TECHNISCHE UNIVERSITÄT
CHEMNITZ

GIT for pedestrians

how 2 git gud

7.02.2020

Maximilian Neumann

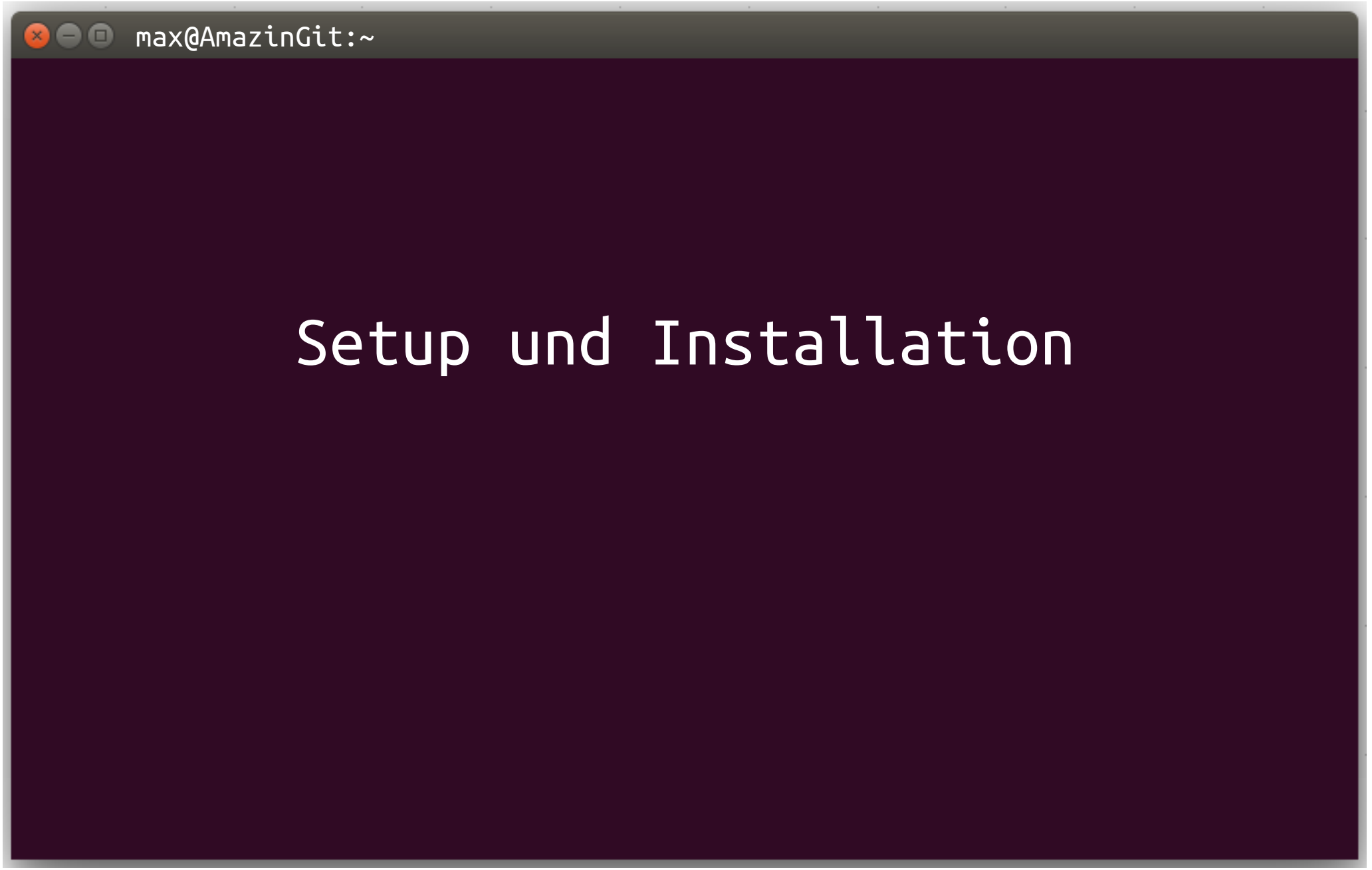
Git

max@AmazinGit:~

Git: - Version Control System (VCS)

