Übung zu Betriebssystembau

Git-Crashkurs

Oktober 2024

Alexander Krause

Arbeitsgruppe Systemsoftware
Technische Universität Dortmund

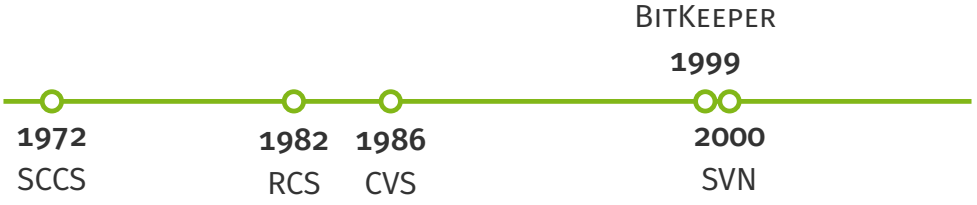
(Mit Material vom Lehrstuhl 4 der FAU)

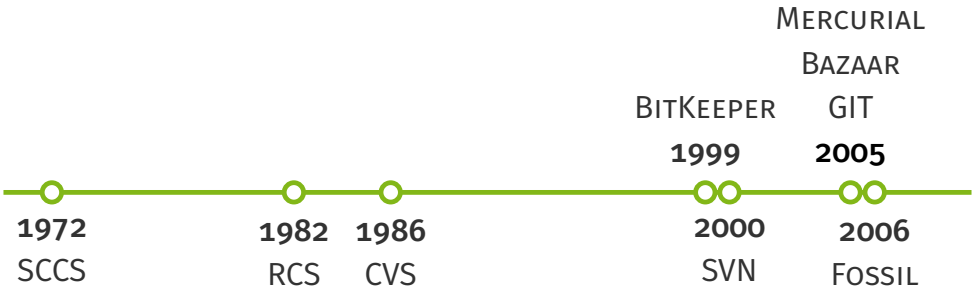
1972
SCCS

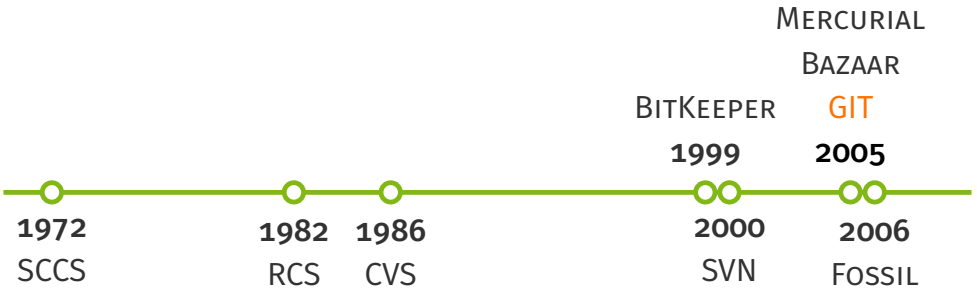

1972
SCCS
1982
RCS


1972
SCCS
1982
RCS
1986
CVS


1972
SCCS
1982
RCS
1986
CVS
2000
SVN


BitKeeper
1999
1972
SCCS
1982
RCS
1986
CVS
2000
SVN


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

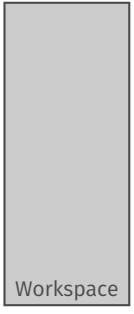

Mercurial
Bazaar
BitKeeper
GIT
1999
2005
1972
SCCS
1982
RCS
1986
CVS
2000
SVN
2006
Fossil

Schlüsselkonzepte von GIT

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

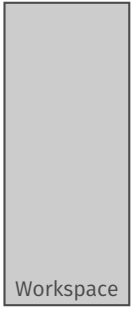
Lokales GIT Repository initialisieren

```
01 stud@bsb:~$ mkdir beispiel
02 stud@bsb:~$ cd beispiel
```



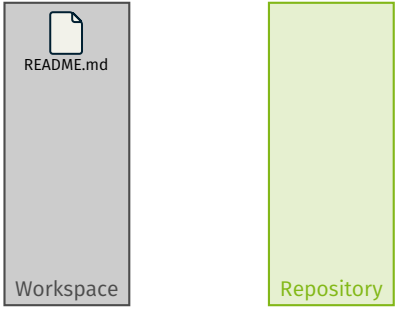
Lokales GIT Repository initialisieren

```
01 stud@bsb:~$ mkdir beispiel
02 stud@bsb:~$ cd beispiel
03 stud@bsb:~/beispiel$ git init
04 Leeres Git-Repository in beispiel/.git/ initialisiert
```



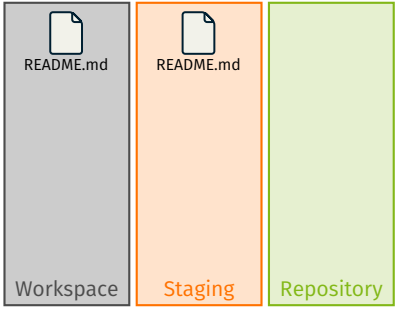
Dateien mit GIT verwalten

```
01 stud@bsb:~/beispiel$ touch README.md
```



Dateien mit GIT verwalten

```
01 stud@bsb:~/beispiel$ touch README.md
02 stud@bsb:~/beispiel$ git add README.md
```

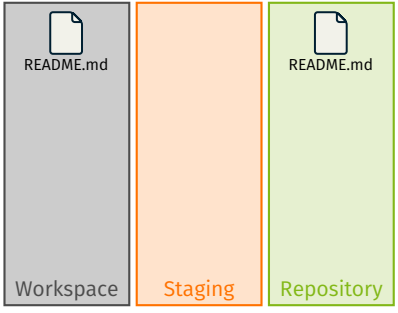


Dateien mit GIT verwalten

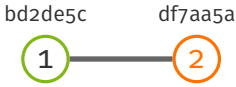
bd2de5c

1

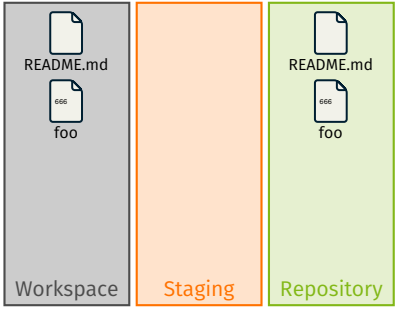
```
01 stud@bsb:~/beispiel$ touch README.md
02 stud@bsb:~/beispiel$ git add README.md
03 stud@bsb:~/beispiel$ git commit -m "Liesmich
       hinzugefügt"
04 [master (Root-Commit) bd2de5c] Liesmich hinzugefügt
05  1 file changed, 0 insertions(+), 0 deletions(-)
06  create mode 100644 README.md
```



