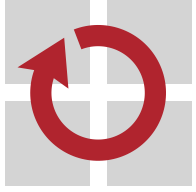
# Übung zu Betriebssysteme

## Organisation und Einführung

18. & 20. Oktober 2017

Andreas Ziegler
Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

# Organisation des Übungsbetriebs

**OOStuBS**

single-core

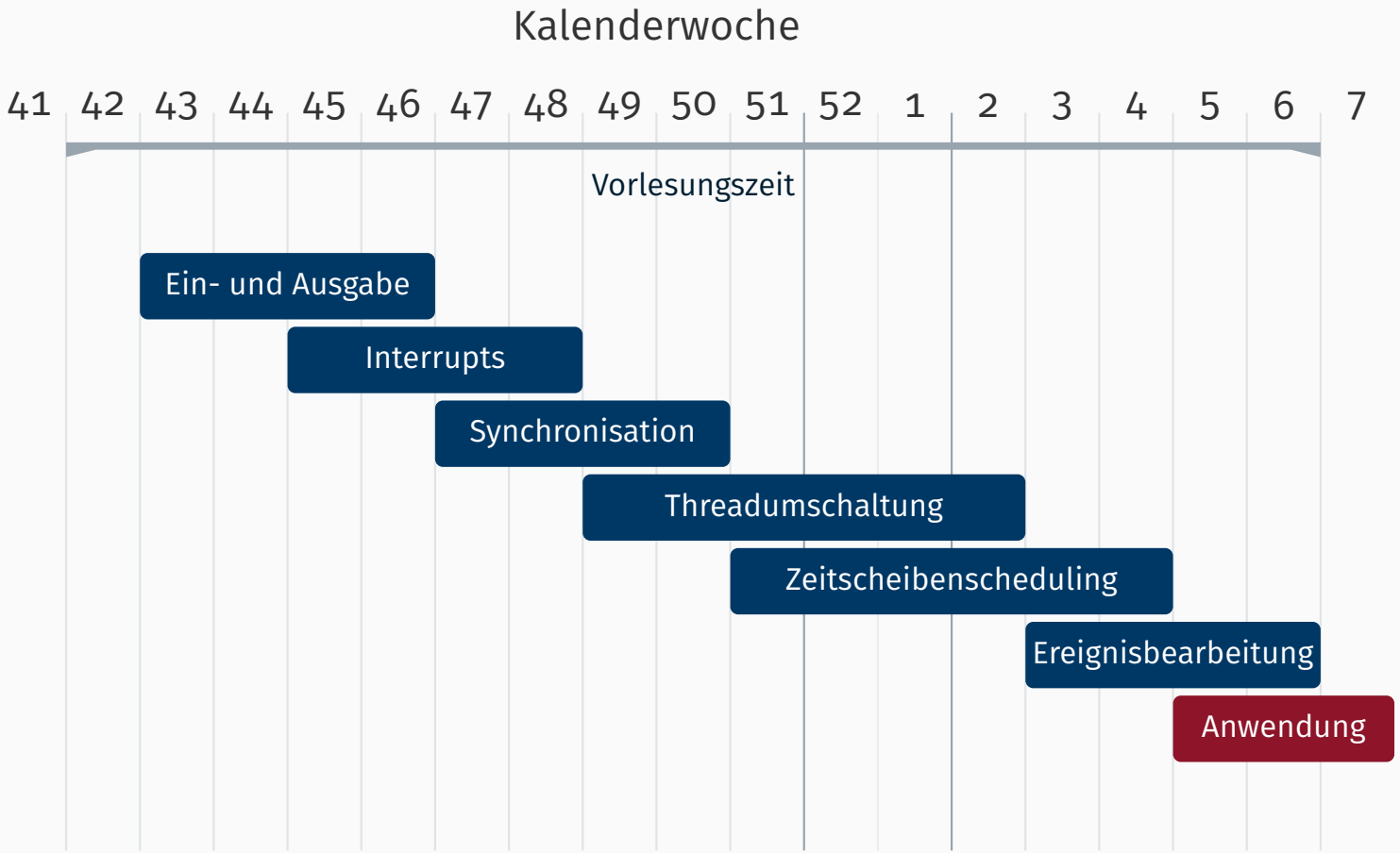
5 ECTS Modul

**MPStuBS**

multi-core

7.5 ECTS Modul

# Zeitplan


Kalenderwoche
41 42 43 44 45 46 47 48 49 50 51 52 1 2 3 4 5 6 7
Vorlesungszeit
Ein- und Ausgabe
Interrupts
Synchronisation
Threadumschaltung
Zeitscheibenscheduling
Ereignisbearbeitung
Anwendung

# Zeitplan

## Oktober 2017

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | **18** | 19 | **20** | 21 | 22 |
| 23 | 24 | **25** | 26 | **27** | 28 | 29 |
| 30 | 31 | | | | | |

## November 2017

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | **8** | 9 | **10** | 11 | 12 |
| 13 | 14 | **15** | 16 | **17** | 18 | 19 |
| 20 | 21 | **22** | 23 | **24** | 25 | 26 |
| 27 | 28 | **29** | 30 | | | |

## Dezember 2017

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | **1** | 2 | 3 |
| 4 | 5 | **6** | 7 | **8** | 9 | 10 |
| 11 | 12 | **13** | 14 | **15** | 16 | 17 |
| 18 | 19 | **20** | 21 | **22** | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

## Januar 2018

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | **10** | 11 | **12** | 13 | 14 |
| 15 | 16 | **17** | 18 | **19** | 20 | 21 |
| 22 | 23 | **24** | 25 | **26** | 27 | 28 |
| 29 | 30 | **31** | | | | |

## Februar 2018

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | 1 | **2** | 3 | 4 |
| 5 | 6 | **7** | 8 | **9** | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | | | | |

**Tafelübung**
für neue Aufgabe
im Aquarium

Abgabe der Aufgabe in
der **Rechnerübung**
im Huber-CIP

- Abgabe nur in festen **2er Gruppen**
- eine (obligatorische) Tafelübung pro Aufgabe
- Anmeldung zur Tafelübung (bis 31. Oktober) im Waffel auf **waffel.informatik.uni-erlangen.de/signup?course=319**
- Informationen und Aufgabenstellung auf **www4.cs.fau.de/Lehre/WS17/V_BS/**
- Quelltextvorlagen der Aufgaben im Gitlab

