Git-Kurs

Für Anfänger und Auffrischer

Prof. Dr. Elisabeth Exempel
Ggf. Funktionsbezeichnung
Instituts- und/oder Fakultäts-, Referats-, Teambezeichnung (maximal drei Zeilen)

Agenda

- Motivation
- Vokabular
- Aufgaben

Motivation – Warum Versionierung?

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

Motivation – Warum Git?

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

Vokabeln

- Repository: Ordner oder Speicherplatz, wo Projekt liegt
- Commit: „Schnappschuss“ des Projekts

  (genauer: der Änderungen zum letzten commit)
- HEAD: Zeiger, der im Normalfall auf den aktuellsten commit zeigt

Aufgabe 1 – Git lokal

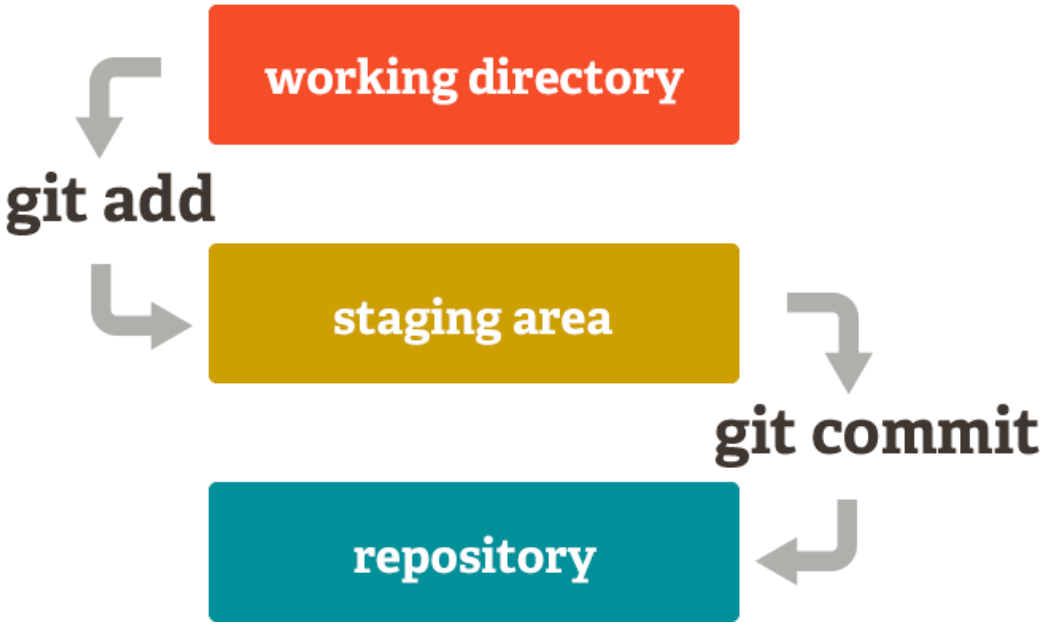
1. git config --list
   1. git config --global user.name "**John Doe**"
   2. git config --global user.email **johndoe@example.com**
2. cd [...]/repos/git-kurs-aufgabe-1
3. git init
4. git config --list
5. git status

Aufgabe 1 – Git lokal: `.gitignore`

1. Definiert Dateien und Verzeichnisse, die von Versionierung nicht berücksichtigt werden sollen
   - Bsp.: IDE-spezifische Daten und Einstellungen, node-modules
2. `touch .gitignore`
3. `vim .gitignore`
   1. → `no-tracking.md`
   2. `ESC :wq`
4. `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. → Einen commit für die Datei readme.md erstellen