Dateien mit GIT verwalten

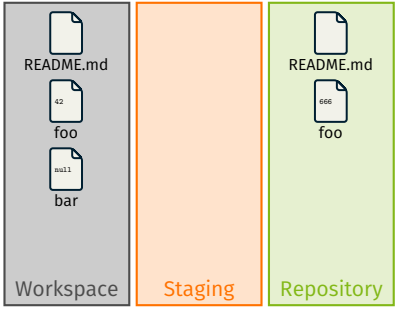



```
01 stud@bsb:~/beispiel$ echo "666" > foo
02 stud@bsb:~/beispiel$ git add foo
03 stud@bsb:~/beispiel$ git commit -m "Datei foo erstellt
       "
04 [master df7aa5a] Datei foo erstellt
05  1 file changed, 1 insertion(+)
06  create mode 100644 foo
```



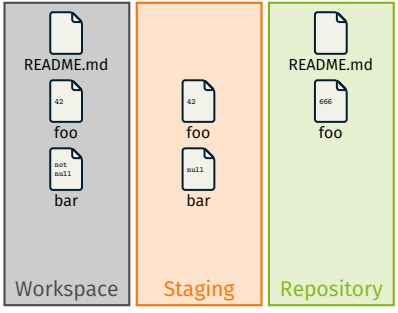
Dateien mit GIT verwalten


```
01 stud@bsb:~/beispiel$ echo "42" > foo
02 stud@bsb:~/beispiel$ echo "null" > bar
03 stud@bsb:~/beispiel$ git status
04 Auf Branch master
05 Änderungen, die nicht zum Commit vorgemerkt sind:
06     geändert: foo
07
08 Unversionierte Dateien:
09     bar
10
11 keine Änderungen zum Commit vorgemerkt
```



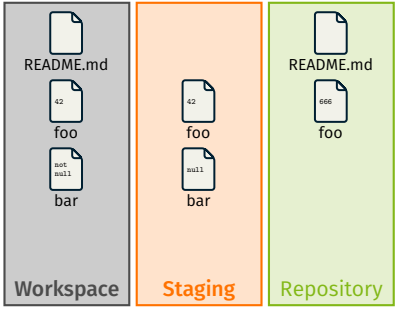
Dateien mit GIT verwalten


```
01 stud@bsb:~/beispiel$ git add foo bar
02 stud@bsb:~/beispiel$ echo "not null" > bar
03 stud@bsb:~/beispiel$ git status
04 Auf Branch master
05 Zum Commit vorgemerkte Änderungen:
06     neue Datei: bar
07     geändert: foo
08
09 Änderungen, die nicht zum Commit vorgemerkt sind:
10     geändert: bar
```



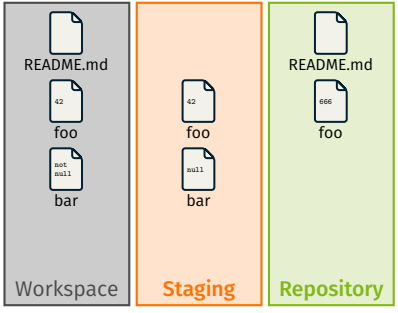
Dateien mit GIT verwalten


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



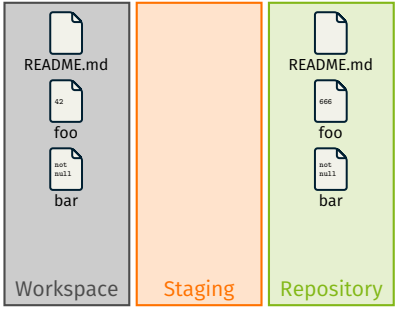
Dateien mit GIT verwalten


```
01 stud@bsb:~/beispiel$ git diff --staged
02 diff --git a/bar b/bar
03 new file mode 100644
04 index 0000000..19765bd
05 --- a/bar
06 +++ b/bar
07 @@ -0,0 +1 @@
08 +null
09 diff --git a/foo b/foo
10 index 7cc86ad..d81cc07 100644
11 [...]
```



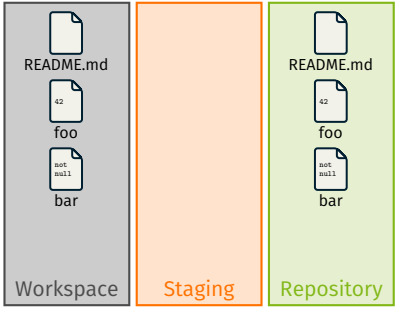
Dateien mit GIT verwalten


```
01 stud@bsb:~/beispiel$ git add bar
02 stud@bsb:~/beispiel$ git commit -m \
03             "Foo korrigiert und Bar erstellt"
04 [master 90f7cfe] Foo korrigiert und Bar erstellt
05  2 files changed, 2 insertions(+), 1 deletion(-)
06  create mode 100644 bar
```

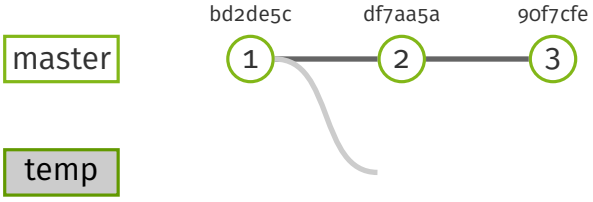


Dateien mit GIT verwalten

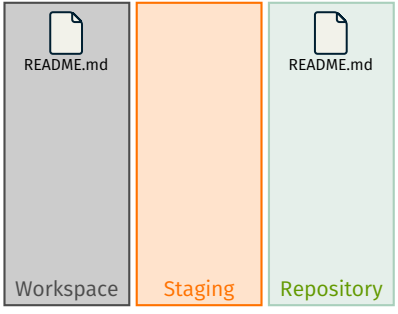

```
01 stud@bsb:~/beispiel$ git shortlog
02 Bernhard Heinloth (3):
03     Liesmich hinzugefügt
04     Datei foo erstellt
05     Foo korrigiert und Bar erstellt
```



