Übung zu Betriebssysteme

Git Crashkurs

Wintersemester 2020/21

Bernhard Heinloth & Christian Eichler

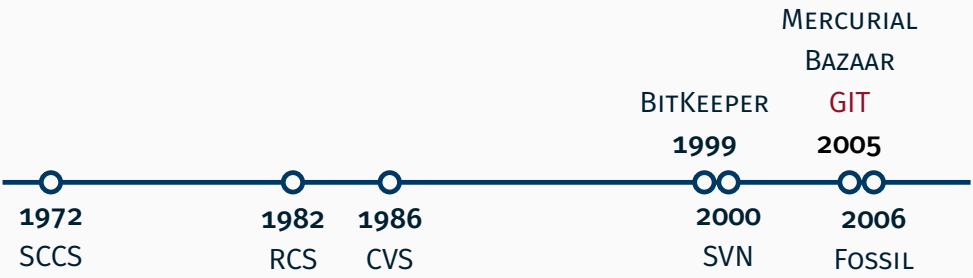
Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Eine kleine Einführung in Versionsverwaltungssysteme


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

Schlüsselkonzepte von GIT

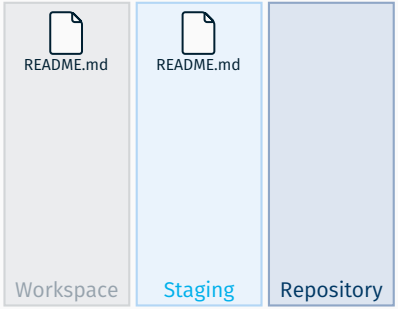
- nicht-lineare Entwicklung (*branch*)
- Integrität durch SHA-1 Hash
- vollständiges Speichern der Daten (*snapshot*)
- dezentral (*clone*)

Lokales GIT Repository initialisieren

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

Dateien mit GIT verwalten

```
~/beispiel$ touch README.md
~/beispiel$ git add README.md
```

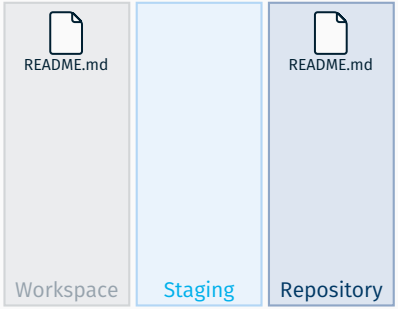


Dateien mit GIT verwalten

bd2de5c

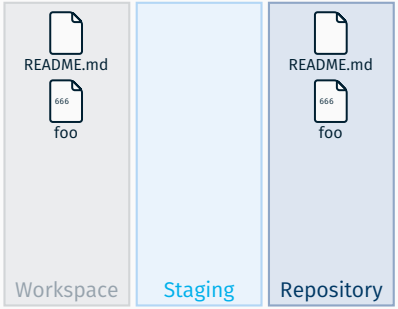
1

```
~/beispiel$ touch README.md
~/beispiel$ git add README.md
~/beispiel$ git commit -m "Liesmich hinzugefügt"
[master (Root-Commit) bd2de5c] Liesmich hinzugefügt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```



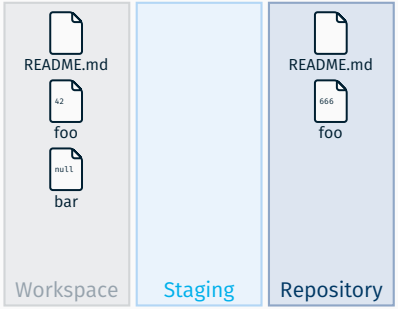
Dateien mit GIT verwalten


```
~/beispiel$ echo "666" > foo
~/beispiel$ git add foo
~/beispiel$ git commit -m "Datei foo erstellt"
[master df7aa5a] Datei foo erstellt
 1 file changed, 1 insertion(+)
 create mode 100644 foo
```



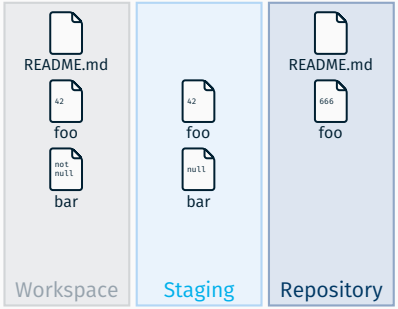
Dateien mit GIT verwalten


