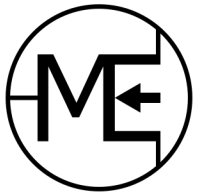
Praktische Einführung in **Git** und **GitHub**

Makerspace Esslingen



Vorbereitung abgeschlossen?

- Git installiert?
- Username, E-Mail und Editor in Git-Config eingetragen?
- GitHub-Account erstellt?
- SSH-Key bei GitHub hinterlegt?
- Mit WLAN verbunden?

Agenda

- Erste Schritte mit Git
- Zusammenarbeit mit GitHub
- Nützliches

- Alles in der Konsole
- Praktisch
- Fragen einfach stellen

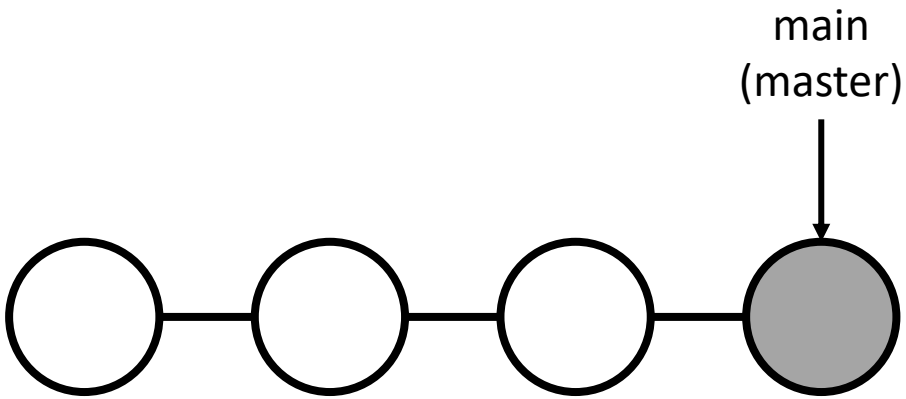
Was ist git?

- Versionsverwaltung
- Variantenverwaltung

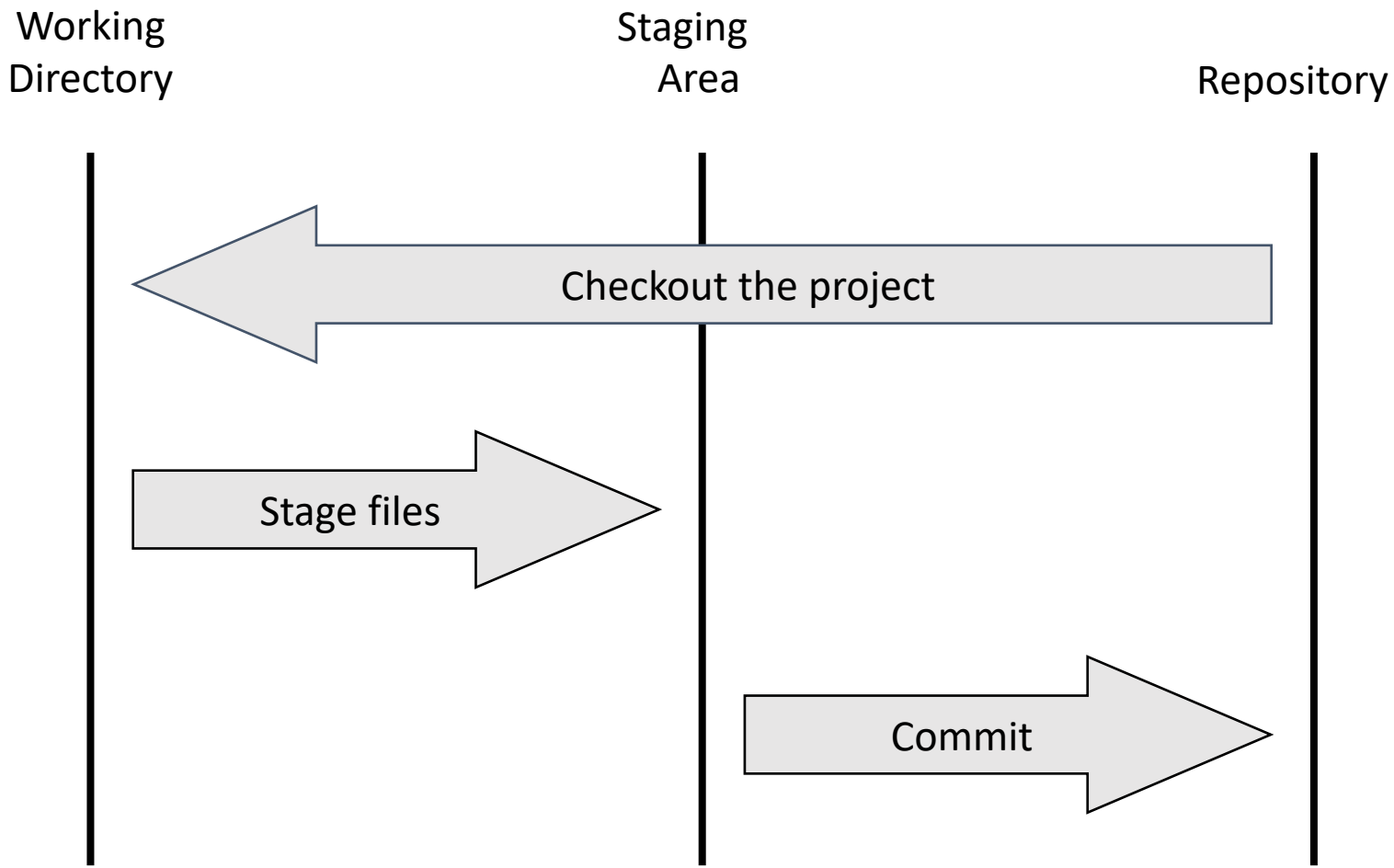
Warum Git?