- Repository dokumentiert und markiert Änderungen der Dateien
- Zugriff auf alte Versionen, dient als Backup
- paralleles Arbeiten an unterschiedlichen Versionen
- Synchronisation zwischen Arbeitsplätzen

→ besonders geeignet für Manuskripte und Quellcode


max@AmazinGit:~
Setup und Installation

Installation von git

max@AmazinGit:~

bereits installiert

```
max@AmazinGit:~$ git --version
```

```
max@AmazinGit:~$ git clone https://github.com/git/git
```

nicht installiert

```
max@AmazinGit:~$ sudo apt-get install git
```

alternativ

Download von git-Website: https://git-scm.com/downloads

Übersicht und Einstieg

max@AmazinGit:~

Interne help Funktion

```
max@AmazinGit:~$ git help
                                ←   Überblick über wichtigste Befehle

max@AmazinGit:~$ git help “Befehl“
                                ←   Details zu einzelnem Befehl
```

Cheatsheet

https://zeroturnaround.com/rebellabs/git-commands-and-best-practices-cheat-sheet

https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf

Erzeugen eines Git-Repository

max@AmazinGit:~

lokales Repository erstellen

```
max@AmazinGit:~$ mkdir GIT
max@AmazinGit:~$ cd GIT

max@AmazinGit:~/GIT$ git init        ←   Initialisiert das aktuelle Verzeichnis
                                         als git-Repository
```

externes Repository kopieren

```
max@AmazinGit:~$ git clone git@gitlab.hrz.tu-chemnitz.de:nemax--tu-
chemnitz.de/procoski-2020-02-07.git
                                     ←   Kopiert ein bereits existierendes
                                         Repository
```

Nutzer und Email konfigurieren

max@AmazinGit:~

entweder lokaler Nutzer

```
max@AmazinGit:~/GIT$ git config user.name  “Max Neumann“
max@AmazinGit:~/GIT$ git config user.email “Maximilian.Neumann@physik.tu-
chemnitz.de
```

oder globaler Nutzer

```
max@AmazinGit:~/GIT$ git config --global user.name  “Maximilian Neumann“
max@AmazinGit:~/GIT$ git config --global user.email
“Maximilian.Neumann@physik.tu-chemnitz.de
```

Kontrolle der Identität

```
max@AmazinGit:~/GIT$ git config --list
```

zusätzliche Nutzereinstellungen

max@AmazinGit:~

Zeilenumbrüche unter Linux

```
max@AmazinGit:~/GIT$ git config --global core.autocrlf input
```