```
~/beispiel$ echo "42" > foo
~/beispiel$ echo "null" > bar
~/beispiel$ git status
Auf Branch master
Änderungen, die nicht zum Commit vorgemerkt sind:
    geändert: foo

Unversionierte Dateien:
    bar

keine Änderungen zum Commit vorgemerkt
```



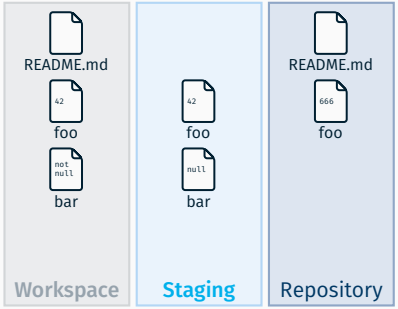
Dateien mit GIT verwalten


```
~/beispiel$ git add foo bar
~/beispiel$ echo "not null" > bar
~/beispiel$ git status
Auf Branch master
Zum Commit vorgemerkte Änderungen:
    neue Datei: bar
    geändert: foo

Änderungen, die nicht zum Commit vorgemerkt sind:
    geändert: bar
```



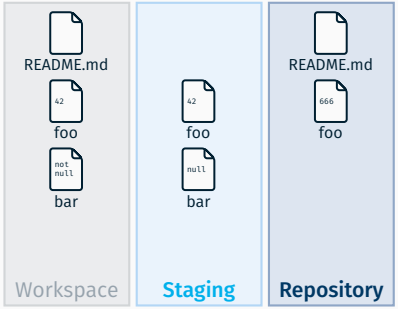
Dateien mit GIT verwalten


```
~/beispiel$ git diff
diff --git a/bar b/bar
index 19765bd..b263a85 100644
--- a/bar
+++ b/bar
@@ -1 +1 @@
-null
+not null
```



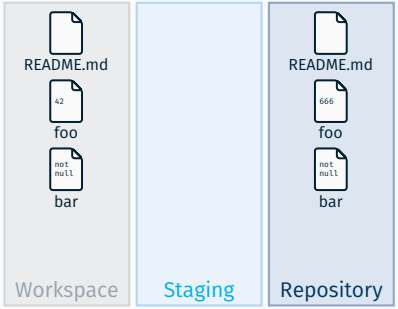
Dateien mit GIT verwalten


```
~/beispiel$ git diff --staged
diff --git a/bar b/bar
new file mode 100644
index 0000000..19765bd
--- a/bar
+++ b/bar
@@ -0,0 +1 @@
+null
diff --git a/foo b/foo
index 7cc86ad..d81cc07 100644
[...]
```



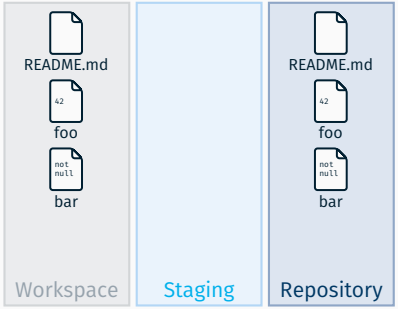
Dateien mit GIT verwalten


```
~/beispiel$ git add bar
~/beispiel$ git commit -m \
            "Foo korrigiert und Bar erstellt"
[master 90f7cfe] Foo korrigiert und Bar erstellt
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 bar
```

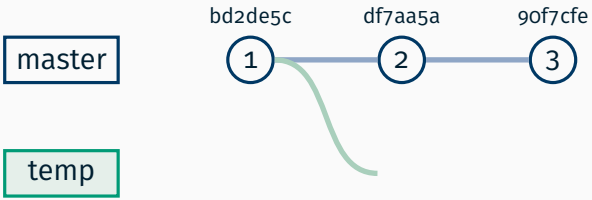


Dateien mit GIT verwalten

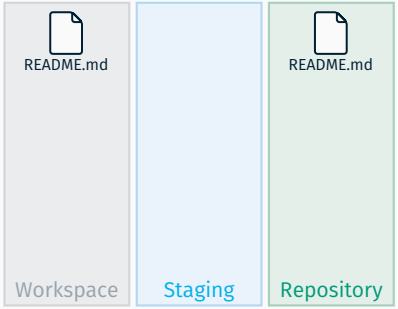

