integrata cegos

Git

Scratch zum Webinar vom 9.-10.1.2020

Informationen

- Referent
  - Rainer Sawitzki
  - training@rainer-sawitzki.de
- 4 Sessions, jeweils 2,5 Stunden
  - Inklusive 15 Minuten Pause
- Ablauf
  - Vortrag
    - Etwa 45 Minuten
  - Präsentation
    - Etwa 45 Minuten
  - Zur Auflockerung kleinere Übungseinheiten
    - Jede 5 – 10 Minuten
  - Fragerunde
    - Nach Bedarf
    - Am Ende, etwa 15 Minuten

Ressourcen

- Dieser Scratch
- Die Präsentation
- Online
  - https://github.com/Javacream/org.javacream.training.gitscm
    - Ein paar Beispiele und Skripte auf GitHub
  - https://git-scm.com/download/win
    - Download Git Portable
  - https://git-scm.com/docs
    - Dokumentation

Git Repositories

- Normales Verzeichnis
- + .git => Git Project Directory
- .git
  - Das Git Repository
- Alles andere: Workspace
- Git Repository selbst:
  - Staging Area
  - Internes Repository
  - Stashes

Git Begriffe

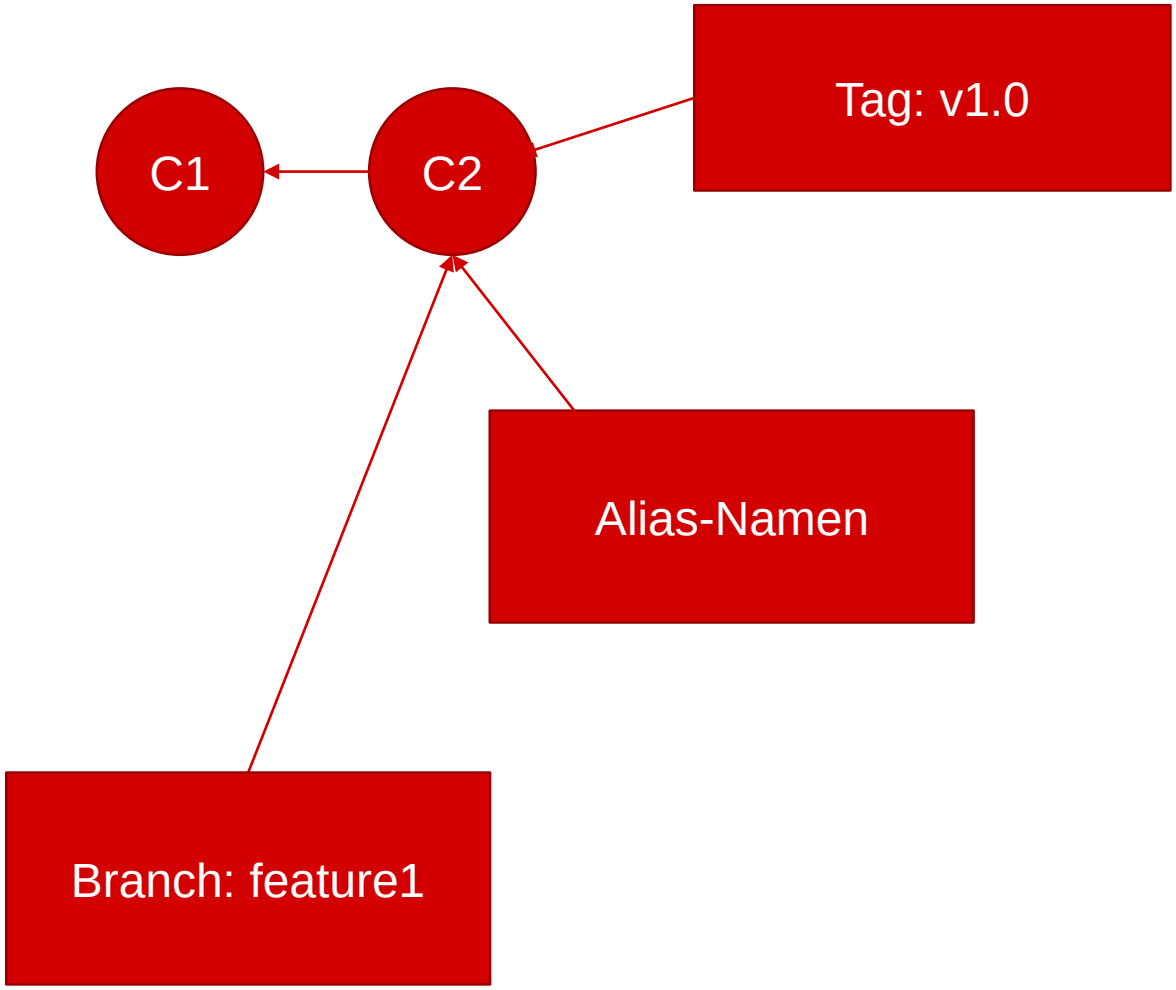
- Content-Objekte
  - Dateien
    - Tree, +++
- Commit-Objekte
  - Liste aller Content-Objekte des aktuellen Standes
  - Autor, Timestamp, Commit-Message

Befehle

- git
  - config –global
    - User, eMail
  - init
  - add
    - Workspace – Staging
  - Commit
    - -m <Commit-Message>
    - Commit-Objekt wird erstellt
    - Und über Hash identifizierbar gemacht
  - status
    - Aktuellen Stand und Diskrepanzen
  - log
    - Aktuelle Commit-Objekte
      - Hash
      - Autor
      - Date
      - Commit-Message

