

Git-Workflows im Alltag

Valentin Haenel

Freelance Consultant and Software Developer

<http://haenel.co>

@esc_

2015-03-21 @ CLT

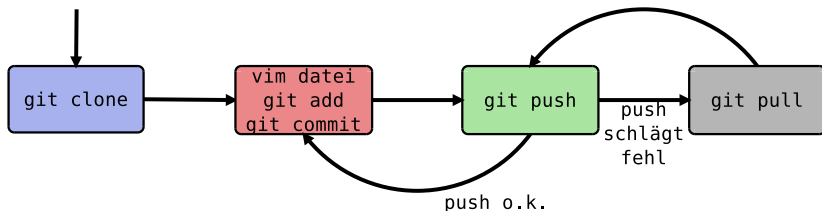


Version: v0.3.0

<https://github.com/esc/clt-2015-git-workflows>

This work is licensed under the *Creative Commons Attribution-ShareAlike 4.0 License*.

Push 'n' Pull Workflow



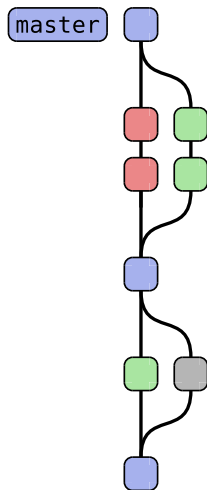
❶ Lokale Änderungen

- `vim datei`
- `git add datei`
- `git commit -m "msg"`

❷ Änderungen veröffentlichen

- `git push`
- Wenn push fehlschlägt:
- `git pull`, dann `git push`

Push 'n' Pull – Resultat

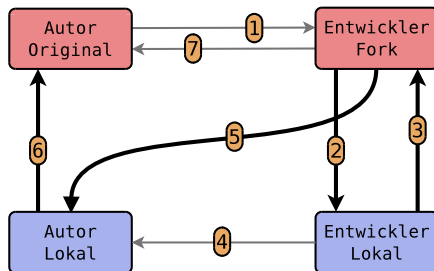


- Vorteile

- Leicht für Anfänger
- Nur weniger Kommandos

- Nachteile

- Es entstehen Merge-Commits
- »Aber wir arbeiten doch alle auf master?!«
- Rebase ist eine Option (für Anfänger?)

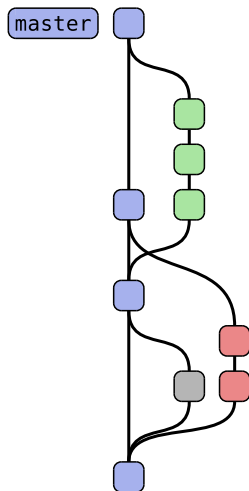


Entwickler:

- 1 Forkt ein Repository
- 2 Klont seinen Fork und macht Commits in einem Feature-Branch
- 3 Pusht den Feature-Branch in seinen Fork
- 4 Eröffnet einen Pull-Request

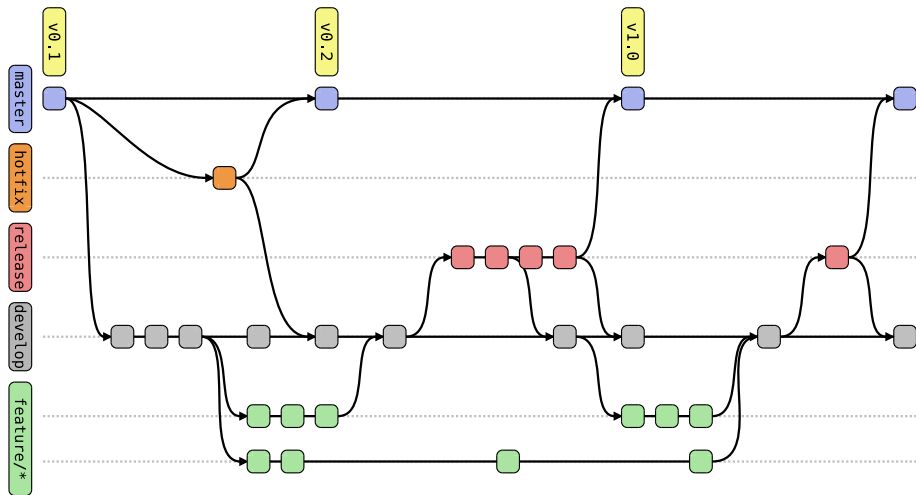
Autor:

- 5 Fetcht den Feature-Branch und mergt ihn
- 6 Pusht ins Original
- 7 **oder:** Mergt den Pull-Request über das Webinterface

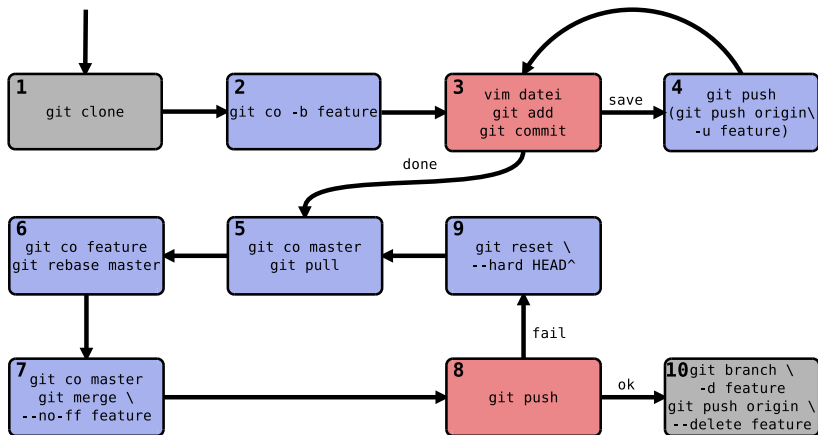


- Vorteile
 - Feature-Branches
- Nachteile
 - Feature-Branches basieren vor dem Merge evtl. nicht auf dem aktuellen Master
 - »Durcheinander«

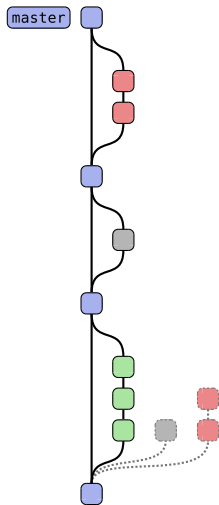
Gitflow – pdftk gitflow.pdf cat 1E output gitflow-rotated.pdf



Rebase 'n' Force-Merge Workflow



Rebase 'n' Force-Merge – Resultat



- Vorteile

- Saubere history
- Feature-Branches
- Merges sinnvoll

- Nachteile

- Verständnis von Git gebraucht
- Mehrere Kommandos nötig