- Dateien mit anderen austauschen
- Nachvollziehbare Historie
- Remote Backup
- Offline Arbeiten möglich

Grundlagen



Die drei Stages



Makerspace Esslingen

Demo

Übung

```
git init
git status
git add
git commit
git diff
git log
git show
```

- Mehrere Commits erstellen
- Mehrere Dateien
- Ins Log schauen

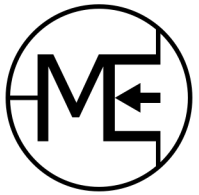
Gerne auch zu zweit

Makerspace Esslingen

Änderungen rückgängig machen

Änderungen vor Commit

- Untracked files
- Changed files
- Files staged for commit



Demo

Übung

```
rm
git restore
git restore --staged

git diff
git status
```

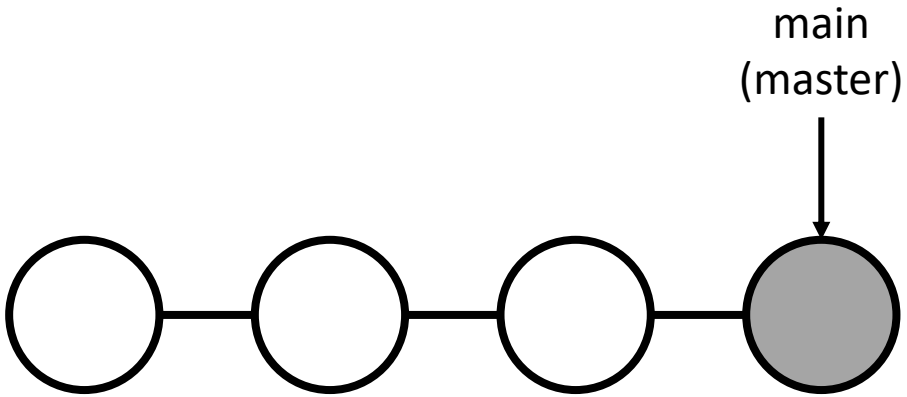
- Datei erstellen und verwerfen
- Änderungen an Datei verwerfen
- Vorgemerkte Änderungen verwerfen

Makerspace Esslingen

Commits rückgängig machen

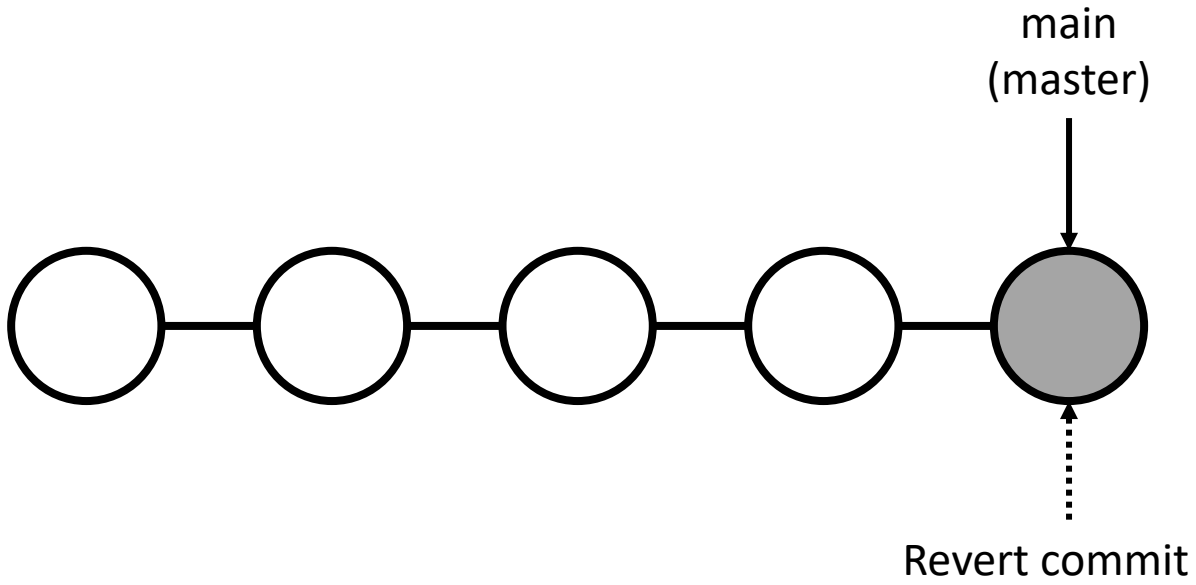
- Revert commit
- Reset

Revert commit



Makerspace Esslingen

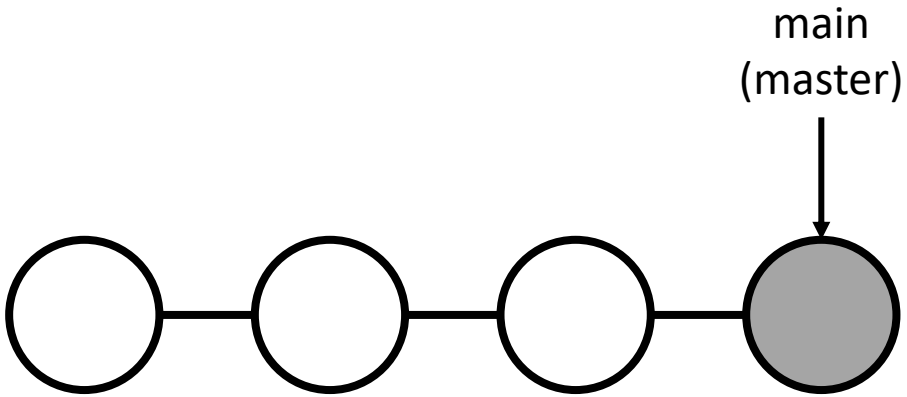
Revert commit




```
git revert $commit_hash
```