Commits

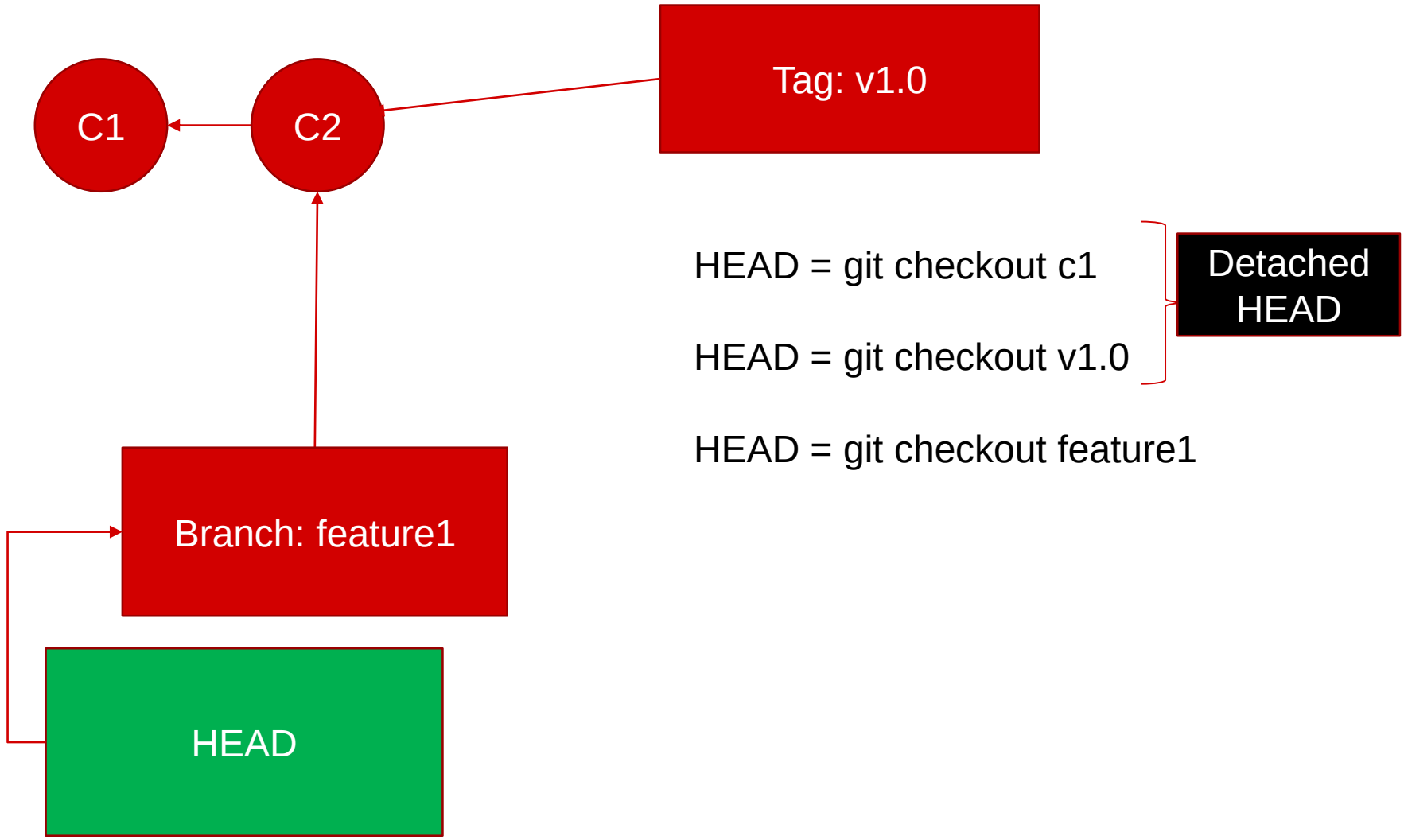
integrata cegos



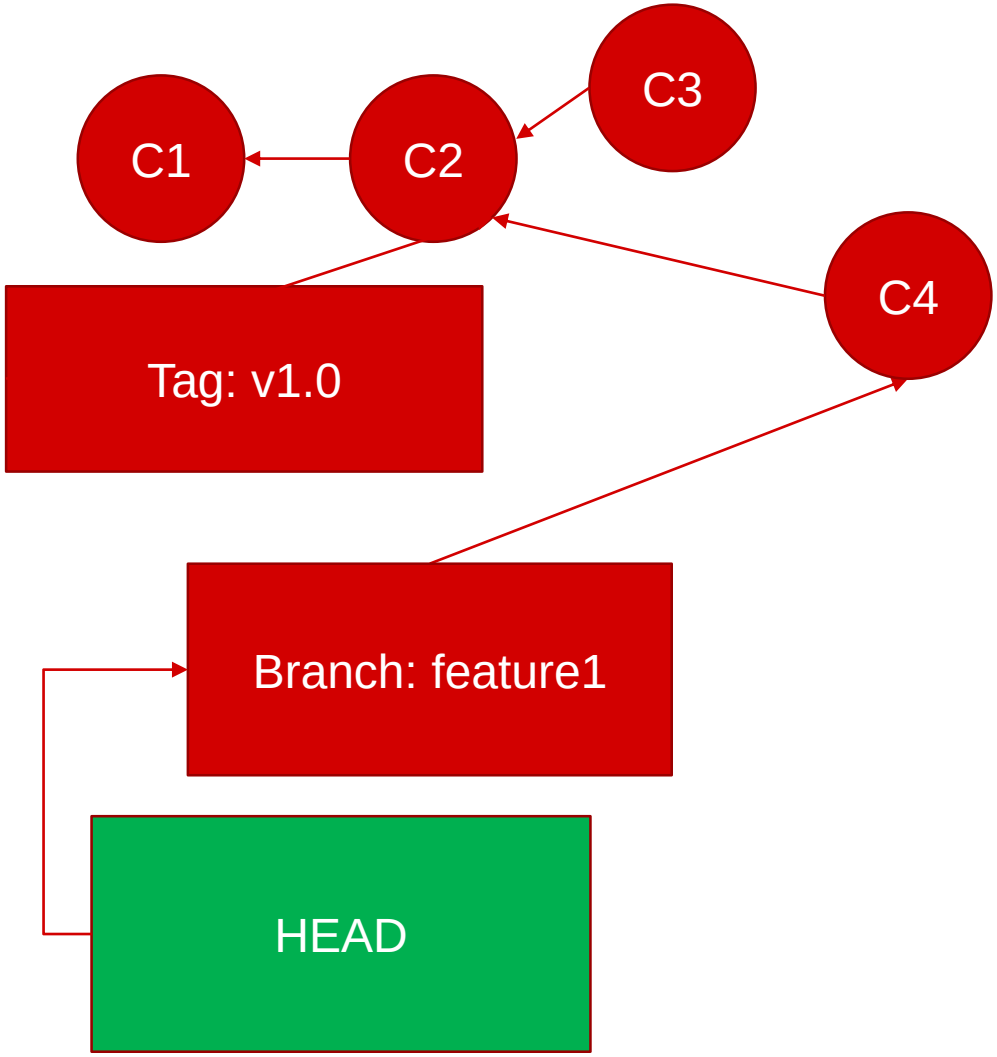
2 Arten von Alias

- Unverrückbare, feste Projektstände
  - Substantive
  - Versionsnamen, v.1.0
  - Tag
    - Git checkout c2
    - Git tag v1.0
    - Git tag –list
    - Git checkout v1.0
- Aktuell laufende Aktionen
  - Verb
  - ToDos, Jira-Ticket
  - Branch
    - Git checkout c1
    - Git branch feature1
    - Git checkout feature1
    - Git branch --list

