GIT UND CI

Übersicht

- Versionskontrolle – Version Control System (VCS)
- Bekannte (alte) Systeme
- Git – verteilte Versionskontrolle
  - Grundlegendes
  - Branches
  - Forks
- Gitlab CI für automatisches Testen

Motivation
Wozu ein VCS?

- Protokollieren von Änderungen
- Nachvollziehbar: *Wer, Wann, Was*
- Gemeinsam an den gleichen Dingen arbeiten

- … und warum all das?
  - Erkennen welche Änderungen zu Fehlern geführt haben
  - Vielfältiges ”experimentieren” – mit anschließenden *roll-back*
  - Genaues *taggen* von Softwareversionen via commit hash

Formen der Versionsverwaltung

- Lokale Versionsverwaltung
  - Nur lokal auf dem Rechner/Platte
  - `rcs` (revision control system)
  - Uralt… (1982)

- Zentrale Versionsverwaltung
  - Ein (!) zentrales Repository (remote, lokal)
  - `cvs` (concurrent versioning system, 1990)
  - `svn` (subversion, 2000)
  - Typischerweise ein hash/snapshot pro Datei

- Verteilte Versionsverwaltung
  - Jeder Entwickler hat (vollständiges) lokales Repository
  - Zentrale Server sind optional (z.B. GitHub, BitBucket, GitLab, ...)
  - `git` (2005), mercurial
  - Hashes/snapshots für jede Revision

Geschichtliches

- Entwickelt von Linus Torwalds … der mal wieder mit etwas unzufrieden war.

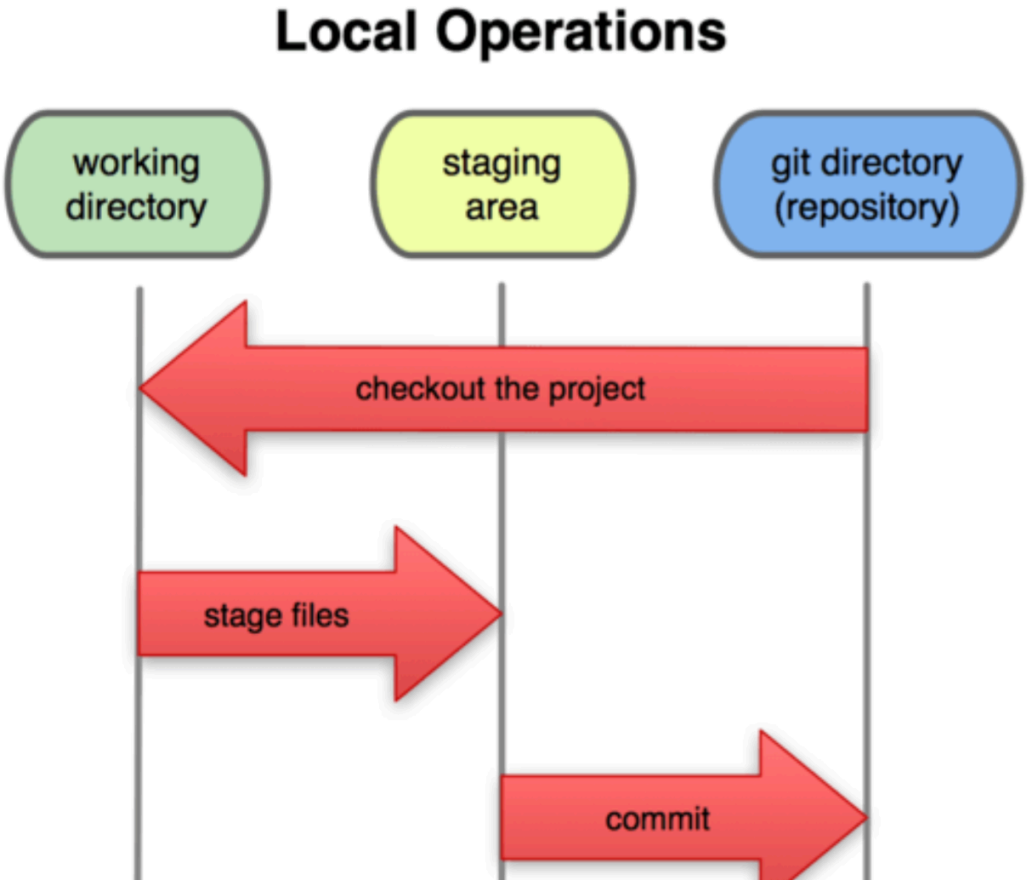
- Erklärte Ziele:
  - Unterstützung verteilter Arbeitsabläufe
  - Hohe Sicherheit gegen unbeabsichtigte oder böswillige Verfälschung
  - Hohe Effizienz