```
~/beispiel$ git shortlog
Bernhard Heinloth (3):
    Liesmich hinzugefügt
    Datei foo erstellt
    Foo korrigiert und Bar erstellt
```



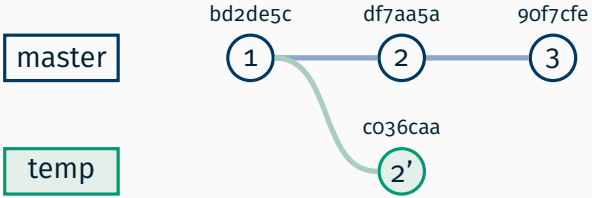
GIT Zweige



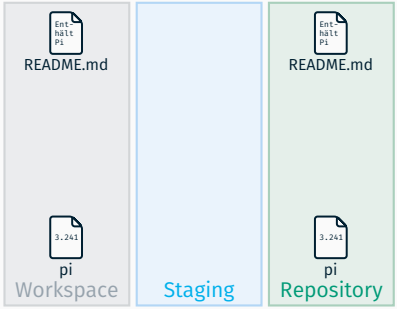

```
~/beispiel$ git branch temp bd2de5c
~/beispiel$ git checkout temp
Zu Zweig »temp« gewechselt
~/beispiel$ git shortlog
Bernhard Heinloth (1):
    Liesmich hinzugefügt
```



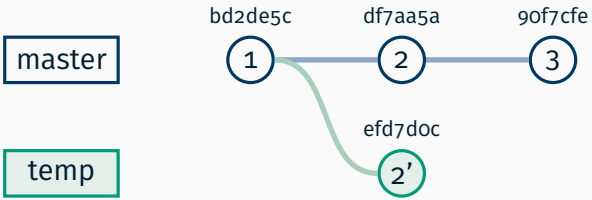
GIT Zweige



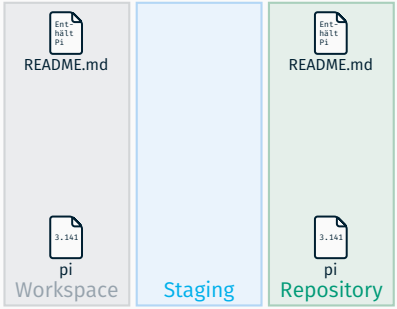

```
~/beispiel$ echo "3.241" > pi
~/beispiel$ echo "Beinhaltet Pi" >> README.md
~/beispiel$ git add .
~/beispiel$ git commit -m "Kreiszahl hinzugefügt"
[temp c036caa] Kreiszahl hinzugefügt
 2 files changed, 2 insertions(+)
 create mode 100644 pi
```



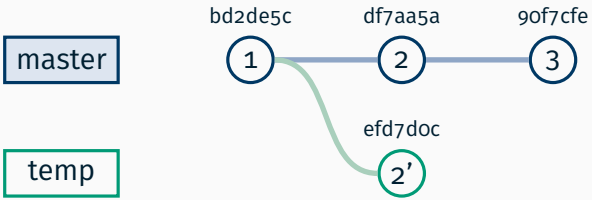
GIT Zweige



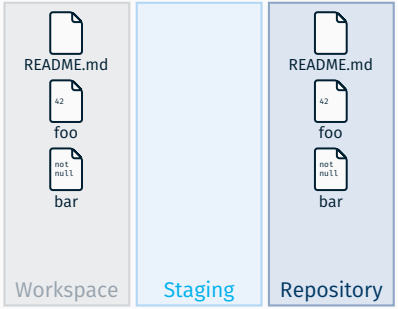

