

Der Kurs

- Ihr sollt lernen, wie man **Git als Versionsverwaltungstool** nutzt und wie Ihr **erste Schritte** beim Entwickeln von **Open Source Projekten** macht.
- **Keine** Programmier-Vorkenntnisse benötigt, Grundkenntnisse Englisch
- Kurs ist an Lernende gerichtet, die
 - das Versionsverwaltungstool **Git** lernen wollen
 - verstehen wollen, wie **Open Source Software** entwickelt wird
 - bei Open Source Projekten **beitragen** möchten

Woche 1: Versionsverwaltung

Was ist ein **VCS**?

fetch

clone

Wie kann ich etwas **speichern**?

init

commit

Was ist ein **Repository**?

Was ist **Git**?

pull

push

Wie komme ich auf den **neuesten Stand**?

Woche 2: Kollaboration

checkout

Was ist ein **Branch**?

log

Wie kann ich etwas rückgängig machen?

reset

Was ist ein **merge Konflikt**?

Was ist das **Git Datenmodell**?

merge

push

Was ist **Open Source**?

Free Software

Github

Was ist Semantic Versioning?

Was ist ein **Pull Request**?

Git Flow

Feature

Was ist ein **Issue**?

Wie **organisiert** man sich in Open Source Projekten?

Woche 4: Beitragen

Wieso sollte ich **beitragen?**

Best Practices

Wie finde ich mein **erstes Issue?**

Continuous Integration

Wie stelle ich einen **Pull Request?**

Wo kann ich **anfangen?**

Was für **Tools** erleichtern mir die Arbeit?

Teaching Team



Sandro Speh

Competetive-Programming

DevOps

Modellierung



Caterina Mandel



Til Schniese

Skripting

Front-End

Projektmanagement

Back-End

Mobile-Development

Git

Theoretische-Informatik

Softwaretechnik

Design-Thinking

Automatisierung

Computer-Vision

Datenbanksysteme

Prozessanalyse

Open-Source

ITSE

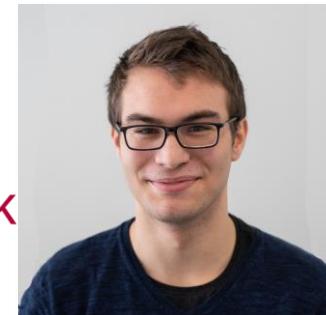
Web-Technologien



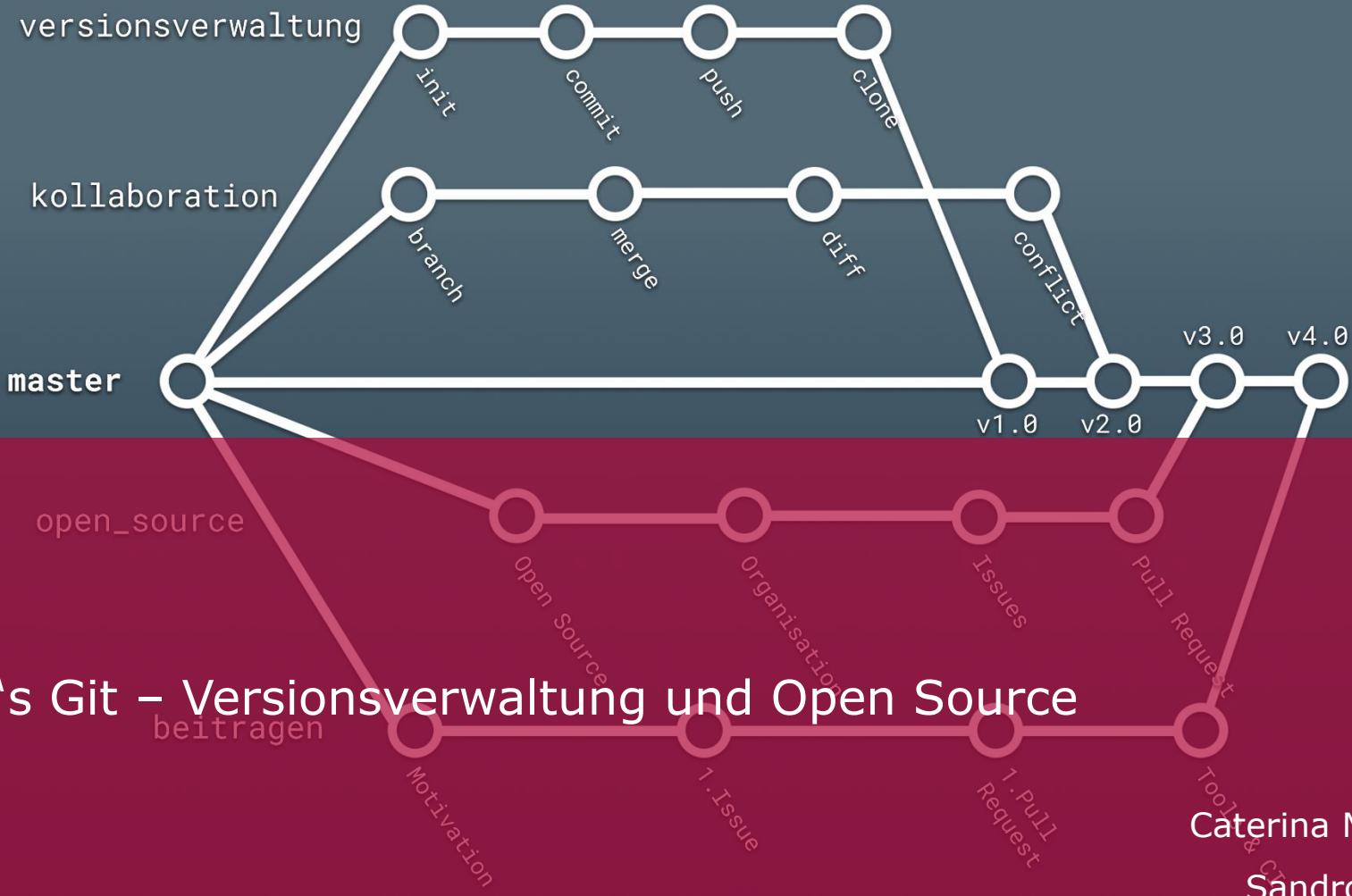
Marc Rosenau

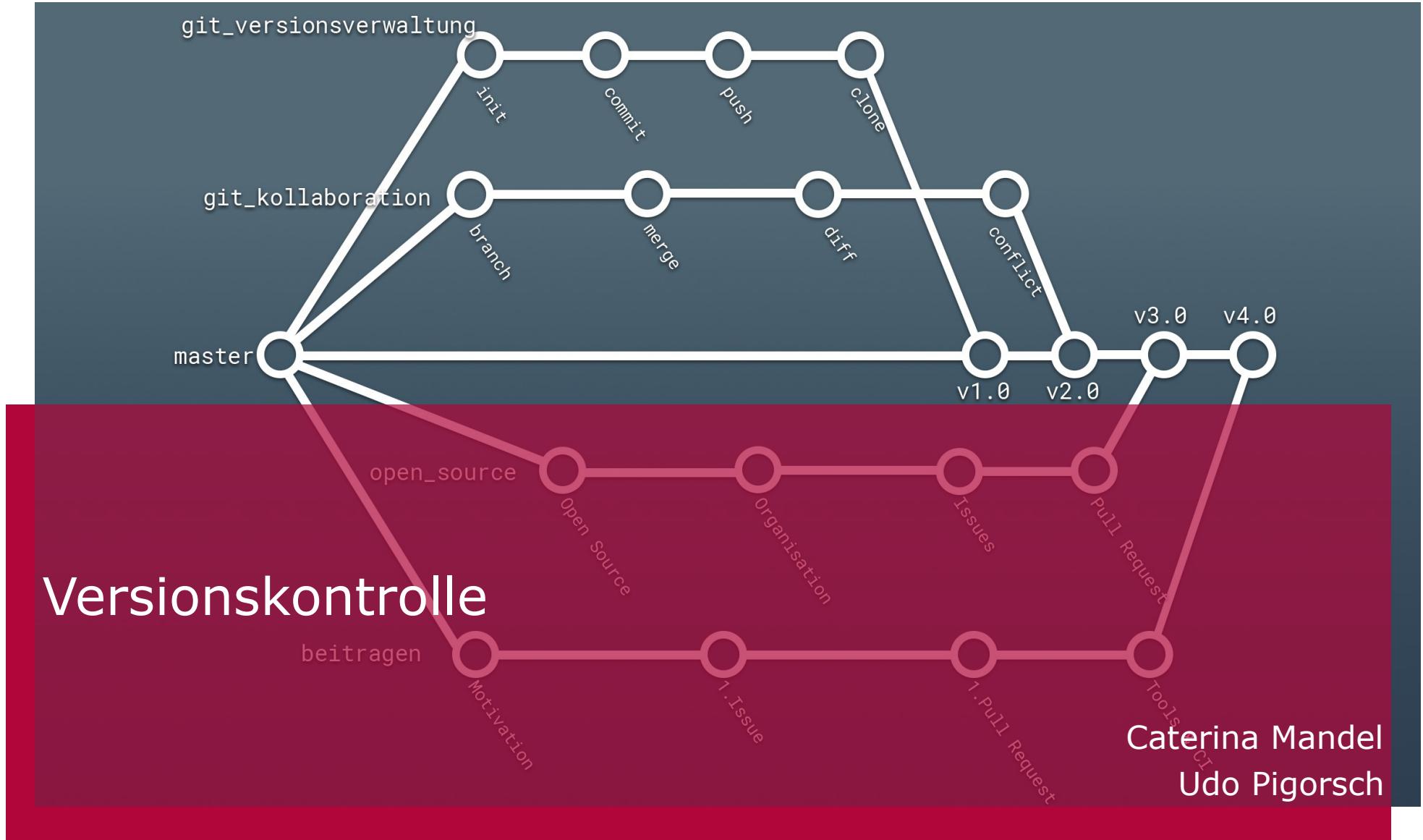


Udo Pigorsch



Leo Wendt





Begriffserklärung

Was ist eine Datei?

- z.B. Text (.txt), Bilder (.jpg), Musik (.mp3)

Begriffserklärung

Was ist eine Datei?

- z.B. Text (.txt), Bilder (.jpg), Musik (.mp3)

Was ist eine Version?

- beschreibt den Inhalt einer Datei zu bestimmten Zeitpunkt
- jede Datei besitzt mindestens eine Version
- Änderungen der Datei erzeugen eine neue Version

Begriffserklärung



Begriffserklärung

Was ist eine Datei?

- z.B. Text (.txt), Bilder (.jpg), Musik (.mp3)

Was ist eine Version?

- beschreibt den Inhalt einer Datei zu bestimmten Zeitpunkt
- jede Datei besitzt mindestens eine Version
- Änderungen der Datei erzeugen eine neue Version

Was ist Versionskontrolle?

- Verwaltung verschiedener Versionen von Dateien

Begriffserklärung

Was ist eine Datei?

- z.B. Text (.txt), Bilder (.jpg), Musik (.mp3)

Was ist eine Version?

- beschreibt den Inhalt einer Datei zu bestimmten Zeitpunkt
- jede Datei besitzt mindestens eine Version
- Änderungen der Datei erzeugen eine neue Version

Was ist Versionskontrolle?

- Verwaltung verschiedener Versionen von Dateien

Warum Versionskontrolle interessant ist?

- ermöglicht digitale Zeitreisen durch die Versionen unserer Dateien

Versionskontrolle



Versionskontrolle



- Datei



- DateiKopie1Geschnitten



- DateiKopie2FarbeBearbeitet



- DateiKopie3Backup

Formen der Versionskontrolle

Lokale Versionskontrolle

- Version Control System - VCS

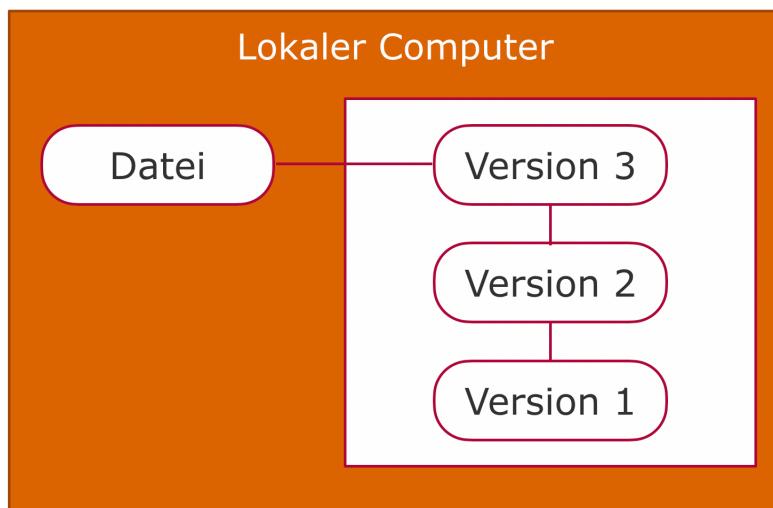
Zentralisierte Versionskontrolle

- Centralised Version Control System - CVCS

Verteilte Versionskontrolle

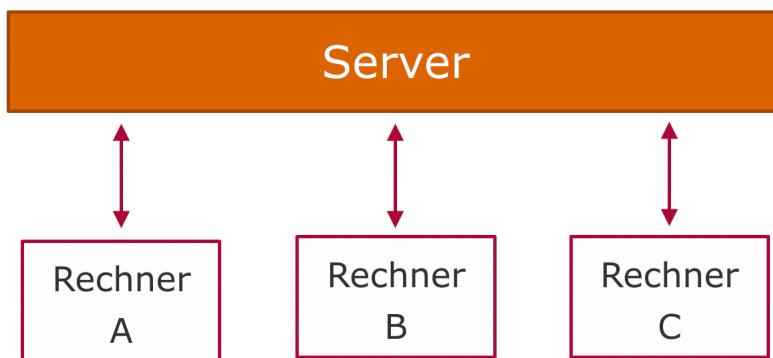
- Distributed Version Control System - DVCS

Formen der Versionskontrolle: Lokale Versionskontrolle



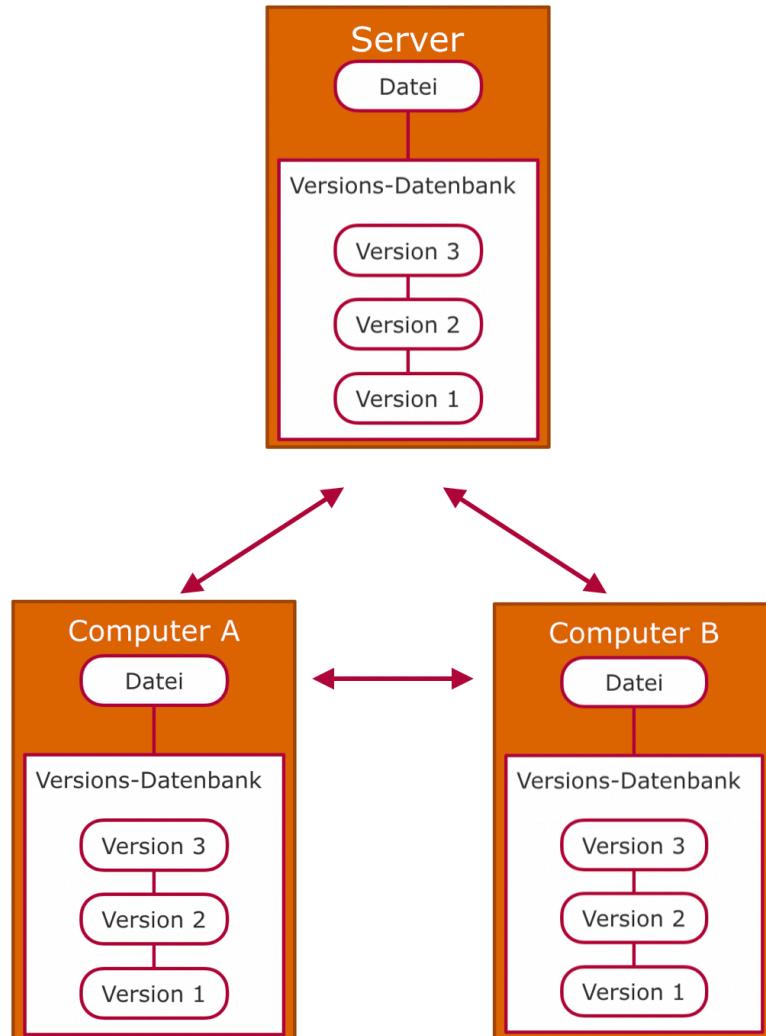
- Unterschiede zwischen Versionen (Patch-Sets) werden gespeichert
- Patch-Sets können kombiniert werden, um die Datei eines Zeitpunktes wiederherzustellen

Formen der Versionskontrolle: Zentralisierte Versionskontrolle



- Ein Server enthält alle versionierten Dateien, Klienten laden diese
- Verbessert die Verwaltung und ermöglicht Kontrolle „wer darf was“
- Single Point of Failure - Ausfall des Servers bedroht die Arbeit aller

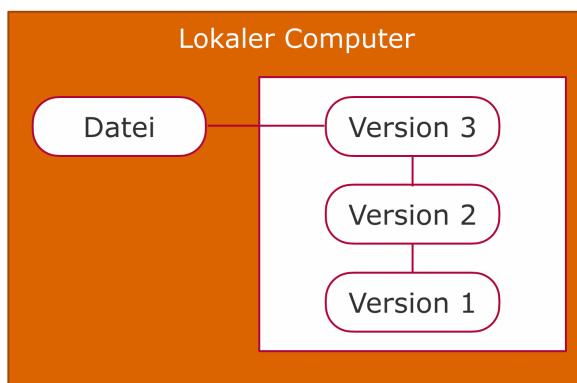
Formen der Versionskontrolle: Verteilte Versionskontrolle



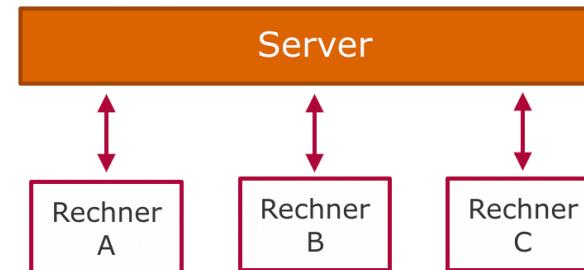
- Klienten spiegeln das Repository mit seiner vollständigen Historie
- Kein Single Point of Failure