  `https://gitlab.cs.fau.de/i4/oostubs`
  `https://gitlab.cs.fau.de/i4/mpstubs`

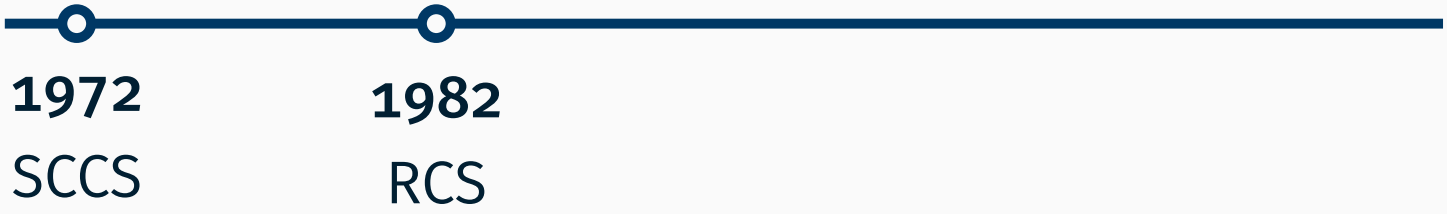
# Kommunikation

- Rechnerübung
- Mail an **bsstud@lists.informatik.uni-erlangen.de**
- **#faui4bs** im IRCnet
- XMPP-MUC **i4bs@conference.cs.fau.de**
- **Raum 0.055** in der Martensstr. 1 (begründeter Notfall)

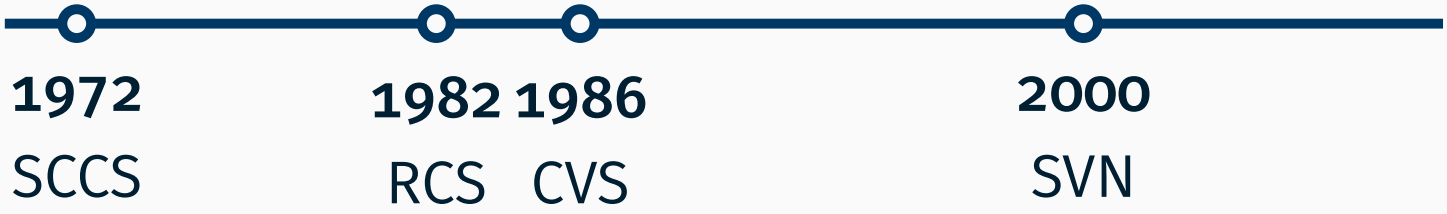
# Eine kleine Einführung in Versionsverwaltungssysteme