Commits und checkout



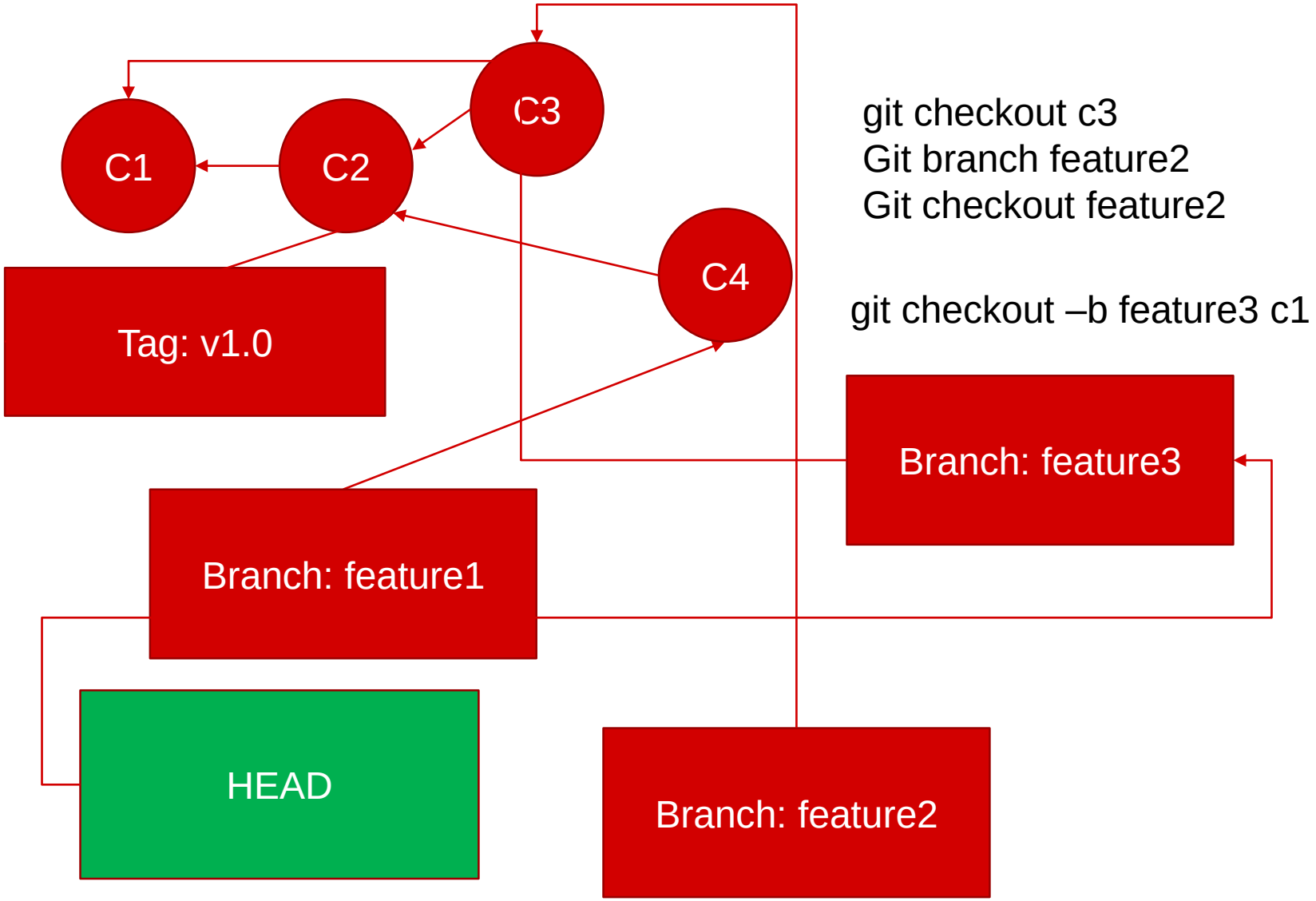
Commits und commit



git checkout v1.0
//Changes
Add + commit

git checkout feature1
//Changes
Add + commit

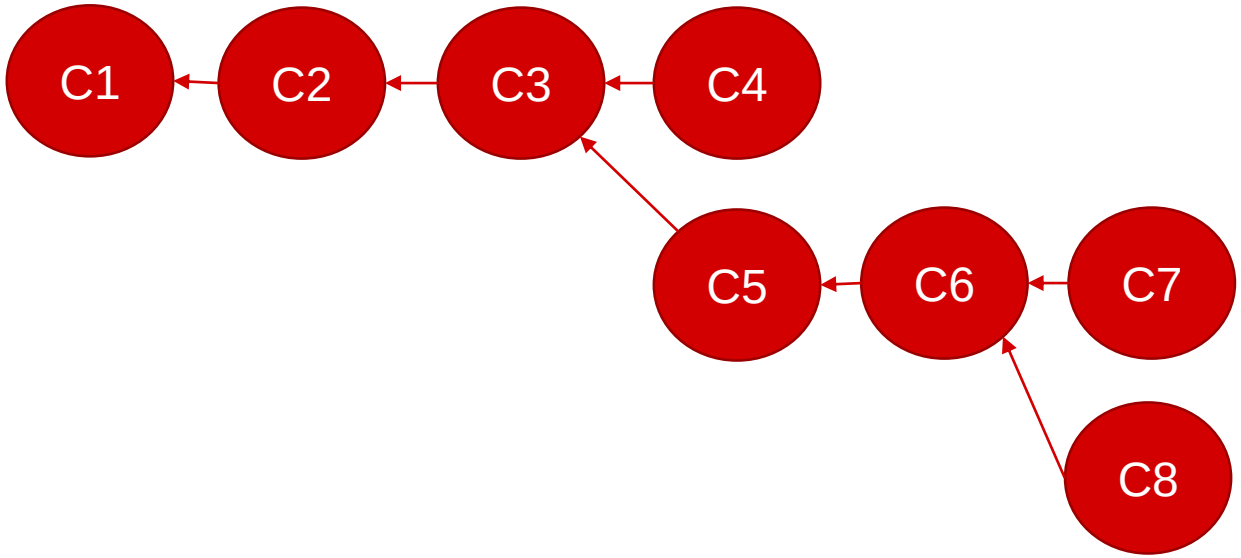
Commits und commit



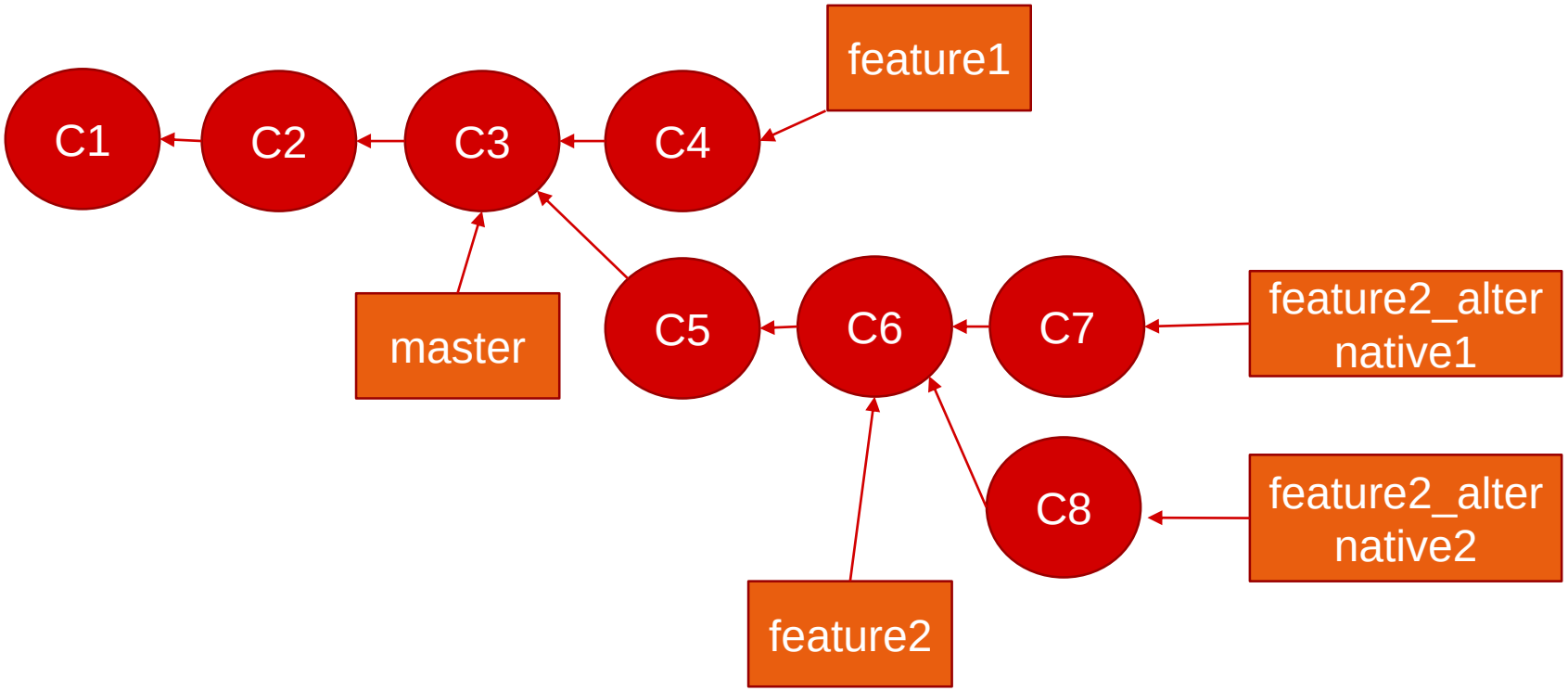
git checkout c3
Git branch feature2
Git checkout feature2

git checkout –b feature3 c1

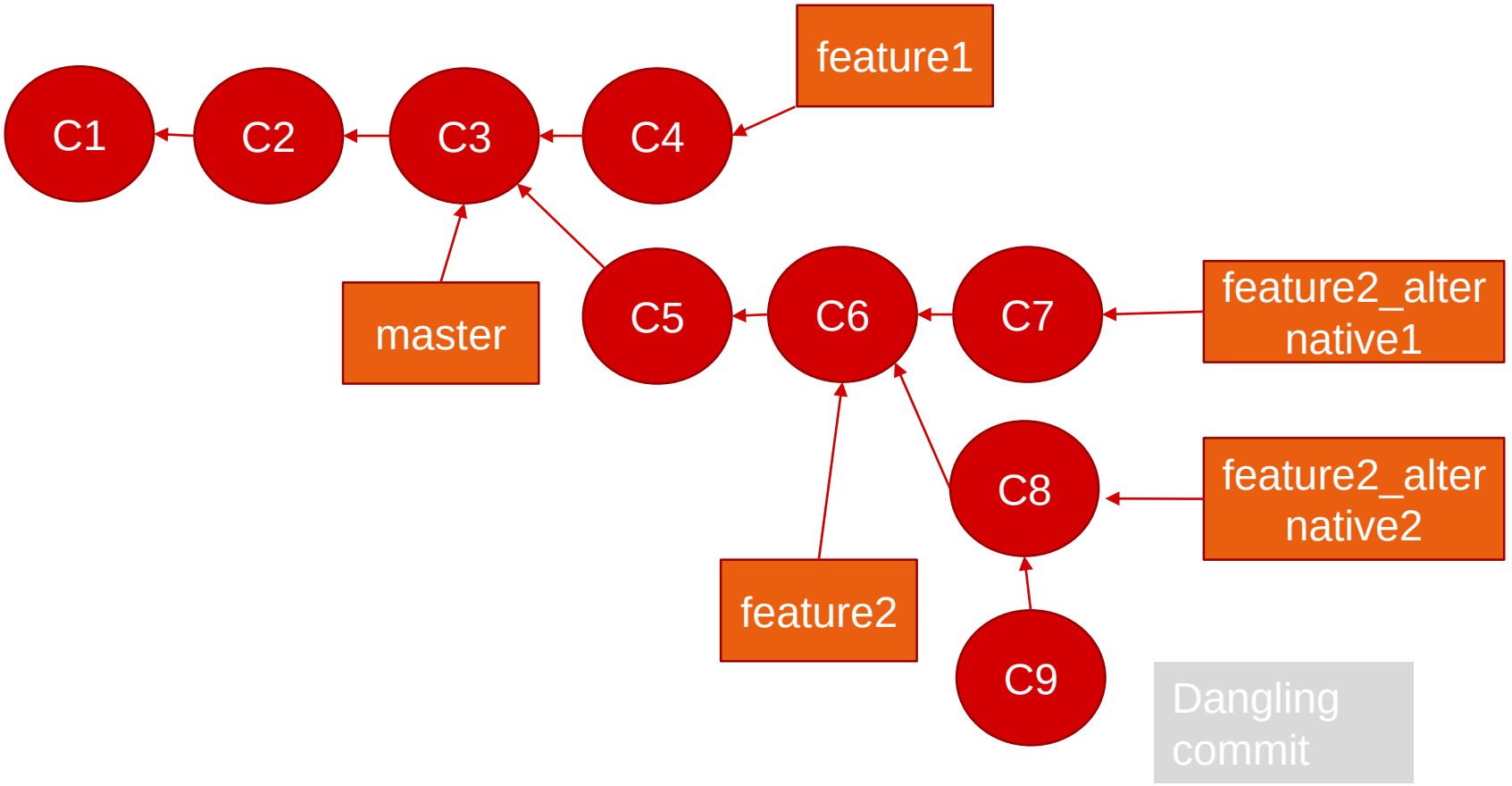
Template



Create-branches.bat



Dangling


feature1
C1
C2
C3
C4
master
C5
C6
C7
feature2_alter native1
C8
feature2_alter native2
feature2
C9
Dangling commit