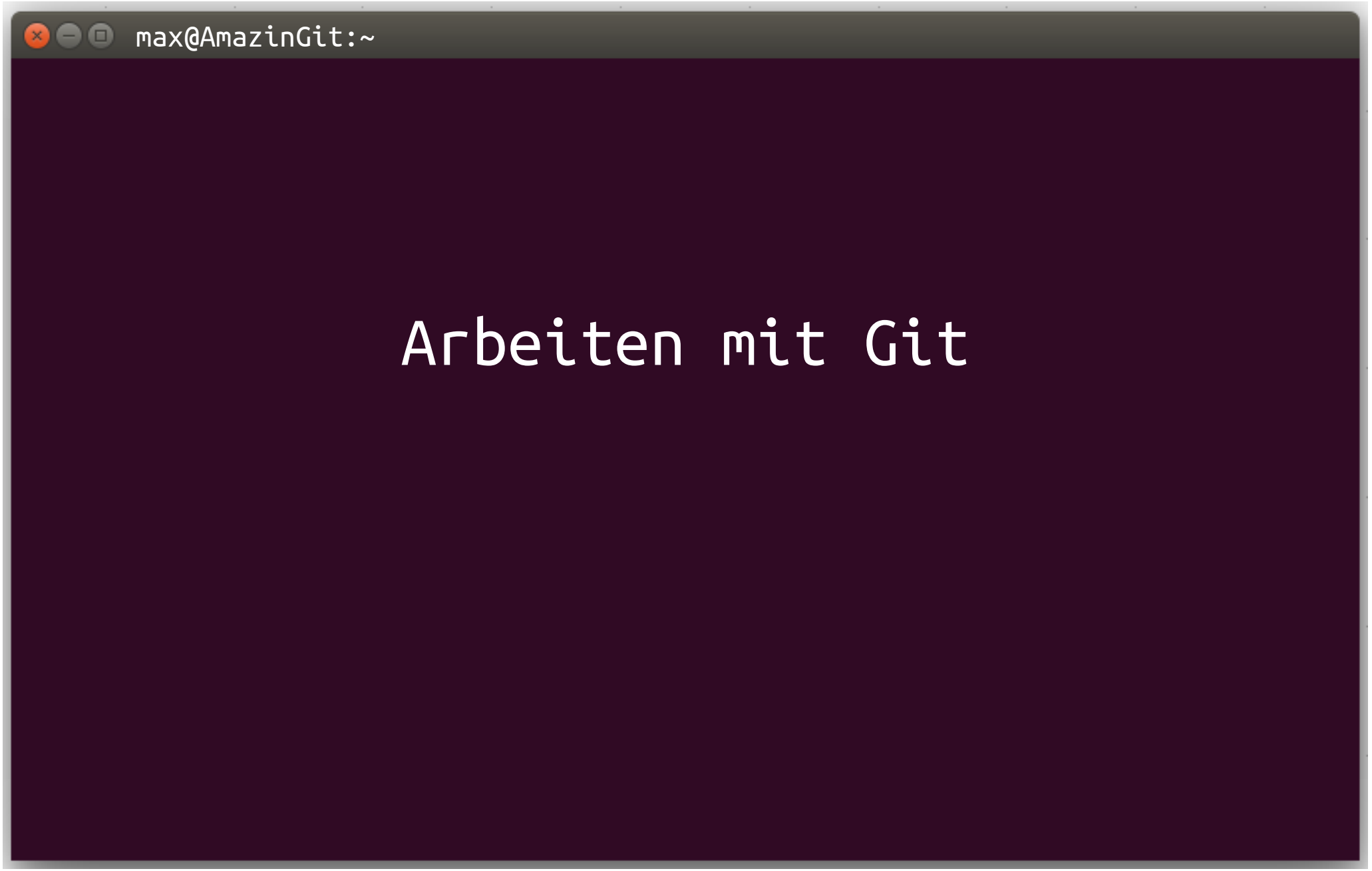
analog Windows

```
max@AmazinGit:~/GIT$ git config --global core.autocrlf true
```