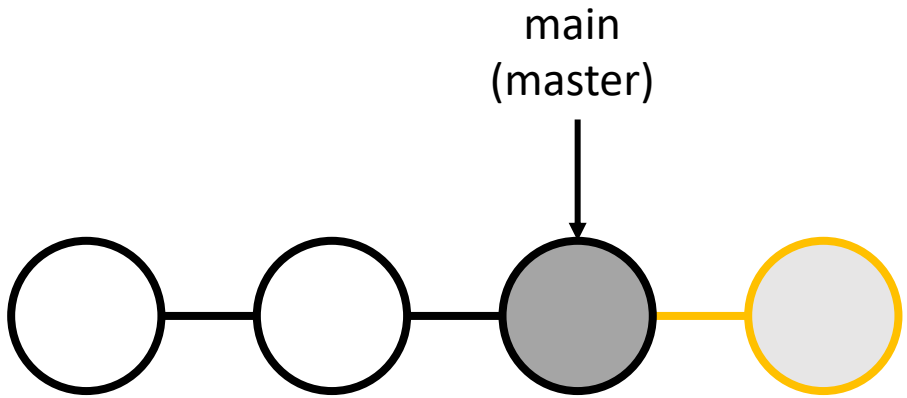
Makerspace Esslingen

Hard reset



Makerspace Esslingen

Hard reset




```
git reset --hard HEAD~1
```

Makerspace Esslingen

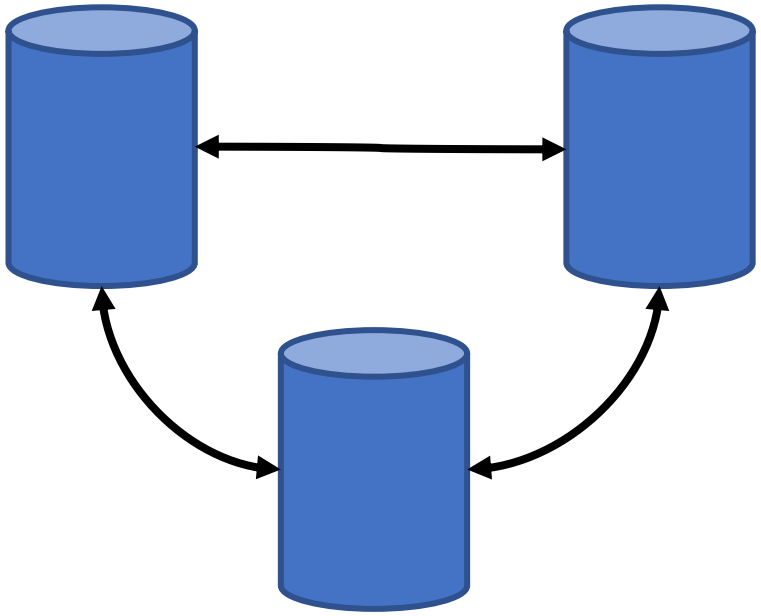
Demo

Übung

```
git revert $COMMIT_HASH
git revert HEAD

git reset --hard HEAD~1

git status
git log
```

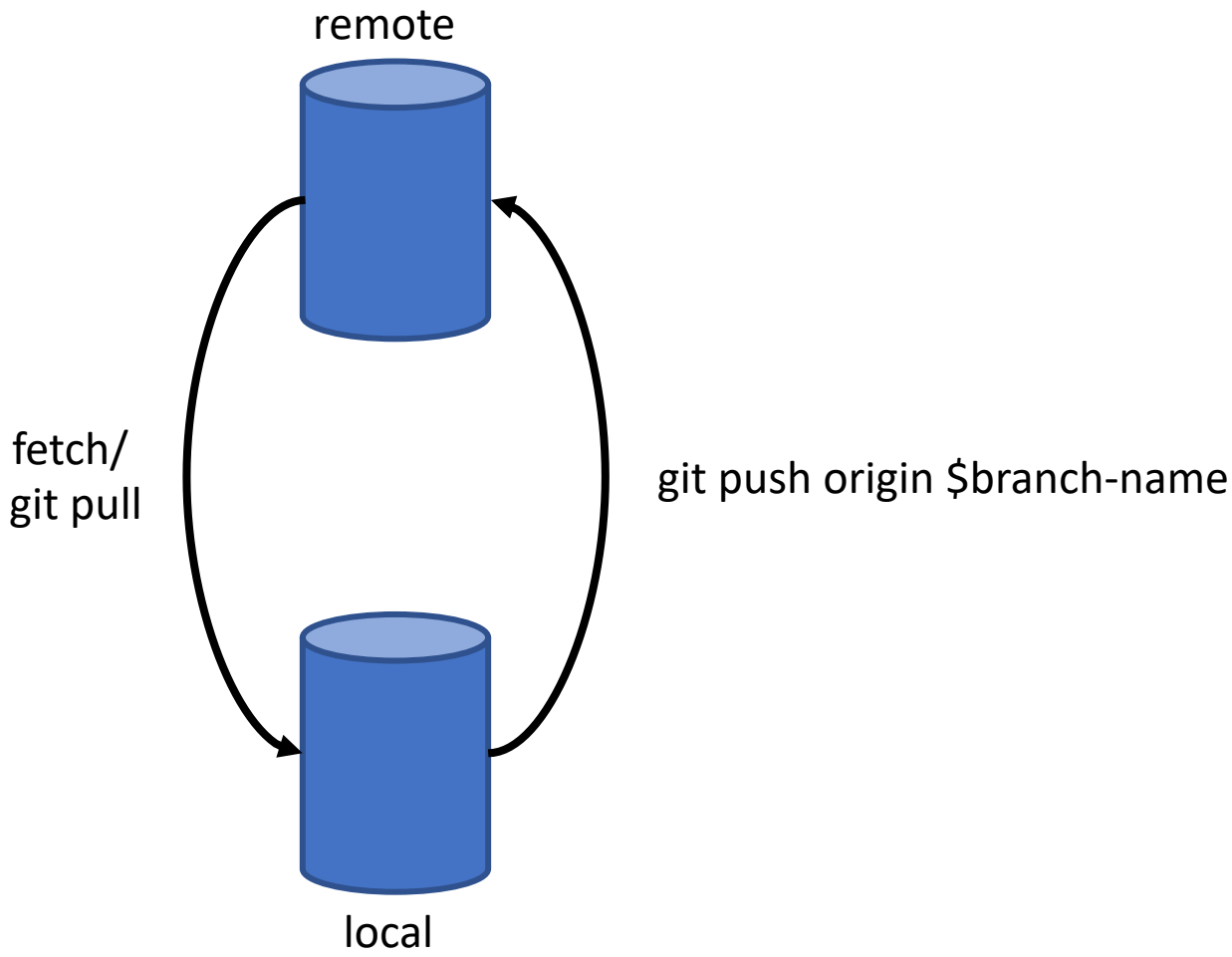
- Revert commit erstellen
  - Hash angeben
  - HEAD verwenden
- Hard reset durchführen
- Verhalten im Log vergleichen