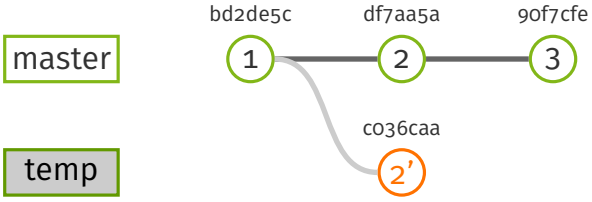
GIT Zweige



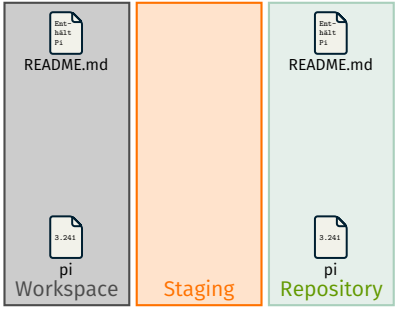

```
01 stud@bsb:~/beispiel$ git branch temp bd2de5c
02 stud@bsb:~/beispiel$ git checkout temp
03 Zu Zweig »temp« gewechselt
04 stud@bsb:~/beispiel$ git shortlog
05 Bernhard Heinloth (1):
06     Liesmich hinzugefügt
```



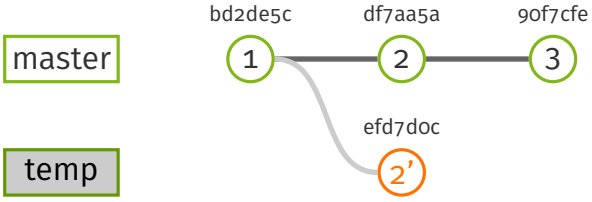
GIT Zweige



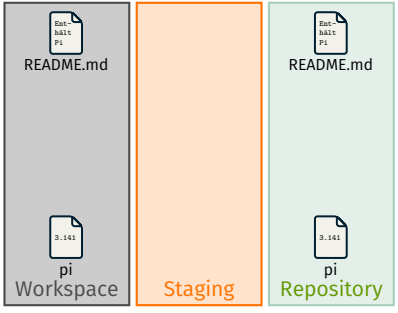

```
01 stud@bsb:~/beispiel$ echo "3.241" > pi
02 stud@bsb:~/beispiel$ echo "Beinhaltet Pi" >> README.md
03 stud@bsb:~/beispiel$ git add .
04 stud@bsb:~/beispiel$ git commit -m "Kreiszahl
       hinzugefügt"
05 [temp c036caa] Kreiszahl hinzugefügt
06  2 files changed, 2 insertions(+)
07  create mode 100644 pi
```



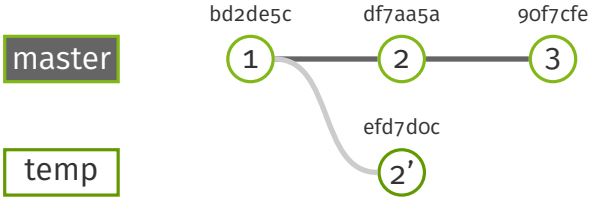
GIT Zweige



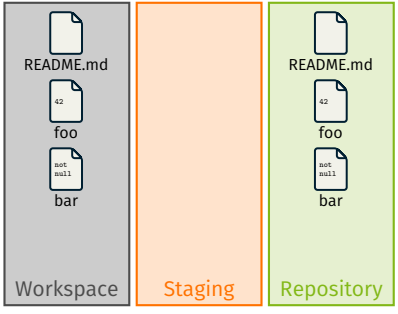

```
01 stud@bsb:~/beispiel$ echo "3.141" > pi
02 stud@bsb:~/beispiel$ git commit -a --amend -m \
03                       "Kreiszahl ergänzt"
04 [temp efd7d0c] Kreiszahl ergänzt
05  Date: Mon Oct 5 12:50:08 2020 +0200
06  2 files changed, 2 insertions(+)
07  create mode 100644 pi
```



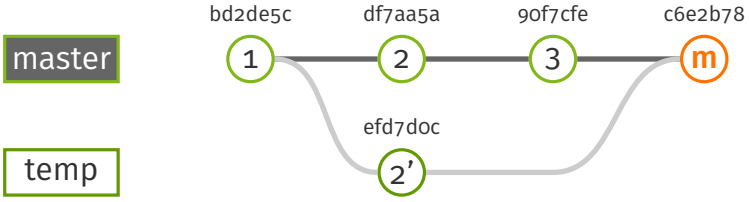
GIT Zweige



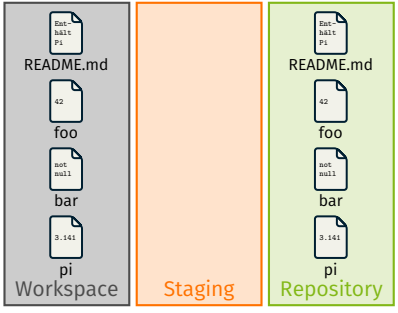

```
01 stud@bsb:~/beispiel$ git branch
02   master
03 * temp
04 stud@bsb:~/beispiel$ git checkout master
05 stud@bsb:~/beispiel$ git branch
06 * master
07   temp
```



