Git Branches: Features unabhängig entwickeln und mit Git verwalten

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

Neues Feature entwickeln/ausprobieren

A---B---C master

Neues Feature entwickeln/ausprobieren

A---B---C master

- 1. git branch wuppie
- 2. git checkout wuppie oder git switch wuppie

Alternativ: git checkout -b wuppie oder git switch -c wuppie

A---B---C master, wuppie

Arbeiten im Entwicklungszweig ...

```
D wuppie /
A---B---C master
```

Problem: Fehler im ausgelieferten Produkt

```
D wuppie
/
A---B---C master
```

- 1. git checkout master
- 2. git checkout -b fix
- 3. Änderungen in fix vornehmen ...

```
D wuppie
/
A---B---C master
\
E fix
```

Fix ist stabil: Integration in master

```
D wuppie
/
A---B---C master
\
E fix
```

- 1. git checkout master
- 2. git merge fix => fast forward von master
- 3. git branch -d fix

```
D wuppie
/
```

Feature weiter entwickeln ...

```
D---F wuppie
/
A---B---C---E master
```

- 1. git switch wuppie
- 2. Weitere Änderungen im Branch wuppie ...

Feature ist stabil: Integration in master

```
D---F wuppie D---F wuppie
/ => / \
A---B---C---E master A---B---C---G master
```

- 1. git checkout master
- 2. git merge wuppie
 - => Kein fast forward möglich: Git sucht nach gemeinsamen Vorgänger

Konflikte beim Mergen

\$ git merge wuppie

Auto-merging hero.java

CONFLICT (content): Merge conflict in hero.java

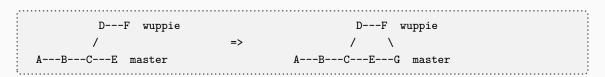
Automatic merge failed; fix conflicts and then commit the result.

Konflikte beim Mergen

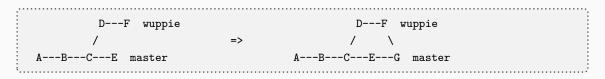
```
$ git merge wuppie
Auto-merging hero.java
CONFLICT (content): Merge conflict in hero.java
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<< HEAD:hero.java
public void getActiveAnimation() {
    return null;
======
public Animation getActiveAnimation() {
    return this.idleAnimation;
>>>>>> wuppie:hero.java
```

Rebasen: Verschieben von Branches



Rebasen: Verschieben von Branches



git rebase master wuppie

D'---F' wuppie / A---B---C---E master

Don't lose your HEAD

- Branches sind wie Zeiger auf letzten Stand (Commit) eines Zweiges
- HEAD: Spezieller Pointer
 - Zeigt auf den aktuellen Branch der Workingcopy
- Früheren Commit auschecken (ohne Branch): "headless state"
 - Workingcopy ist auf früherem Commit
 - Kein Branch => Änderungen gehen verloren!

Wrap-Up

- Anlegen von Branches mit git branch
- Umschalten der Workingcopy auf anderen Branch: git checkout oder git switch
- Mergen von Branches und Auflösen von Konflikten: git merge
- Verschieben von Branches mit git rebase

LICENSE



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.