Formen der Versionskontrolle

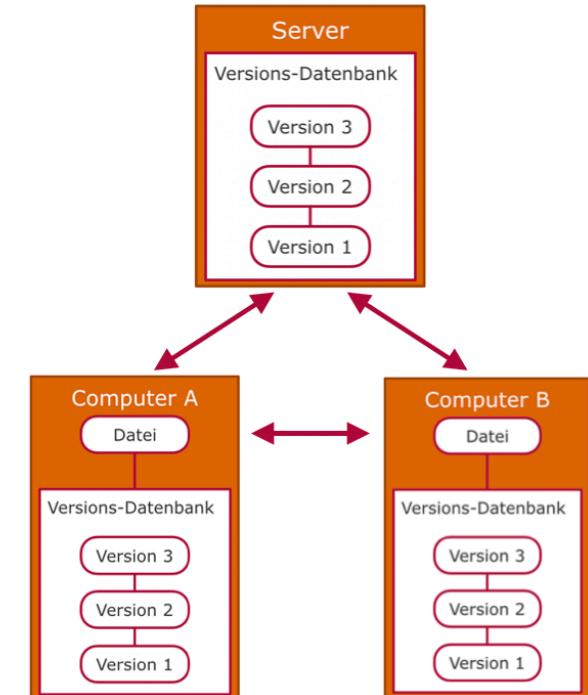
Lokale
Versionskontrolle

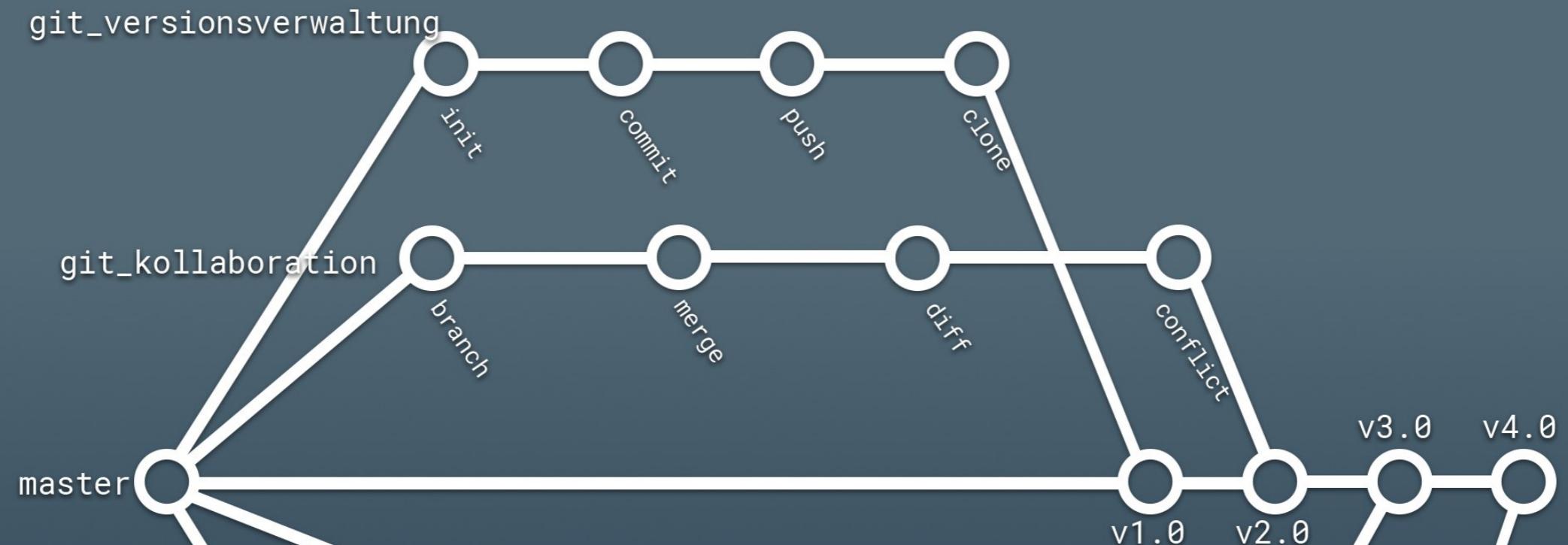


Zentralisierte
Versionskontrolle



Verteilte
Versionskontrolle





Die Entstehungsgeschichte von Git

beitragen

Motivation

open_source

open source

1.Issue

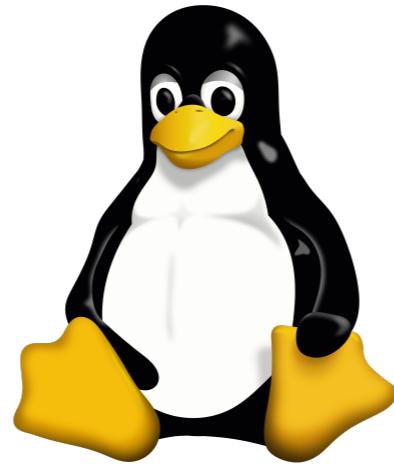
Organisation

1.Pull Request

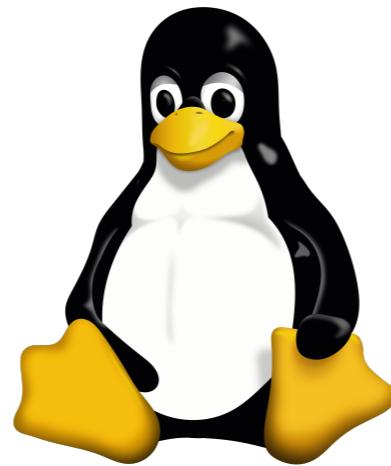
Issues

Caterina Mandel
Udo Pigorsch

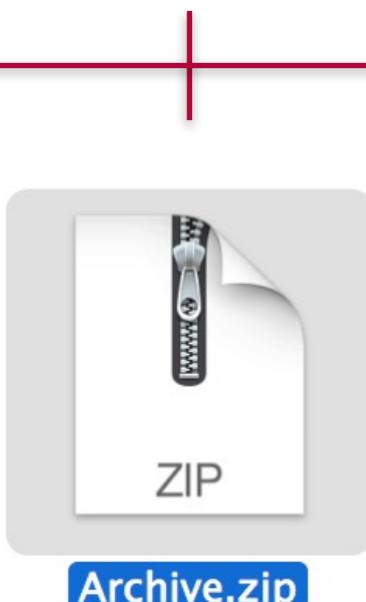
Die Entstehungsgeschichte von Git



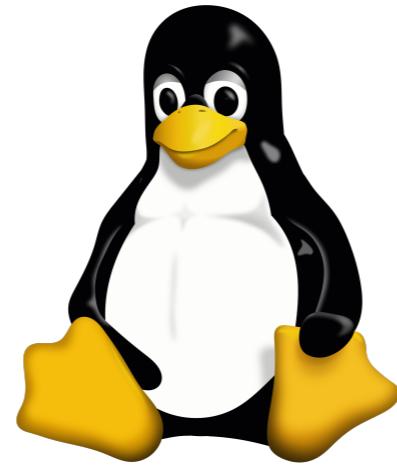
Die Entstehungsgeschichte von Git



1991

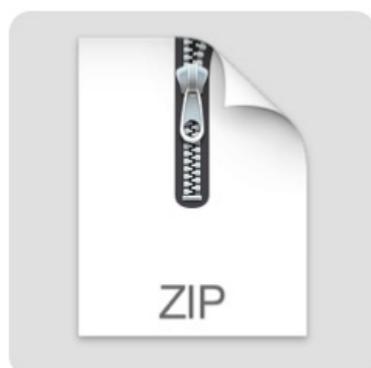


Die Entstehungsgeschichte von Git



1991

2002

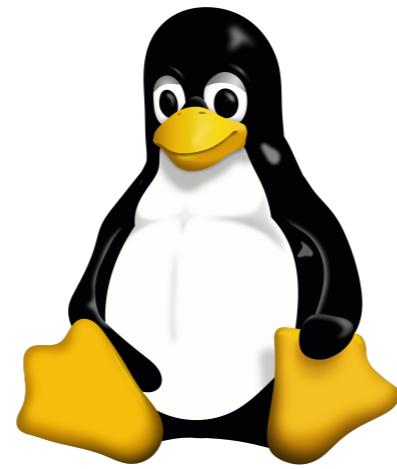


ZIP

Archive.zip

**DVCS
BitKeeper**

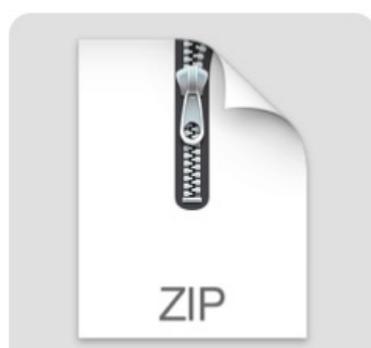
Die Entstehungsgeschichte von Git



1991

2002

2005

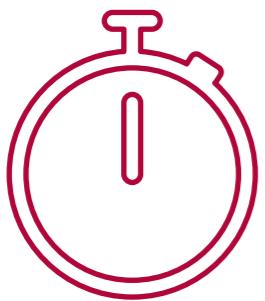


Archive.zip

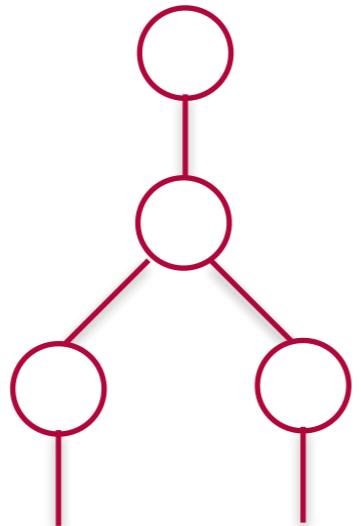
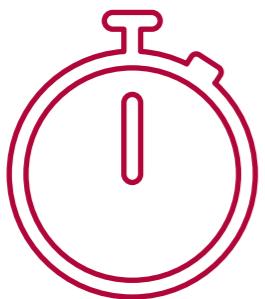
DVCS
BitKeeper



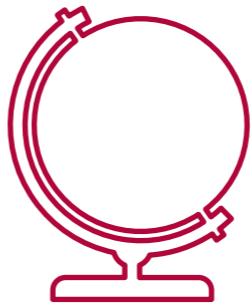
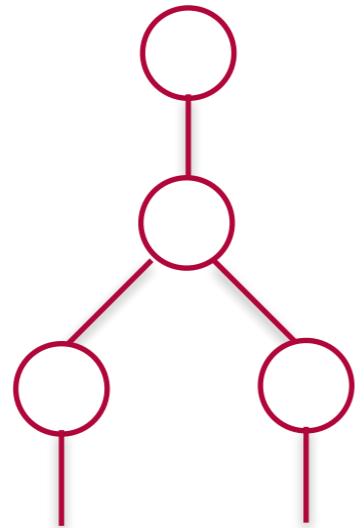
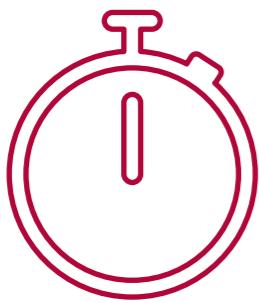
Anforderungen der Entwickler an Git



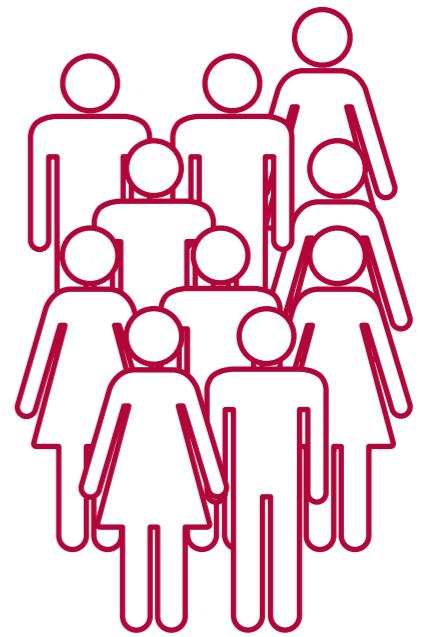
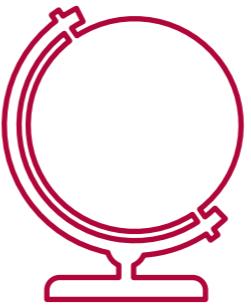
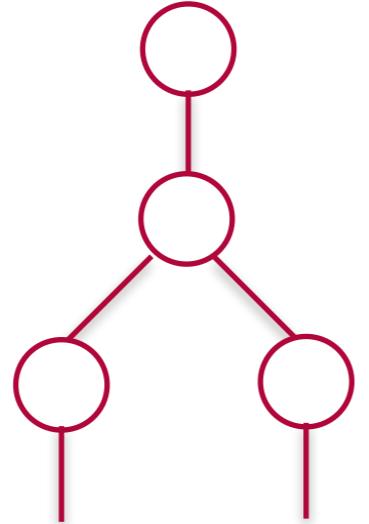
Anforderungen der Entwickler an Git

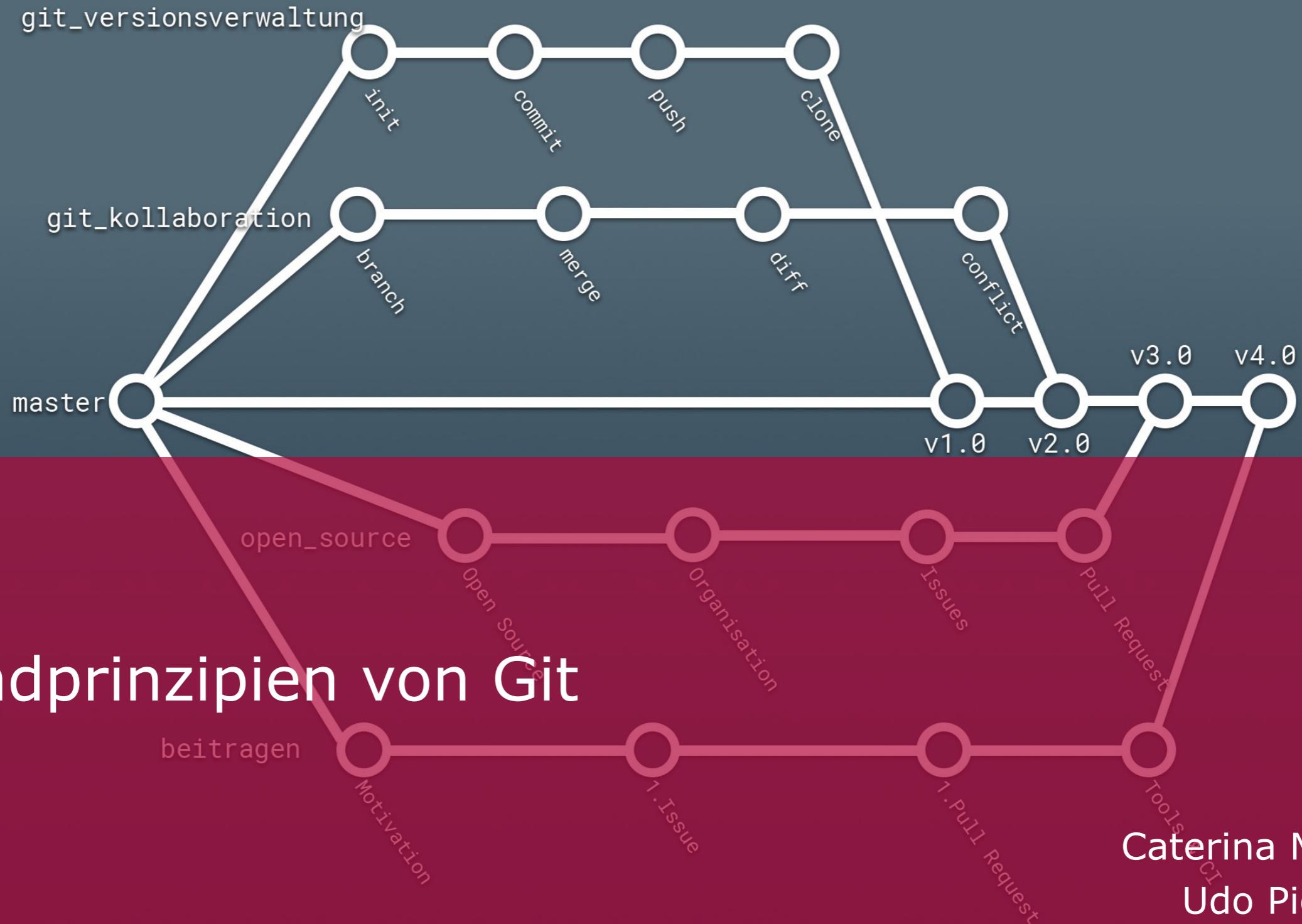


Anforderungen der Entwickler an Git



Anforderungen der Entwickler an Git





Grundprinzipien von Git

1. Snapshots, statt Unterschiede
2. Fast jede Operation ist lokal
3. Integrität
4. Git fügt Daten hinzu
5. Die drei Zustände