Remotes



Makerspace Esslingen

Remotes



Makerspace Esslingen

Demo

Übung

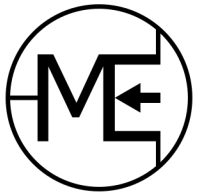
```
git clone $REPO_URL
```

```
git remote add origin
$REPO_URL
```

```
git push
```

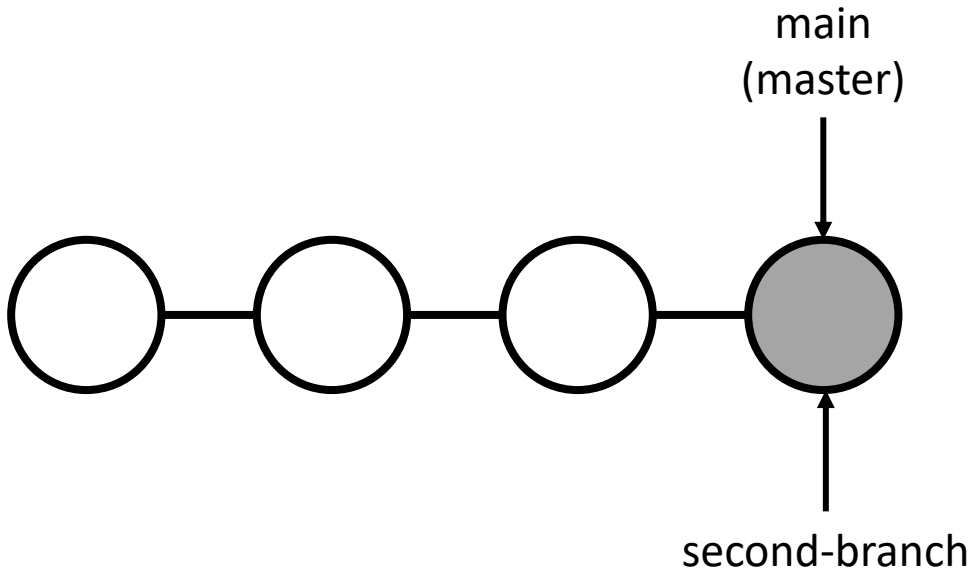
```
git pull
```

- Repository auf GitHub erstellen
- Klonen
- Bestehende Historie in Repository auf GitHub pushen