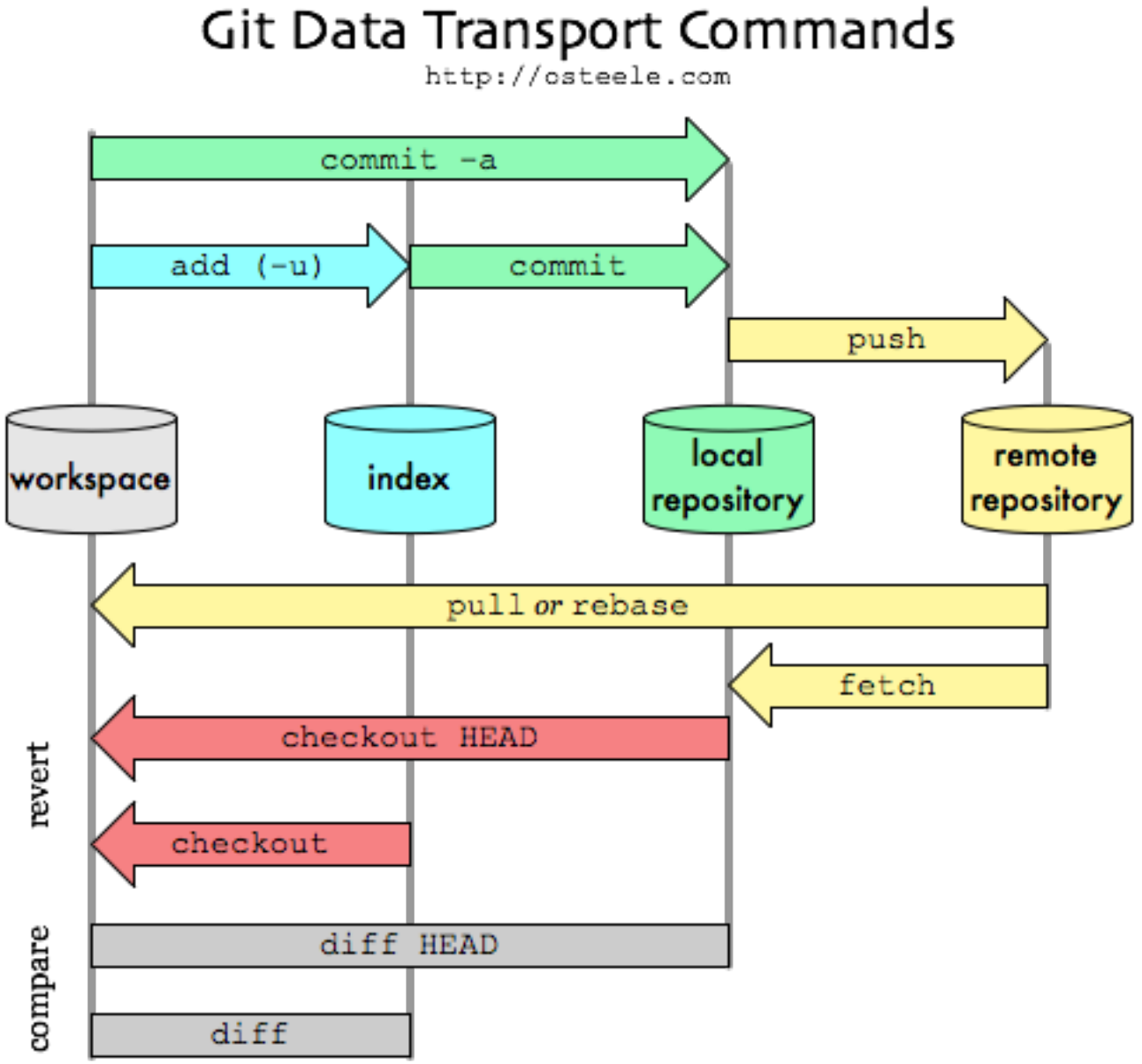
← Git verarbeitet Zeilenumbrüche automatisch, relevant falls unterschiedliche Betriebssysteme verwendet werden

Standard-Editor setzen

```
max@AmazinGit:~/GIT$ git config --global core.editor “gedit“
```


max@AmazinGit:~
Arbeiten mit Git

Gits drei / vier Instanzen



https://blog.osteele.com/2008/05/my-git-workflow/

Arbeitsverzeichnis

max@AmazinGit:~

Erzeugen von Test-Dateien

```
max@AmazinGit:~/GIT$ touch test1 test2 test3
                                    ←   erzeugt drei Dateien
```

aktueller Status des Repository

```
max@AmazinGit:~/GIT$ git status
                                    ←   zeigt Unterschiede zum Repository
```

Index / Staging / Cache

max@AmazinGit:~

Der Index ist zur Aufnahme von Dateien gedacht, die zu einem späteren Zeitpunkt in das Repository aufgenommen / gespeichert werden.

Dateien in Index aufnehmen

```
max@AmazinGit:~/GIT$ git add “Datei“
                                    ←   nimmt einzelne Datei auf

max@AmazinGit:~/GIT$ git add .
                                    ←   nimmt alle neuen Dateien auf
```

lokales Repository

max@AmazinGit:~

Das Repository ist das Herzstück von git. Erlaubt Zugriff auf alle aktuellen und vorhergangenen Commits.

Aufnahme in das Repository / speichern

```
max@AmazinGit:~/GIT$ git commit “Datei“
                                ←   öffnet Editor für Commit-Nachricht,
                                    speichert Datei

max@AmazinGit:~/GIT$ git commit “Datei“ -m “commit-Nachricht“
                                ←   Option -m lässt die commit-Nachricht
                                    direkt anhängen, ohne Editor
```

commit von mehreren Dateien im Index

```
max@AmazinGit:~/GIT$ git commit -am “mehrere Dateien hinzugefügt“
```

Änderung von Dateien und Vergleich

max@AmazinGit:~

ändern einer Datei

```
max@AmazinGit:~/GIT$ echo “Hallo Welt!“ > test1
```

Vergleich / Unterschiede finden

Es kann sinnvoll sein die Änderungen zu kontrollieren bevor sie ins Repository aufgenommen werden.

```
max@AmazinGit:~/GIT$ git diff
```

← vergleicht gesamtes Arbeitsverzeichnis mit Repository

```
max@AmazinGit:~/GIT$ git diff “test1“
```

← gleicht nur test1 ab

Danach kann wieder mit commit gespeichert werden.