Snapshots vs. Unterschiede

Check-Ins over Time

Version 1

Datei A

Datei B

Datei C

Snapshots vs. Unterschiede

Check-Ins over Time

Version 1

Version 2

Datei A

Datei B

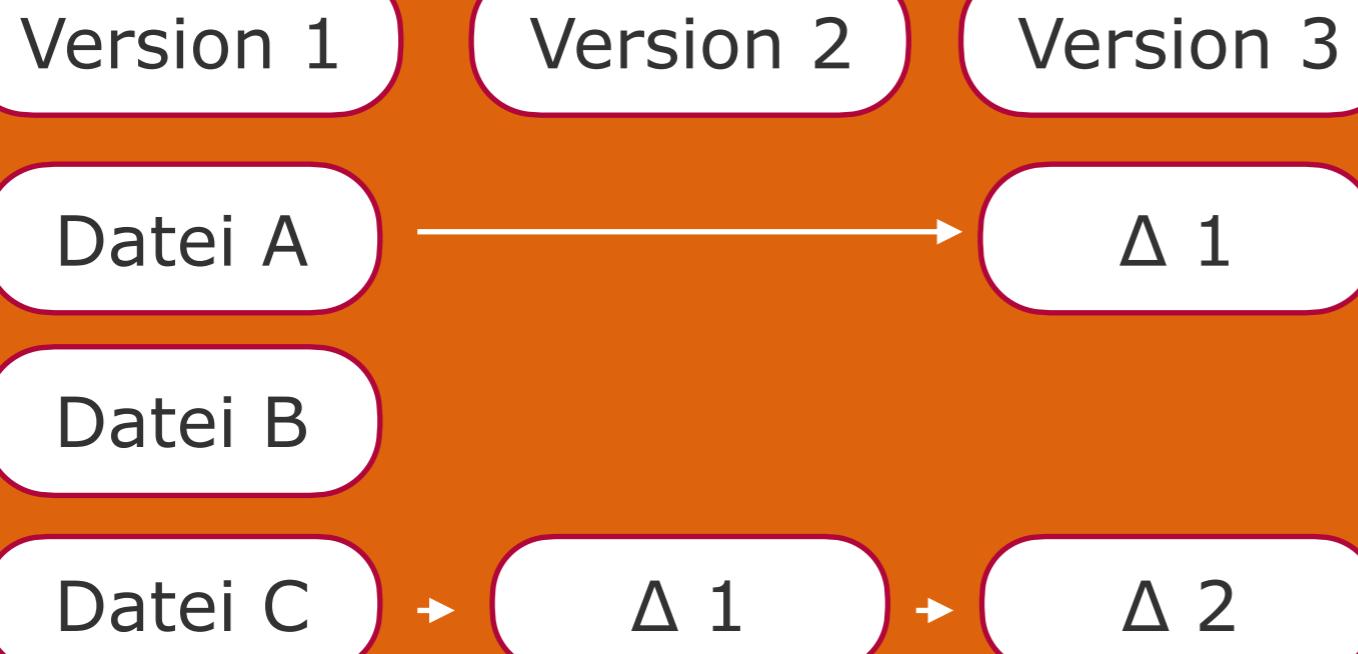
Datei C



$\Delta 1$

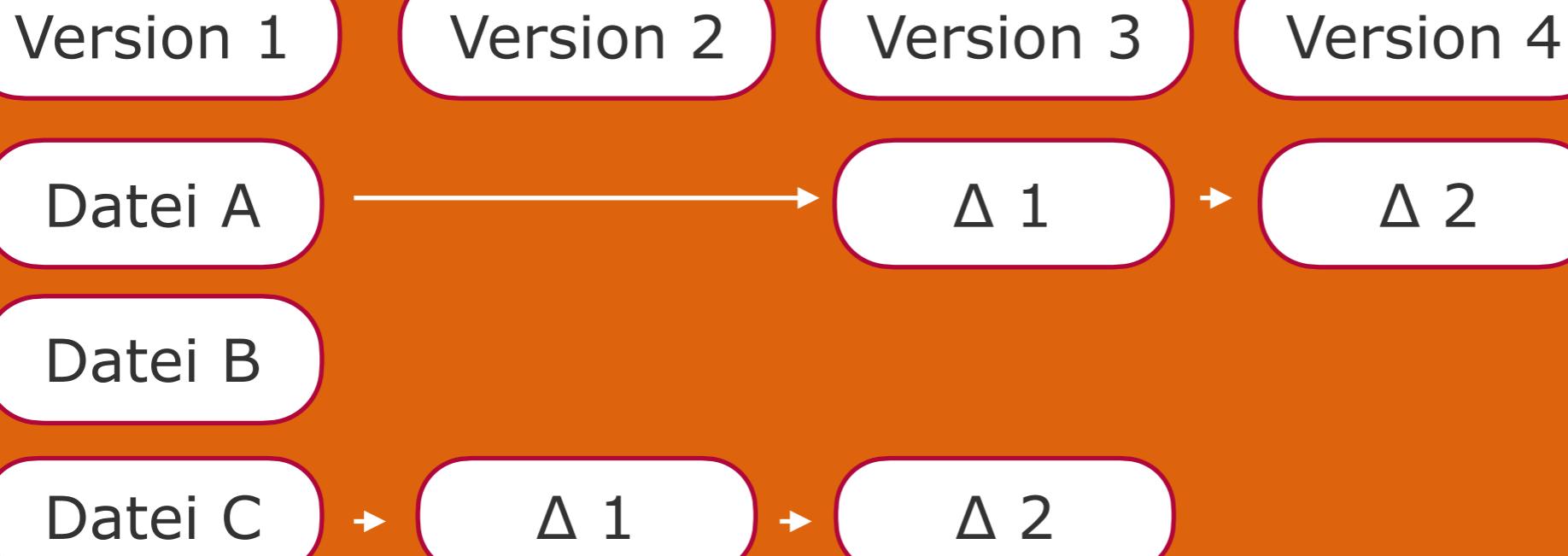
Snapshots vs. Unterschiede

Check-Ins over Time



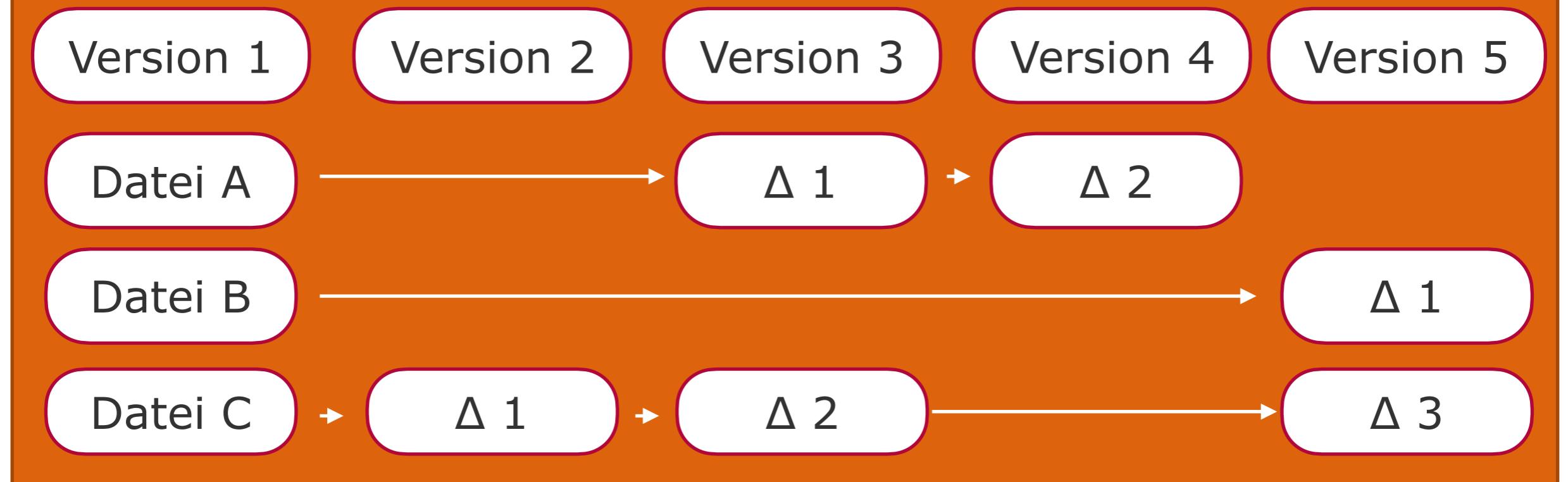
Snapshots vs. Unterschiede

Check-Ins over Time



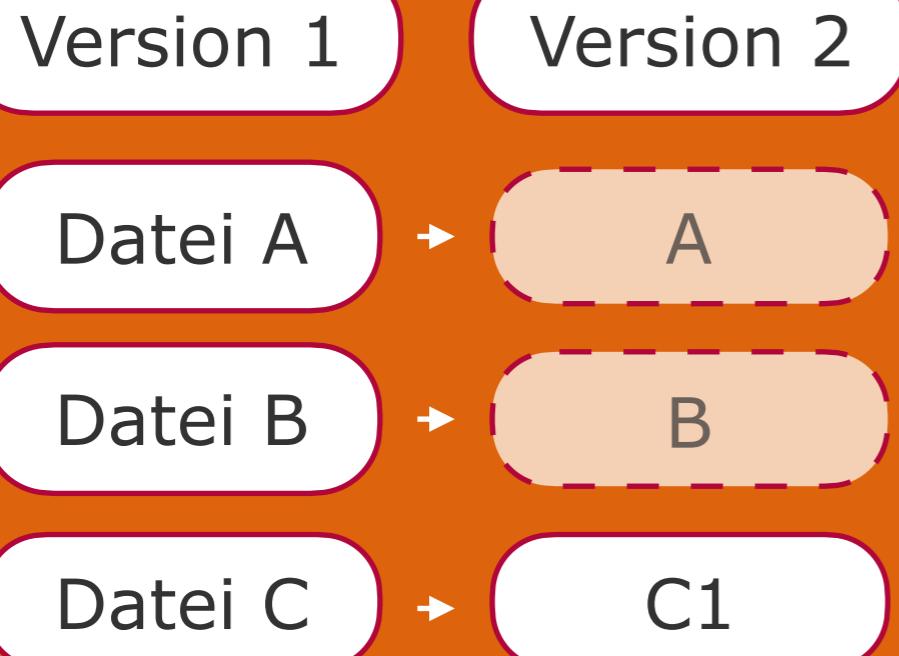
Snapshots vs. Unterschiede

Check-Ins over Time



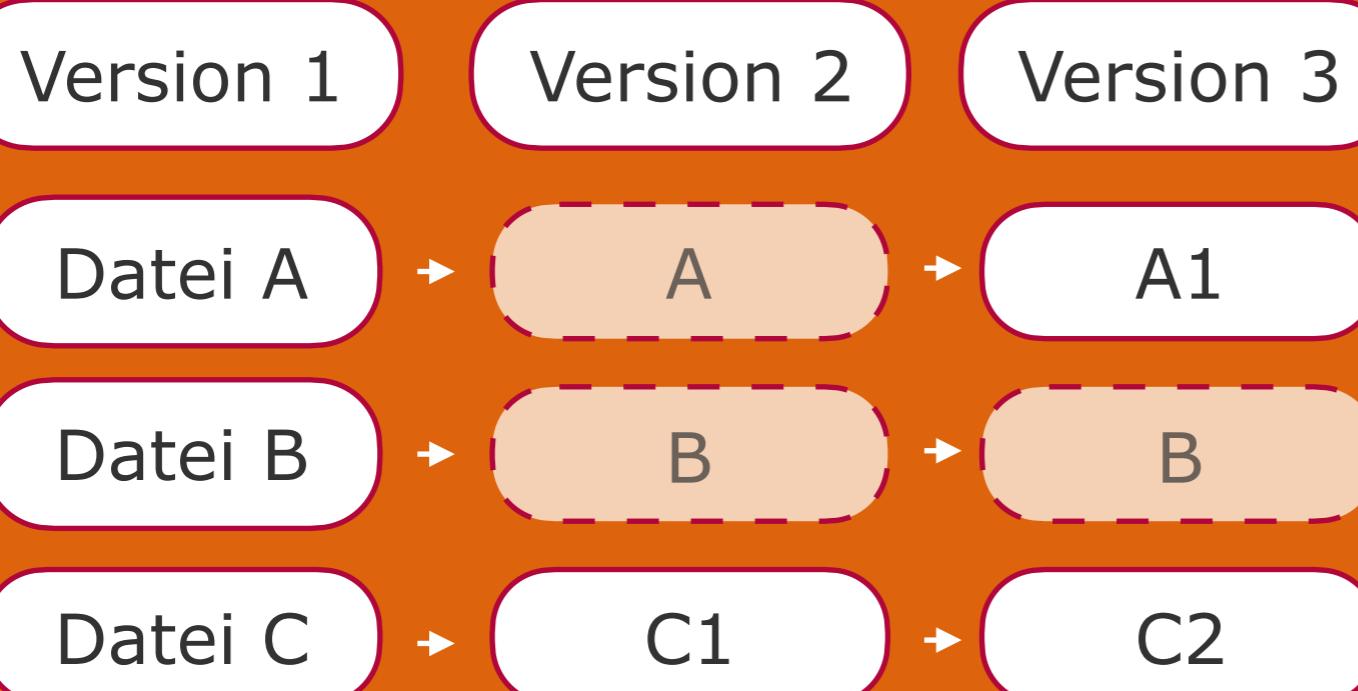
Snapshots vs. Unterschiede

Check-Ins over Time



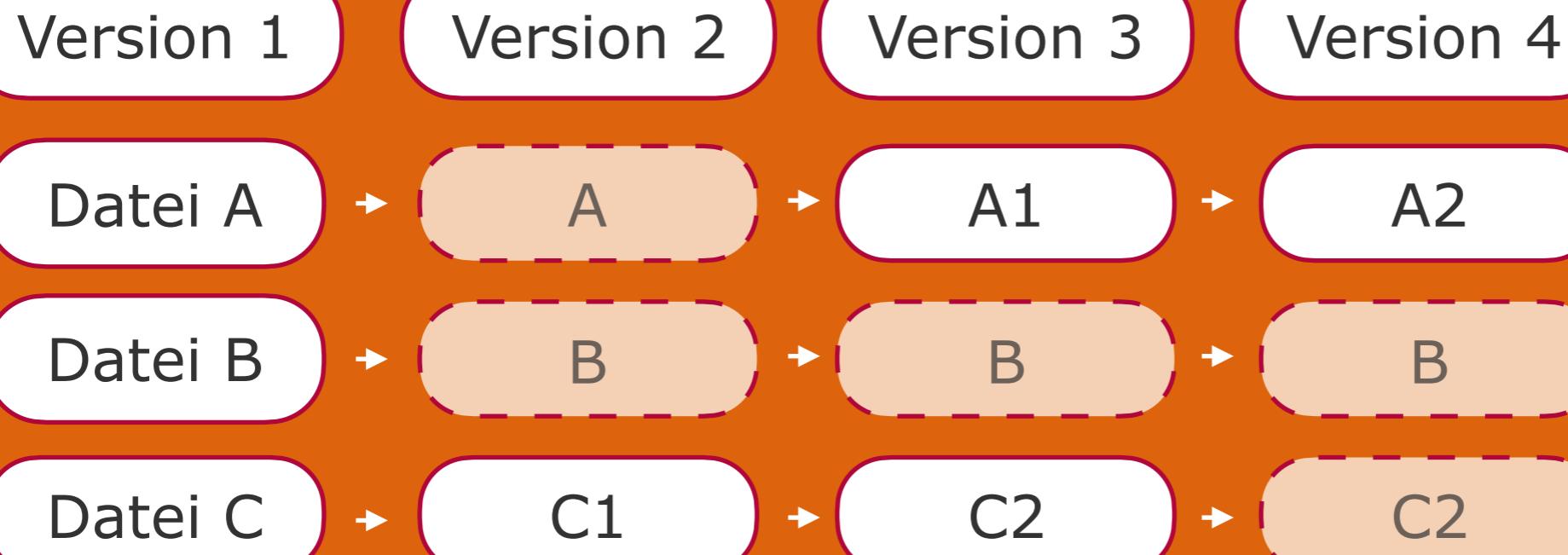
Snapshots vs. Unterschiede

Check-Ins over Time



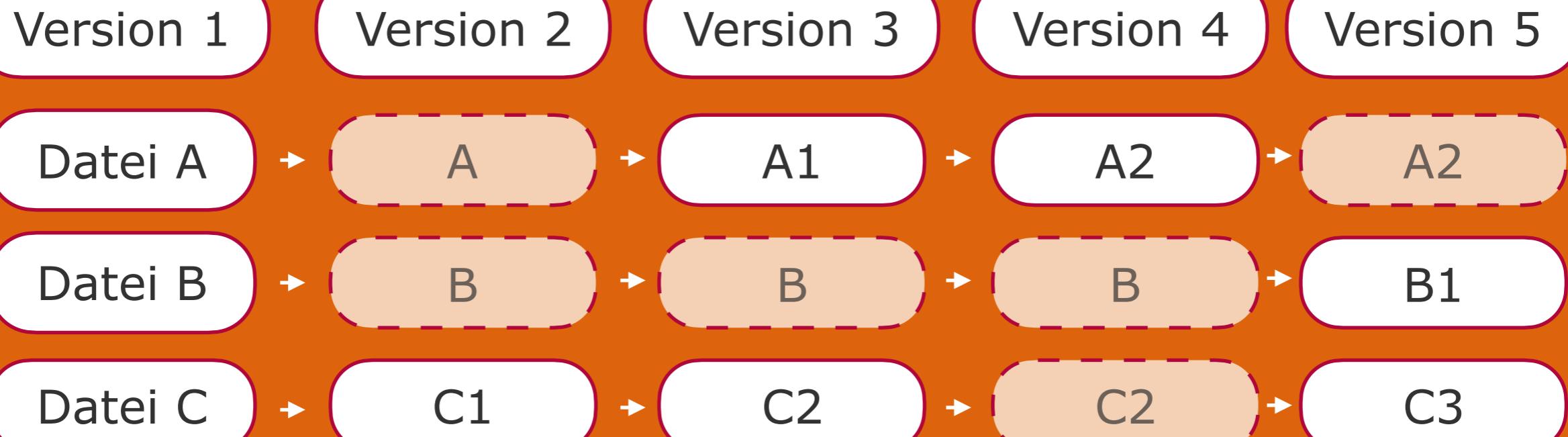
Snapshots vs. Unterschiede

Check-Ins over Time



Snapshots vs. Unterschiede

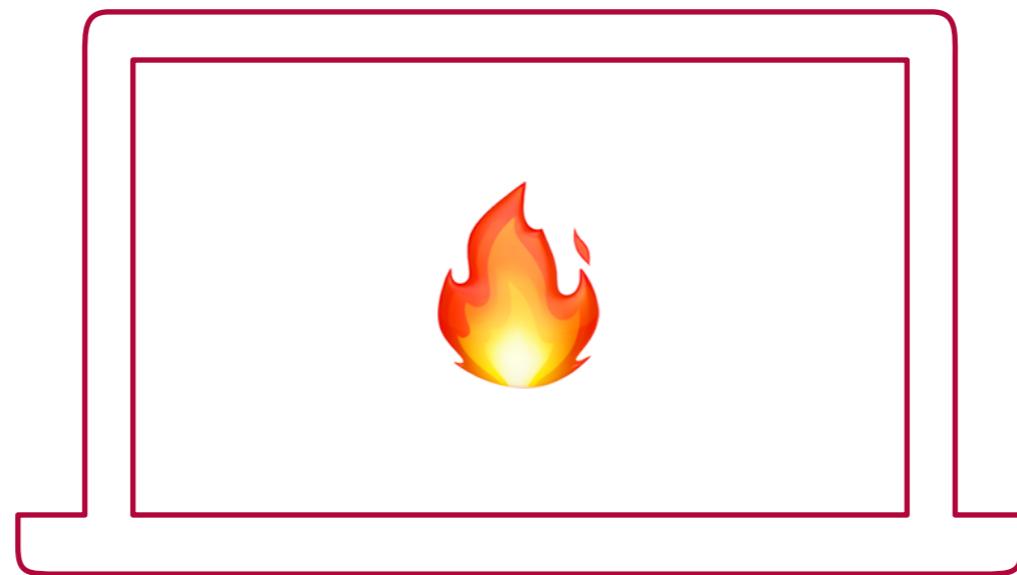
Check-Ins over Time



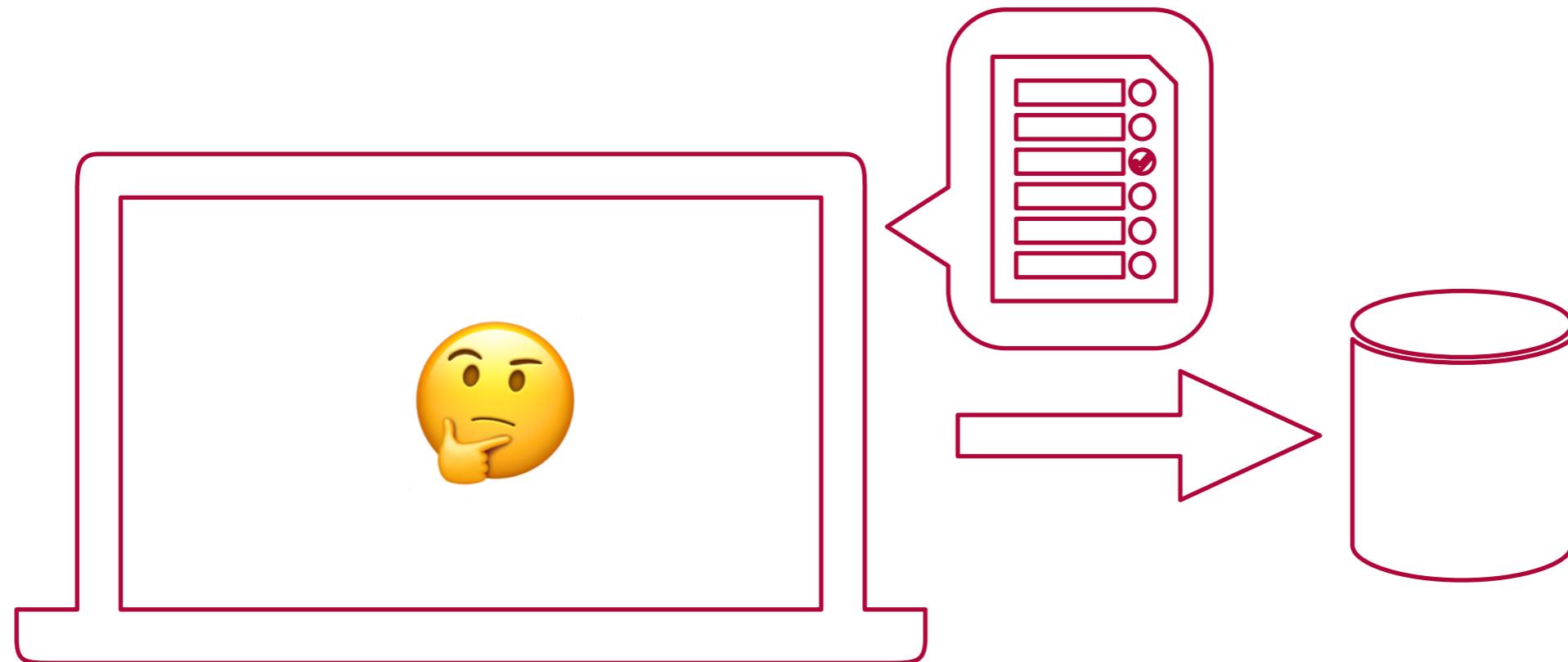
Grundprinzipien von Git

1. Snapshots, statt Unterschiede ✓
2. Fast jede Operation ist lokal
3. Integrität
4. Git fügt Daten hinzu
5. Die drei Zustände