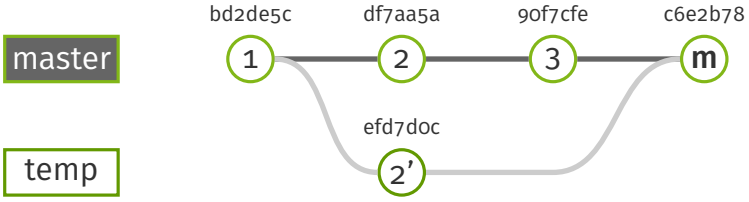
GIT Zweige zusammenführen



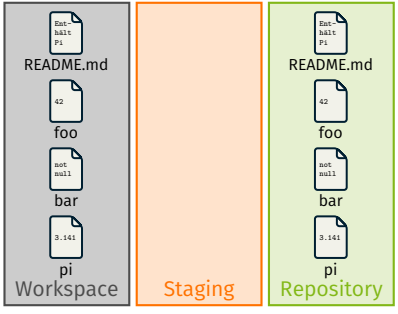

```
01 stud@bsb:~/beispiel$ git merge temp
02 Merge made by the 'recursive' strategy.
03  README.md | 1 +
04  pi        | 1 +
05  2 files changed, 2 insertions(+)
06  create mode 100644 pi
07 stud@bsb:~/beispiel$ git log
08 commit c6e2b781a60785fa2fac0d7467b5b0e7c9a7fb8c
09 Merge: 90f7cfe efd7d0c
10 Author: Bernhard Heinloth <heinloth@cs.fau.de>
11 [...]
```



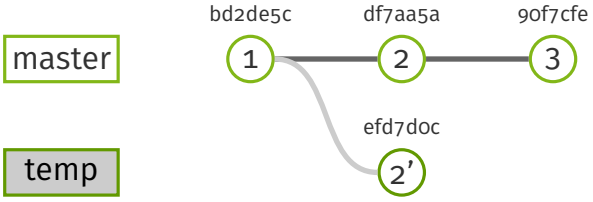
GIT Zweige zusammenführen



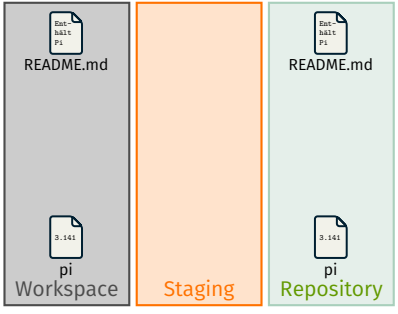

```
01 stud@bsb:~/beispiel$ git shortlog
02 Bernhard Heinloth (5):
03     Liesmich hinzugefügt
04     Datei foo erstellt
05     Foo korrigiert und Bar erstellt
06     Kreiszahl ergänzt
07     Merge branch 'temp'
```



Alternativ: Die GIT Geschichte neu schreiben

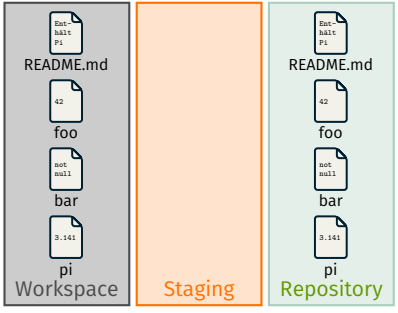



```
01 stud@bsb:~/beispiel$ git branch
02   master
03 * temp
```

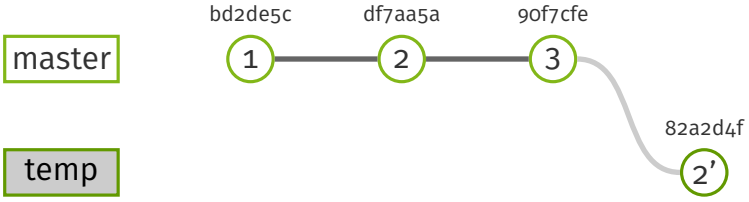


Alternativ: Die GIT Geschichte neu schreiben

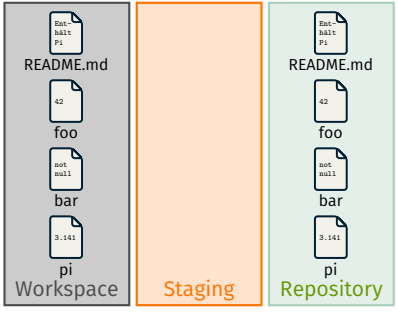

```
01 stud@bsb:~/beispiel$ git rebase master
02 Zunächst wird der Branch zurückgespult,
03 um Ihre Änderungen darauf neu anzuwenden...
04 Wende an: Kreiszahl ergänzt
05 stud@bsb:~/beispiel$ git log
06 commit 82a2d4f28d9986560aa75ea429ac5f510eebfd99
07 Merge: 90f7cfe efd7d0c
08 Author: Bernhard Heinloth <heinloth@cs.fau.de>
09 Date: Mon Oct 5 12:50:08 2020 +0200
10
11     Kreiszahl ergänzt
```



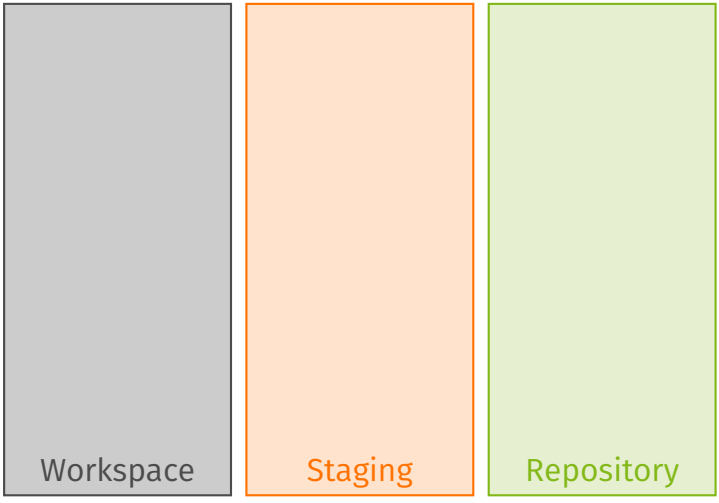
Alternativ: Die GIT Geschichte neu schreiben