- Rapide Entwicklung
  - Erste Arbeiten am 3. April 2005
  - Erste Veröffentlichung am 6. April
  - ”Bootstrapped” ab dem 7. April (also als VCS das sich selbst versioniert)

Grundlegende Operationen auf lokalem Repository

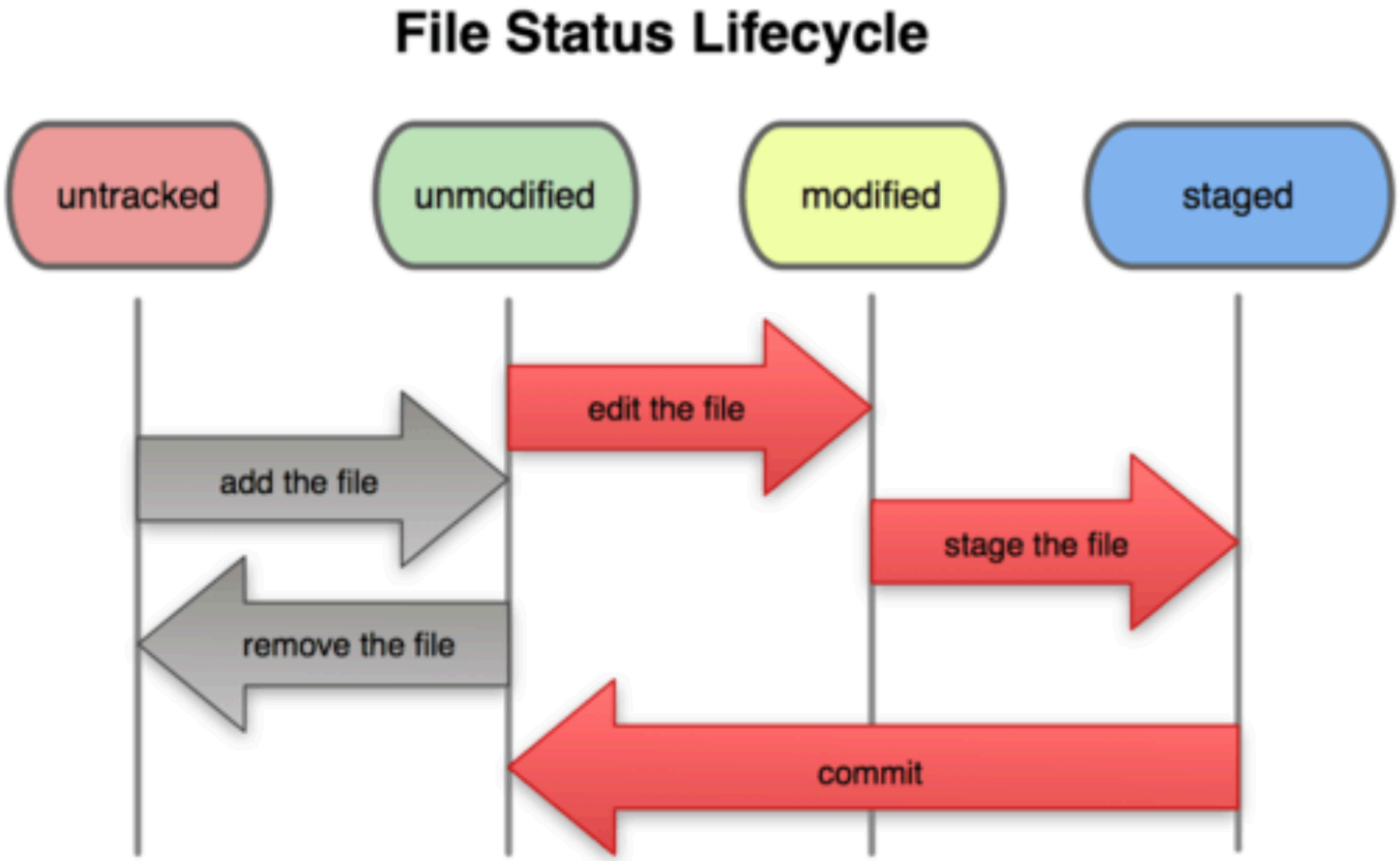
- `git init` — Initialisiert ein lokales Repository
- `git add <file>` — Merkt eine (geänderte) Datei für Versionskontrolle vor
- `git commit` — Fixiert einen aktuellen Stand basierend auf vorangegangene git add Befehle
- `git branch` — Erstellt eine Verzweigung, zunächst vom gleichen Stand
- `git checkout <branch>` — Wechselt in eine/n andere/n Version/Branch
- `git merge <branch>` — Zusammenbringen von unterschiedlichen Versionen

