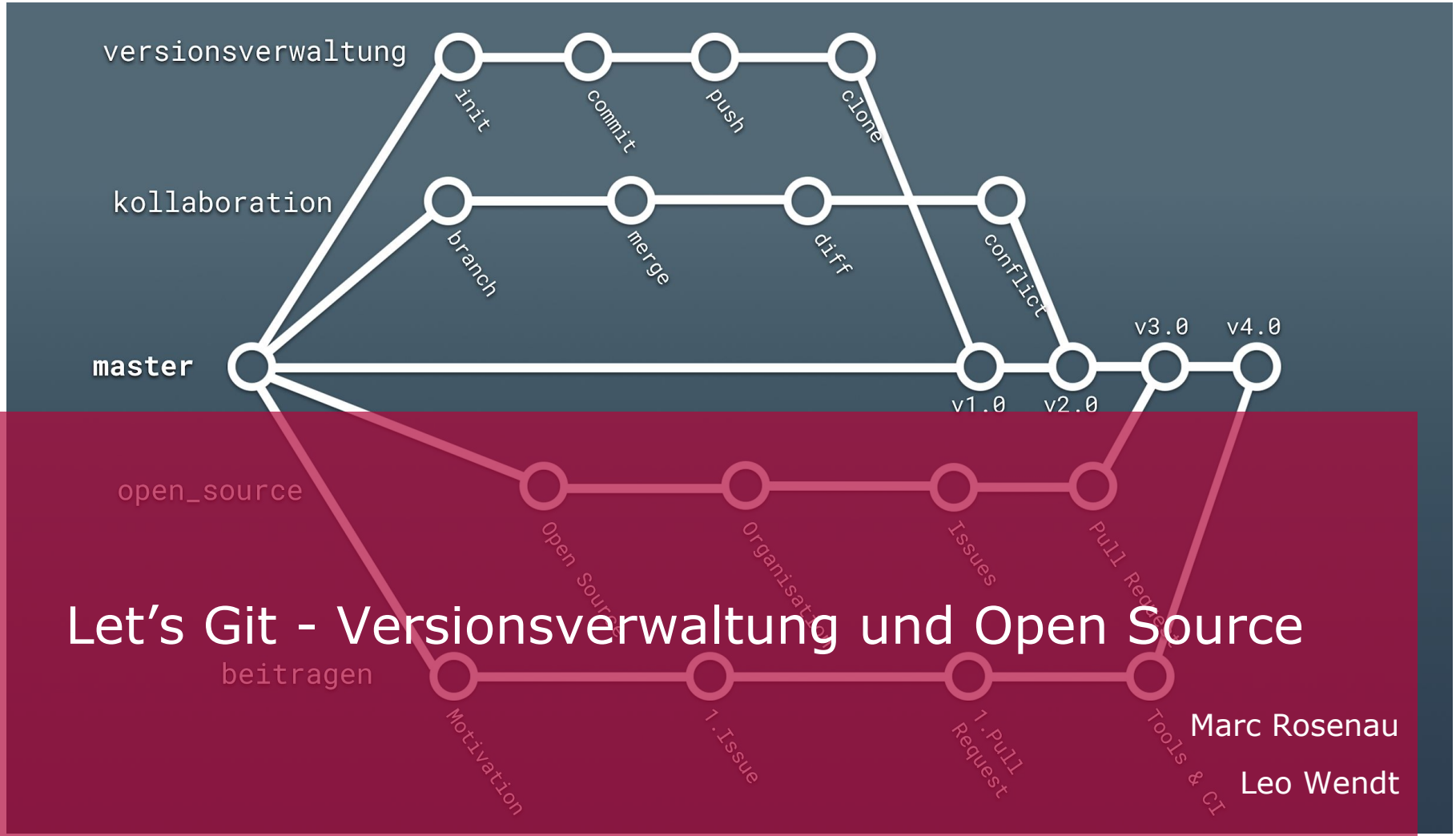
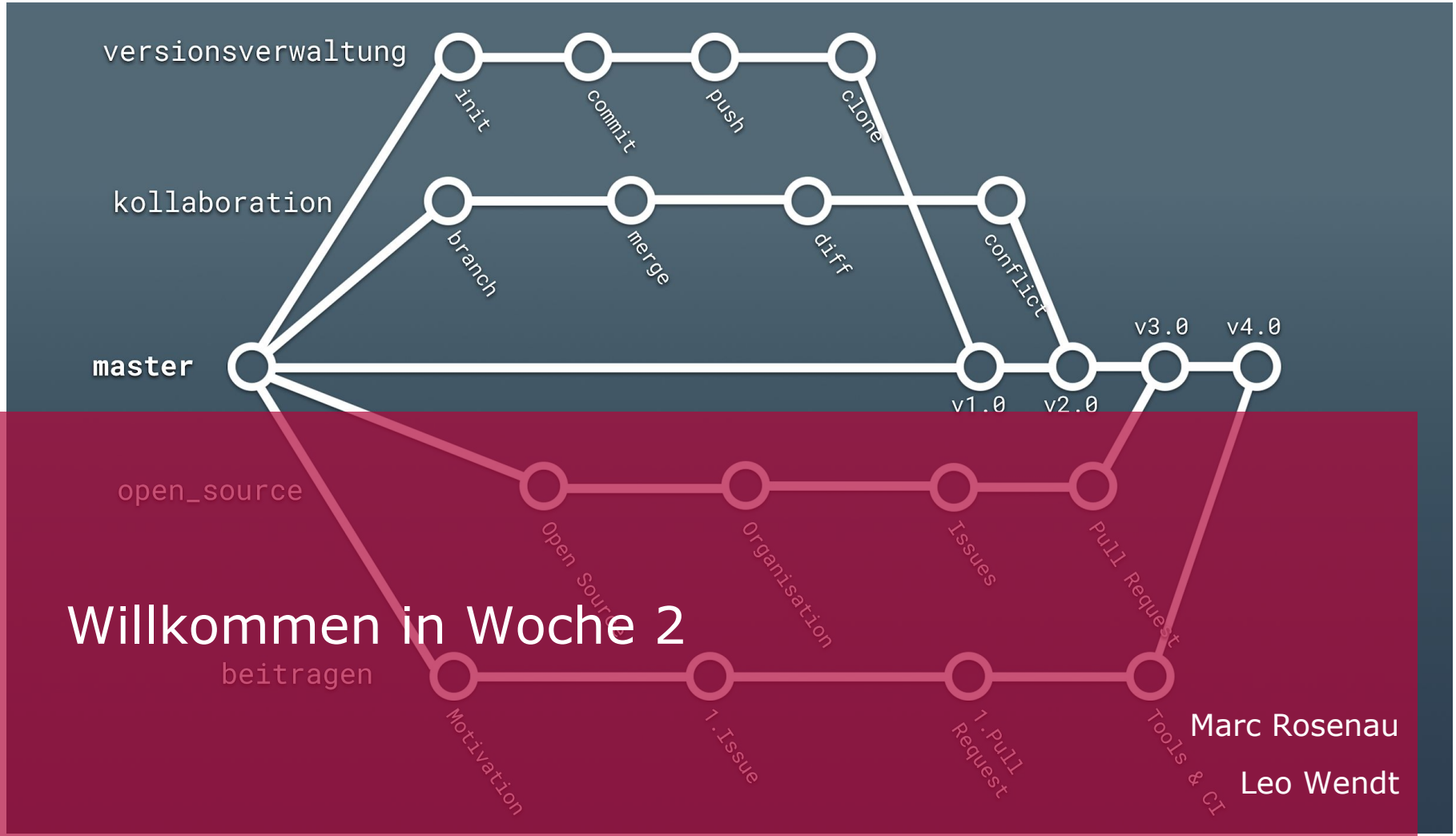


Let’s Git - Versionsverwaltung und Open Source

Marc Rosenau

Leo Wendt



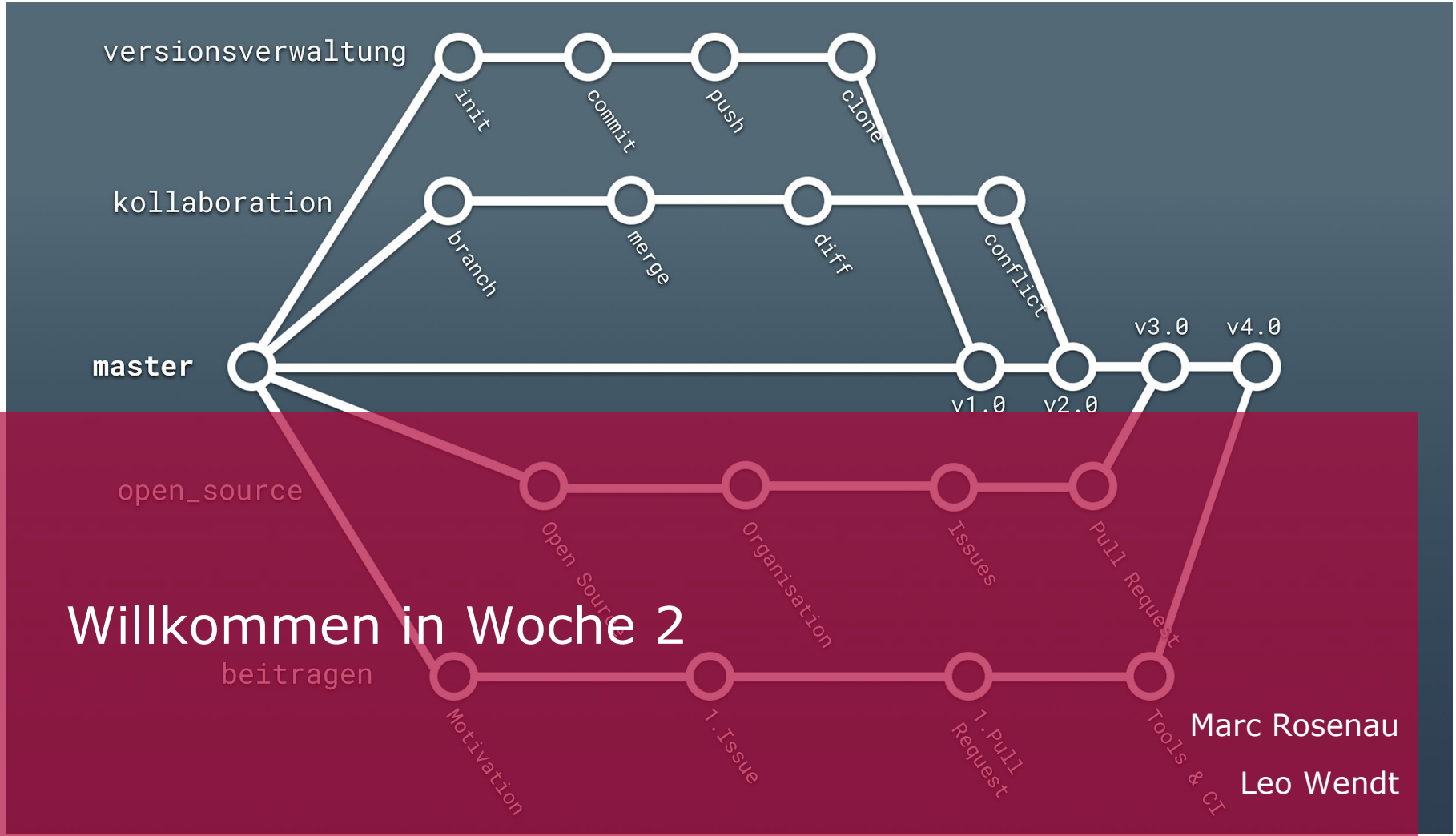
Willkommen in Woche 2

Marc Rosenau

Leo Wendt

Was passiert?

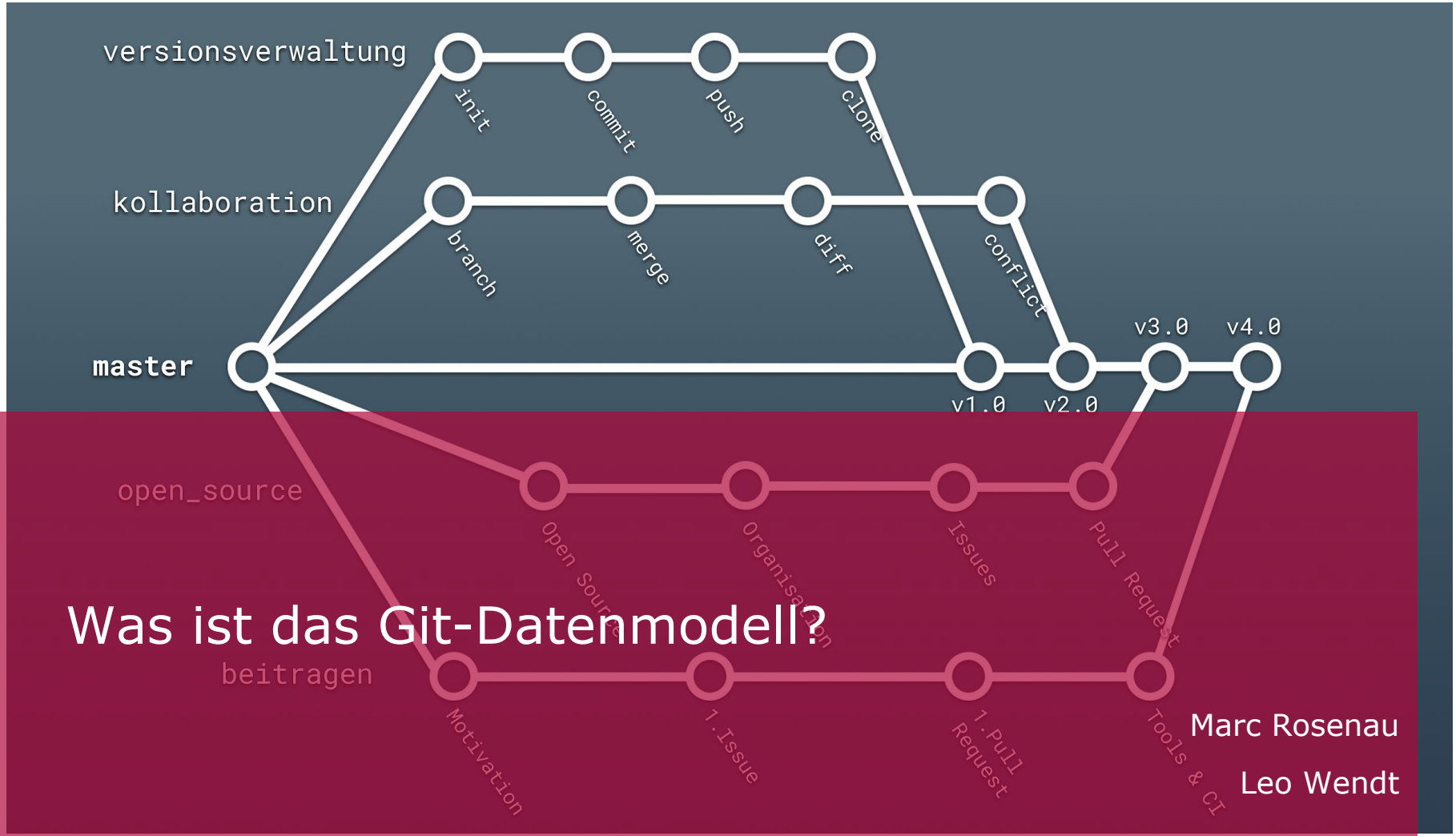
- **Was ist das Git Datenmodell?**
- **Was ist ein Branch?**
- **Was ist ein Merge?**
- **Wie behebe ich einen Merge Konflikt?**
- **Wie sehe ich die Geschichte des Repository?**
- **Wie korrigiere ich mein Repository?**
- **Wie kehre ich zu einer Version zurück?**



Willkommen in Woche 2

Marc Rosenau

Leo Wendt

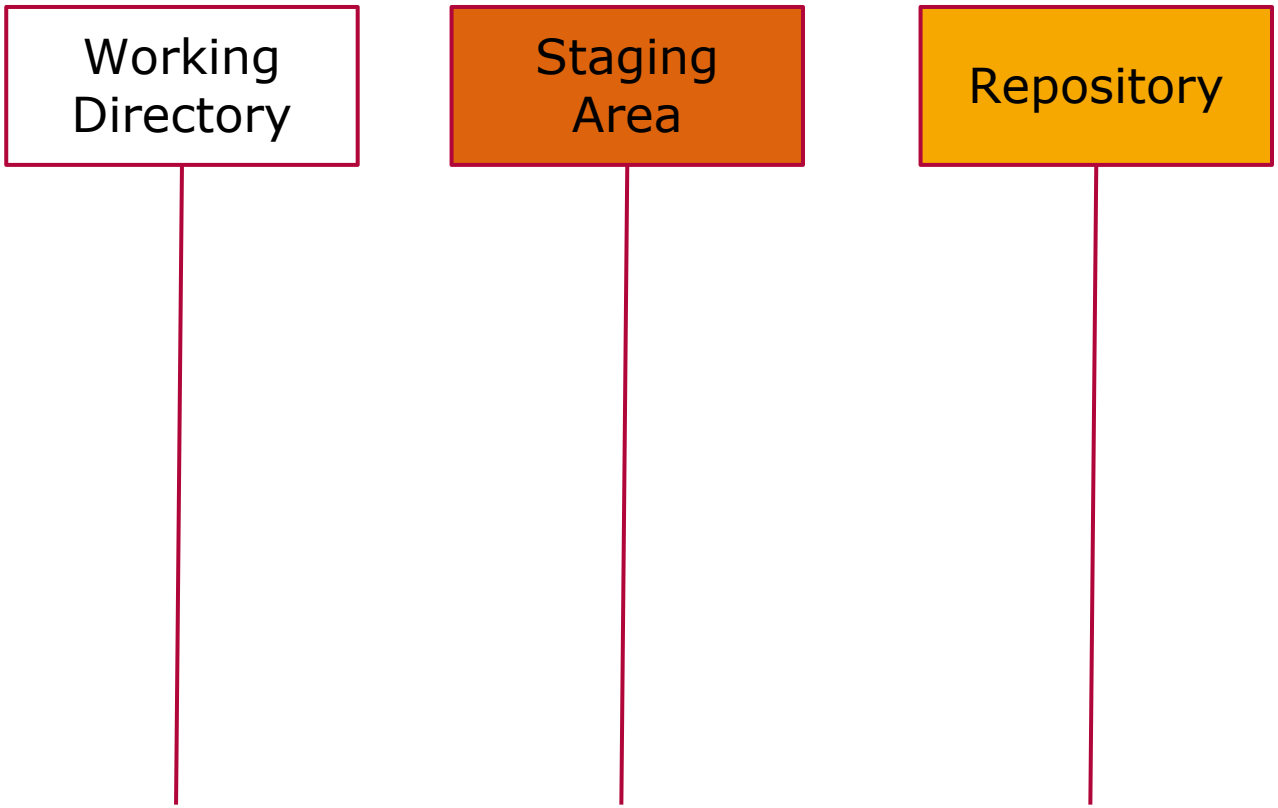


Was ist das Git-Datenmodell?

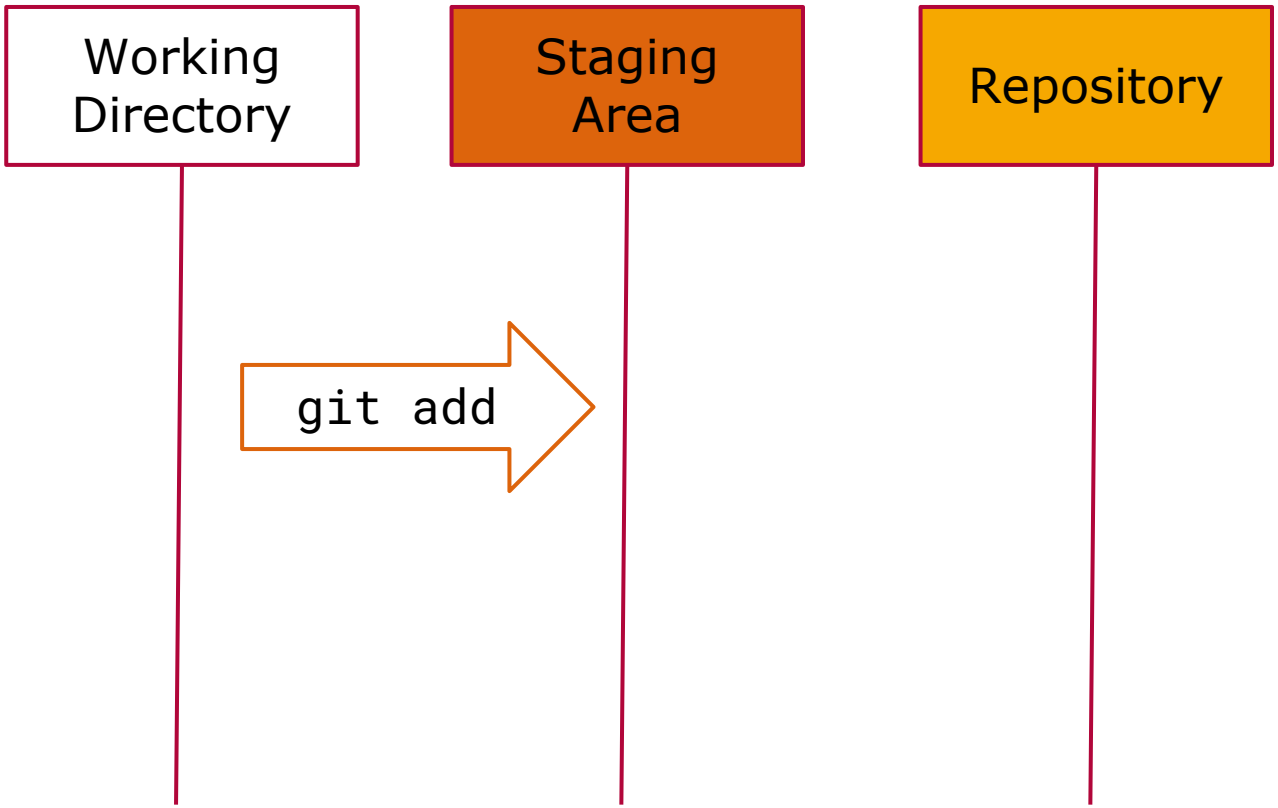
Marc Rosenau

Leo Wendt

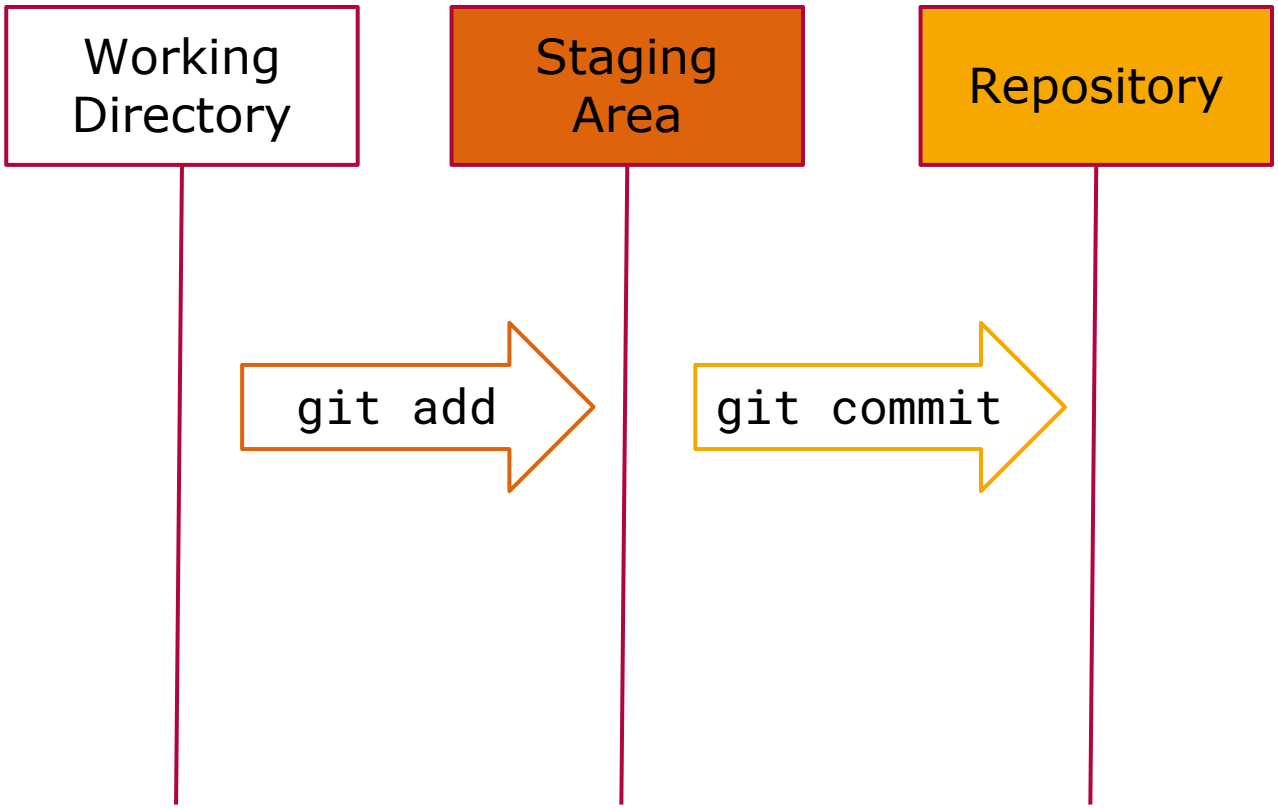
Recap



Recap



Recap

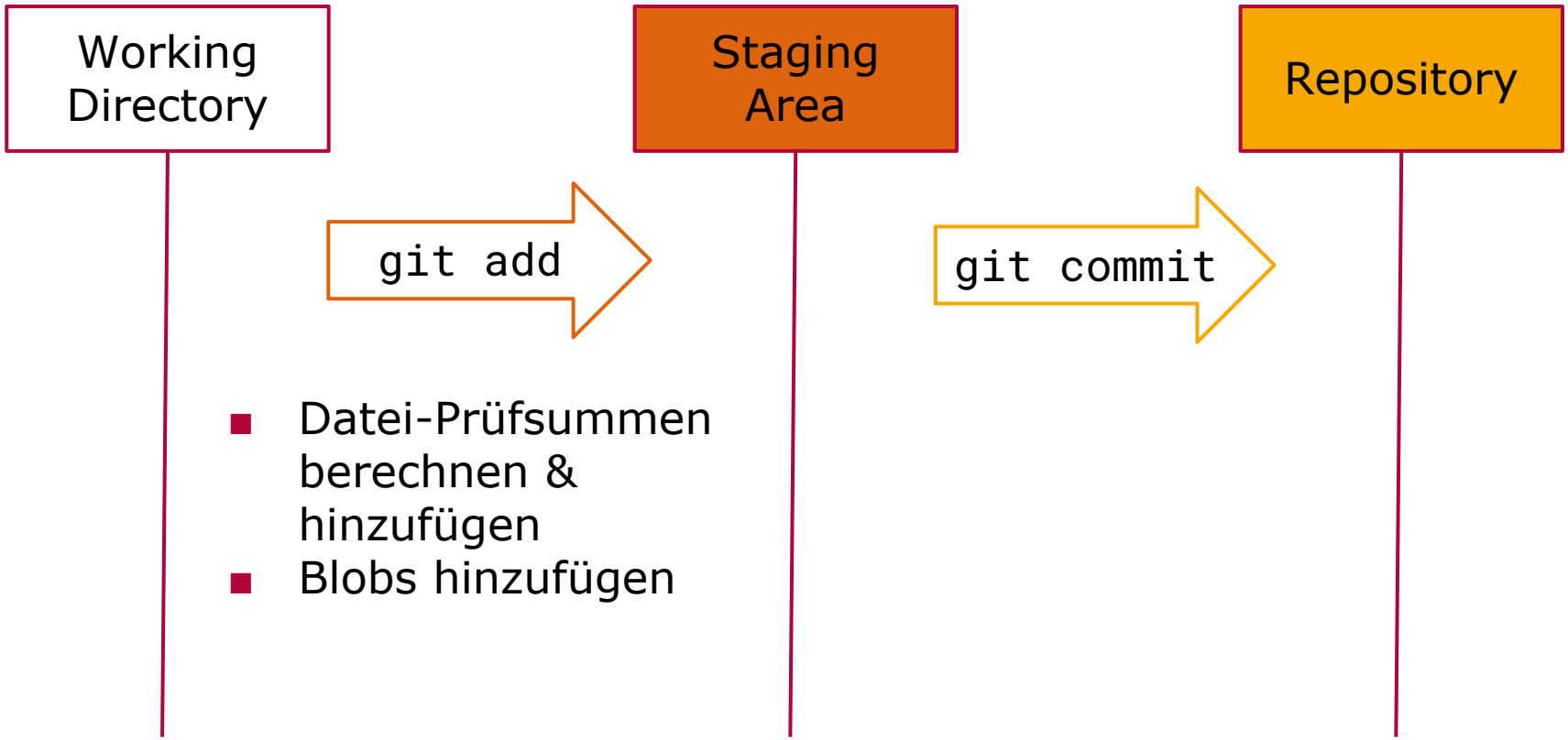

Working Directory
Staging Area
Repository
git add
git commit

Git Baum und Datenmodell

- Stagen von Dateien
  - **speichert** die Version der Datei im Git Repository (“blobs”)
  - **berechnet** Prüfsumme für jede Datei
  - fügt Prüfsummen zur Staging Area hinzu

```
$ git add liste1.txt liste2.txt
```