Branches

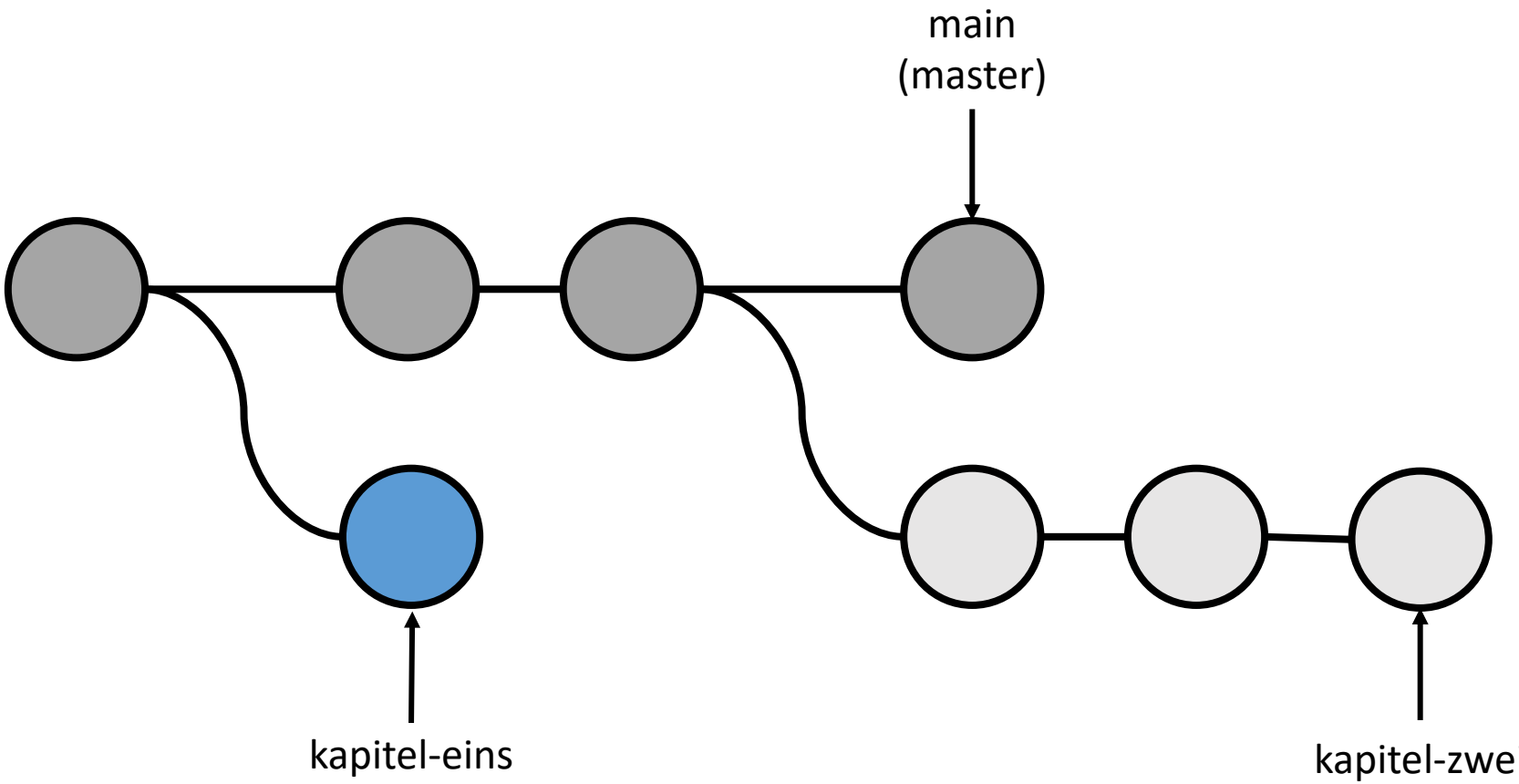
Branches




```
git checkout -b $branch_name
```