Ablauf (high leven)



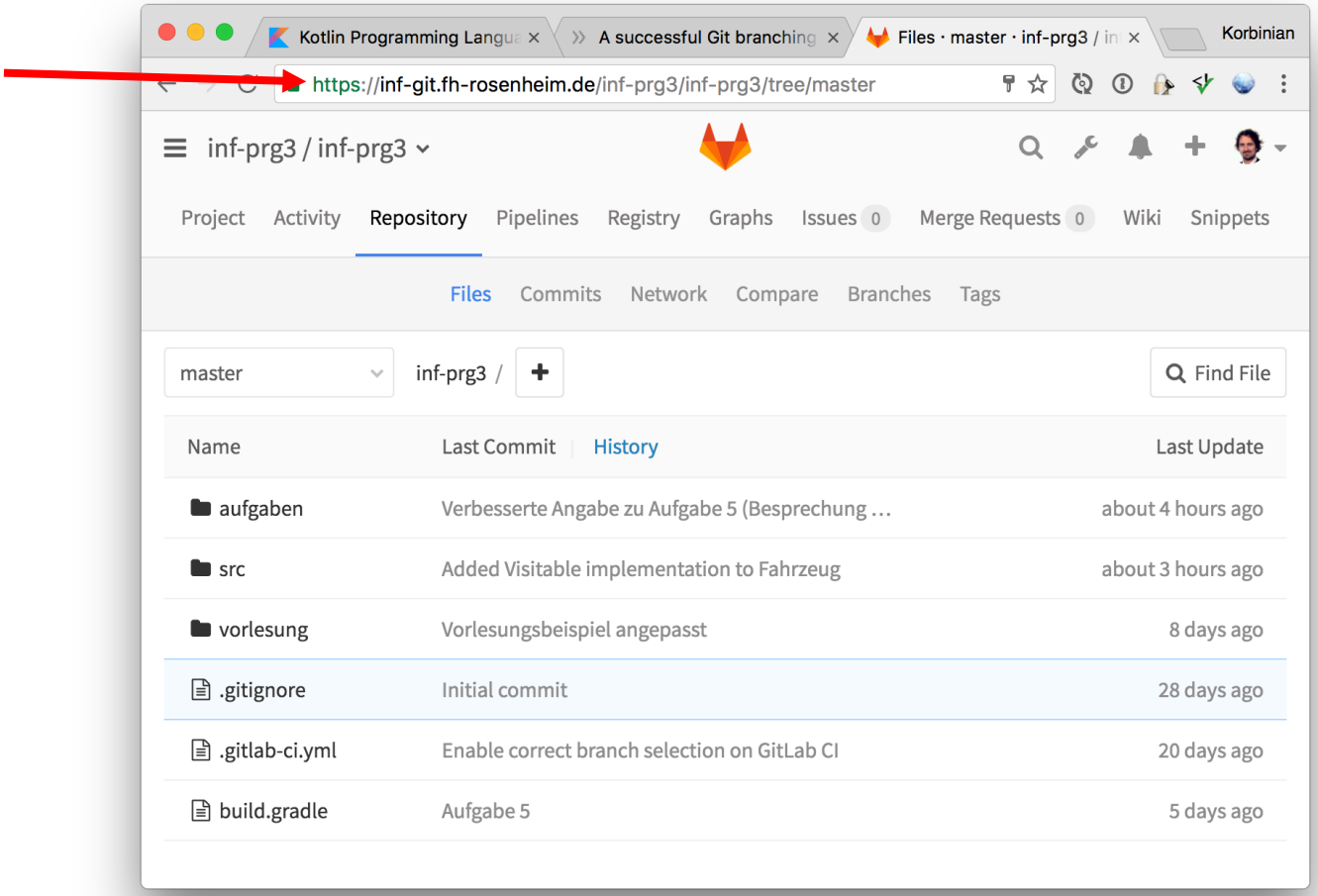
Jens Sandmann, 2013

Ablauf (low-level)