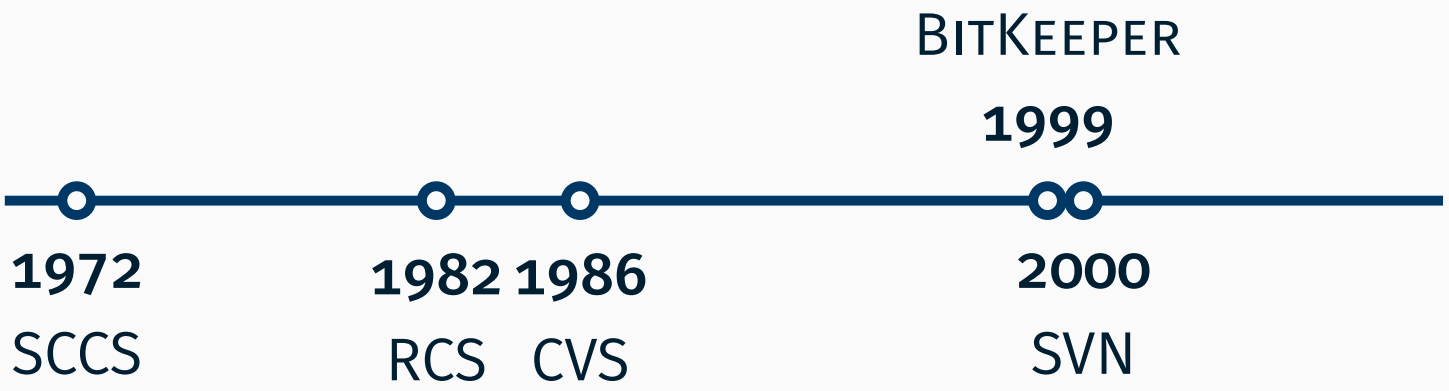
**1972**

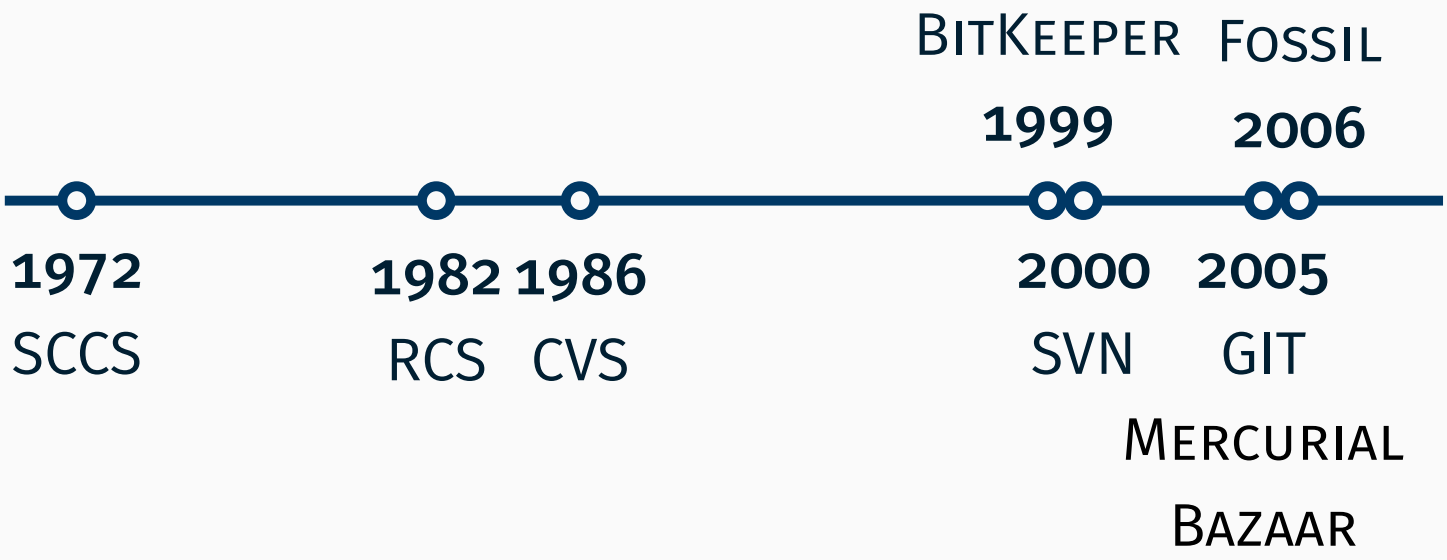
SCCS

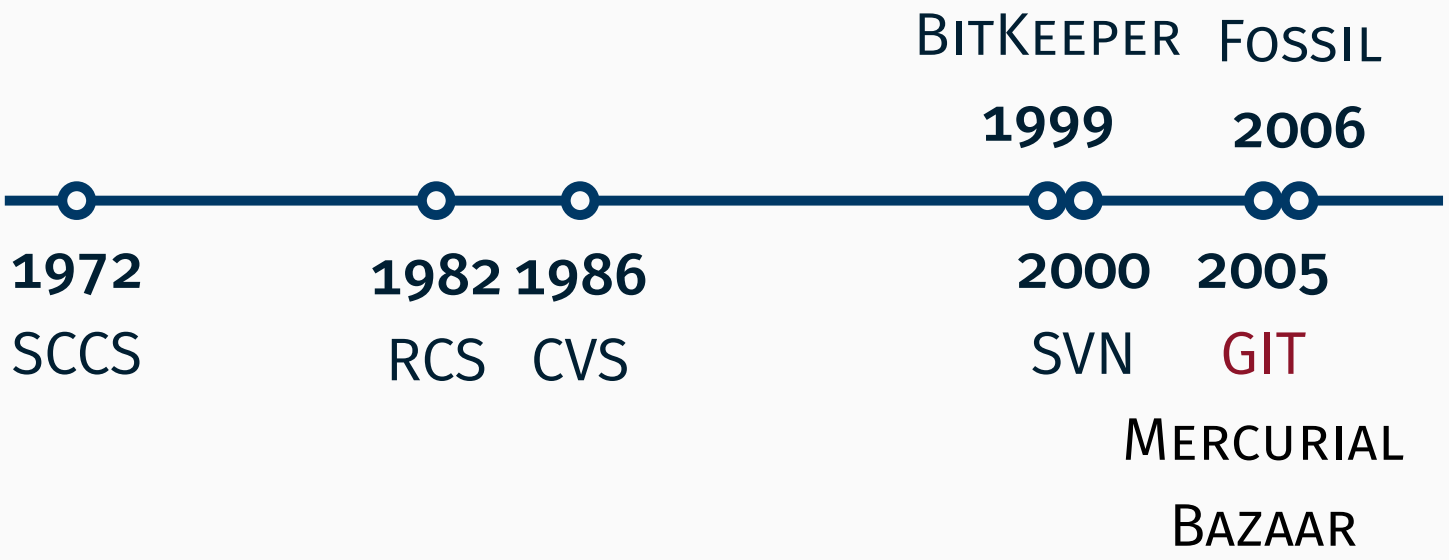

1972
SCCS
1982
RCS


1972
SCCS
1982
RCS
1986
CVS


1972
SCCS
1982
RCS
1986
CVS
2000
SVN

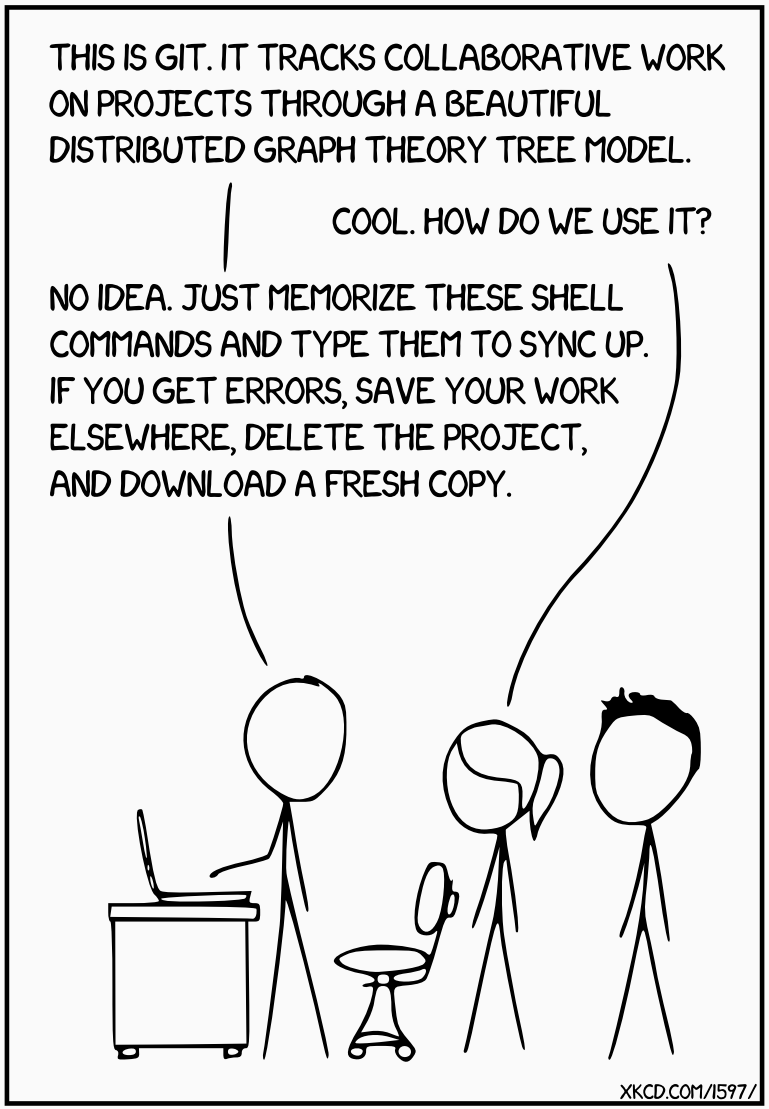

BitKeeper
1999
1972
SCCS
1982
RCS
1986
CVS
2000
SVN


BitKeeper
Fossil
1999
2006
1972
1982
1986
2000
2005
SCCS
RCS
CVS
SVN
GIT
Mercurial
Bazaar