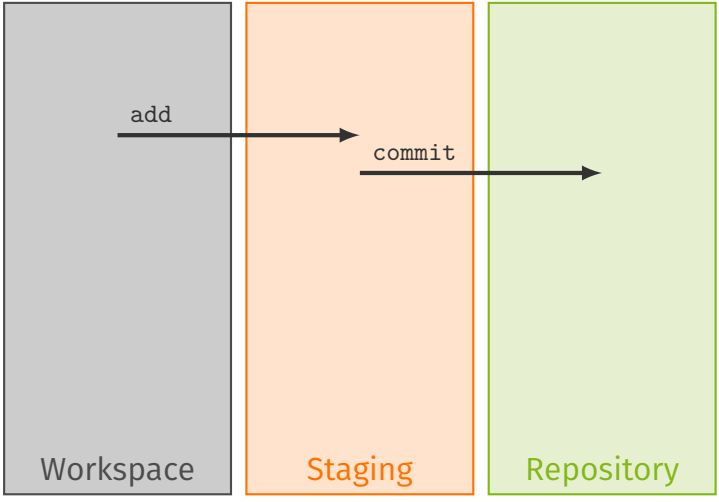

```
01 stud@bsb:~/beispiel$ git shortlog
02 Bernhard Heinloth (4):
03     Liesmich hinzugefügt
04     Datei foo erstellt
05     Foo korrigiert und Bar erstellt
06     Kreiszahl ergänzt
```



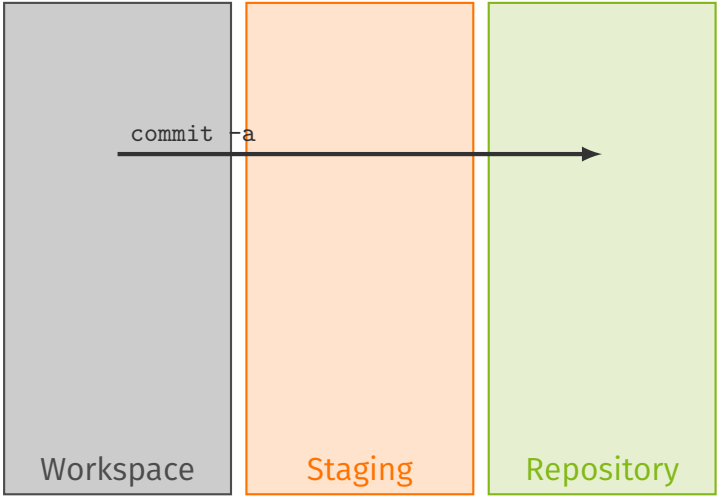
Überblick: Dateien in GIT ein- und auschecken



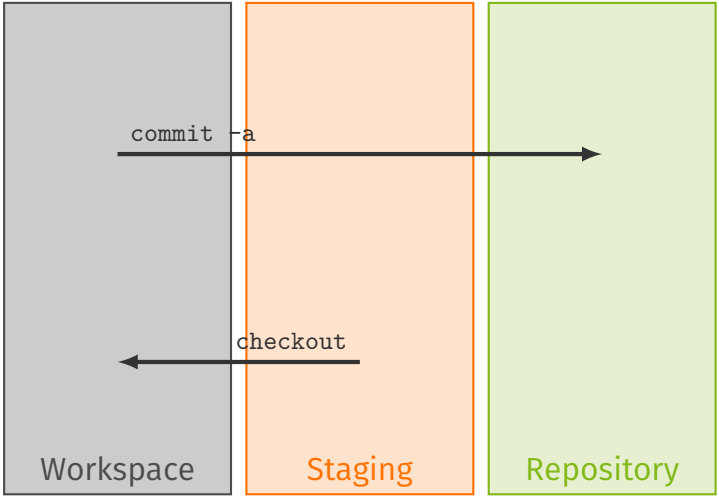
Überblick: Dateien in GIT ein- und auschecken



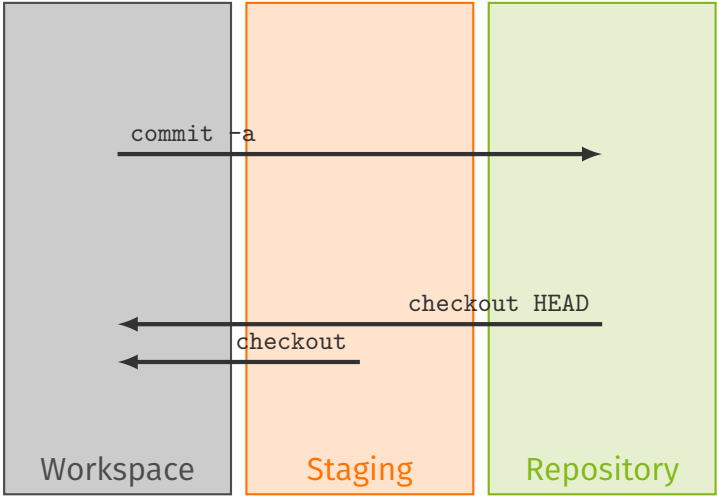
Überblick: Dateien in GIT ein- und auschecken



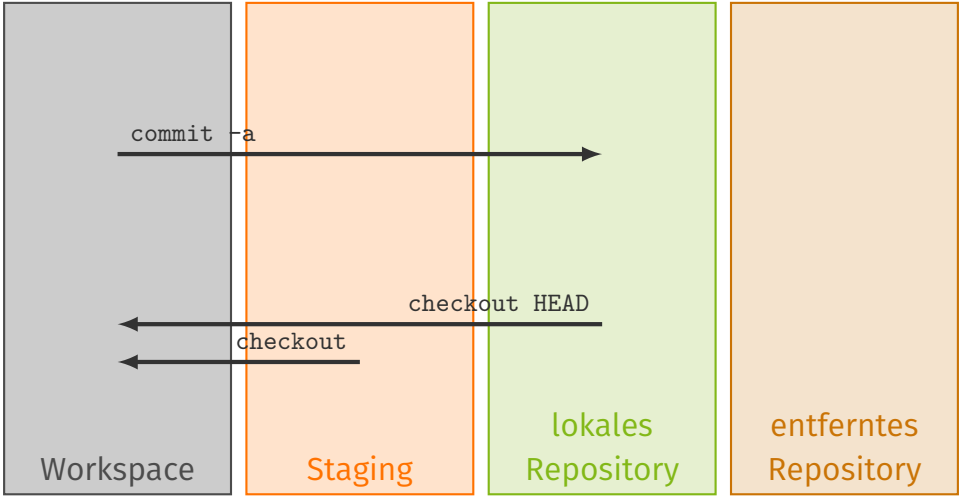
Überblick: Dateien in GIT ein- und auschecken