Übung Branches

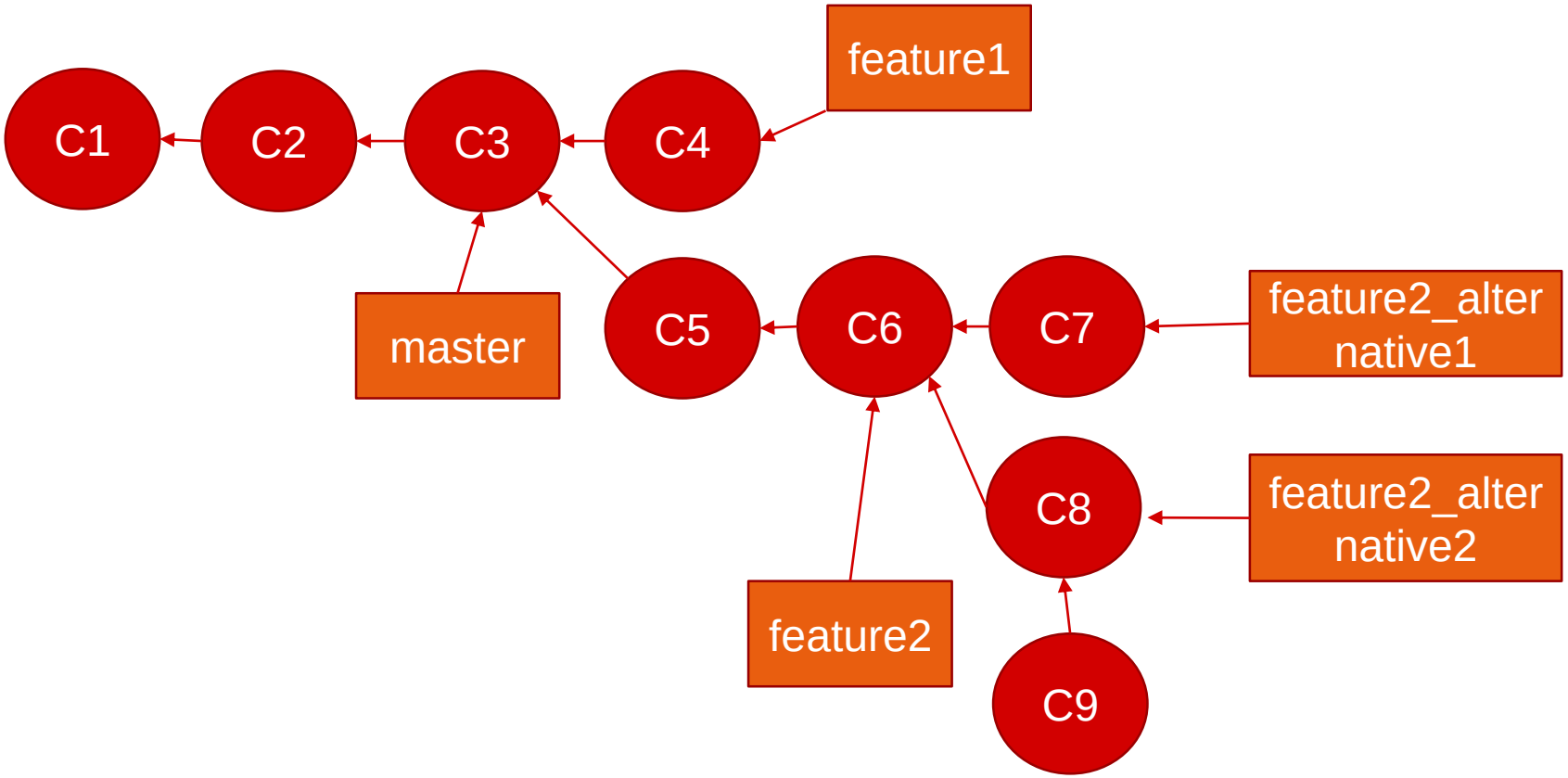
integrata cegos

- Create-branches.bat|sh
- git log --all --decorate --oneline –graph
- Garbage Collection
  - git fsck --unreachable --no-reflogs
  - git reflog expire --expire-unreachable=now --all
  - rem initialize garbage collection
  - git gc --prune=now
- Ideensammlung:
  - FRAGEN STELLEN!
  - Detached HEAD
  - Neue Branches erstellen
  - Löschen feature1-> Dangling commits

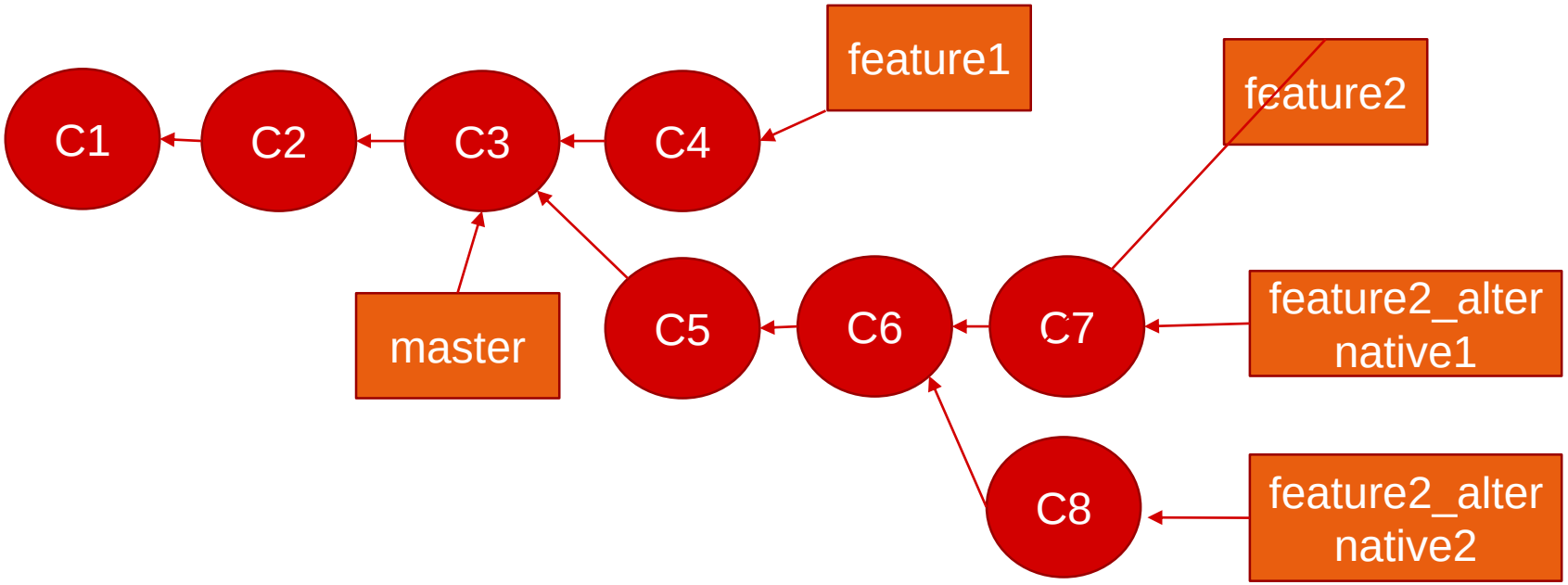
Zusammenführen von Projektständen

- Merge
- Rebase
- Cherry Picking
  - Aktuell in der Community eher als Anti Pattern diskutiert
- Interactive Rebasing

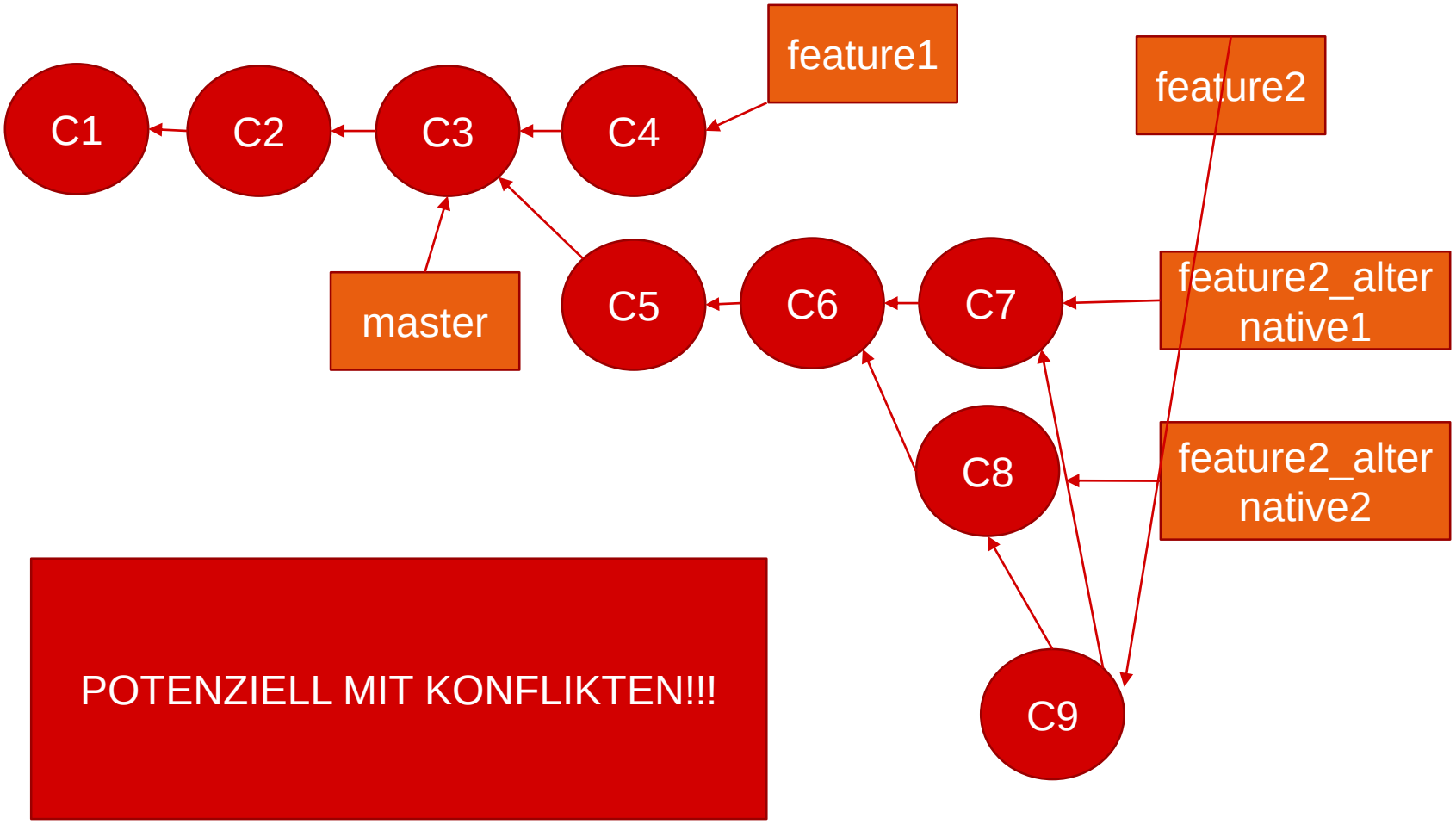
Merge


feature1
C1
C2
C3
C4
master
C5
C6
C7
feature2_alter
native1
C8
feature2_alter
native2
feature2
C9