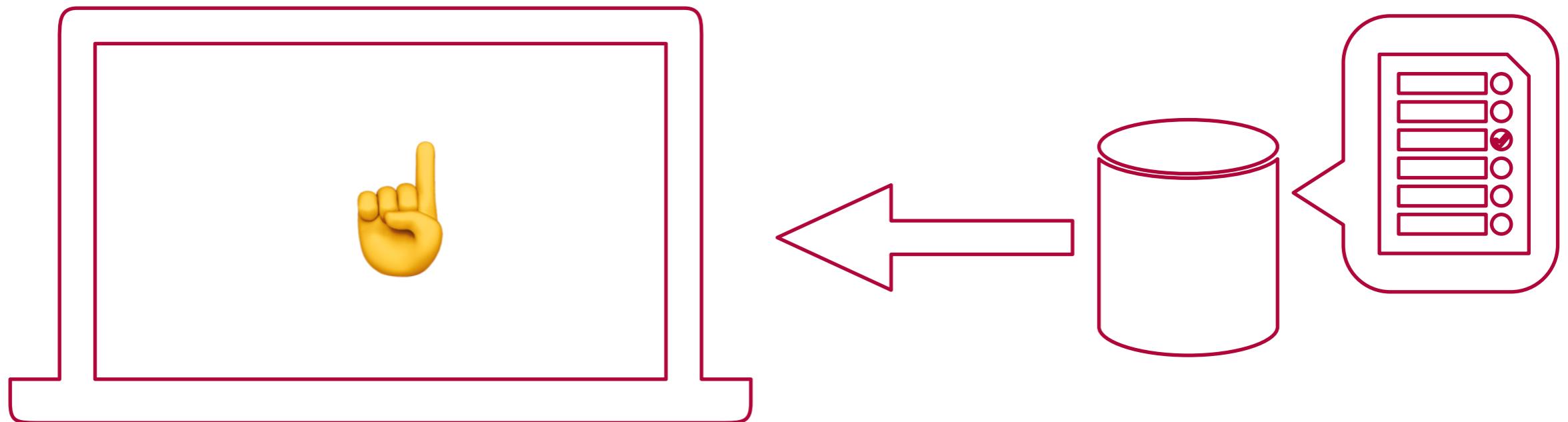
Performance



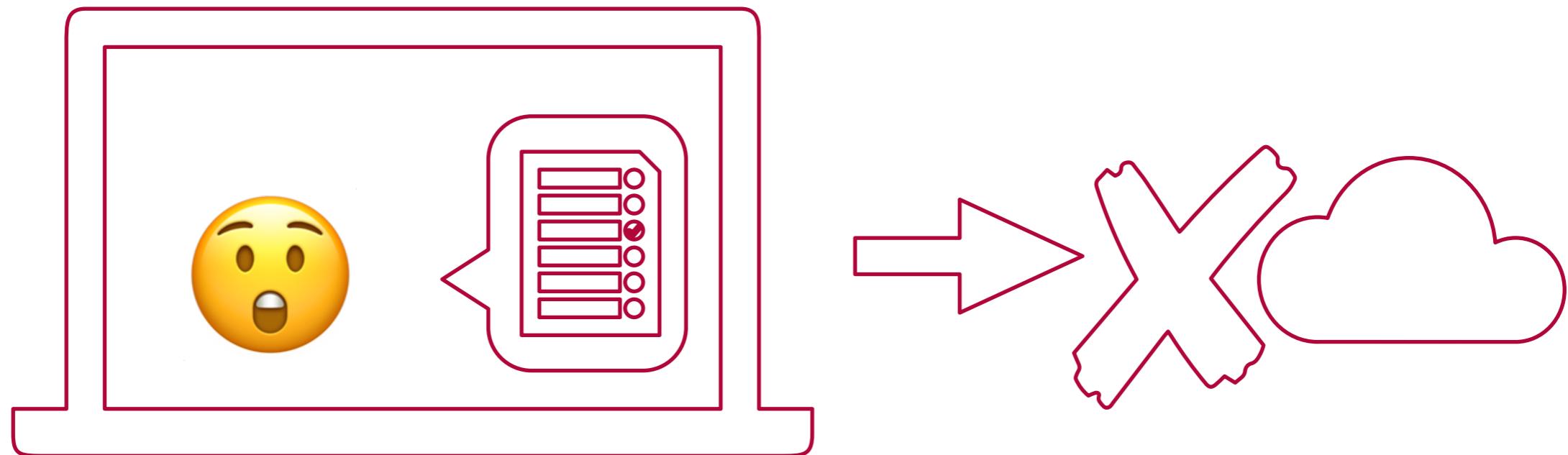
Fast jede Operation ist lokal



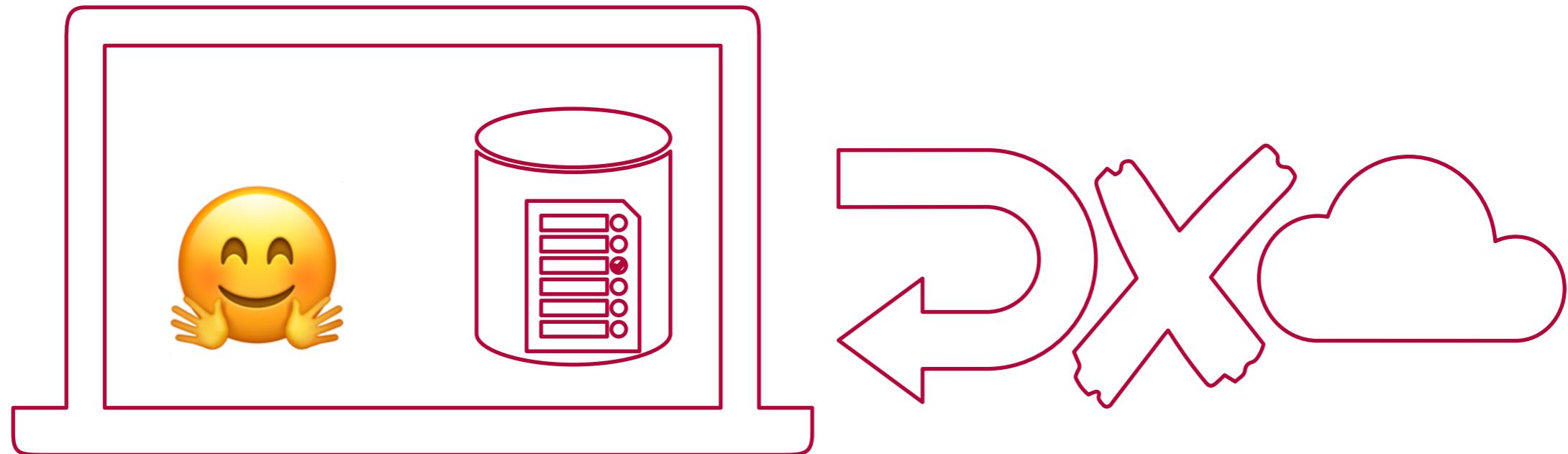
Fast jede Operation ist lokal



Fast jede Operation ist lokal



Fast jede Operation ist lokal



Grundprinzipien von Git

1. Snapshots, statt Unterschiede ✓
2. Fast jede Operation ist lokal ✓
3. Integrität
4. Git fügt Daten hinzu
5. Die drei Zustände



24b9da6552252987



24b9da65522525as

Grundprinzipien von Git

1. Snapshots, statt Unterschiede ✓
2. Fast jede Operation ist lokal ✓
3. Integrität ✓
4. Git fügt Daten hinzu
5. Die drei Zustände

Snapshots vs. Unterschiede

Check-Ins over Time

Version 1

Version 2

Version 3

Version 4

Version 5

Datei A

A

A1

A2

A2

Datei B

B

B1



Datei C

C1

C2

Grundprinzipien von Git

1. Snapshots, statt Unterschiede ✓
2. Fast jede Operation ist lokal ✓
3. Integrität ✓
4. Git fügt Daten hinzu ✓
5. Die drei Zustände