BitKeeper
Fossil
1999
2006
1972
1982
1986
2000
2005
SCCS
RCS
CVS
SVN
GIT
Mercurial
Bazaar

# Schlüsselkonzepte von GIT

- Integrität durch SHA-1 Hash
- vollständiges Speichern der Daten (*snapshot*)
- dezentral (*clone*)
- nicht-lineare Entwicklung (*branch*)

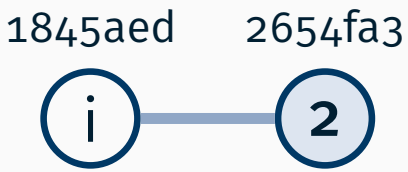

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.
XKCD.COM/1597/

```
heinloth:~$ mkdir beispiel
heinloth:~$ cd beispiel
heinloth:~/beispiel$ git init
Leeres Git-Repository in  /beispiel/.git/ initialisiert
```

```
heinloth:~/beispiel$ touch README.md
heinloth:~/beispiel$ git add README.md
heinloth:~/beispiel$ git commit -m "initialer commit"
[master (Basis-Commit) 1845aed] initialer commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```

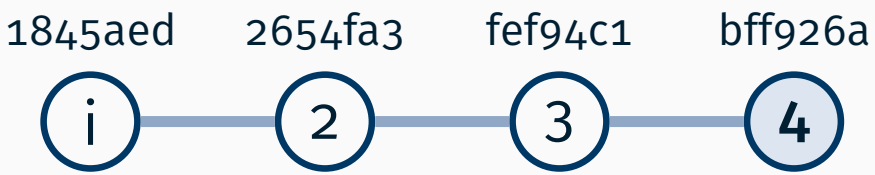
1845aed

```
01 heinloth:~/beispiel$ echo "23" > foo
02 heinloth:~/beispiel$ git add foo
03 heinloth:~/beispiel$ git commit -m "hier sollte eine
       gute nachricht stehen"
04 [master 2654fa3] hier sollte eine gute nachricht stehen
05  1 file changed, 1 insertion(+)
06  create mode 100644 foo
```

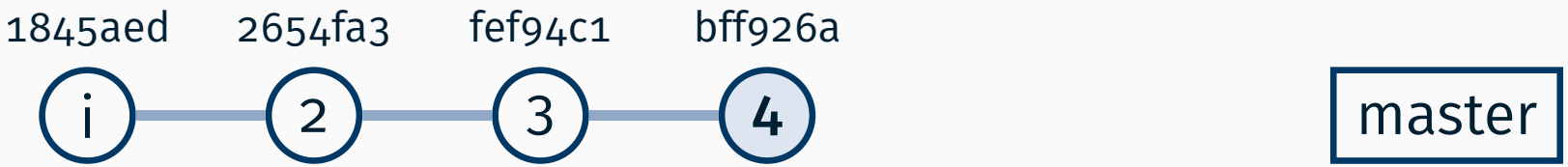