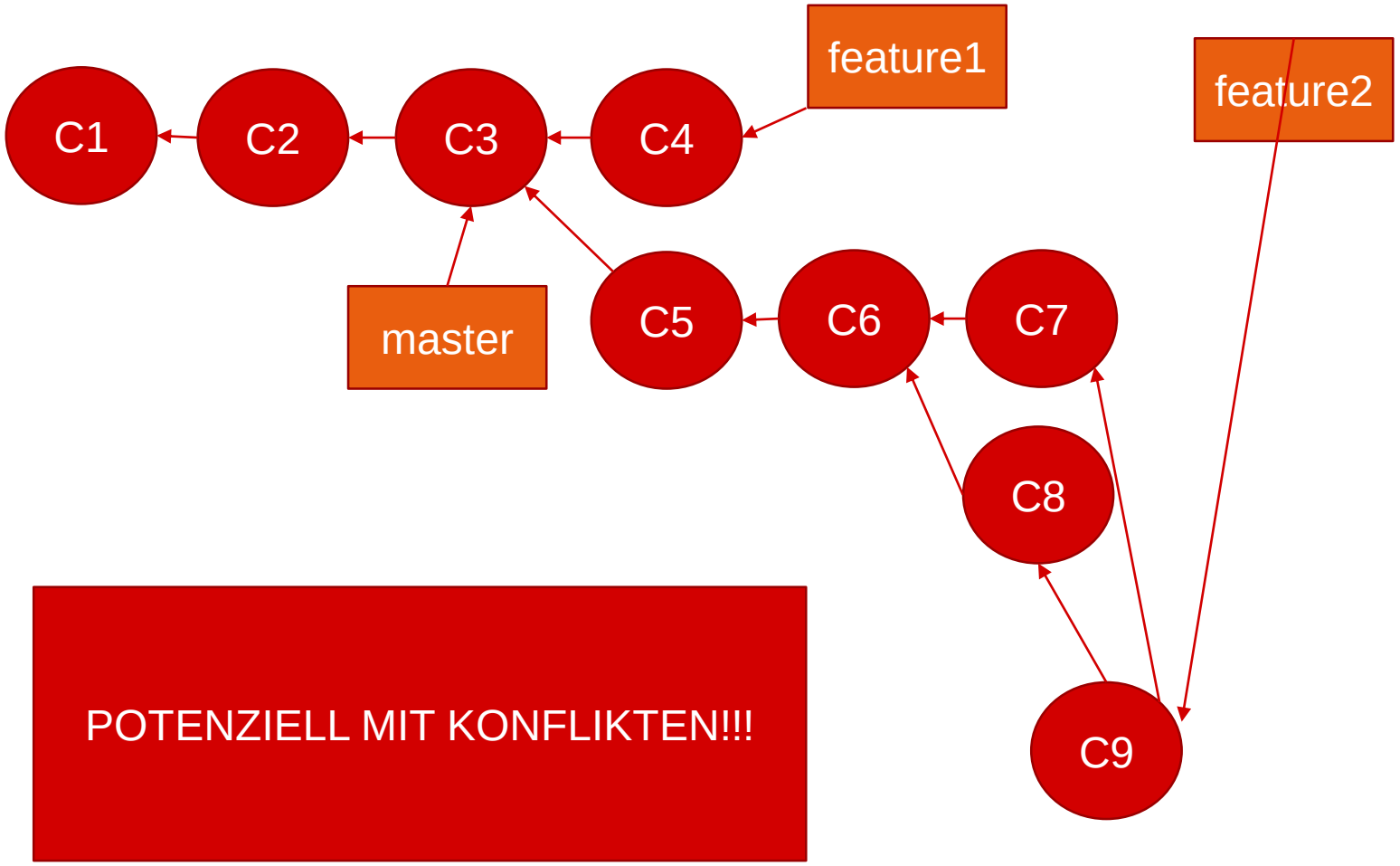
Merge feature 2 mit alt1: fast Forward



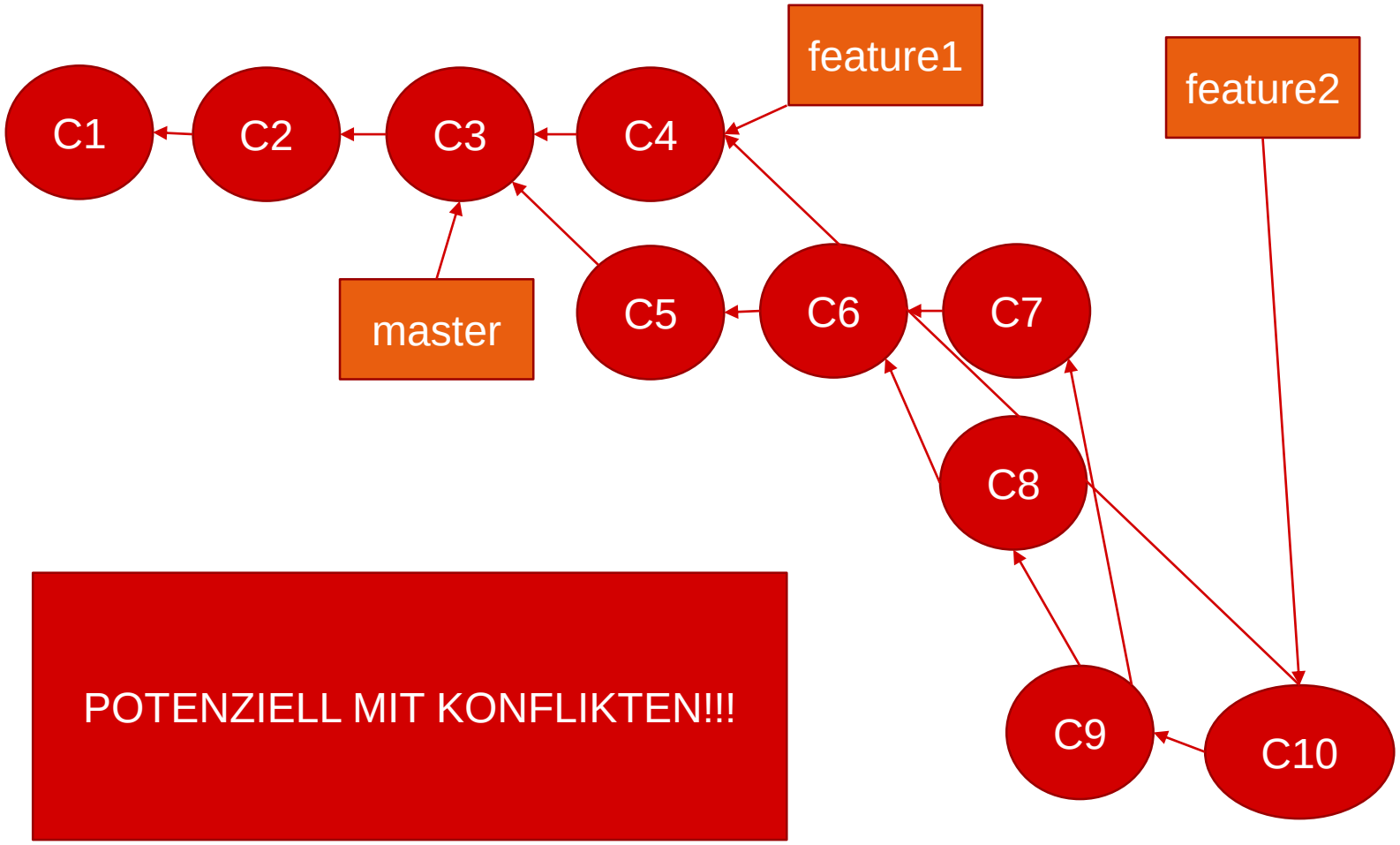
Merge feature 2 mit alt2: Recursive Merge