Die drei Zustände



Modified

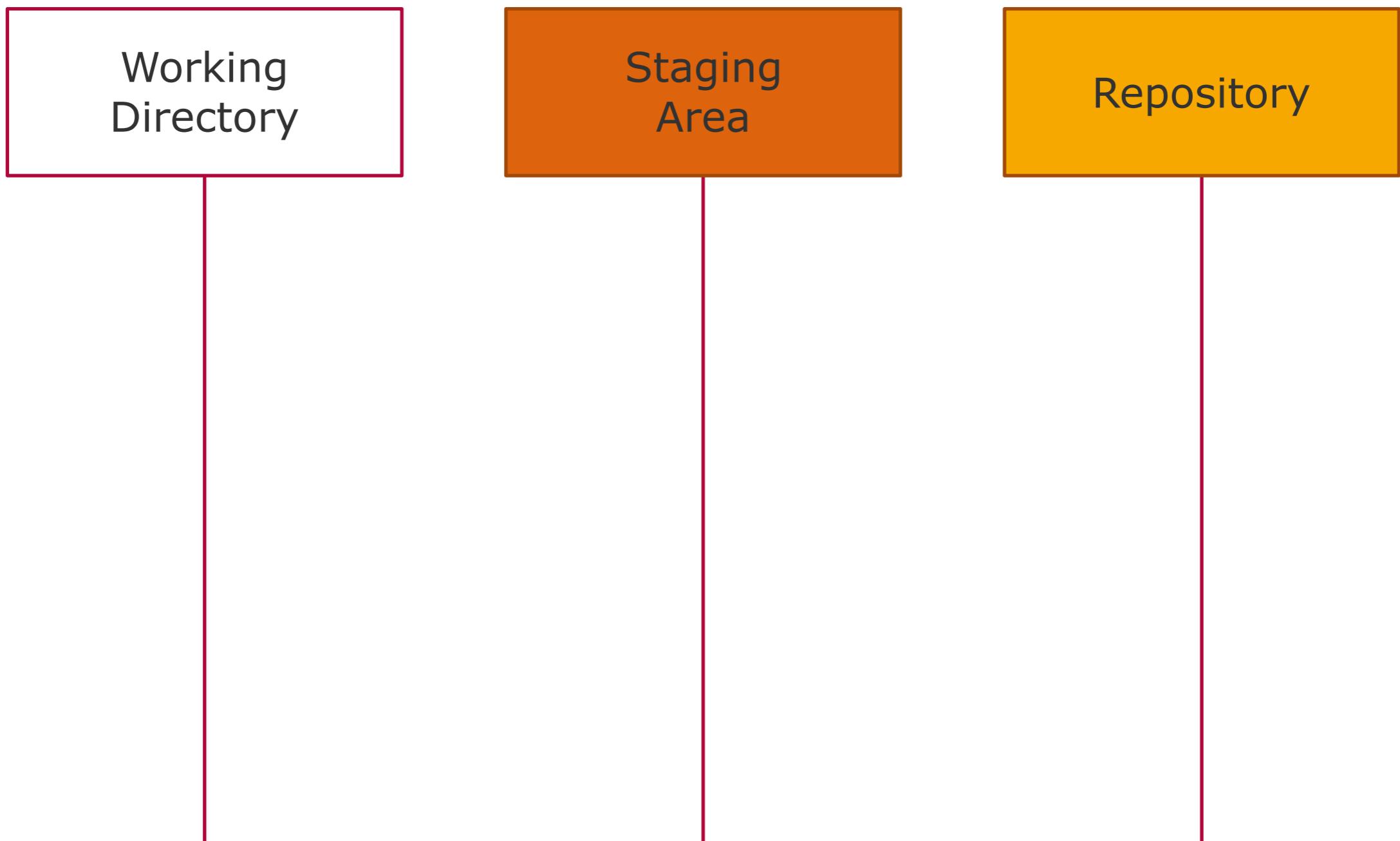


Staged

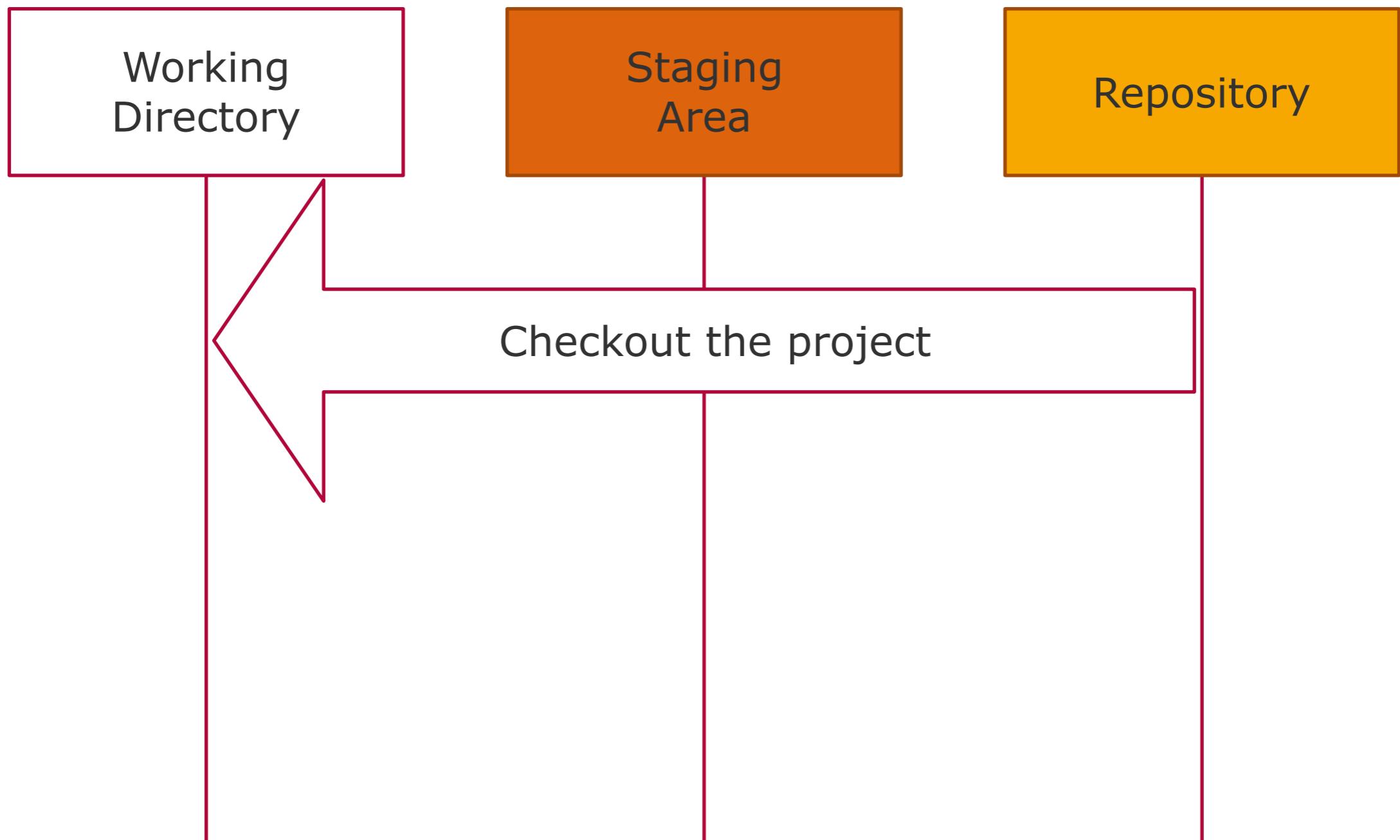


Committed

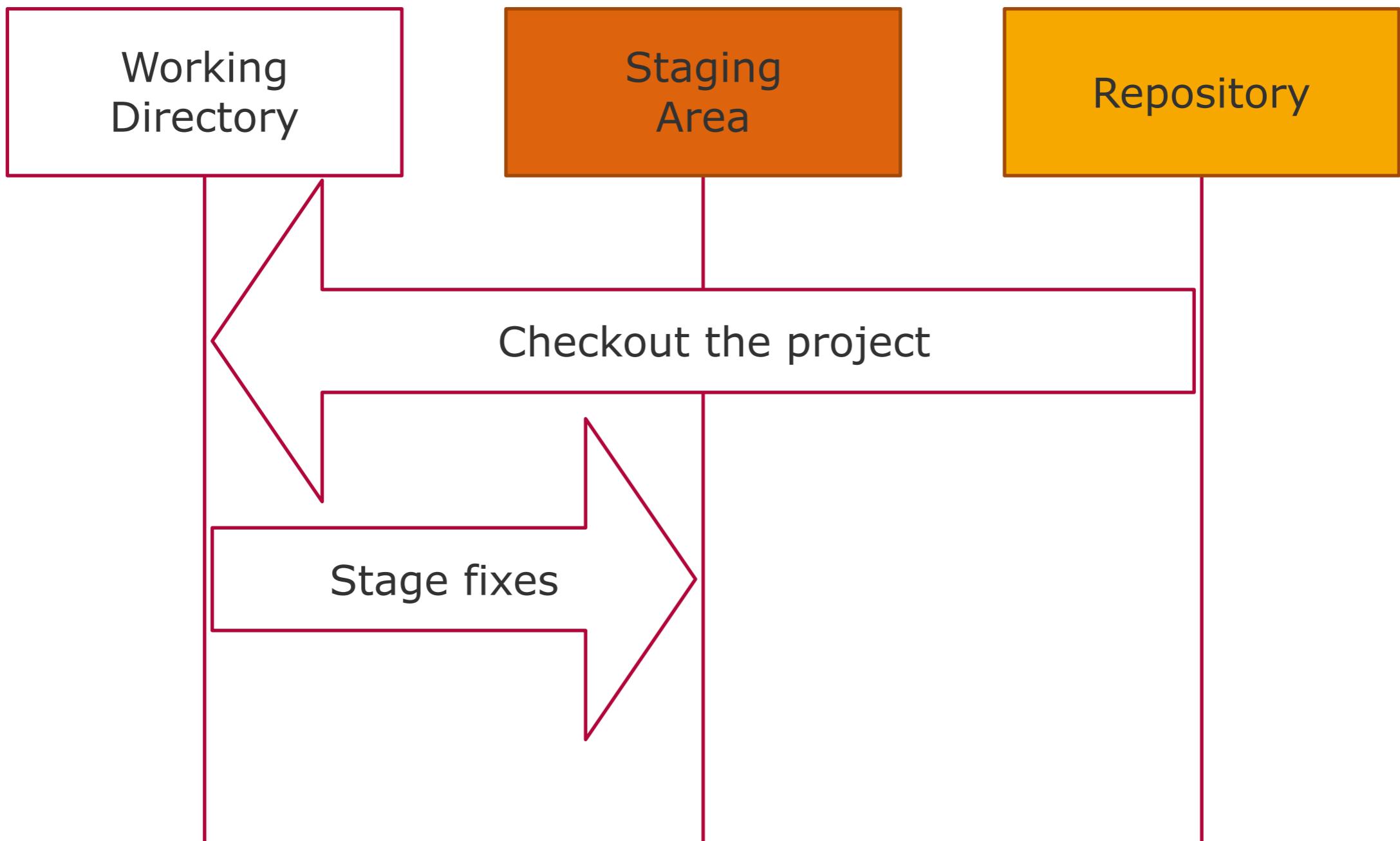
Die drei States



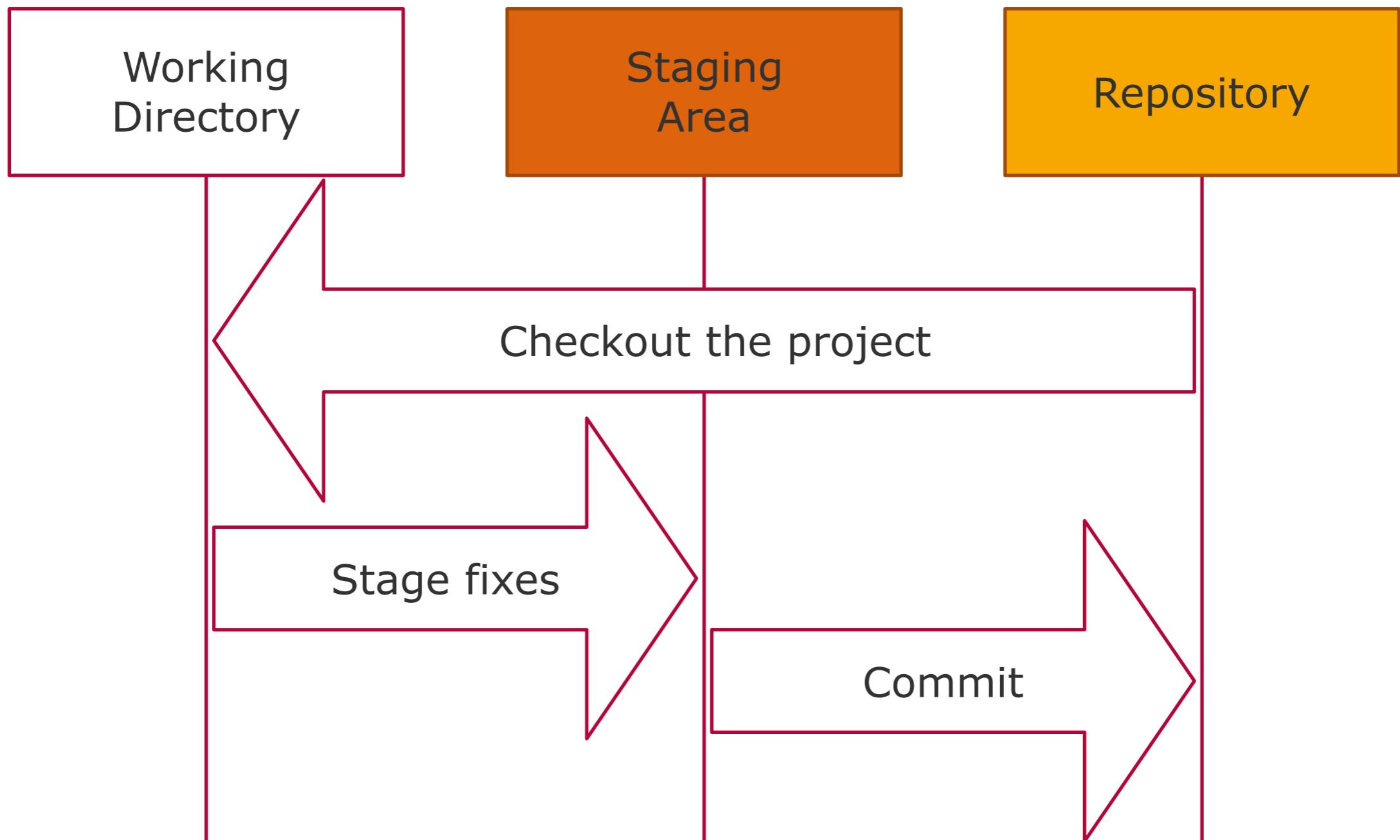
Die drei States



Die drei States



Die drei States



Git cheat sheet

Git verstehen

Snapshots, lokale, Integrität, fügt Daten hinzu

Modified ↔ Staged ↔ Committed

Git installieren

Paketmanager bzw. <https://git-scm.com/download>

Erstelle ein leeres Git-Repository / Reinitialisiere ein vorhandenes

\$ git init

Datei zum Index hinzufügen

\$ git add

Datei aus dem Arbeitsbaum und aus dem Index entfernen

\$ git rm

Änderungen am Repository aufzeichnen

\$ git commit

Verwalte einen Satz von verfolgten Repositories

\$ git remote

Klone ein Repository in ein neues Verzeichnis

\$ git clone

Aktualisiere die Remote-Repositories zusammen mit den zugehörigen Objekten

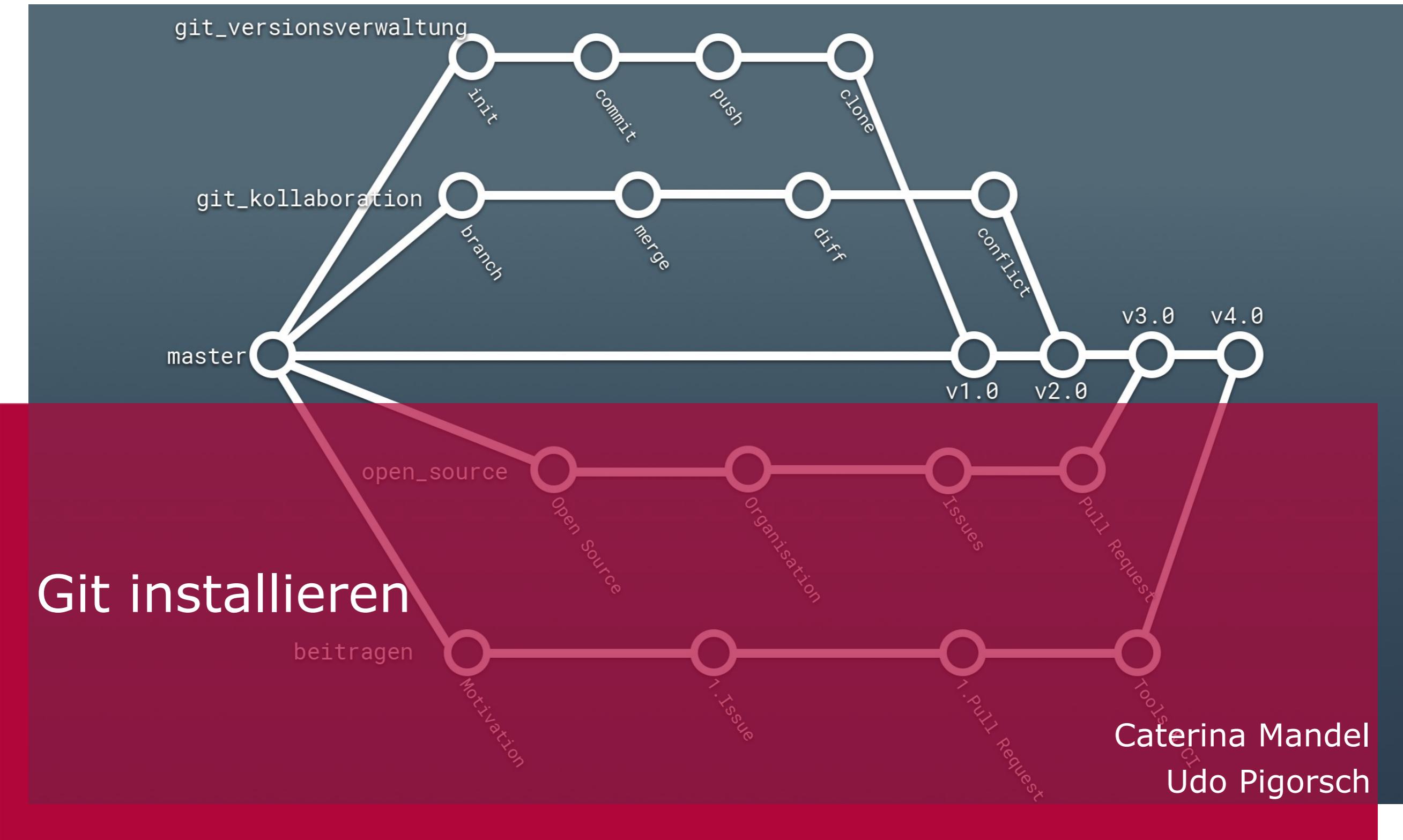
\$ git push

Objekte und Referenzen aus einem anderen Repository herunterladen

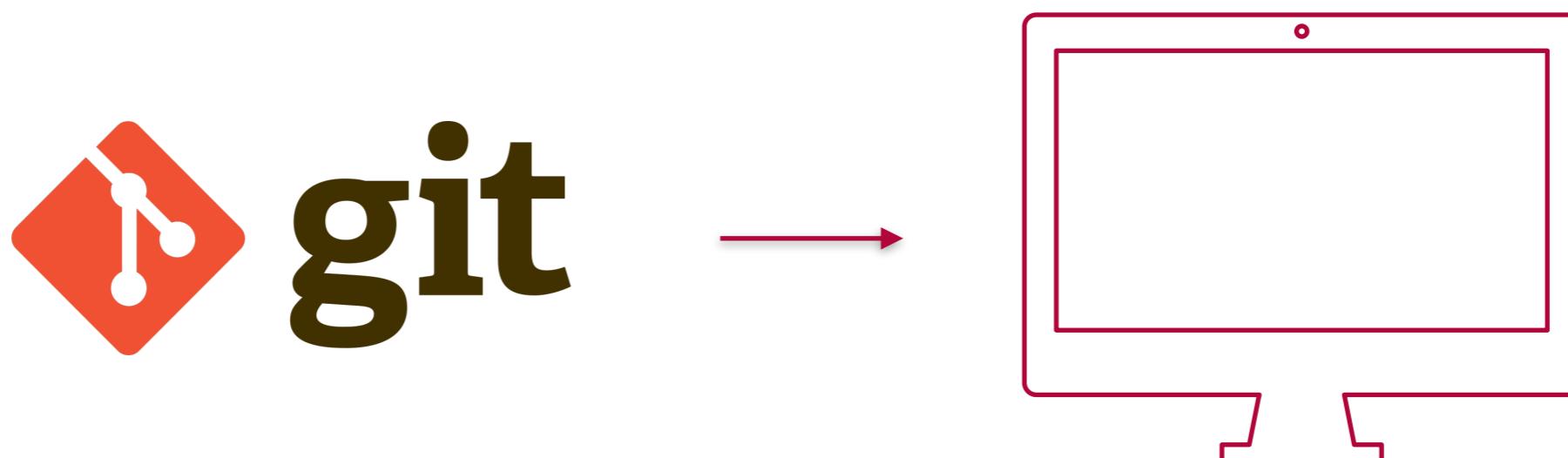
\$ git fetch

Aus einem anderen Repository oder einem lokalen Zweig holen und in diesen integrieren

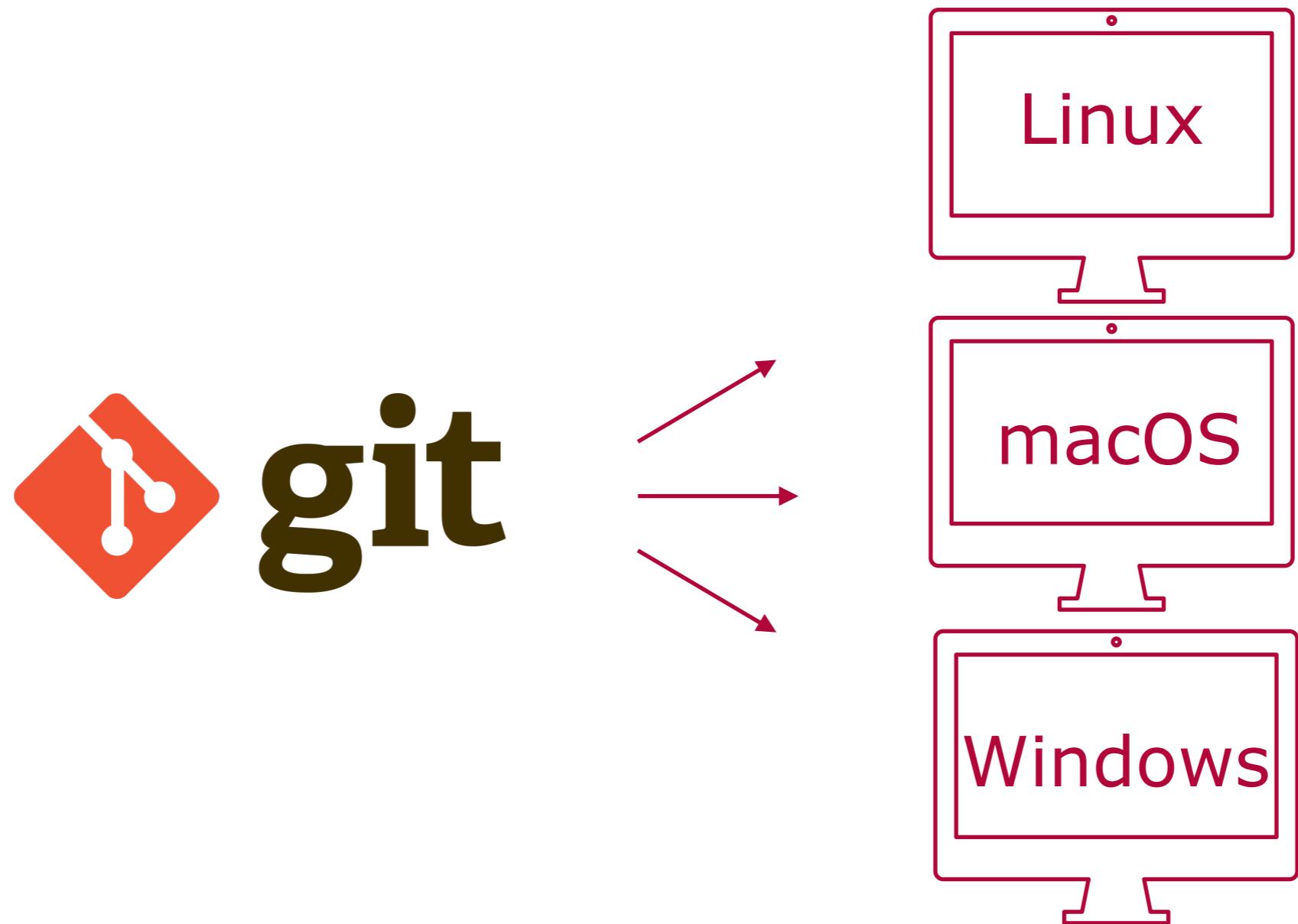
\$ git pull



Git installieren



Git installieren

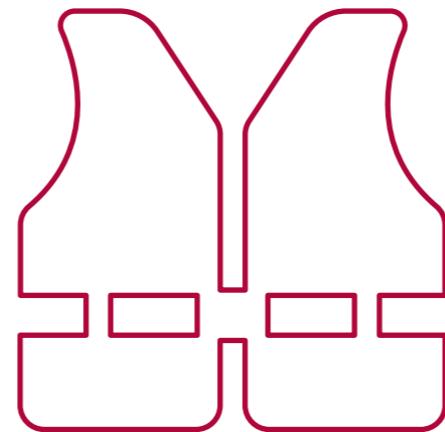


Git installieren

Paketmanager



Installationsassistent



Source Code
selbst kompilieren

01001110

Git für Linux installieren

Distribution Fedora

- Paketverwaltungstool: dnf
- \$ dnf install git

Distribution Ubuntu

- Paketverwaltungstool: apt
- \$ apt-get install git

Git für macOS installieren

Xcode Kommandozeilen-Tools

- `$ git --version`

Binärer Installationsassistent

- <https://git-scm.com/download/mac>
- Vorsicht: Download startet automatisch

Git für Windows installieren

Binärer Installationsassistent

- <https://git-scm.com/download/win>
- Vorsicht: Download startet automatisch

Git config

Git an die eigenen Bedürfnisse anpassen

- `~/.gitconfig` bzw. `~/.config/git/config`
- config Datei im aktuellen Repository

Einstellungen anzeigen

- `$ git config --list`

Nutzername und E-Mail angeben

- `$ git config --global user.name "Git.Nutzer"`
- `$ git config --global user.email "git.nutzer@examplemail.com"`

Git config

```
1 [user]
2     name = Pavan Kumar Sunkara
3     email = pavan.sss1991@gmail.com
4     username = pk sunkara
5 [core]
6     editor = vim
7     whitespace = fix,-indent-with-non-tab,trailing-space,cr-at-eol
8     excludesfile = ~/.gitignore
9 [sendemail]
10    smtpencryption = tls
11    smtpserver = smtp.gmail.com
12    smtpuser = pavan.sss1991@gmail.com
13    smtppass = password
14    smtpserverport = 587
15 [web]
16    browser = google-chrome
17 [instaweb]
18    httpd = apache2 -f
19 [rerere]
20    enabled = 1
21    autoupdate = 1
22 [push]
23    default = matching
24 [color]
25    ui = auto
26 [color "branch"]
27    current = yellow bold
28    local = green bold
29    remote = cyan bold
30 [color "diff"]
31    meta = yellow bold
32    frag = magenta bold
33    old = red bold
34    new = green bold
35    whitespace = red reverse
```

Git cheat sheet

Git verstehen

Snapshots, lokale, Integrität, fügt Daten hinzu

Modified ↔ Staged ↔ Committed

Git installieren

Paketmanager bzw. <https://git-scm.com/download>

Erstelle ein leeres Git-Repository / Reinitialisiere ein vorhandenes

\$ git init

Datei zum Index hinzufügen

\$ git add

Datei aus dem Arbeitsbaum und aus dem Index entfernen

\$ git rm

Änderungen am Repository aufzeichnen

\$ git commit

Verwalte einen Satz von verfolgten Repositories

\$ git remote

Klone ein Repository in ein neues Verzeichnis

\$ git clone

Aktualisiere die Remote-Repositories zusammen mit den zugehörigen Objekten

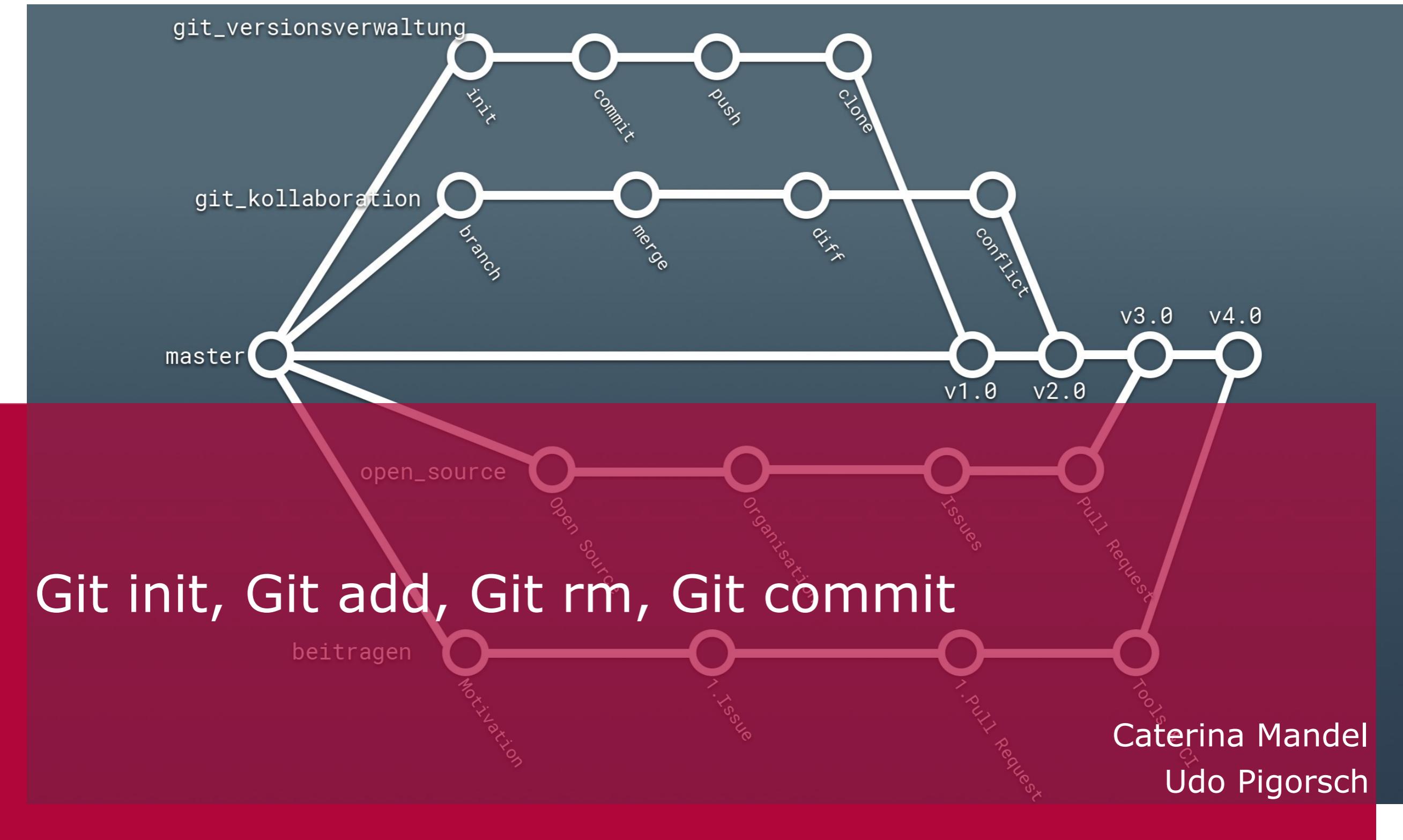
\$ git push

Objekte und Referenzen aus einem anderen Repository herunterladen

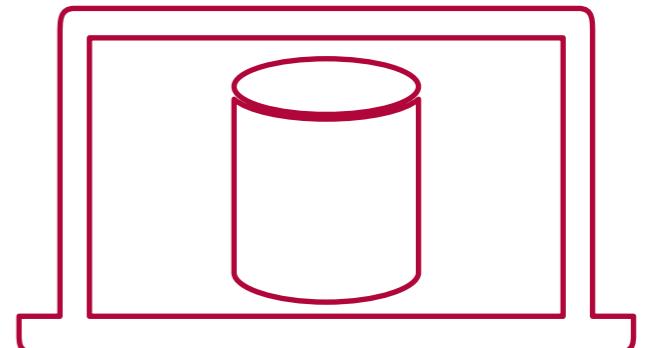
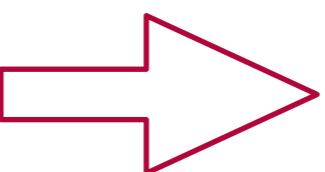
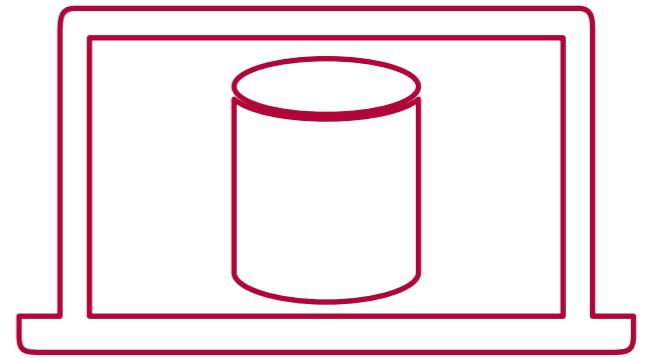
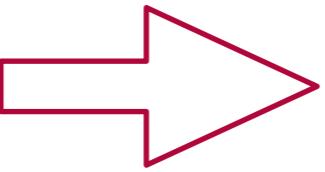
\$ git fetch

Aus einem anderen Repository oder einem lokalen Zweig holen und in diesen integrieren

\$ git pull



Git init



Git init

```
$ cd /Pfad/Projekt
```

```
$ git init
```

Git add

```
$ git add beispielbild.jpg
```

```
$ git add *.jpg
```

```
$ git commit
```

Git commit

```
$ git add beispielbild.jpg
```

```
$ git add *.jpg
```

a) **\$ git commit**

oder

b) **\$ git commit -m 'initial project version'**

Git add

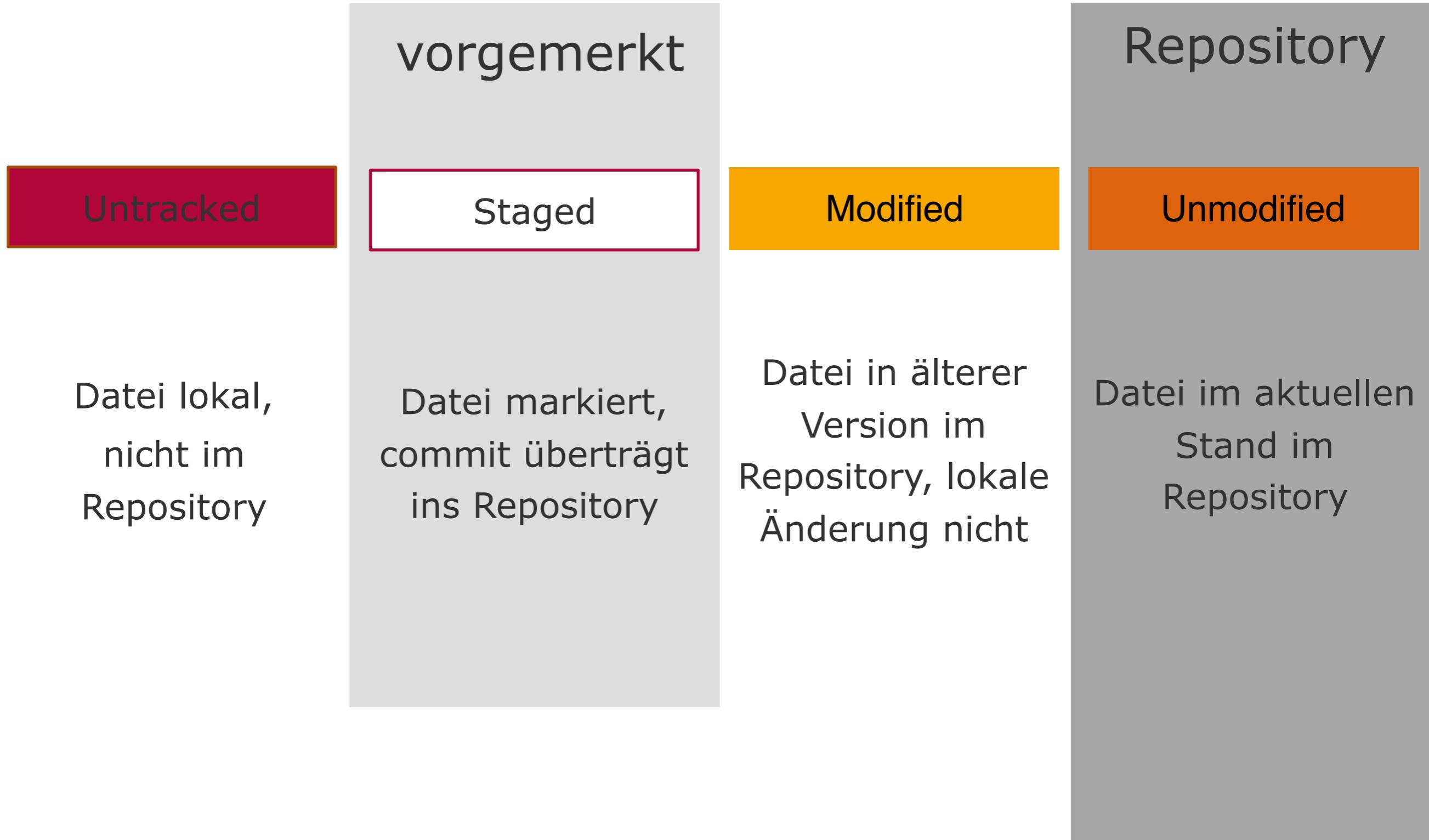
```
$ git add beispielbild.jpg
```

```
$ git add *.jpg
```

