Git für FLEUR

Ingo Heimbach (i.heimbach@fz-juelich.de)

3. Mai 2016

Inhaltsverzeichnis

1.1 Weshalb dieser Vortrag?

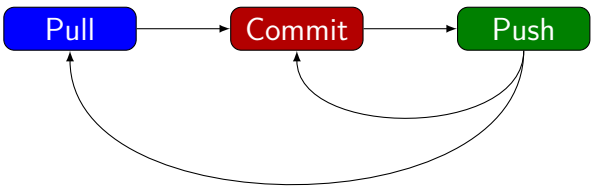
- Ziel: Mehr **Struktur** für die Code-Verwaltung

- Git alleine
  - liefert **Grundlage** zur Kollaboration
  - gibt **keine Vorgaben** zur Struktur/Arbeitsweise!

⇒ Weitere **Konventionen** in der Gruppe **notwendig**

1.2 Linearer Workflow

- **Einfachste** Art Git zu verwenden



- An CVS/SVN angelehnt

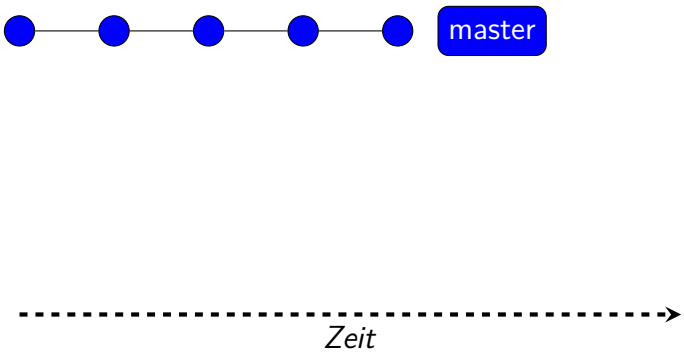
- **Simpel**, **wenig Overhead**

- Stabile und Entwicklungsversionen miteinander **vermengt**

1.3 Gitflow für FLEUR

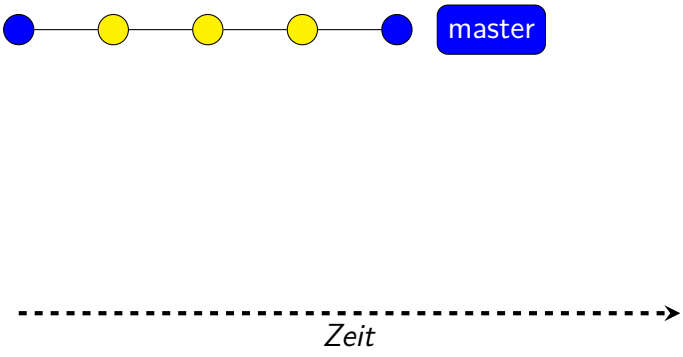
i)

Simpler, **linearer Workflow**:



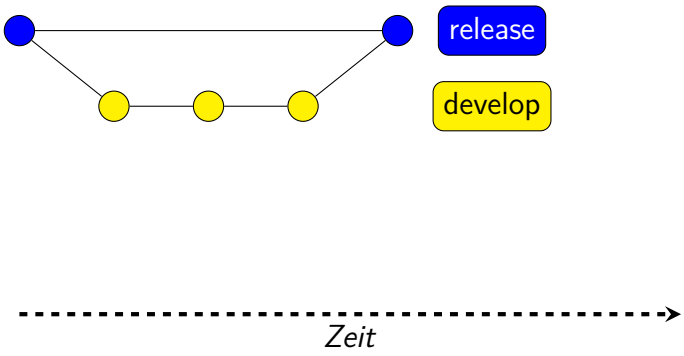
ii)

Differenzierung zwischen Release- und Entwicklungsversionen:



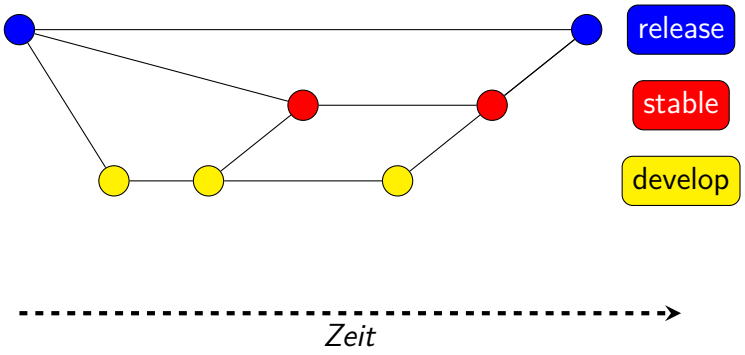
iii)