Jens Sandmann, 2013

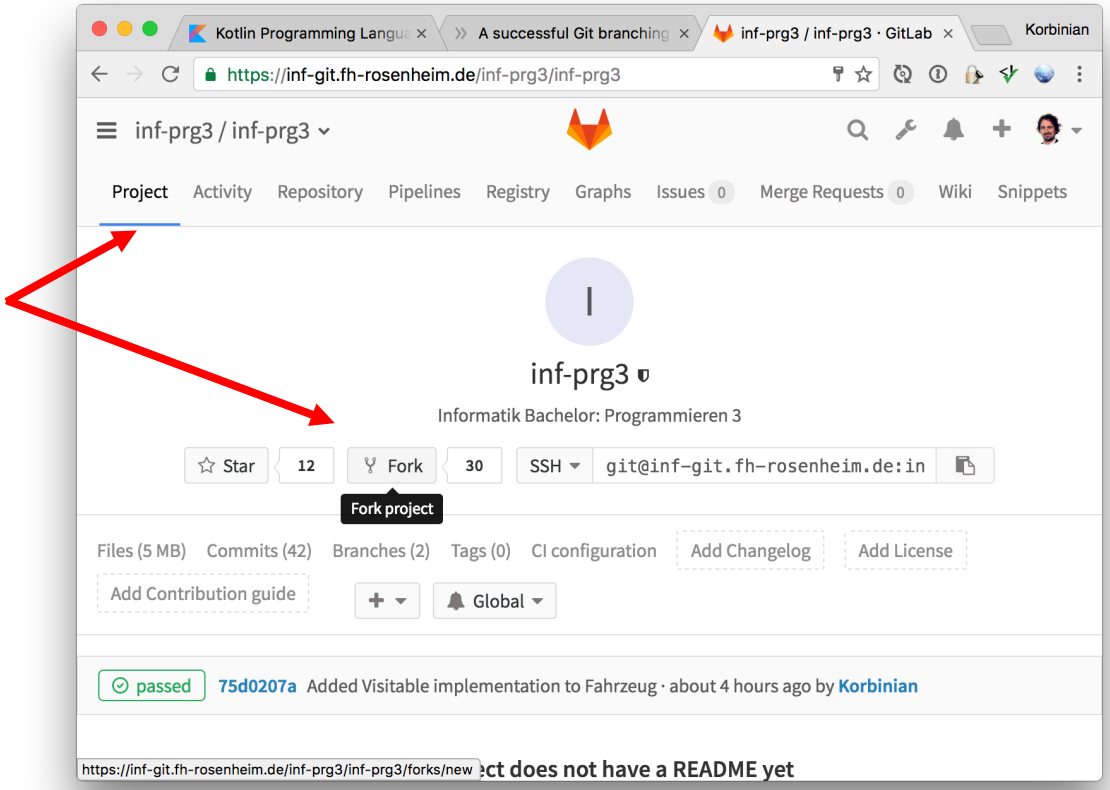
Remote Repositories



Grundlegende Operationen auf/mit *remote* Repositories

- `git clone <repo>` — Ein remote Repository lokal klonen; dies ist meist der erste Schritt (statt `git init`).
- `git push` — Lokale *commits (und branches)* nach remote pushen; nur wenn berechtigt/autorisiert!
- `git add remote <name> <path>` — Ein weiteres remote Repository mit Alias hinzufügen (wichtig, wenn das originale Repo ein *fork* ist)

Forks
Abspaltung vom “Original”

