Verteilte Systeme – Übung

Michael Eischer, Laura Lawniczak, Christopher Eibel,
Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.cs.fau.de

Sommersemester 2019

Überblick

Organisatorisches
- Übung
- Versionsverwaltung mit Git

Termine und Ansprechpartner

- Tafelübung
  - Mittwoch: 12:15–13:45 Uhr, Raum 0.031-113
- Rechnerübung
  - Mittwoch: 14:00–16:00 Uhr, CIP-Pool 02.151{a,b}-113
  - Betreuung (nur) bei Bedarf
- Ansprechpartner
  - Michael Eischer Raum 0.045 `eischer@cs.fau.de`
  - Laura Lawniczak Raum 0.041 `lawniczak@cs.fau.de`
  - Christopher Eibel Raum 0.042 `ceibel@cs.fau.de`
  - Tobias Distler Raum 0.039 `distler@cs.fau.de`
  - Alle: `vs@i4.informatik.uni-erlangen.de`

Übungsaufgaben

- Anmeldung (per WAFFEL)

  `https://waffel.informatik.uni-erlangen.de/signup/?univisid=20545691`
- Bearbeitung
  - Persönliches Projektverzeichnis: `/proj/i4vs/<loginname>`
  - Bearbeitung in Gruppen
    - 3 Teilnehmer pro Gruppe
    - Festlegung der Gruppenzugehörigkeit: `/proj/i4vs/bin/vsgroups`
  - Empfehlung: Git-Repository für die gesamte Gruppe → siehe Folie 0–6 ff.
- Abgabe
  - Präsentation der eigenen Implementierung
  - Falls eine Präsentation am Abgabetermin nicht möglich sein sollte: Rechtzeitige Mitteilung an Übungsleiter (per Mail / persönlich)

Überblick

GitLab: Repository erstellen

- Von allen Mitgliedern einer Gruppe durchzuführen
  - Benutzerkonto erstellen: `https://gitlab.cs.fau.de` → „Sign up“
  - Öffentlichen SSH-Schlüssel hinzufügen:
    → Oben rechts auf das Profil-Logo und auf „Settings“ klicken
    → Reiter „SSH Keys“ auswählen
    → Existierenden oder neu erstellten SSH-Schlüssel hinzufügen
    (Bei Nutzung von Eclipse: **Schlüsseltyp RSA** verwenden)
    (siehe auch: `https://gitlab.cs.fau.de/help/ssh/README`)
- Nur durch ein Gruppenmitglied durchzuführen
  - Projekt für Gruppe erstellen
    → Auf „+“-Button und dann „New Project“ klicken
    * **„Visibility Level“ = Private**
  - Gruppenmitglieder hinzufügen
    → Projekt bzw. Repository auswählen
    * „Settings“ → „Members“ auswählen
    * Namen der Gruppenmitglieder eingeben
    * Auswahlbox: „role permission“ (statt „Guest“) auf „Maintainer“ setzen
    * Auf „Add to project“-Button klicken

Lokales Repository erstellen `git clone/git config`

- Erstellen einer **lokalen** Arbeitskopie über ein **entferntes** Repository
  - Befehl: `> git clone <URL>`
  - Beispiel: `git clone` über SSH (SSH-Schlüssel nötig, siehe Folie 0–6)

```
> git clone git@gitlab.cs.fau.de:mustermann/vsue.git
```

  - URL des GitLab-Repository steht auf der jeweiligen Projektübersichtsseite
- Konfiguration (einmalig pro Benutzer notwendig)
  - E-Mail-Adresse und Name für Commits festlegen

```
> git config --global user.name "Max Mustermann"
> git config --global user.email max@mustermann.de
```

  - Dokumentation: `man 1 git-config`
- Weitere Informationen zu Git: `https://git-scm.com/book/en/v2`

Änderungen vormerken und überprüfen `git add/git status`

- Neue Datei(en) / Dateiänderungen für Commit vormerken

```
> git add <file(s)-to-add>
```

↪ Spätere Änderungen müssen erneut explizit vorgemerkt werden

- Vorgemerkte Änderungen überprüfen

```
> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Makefile
```

Änderungen ein-/ausprüfen

`git commit/push/git pull`

- Vorgemerkte Änderungen mittels Aufruf von `commit` dauerhaft in **lokales** Repository übernehmen

```
> git commit
```

- Commits vom lokalen in das **entfernte** Repository einprüfen

```
> git push
```

↪ Lokales Repository muss vorher aktualisiert werden, wenn entferntes Repository weitere, noch nicht lokal vorhandene Commits enthält

- **Lokales** Repository aktualisieren

```
> git pull
```

↪ Zustand aus entferntem Repository holen und lokal integrieren
↪ Eventuell Konfliktauflösung notwendig, siehe nächste Folie

Konfliktbewältigung

- Konflikt feststellen

```
> git pull
[...]
   1b09b5d..39efa77  master -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

> cat README.md
<<<<<<< HEAD
TODO: Structure and fill this README.
=======
## Synopsis

## Installation
>>>>>>> 39efa77d814d4aebfecd37da8d252cfc80091907
```

- Konflikt in Datei manuell auflösen und Ergebnis einprüfen

```
> git add README.md
> git commit
```