Makerspace Esslingen

Branches



Makerspace Esslingen

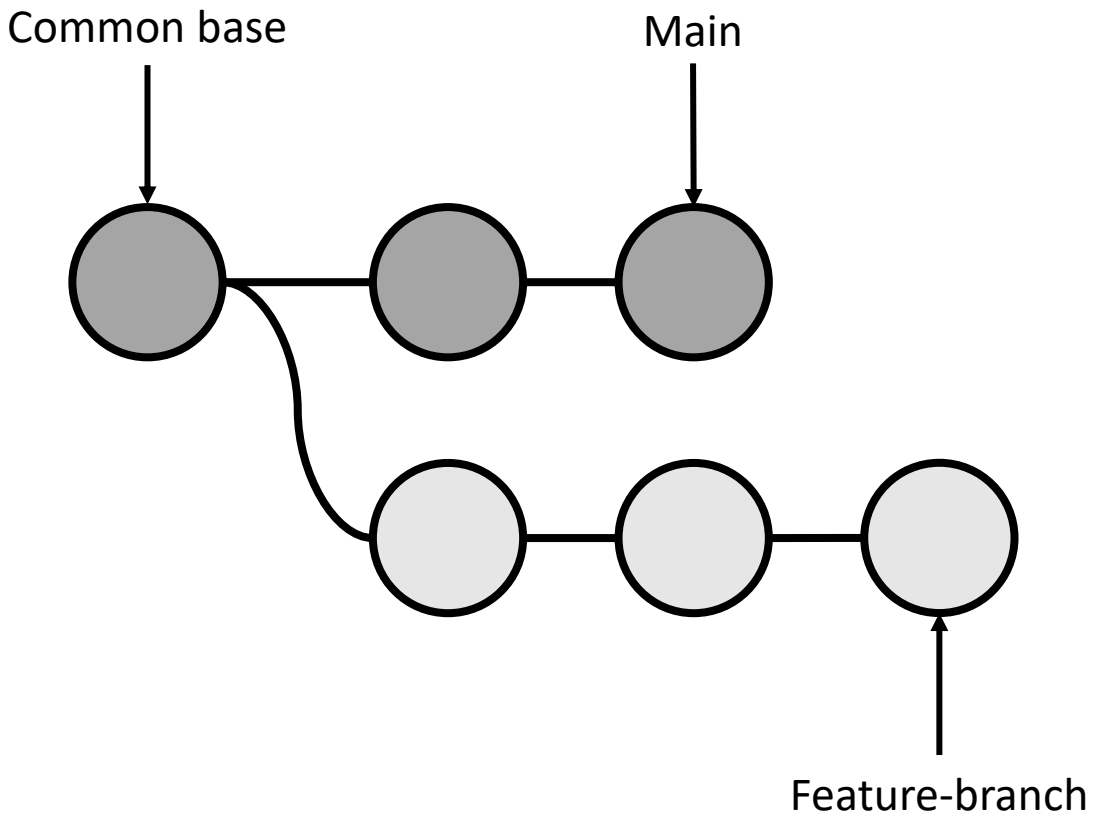
Demo

Übung

```
git checkout -b $name

git checkout $name

git log --graph --all
```

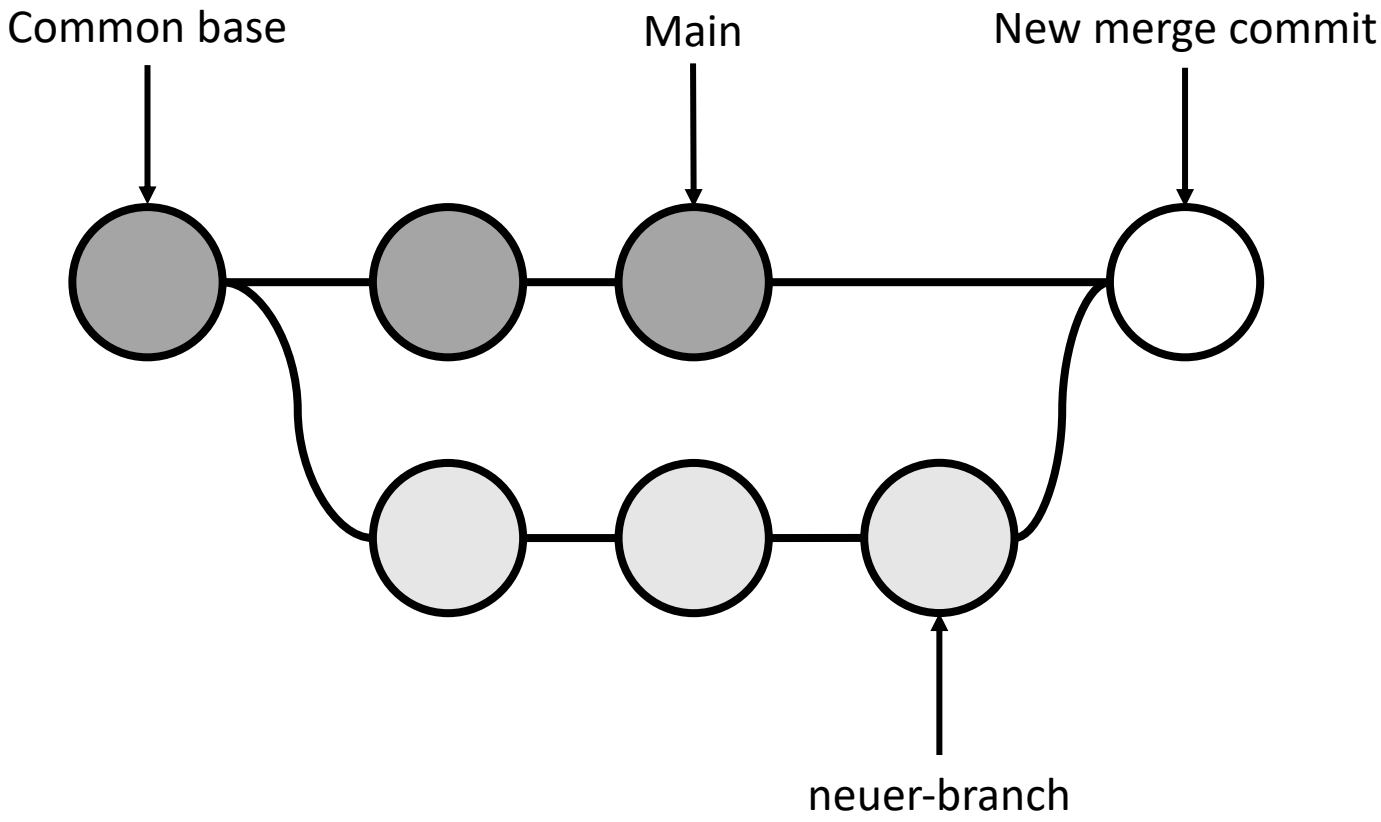
- Mehrere Branches anlegen
- Commits auf Branches erstellen
- Historie im Log anschauen