Commit Historie

max@AmazinGit:~

Ausgabe aller commits

```
max@AmazinGit:~/GIT$ git log
                              ←   Ausgabe von Hash-Wert, Autor, Datum, Nachricht

max@AmazinGit:~/GIT$ git log --stat
                              ←   zeigt welche Dateien geändert wurden



max@AmazinGit:~/GIT$ git log -p
                              ←   zeigt zusätzlich die genauen Änderungen an

max@AmazinGit:~/GIT$ git show
                              ←   analog git log -p
```

schöner

```
max@AmazinGit:~/GIT$ git log --graph --all --decorate
                              ←   zeigt Struktur, alle Branches, HEAD und
                                  Branch-Namen
```

Commit Historie Recherche

max@AmazinGit:~

spezifisches Durchsuchen der Historie

```
max@AmazinGit:~/GIT$ git log “Datei“
                              ←   Commits zu einer bestimmten Datei

max@AmazinGit:~/GIT$ git log -n5
                              ←   letzte 5 Commits, n nicht notwendig

max@AmazinGit:~/GIT$ git log --author=Max
                              ←   Commits eines bestimmten Autors

max@AmazinGit:~/GIT$ git log --grep=commit
                              ←   Commits mit Schlüsselwort in der Nachricht

max@AmazinGit:~/GIT$ git log --since “1 hour ago“
                              ←   Commits jünger als 1Stunde, analog --till,
                              ←   auch Datum möglich
```

Details eines einzelnen Commits

```
max@AmazinGit:~/GIT$ git show Hash-Wert
                              ←   Details zu Commit mit entsprechendem Hash-Wert
```

Branching

max@AmazinGit:~

Bisher haben wir nur den master-Branch benutzt. Um an einem separaten Teil des Projektes zu arbeiten können weitere Verzweigungen eingeführt werden, die später wieder mit dem Hauptzweig verschmolzen oder gelöscht werden können.

neuen Branch erzeugen

```
max@AmazinGit:~/GIT$ git branch “Zweig1“
                              ←   Erzeugen eines neuen Branch

max@AmazinGit:~/GIT$ git checkout “Zweig1“
                              ←   Setzt den neuen Branch als HEAD
```

oder kürzer

```
max@AmazinGit:~/GIT$ git checkout -b “Zweig1“
```

Überblick über branching mit log

```
max@AmazinGit:~/GIT$ git log --graph --decorate --all
```

laden aus dem Repository

max@AmazinGit:~

Wenn Dateien verloren gehen oder Änderungen rückgängig gemacht werden sollen.