```
~/beispiel$ echo "3.141" > pi
~/beispiel$ git commit -a --amend -m \
                    "Kreiszahl ergänzt"
[temp efd7d0c] Kreiszahl ergänzt
 Date: Mon Oct 5 12:50:08 2020 +0200
 2 files changed, 2 insertions(+)
 create mode 100644 pi
```



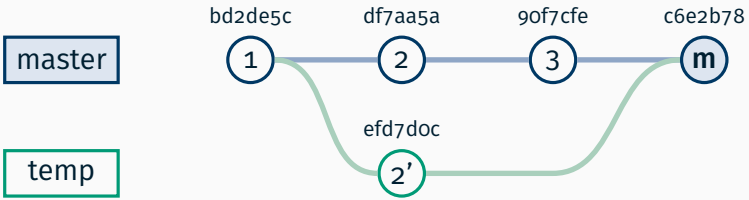
GIT Zweige



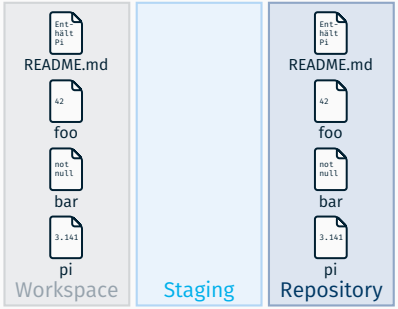

```
~/beispiel$ git branch
  master
* temp
~/beispiel$ git checkout master
~/beispiel$ git branch
* master
  temp
```



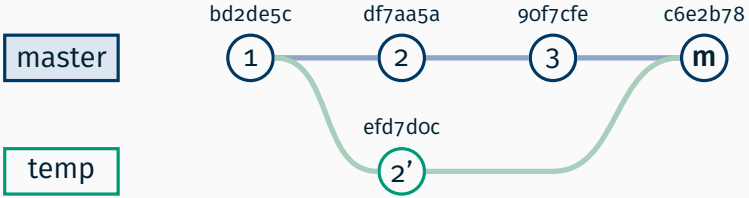
GIT Zweige zusammenführen



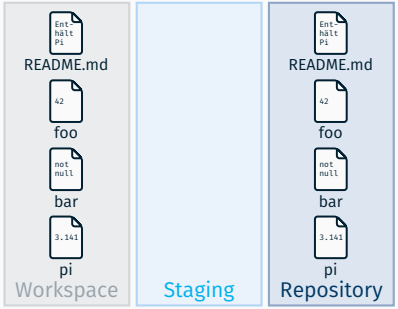

```
~/beispiel$ git merge temp
Merge made by the 'recursive' strategy.
 README.md | 1 +
 pi        | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 pi
~/beispiel$ git log
commit c6e2b781a60785fa2fac0d7467b5b0e7c9a7fb8c
Merge: 90f7cfe efd7d0c
Author: Bernhard Heinloth <heinloth@cs.fau.de>
[...]
```



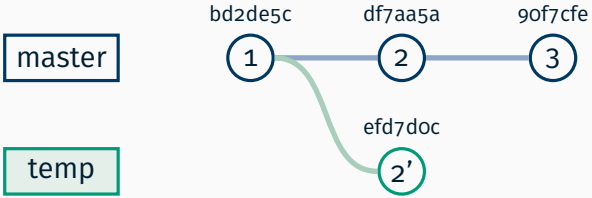
GIT Zweige zusammenführen



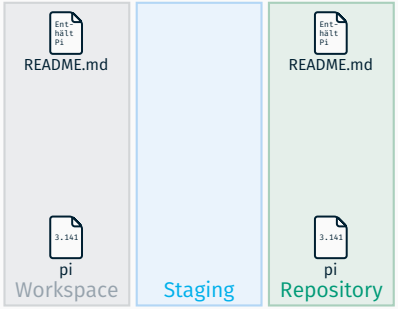