Merge



Makerspace Esslingen

Merge



Makerspace Esslingen

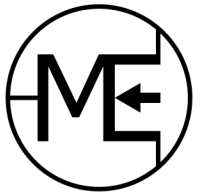
Demo

Übung

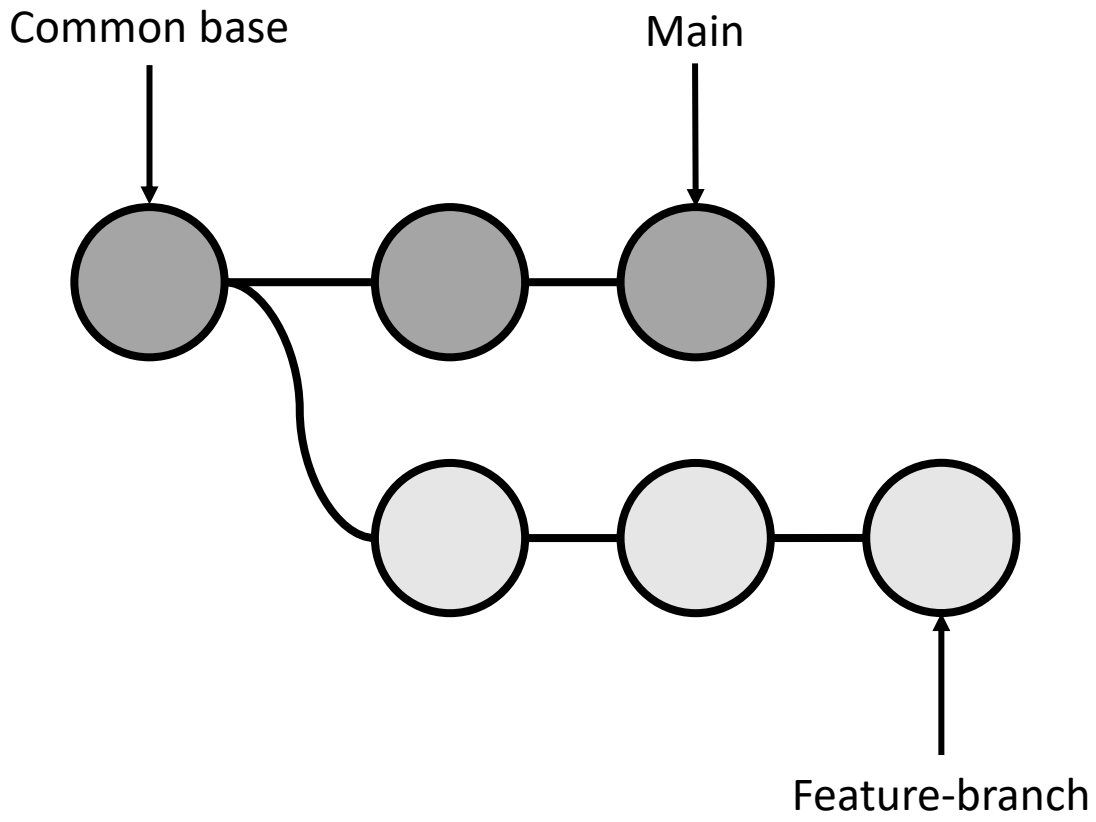
```
git checkout -b $name

git merge $branchname

git log --graph --all
```

- Branch mit Commits erstellen
- Commits auf main erstellen
- Neuen Branch in main mergen
- Historie im Log anschauen

Makerspace Esslingen

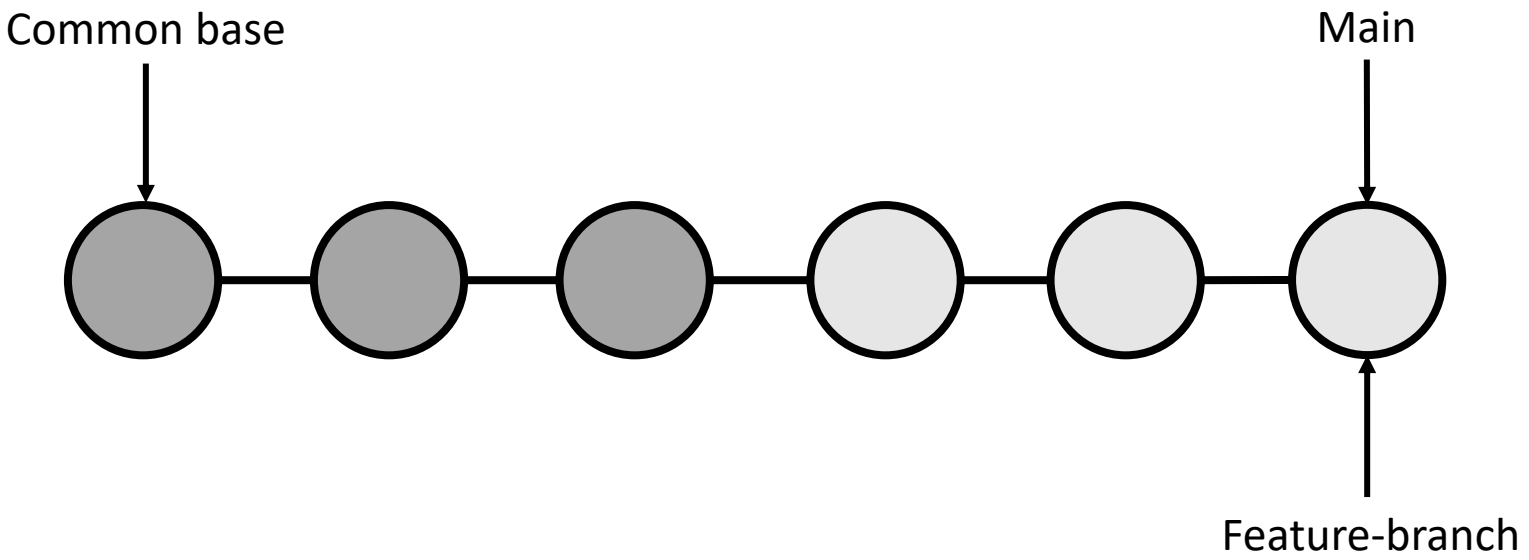


Rebase



Makerspace Esslingen

Rebase



Makerspace Esslingen

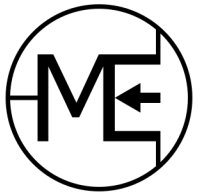
Demo

Übung

```
git checkout -b $name

git rebase $branchname

git log --graph --all
```

- Branch mit Commits erstellen
- Commits auf main erstellen
- Neuen Branch auf main rebasen
- Historie im Log anschauen

Makerspace Esslingen



Fast geschafft!

