

Middleware – Cloud Computing – Übung

Christopher Eibel, Michael Eischer, Tobias Distler

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

Wintersemester 2018/19



Überblick

Organisatorisches
Übung
Versionsverwaltung mit Git



Termine und Ansprechpartner

- Tafelübung
 - Montag: 16:15–17:45 Uhr
 - Raum 0.031-113
- Rechnerübungen
 - Mittwoch: 16:00–18:00 Uhr (ab 24.10.)
 - CIP-Pool: Raum 00.153-113
 - Betreuung (nur) bei Bedarf
- Ansprechpartner
 - Michael Eischer Raum 0.045 eischer@cs.fau.de
 - Tobias Distler Raum 0.039 distler@cs.fau.de
 - Christopher Eibel Raum 0.045 ceibel@cs.fau.de
 - Alle: mw@i4.informatik.uni-erlangen.de



Übungsaufgaben

- Anmeldung
 - Web-Anmeldesystem *Waffel*
 - <https://waffel.informatik.uni-erlangen.de/signup/?univisid=40740560>
- Bearbeitung
 - Persönliches Projektverzeichnis: /proj/i4mw/<loginname>
 - Bearbeitung in Gruppen
 - 3 Teilnehmer pro Gruppe
 - Festlegung der Gruppenzugehörigkeit: /proj/i4mw/bin/mwgroups
→ Antwort-E-Mail: Gruppennummer, für Aufgaben benötigte Zugangsdaten
 - 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)



Organisatorisches

Übung

Versionsverwaltung mit Git



- 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 (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 „Master“ 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/mwue.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 ebenfalls 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
```



- 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



- 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  
>>>>>> 39efa77d814d4aebfec37da8d252cfc80091907
```

- Konflikt in Datei manuell auflösen und Ergebnis einprüfen

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