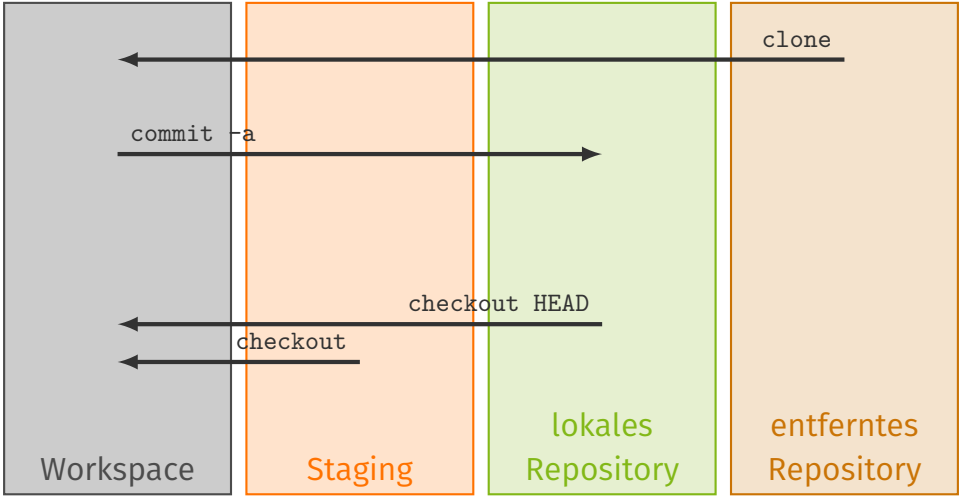
Überblick: Dateien in GIT ein- und auschecken



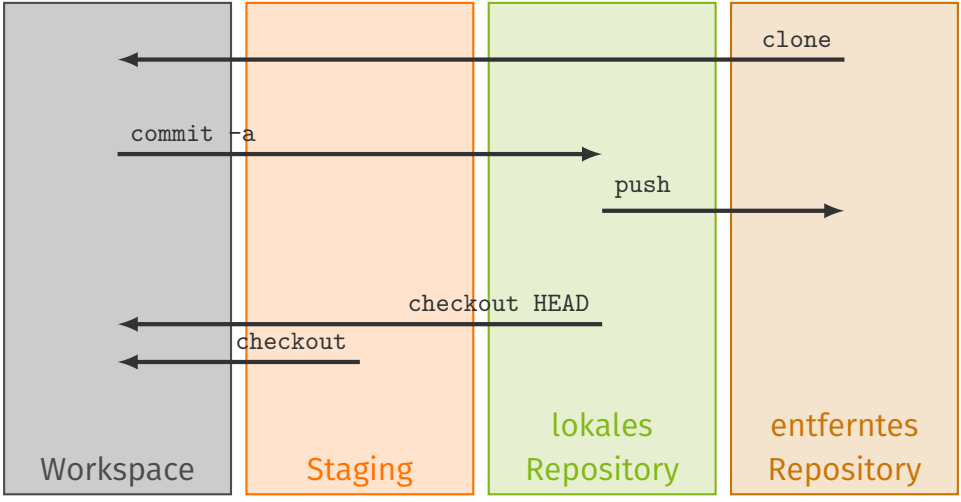
Überblick: Dateien in GIT ein- und auschecken



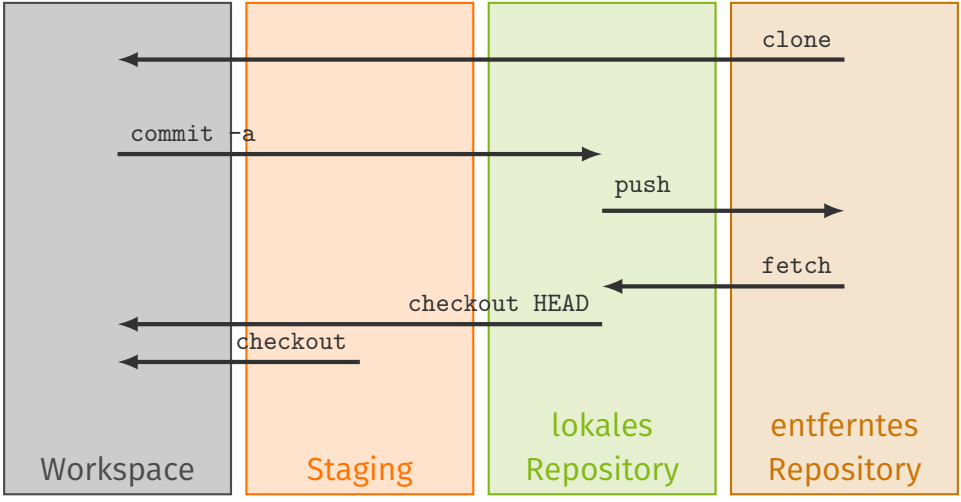
Überblick: Dateien in GIT ein- und auschecken



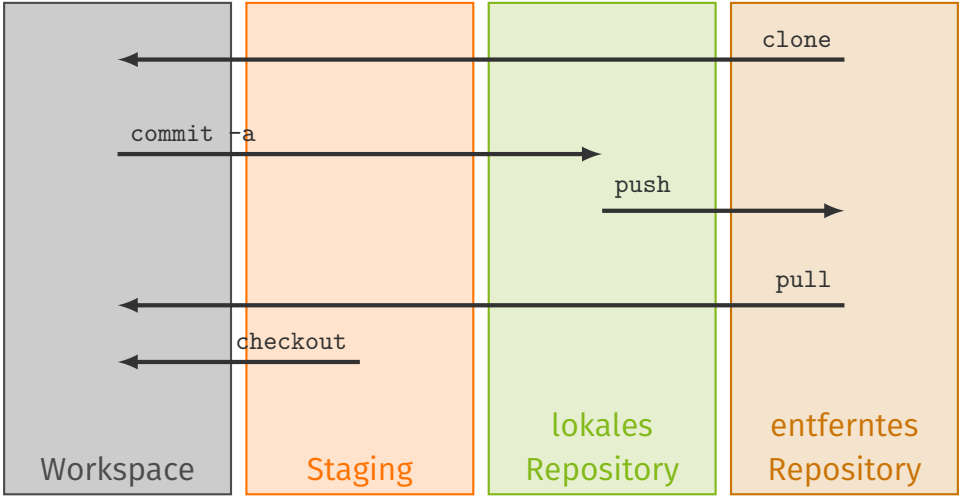
Überblick: Dateien in GIT ein- und auschecken