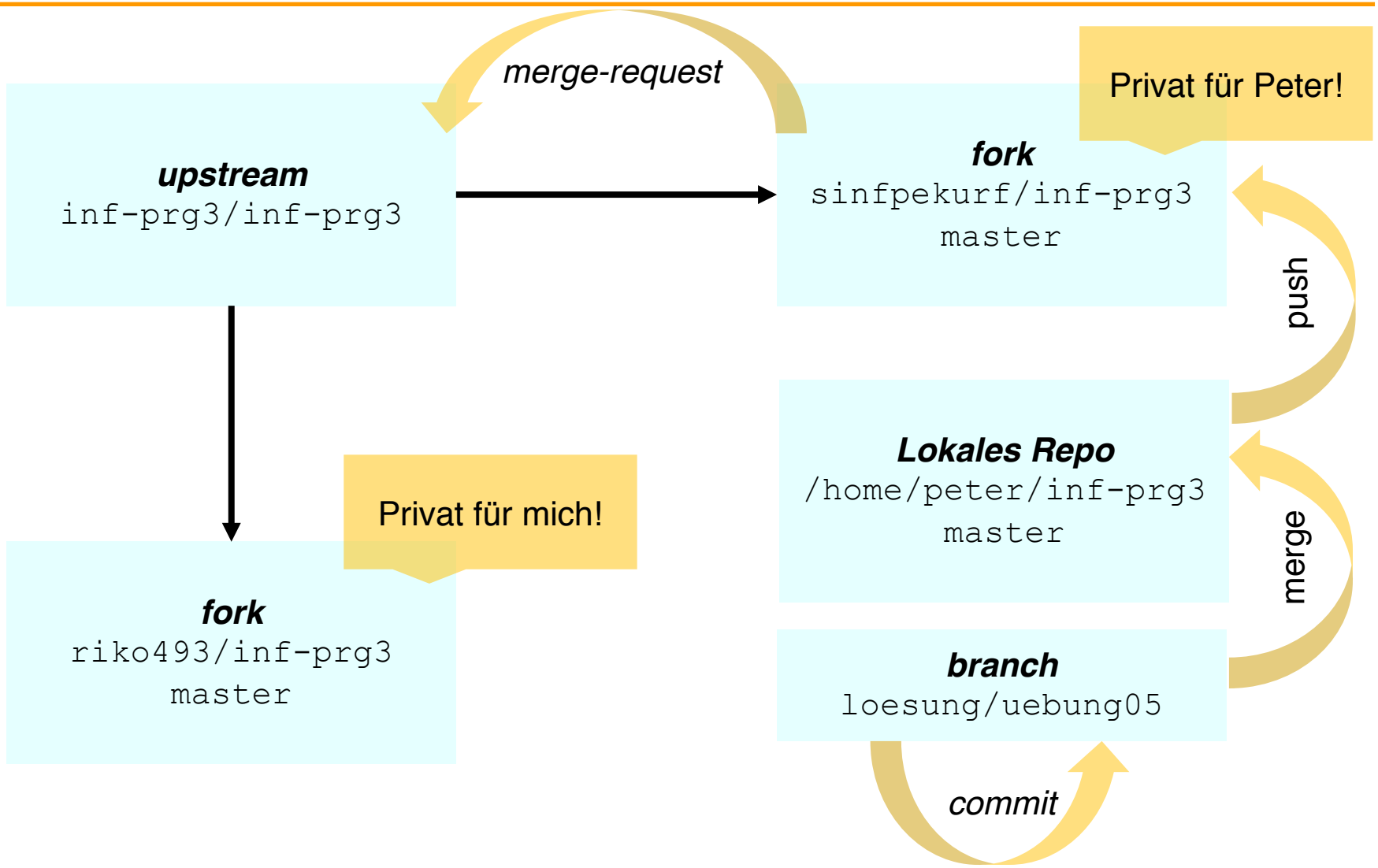


- Erstellt ein “eigenes” remote Repo als Kopie vom “Original” (“*upstream*”)
  - Typischerweise nur über Weboberfläche

Fork synchronisieren (Ablauf)