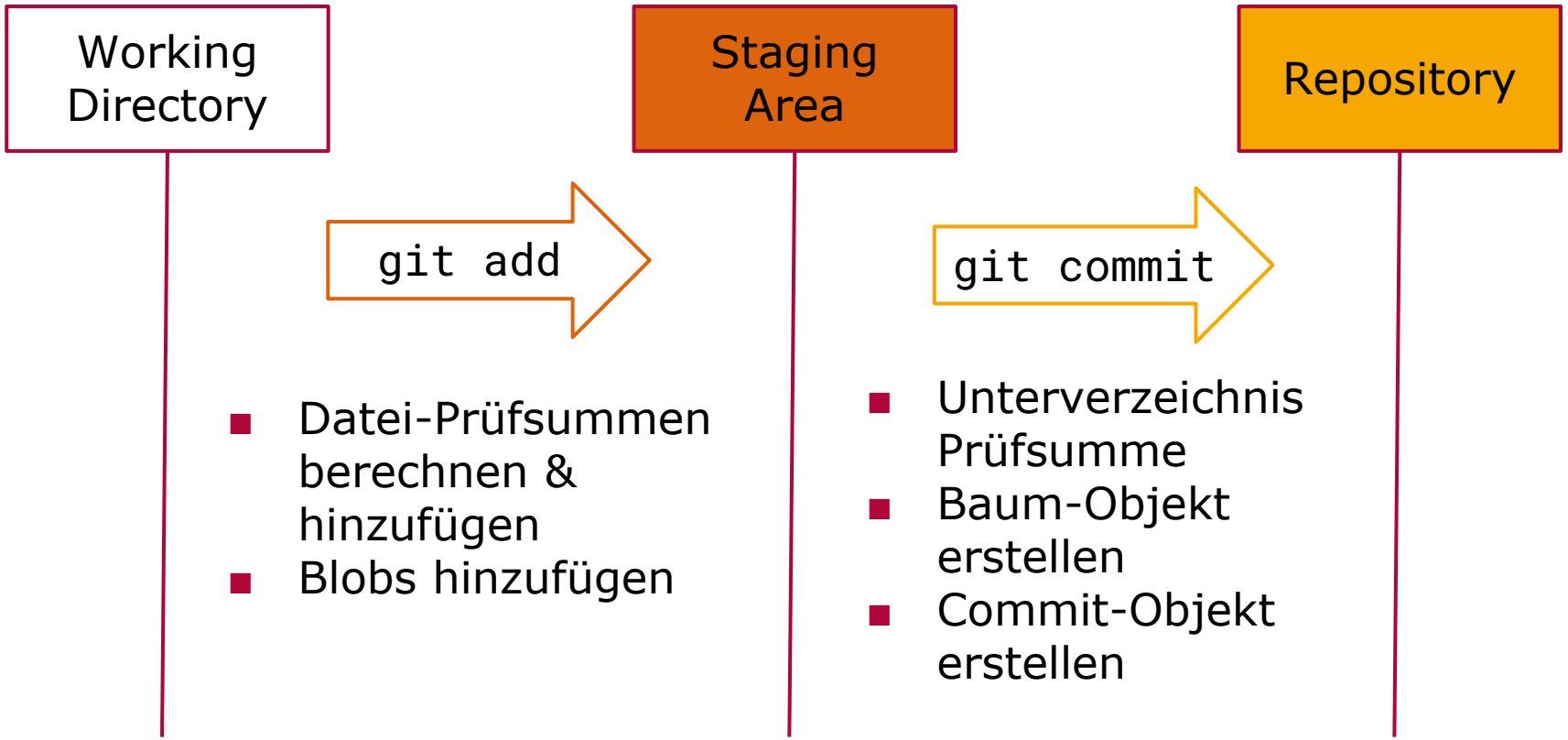
Recap


Working Directory
Staging Area
Repository
git add
git commit
Datei-Prüfsummen berechnen & hinzufügen
Blobs hinzufügen

Git Baum und Datenmodell

- Committen von Dateien
  - Git erstellt Prüfsummen für Unterverzeichnisse und speichert sie als **Baum-Objekt** mit Zeigern auf gespeicherte Dateiversion
  - Git erstellt ein **Commit-Objekt** mit Metadaten und einen Zeiger auf das **Baum-Objekt**
    - Metadaten sind u.a. Namen und Email-Adresse, Commit-Nachricht und Zeiger zu den Commits, die direkt davor kamen

Recap


Working Directory
Staging Area
Repository
git add
git commit
Datei-Prüfsummen berechnen & hinzufügen
Blobs hinzufügen
Unterverzeichnis Prüfsumme
Baum-Objekt erstellen
Commit-Objekt erstellen

Beispiel

Liste1.txt

```
# Liste 1

Hier sammeln wir
Geschenke
```

Liste2.txt

```
# Liste 2

Hier sammeln wir
Rezepte
```

Beispiel

```
blob size
# Liste 1

Hier sammeln wir
Geschenke
```

```
blob size
# Liste 2

Hier sammeln wir
Rezepte
```

Beispiel

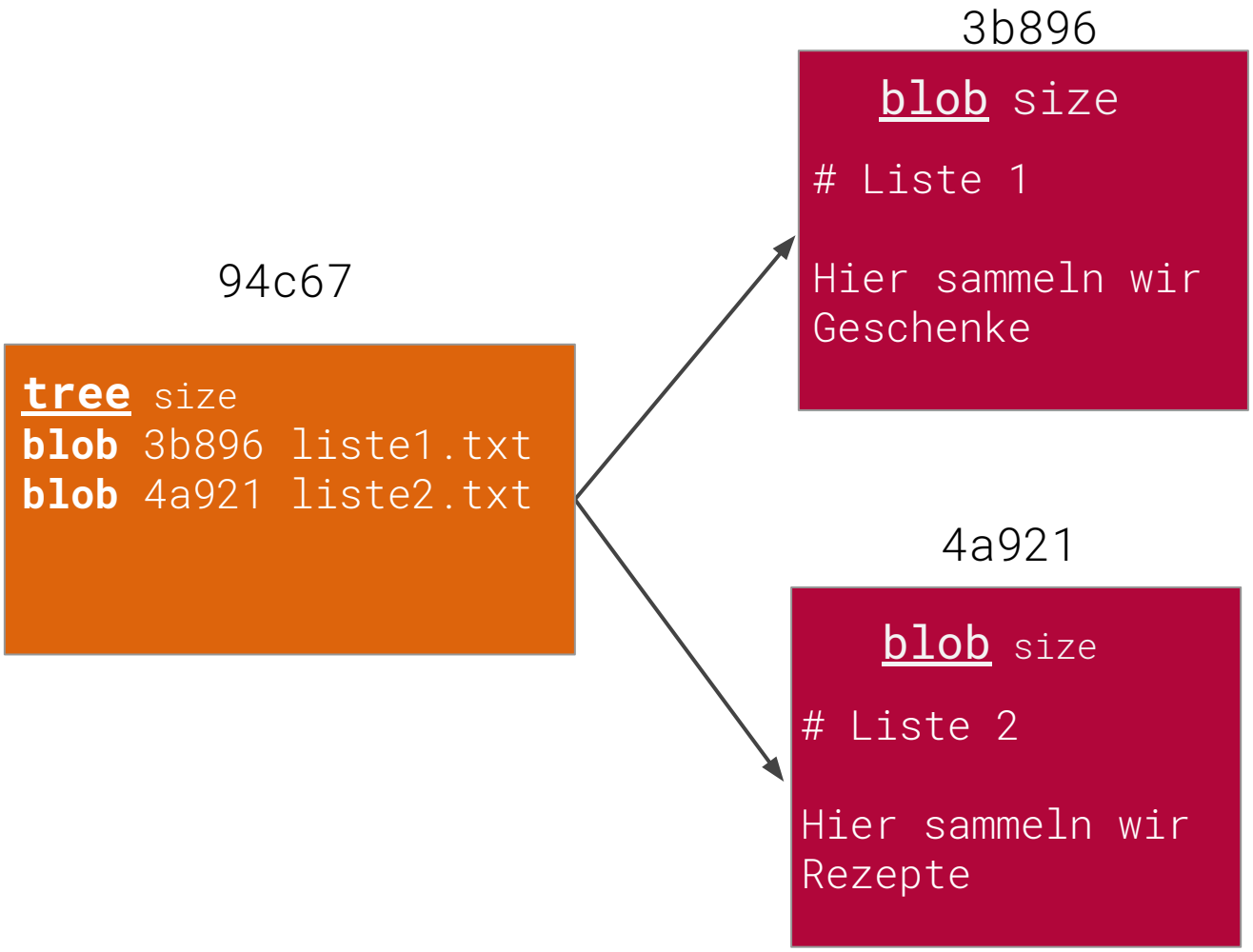
3b896

```
blob size

# Liste 1

Hier sammeln wir
Geschenke
```

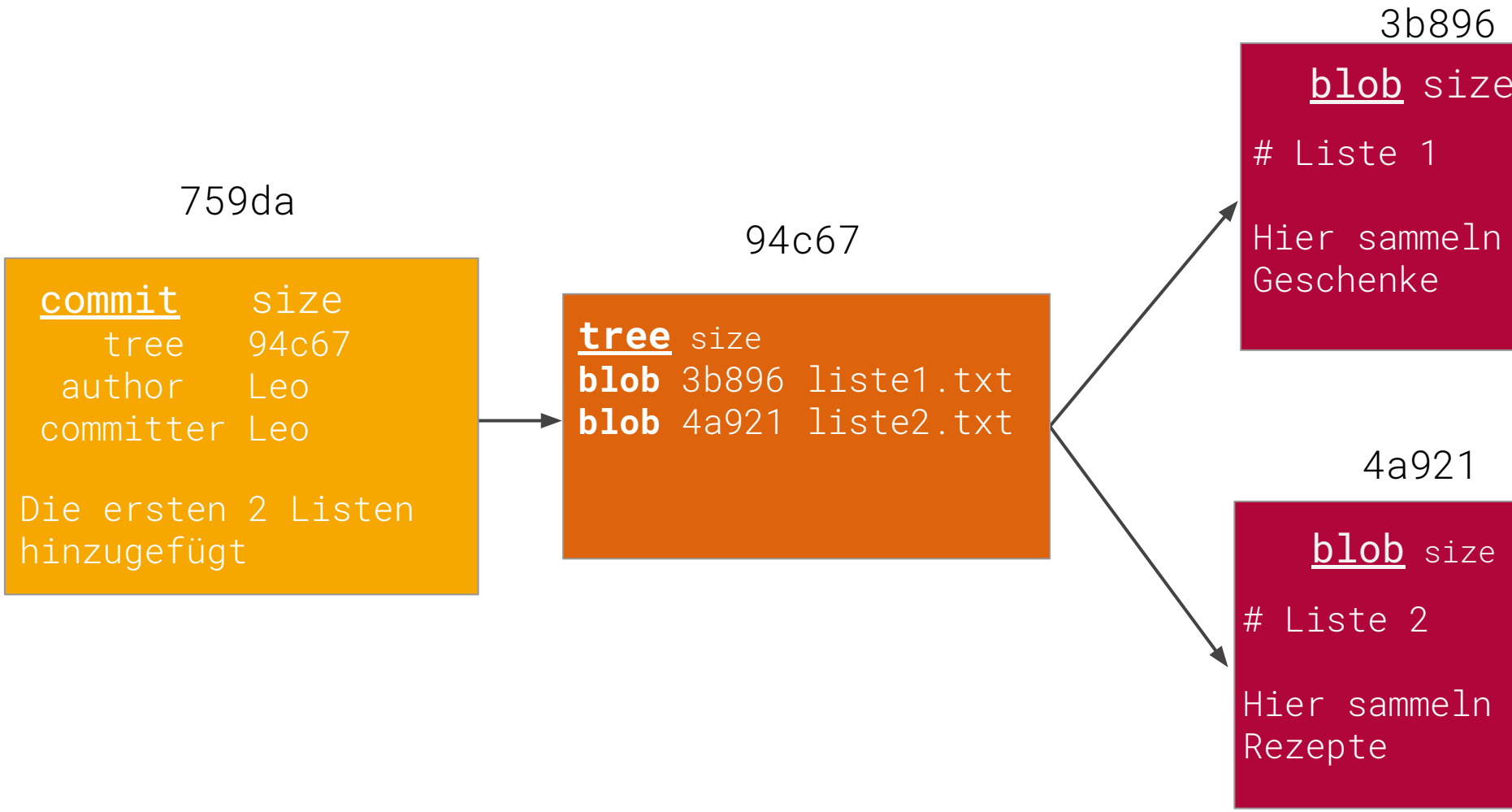
4a921

```
blob size

# Liste 2

Hier sammeln wir
Rezepte
```

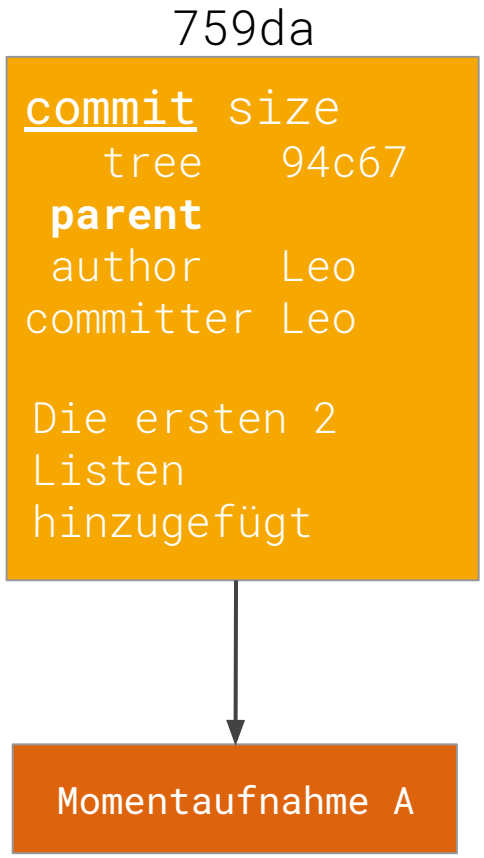
Beispiel


3b896
blob size
Liste 1
Hier sammeln wir Geschenke
94c67
tree size
blob 3b896 liste1.txt
blob 4a921 liste2.txt
4a921
blob size
Liste 2
Hier sammeln wir Rezepte

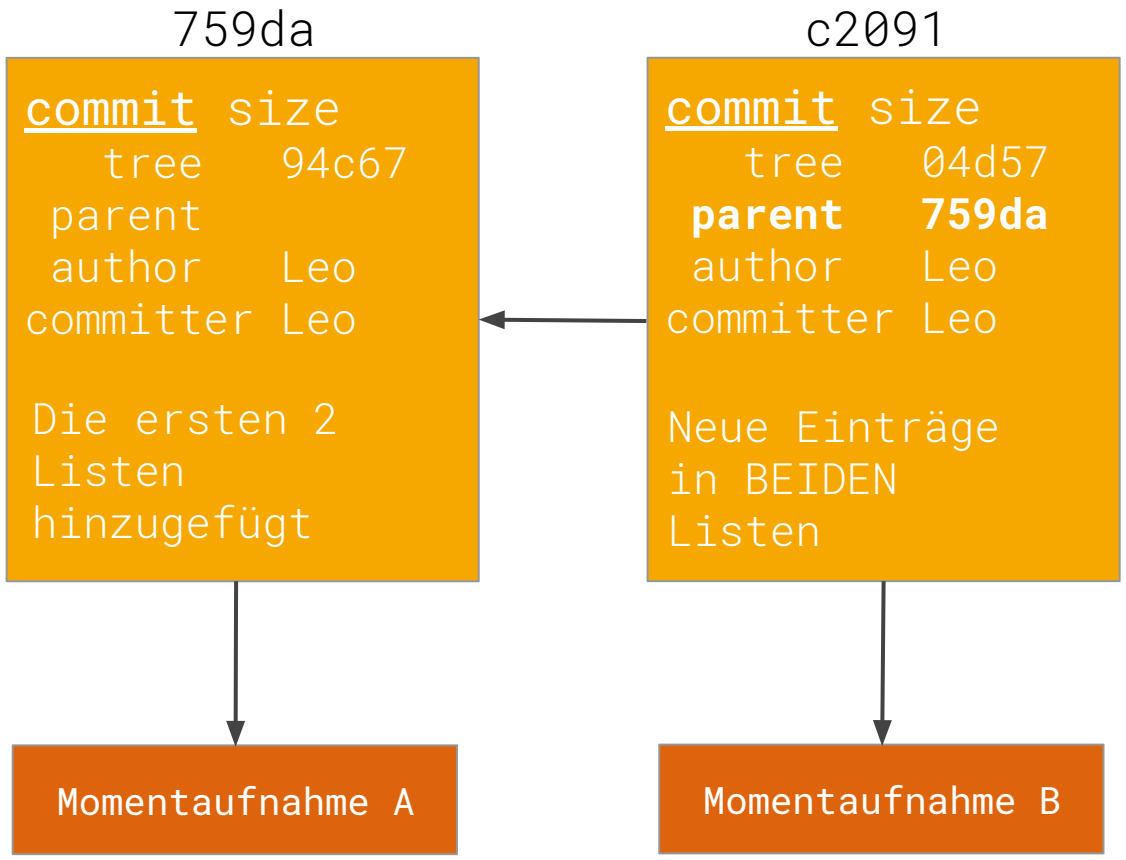
Beispiel


759da
commit size
tree 94c67
author Leo
committer Leo
Die ersten 2 Listen hinzugefügt
94c67
tree size
blob 3b896 liste1.txt
blob 4a921 liste2.txt
3b896
blob size
Liste 1
Hier sammeln Geschenke
4a921
blob size
Liste 2
Hier sammeln Rezepte