```
~/beispiel$ git shortlog
Bernhard Heinloth (5):
    Liesmich hinzugefügt
    Datei foo erstellt
    Foo korrigiert und Bar erstellt
    Kreiszahl ergänzt
    Merge branch 'temp'
```



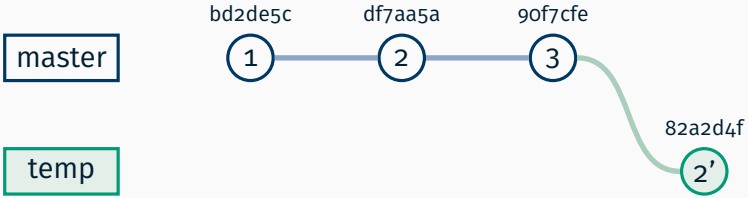
Alternativ: Die GIT Geschichte neu schreiben



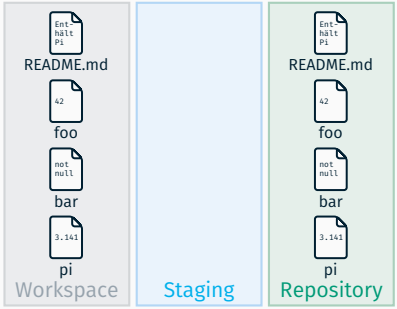

```
~/beispiel$ git branch
  master
* temp
```



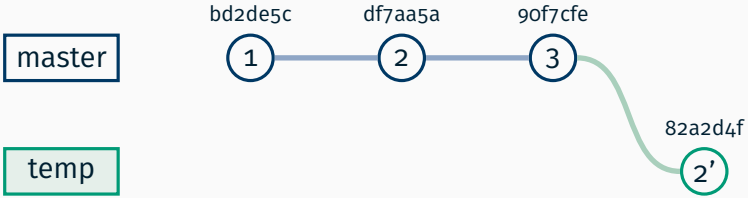
Alternativ: Die GIT Geschichte neu schreiben



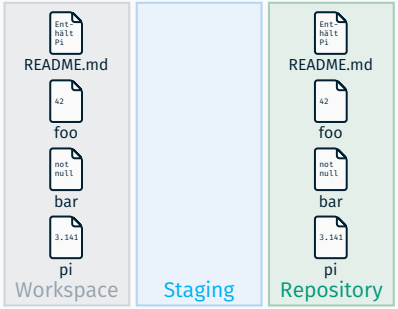

```
~/beispiel$ git rebase master
Zunächst wird der Branch zurückgespult,
um Ihre Änderungen darauf neu anzuwenden...
Wende an: Kreiszahl ergänzt
~/beispiel$ git log
commit 82a2d4f28d9986560aa75ea429ac5f510eebfd99
Merge: 90f7cfe efd7d0c
Author: Bernhard Heinloth <heinloth@cs.fau.de>
Date: Mon Oct 5 12:50:08 2020 +0200

    Kreiszahl ergänzt
```



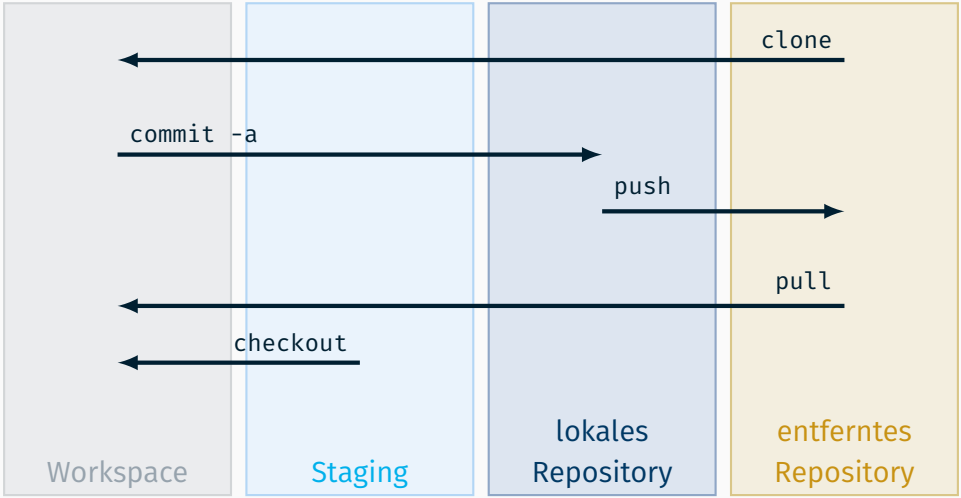
Alternativ: Die GIT Geschichte neu schreiben