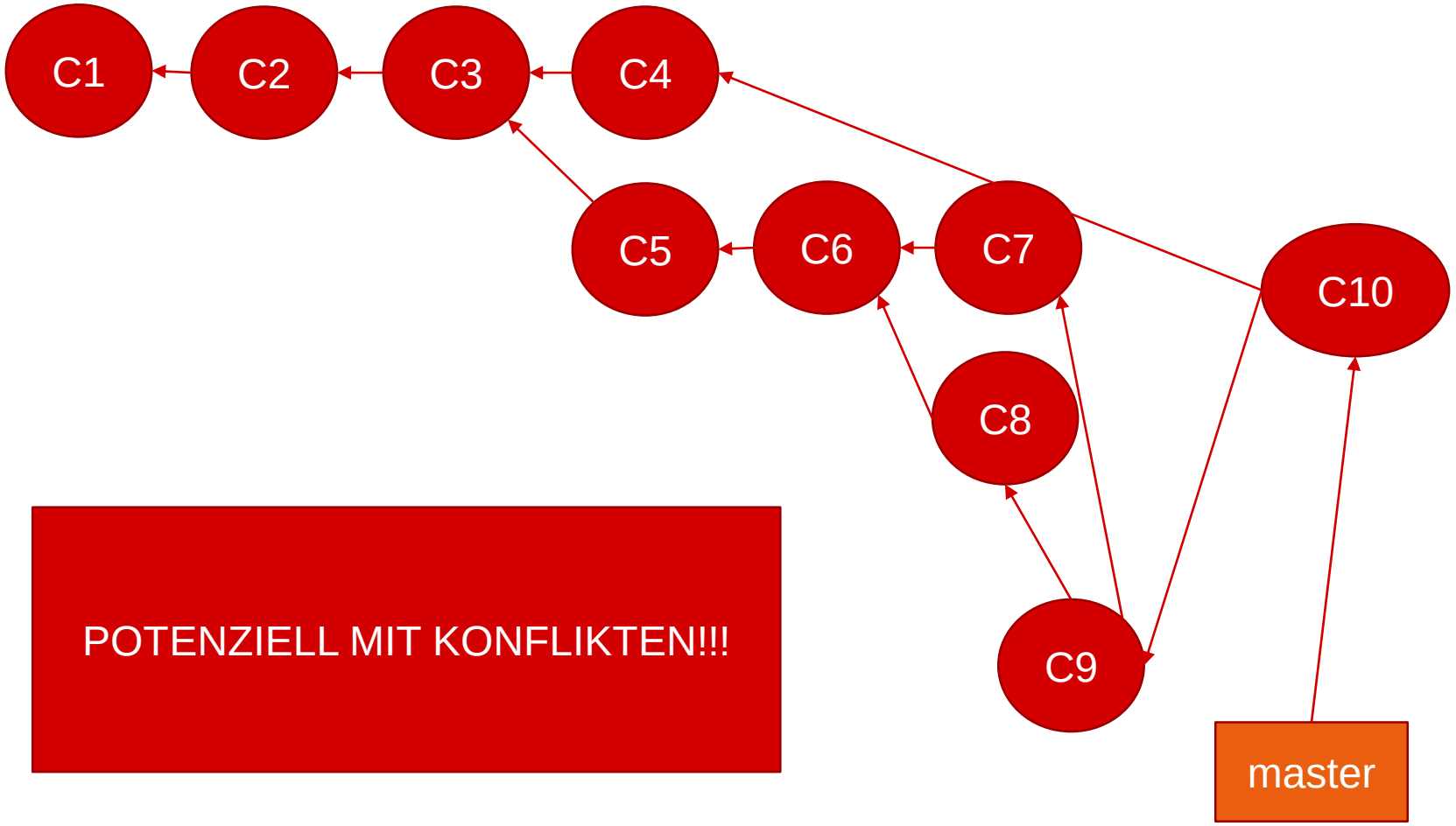
Merge feature 2 mit alt2:



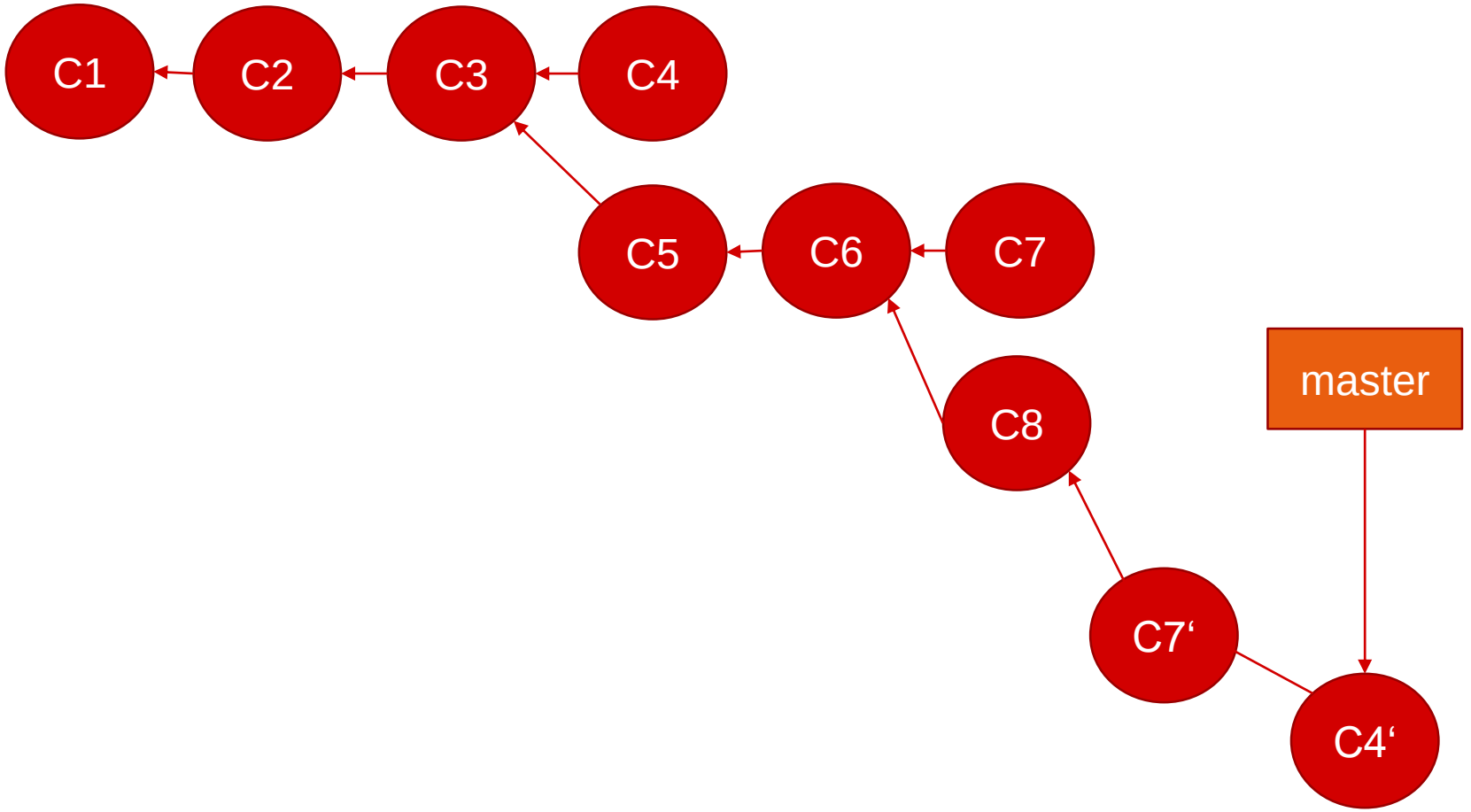
Merge feature2 mit feature1:



Merge master mit feature2:

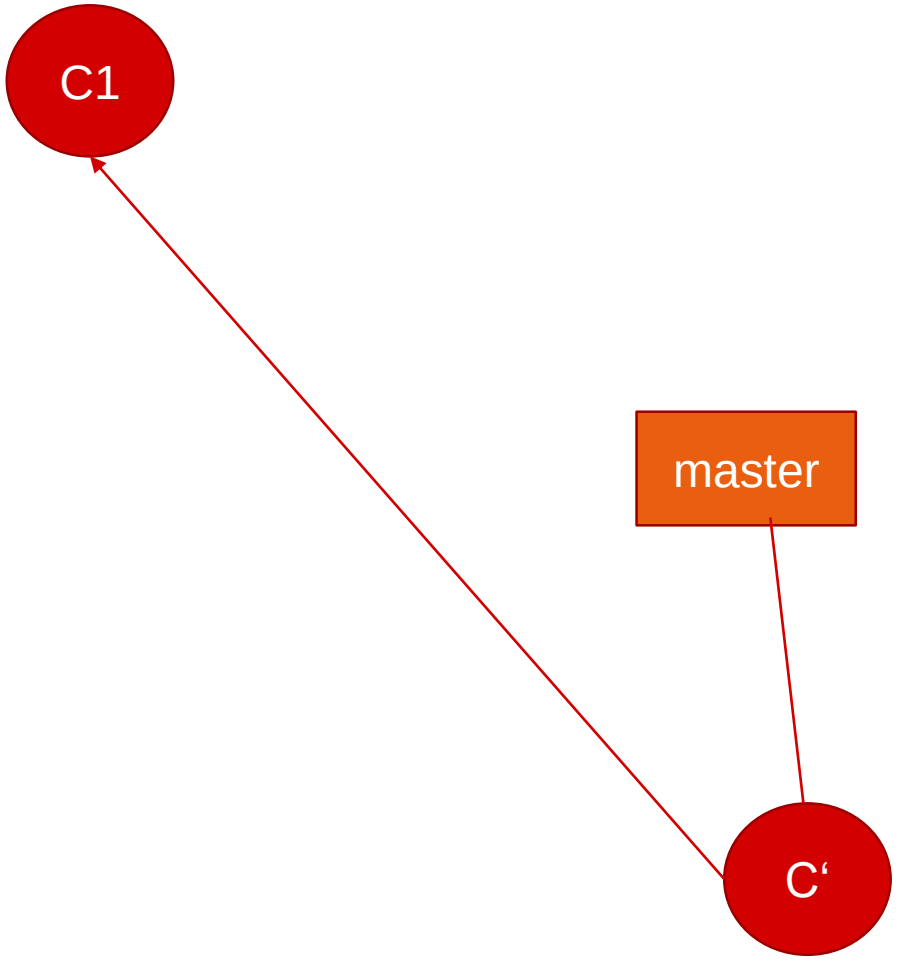


Rebasing



Interactive Rebasing

integrata cegos



Verteilte Repositories

git remote add file:Server s1
git remote add file:Server2 s2

URL-Protokolle

- http/https
- ssh
- file

fetch

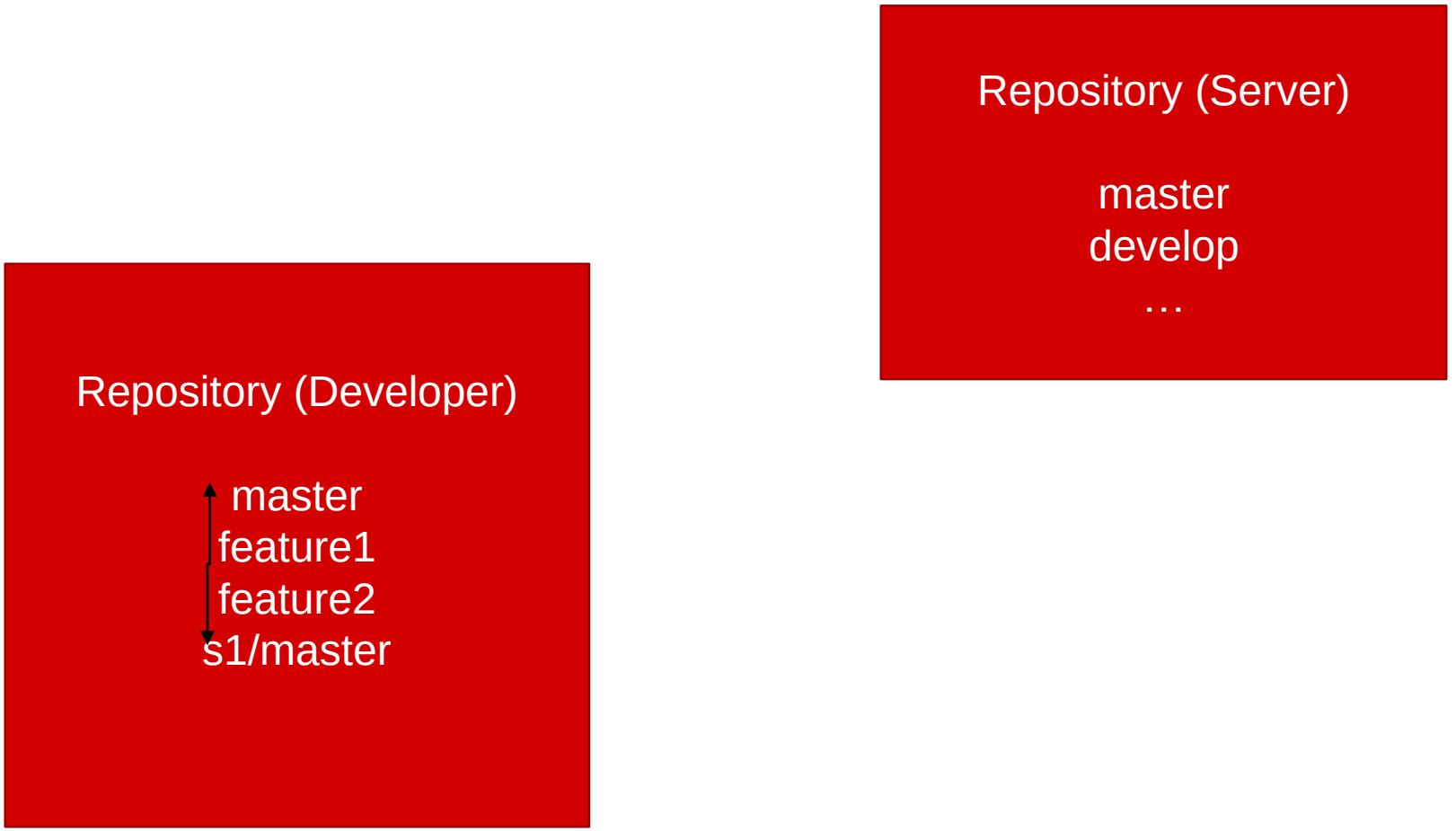
- Remote Branch:
  - fetch
  - Dateitransfer aller Commits und aller Contents

Remote Branches

- s1/master
  - Normal: Alias auf ein Commit
  - Checkout möglich
  - Aus Sicht des Developer-Repository ist der Branch read-only
    - KEIN COMMIT MÖGLICH
  - Nachträgich fetch ist immer möglich, ohne Konflikte
- Lokale Branches können mit einem Remote Branch synchronisiert werden
  - Commit-fähig

Verteilte Repositories

git remote add file:Server s1



push

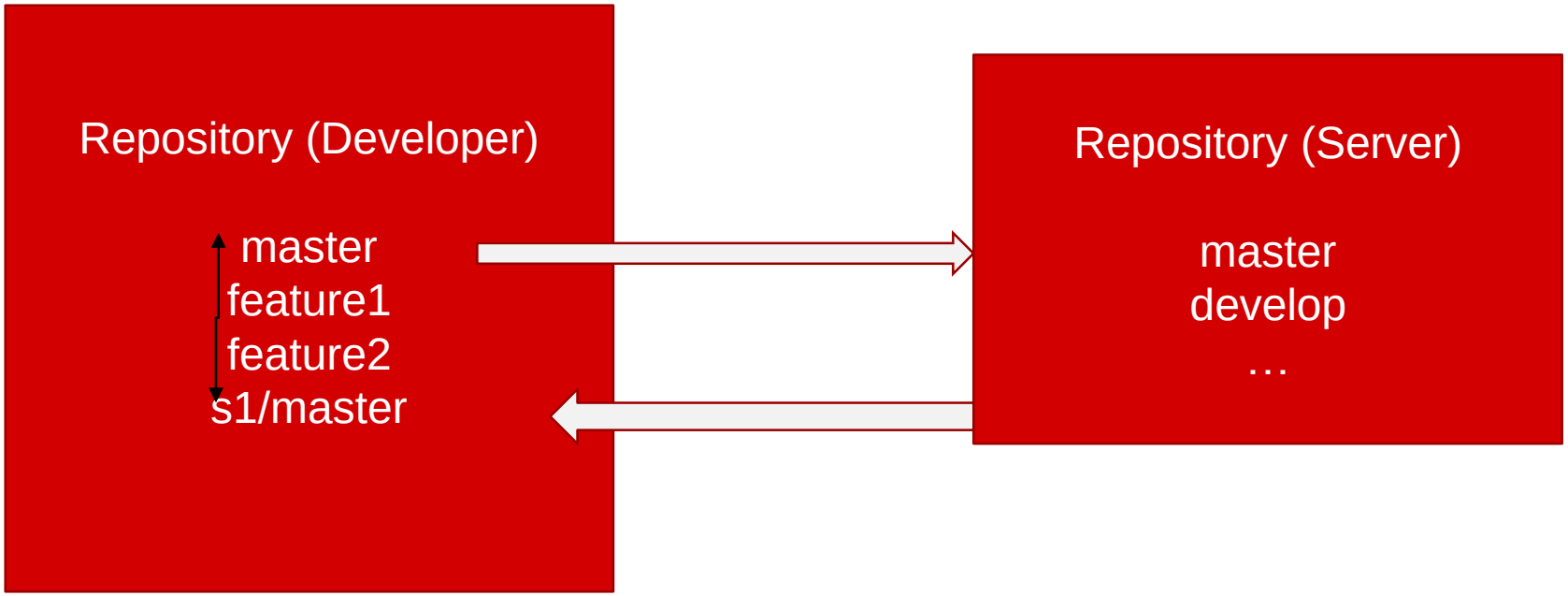
- Push synchronisiert den s1/master mit dem lokalen master
  - THESE!
- In Wirklichkeit komplizierter
  - Push kann fehlschlagen!!!
- Lösung:
  - Push -> fehler
  - Fetch
  - Lokalen branch mit remote branch mergen
    - Alternativ fetch/merge: pull
    - Meistens : fetch – rebase oder pull --rebase
  - Konflikte lösen
  - Commit
  - Push