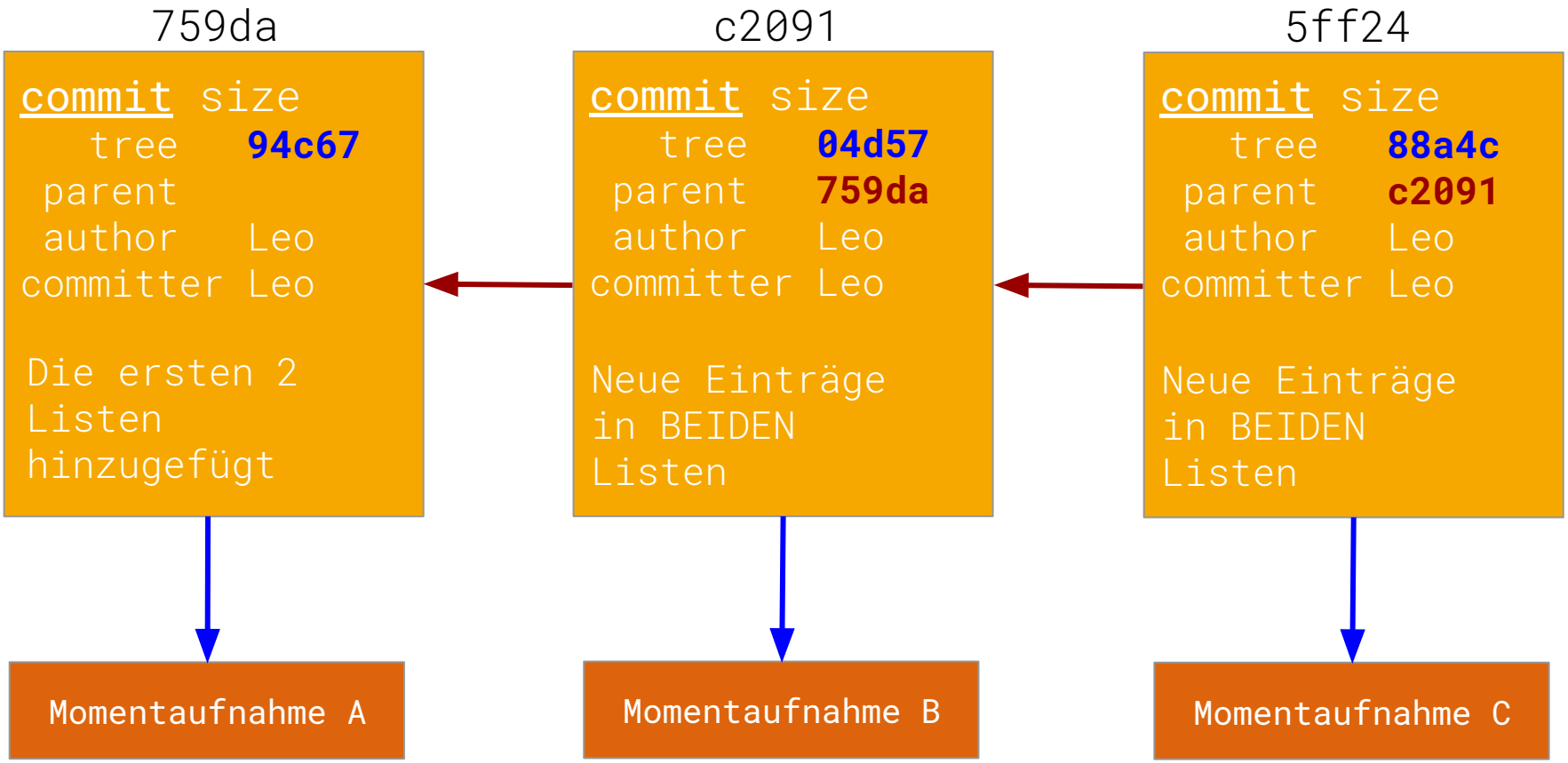
Beispiel mehrere Commits



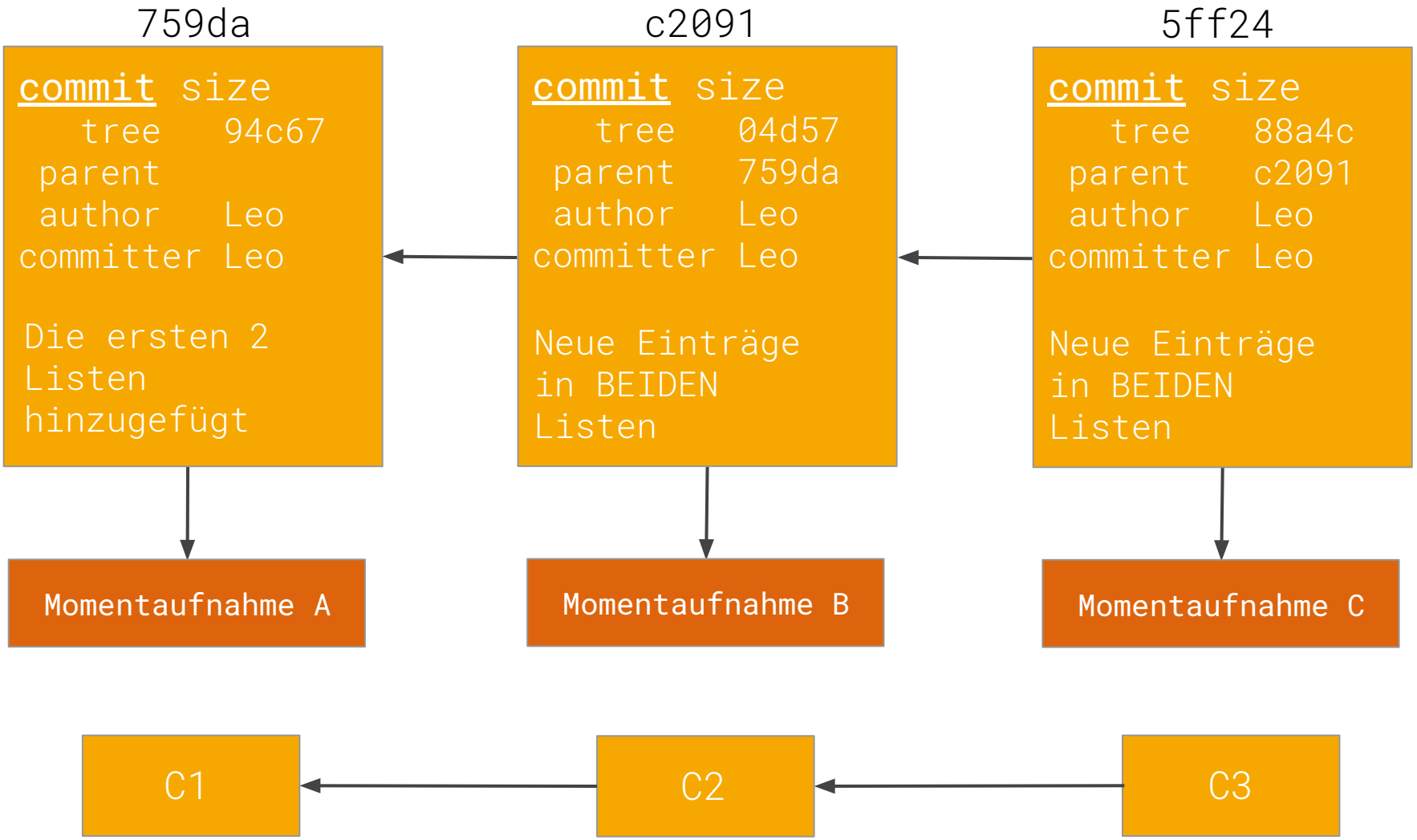
Beispiel mehrere Commits

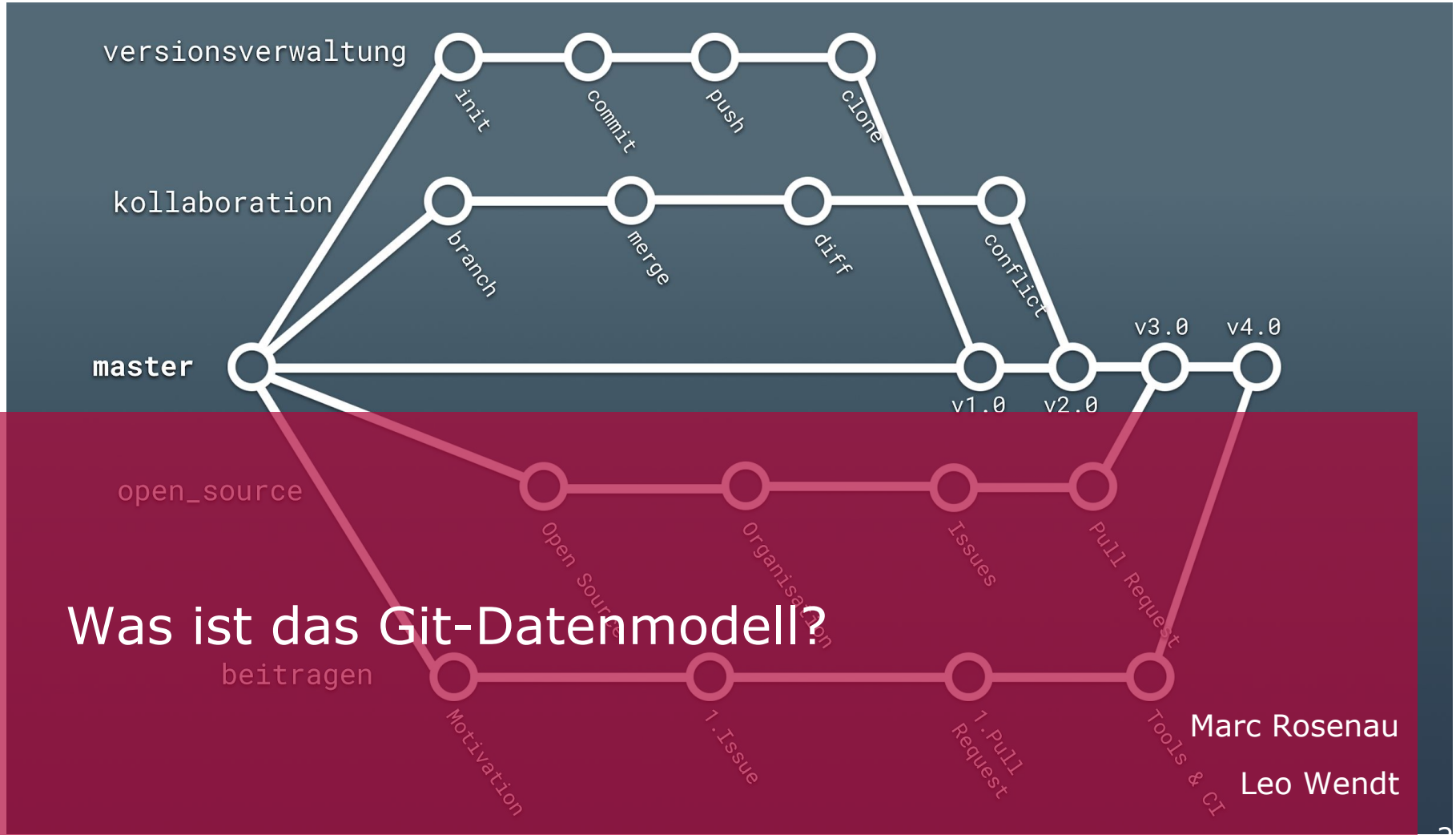


Beispiel mehrere Commits



Beispiel mehrere Commits

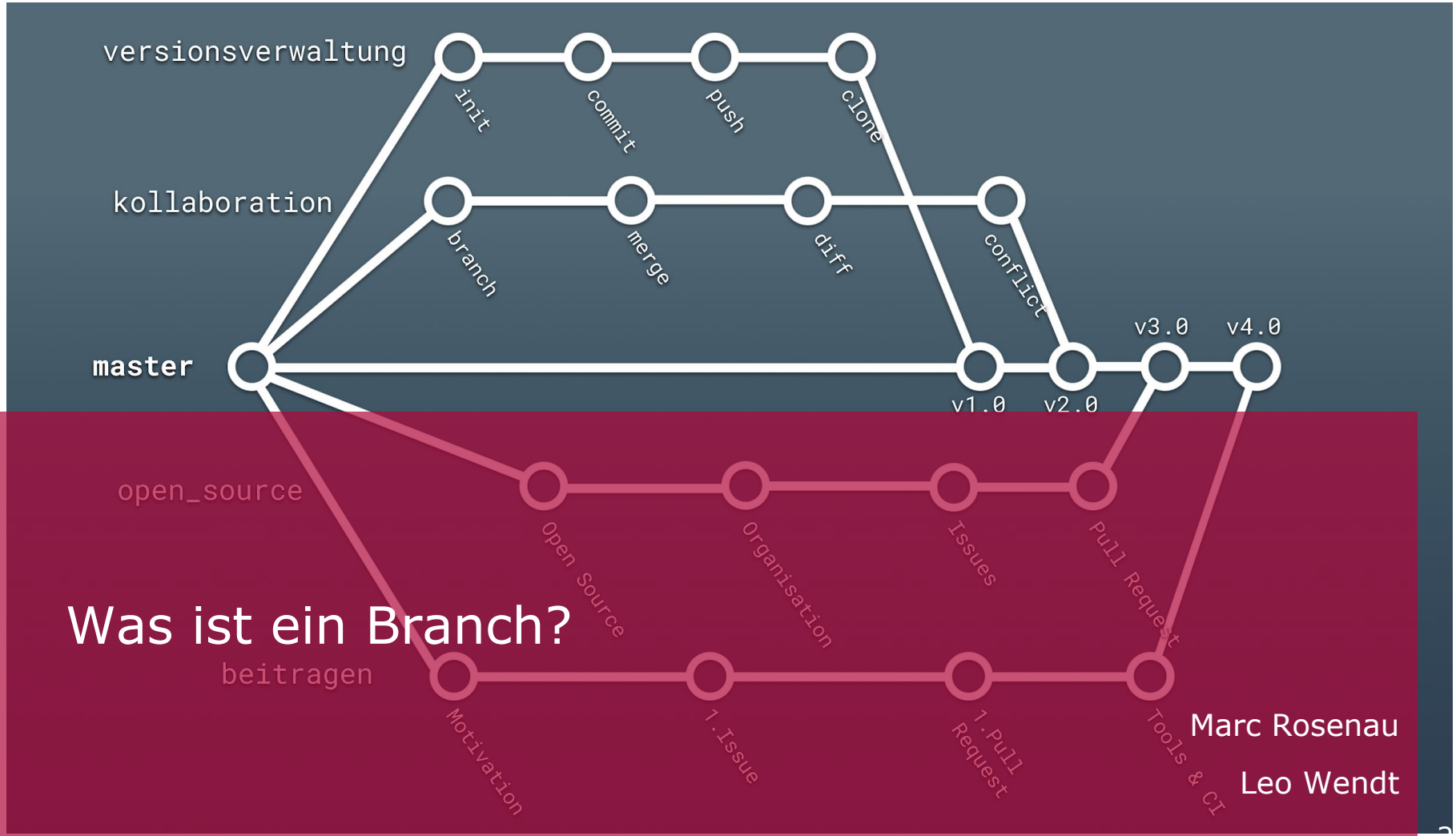




Was ist das Git-Datenmodell?

Marc Rosenau

Leo Wendt



Was ist ein Branch?

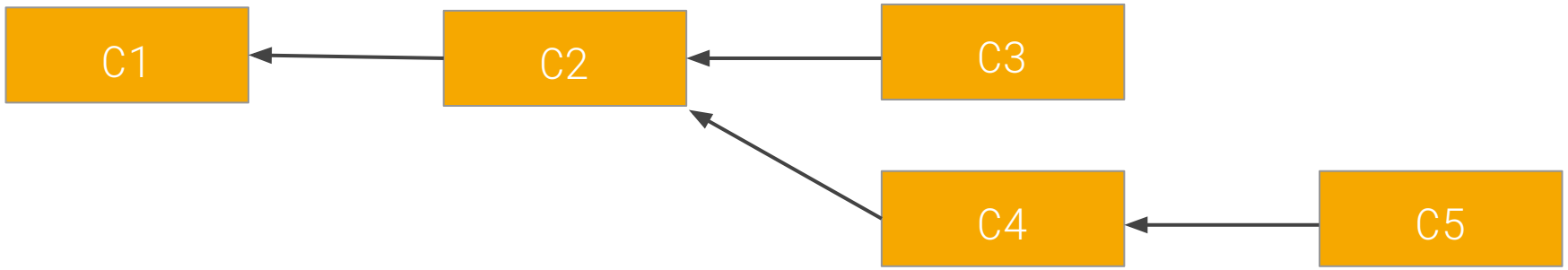
Marc Rosenau

Leo Wendt

Was ist Branching?

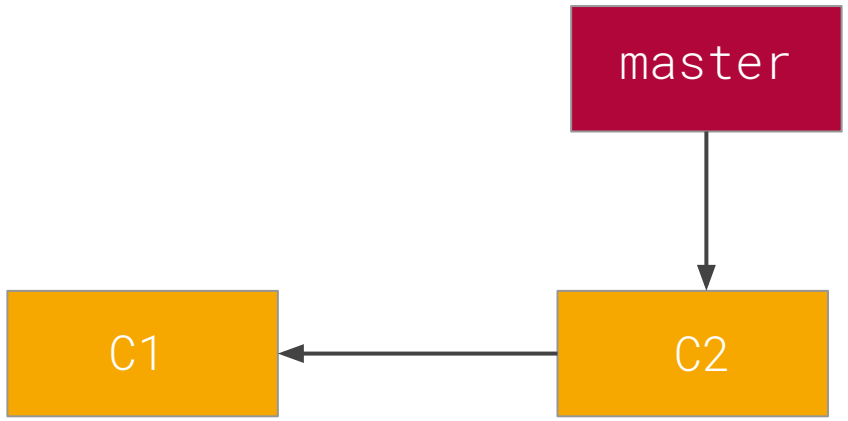
- **Branching:**
  - ermöglicht weiterarbeiten ohne Hauptlinie zu verändern
  - ist in **Git** sehr klein, Operationen damit sehr schnell
  - Git ermutigt deshalb Arbeitsweisen, wo viele Branches erstellt und wieder zusammengeführt werden

→ erlaubt eine produktivere Weise zu arbeiten

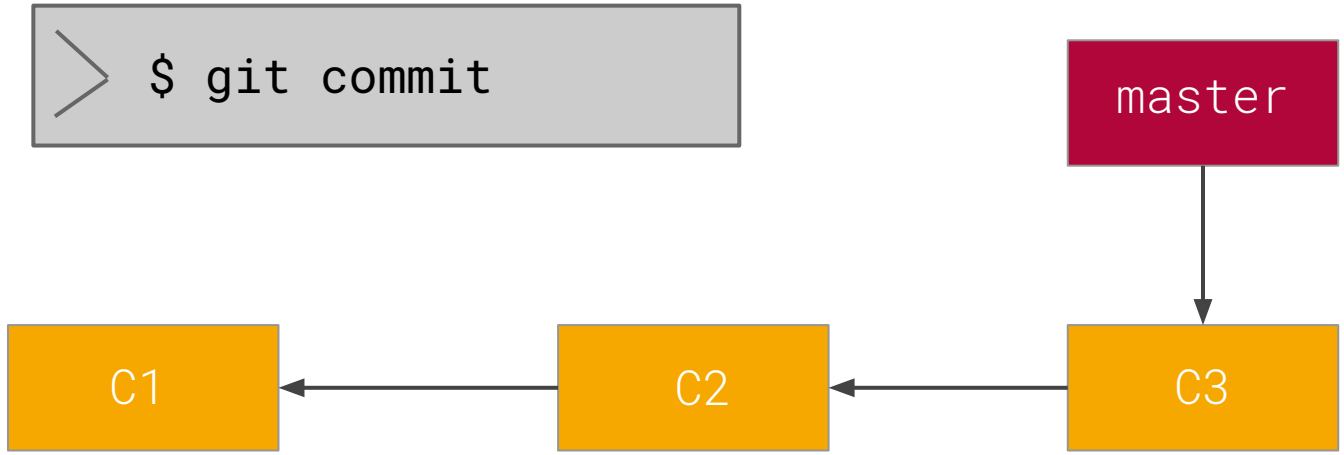


Was ist ein Branch?

- Ein **Branch** ist eine Abzweigung
- Ein **Branch** ist ein beweglicher **Zeiger** auf einen Commit
- Jedes Mal wenn ihr committet, wird der **Zeiger** des **Branches** automatisch bewegt
- Standardbranch ist “`master`”



Was ist ein Branch?



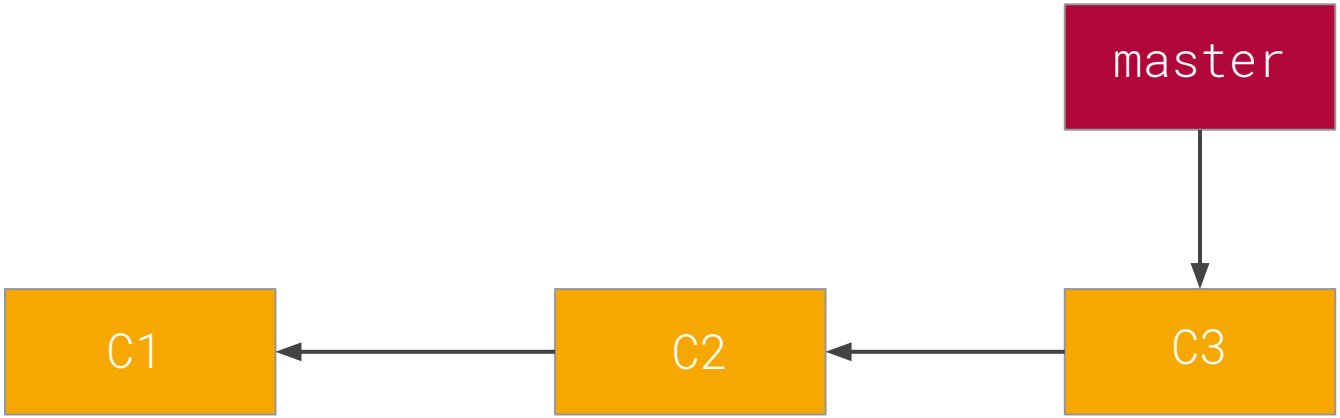
Einen neuen Branch erstellen

- Erstellung eines **Branches** = Erstellung neuer **Zeiger**
- Dies macht ihr mit dem folgenden Befehl:

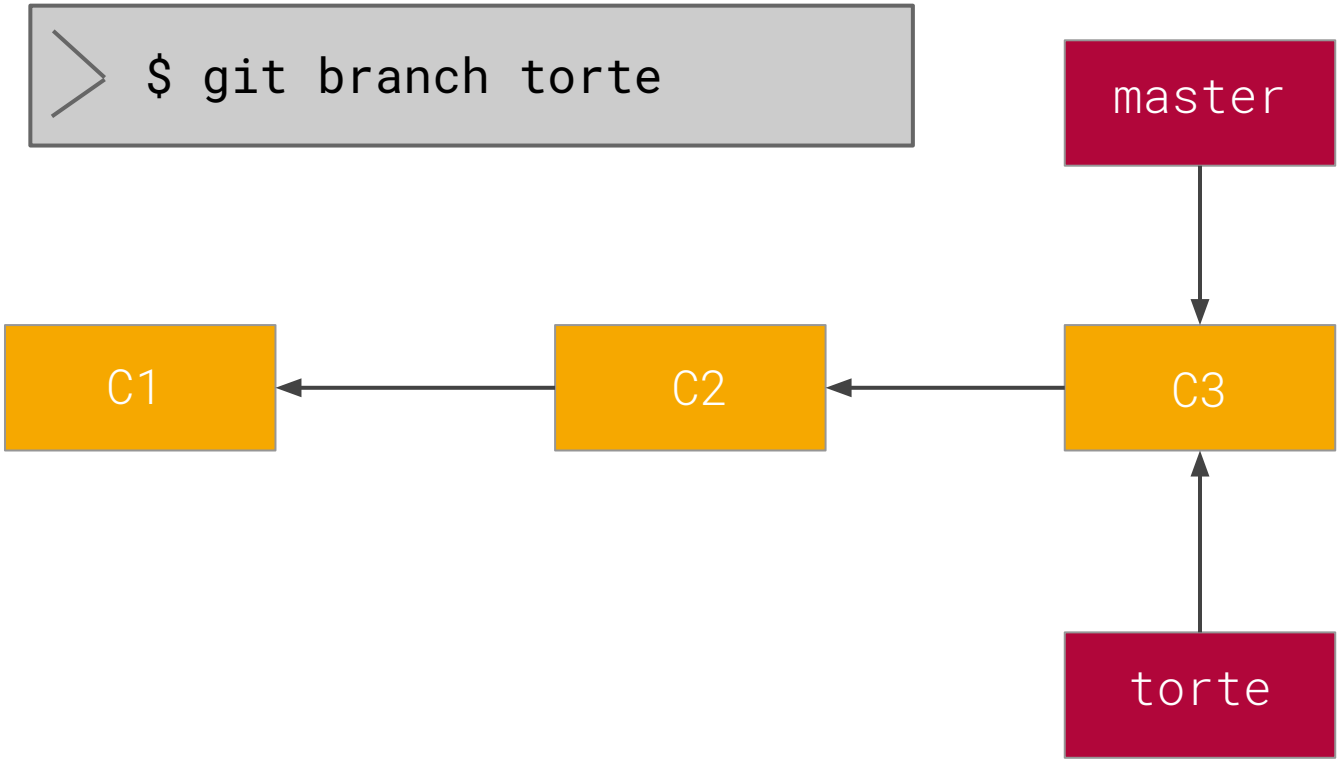
```
$ git branch <branchname>
```

- Der **Zeiger** zeigt auf Commit, auf dem ihr seid

Einen neuen Branch erstellen



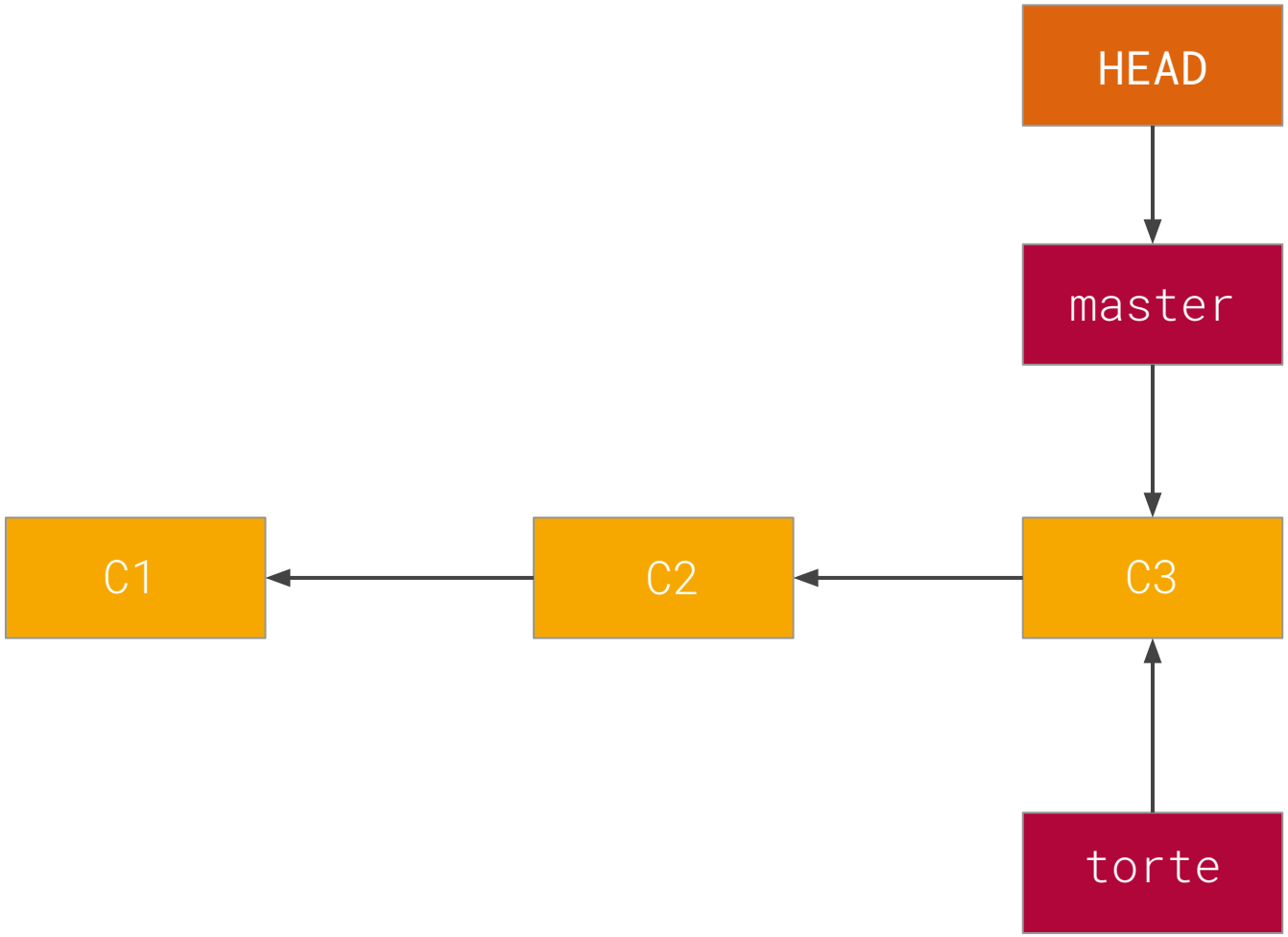
Einen neuen Branch erstellen



Was ist der HEAD?

- Der `git branch` Befehl legt **Branch** an aber wechselt ihn nicht
- Es gibt einen besonderen Zeiger, welcher HEAD genannt wird
- HEAD ist **Zeiger** zum aktuellen **Branch**

Was ist der HEAD?