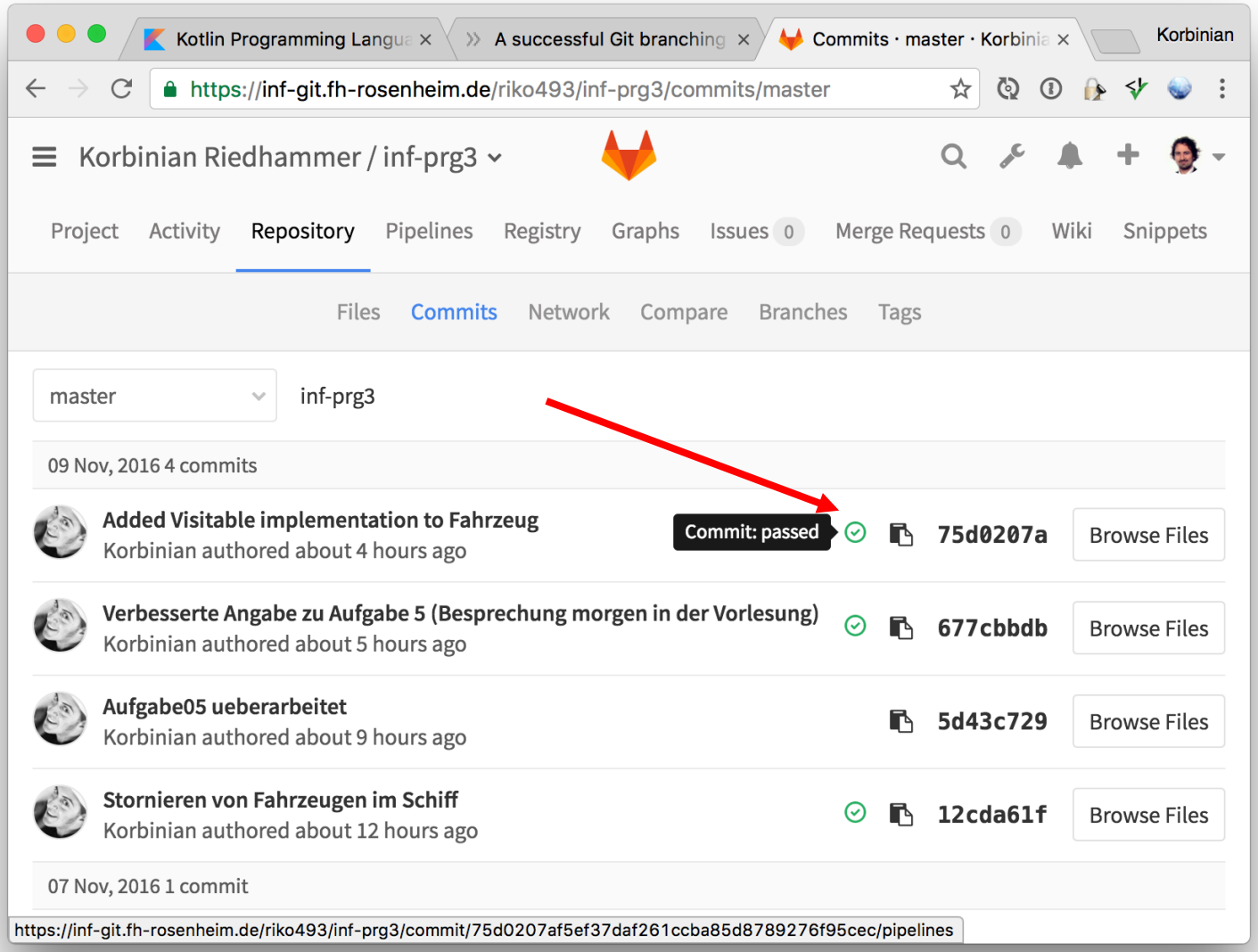
- `git checkout master`
  - Im eigenen lokalen Repo auf master wechseln
- `git remote add upstream <upstream-url>`
  - Ein (weiteres) remote Repo hinzufügen mit Alias `upstream`
- `git fetch upstream`
  - Alle Änderungen des Repo `upstream` holen
- `git merge upstream/master`
  - Master branch des upstream's in den lokalen master mergen
- Um auch lokale branches aktuell zu halten, fuer jeden solcen:
  - `git checkout branchname`
  - `git merge master`

