# Inhalt

GIT ..... 2
Bereiche ..... 2
Funktionen ..... 2
Branches vs. Tags ..... 2
Branch ..... 2
Tag ..... 2
Merge ..... 3
Rebase ..... 3
Merge vs. Rebase ..... 4
Issues ..... 4
Bestandteile ..... 4

# GIT

Git ist eine **Versionsverwaltungssoftware**. Damit werden folgende Aspekte bei der Software-Entwicklung verwaltet:

- Verwaltung von Dateien
- Änderungen von Dateien
- Zugriffskontrolle
- Parallele Entwicklung
- Wiederherstellung früherer Versionen

Jeder Commit hat eine einheitliche Kennung, die aus einem Hash-Wert besteht, der durch den Inhalt und die Metadaten des Commits generiert wird.

## Bereiche

*Seite 43, 99*

- **Working-Directory**: Ordner, in dem die Dateien enthalten sind, die bearbeitet werden
- **Staging-Area**: Bereich im der nächste Commit vorbereitet wird
- **Local Repository**: Lokale Version des Remote-Repositories
- **Remote Repository (Origin):** Ort an dem alle Commits und die entsprechende Historie gespeichert wird

## Funktionen

*Seite 111*

- **Checkout**: Wechselt zu einem Branch oder einer früheren Version einer Datei, um Änderungen daran vorzunehmen
- **Commit**: Erstellt eine neue Revision mit den aktuellen Änderungen
- **Revert**: Stellt eine frühere Version wieder her
- **Pull**: Holt den aktuellen Stand eines Repositories vom Origin
- **Push**: Bringt den lokalen Stand des Repositories zum Origin (Erstellt standardmäßig keine neuen Branches am Origin, die es lokal gibt)

## Branches vs. Tags

### Branch

*Seite 66/67, 103*

- Parallele Entwicklung
- Ideen ausprobieren, ohne den Main-Branch zu beeinflussen
- Ein Pointer auf den letzten Commit einer Commit-Kette

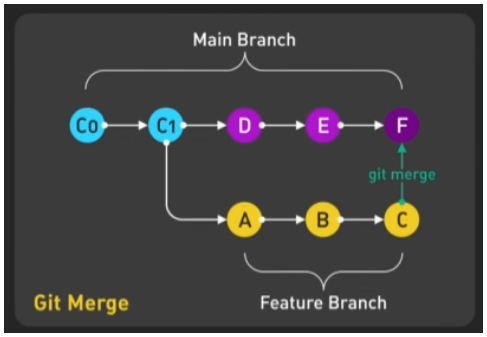
### Tag

*Seite 116*

- Pointer auf einen einzelnen Zustand, der mit einem Namen versehen wird
- Zustand ist über den Namen abrufbar
- z.B. Release-Version „v10.2.5”

## Merge

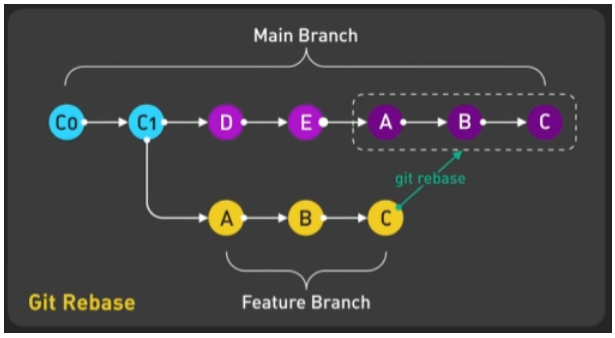
*Seite 76, 80, 84*



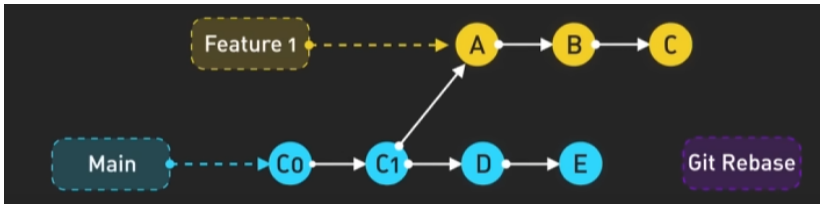

- Zusammenführen von zwei Branches unter Behandlung potenzieller Dateikonflikte
- Macht immer einen Commit zum Abschluss
- Um „feat-1“ in „main“zu mergen muss „main“ der aktuelle Branch sein (Checkout main)
- Konflikt tritt auf, wenn in der gleichen Datei an der gleichen Stelle in den betroffenen Branches unterschiedliche Informationen sind

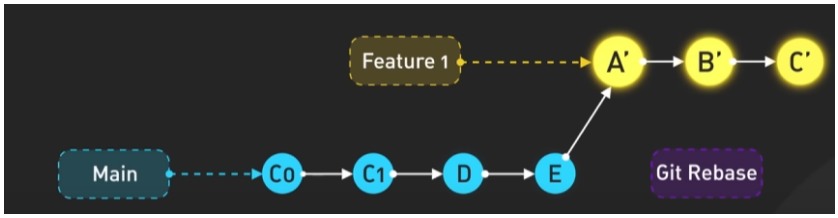
## Rebase

*Seite 87, 94*




- Hängt eine Reihe von Commits einfach an einen anderen Branch an
- Z.B., um einen Feature-Branch Up-To-Date zu halten zum Main-Branch
    - Genauer: Der Main Branch hat sich weiterentwickelt während der Feature-Branch entwickelt wurde. Um die Änderungen aus dem Main-Branch im Feature-Branch zu haben kann die ursprüngliche Abzweigung vom Main-Branch auf den aktuell letzten Commit des Main Branch gesetzt werden. Der Feature-Branch baut dann auf den aktuellen Stand des Main-Branches auf. Dabei kann es zu Konflikten kommen, wenn in beiden Branches die gleichen Stellen unterschiedlich bearbeitet wurden.



### Merge vs. Rebase

| **Merkmal** | **Merge** | **Rebase** |
|---|---|---|
| Historie | Beibehaltung mit Merge-Commit | Umschreiben, lineare Historie |
| Konflikt | Kann bei Merge oder vorher auftreten | Bei jedem Commit möglich |
| Lesbarkeit | Zeigt parallele Arbeit gut | Ist sauberer und linear |
| Verwendung | Zusammenarbeit, Dokumentation | Aufräumen vor Push/Pull Request |

## Issues

Issues sind vielseitig und können aus folgenden Aspekten entstehen:

- Problem / Bug
- Feature-Request
- Verbesserung
- ToDos

### Bestandteile

- **Titel**: Knappe Beschreibung des Anliegen
- **Beschreibung**: Ausführliche Erklärung für das Issue
- **Labels**: Zur Organisation, Bewertung und Eingliederung von Issues (Bug, Wont Fix, Enhancement, Good first issue)
- **Meilensteine**: Assoziation von Issues mit anderen Themen des Projekts (Release, Deadline)
- **Assignee**: Verwantwortliche Person(en) zur Bearbeitung des Issues
- **Kommentare**: Diskussion zum Issue
- **Zustand**: Offen/Geschlossen