Differenzierung zwischen Release- und Entwicklungsversionen:



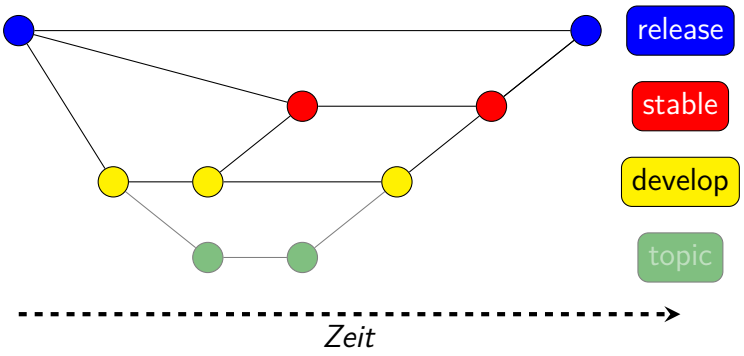
iv)

Zusätzliche Zwischenebene mit **stabilen** Versionen:



v)

Optionale Ebene für **längere Entwicklungen**:



vi)

Release-Version per **Tag** markieren:

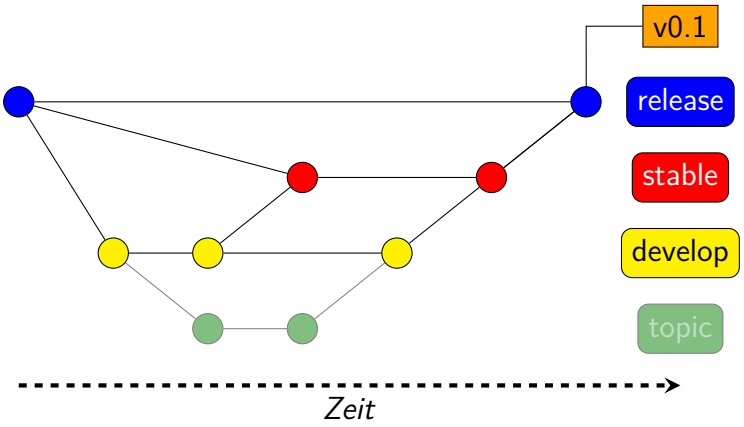


Abbildung 1.3.1: Verbesserter Workflow

1.4 Restriktionen

- **Entwicklergruppen**:
  - Hauptentwickler
  - Standardentwickler

- Nur Hauptentwickler dürfen Commits und Tags zu stable und release hinzufügen
- develop darf von **allen** Entwicklern gepusht werden
- In stable und release dürfen **ausschließlich Merge**-Commits gepusht werden

1.5 Anwendungsbeispiel

1.5.1 Code beitragen

1.5.1.1 Kleine Änderungen

```
git clone git@ifflinux:fleur && cd fleur
git checkout --track origin/develop
# Entwickeln
git commit -a
git push
```

Alternativ bei schon geklontem Repository:

```
cd fleur
git checkout develop
git pull
# Entwickeln
git commit -a
git push
```

1.5.1.2 Größere Änderungen / Langzeitentwicklungen

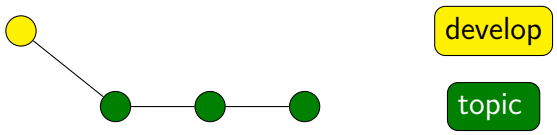
```
git checkout develop
git pull
git checkout -b <feature-Name>
# Entwickeln, evtl. git add (neue Dateien)
git commit -a
# Weiterentwickeln, evtl. git add (neue Dateien)
git commit -a
...
git checkout develop
git merge --no-ff <feature-Name>
git push
```

`--no-ff` nicht vergessen!
Mergen in **stable** und **release** funktioniert analog

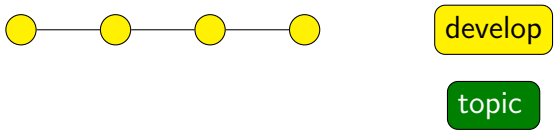
1.5.1.3 Wieso `--no-ff`?