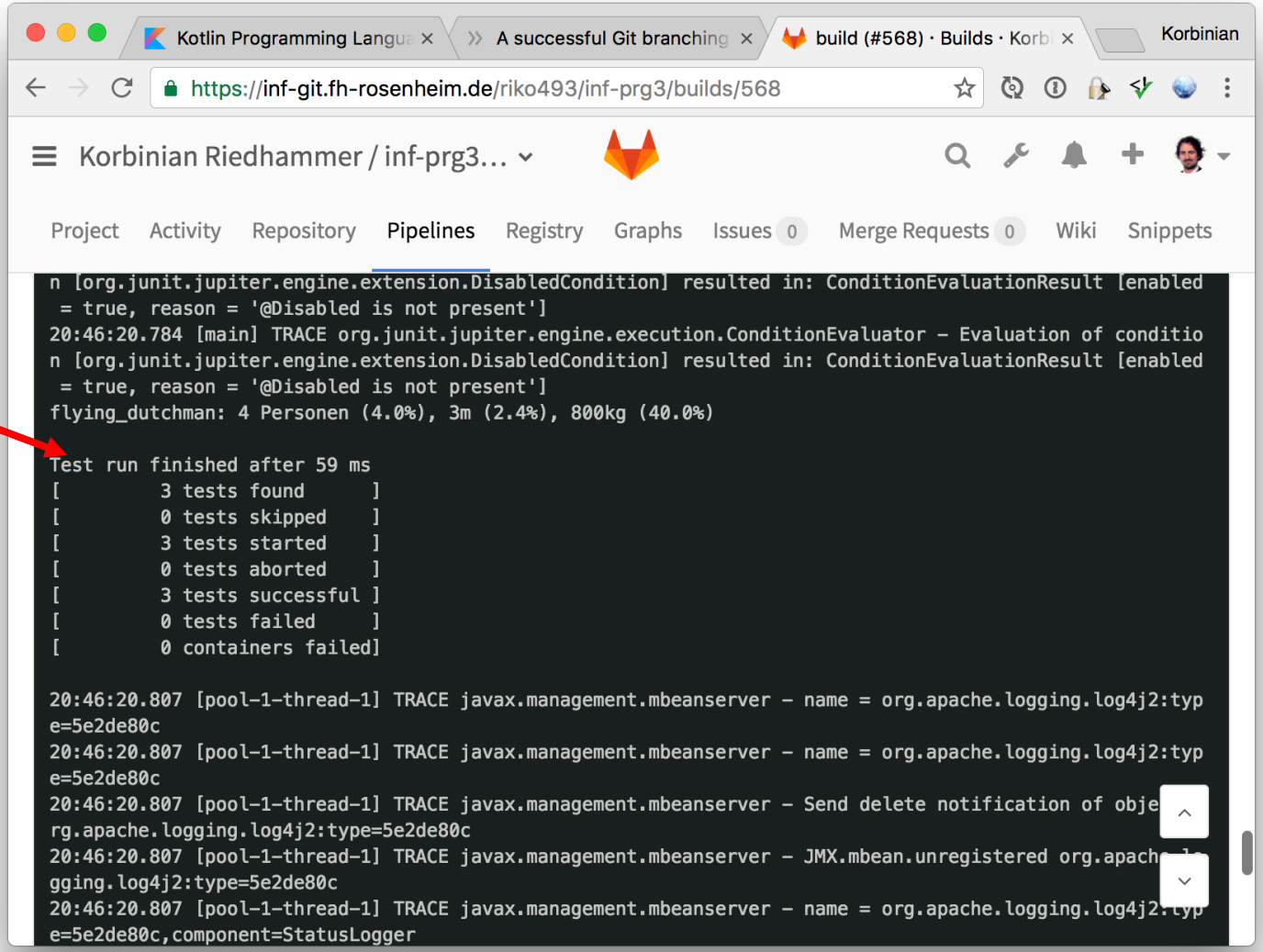
Workflow im Ganzen



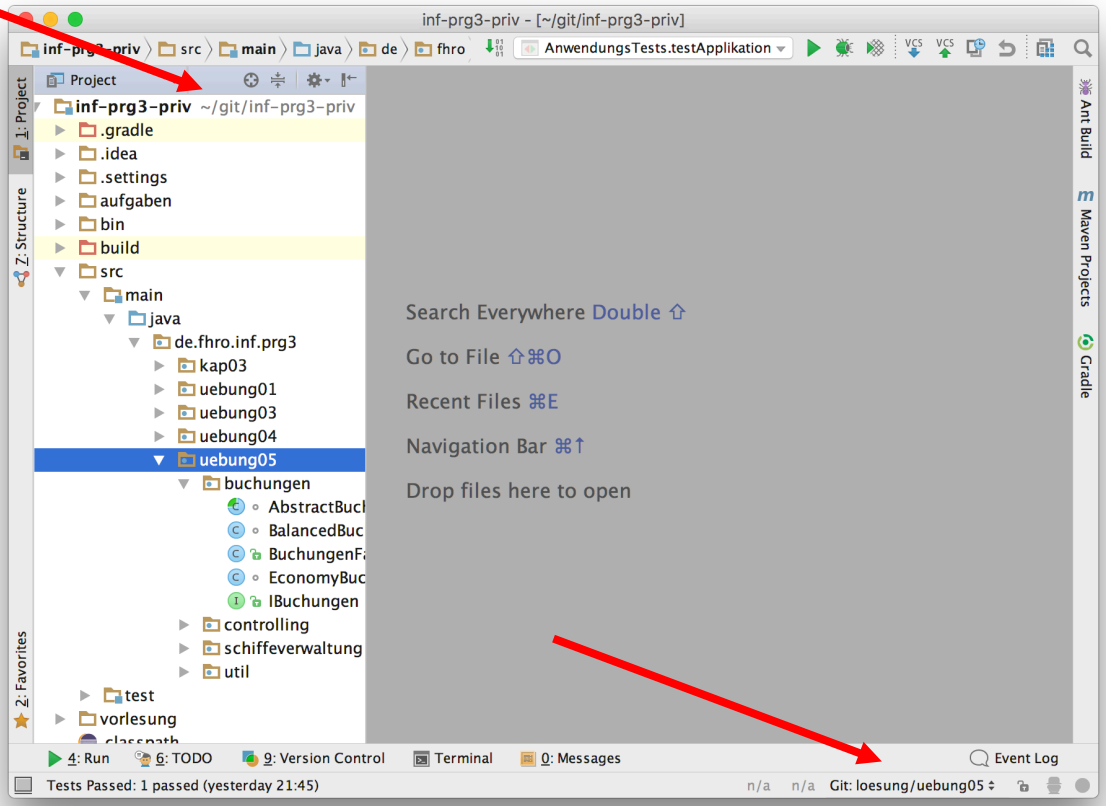
Gitlab CI

- Familiarität mit git (bzw. Versionierung im Allgemeinen)
- Privater Fork für Übungsgruppen (2-3 Studenten)
- Regelmäßige Synchronisierung mit `upstream`
- Bei commit & push in eigenes Repository läuft eine Serie von automatischen Tests
  - Wenn das `package` der aktuellen Übung im `branch` Namen steht…
    also z.B. loesung/uebung05
  - Dann werden alle `JUnit` Tests in dem `package` `de.fhro.inf.prg3.uebung05.tests` auf dem Server (!) ausgeführt


Kotlin Programming Langua
A successful Git branching
Commits · master · Korbinia
Korbinian
https://inf-git.fh-rosenheim.de/riko493/inf-prg3/commits/master
Korbinian Riedhammer / inf-prg3
Project
Activity
Repository
Pipelines
Registry
Graphs
Issues 0
Merge Requests 0
Wiki
Snippets
Files
Commits
Network
Compare
Branches
Tags
master
inf-prg3
09 Nov, 2016 4 commits
Added Visitable implementation to Fahrzeug
Korbinian authored about 4 hours ago
Commit: passed
75d0207a
Browse Files
Verbesserte Angabe zu Aufgabe 5 (Besprechung morgen in der Vorlesung)
Korbinian authored about 5 hours ago
677cbbdb
Browse Files
Aufgabe05 ueberarbeitet
Korbinian authored about 9 hours ago
5d43c729
Browse Files
Stornieren von Fahrzeugen im Schiff
Korbinian authored about 12 hours ago
12cda61f
Browse Files
07 Nov, 2016 1 commit
https://inf-git.fh-rosenheim.de/riko493/inf-prg3/commit/75d0207af5ef37daf261ccba85d8789276f95cec/pipelines


Kotlin Programming Langua
A successful Git branching
build (#568) · Builds · Korbi
Korbinian