- Optimistic Locking
  - Git unterstützt kein Pessimistic Locking

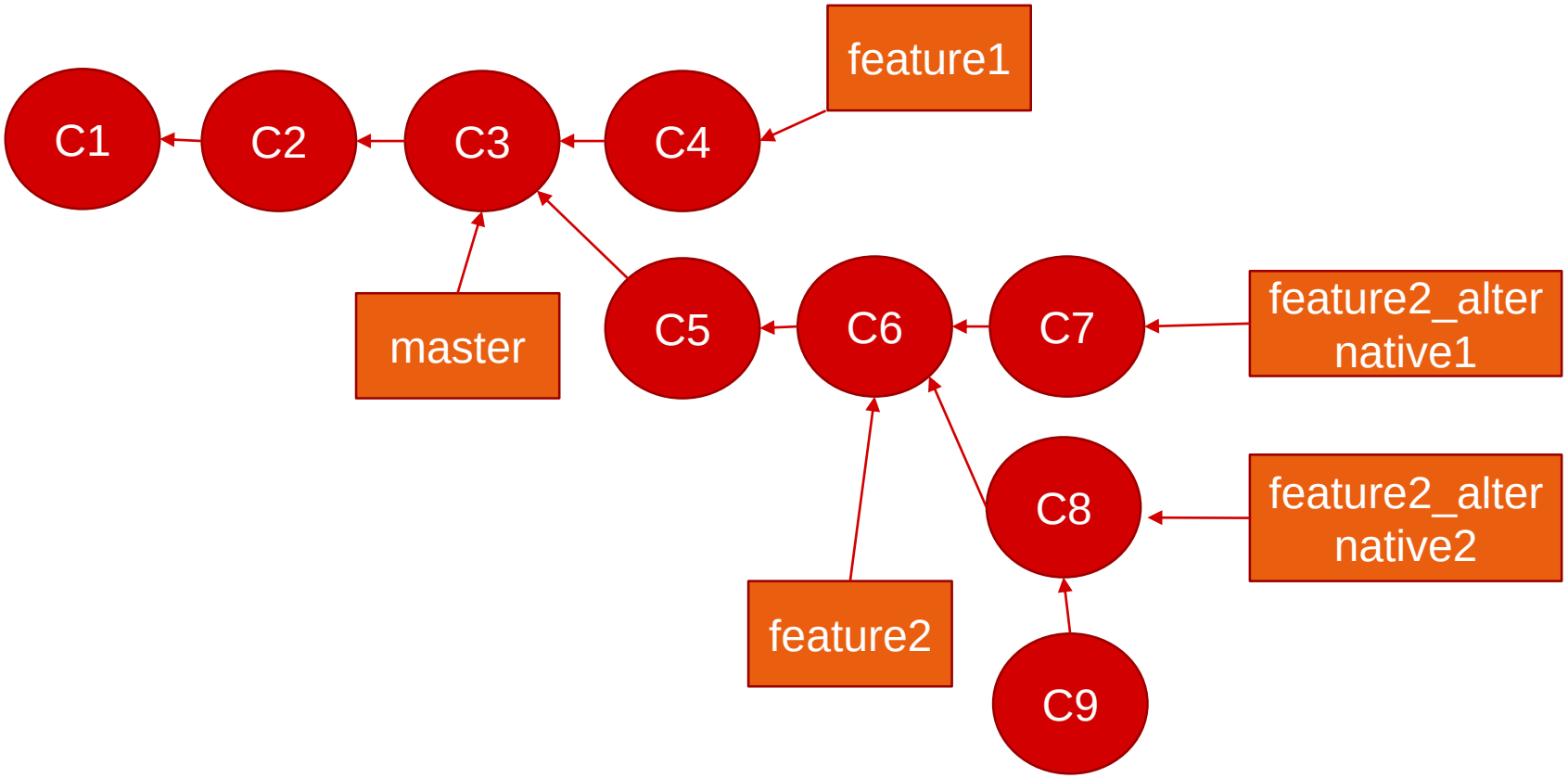
Remote Repositories

- Remote Konfiguration
  - Entferntes Repository url + namespace
  - Lokaler Branch wird als „Spiegel“ des Remote Branches konfiguriert
    - Konfiguration des up- und downstreams
  - Befehle:
    - remote add
    - git clone url
      - Default Namespace: „origin“
- Befehle: fetch, pull, pull –rebase, push

push


Repository (Developer)
master
feature1
feature2
s1/master
Repository (Server)
master
develop
…

Template


feature1
C1
C2
C3
C4
master
C5
C6
C7
feature2_alter
native1
C8
feature2_alter
native2
feature2
C9

Übung: File-Server-Repo

integrata cegos

- create-file-server.bat
- git clone
- Fetch, pull, push…
- Konflikte erzeugen/lösen

Git Workflows

integrata cegos

- Best Practice-Lösungen
- Katalog von „Design Patterns“ = Git Workflows

- Gemeinsames Element: Kurzlebige Feature Branches

- Atlassian: Git Flow
  - Release (Release-Versionen, stabil)
  - Developer (Langlebig, aktuelle Stand der Entwicklung, pontenziell inkonsistent)
  - Feature-Branches
  - Hotfixes
- GitHub Flow
  - Master , releases sind Tags auf dem Master
  - Feature-Entwicklung und HotFixes sind kurzlebige Feature-Branches

Git Client

- Übersicht
  - GIT Gui
  - IDEs
  - Visual Source Tree (Atlassian)
- Features
  - History,
  - Diffs,
  - Suchen
  - Blame

Git Server

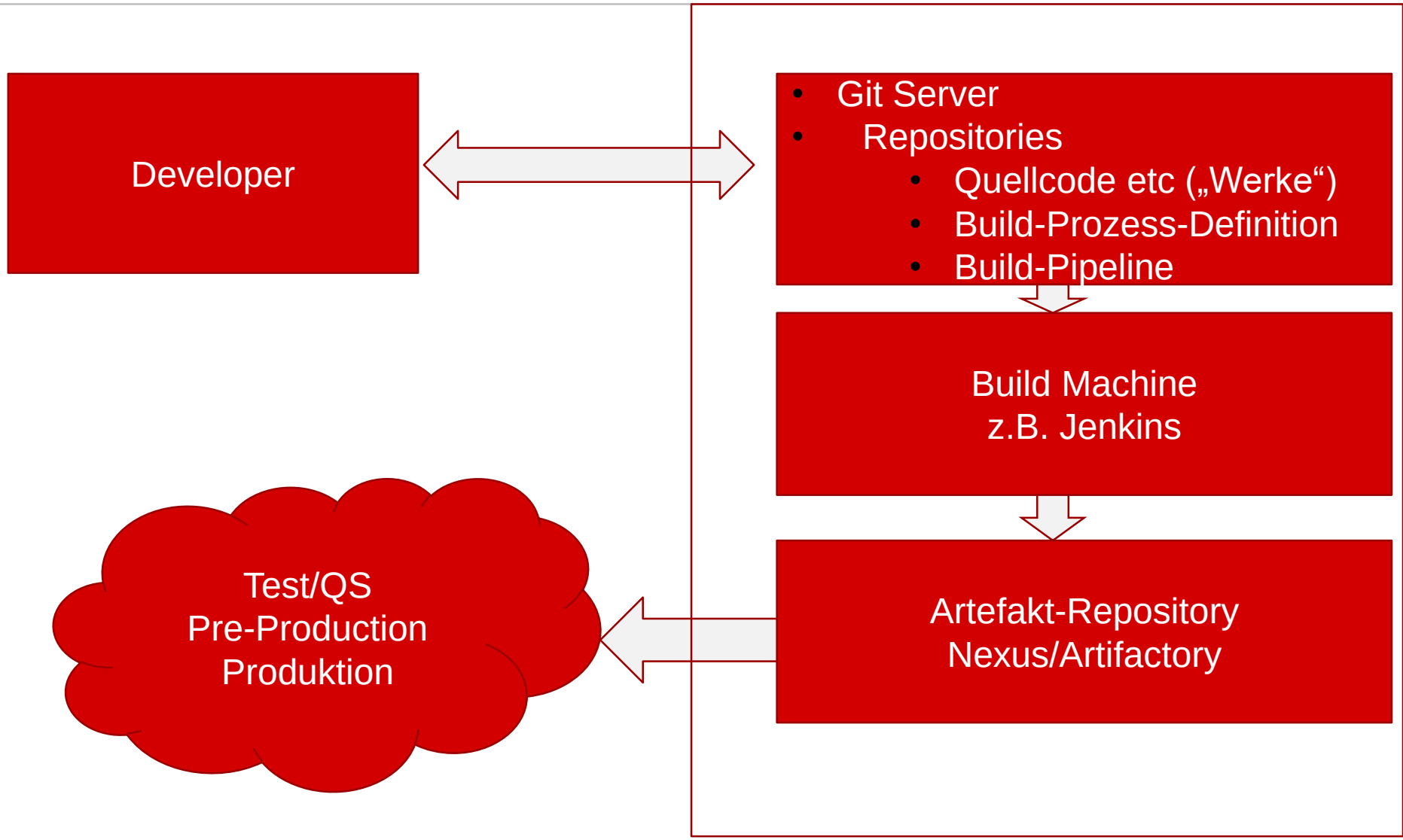
- Git Core
  - Standard Versionsverwaltung
    - Commits
    - Branches
    - Mergen…
- Git Server -> Nicht Bestandteil der Git Community
  - Zentrale Dateiablage
    - Gemeinsamer Zugriff
    - Datei-Sicherung
  - Authentifizierung
  - Autorisierung
  - Plattform für die Team-Kommunikation
  - Push-Verfahren: Recursive merge ohne Konflikte

Server Produkte

- GitHub (Microsoft)
  - Team Foundation Server (Microsoft)
- GitLab
- Bitbucket (Atlassian)

- Standalone: Server wird im Unternehmen betrieben
- Cloud
  - Private
  - Public

Gesamtbild der Software-Entwicklung



Build-Pipeline

- Definiert über den Build-Prozess
- Ausgeführt
  - „Händisch“
  - Scheduled („Nightly Build“)
  - Continuous CI/CD
    - Integration (Build-Prozess inklusive Tests)
    - Deployment (Nach erfolgreicher Test-Phase wird das Artefakt in Produktion gebracht)
    - Delivery