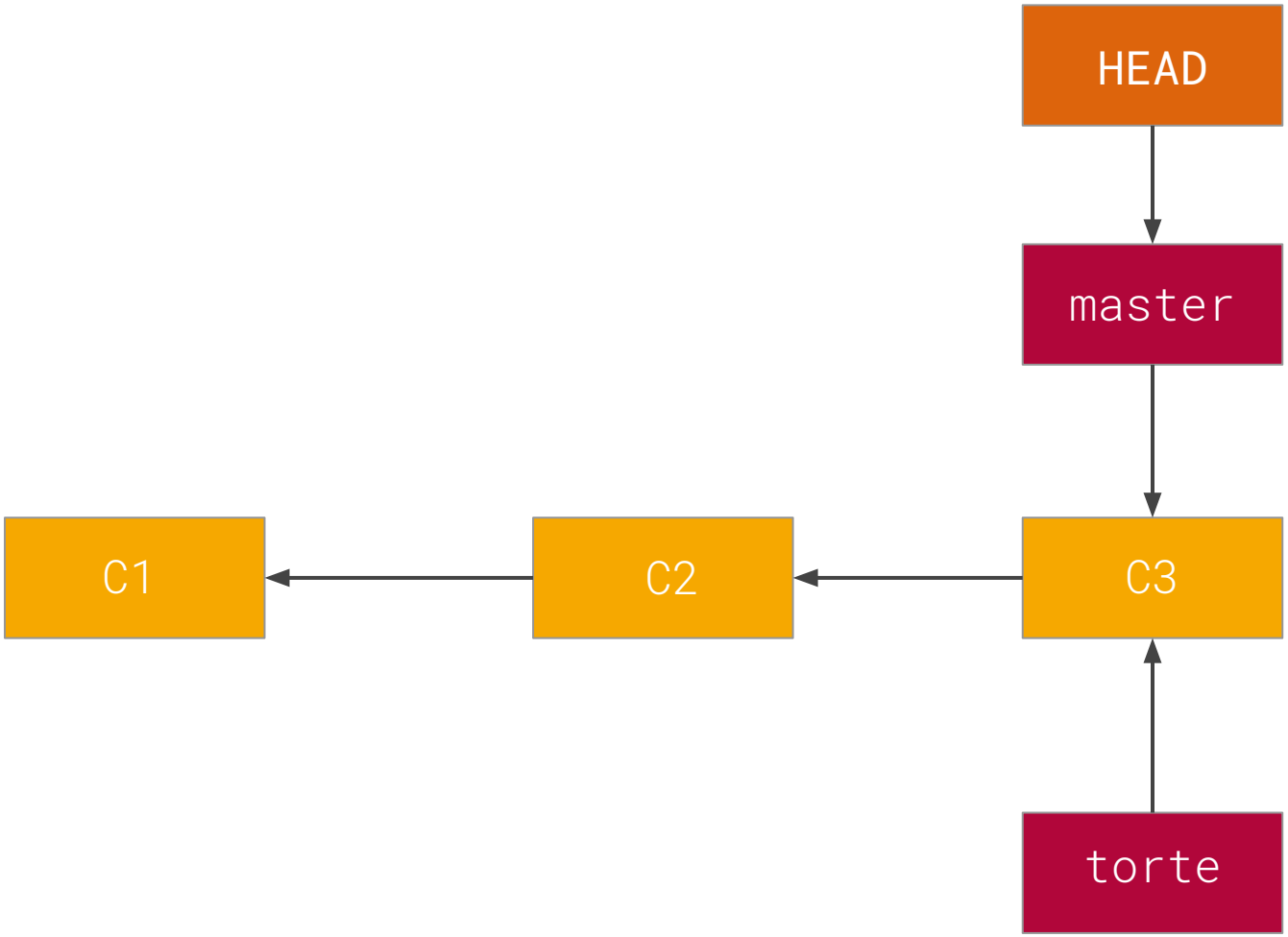
Branches wechseln

- Um zu existierenden **Branch** zu wechseln, benutzt

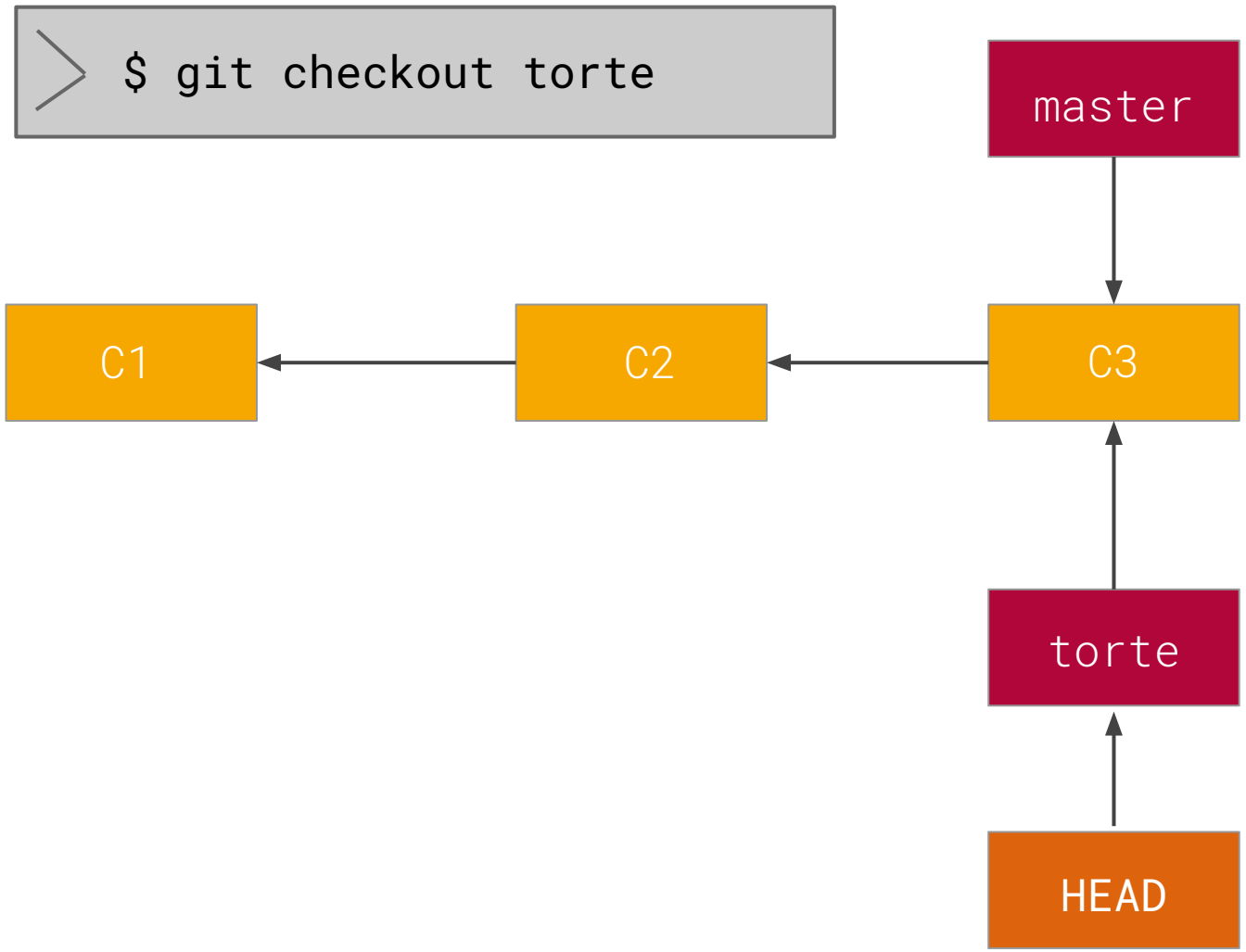
```
$ git checkout <branchname>
```

- Dies lässt HEAD auf **Branch** zeigen

Branches wechseln

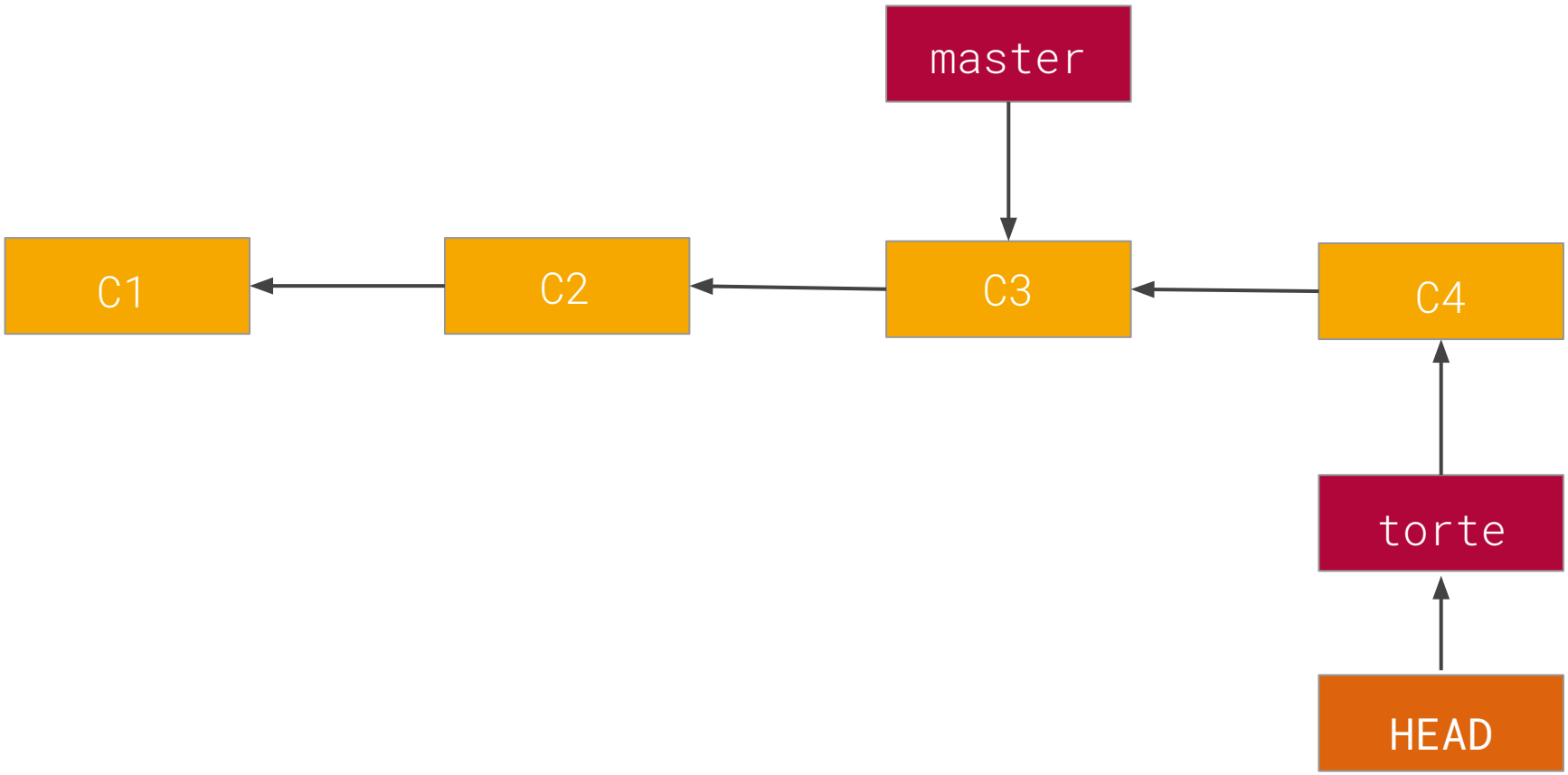


Branches wechseln



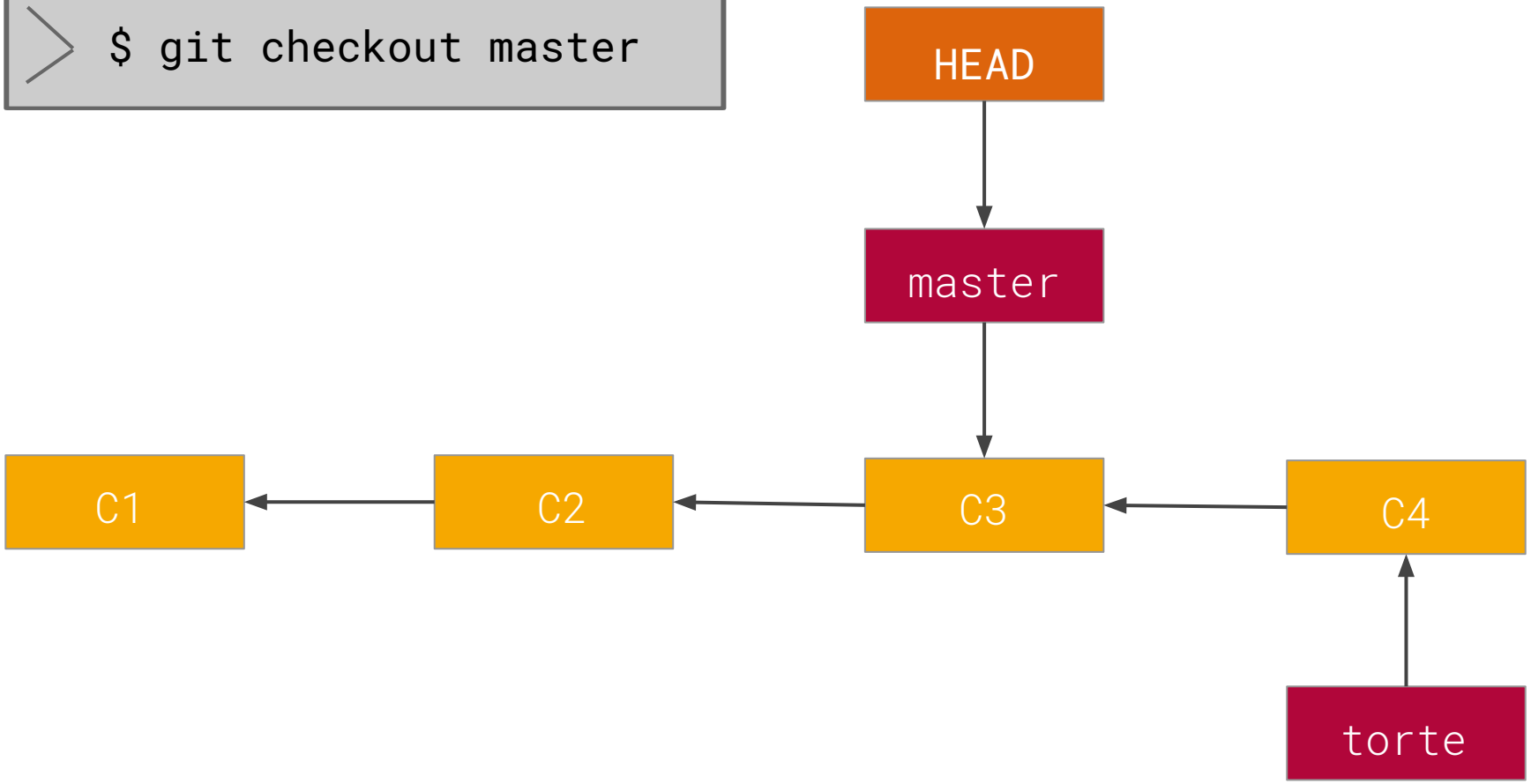
Branches wechseln

```
$ git commit
```



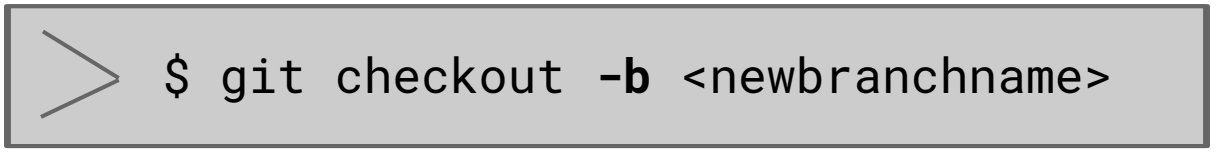
Branches wechseln

```
$ git checkout master
```



Checkout und Branch zur gleichen Zeit

- Gleichzeitig einen neuen **Branch** anlegen und auf ihn wechseln:



Cheat Sheet

Erstellen von Branches

```
$ git branch <name>
```

Wechseln von Branches

```
$ git checkout <branch>
```

Zusammenführen von Branches

```
$ git merge <branch>
```

Commit Historie anzeigen

```
$ git log
```

Unstaging von Datei

```
$ git reset HEAD <datei>
```

Änderungen an Datei verwerfen

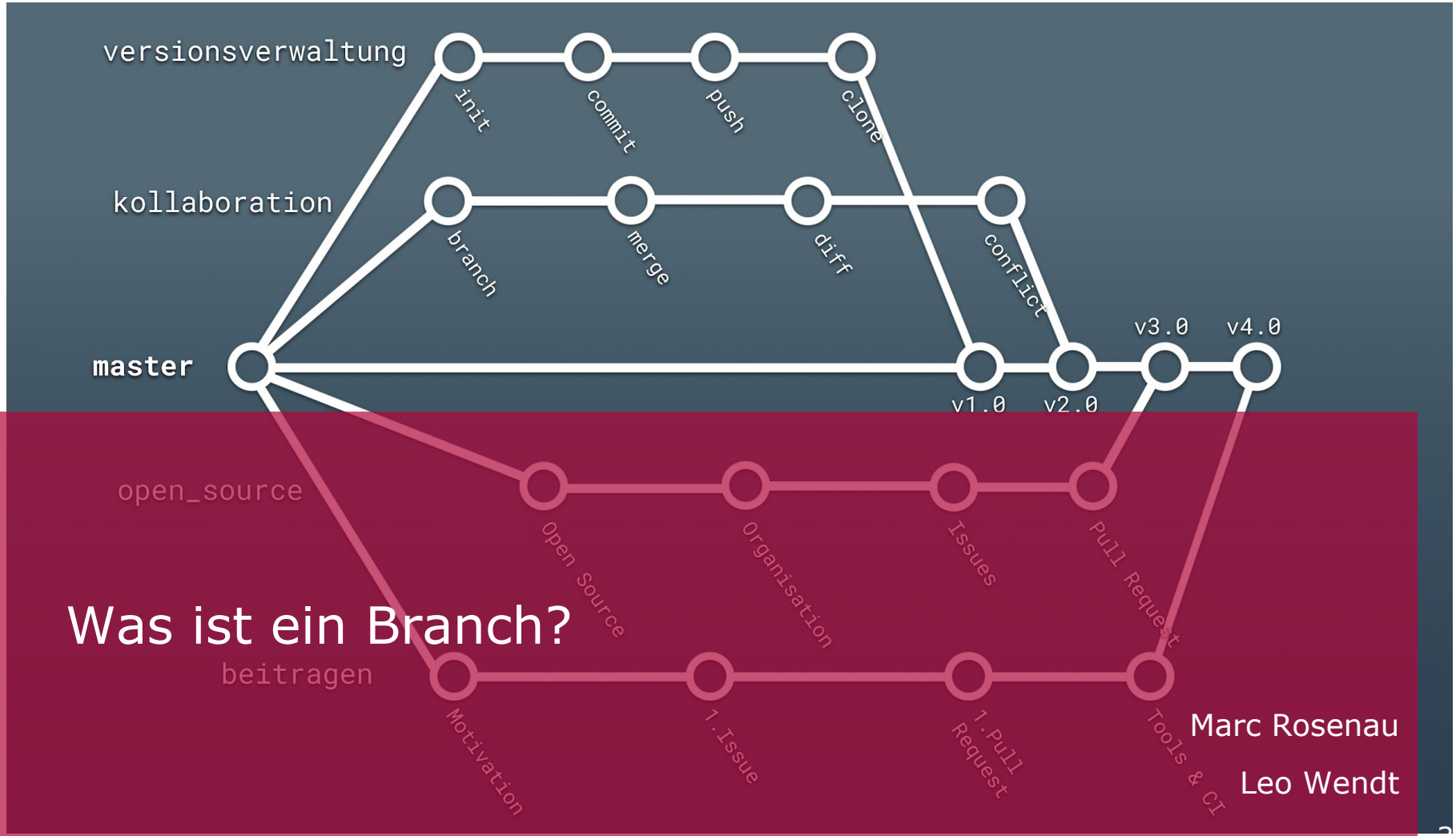
```
$ git checkout -- <datei>
```

Letzten Commit verändern

```
$ git commit --amend
```

Zu Commit zurückkehren

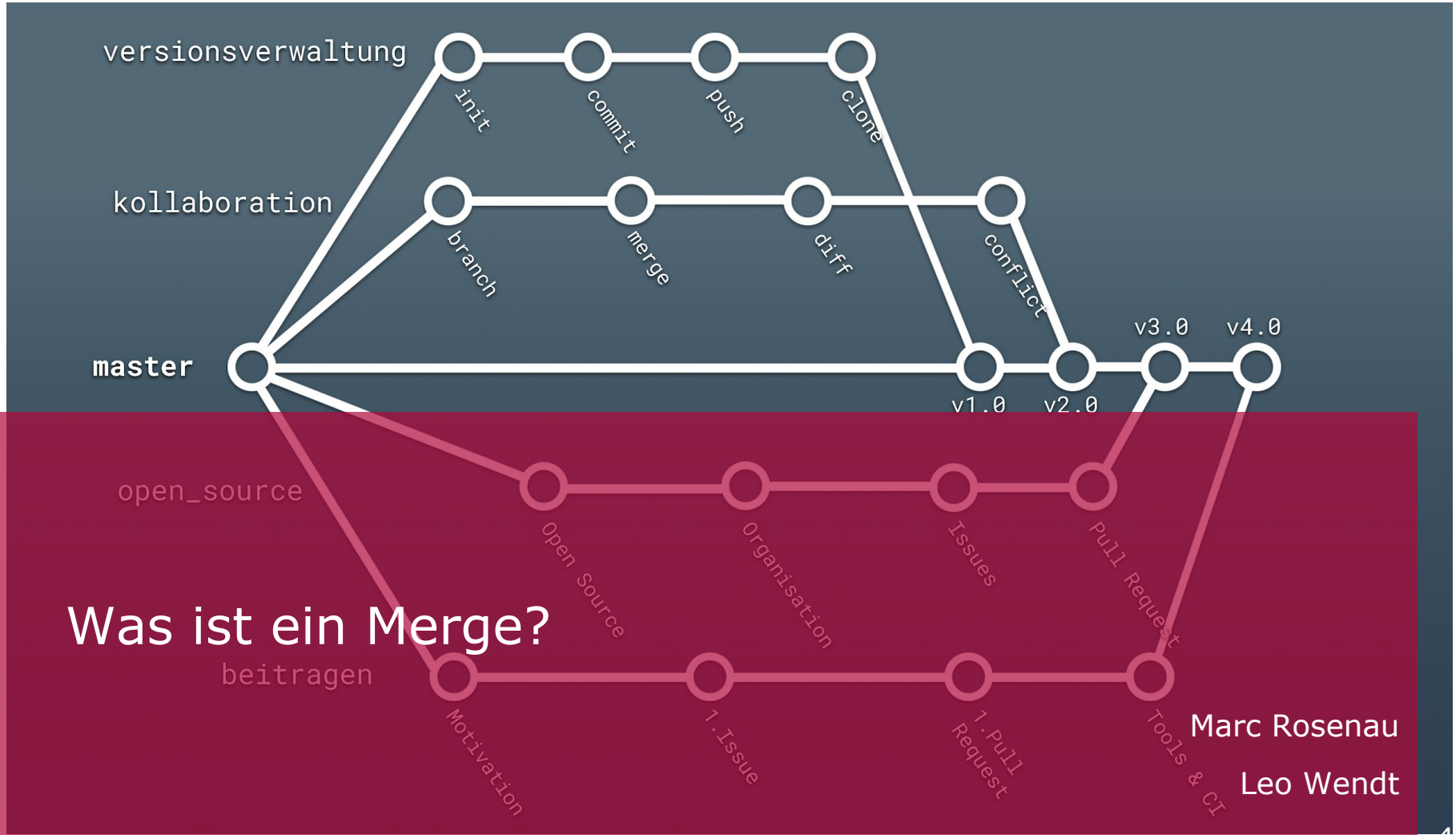
```
$ git reset <commit>
```



Was ist ein Branch?

Marc Rosenau

Leo Wendt

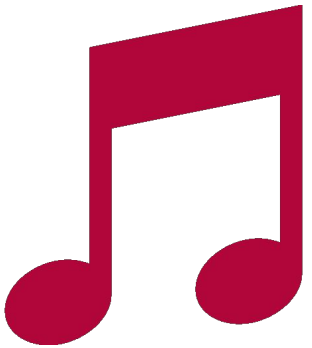


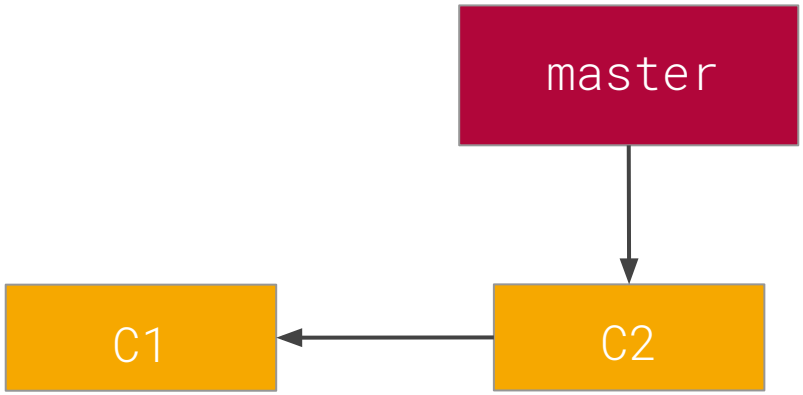
Was ist ein Merge?

Marc Rosenau

Leo Wendt

Einleitung

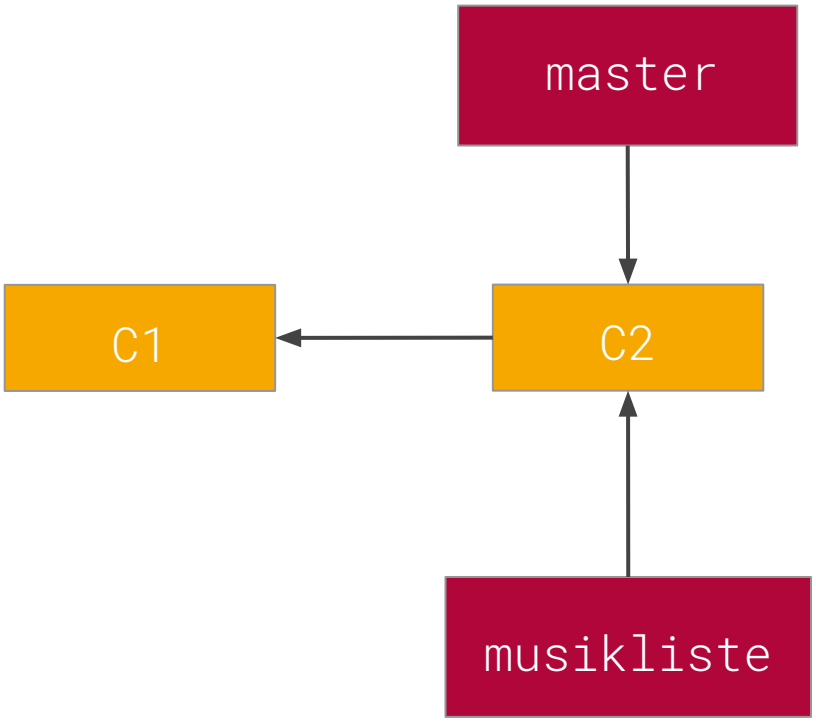




Erstellen eines Branch

```
$ git checkout -b musikliste
```

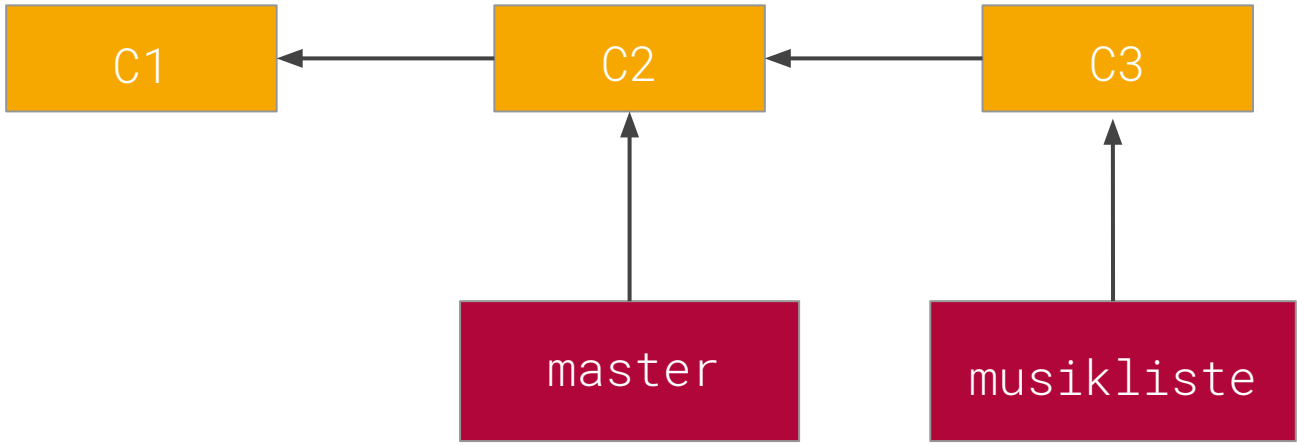
```
Switched to a new branch "musikliste"
```



Arbeiten an der Musikliste

Bearbeiten und Committen von Dateien

```
$ git commit -m ‘Überraschungslied hinzugefügt’
```



Arbeiten an der Musikliste

Wechseln zum Master

- Zurückwechseln auf **Master-Branch**

```
$ git checkout master
```

```
Switched to branch ‘master’
```

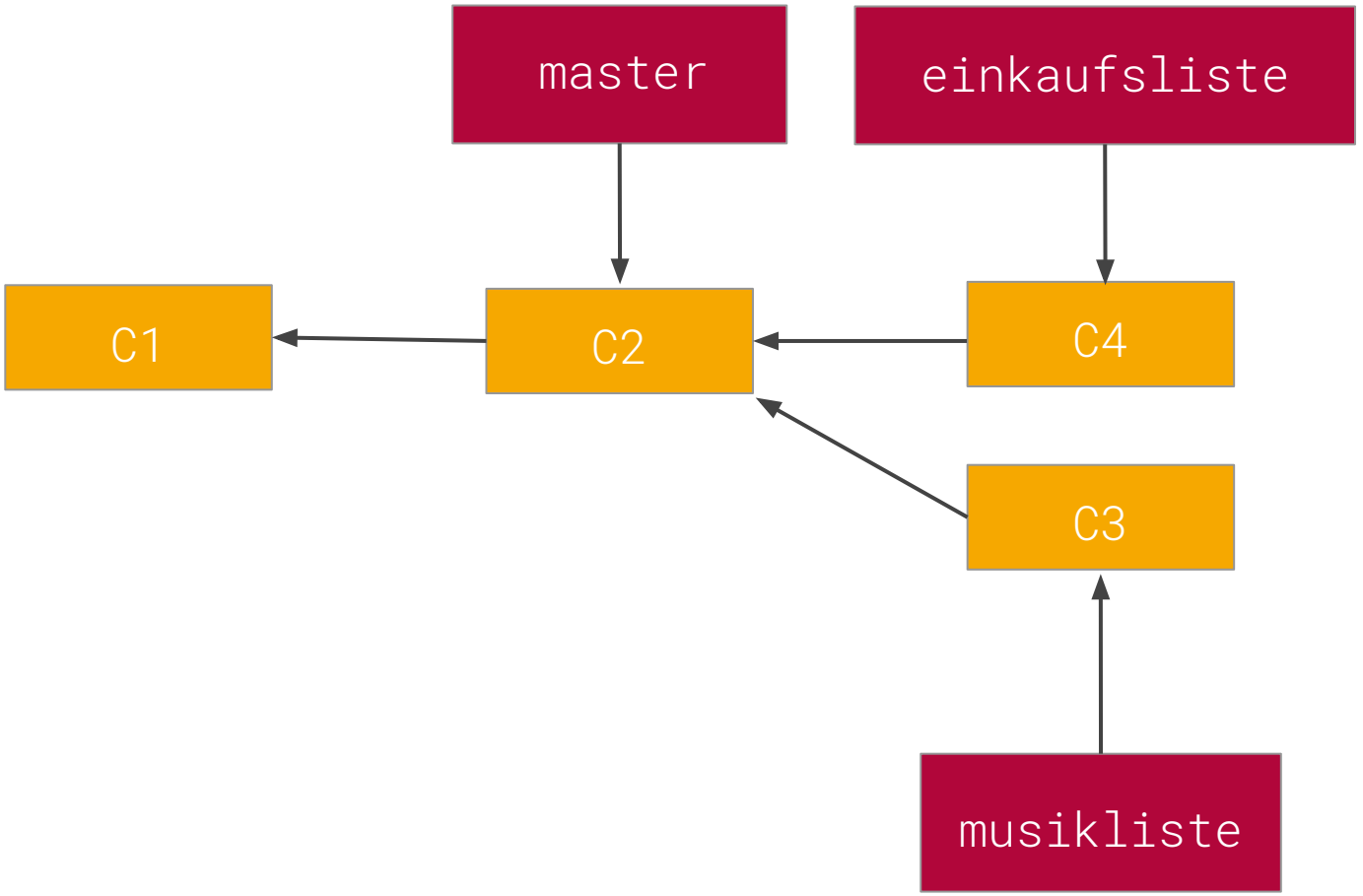
- Arbeitsverzeichnis selben Zustand wie vor dem Arbeiten an Musikliste
- Arbeit an Musikliste aber auf Branch gespeichert

Erstellen des Einkaufslisten-Branch

```
> $ git checkout -b einkaufsliste
Switched to a new branch einkaufsliste
```

```
> $ git commit -m 'alle Zutaten hinzugefügt'
[einkaufsliste 1fb7853] alle Zutaten hinzugefügt
 1 file changed, 6 insertions(+)
```

Erstellen des Einkaufslisten-Branch



Branches zusammenführen

- Zusammenführungen von Branches erfolgt mit:

```
$ git merge <branchname>
```

Mergen des Einkaufslisten-Branch

```
$ git checkout master
```

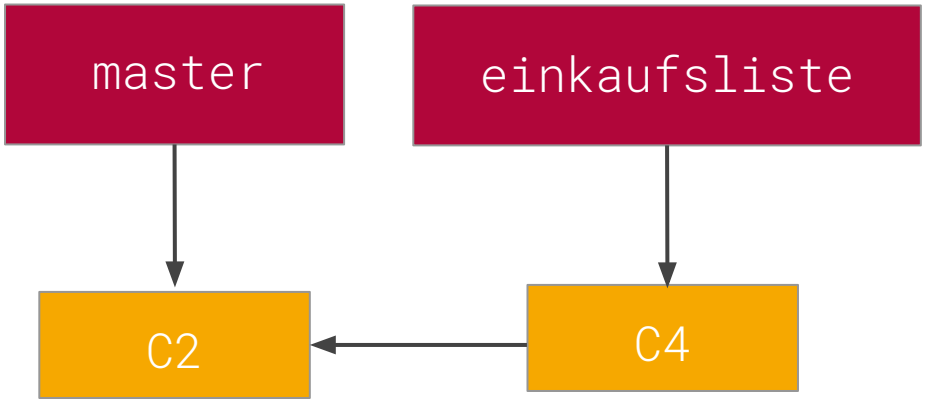
```
Switched to branch ‘master’
```

```
$ git merge einkaufsliste
```

```
Updating f42c576..3a0874c
Fast-forward
einkaufsliste.txt | 6 ++
1 file changed, 6 insertions(+)
```