https://inf-git.fh-rosenheim.de/riko493/inf-prg3/builds/568
Korbinian Riedhammer / inf-prg3...
Project
Activity
Repository
Pipelines
Registry
Graphs
Issues 0
Merge Requests 0
Wiki
Snippets
n [org.junit.jupiter.engine.extension.DisabledCondition] resulted in: ConditionEvaluationResult [enabled
= true, reason = '@Disabled is not present']
20:46:20.784 [main] TRACE org.junit.jupiter.engine.execution.ConditionEvaluator - Evaluation of conditio
n [org.junit.jupiter.engine.extension.DisabledCondition] resulted in: ConditionEvaluationResult [enabled
= true, reason = '@Disabled is not present']
flying_dutchman: 4 Personen (4.0%), 3m (2.4%), 800kg (40.0%)
Test run finished after 59 ms
[3 tests found]
[0 tests skipped]
[3 tests started]
[0 tests aborted]
[3 tests successful]
[0 tests failed]
[0 containers failed]
20:46:20.807 [pool-1-thread-1] TRACE javax.management.mbeanserver - name = org.apache.logging.log4j2:typ
e=5e2de80c
20:46:20.807 [pool-1-thread-1] TRACE javax.management.mbeanserver - name = org.apache.logging.log4j2:typ
e=5e2de80c
20:46:20.807 [pool-1-thread-1] TRACE javax.management.mbeanserver - Send delete notification of obje
rg.apache.logging.log4j2:type=5e2de80c
20:46:20.807 [pool-1-thread-1] TRACE javax.management.mbeanserver - JMX.mbean.unregistered org.apach
gging.log4j2:type=5e2de80c
20:46:20.807 [pool-1-thread-1] TRACE javax.management.mbeanserver - name = org.apache.logging.log4j2:typ
e=5e2de80c,component=StatusLogger

IntelliJ



- IDE hilft mit VCS
  - Projekt direkt von VCS auschecken (`clone`)
  - Geänderte/Neue Dateien für `commit` vormerken