spezifisches Durchsuchen der Historie

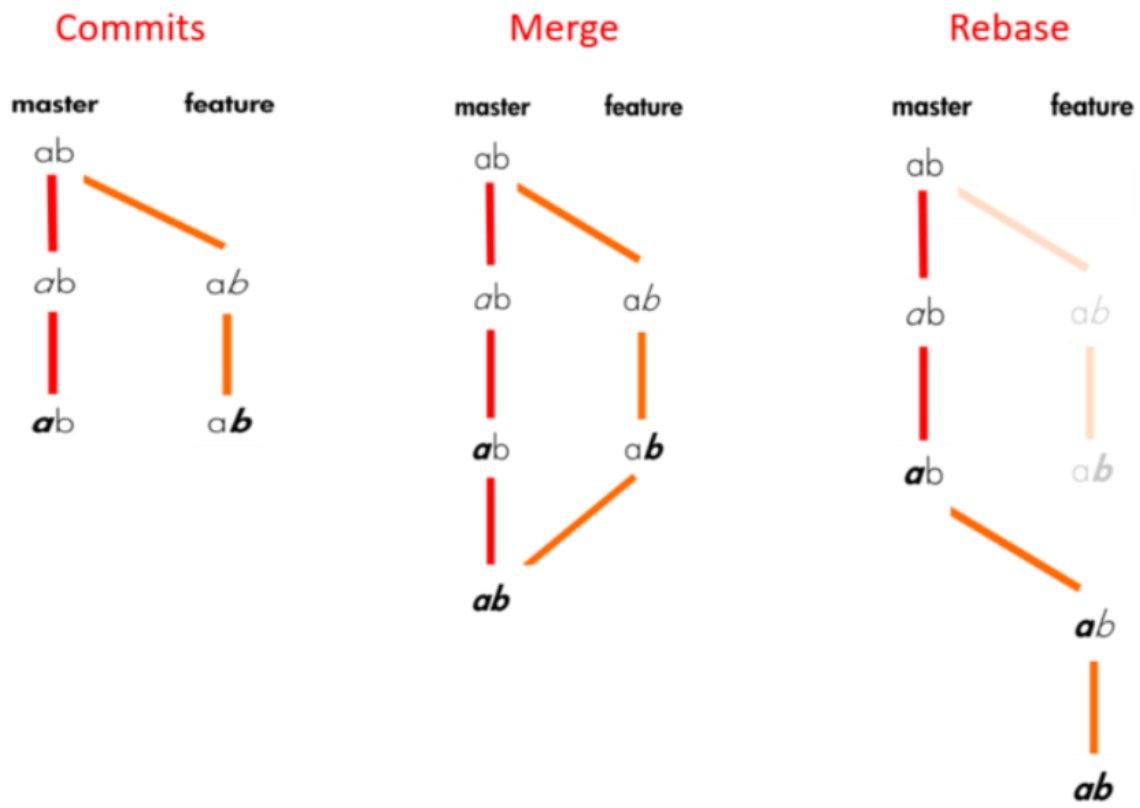
```
max@AmazinGit:~/GIT$ git checkout „“Hash-Wert“
                              ←   stellt den entsprechenden Commit wieder her

max@AmazinGit:~/GIT$ git checkout --test1
                              ←   setzt test1 auf den Zustand im HEAD zurück,
                                  nur Änderungen im Arbeitsverzeichnis betroffen

max@AmazinGit:~/GIT$ git checkout HEAD~1
                              ←   lädt den vorhergehenden Commit
```

Integrieren der Änderung in master

max@AmazinGit:~

```
                                zurück zum master

max@AmazinGit:~/GIT$ git checkout master
                                ←   wichtig: wechseln des Branch aktualisiert das
                                    Arbeitsverzeichnis / Änderungen ohne commit
                                    gehen verloren

                                Branch integrieren

max@AmazinGit:~/GIT$ git merge “Zweig1“
                                ←   Verschmilzt Branch mit Ausgangspunkt

                             unbenötigten Branch löschen

max@AmazinGit:~/GIT$ git branch -d “Zweig1“
                                  ←   entfernt Branch, kein checkout mehr mgl.

                            alternativ zu merge → rebase

max@AmazinGit:~/GIT$ git checkout „Zweig1“
max@AmazinGit:~/GIT$ git rebase “master“
                                  ←   setzt Ausgangspunkt von Zweig1 von
                                      ursprünglichen Commit auf HEAD von master
```

merge vs rebase



Rebase: -ändert Historie
-zeigt einzelne Konflikte
-schlecht reversible

→ gut für Einzelarbeit, streamlined log

Merge: -behält Historie
-alle Konflikte
-einfach reversible

→ für Gruppenarbeit, cluttered log

https://hackernoon.com/git-merge-vs-rebase-whats-the-diff-76413c117333

Konflikte

max@AmazinGit:~

Es können Konflikte bei merge auftreten, wenn die gleichen Dateien bearbeitet wurden. Diese müssen i.d.R manuell beseitigt werden.

```
max@AmazinGit:~/GIT$ git checkout -b zweig2
max@AmazinGit:~/GIT$ echo bonjour > test1
max@AmazinGit:~/GIT$ git commit test1 -m “branch commit“
                              ←   test1 in Branch geändert

max@AmazinGit:~/GIT$ git checkout master
max@AmazinGit:~/GIT$ echo hallo > test1
max@AmazinGit:~/GIT$ git commit test1 -m “master commit“
                              ←   test1 in master geändert

max@AmazinGit:~/GIT$ git merge zweig2
                              ←   merge verursacht Konflikt

max@AmazinGit:~/GIT$ gedit test1
                              ←   manuelle Bearbeitung der Datei

max@AmazinGit:~/GIT$ git commit -i test1 -m “merge Konflikt behoben“
                              ←   Commit mit behobenen Konflikt
```