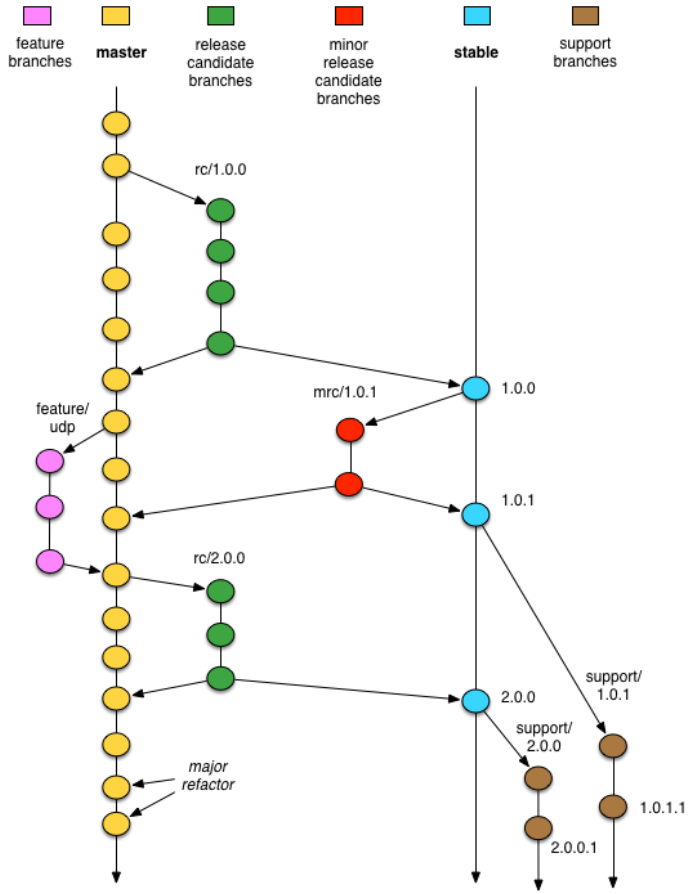


Aufgabe 1 – Git lokal: `commit`

1. `echo “# Aufgabe 1“ >> readme.md`
2. `git add readme.md`
3. `git commit`
   1. Wenn keine commit-message angegeben wird, wird Editor geöffnet um diese zu ergänzen
   2. commit-message schreiben und Editor verlassen (vim: ‘`ESC :wq`‘ etc.)
4. `git status`

Aufgabe 2 – Git branch

1. `git branch`
2. `git branch feature-x`
3. `git status`
4. `git checkout feature-x`
5. `touch feature-datei.md` → neue Datei commiten
6. `git status`
7. `git merge master`
8. `git checkout master`
9. `git merge feature-x`



Aufgabe 3 – Git remote

- Git: dezentrales System zur Versionskontrolle
- GitHub: Webservice, der Git-Repositories hostet und Git-Funktionalitäten zur Verfügung stellt

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

Aufgabe 3 – Git remote

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

Aufgabe 4-6 – Vorbereitung

1. Repository klonen: https://github.com/git-kurs/TestRepo
2. Eigenen branch anlegen („`branch_`**`<nachname>`**“)

- Benötigte Befehle: : `git clone <url>, git branch <branch>, git checkout <branch>, git push origin <branch>, cd <verzeichnis>`

Aufgabe 4

1. Datei im root-Verzeichnis anlegen: `testfile_<nachname>.txt`
2. In die Datei „`port = 3000`“ schreiben
3. Änderungen commiten und auf eigenen branch pushen
4. Änderungen in den master branch mergen (in dem branch sein, in den man mergen will)

- Benötigte Befehle: : `touch <dateiname>, vim <dateiname>` (CTRL + C → Schreibmodus beenden, „:wq“ -> Speichern und Verlassen des Editors), `git add <dateiname>, git commit -m “<message>“, git push, git merge <branch>`

Aufgabe 5

1. Im `master`: „`port = 80`“ ändern und commiten
2. Im eigenen branch: „`port = 8080`“ ändern und commiten
3. `master` in eigenen branch mergen
4. Merge-Konflikt beheben (8080 als richtige Variante), dann eigenen branch pushen
5. Zum `master` wechseln und eigenen branch hinein mergen

- Benötigte Befehle: : `touch <dateiname>, vim <dateiname>` (CTRL + C → Schreibmodus beenden, „:wq“ -> Speichern und Verlassen des Editors), `git add <dateiname>, git commit -m “<message>“, git push, git merge <branch>`

Aufgabe 6

1. Weil aktueller Stand fehlerhaft: Repo auf commit „`add client`“ zurück setzen
   - `git checkout` **`<commit-hash>`**
2. Dies als neuen akutellsten Stand auf eigenen branch pushen

- Benötigte Befehle: : `touch <dateiname>, vim <dateiname>` (CTRL + C → Schreibmodus beenden, „:wq“ -> Speichern und Verlassen des Editors), `git add <dateiname>, git commit -m “<message>“, git push, git merge <branch>`