

n39 Workshop - git

2023-11-22

David

<https://github.com/24367dfa>

Was ist git?

- VCS - Version Control System
- ist auf Linus Torvalds Mist gewachsen - für Entwicklung von Linux
- vorallem für Source Code
- funktioniert aber für jegliche plaintext files
- inzwischen das am weitesten verbreitete VCS tool - Industriestandard

Warum will man das benutzen?

- nie wieder `my_project_version12.zip`,
`my_project_version12_final.zip`,
`my_project_version12_final_jetzt_aber_wirklich.zip`
- Ausprobieren ohne Angst
- Zusammenarbeit mit anderen
- "Backups" von Software-Projekten

Basics: `git init`

```
mkdir my_awesome_repo && cd my_awesome_repo  
git init  
touch README.md  
git add README.md  
git commit
```

Basics: `git status`

```
git status
```

Einschub: `git config`

```
git config --set --global user.email="david.kilias@gmail.com"  
git config --set --global user.name="David Kilias"
```

Basics: `git branch`

```
git branch  
git branch my-test-branch  
git checkout my-test-branch  
git checkout main  
git branch -d my-test-branch
```

Basics: git remote

```
# TODO vorher in github repo anlegen  
git remote add origin git@github.com:24367dfa/my_awesome_repo.git  
git push -u origin main
```

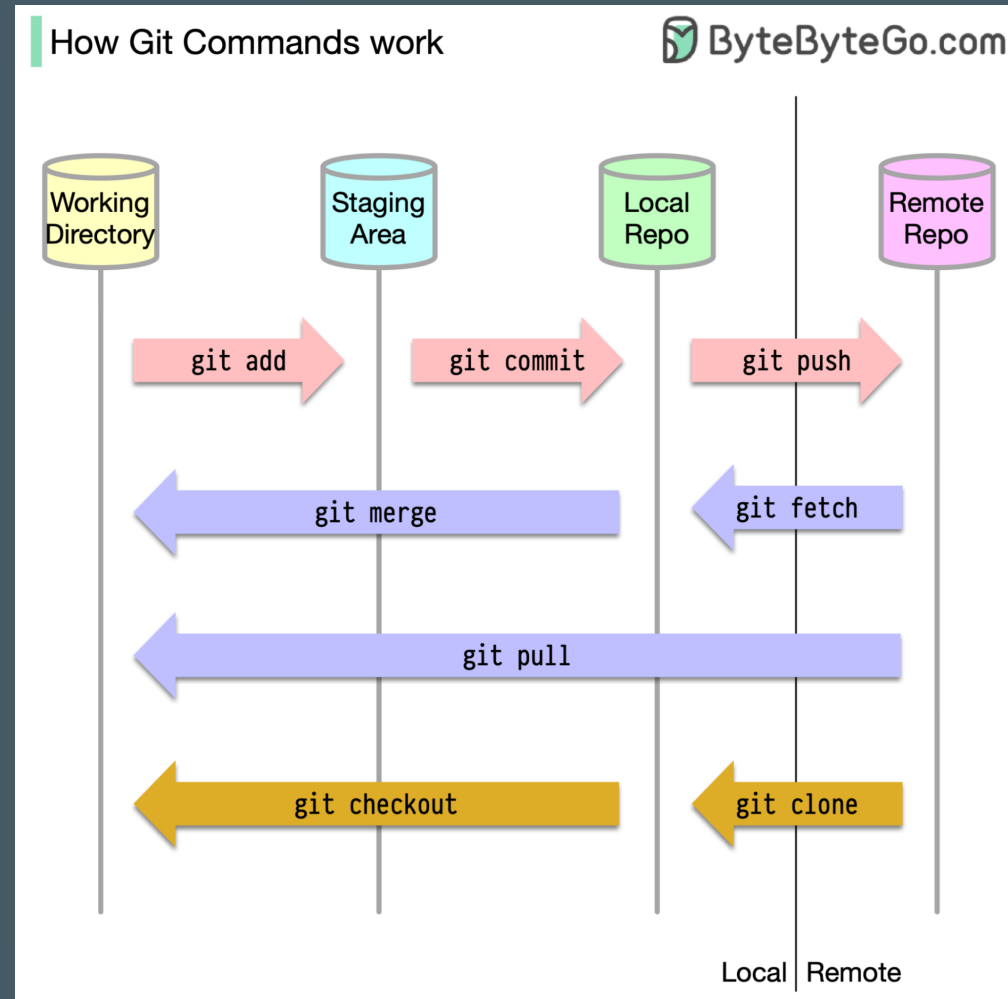
ein Repository kann mehr als ein remote haben

Einschub: SSH Key

```
ssh-keygen -t ed25519 -f ~/.ssh/id_workshop  
cat ~/.ssh/id_workshop.pub | xclip -selection c
```

bei github hinterlegen <https://github.com/settings/keys>

Basics: Zusammenfassung



Einschub: Nutzt Werkzeuge

- das geht alles auf CLI
- für die Grundlagen und bei Problemen super sinnvoll, das zu beherrschen
- für alle ernstzunehmenden Editoren/IDE gibt es Plugins

Intermediate: `git log`

```
git log  
git log -5  
git log -L 1,10:README.md
```

Intermediate: `git tag`

Name für einen bestimmten Commit

```
git tag v1.0.0  
git push --tags
```

Achtung: einmal veröffentlichte Tags nicht löschen!