Überblick: Dateien in GIT ein- und auschecken



Überblick: Dateien in GIT ein- und auschecken



GIT Repository Hosting

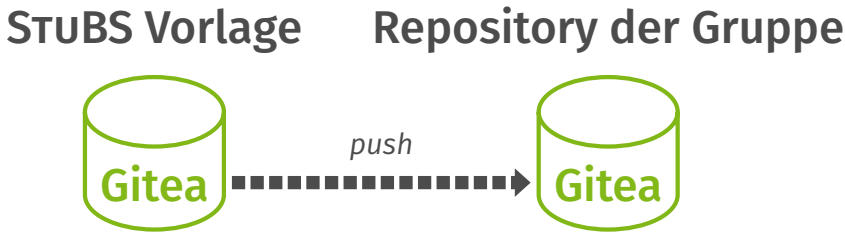
- Viele (kommerzielle) Anbieter, meist mit Weboberfläche zur Verwaltung:

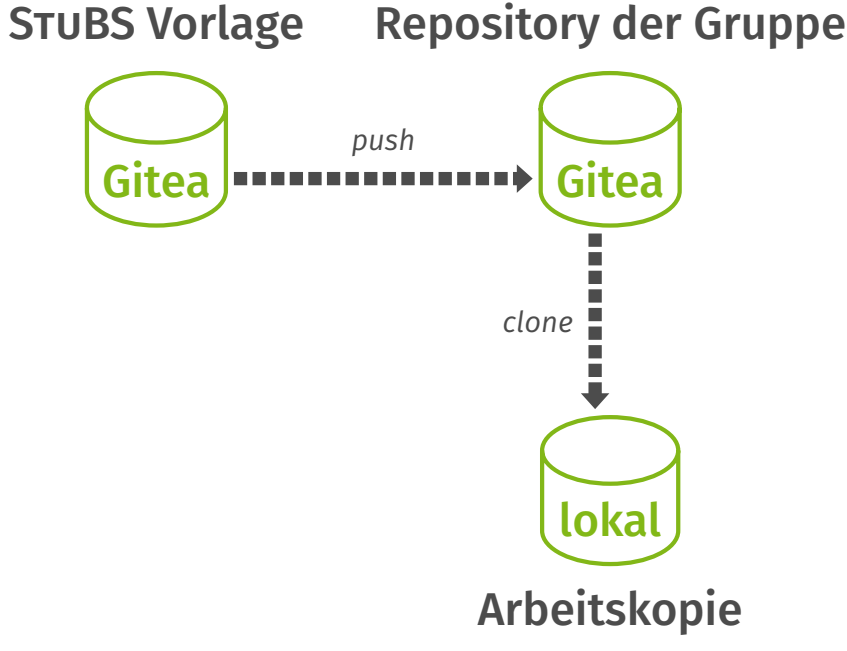
GitHub  Atlassian Bitbucket

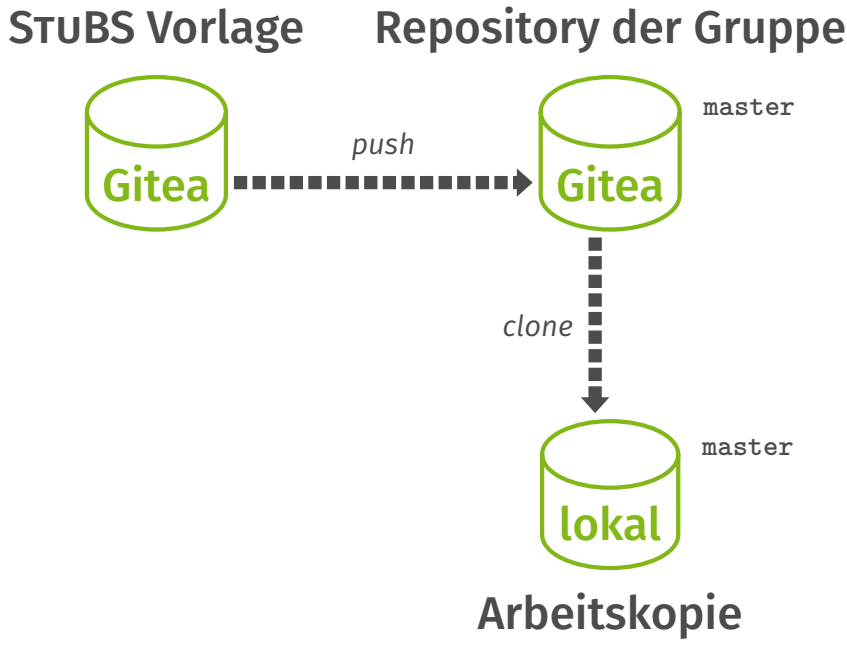
- Eigene GITEA-Instanz des Departments Informatik auf `git.cs.tu-dortmund.de`
  - erlaubt kostenlos private Repos
  - unterstützt *Continuous Integration* (CI)
  - keine Registrierung notwendig, Anmeldung über *IRB-Account*, bitte jede(r) einmal anmelden