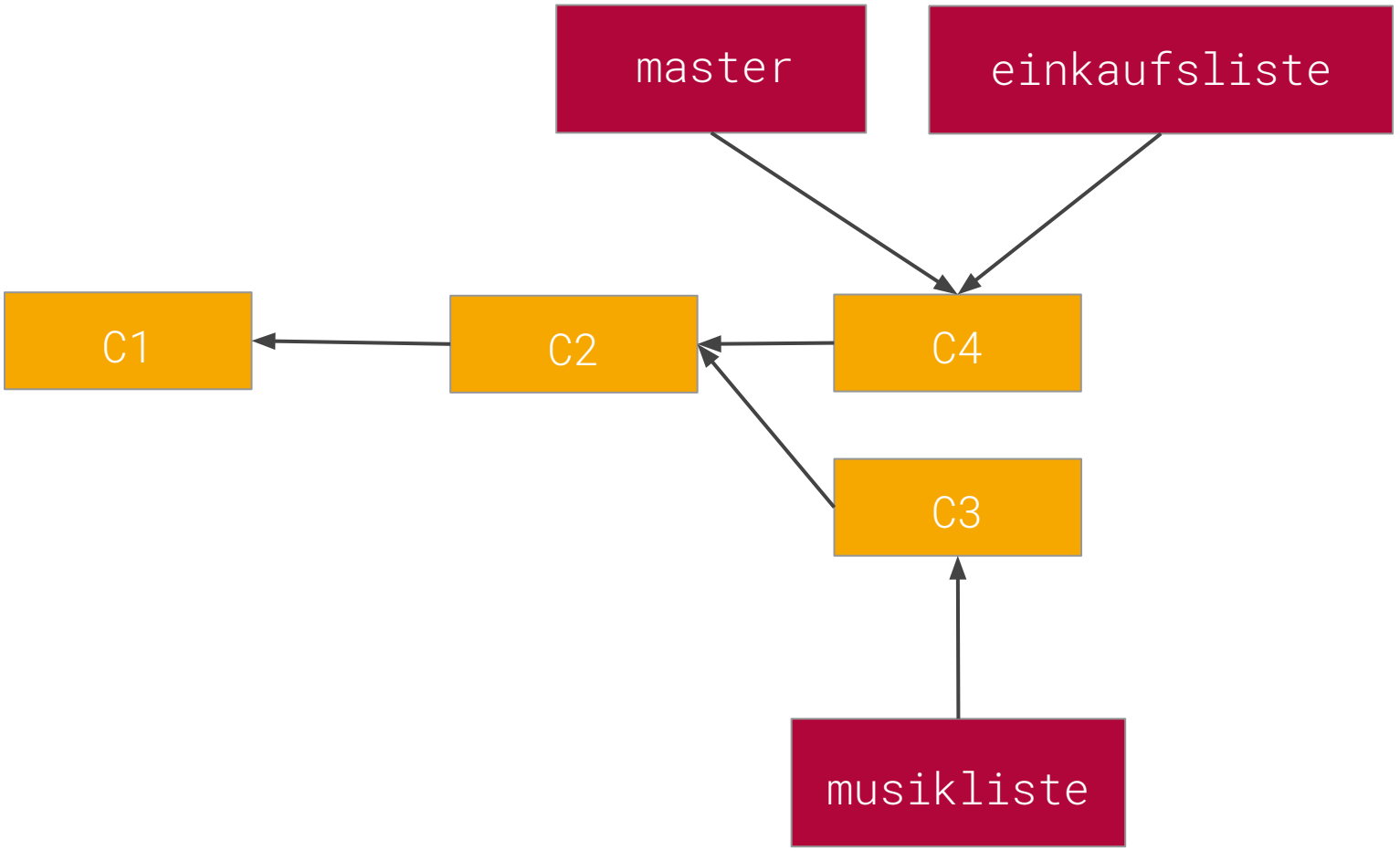
Mergen des Einkaufslisten-Branch

- **C2** direkt vor **C4**

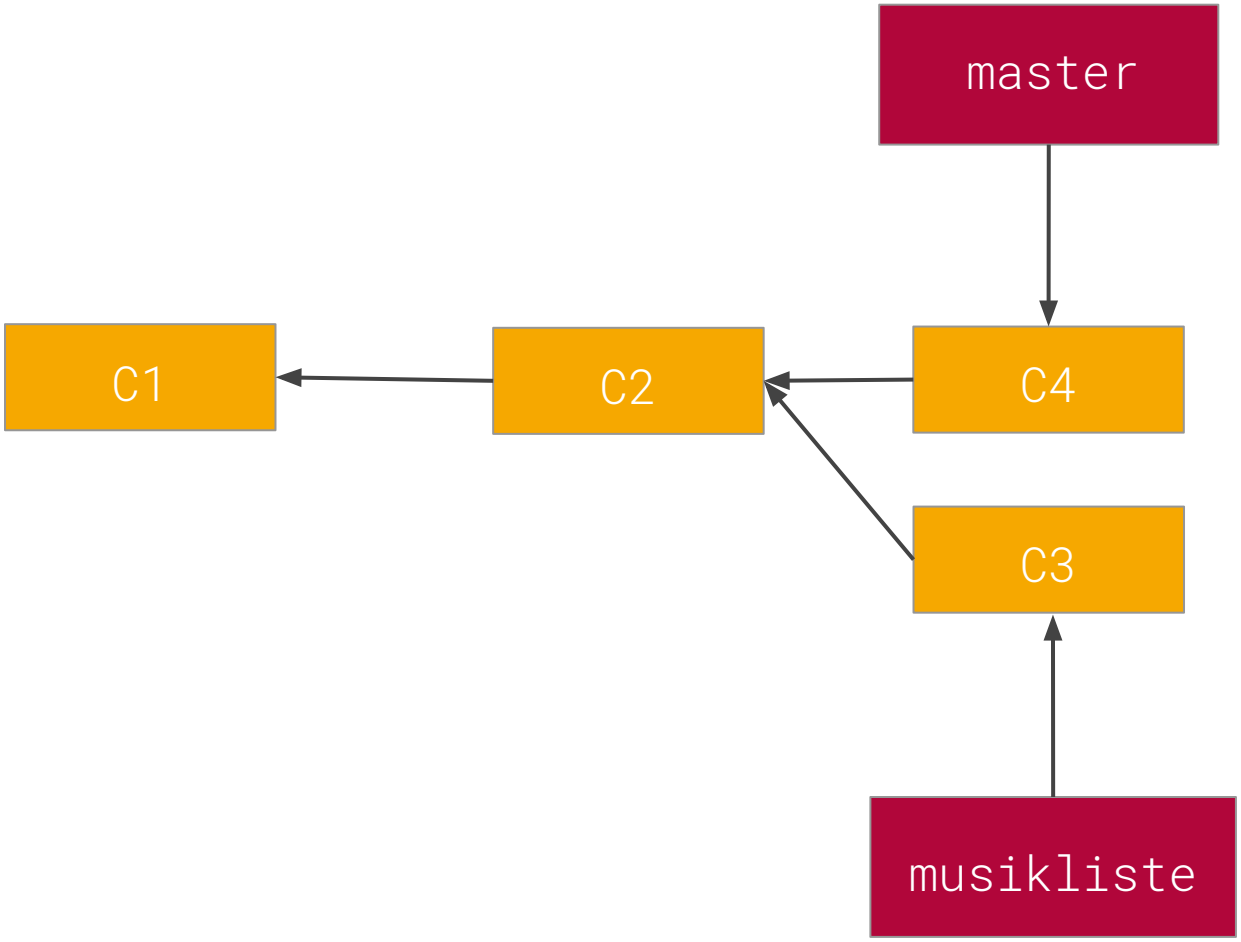


- Git bewegt Zeiger einfach nach vorn
- **Master**-Branch und **Einkaufsliste**-Branch zeigen auf selben Commit

Mergen des Einkaufslisten-Branch



Mergen des Einkaufslisten-Branch



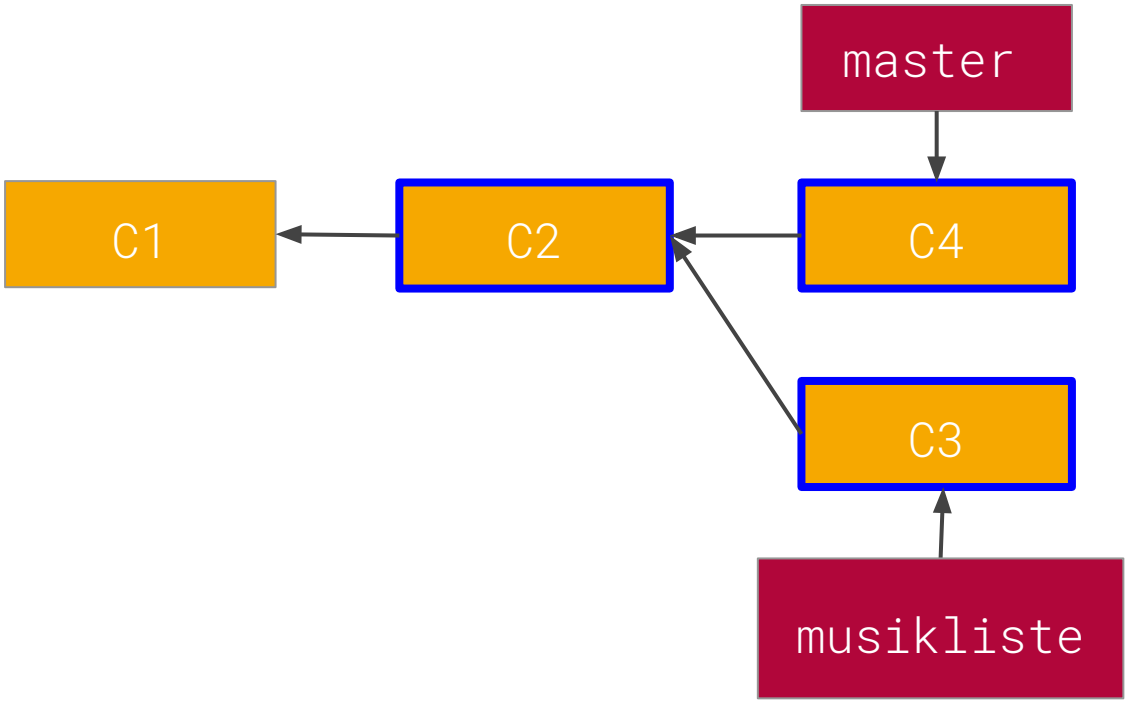
Mergen des Musiklisten-Branch

```
$ git merge musikliste
```

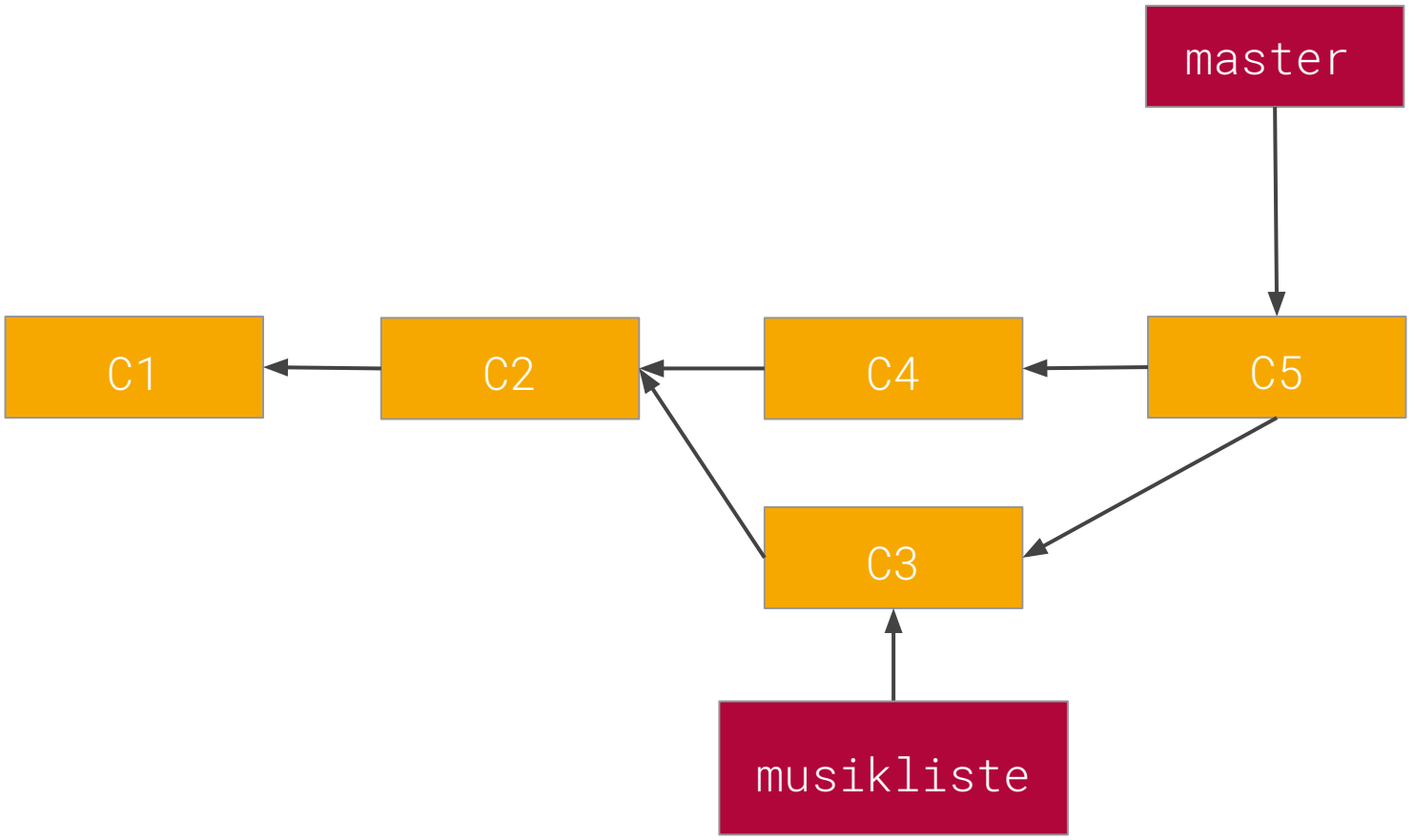
```
Merge made by the 'recursive' strategy.
musik.txt |    1 +
1 file changed, 1 insertion(+)
```

Mergen des Musiklisten-Branch

- Branches an vorherigen Zeitpunkt geteilt
  - `master` ist kein unmittelbarer Vorgänger
- wird von git zusammengeführt



Mergen des Musiklisten-Branch



Cheat Sheet

Erstellen von Branches

```
$ git branch <name>
```

Wechseln von Branches

```
$ git checkout <branch>
```

Zusammenführen von Branches

```
$ git merge <branch>
```

Commit Historie anzeigen

```
$ git log
```

Unstaging von Datei

```
$ git reset HEAD <datei>
```

Änderungen an Datei verwerfen

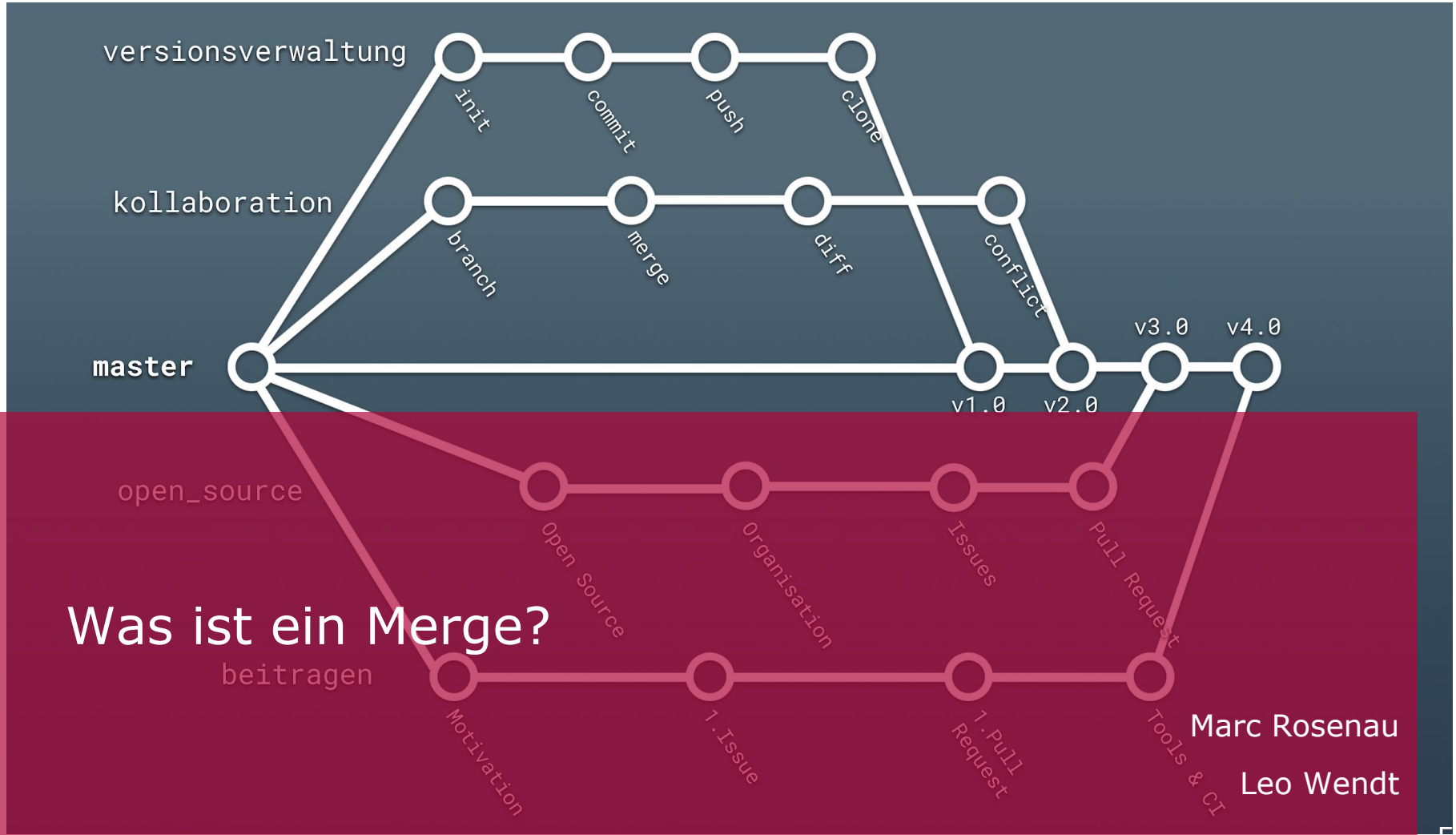
```
$ git checkout -- <datei>
```

Letzten Commit verändern

```
$ git commit --amend
```

Zu Commit zurückkehren

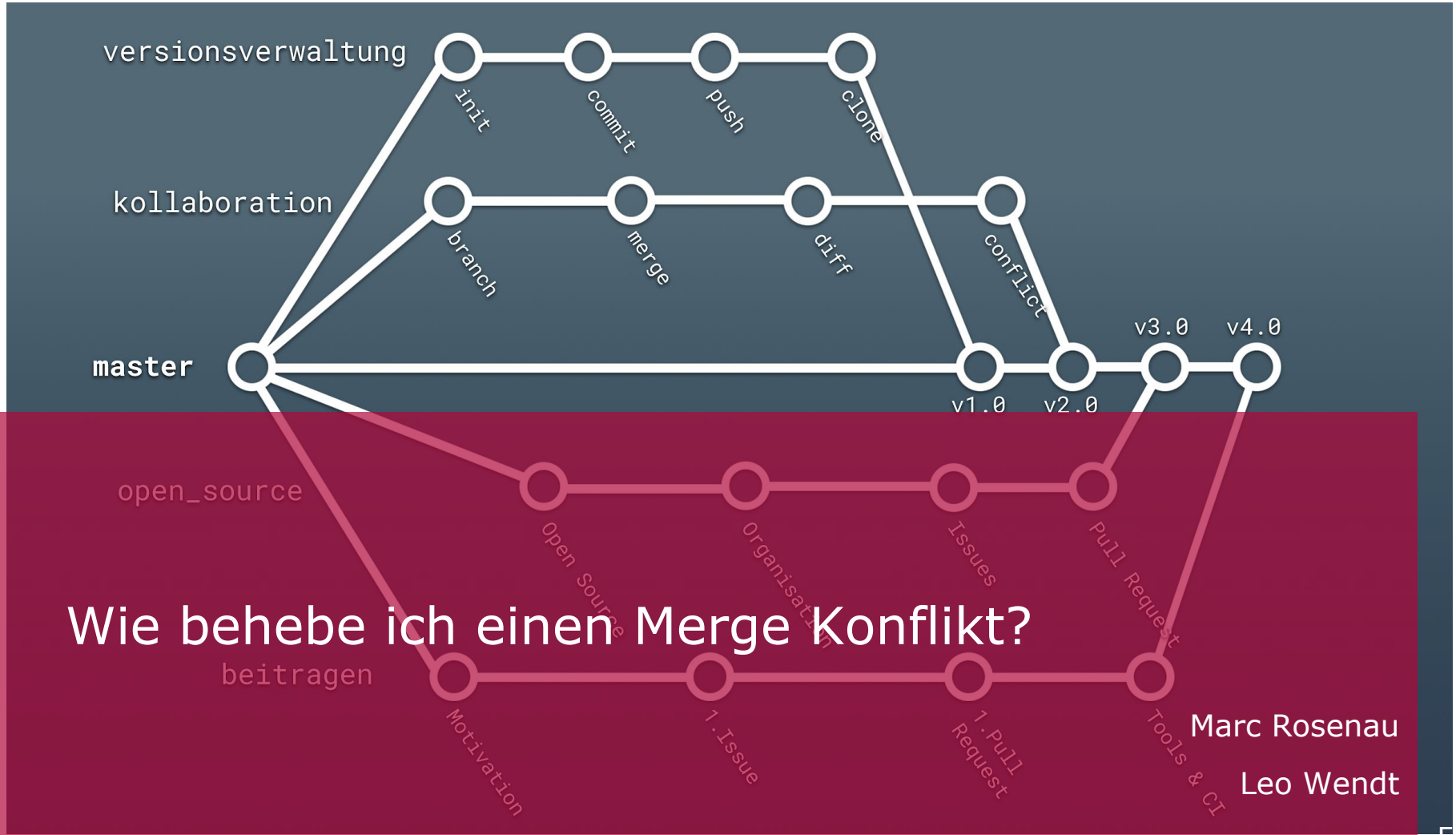
```
$ git reset <commit>
```



Was ist ein Merge?

Marc Rosenau

Leo Wendt



Wie behebe ich einen Merge Konflikt?

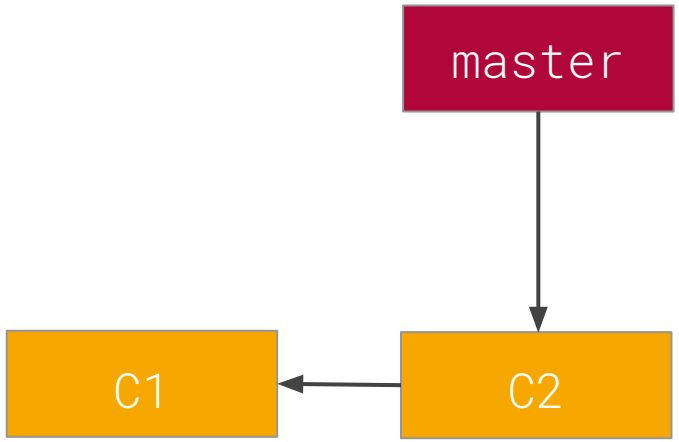
Marc Rosenau

Leo Wendt

Einfache Merge Konflikte

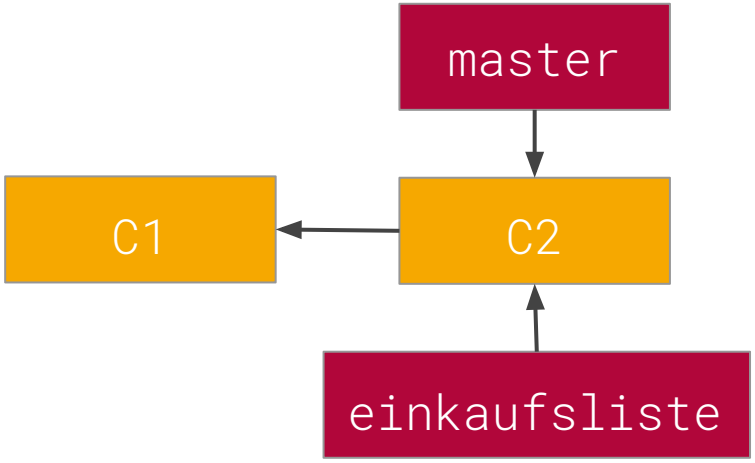
master-Branch
gästeliste.txt

-Sandro
-Til
-Caterina
-Udo



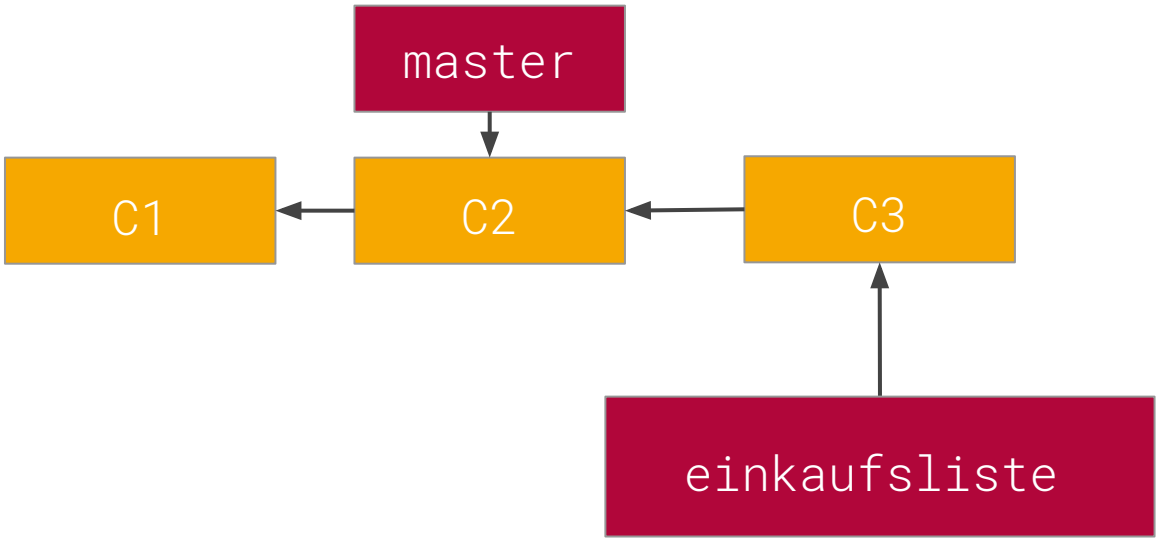
Einfache Merge Konflikte

master-Branch gästeliste.txt	einkaufliste-Branch gästeliste.txt
-Sandro -Til -Caterina -Udo	-Sandro -Til -Caterina -Udo