Tagging

max@AmazinGit:~

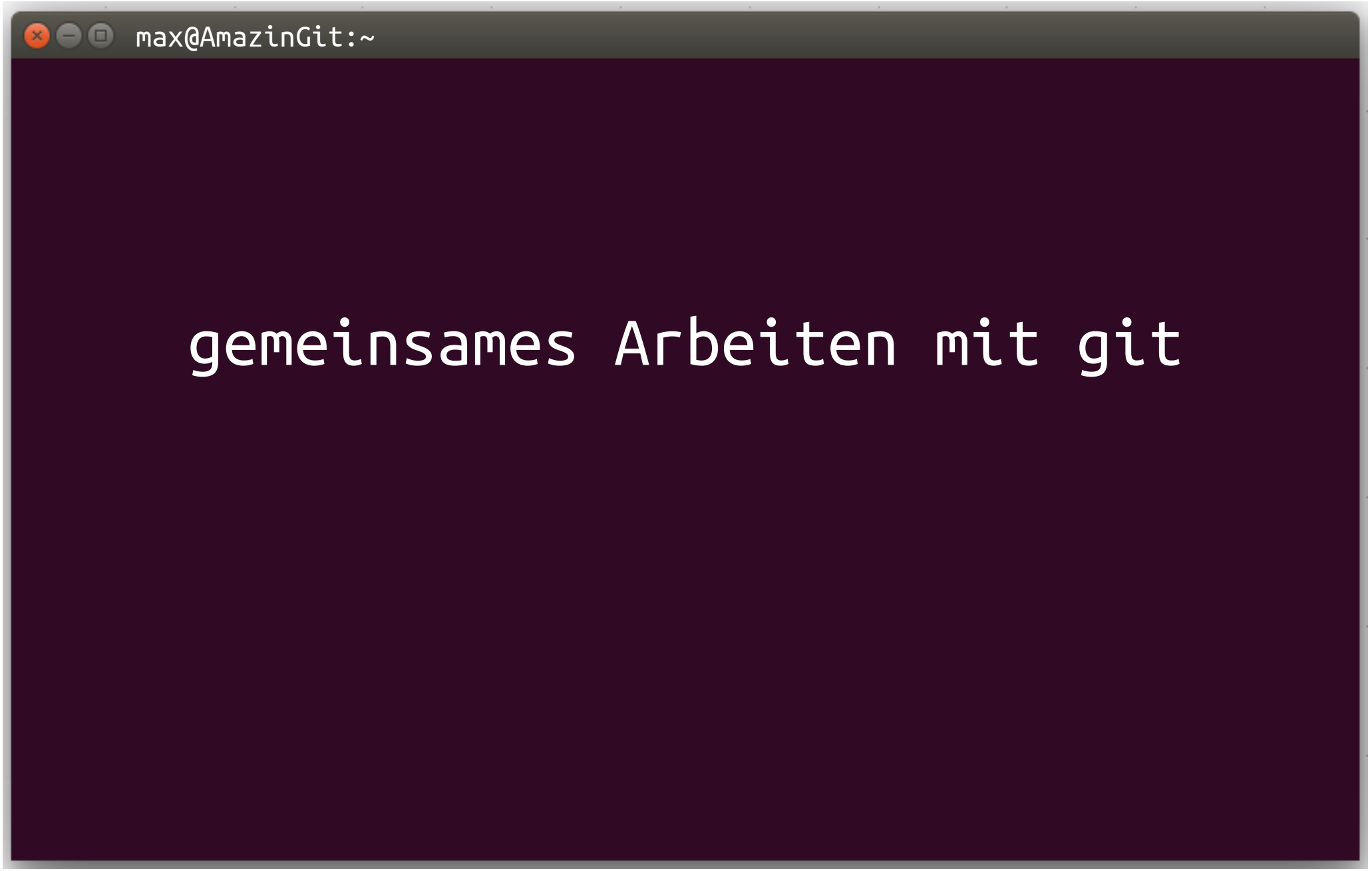
Ein Tag dient zur Markierung eines bestimmten Commits.

