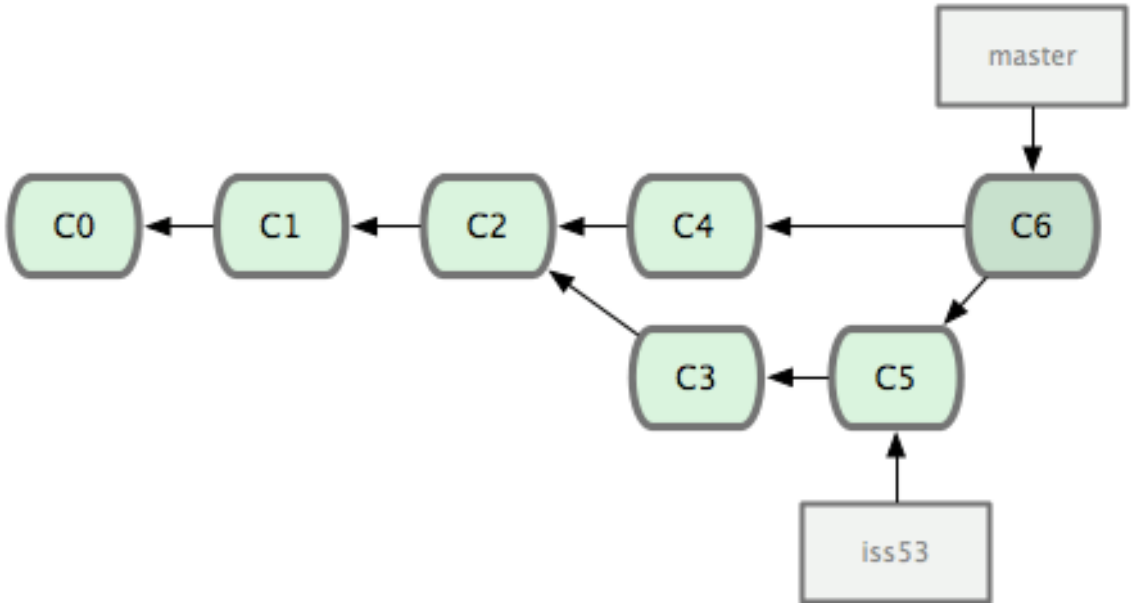


Aufgabe 1 – Git lokal: `diff`

1. `echo "# Aufgabe 1" >> readme.md`
2. `git diff`
3. commit für Ändern der readme.md erstellen
4. `git log`

Aufgabe 2 – Git branch

1. `git branch feature-x`
2. `git branch`
3. `git checkout feature-x`
4. `touch feature-x.md`
5. commit für feature-x.md erstellen
6. `git merge master`
7. `git checkout master`
8. `git merge feature-x`



Aufgabe 3 – Git remote: clone

Git: Dezentrales System zur Versionskontrolle

GitHub: Webservice, der Git-Repositories hostet

und Git- und weiter Funktionalitäten zur Verfügung stellt

1. Auf GitHub neues Repository anlegen; clone-Link kopieren
2. In `repos`-Ordner wechseln
3. `git clone` **`<repo-clone-link>`**
4. `cd` **`<repo-name>`**

Aufgabe 3 – Git remote

1. `touch remote-test.md`
2. commit für remote-test.md erstellen
3. `git push origin master`
4. Auf Github readme.md anpassen
5. `git status`
6. `git fetch`
7. `git pull origin master`

Aufgabe 4 – Merge-Konflikt

1. In `repos`-Ordner wechseln
2. `git clone https://github.com/git-kurs/aufgabe-4.git`
3. `cd aufgabe-4`
4. README.md Kontaktinfos mit beliebigem Editor anpassen
5. commit für README.md erstellen
6. `git pull origin master`
7. Merge-Konflikt lösen
8. commit für Merge-Konflikt erstellen