```
01 heinloth:~/beispiel$ echo "42" > foo
02 heinloth:~/beispiel$ git add foo
03 heinloth:~/beispiel$ git commit -m "noch mal ein commit"
04 [master fef94c1] noch mal ein commit
05  1 file changed, 1 insertion(+), 1 deletion(-)
```

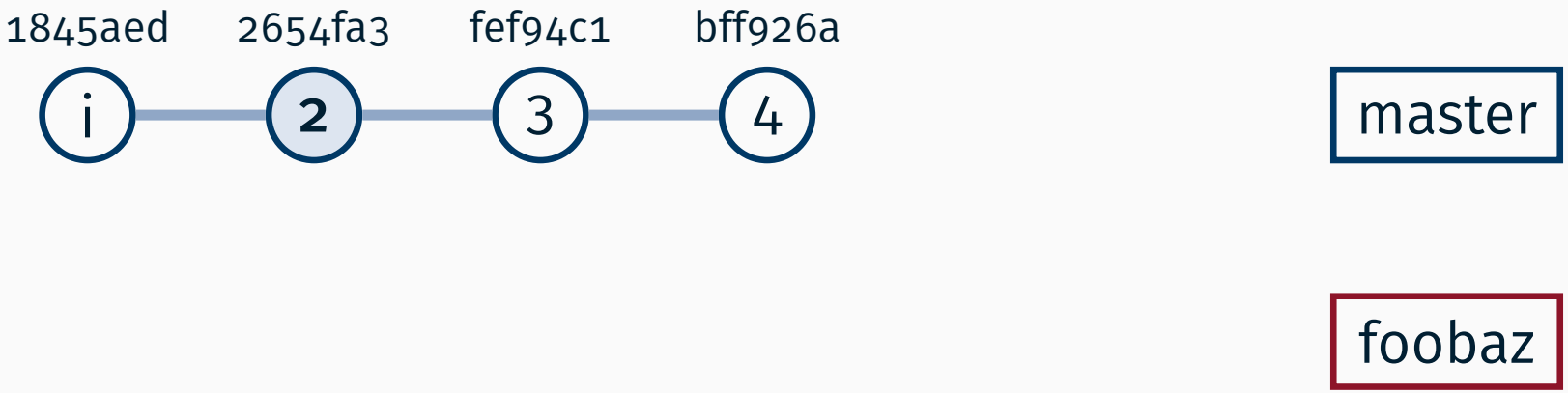
```
01 heinloth:~/beispiel$ echo "1337" > bar
02 heinloth:~/beispiel$ echo "zweiundvierzig" > foo
03 heinloth:~/beispiel$ git add bar foo
04 heinloth:~/beispiel$ git commit -m "aenderungsnachricht"
05 [master bff926a] aenderungsnachricht
06  2 files changed, 2 insertions(+), 1 deletion(-)
07  create mode 100644 bar
```



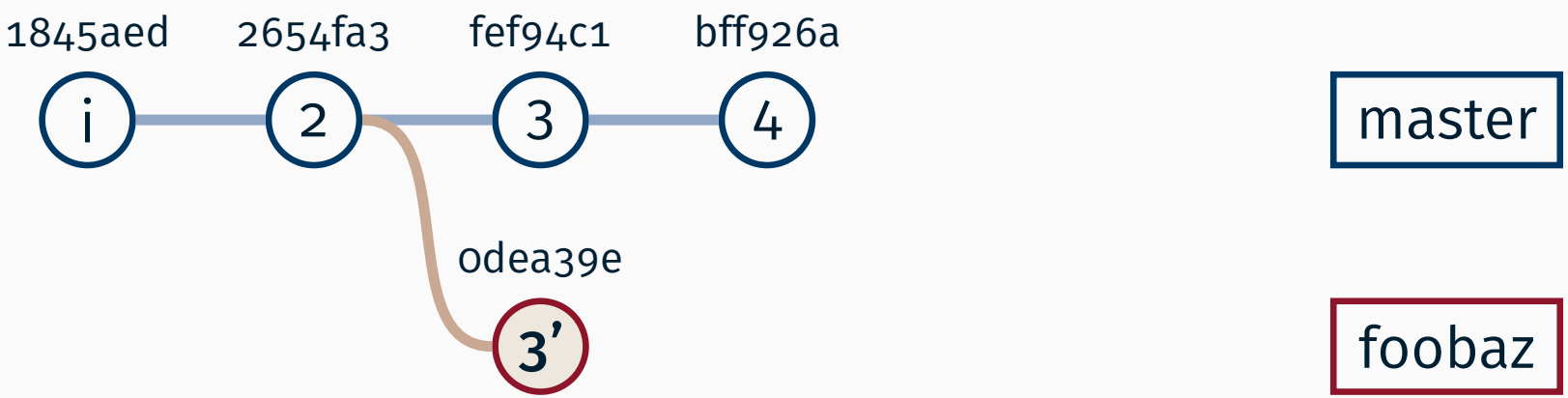
```
heinloth:~/beispiel$ git status
Auf Branch master
nichts zu committen, Arbeitsverzeichnis unverändert
```



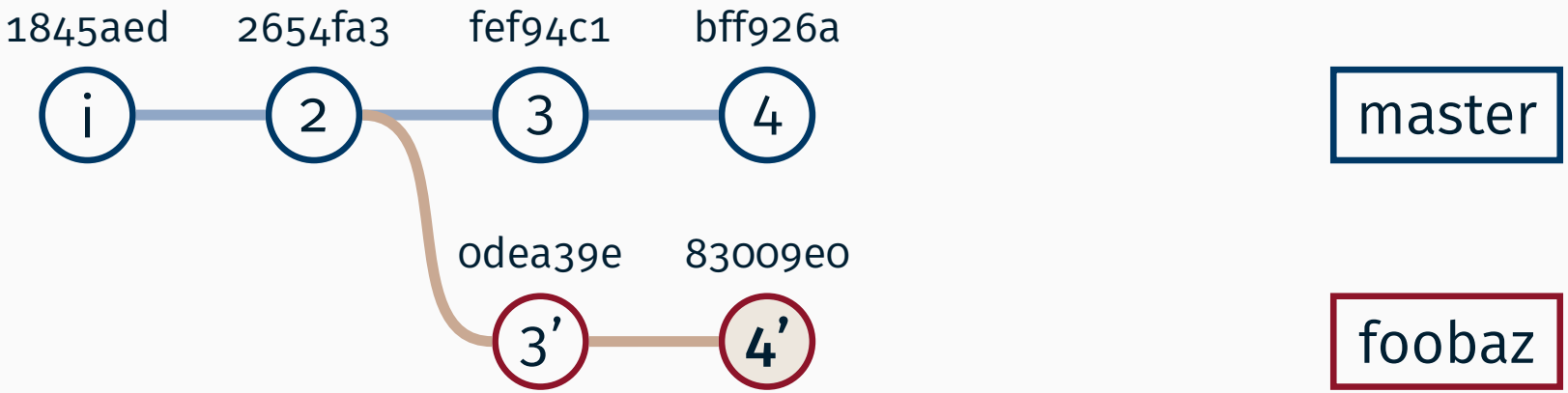
```
heinloth:~/beispiel$ git checkout -b foobaz 2654fa3
Gewechselt zu einem neuem Branch 'foobaz'
heinloth:~/beispiel$ git shortlog
Bernhard Heinloth (2):
 initialer commit
 hier sollte eine gute nachricht stehen
```



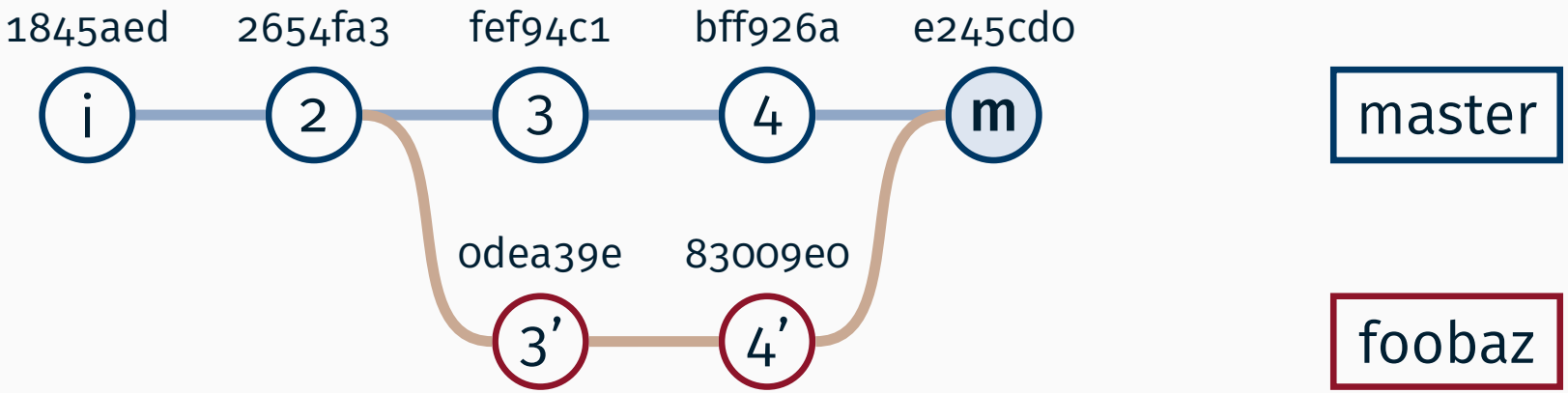
```
heinloth:~/beispiel$ echo "3.141" > baz
heinloth:~/beispiel$ git add .
heinloth:~/beispiel$ git commit -m "neue datei"
[foobaz 0dea39e] neue datei
 1 file changed, 1 insertion(+)
 create mode 100644 baz
```

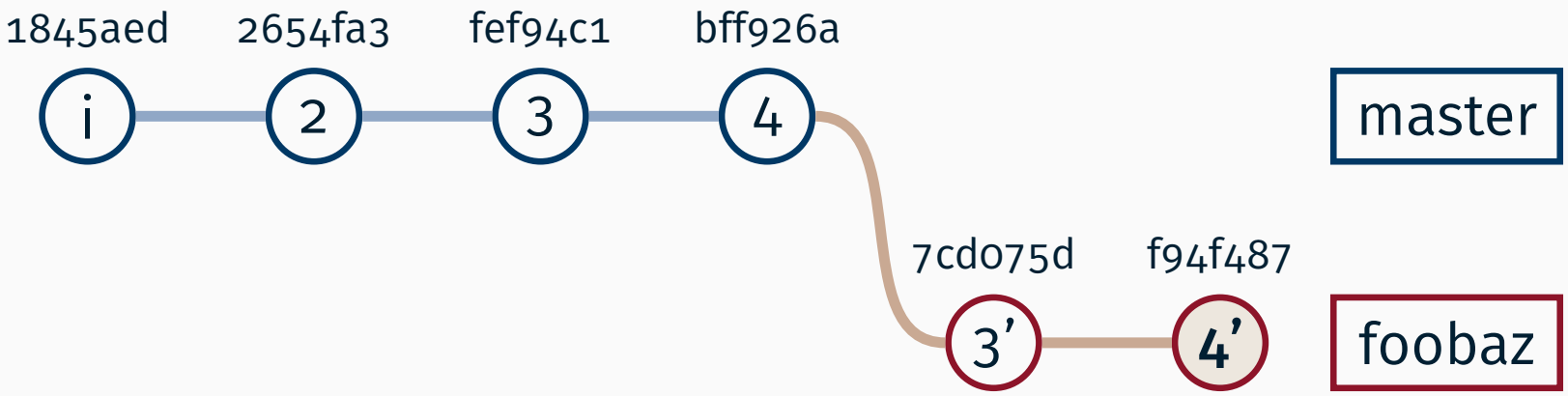