Erzeugen eines tag

```
max@AmazinGit:~/GIT$ git tag “v1.0.00“
                                    ←   markiert HEAD mit leightweight tag

max@AmazinGit:~/GIT$ git tag “v1.0.00“ 38ef811ed1e
                                    ←   markiert Commit mit entsprechendem
                                        Hash, Teil des Wertes reicht falls
                                        eindeutig

max@AmazinGit:~/GIT$ git tag -a “v1.0.01“ -m “tag-message“
                                    ←   markiert HEAD mit annotated tag
```

Ausgabe von tags

```
max@AmazinGit:~/GIT$ git tag
                                  ←   Liste aller Tags

max@AmazinGit:~/GIT$ git show “tag“
                                  ←   Ausgabe des Tags mit Nachricht, Autor,...
```


max@AmazinGit:~
gemeinsames Arbeiten mit git

SSH-Key

max@AmazinGit:~

Ein SSH-Schlüssel stellt die Verbindung zwischen TUC-Account und dem benutzten Rechner her. Anleitung unter https://gitlab.com/help/ssh/README.md

```
max@AmazinGit:~/GIT$ ssh-keygen -t ed25519 -C "email@example.com"
                                  ←   erzeugt SSH-Key pair

max@AmazinGit:~/GIT$ xclip -sel clip < ~/.ssh/id_ed25519.pub
                                  ←   kopiert public key
```

SSH-Key bei bei GitLab → Profile → Settings → SSH Key einfügen

externes Repository laden

max@AmazinGit:~

https://gitlab.hrz.tu-chemnitz.de/
ermöglicht das Erstellen und Teilen eines externen Repository über die TUC-ID.

externes Repository kopieren

```
max@AmazinGit:~$ mkdir GIT2
max@AmazinGit:~$ cd GIT2

max@AmazinGit:~/GIT2$ git clone git@gitlab.hrz.tu-chemnitz.de:nemax--tu-
chemnitz.de/procoski-2020-02-07.git
                                  ←   Kopiert ein bereits existierendes
                                      Repository
```

alternativ über https

```
max@AmazinGit:~/GIT2$ git clone https://gitlab.hrz.tu-chemnitz.de:nemax--tu-
chemnitz.de/procoski-2020-02-07.git
```

eigene Kopie aktualisieren

max@AmazinGit:~

Kopie des Repository akutalisieren

```
max@AmazinGit:~/GIT2$ git fetch origin
                                    ←   Lädt Änderungen im original Remote
max@AmazinGit:~/GIT2$ git merge origin/master
                                    ←   kombiniert Remote mit Master

                                        oder

max@AmazinGit:~/GIT2$ git pull
                                    ←   kombiniert fetch und merge (origin u.
                                        origin/master sind standard)


                                alternativ rebase

max@AmazinGit:~/GIT2$ git fetch origin
max@AmazinGit:~/GIT2$ git rebase origin/master

                                        oder

max@AmazinGit:~/GIT2$ git pull --rebase
                                    ←   kombiniert fetch und rebase
```

ext. Repository updaten

max@AmazinGit:~

Um eigene Änderungen auf das externe Repository zu laden muss zunächst ein merge/rebase mit einer aktuellen Kopie stattfinden.

eigene Änderungen laden

```
max@AmazinGit:~/GIT2$ git push
                              ←   gibt aktuellen Branch an origin weiter

max@AmazinGit:~/GIT2$ git push origin master
                              ←   gibt master an origin

max@AmazinGit:~/GIT2$ git push origin --all
                              ←   gibt alle Branches an origin weiter
```

Konflikte treten auf, wenn mehrere Personen gleichzeitig an den selben Stellen arbeiten. Man kann diesen durch Koordination und Branching entgegenwirken.