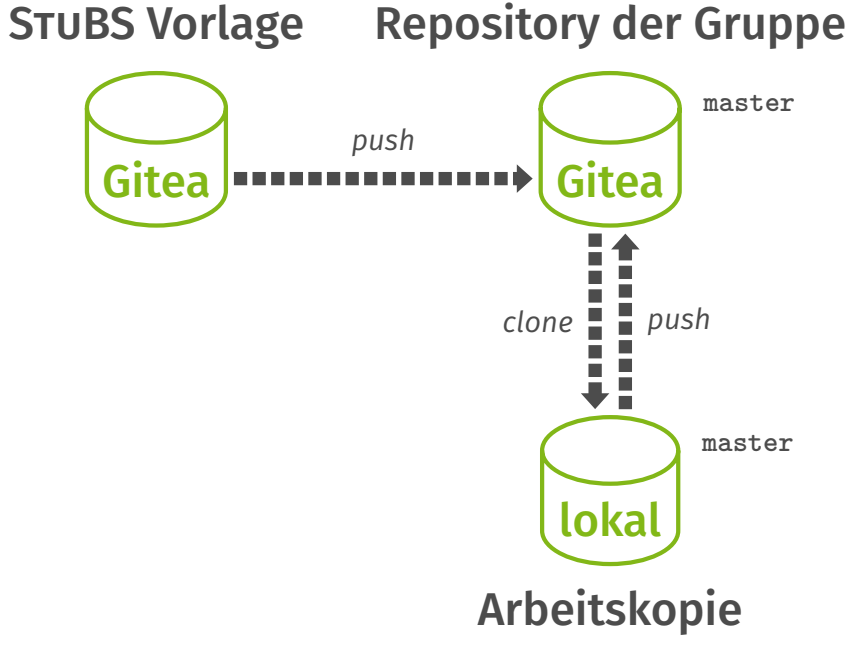
→ GITEA Übungsrepo wird automatisch nach Anmeldung erstellt → Siehe Übungsfolien zu `u00-organisation`

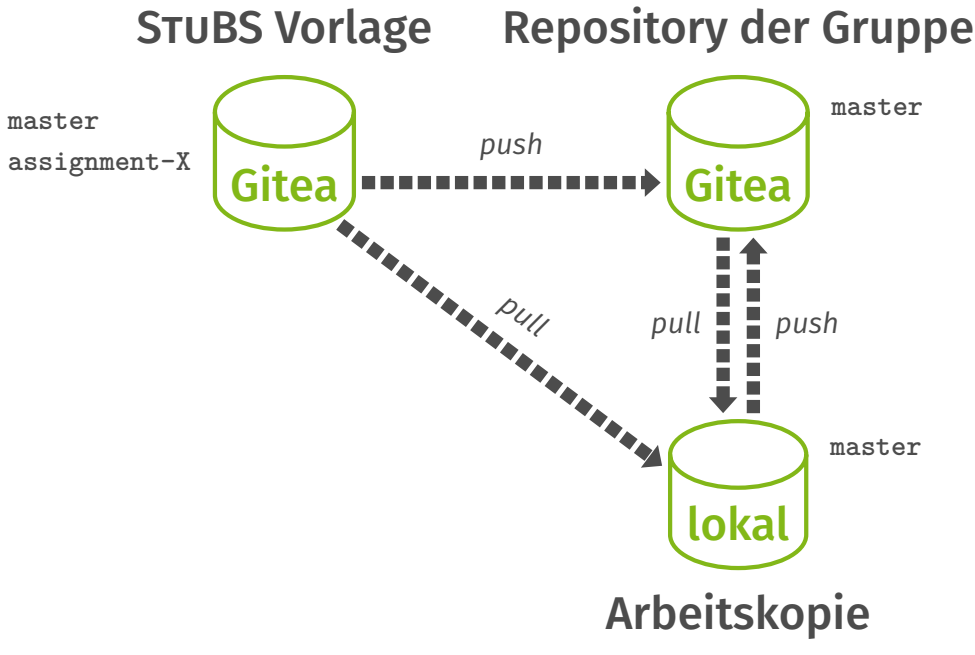
StuBS Vorlage


StuBS Vorlage
Repository der Gruppe
Gitea
push
Gitea


StuBS Vorlage
Repository der Gruppe
Gitea
push
Gitea
clone
lokal
Arbeitskopie


StuBS Vorlage
Repository der Gruppe
Gitea
push
Gitea
master
clone
master
lokal
Arbeitskopie


STUBS Vorlage
Repository der Gruppe
Gitea
push
Gitea
master
clone
push
master
lokal
Arbeitskopie


STUBS Vorlage
Repository der Gruppe
master
assignment-X
Gitea
push
Gitea
master
pull
pull
push
master
lokal
Arbeitskopie

Vorbereitung

- Name und Mailadresse setzen

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

- GIT Werkzeuge anpassen

```
$ git config --global core.editor nano
$ git config --global merge.tool meld
$ git config --list
```

- *Optional:* Aliase definieren

```
$ git config --global alias.word-diff=diff --word-diff=color -b
$ git word-diff
```

- *Optional:* SSH-Key erstellen und öffentlichen Schlüssel in GITEA eintragen

Arbeiten mit entferntem GITEA Repository

- Entferntes GITEA Übungsrepo der Gruppe XX (lokal) klonen

```
$ git clone git@gssh://git@git.cs.tu-dortmund.de:2222/BSB-WS22/gruppe-XX.git
```

- Neue Änderungen (*commits*) in GITEA kopieren

```
$ git push
```

  Ggf. Zweig `aufgabe-X` in GITTEA anlegen

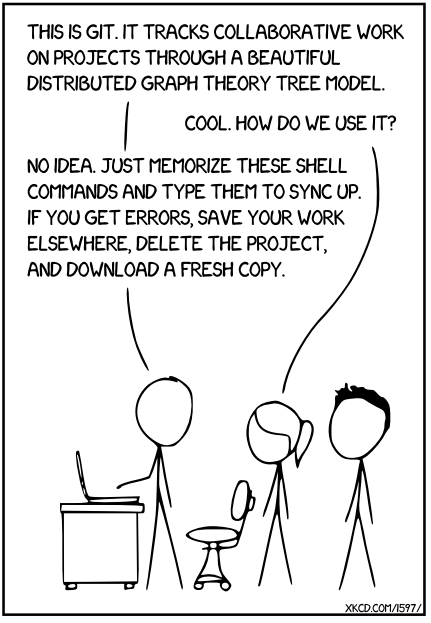
```
$ git push --set-upstream origin aufgabe-X
```

- Änderungen aus GITEA 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