```
heinloth:~/beispiel$ echo "pi" > baz
heinloth:~/beispiel$ git commit -am "blub"
[foobaz 83009e0] blub
 1 file changed, 1 insertion(+), 1 deletion(-)
```



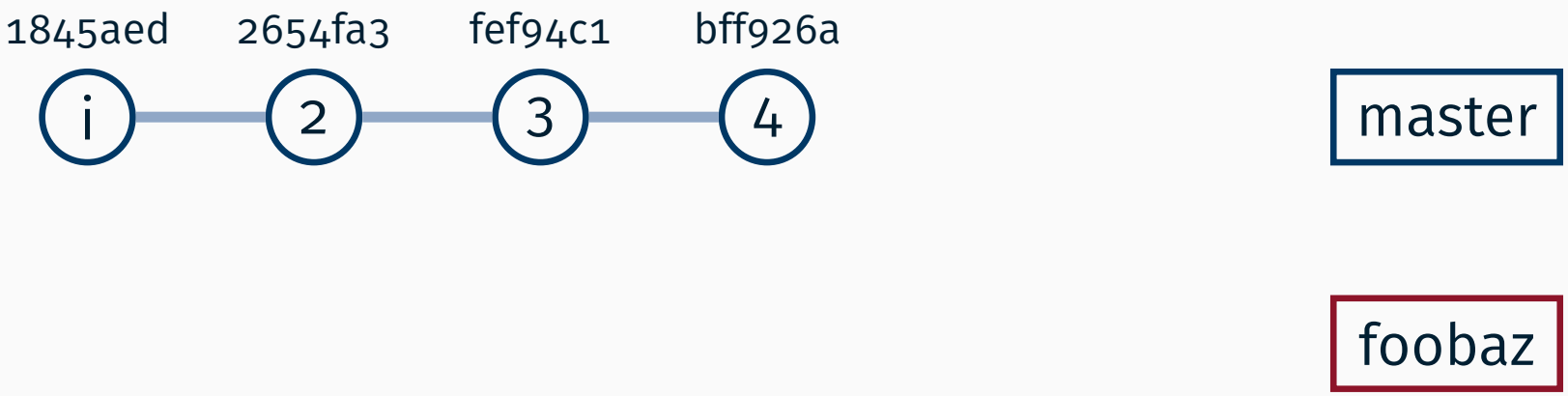
```
heinloth:~/beispiel$ git checkout master
heinloth:~/beispiel$ git merge foobaz -m "merge commit"
Merge made by the 'recursive' strategy.
 baz | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 baz
```




1845aed
2654fa3
fef94c1
bff926a
1
2
3
4
master
7cd075d
f94f487
3'
4'
foobaz

# Alternativ: Die Geschichte neu schreiben