```
~/beispiel$ git shortlog
Bernhard Heinloth (4):
    Liesmich hinzugefügt
    Datei foo erstellt
    Foo korrigiert und Bar erstellt
    Kreiszahl ergänzt
```



Überblick: Dateien in GIT ein- und auschecken



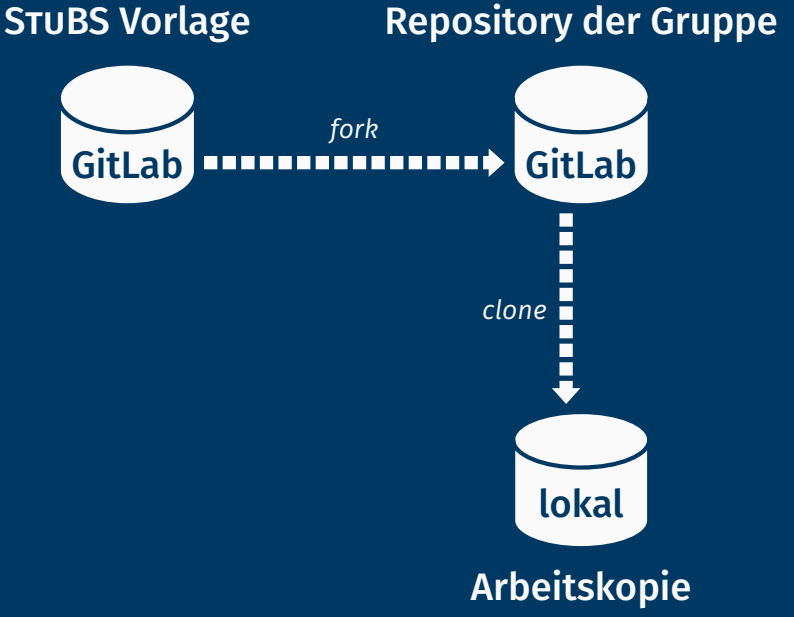
GIT Repository Hosting

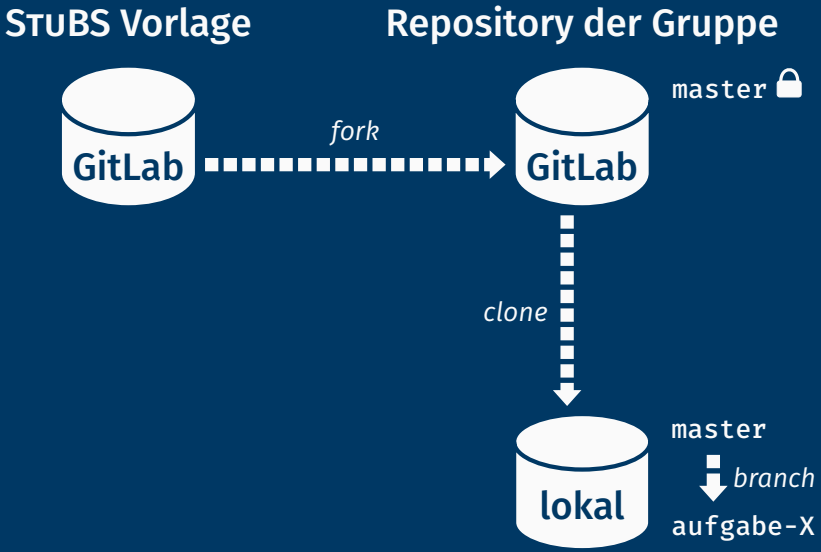
- Viele (kommerzielle) Anbieter, meist mit Weboberfläche zur Verwaltung:

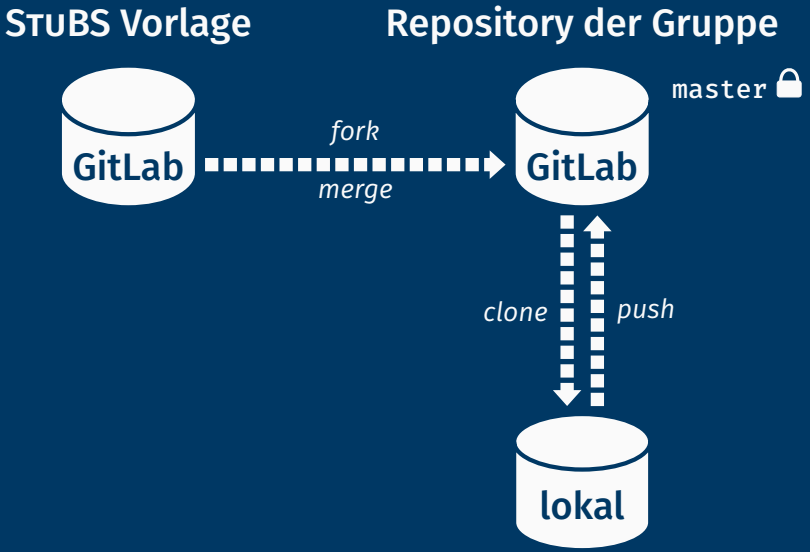
GitHub  Atlassian Bitbucket

- Eigene GitLab-Instanz des Departments Informatik auf `gitlab.cs.fau.de`
  - erlaubt kostenlos private Repos
  - unterstützt *Continuous Integration* (CI)
  - mit IDM verknüpft
    (keine Registrierung notwendig, Anmeldung über *FAU Single Sign-On*)

→ GitLab Übungsrepo wird automatisch nach Anmeldung erstellt


StuBS Vorlage
Repository der Gruppe
GitLab
fork
GitLab
clone
lokal
Arbeitskopie


StuBS Vorlage
Repository der Gruppe
GitLab
fork
GitLab
master
clone
lokal
master
branch
aufgabe-X
Arbeitskopie


StuBS Vorlage
Repository der Gruppe
GitLab
fork
merge
GitLab
master
clone
push
lokal
Arbeitskopie

Arbeiten mit entferntem GitLab Repository

- Entferntes GitLab Übungsrepo der Gruppe XX (lokal) klonen