Intermediate: `git merge`

Branches zusammenführen

```
git checkout -b my-new-feature  
touch feature.md  
git add feature.md  
git commit  
git checkout main  
git merge my-new-feature
```

Intermediate: `git rebase`

Eigenen Branch ans Ende verpflanzen

Falls nach dem Abzweigen Änderungen am Zielbranch passiert sind

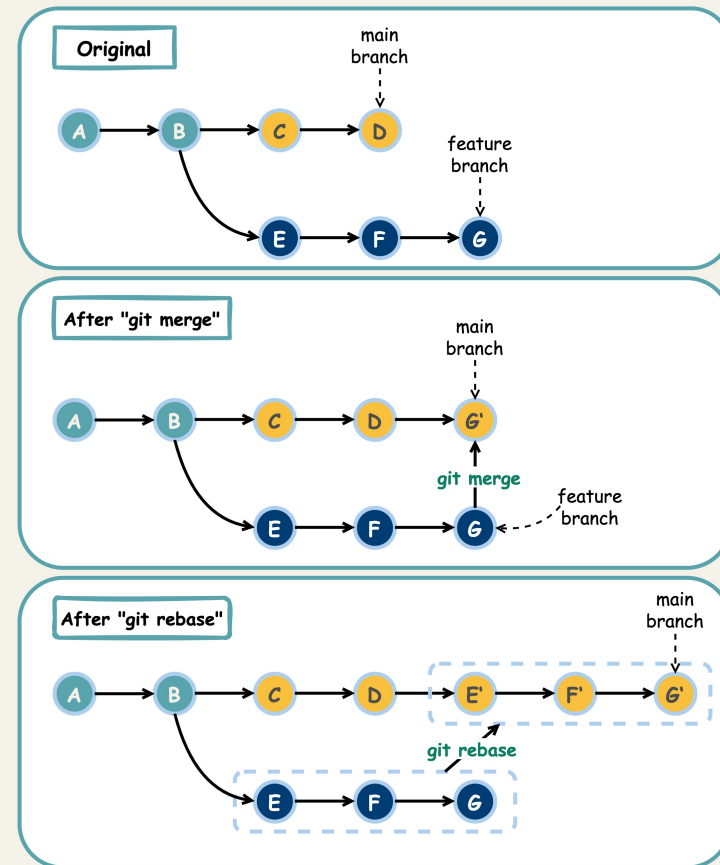
```
git checkout -b my-new-feature  
touch feature.md  
git add feature.md  
git commit  
# some days later  
git fetch  
git rebase -i origin/main  
git push --force-with-lease
```

Kein Force-Push auf default Branches!

Intermediate: `git rebase` vs `git merge`

Git Merge vs. Git Rebase

 blog.bytebytego.com



Intermediate: `git reset`

verwerfen aller Änderungen

```
git reset --
```

Verwerfen der Änderungen an einer Datei

```
git reset -- example.file
```

Zurücksetzen auf zB den Stand in einem remote

```
git reset --hard origin/master
```

github (gitlab, codeberg, ...)

- Social Network für Entwickler
- Hosted git mit Drumherum
- Wiki, Diskussionen, issue tracker, CI, ...

github: Pull Requests

Zusammenarbeit mit anderen
Beiträge zu fremden git Repositories
Review, Feedback, Diskussion
CI Pipelines mit Tests/Checks etc.

github: Fork

Für die Arbeit an Projekten, bei denen man nicht Mitglied ist
Private Kopie mit Verknüpfung zum `upstream` Repository
Pullrequests vom `fork` zu `upstream`

github: Actions (CI/CD)

Einen Computer mit dem Repo definierte Dinge tun lassen

- linting
- static code analysis
- build
- test
- release
- ...

github: Actions - Trigger

- Schedule
- Pull Request Events
- Commit Events
- [github Doku: workflow triggers](#)

github: Actions Beispiele

- [Veröffentlichung dieser Präsentation](#)
- [Build und Release von Docker Containern](#)
- [Versand der Reminder fürs Plenum](#)

Advanced: Conventional Commits

Was schreibe ich eigentlich in meine Commit Messages rein und warum?

Conventional Commits

Semantic Versioning

Keep a Changelog

Advanced: gitmoji 🤪

Projekt unsere git history soll schöner werden

gitmoji.dev

Links

[git docs](#)
[oh-shit-git](#)

Bildquellen

bytebytego.com

Lizenz