Weitere Infos

Dokumentation
https://git-scm.com/docs
https://rogerdudler.github.io/git-guide/
https://git-scm.com/book/en/v2
http://think-like-a-git.net/
https://www.atlassian.com/de/git/tutorials/comparing-workflows

Cheat Sheets
http://git-cheatsheet.com/
https://about.gitlab.com/images/press/git-cheat-sheet.pdf
https://education.github.com/git-cheat-sheet-education.pdf

.gitignore

- Eine gitignore Datei spezifiziert Dateien,
- die Git nicht verwalten soll

- Dokumentation: https://git-scm.com/docs/gitignore

.gitconfig

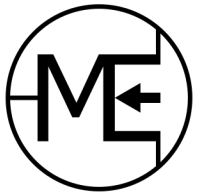
- Name und E-Mail
- Standardeditor
- Template für Commit-Message
- Aliase

- Dokumentation: https://www.git-scm.com/book/en/v2/Customizing-Git-Git-Configuration

Stash

- Änderungen zwischenspeichern
- ohne Commits zu erstellen

- `git stash`
- `git stash pop`



Tags

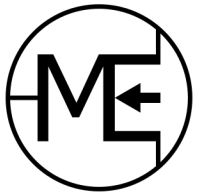
- Commits mit einem Namen versehen

- `git tag -a v1.2 -m "Release 1.2"`

Amend

- Änderungen an letzten Commit anhängen
- Commitmessage des letzten Commits ändern

- `git commit --amend`



Interactive Rebase

- Commitmessages nachträglich ändern
- Commits nachträglich bearbeiten
- Reihenfolge von Commits ändern
- Uvm.

```
git rebase -i $commit_hash
```

Blame

Herausfinden von wem eine Änderung ist

```
git blame $dateiname
```

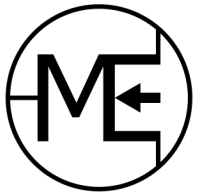
Makerspace Esslingen

Bisect

Einen fehlerhaften Commit suchen und finden

```
git bisect start

# der aktuelle Commit ist schlecht
git bisect bad

# Aber dieser alte Commit war gut
git bisect good $commit_hash
```

Makerspace Esslingen

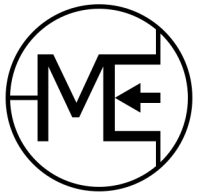


Hooks

Befehle automatisiert ausführen

Dokumentation: https://git-scm.com/docs/githooks

Workflows



Trunk based development

- Alle arbeiten auf master/main
- Kennt man aus SVN

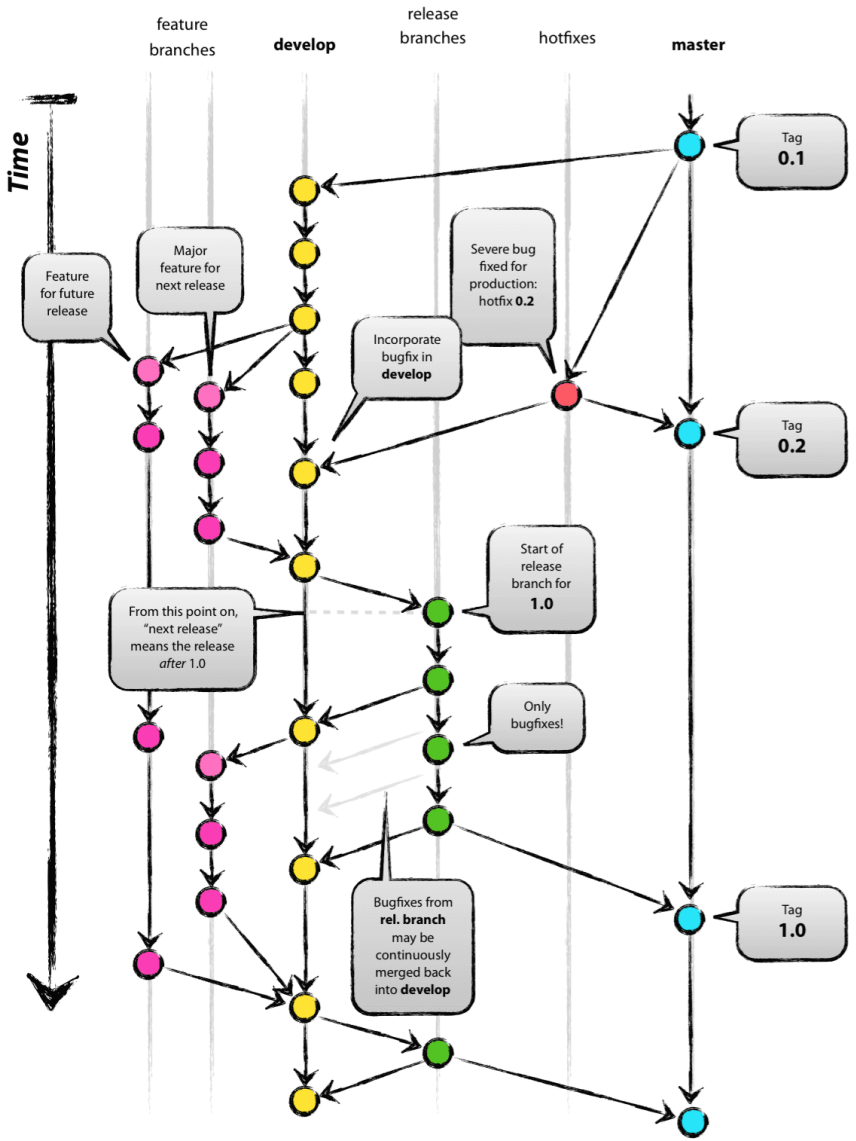
- https://trunkbaseddevelopment.com/

Feature Branch Workflow

- Ein Branch pro Feature
- Code Review, dann Merge

- https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Git Flow



- https://nvie.com/posts/a-successful-git-branching-model/
- https://leanpub.com/git-flow/read#leanpub-auto-git-flow-workflow

Makerspace Esslingen