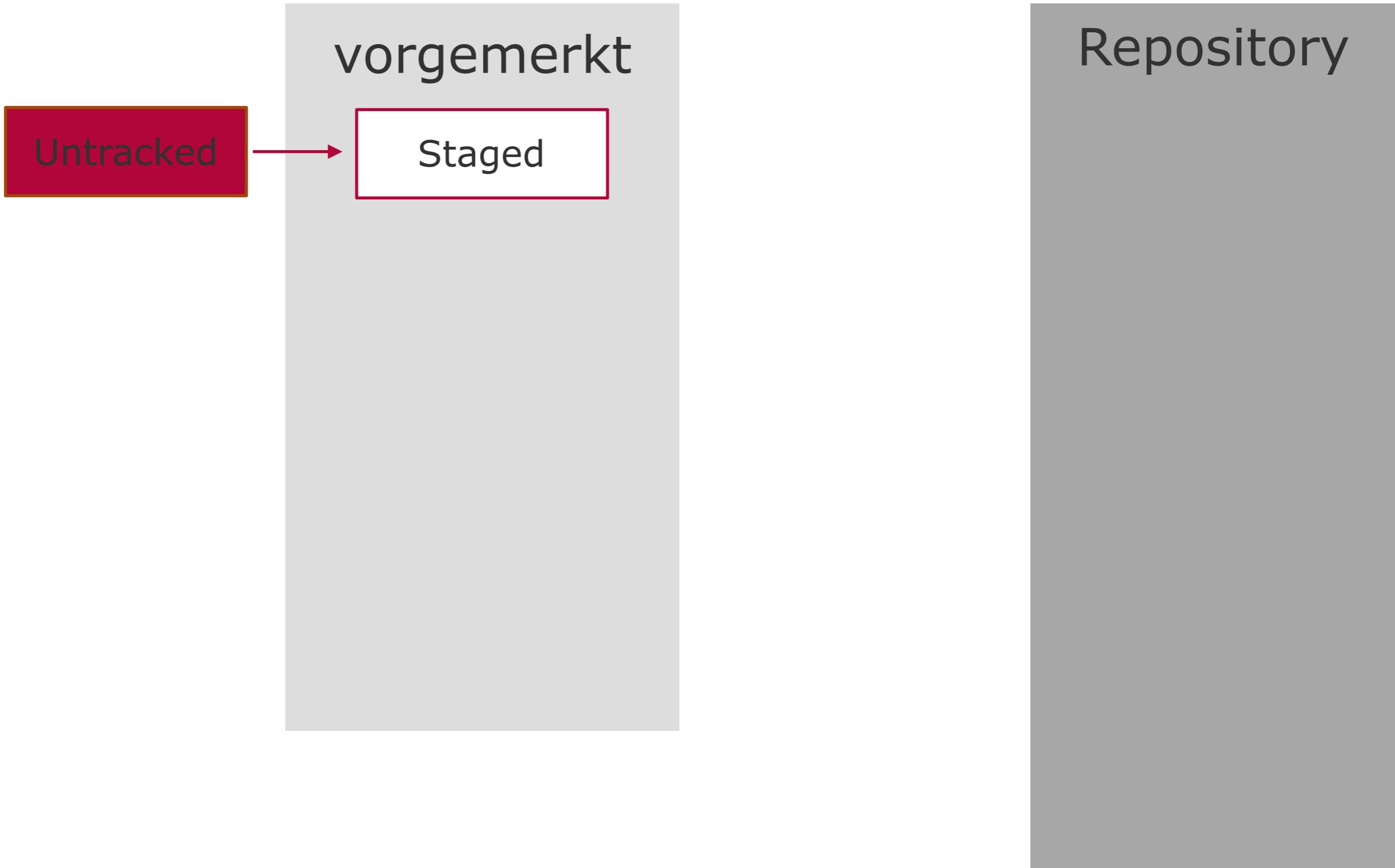
```
$ git commit
```

```
$ git remove beispielbild.jpg
```