```
01 heinloth:~/beispiel$ git rebase master
02 Zunächst wird der Branch zurückgespult,
03 um Ihre Änderungen darauf neu anzuwenden...
04  Wende an: blub
05  Wende an: neue datei
```

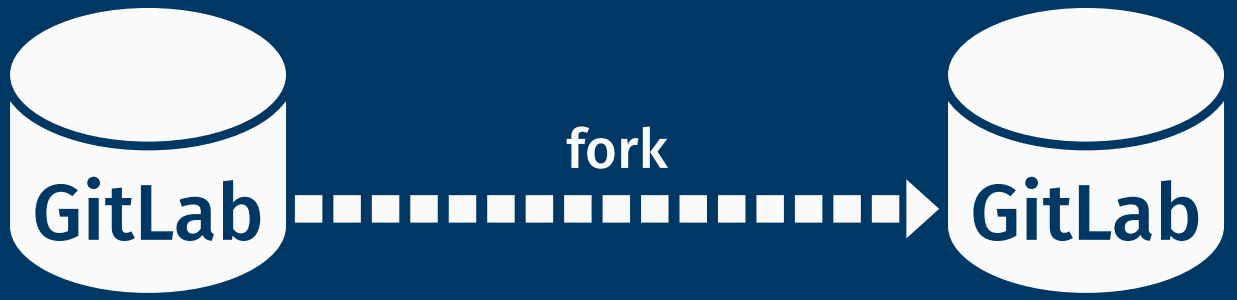


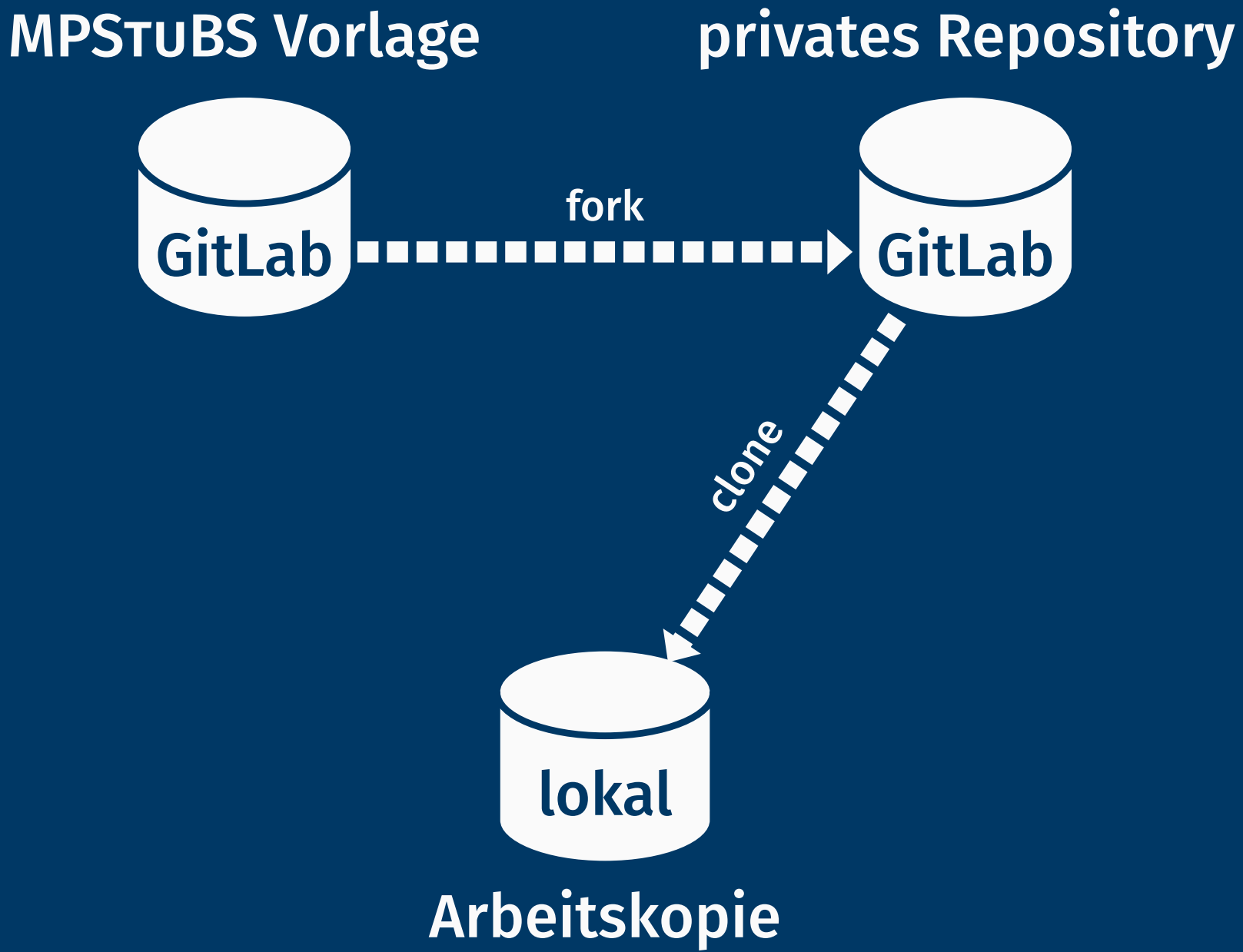
# Workflow

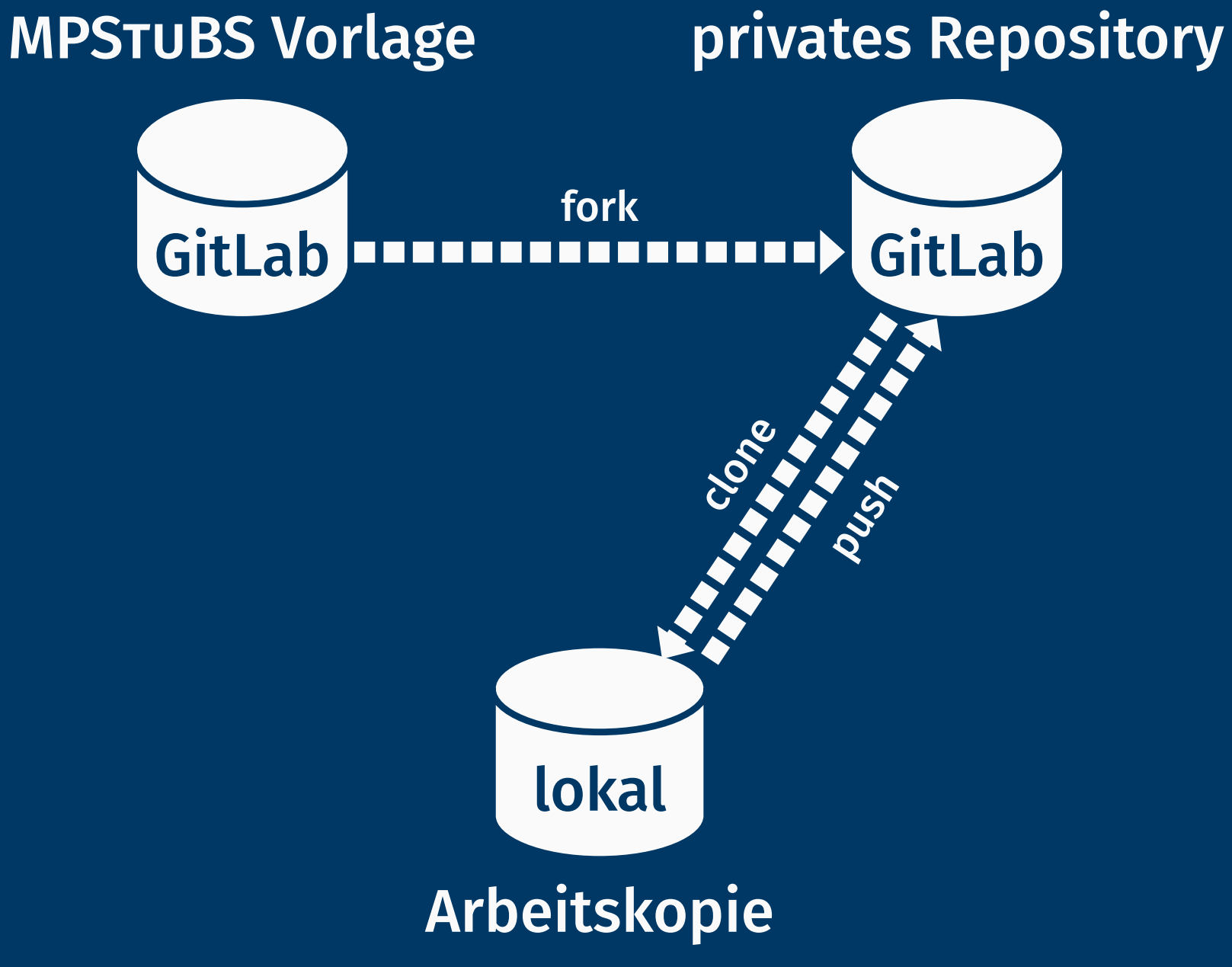
# MPStuBS Vorlage

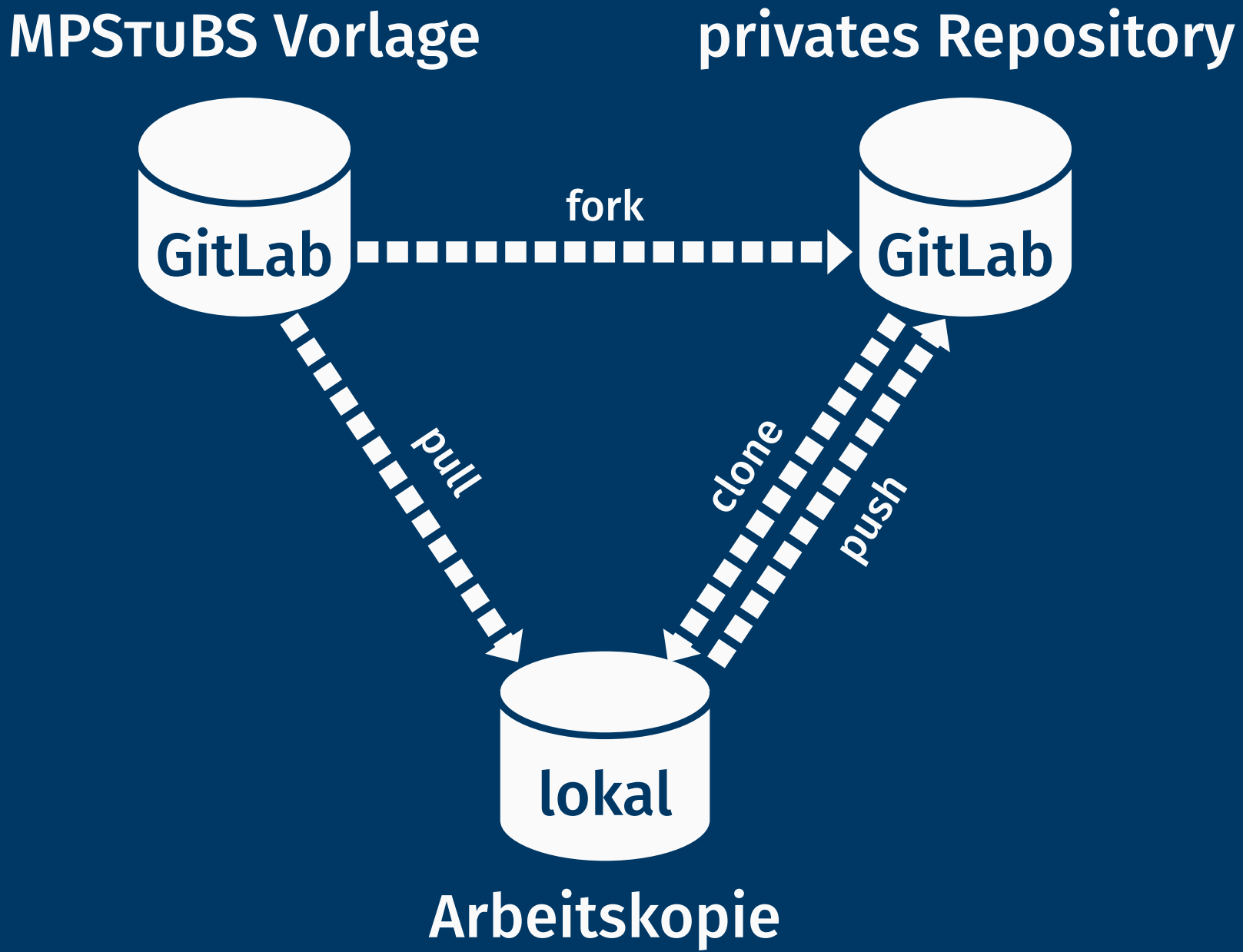
MPStuBS Vorlage

privates Repository




MPStuBS Vorlage
privates Repository
GitLab
fork
GitLab
clone
lokal
Arbeitskopie


MPStuBS Vorlage
privates Repository
GitLab
fork
GitLab
clone
push
lokal
Arbeitskopie


MPStuBS Vorlage
privates Repository
GitLab
fork
GitLab
pull
clone
push
lokal
Arbeitskopie

# Cheatsheet

| | |
|---|---|
| **git init** | neues Repository im aktuellen Verzeichnis erstellen |
| **git add *Datei*** | Datei als Kandidat für den nächsten *commit* markieren |
| **git commit** | Änderungen versionieren |
| **git diff** | unversionierte Änderungen anzeigen |
| **git show** | neuste (versionierte) Änderungen anzeigen |
| **git status** | Änderungen zum Vorgänger anzeigen |
| **git branch** | verfügbare Zweige anzeigen |
| **git log** | Historie anzeigen |
| **man git-*Option*** | Hilfe anzeigen, z.B. `man git-add` |

# Cheatsheet (entfernte Quellen)

| | |
|---|---|
| **git clone *URL*** | initiales Kopieren von einer Quelle |
| **git fetch *Name*** | Änderungen aus entfernter Quelle holen |
| **git pull *Name*** | kurz für holen und zusammenfügen |
| **git checkout *Zweig*** | Aktuellen Zweig wechseln |
| **git remote add *URL*** | entfernte Quellen hinzufügen |
| **git push *Name*** | in entfernte Quelle übertragen |

# Fragen?