```
$ git clone git@gitlab.cs.fau.de:i4-exercise/bs/ws20/mpstubs-gruppe-XX.git
```

- Neue Änderungen (*commits*) in GitLab kopieren

```
$ git push
```

  Ggf. Zweig `aufgabe-X` in GitLab anlegen

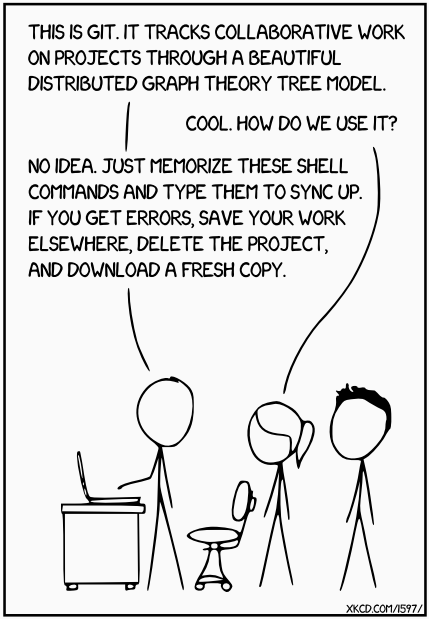
```
$ git push --set-upstream origin aufgabe-X
```

- Änderungen aus GitLab laden

```
$ git pull
```

  Bei Problemen (*merge conflict*) mit Werkzeug (hier: Meld) lösen

```
$ git mergetool --tool=meld
```


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/

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 push *Name*	in entfernte Quelle übertragen