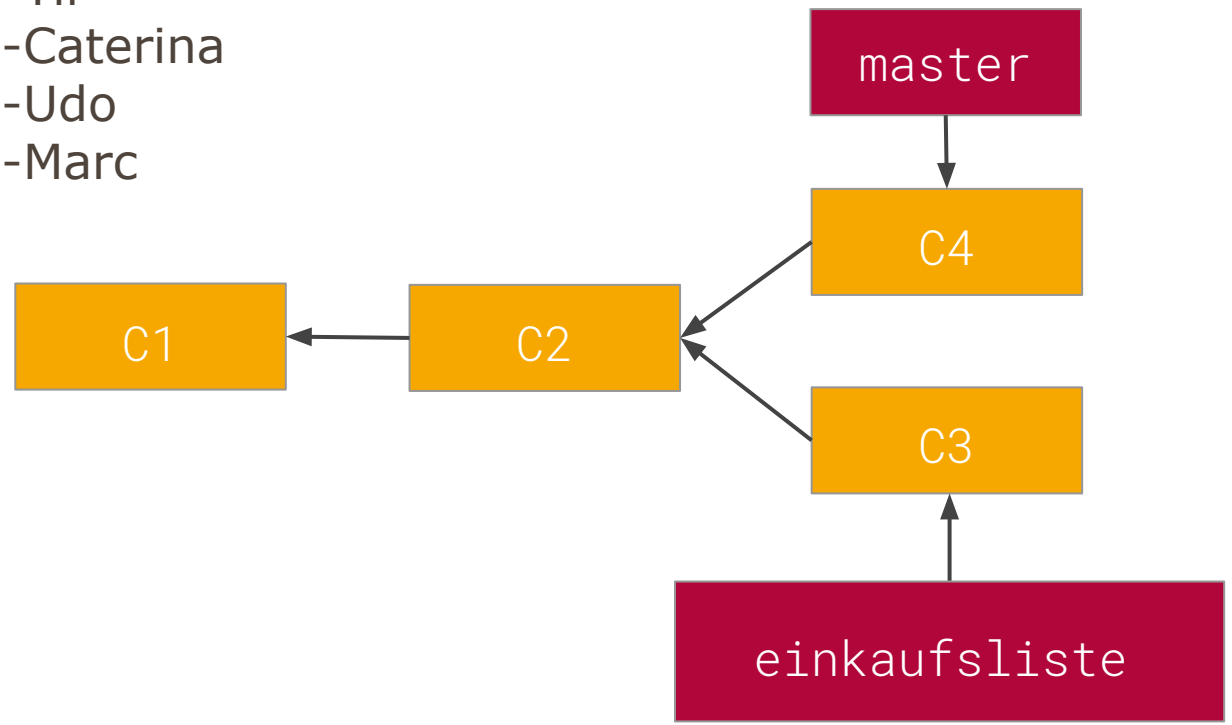
Einfache Merge Konflikte

master-Branch gästeliste.txt	einkaufliste-Branch gästeliste.txt
-Sandro -Til -Caterina -Udo	-Sandro -Til -Caterina -Udo -Marc



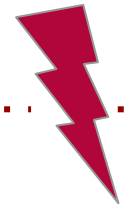
Einfache Merge Konflikte

master-Branch gästeliste.txt	einkaufliste-Branch gästeliste.txt
-Sandro -Til -Caterina -Udo -Leo	-Sandro -Til -Caterina -Udo -Marc



Einfache Merge Konflikte

master-Branch - gästeliste.txt	einkaufliste-branch - gästeliste.txt
-Sandro -Til -Caterina -Udo -Leo	-Sandro -Til -Caterina -Udo -Marc



Einfache Merge Konflikte

```
$ git merge einkaufsliste

Auto-merging gästeliste.txt
CONFLICT (content): Merge conflict in gästeliste.txt
Automatic merge failed; fix conflicts and then
commit the result.
```

- Git konnte automatisch **keinen** neuen Merge-Commit erstellen
- Prozess angehalten bis der Konflikt beseitigt ist

Konflikt anschauen

```
$ git status
```

```
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:      gästeliste.txt

no changes added to commit (use "git add" and/or
"git commit -a")
```

Konflikt anschauen

```
$ git status
```

```
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:      gästeliste.txt

no changes added to commit (use "git add" and/or
"git commit -a")
```

Konflikt anschauen

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:      gästeliste.txt

no changes added to commit (use "git add" and/or
"git commit -a")
```

Konfliktmarkierungen

- gästeliste.txt enthält einen Bereich der so aussieht:

```
-Caterina
-Udo
<<<<<<< HEAD
-Leo
=======
-Marc
>>>>>>> einkaufsliste
```

Konfliktmarkierungen

- Obere Teil (alles oberhalb von =====) ist die Version vom HEAD
- Untere Teil (alles unterhalb von =====) ist die Version vom **`einkaufsliste`**-Branch
- Konfliktlösung:
  - Für einen Teil entscheiden
  - Inhalte selbständig zusammenführen

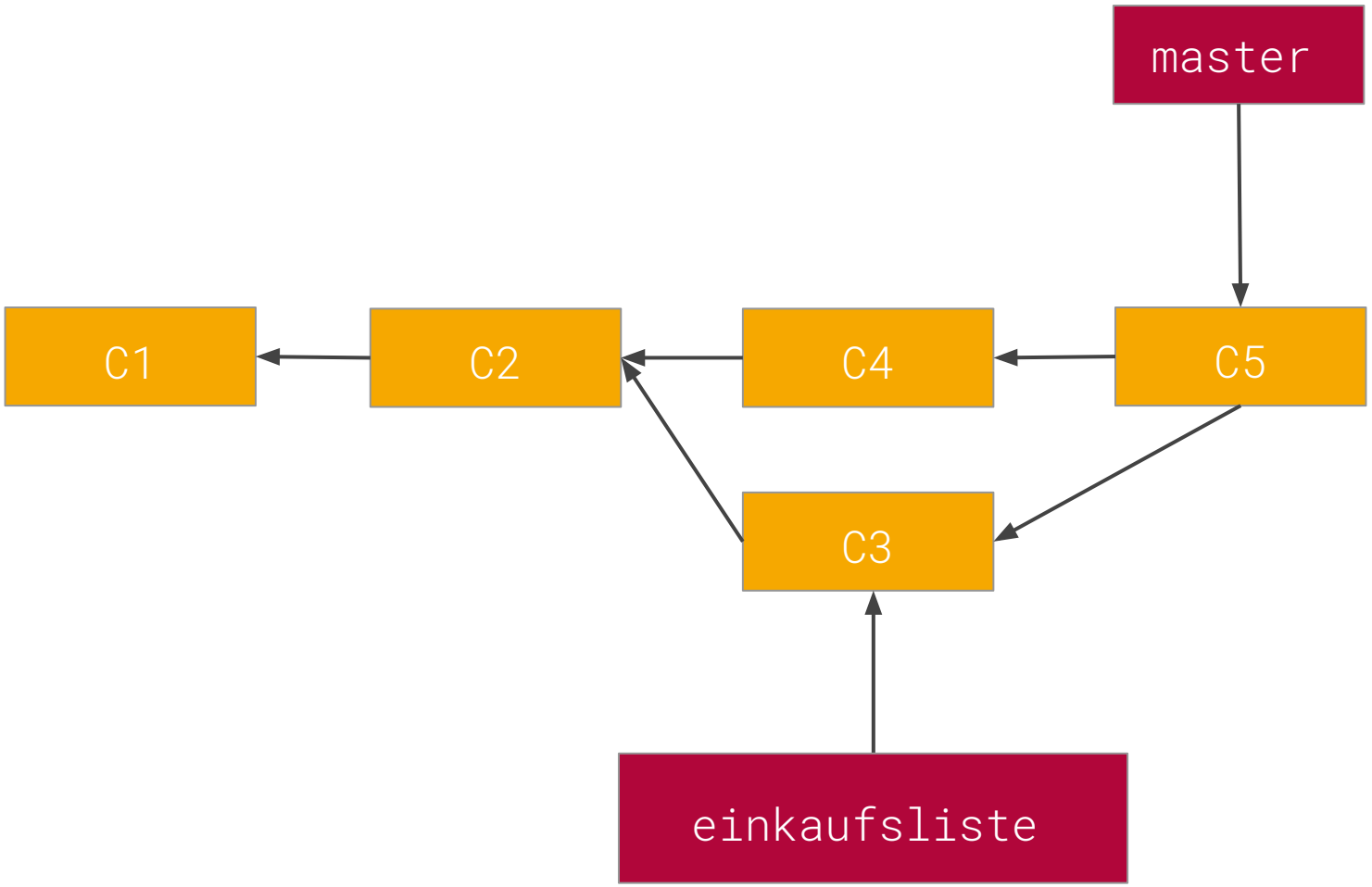
Beispiellösung

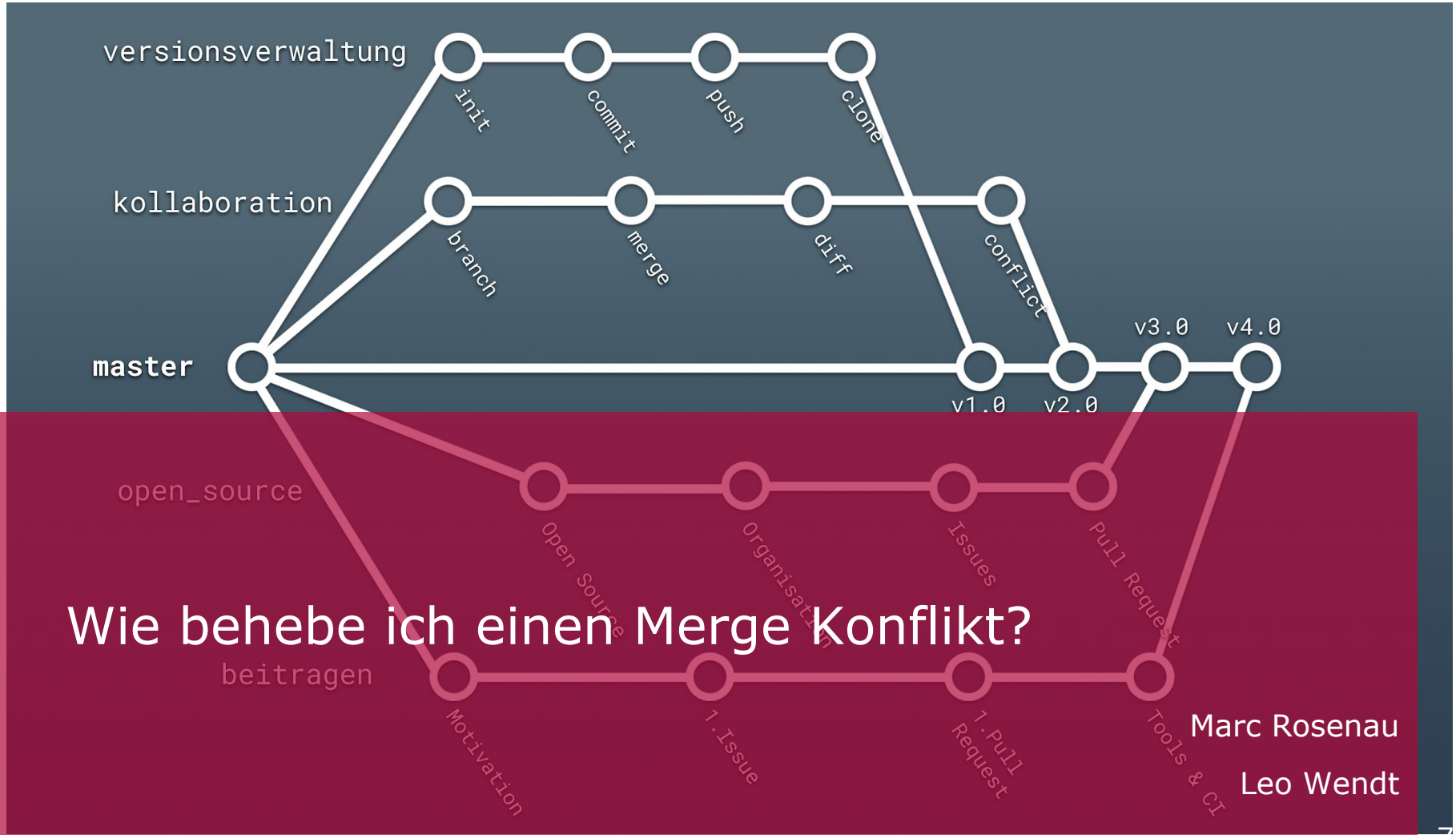
- Gesamten Block ersetzen durch:

```
-Caterina
-Udo
-Leo
-Marc
```

- Zeilen mit `<<<<<<<`, `=======`, und `>>>>>>>` vollständig entfernen

Mergen des Musiklisten-Branch

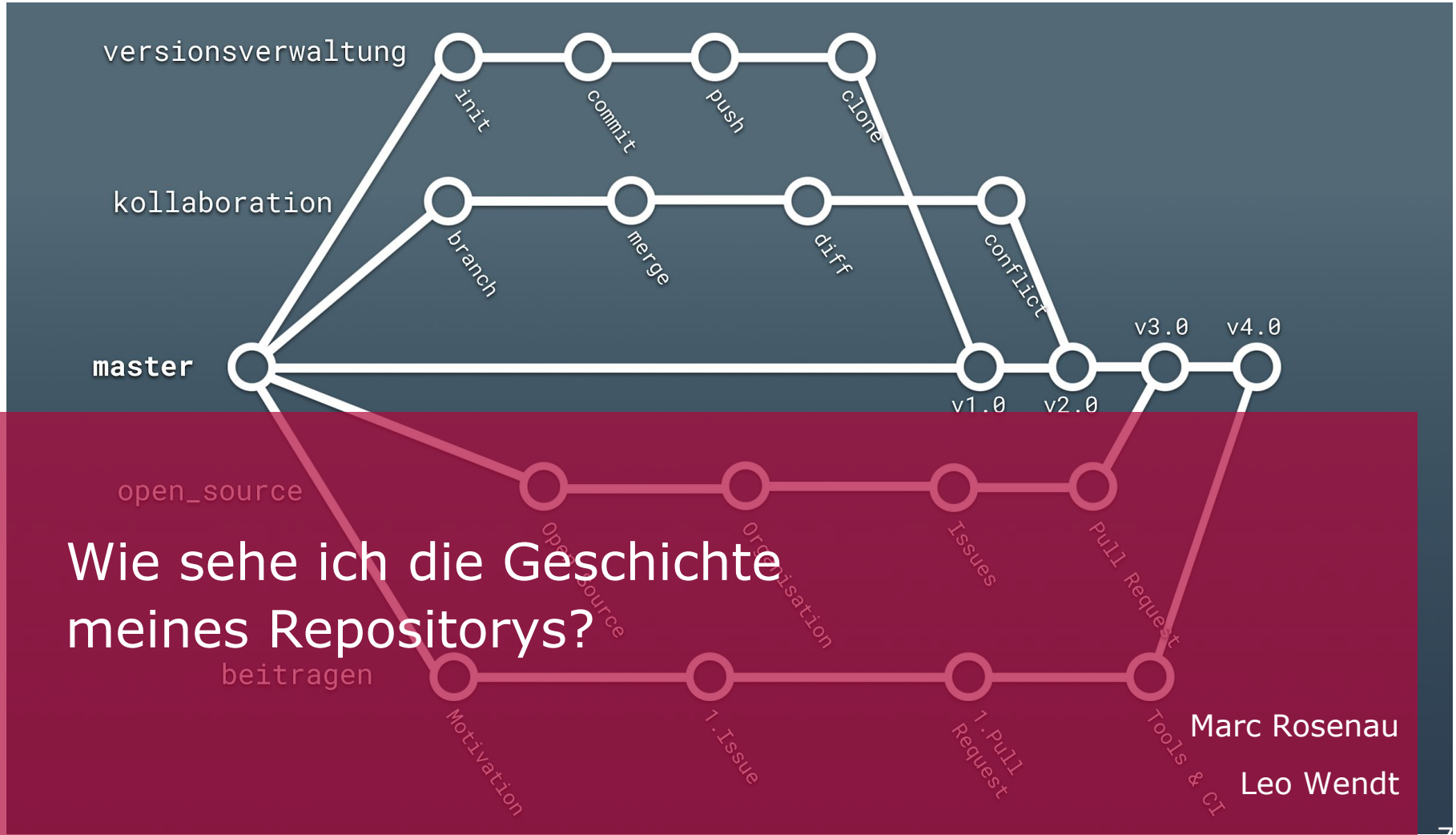




Wie behebe ich einen Merge Konflikt?

Marc Rosenau

Leo Wendt



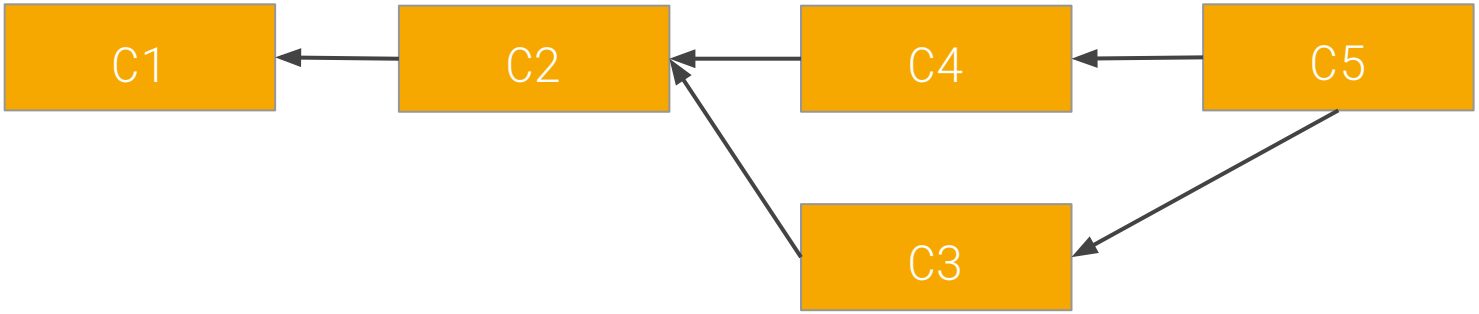
Wie sehe ich die Geschichte meines Repositorys?

Marc Rosenau

Leo Wendt

Anzeigen der Commit-Historie

- Nach Arbeiten im Repository und Erstellen von Commits, entsteht Commit-Historie
- Anschauen kann man diese mit: `$ git log`



Anzeigen der Commit-Historie

```
$ git log

commit 085bb3bcb608e1e8451d4b2432f8ecbe630
Author: Leo Wendt <leo@hpi.de>
Date:   Sat Mar 15 16:40:33 2019 -0700

    Marc in Gästeliste geschrieben

commit a11bef06a3f659402fe7563abf99ad00de2
Author: Leo Wendt <leo@hpi.de>
Date:   Sat Mar 15 10:31:28 2019 -0700

    Gästeliste angelegt
```

Prüfsumme

Commit-Nachricht

Prüfsumme

Commit-Nachricht

Git log Optionen

- `git log` hat viele Optionen um Ausgabe zu konfigurieren
- Einige hilfreiche Optionen sind
  - `-p` oder `--patch`: zeigt Unterschied (die patch-Ausgabe) an, der bei Commit eingefügt wurde
  - `-2` nur die letzten beiden Einträge, entsprechend `-<Zahl>` die letzten **<Zahl>** Einträge
  - `--stat`: eine kurze Statistik zu den Commits
  - `--pretty`: die Commits anders formatiert anzeigen lassen

Cheat Sheet

Erstellen von Branches

```
$ git branch <name>
```

Wechseln von Branches

```
$ git checkout <branch>
```

Zusammenführen von Branches

```
$ git merge <branch>
```

Commit Historie anzeigen

```
$ git log
```

Unstaging von Datei

```
$ git reset HEAD <datei>
```

Änderungen an Datei verwerfen

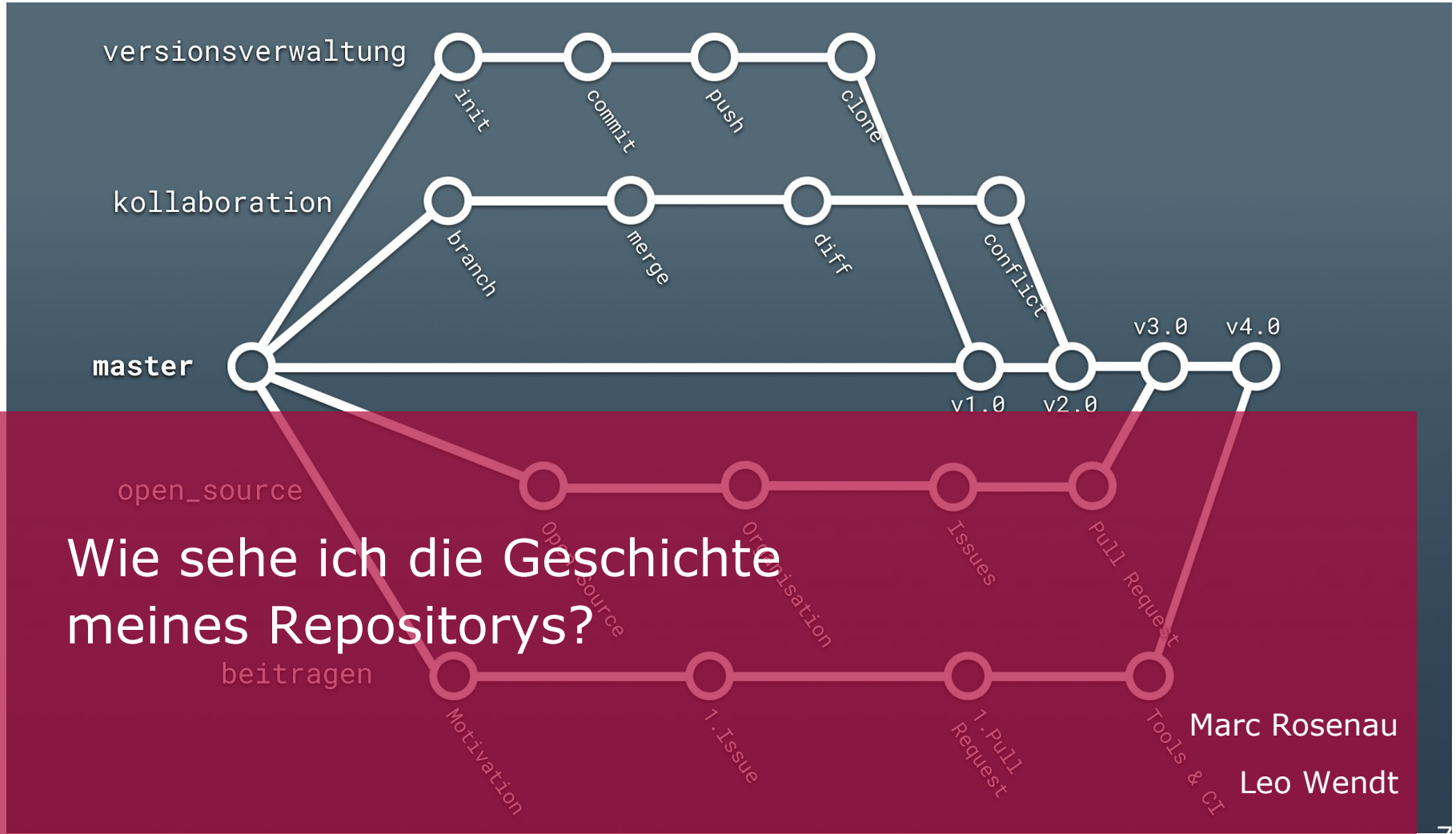
```
$ git checkout -- <datei>
```

Letzten Commit verändern

```
$ git commit --amend
```

Zu Commit zurückkehren

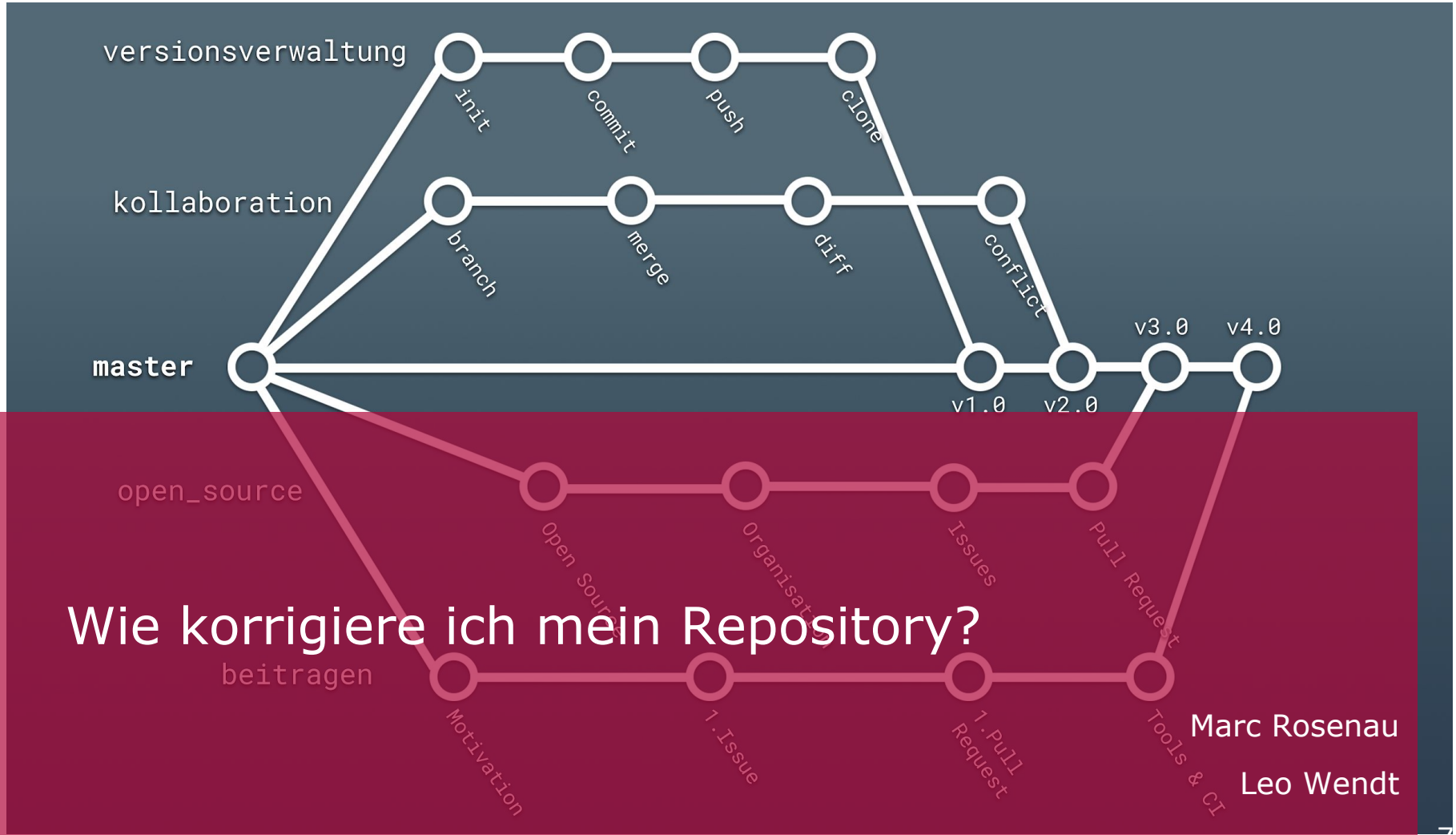
```
$ git reset <commit>
```



Wie sehe ich die Geschichte meines Repositorys?

Marc Rosenau

Leo Wendt



Wie korrigiere ich mein Repository?

Marc Rosenau

Leo Wendt

Den letzten Commit verändern

- Gästeliste geschrieben

```
$ git commit -m ‘Gästeliste hinzufegü’
```

- Fehler in Commit Message!

Den letzten Commit verändern

```
$ git commit --amend
```

- führt “normalen” Commit aus
- “Neuer” Commit ersetzt “alten”
- Wenn nichts verändert wurden, wird mit dem Befehl nur Commit-Nachricht bearbeitet

Den letzten Commit verändern

- Gästeliste geschrieben

```
> $ git commit -m ‘Gästeliste hinzufegü’
```

- Fehler in Commit Message!