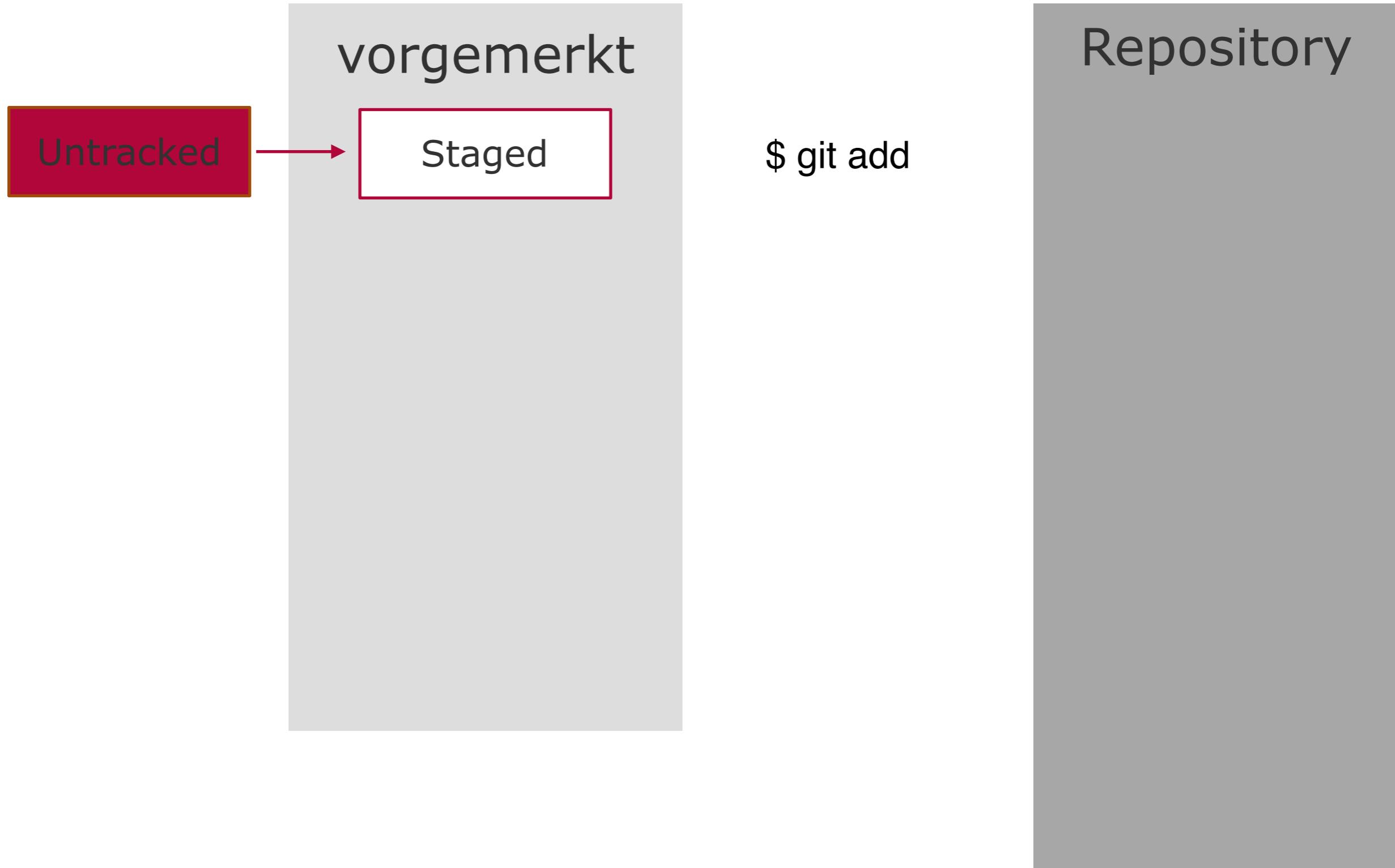
Git commit



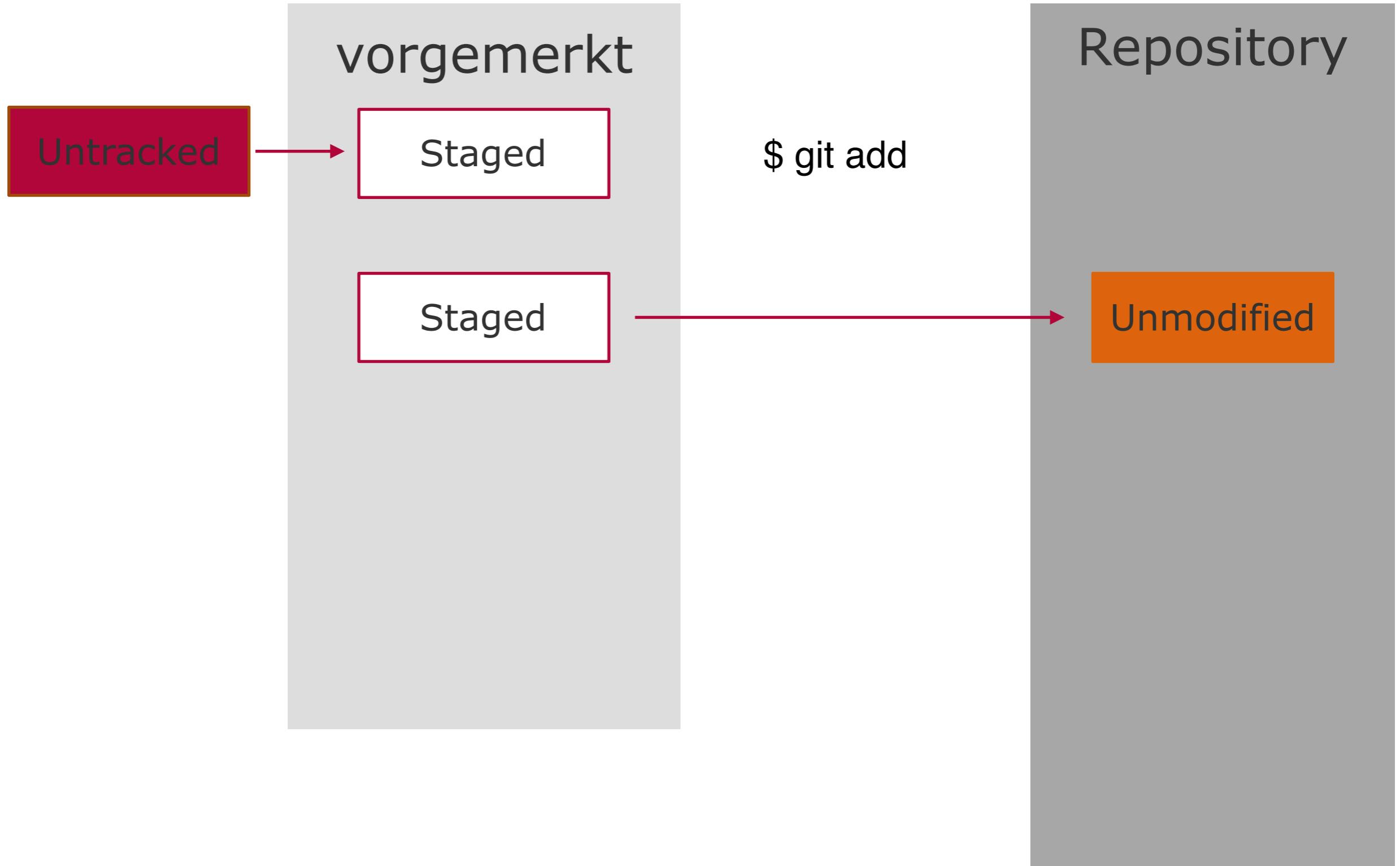
Git commit



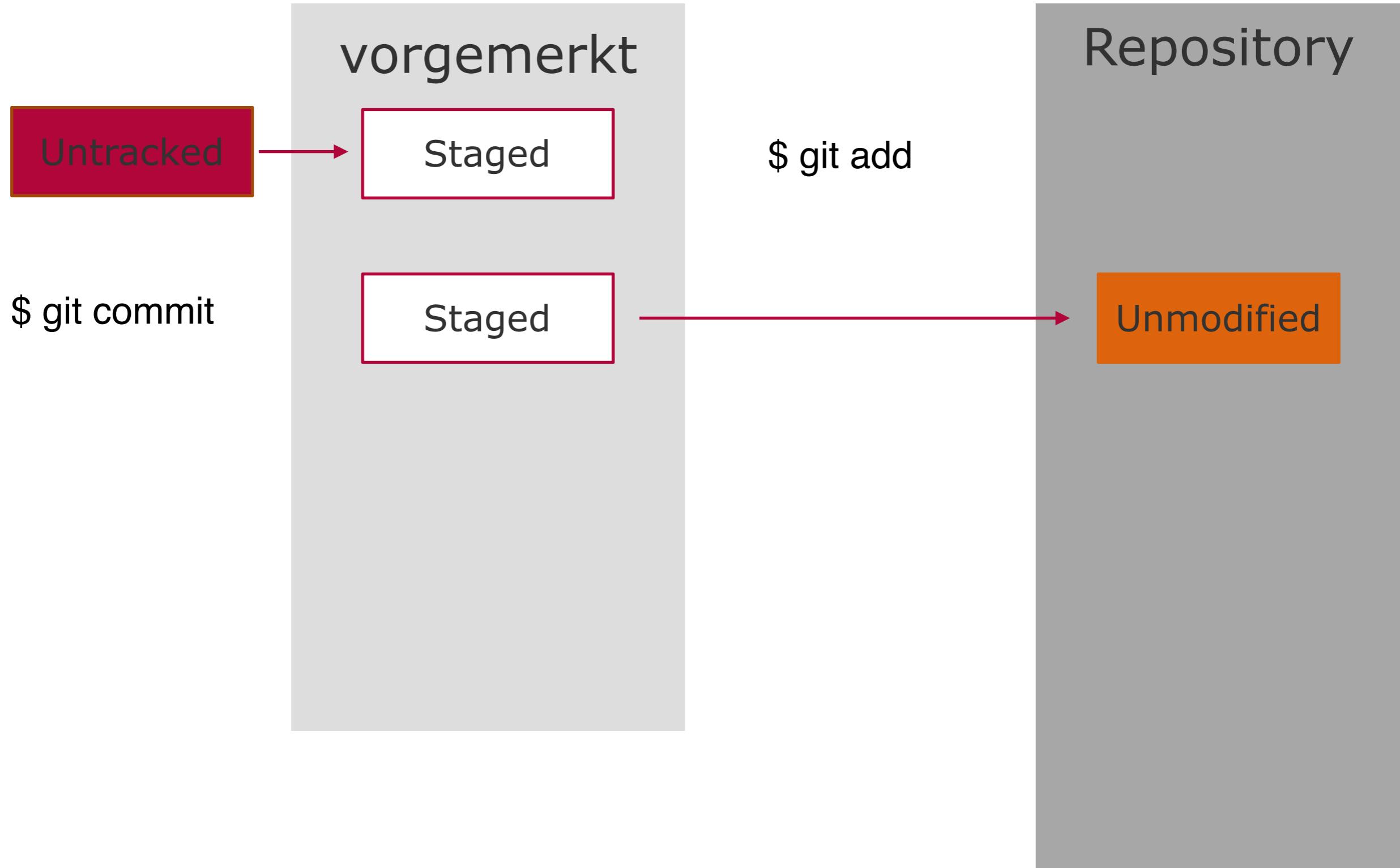
Git commit



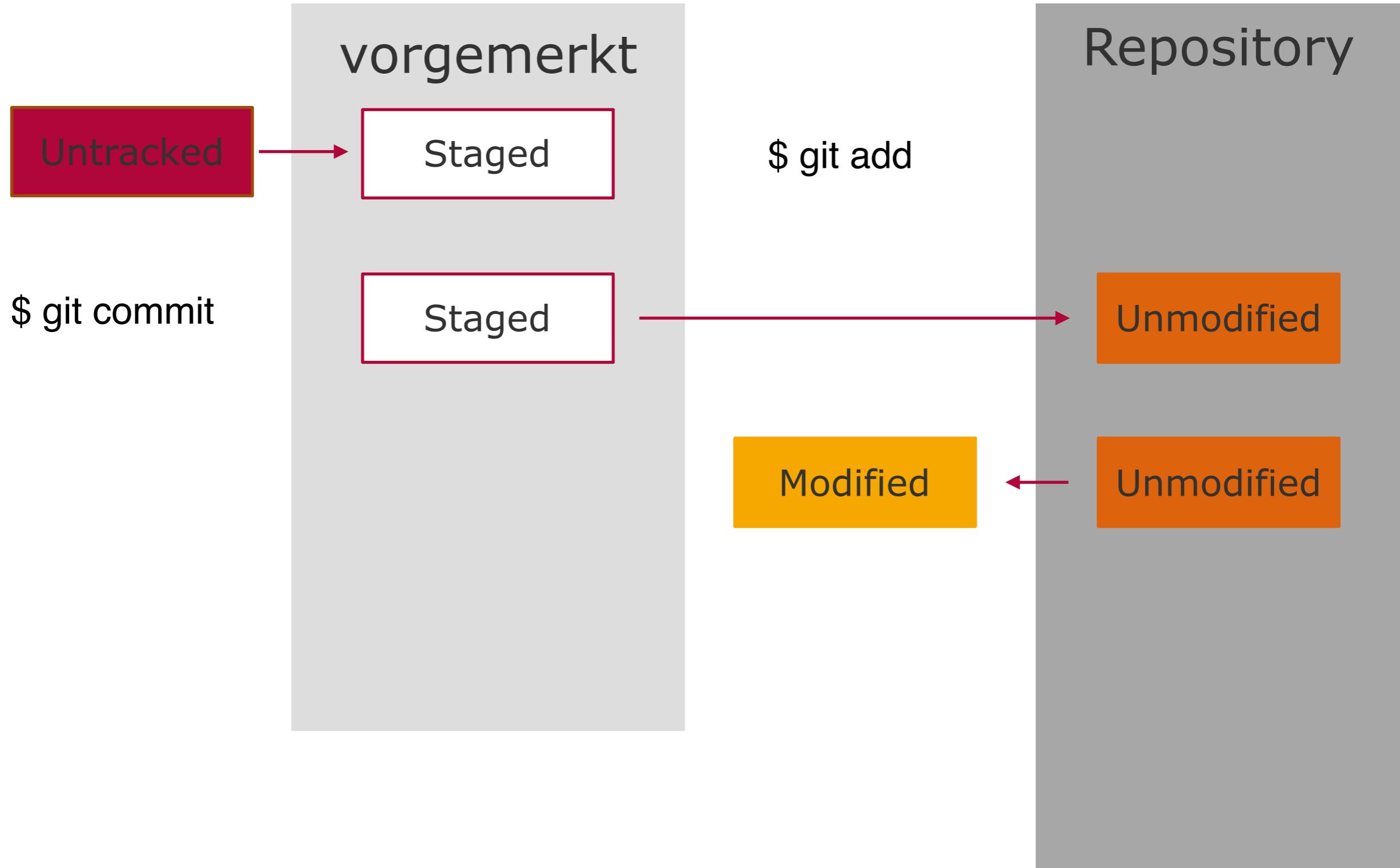
Git commit



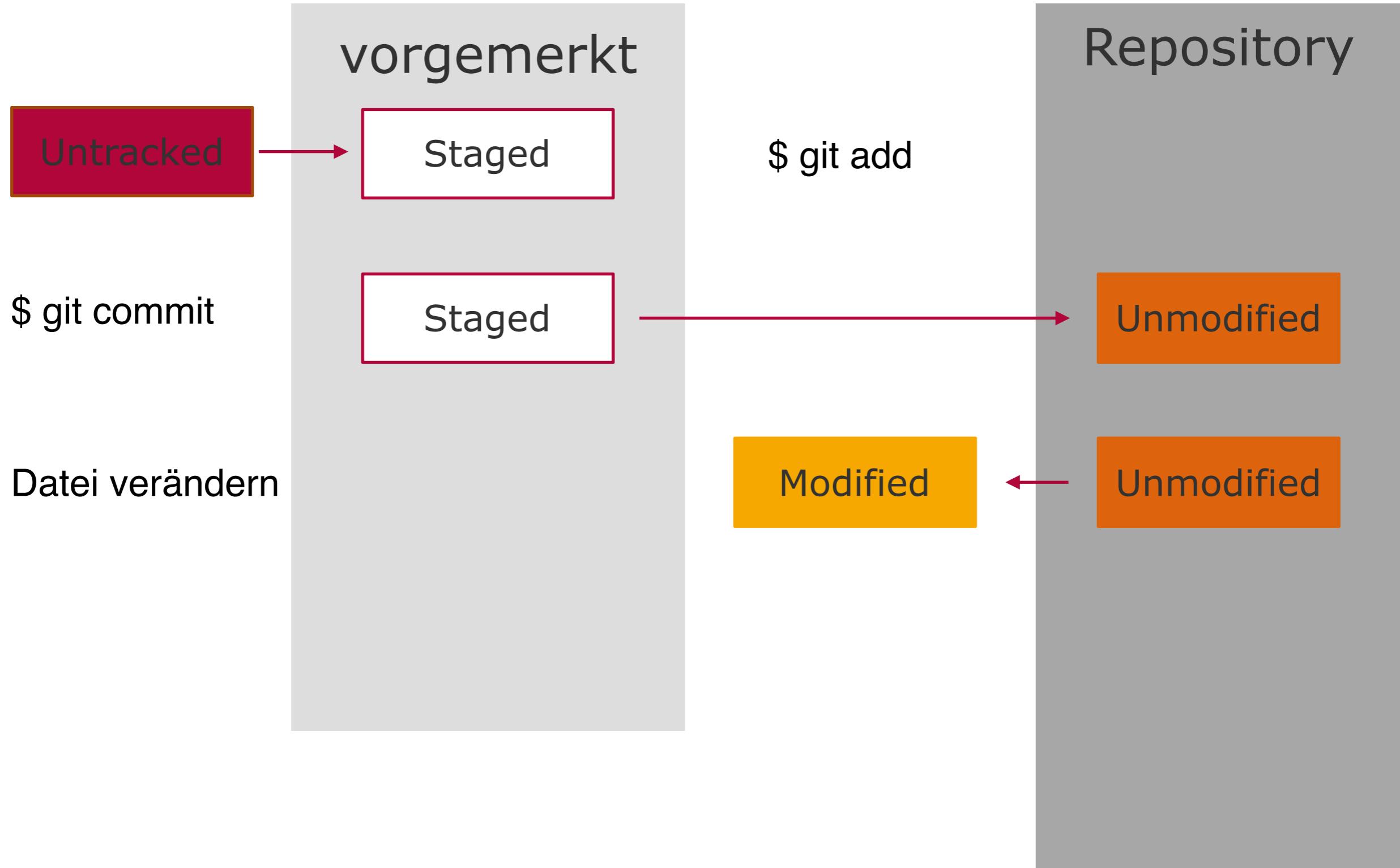
Git commit



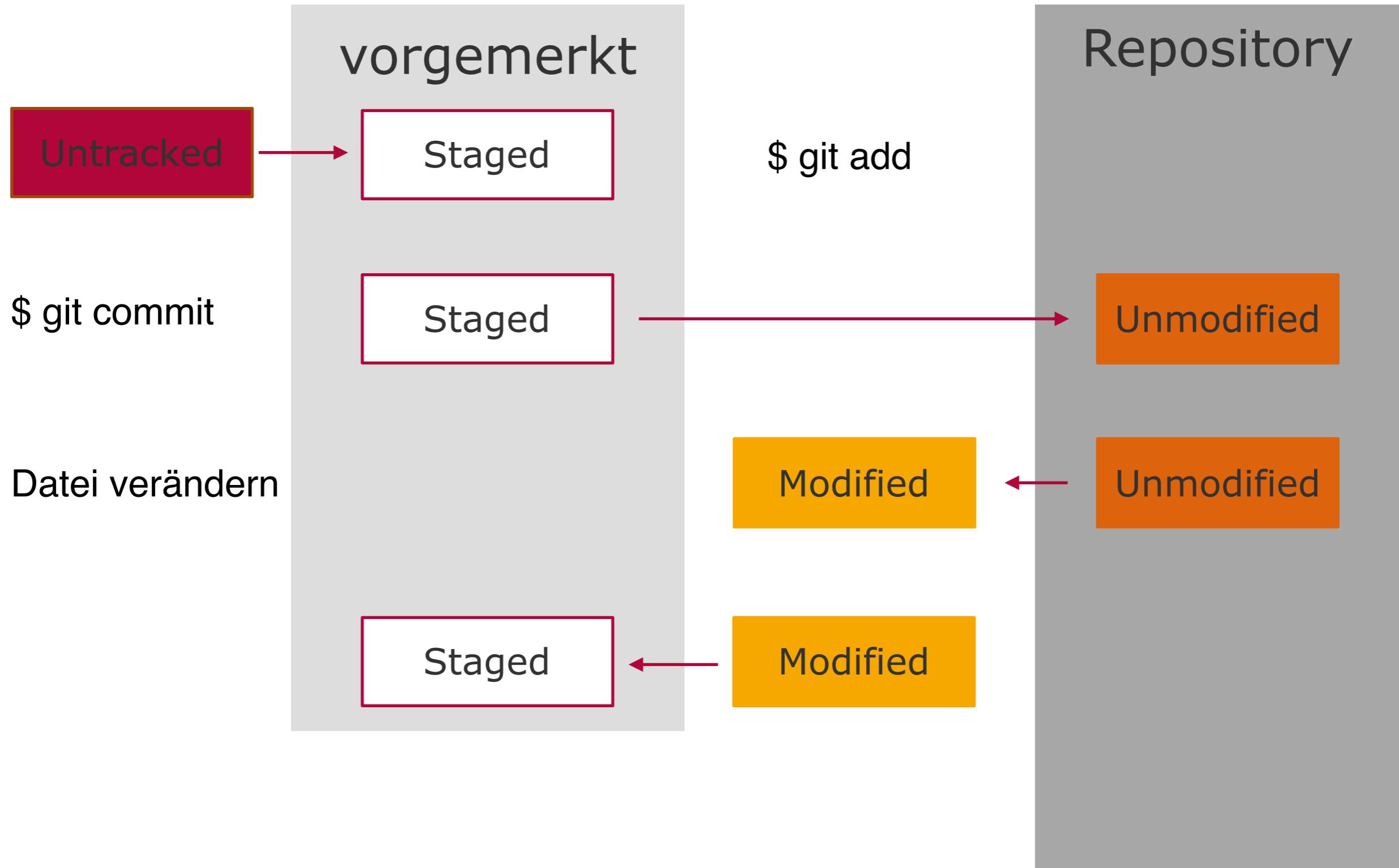
Git commit



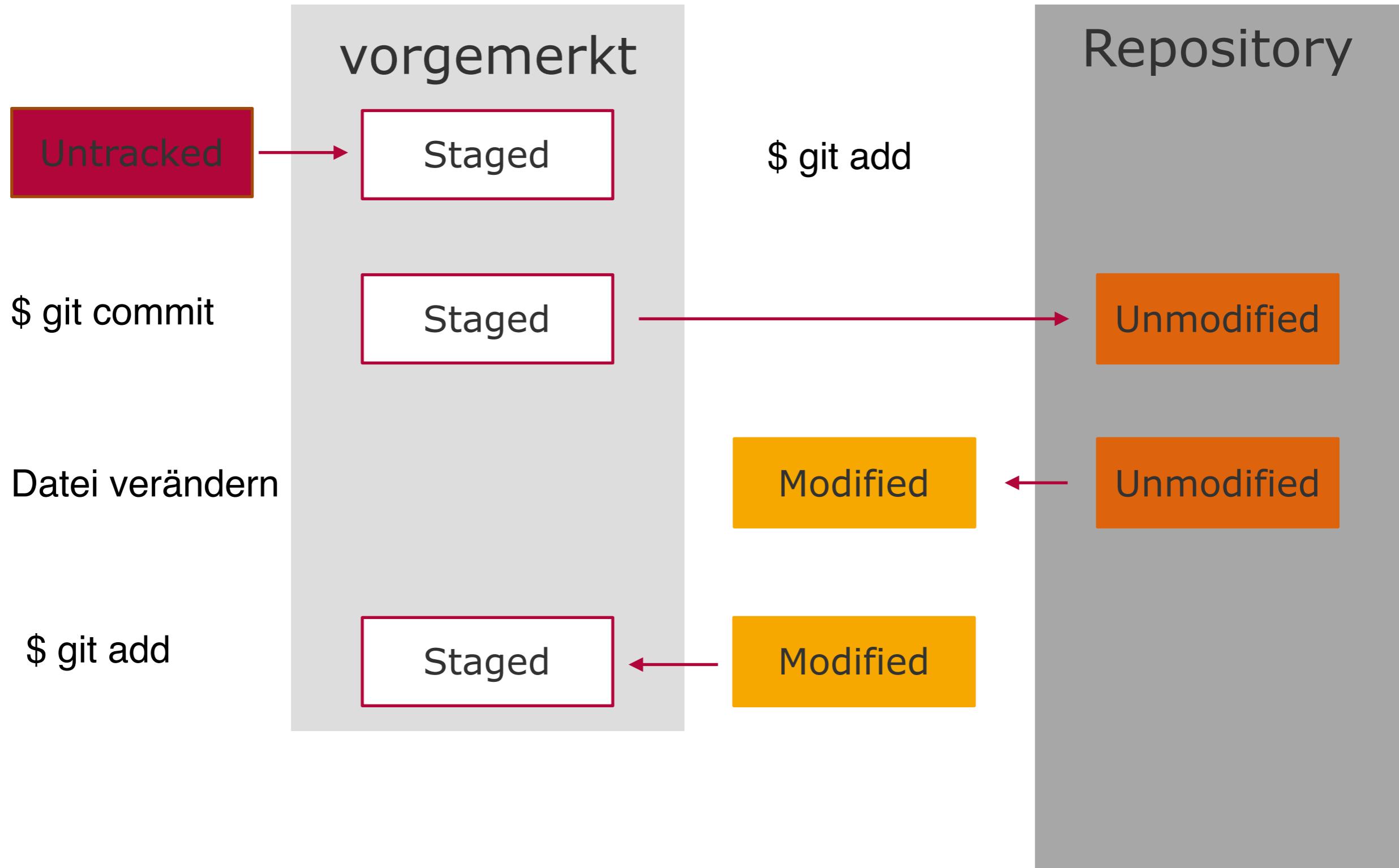
Git commit



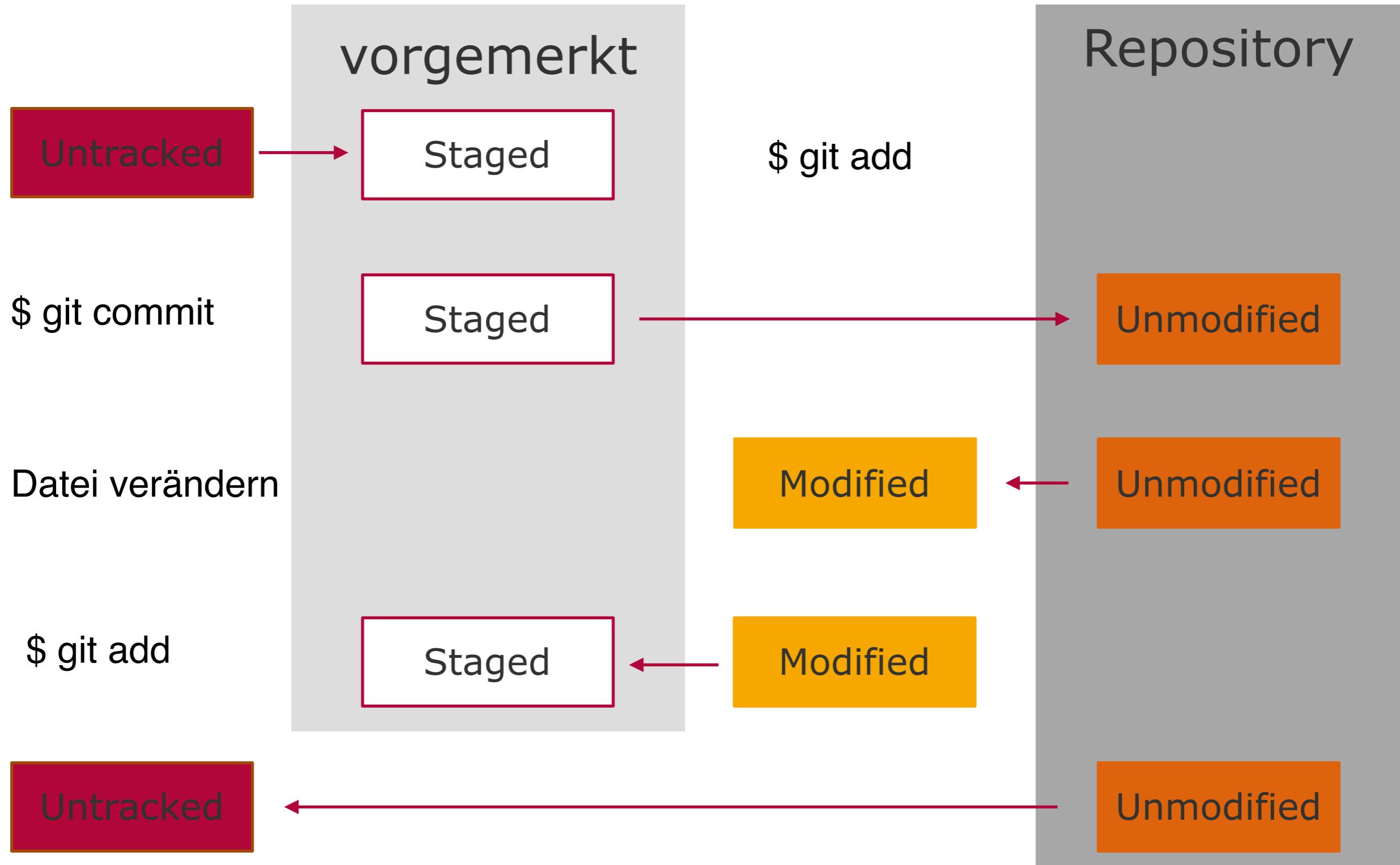
Git commit



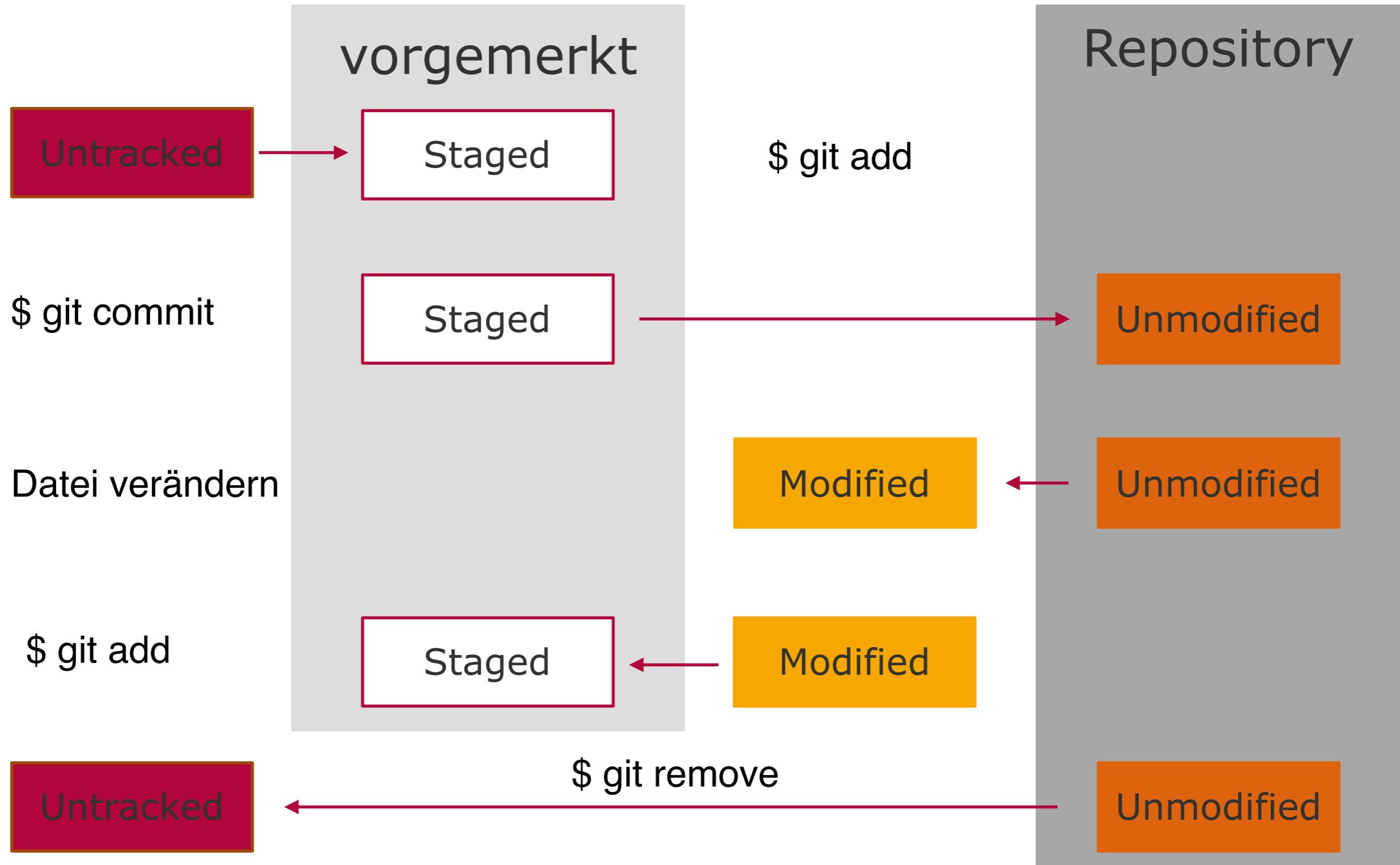
Git commit



Git commit



Git commit



Git cheat sheet

Git verstehen

Snapshots, lokale, Integrität, fügt Daten hinzu

Modified ↔ Staged ↔ Committed

Verwalte einen Satz von verfolgten Repositories

\$ git remote

Git installieren

Paketmanager bzw. <https://git-scm.com/download>

Erstelle ein leeres Git-Repository / Reinitialisiere ein vorhandenes

\$ git init

Klone ein Repository in ein neues Verzeichnis

\$ git clone

Datei zum Index hinzufügen

\$ git add

Aktualisiere die Remote-Repositories zusammen mit den zugehörigen Objekten

\$ git push

Datei aus dem Arbeitsbaum und aus dem Index entfernen

\$ git rm

Objekte und Referenzen aus einem anderen Repository herunterladen

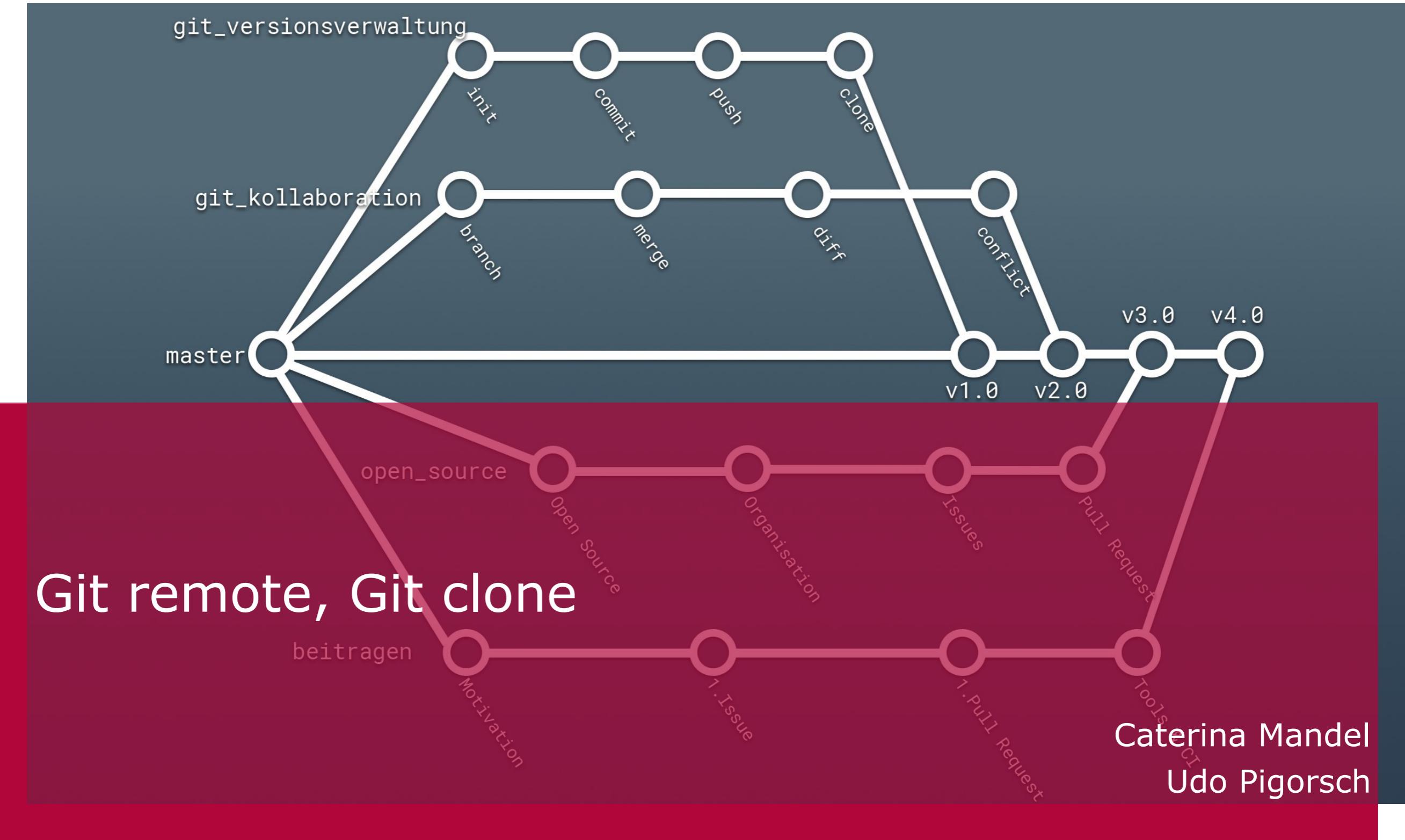
\$ git fetch

Änderungen am Repository aufzeichnen

\$ git commit

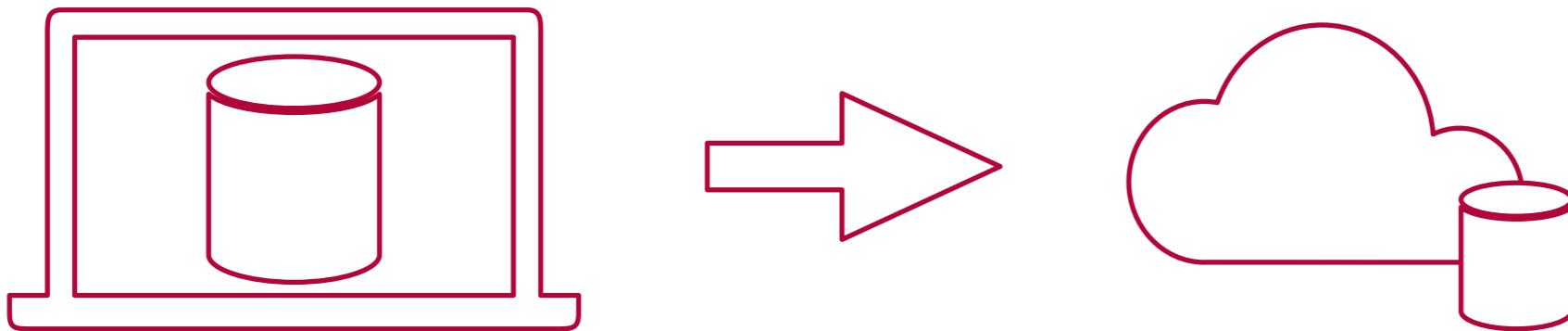
Aus einem anderen Repository oder einem lokalen Zweig holen und in diesen integrieren

\$ git pull



Git remote

```
$ git remote
```



Git remote

```
$ git remote add <alias><url>
```

```
$ git remote add repo https://github.com/repo
```

```
$ git remote rename <alias><neues alias>
```

```
$ git remote rename repo projekt1
```

```
$ git remote remove <alias>
```

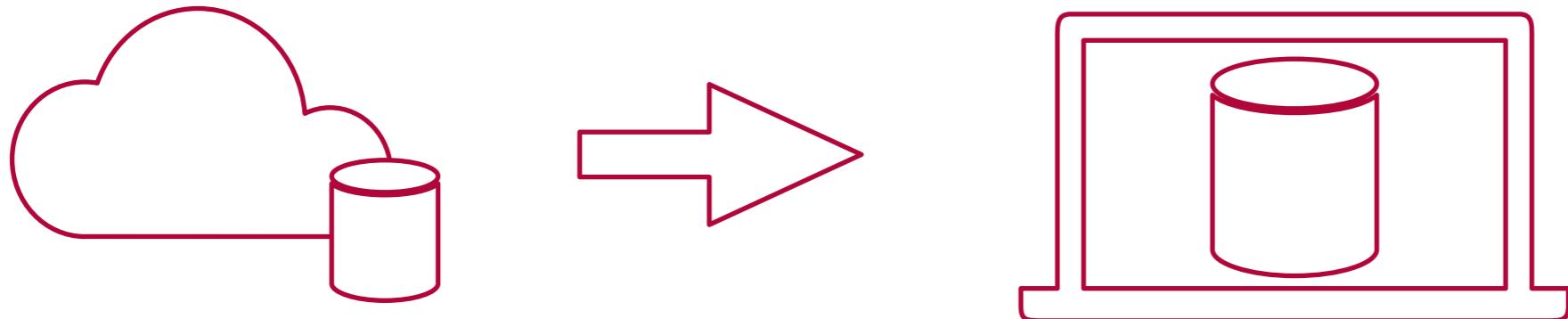
oder: \$ git remote rm <alias>

```
$ git remote rm projekt1
```

Git clone

```
$ git clone <url> <alias>
```

```
$ git clone https://github.com/repo repo
```



- Vollständige Kopie fast aller Daten
- Versionierung jeder Datei

Git cheat sheet

Git verstehen:

Snapshots, lokale, Integrität, fügt Daten hinzu

Modified ↔ Staged ↔ Committed

Git installieren:

Paketmanager bzw. <https://git-scm.com/download>

Erstelle ein leeres Git-Repository / Reinitialisiere ein vorhandenes

\$ git init

Datei zum Index hinzufügen

\$ git add

Datei aus dem Arbeitsbaum und aus dem Index entfernen

\$ git rm

Änderungen am Repository aufzeichnen

\$ git commit

Verwalte einen Satz von verfolgten Repositories

\$ git remote

Klone ein Repository in ein neues Verzeichnis

\$ git clone

Aktualisiere die Remote-Repositories zusammen mit den zugehörigen Objekten

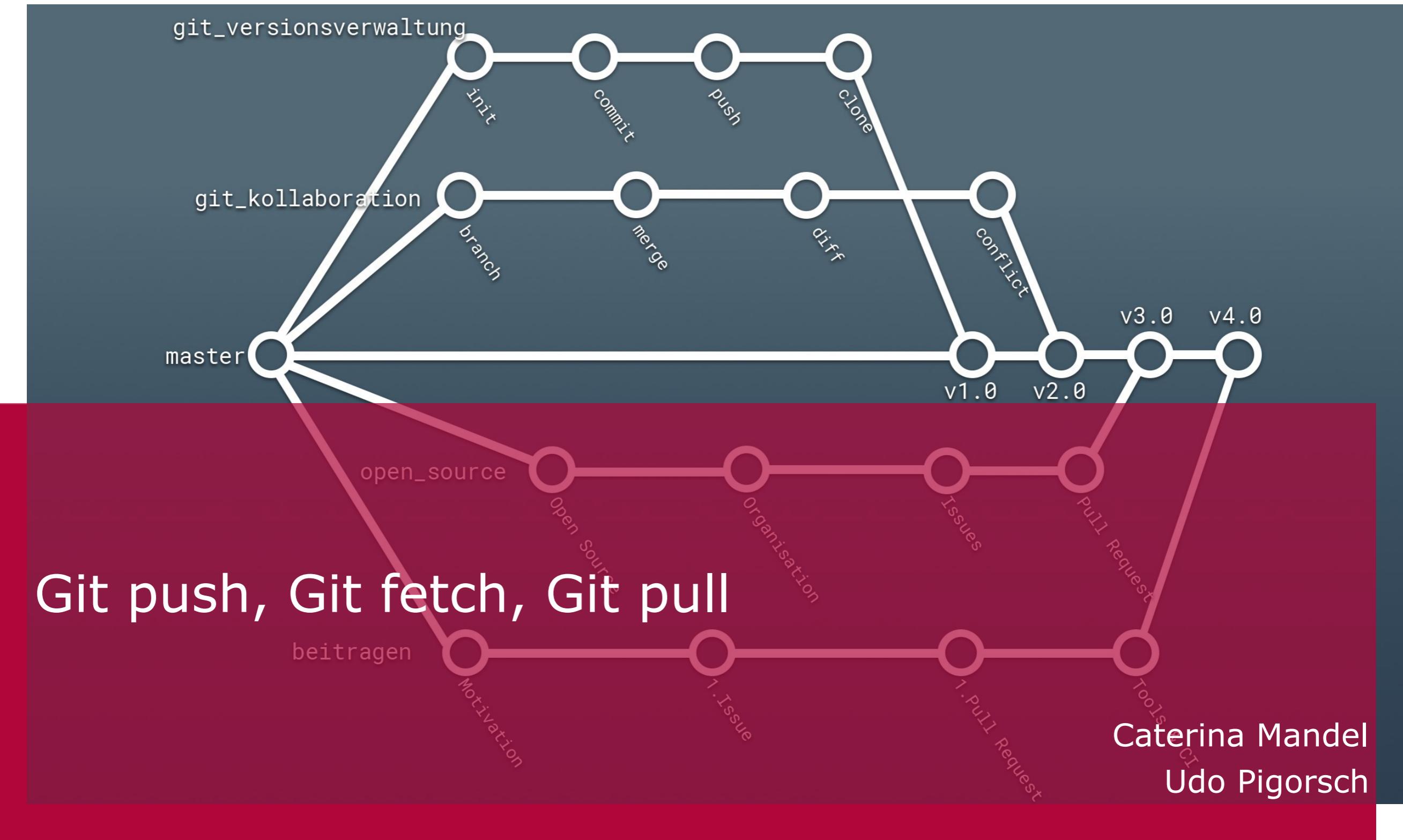
\$ git push

Objekte und Referenzen aus einem anderen Repository herunterladen

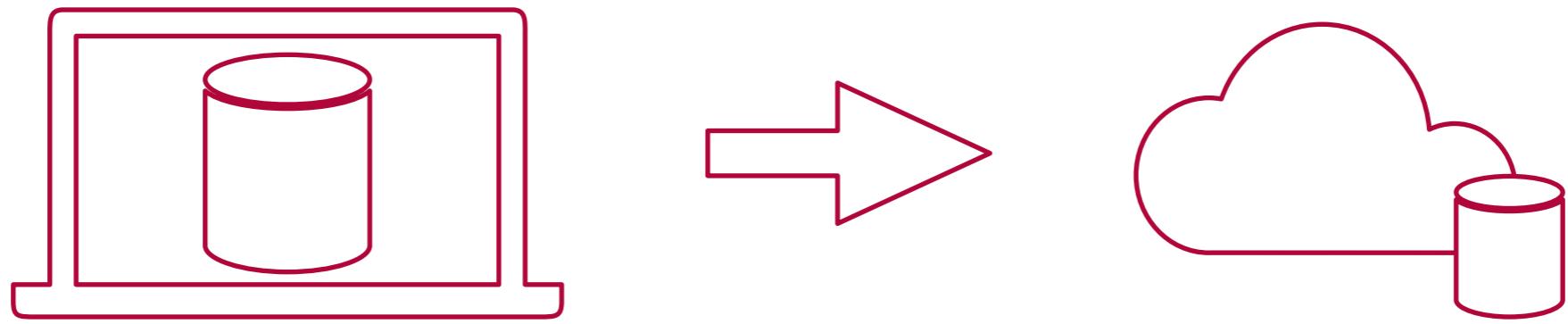
\$ git fetch

Aus einem anderen Repository oder einem lokalen Zweig holen und in diesen integrieren

\$ git pull



Git push

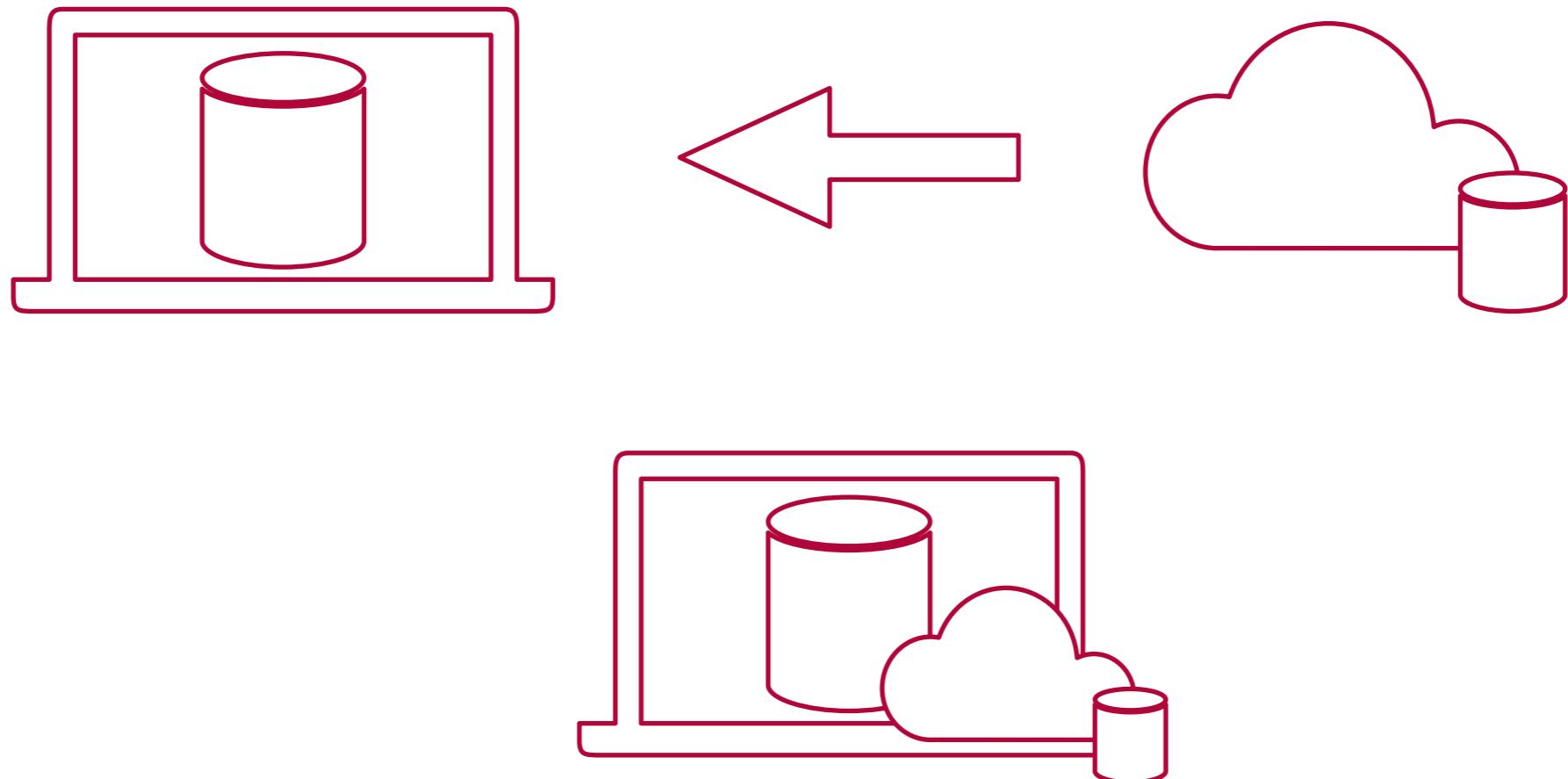


Git push

```
$ git push origin/<branch>
```

```
$ git push origin/master
```

Git fetch

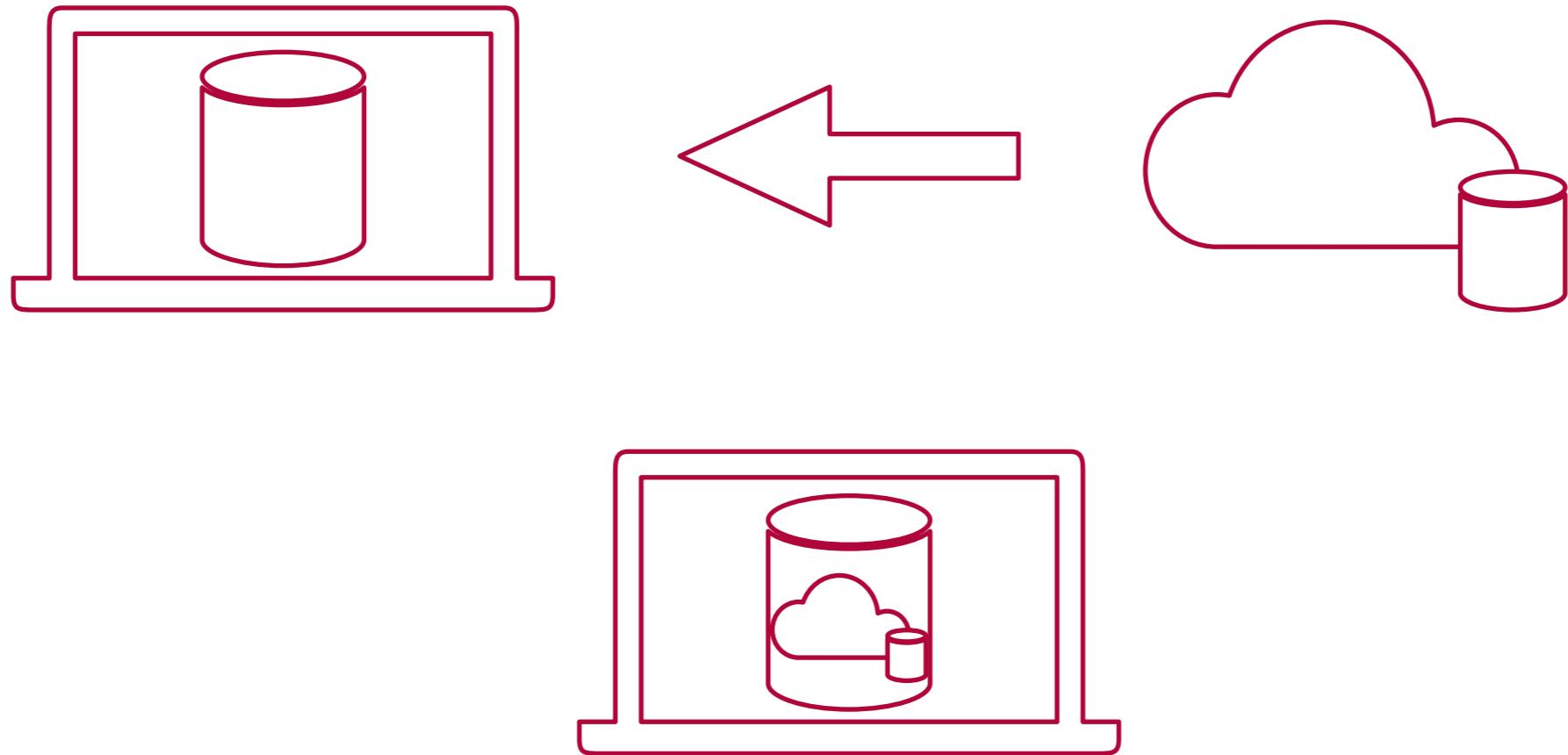


Git fetch

```
$ git fetch origin
```

```
$ git merge origin/<branch>
```

Git pull



Git pull

\$ git pull =

\$ git fetch +

\$ git merge

Git cheat sheet

Git verstehen:

Snapshots, lokale, Integrität, fügt Daten hinzu

Modified ↔ Staged ↔ Committed

Git installieren:

Paketmanager bzw. <https://git-scm.com/download>

Erstelle ein leeres Git-Repository /
Reinitialisiere ein vorhandenes

\$ git init

Datei zum Index hinzufügen

\$ git add

Datei aus dem Arbeitsbaum und aus dem Index entfernen

\$ git rm

Änderungen am Repository aufzeichnen

\$ git commit

Verwalte einen Satz von verfolgten Repositories

\$ git remote

Klonen ein Repository in ein neues Verzeichnis

\$ git clone

Aktualisiere die Remote-Repositories zusammen mit den zugehörigen Objekten

\$ git push

Objekte und Referenzen aus einem anderen Repository herunterladen

\$ git fetch

Aus einem anderen Repository oder einem lokalen Zweig holen und in diesen integrieren

\$ git pull