- **Ohne** `--no-ff`:

i)

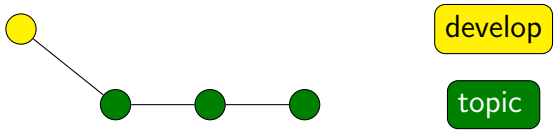


ii)

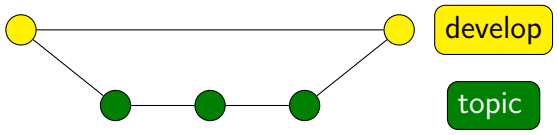


- **Mit** `--no-ff`:

i)



ii)



1.6 Zugang zum Git-Server

- Lesender Zugriff:

```
ssh git@ifflinux perms fleur + READERS <Mail-Adresse>
```

- Schreibender Zugriff:

```
ssh git@ifflinux perms fleur + WRITERS <Mail-Adresse>
```

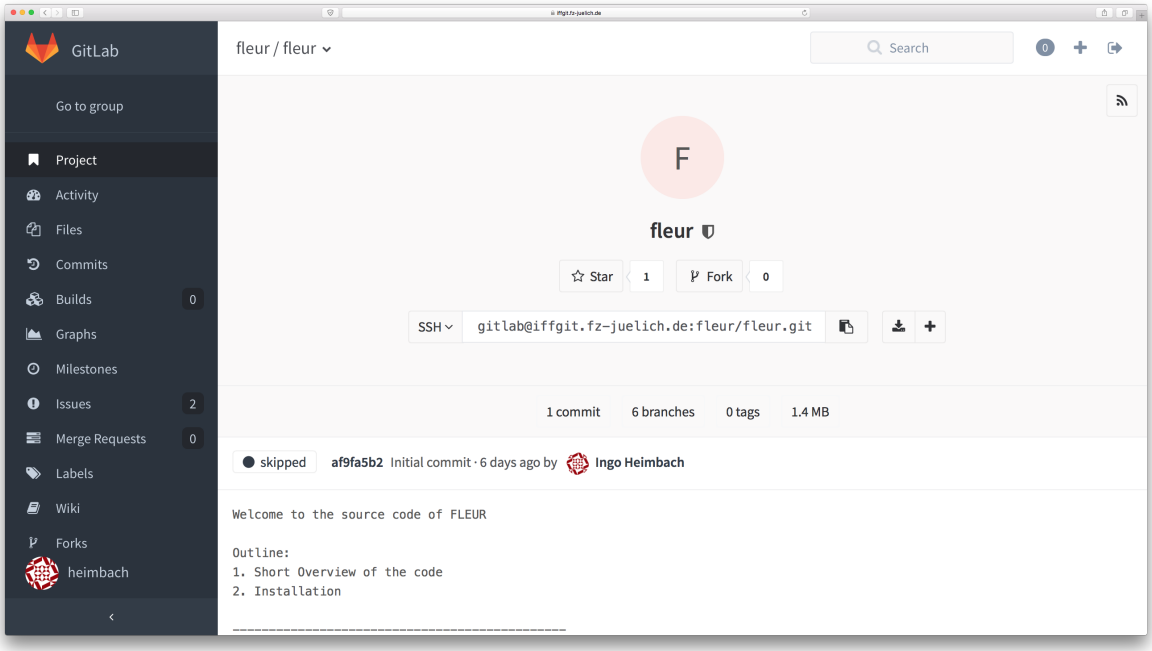
- Hauptentwickler hinzufügen:

```
ssh git@ifflinux perms fleur + OWNERS <Mail-Adresse>
```

1.7 GitLab

- Momentaner Git-Server:
  - **Gitolite**
  - Konfiguration nur per ssh, keine Weboberfläche
- Mittelfristig: Freigabe eines **GitLab**-Servers
  - **Webzugriff** inkl. Wiki, Issue-Tracker, grafische Logs usw.
  - Weiterhin per Kommandozeile steuerbar
  - Testphase: FLEUR wird **readonly** auf GitLab synchronisiert

  ⇒ GitLab kann mit FLEUR getestet werden
  - `https://iffgit.fz-juelich.de/`



1.8 Zugang zum Git-Server

1. RSA-Key-Paar erzeugen, falls nicht vorhanden:

```
ssh-keygen -b 4096 -t rsa -C<Mail-Adresse>
```

2. Öffentlichen Schlüsselteil (**`.pub`**) an **`i.heimbach@fz-juelich.de`** senden