```
> $ git commit --amend -m ‘Gästeliste hinzugefügt’
```

Unstaging einer gestageten Datei

- Spieleliste verändert
- Materialliste verändert
- möchten Änderungen in 2 Commits committen

- Wie kann eine der beiden aus Staging Area entfernen werden?

Unstaging einer gestageten Datei

```
$ git status
```

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   materialliste.txt
                spieleliste.txt
```

Unstaging einer gestageten Datei

```
$ git status

On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

      modified:   materialliste.txt
                  spieleliste.txt
```

Unstaging mit git reset

```
$ git reset HEAD materialliste.txt
```

```
Unstaged changes after reset:
M   materialliste.txt
```

Unstaging mit git reset

```
$ git status
```

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
    modified:    spieleliste.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)
    modified:   materialliste.txt
```

Unstaging mit git reset

```
$ git status


On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
    modified:    spieleliste.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)
    modified:   materialliste.txt
```

Änderungen an Datei verwerfen

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)
    modified:   materialliste.txt
```

Git checkout auf Datei

```
$ git checkout -- materialliste.txt
```

```
$ git status
```

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
     modified:    spieleliste.txt
```

Git checkout auf Datei

- **Achtung**: `git checkout -- <file>` ist riskant
- Alle lokalen Änderungen verschwinden für immer!
- Alles, was committed wurde, kann in Git wiederhergestellt werden

Cheat Sheet

Erstellen von Branches

```
$ git branch <name>
```

Wechseln von Branches

```
$ git checkout <branch>
```

Zusammenführen von Branches

```
$ git merge <branch>
```

Commit Historie anzeigen

```
$ git log
```

Unstaging von Datei

```
$ git reset HEAD <datei>
```

Änderungen an Datei verwerfen

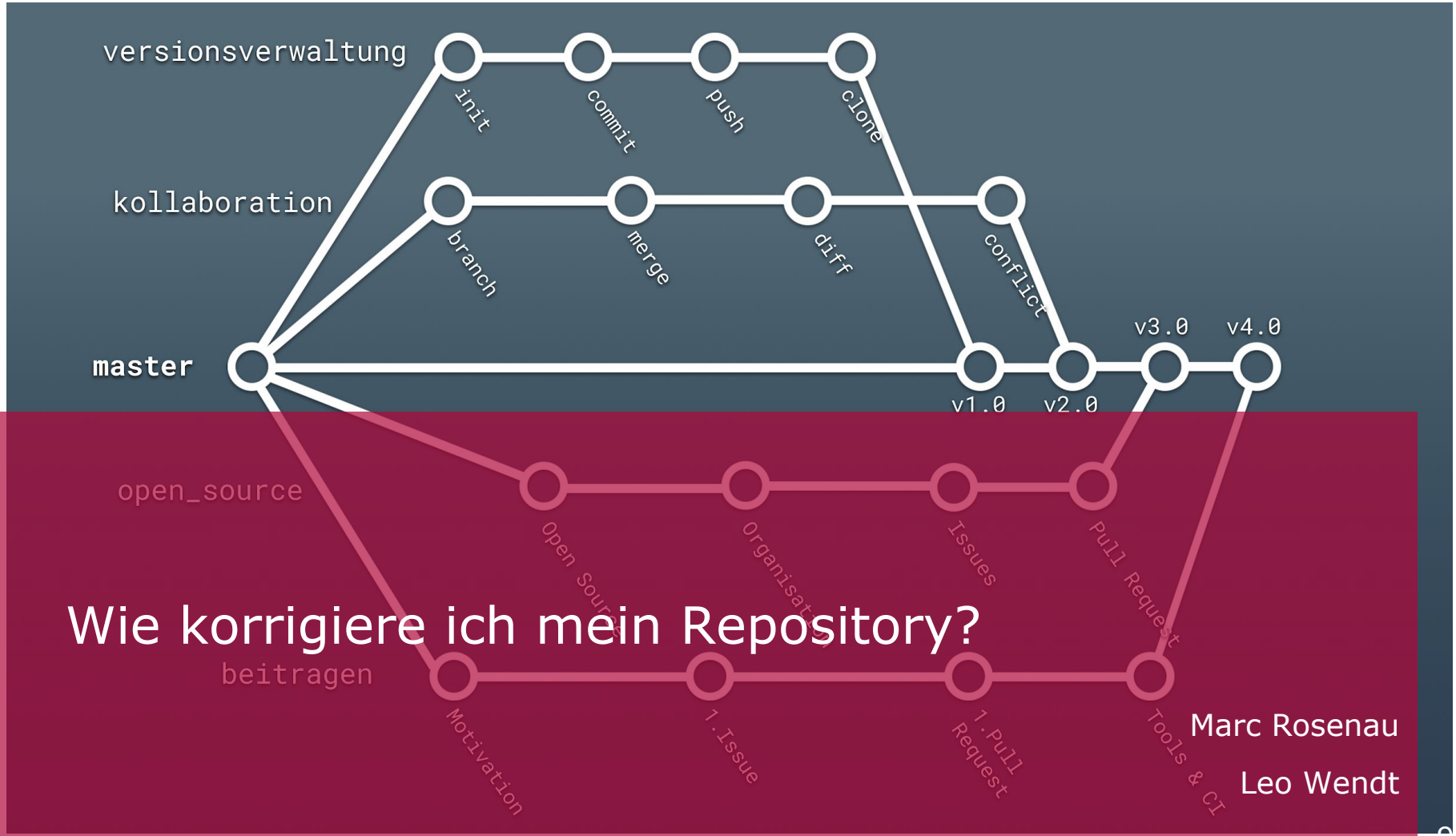
```
$ git checkout -- <datei>
```

Letzten Commit verändern

```
$ git commit --amend
```

Zu Commit zurückkehren

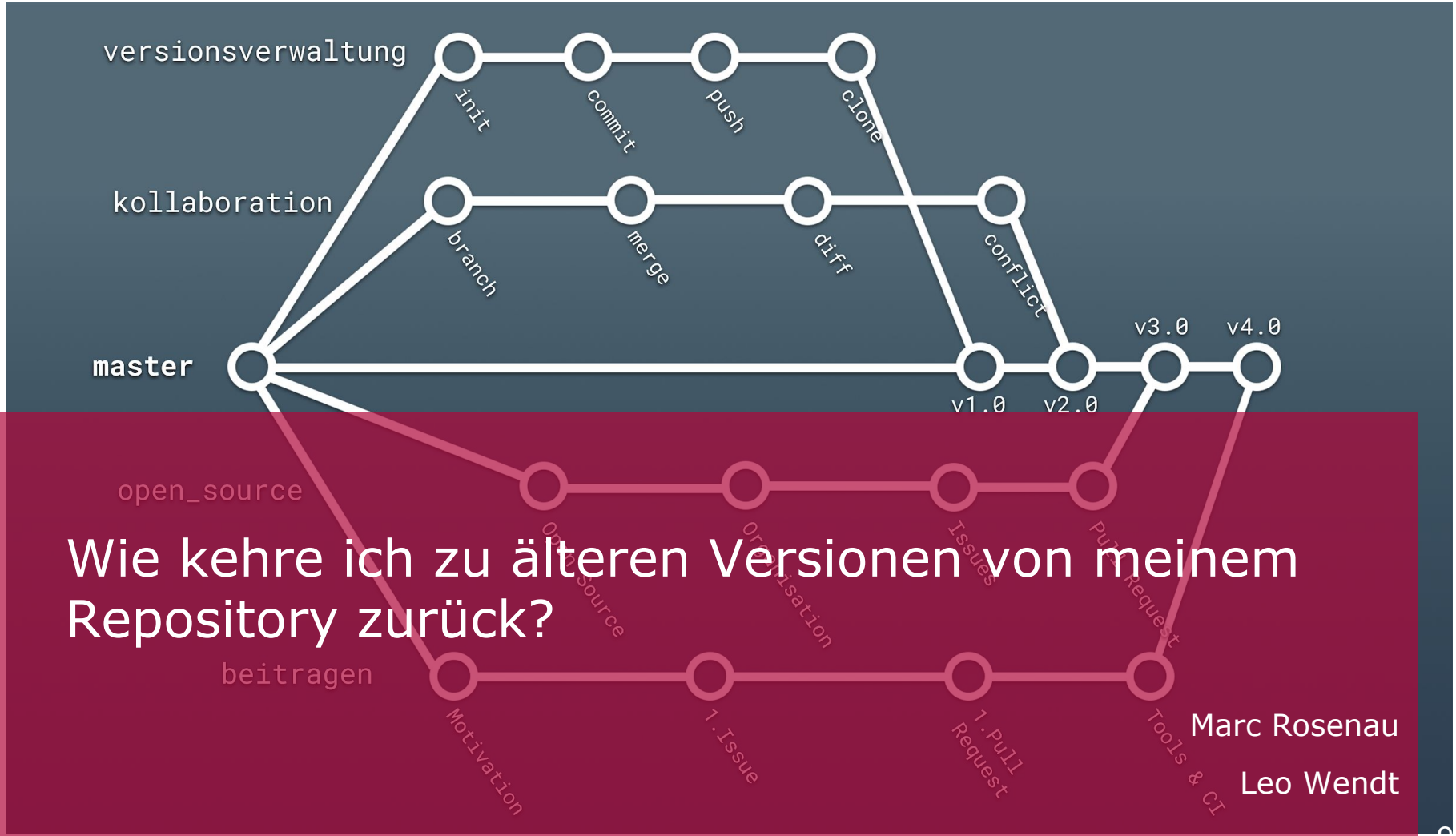
```
$ git reset <commit>
```



Wie korrigiere ich mein Repository?

Marc Rosenau

Leo Wendt

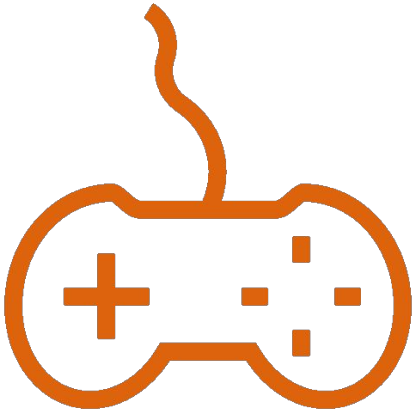


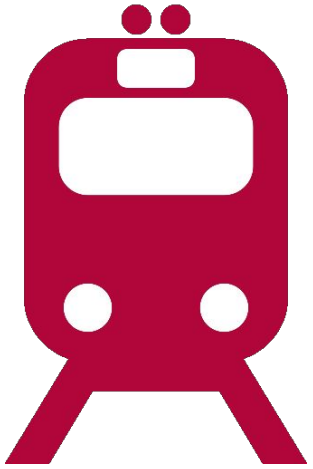
Wie kehre ich zu älteren Versionen von meinem Repository zurück?

Marc Rosenau

Leo Wendt

Story





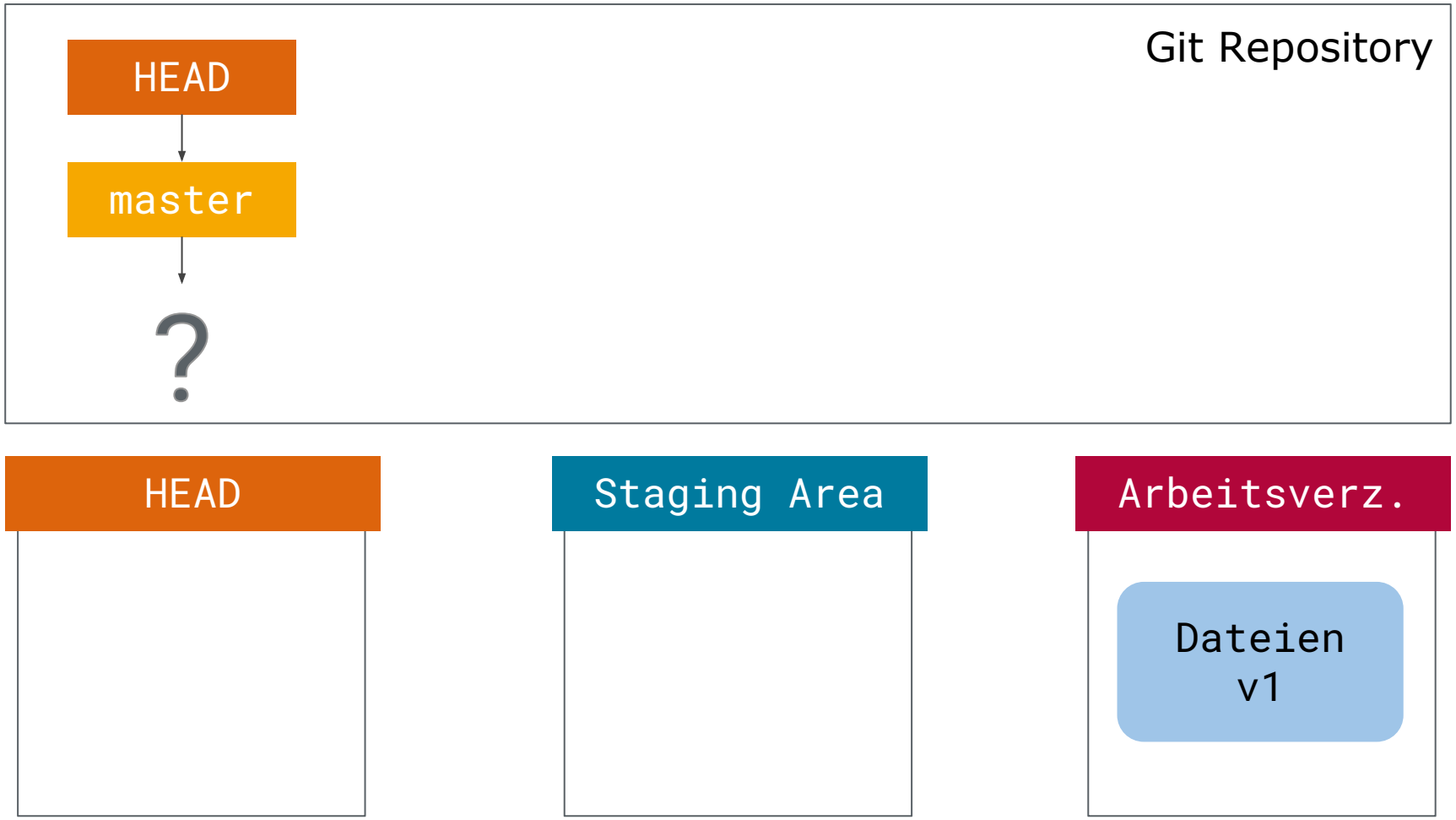
Was macht git reset auf Commits?

- git reset lässt sich unterschiedlich ausführen
- dadurch verschiedene Auswirkungen

```
$ git reset HEAD~
```

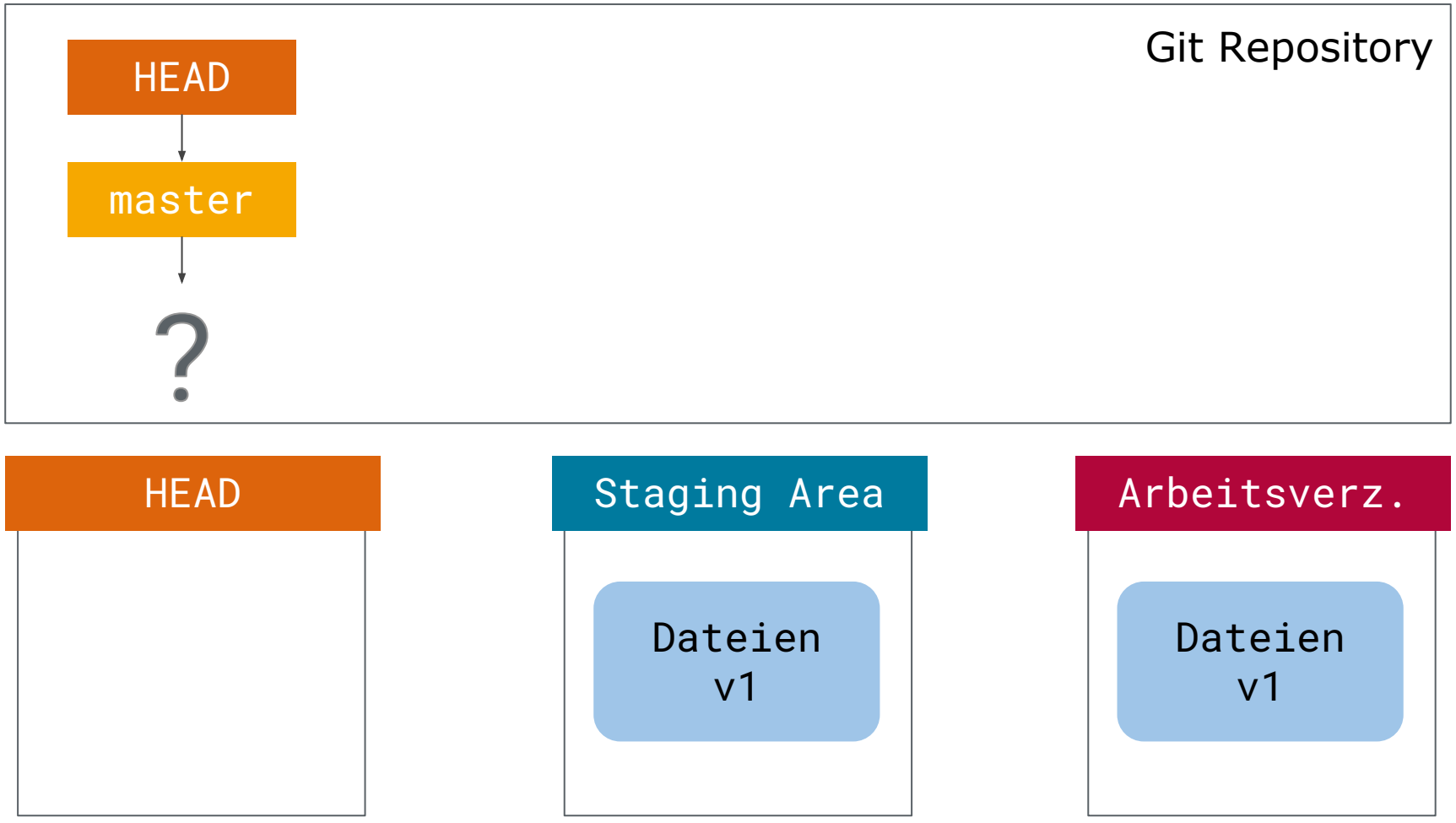
Baum	Rolle
HEAD	Schnappschuss des letzten commit
Staging Area	Dateien die committed werden sollen
Arbeitsverzeichnis	lokale Dateien

Arbeiten mit dem Repository

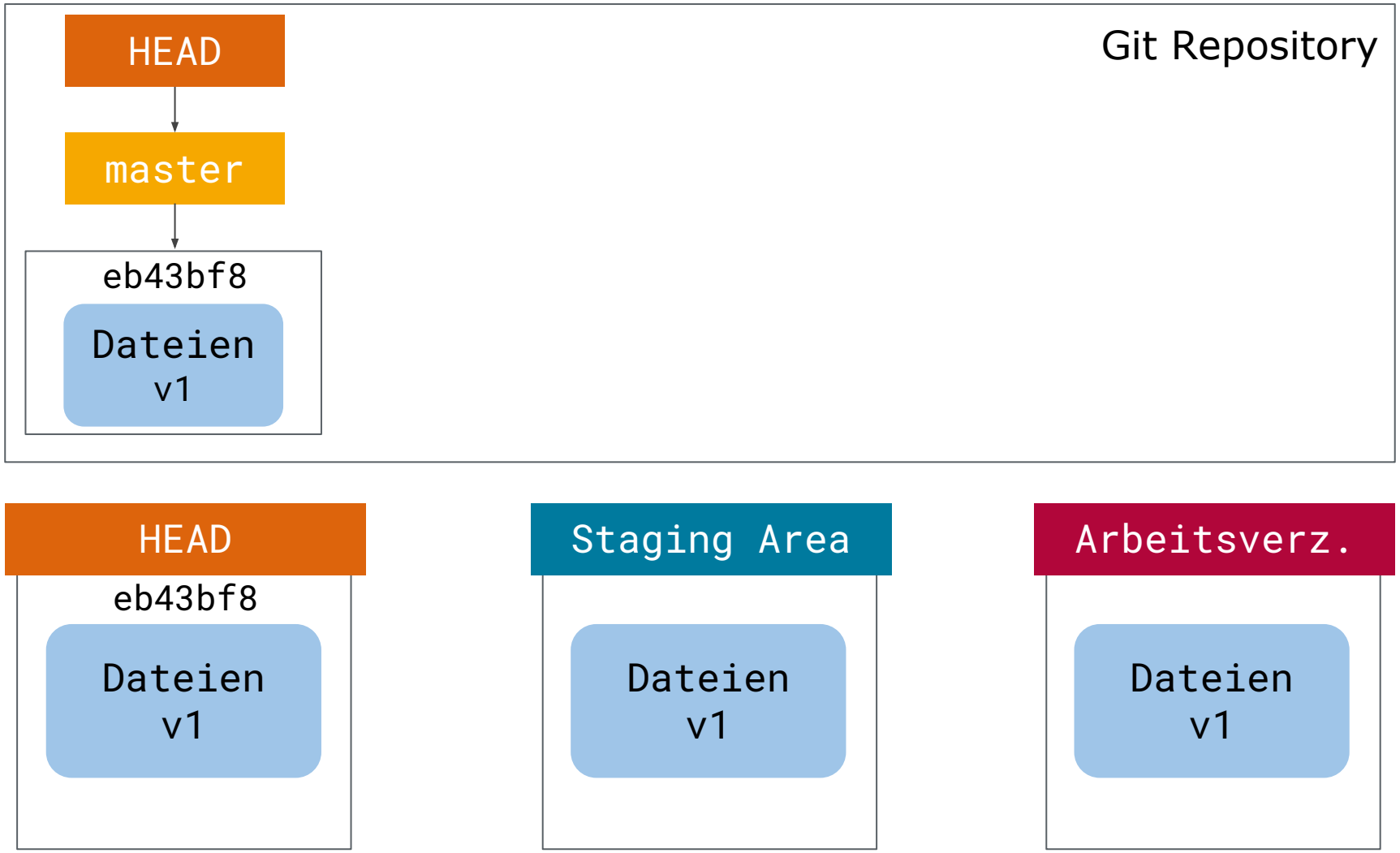


An diesem Punkt hat nur das Arbeitsverzeichnis unsere Datei.

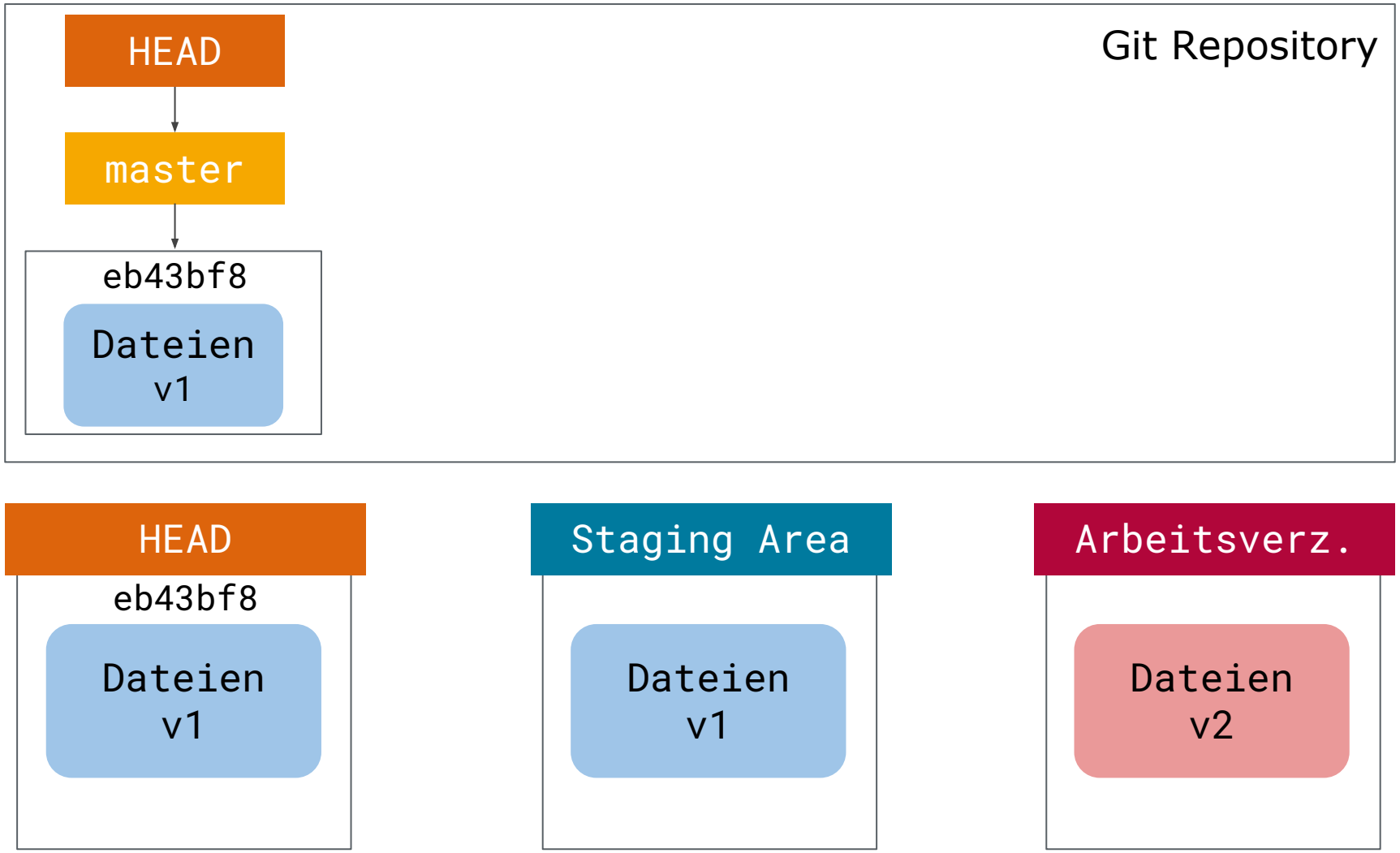
Arbeiten mit dem Repository



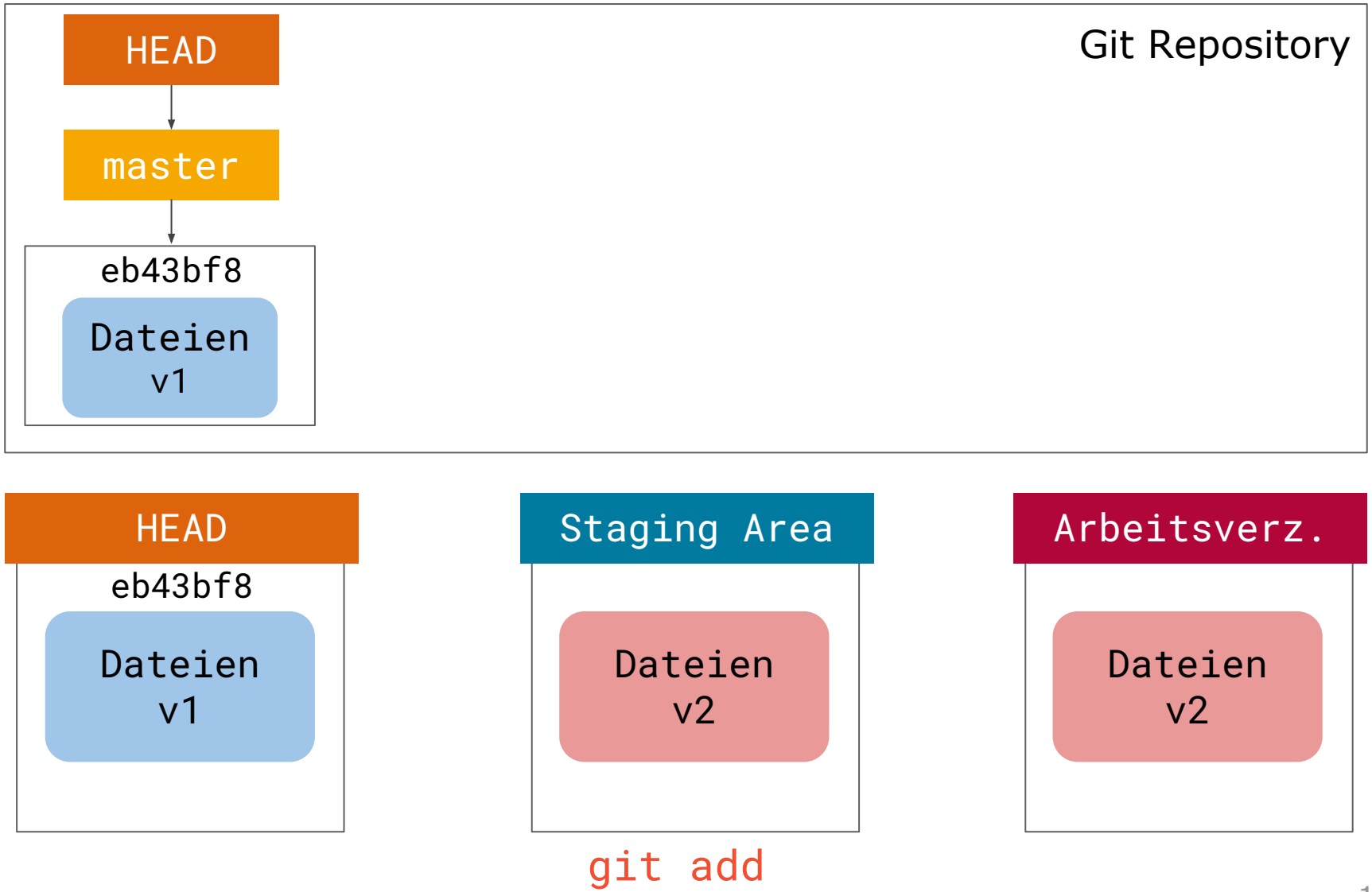
Arbeiten mit dem Repository



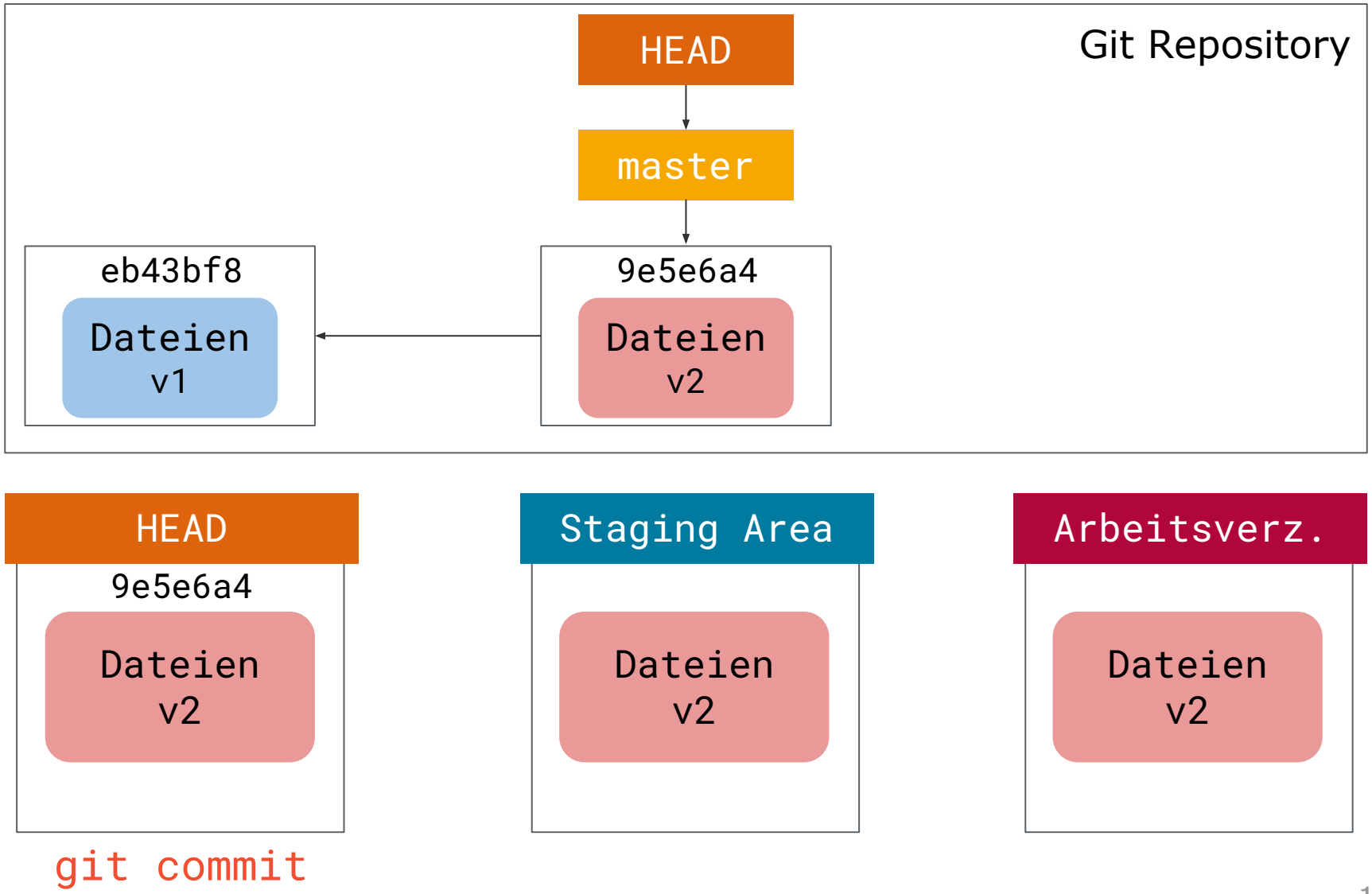
Arbeiten mit dem Repository



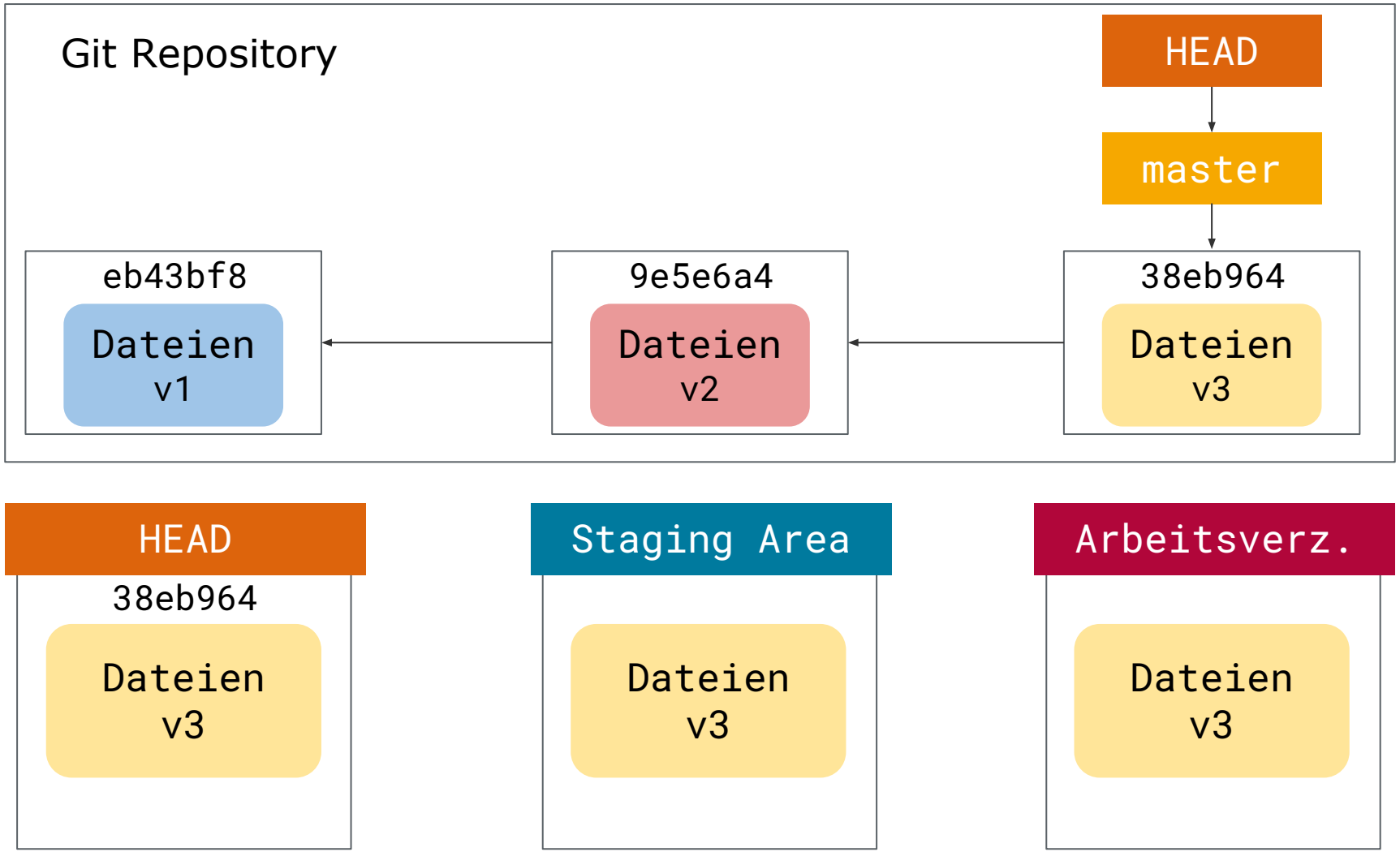
Arbeiten mit dem Repository



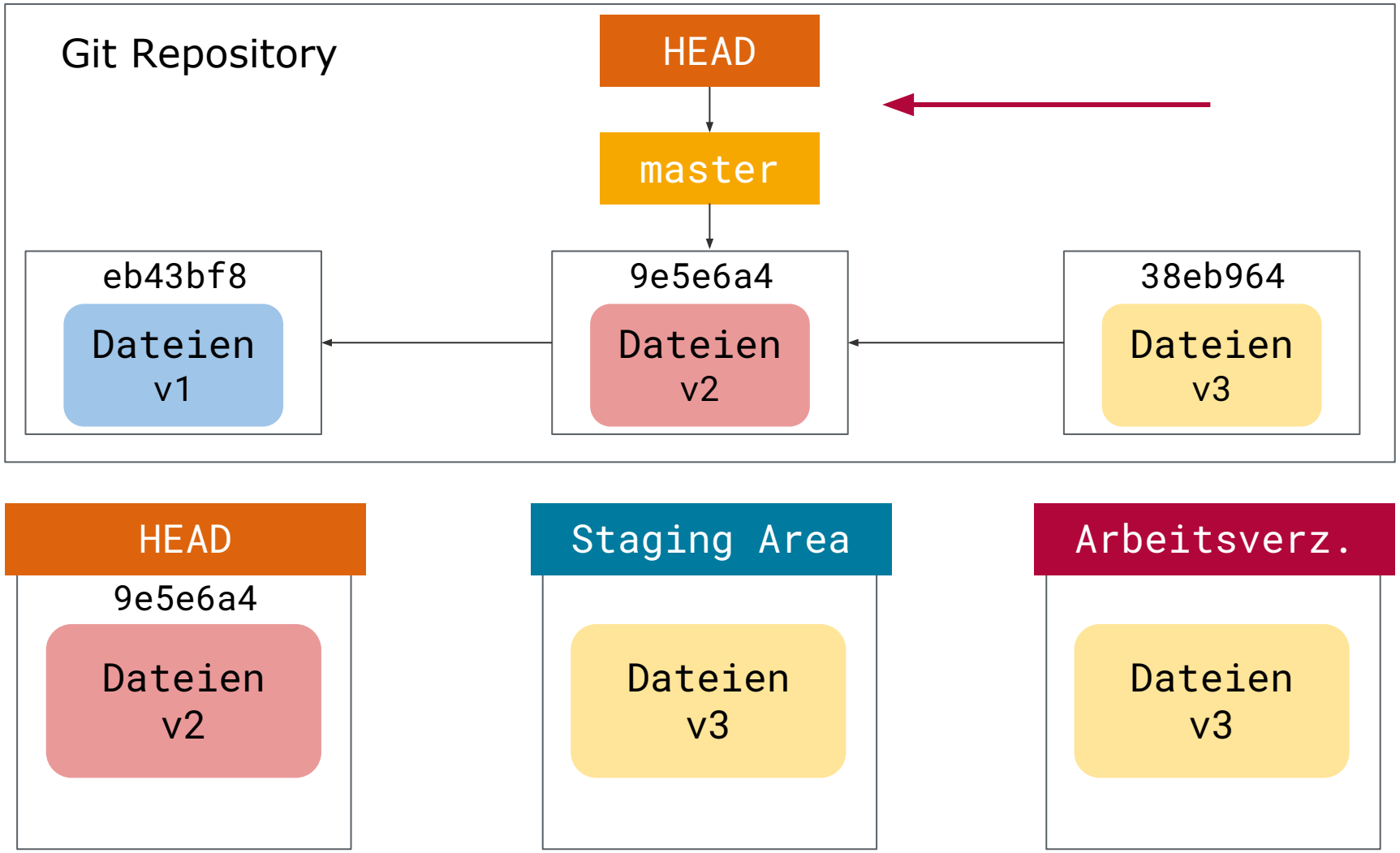
Arbeiten mit dem Repository



Wie funktioniert Reset?

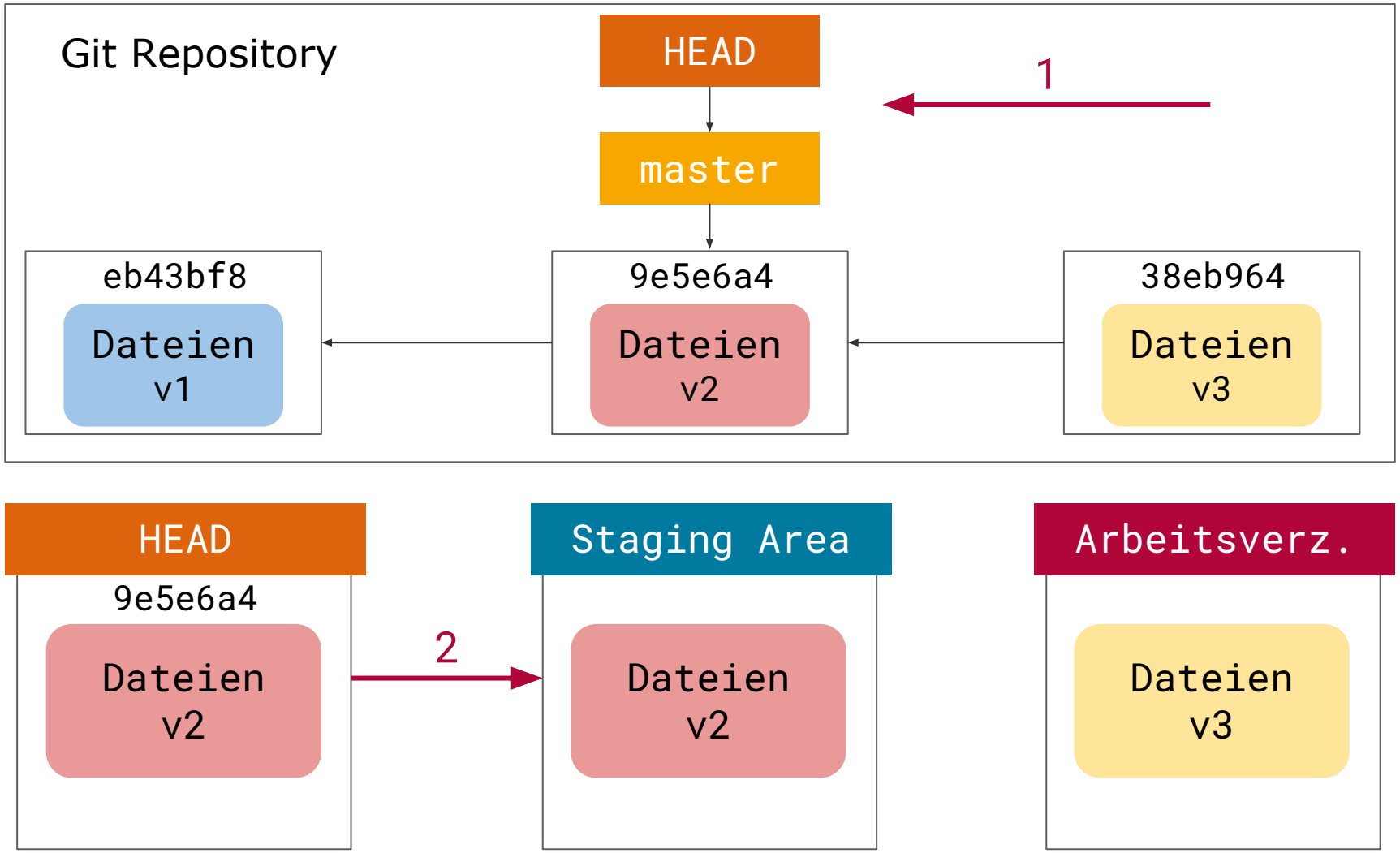


Schritt 1 - HEAD bewegen



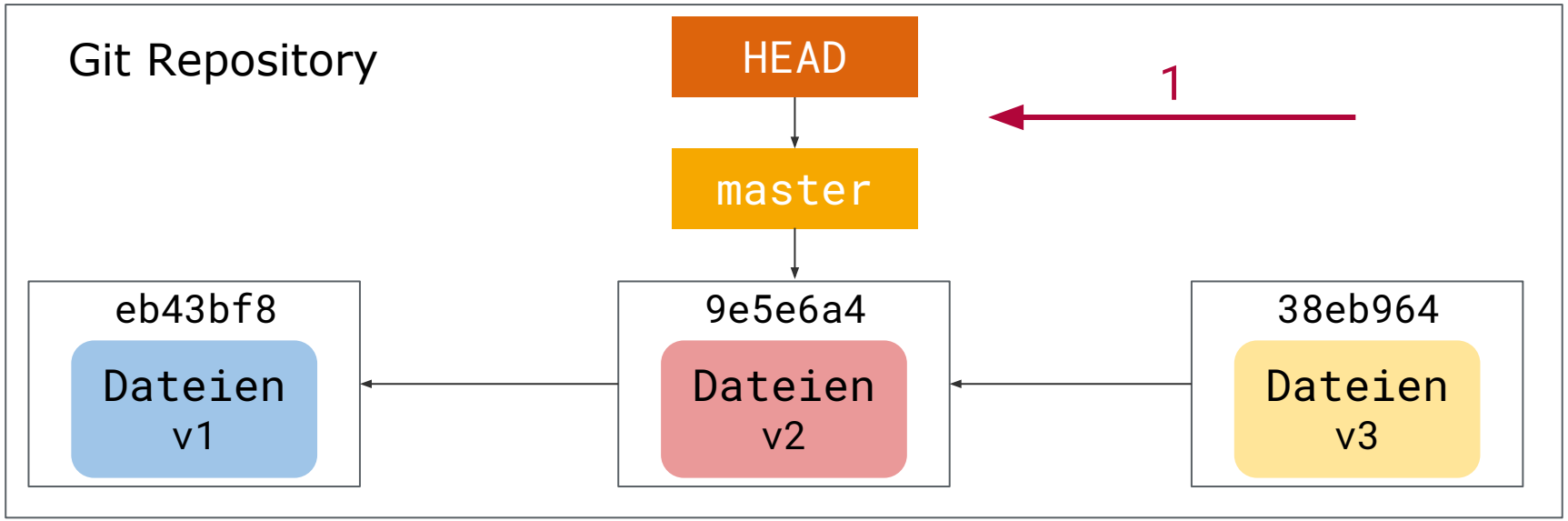

```
git reset --soft HEAD~
```

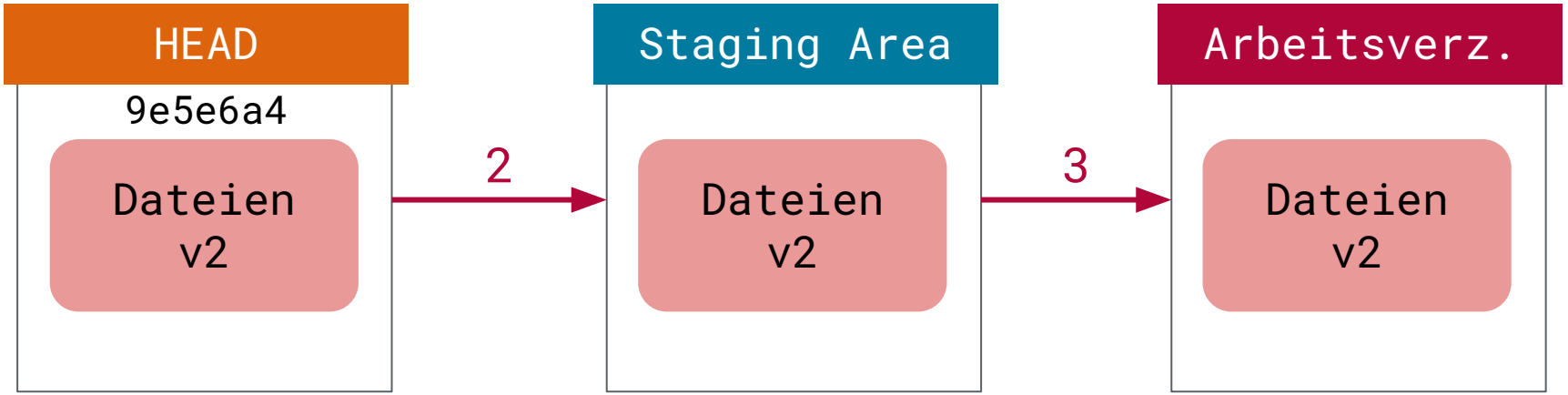
Schritt 2 - Staging Area updaten



git reset --mixed HEAD~ oder git reset HEAD~

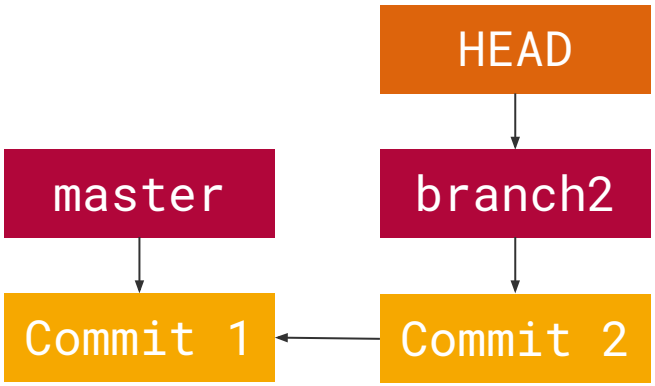
Schritt 3 - das Arbeitsverzeichnis updaten



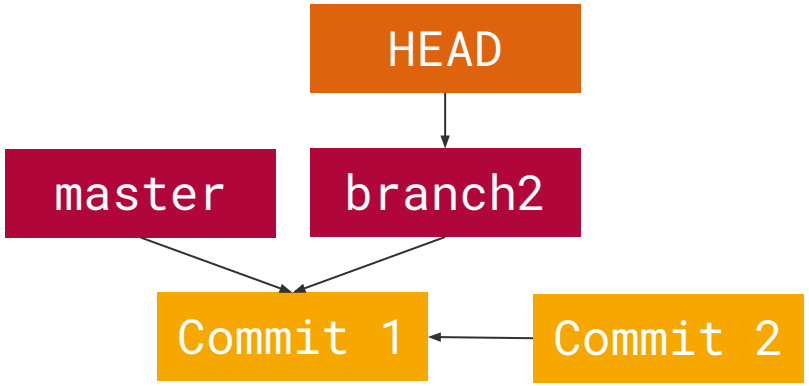



```
git reset --hard HEAD~
```

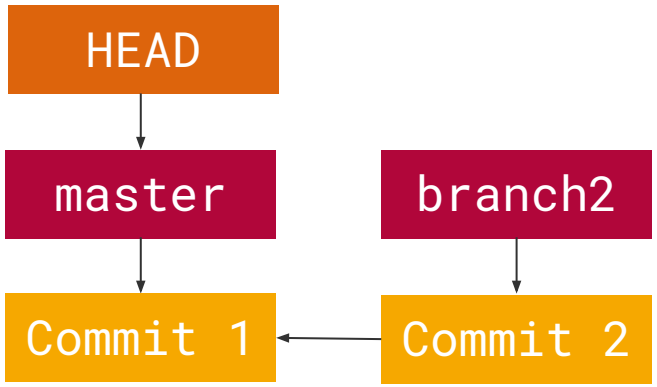
Reset vs. Checkout



Vor git Befehl



nach reset



nach checkout

Cheat Sheet

	Was bewegt sich?	**Staging Area**	**Arbeitsverz.**	**Sicher?**
`reset --soft`	Branch	Nein	Nein	Ja
`reset --mixed`	Branch	Ja	Nein	Ja
`reset --hard`	Branch	Ja	Ja	**Nein**

Cheat Sheet

Erstellen von Branches

```
$ git branch <name>
```

Wechseln von Branches

```
$ git checkout <branch>
```

Zusammenführen von Branches

```
$ git merge <branch>
```

Commit Historie anzeigen

```
$ git log
```

Unstaging von Datei

```
$ git reset HEAD <datei>
```

Änderungen an Datei verwerfen

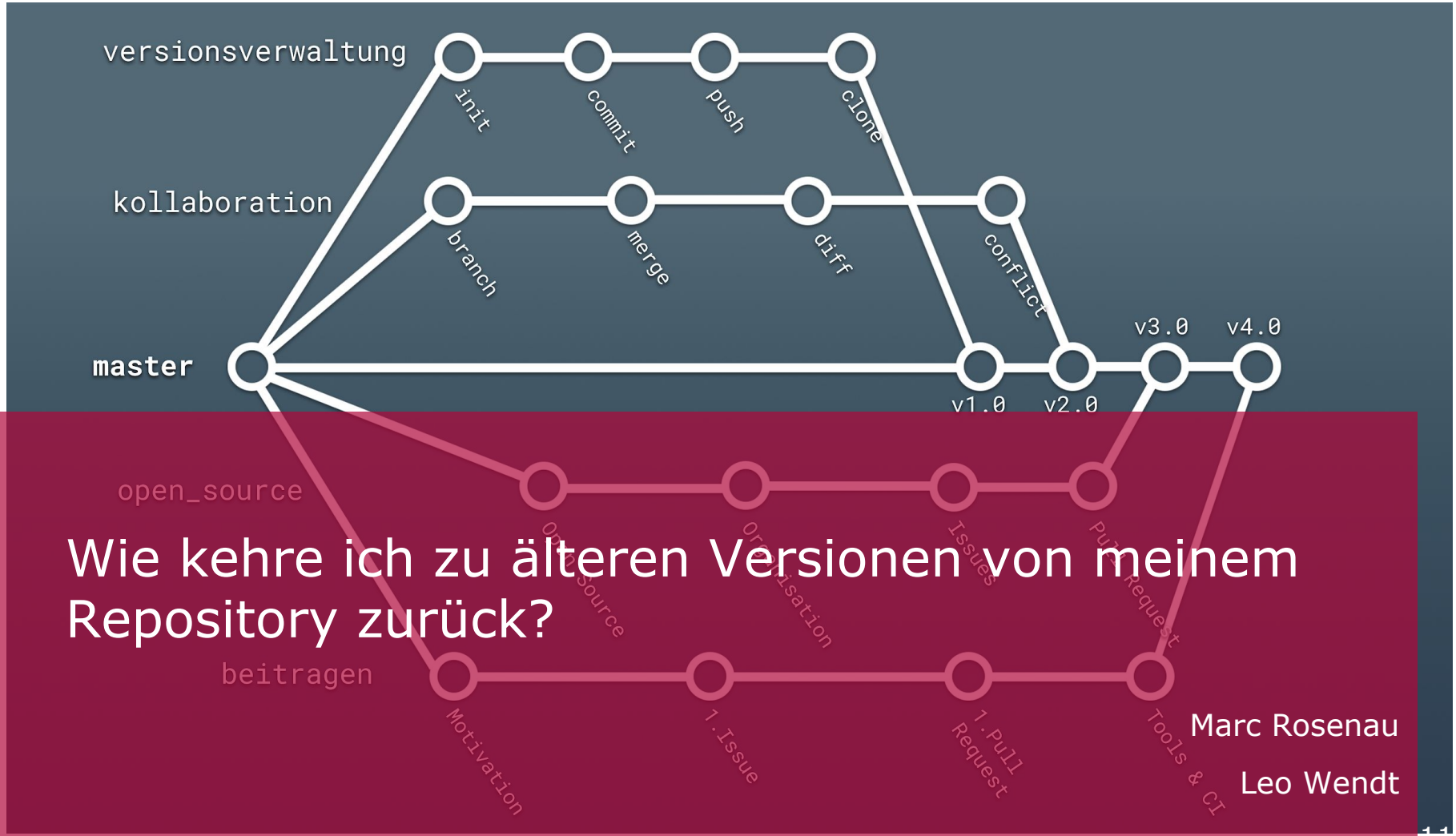
```
$ git checkout -- <datei>
```

Letzten Commit verändern

```
$ git commit --amend
```

Zu Commit zurückkehren

```
$ git reset <commit>
```



Wie kehre ich zu älteren Versionen von meinem Repository zurück?

Marc Rosenau

Leo Wendt