

5.

Git Teil 2

Remotes

- Andere Nodes/Klone des Repositories
 - z.B. Original oder Fork auf GitHub

Remotes auflisten

```
$ git remote
origin
$ git remote -v
origin git@github.com:LukasKalbertodt/pir-task-wwacker-ssorglos.git (fetch)
origin git@github.com:LukasKalbertodt/pir-task-wwacker-ssorglos.git (push)
```

Identifiziert durch URL!

Namen sind nur lokale
Kürzel für die URLs



- Typische Namen für Remotes
 - **origin**: Standardname, oft der eigene Fork
 - **upstream**: oft Bezeichnung für Original-Repository

Remotes: weitere Befehle

Remote hinzufügen

```
$ git remote add <name> <url>
```

Remote löschen

```
$ git remote remove <name>
```

Remotes umbenennen

```
$ git remote rename <old-name> <new-name>
```

- URL-Typen:
 - <https://github.com/PirTest/pir-task-wwacker-ssorglos.git> (HTTP)
 - <git@github.com:PirTest/pir-task-wwacker-ssorglos.git> (SSH)
 - Lokale Remotes ebenfalls möglich! (z.B. ~/my_other_clone/)

Remote Branches

Branches auflisten

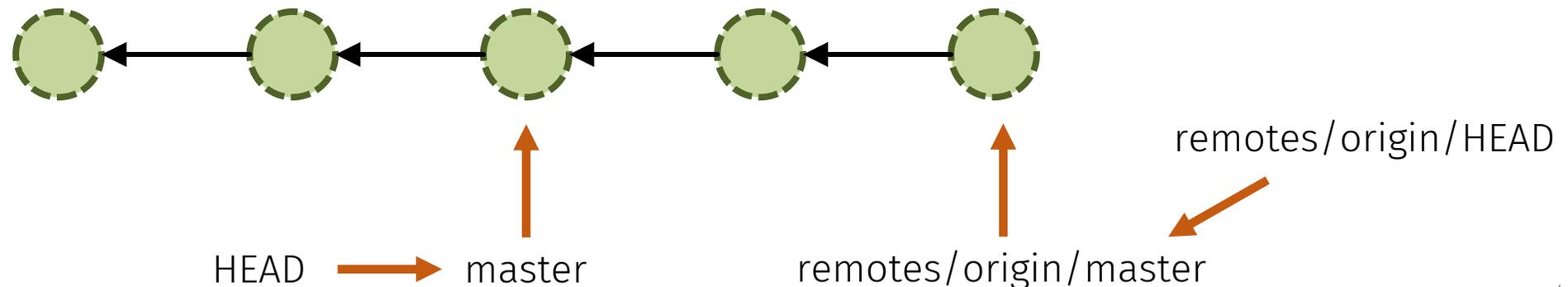
```
$ git branch
* master
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
```

Lokale Branches
(master)

≠

Remote Branches
(remotes/origin/master)

- Lokaler „Cache“ von Branches auf dem Remote
 - Zeigt den Stand, als das letzte mal der Remote kontaktiert wurde

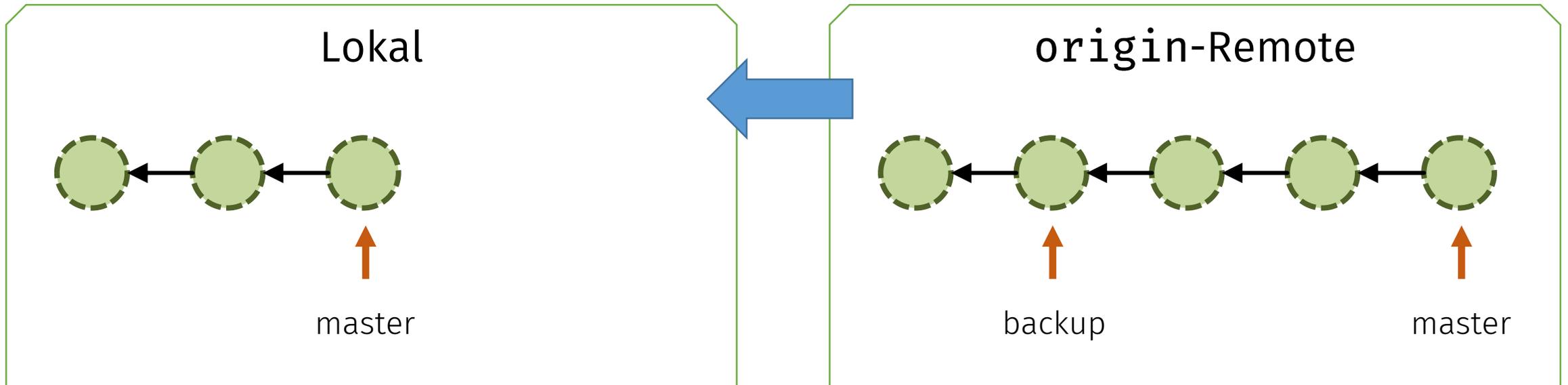


Remote Branches aktualisieren

„Git Fetch“

```
$ git fetch <remote>
```

- Kontaktiert Remote, aktualisiert Remote Branches
- Läd alle Commits zur Vervollständigung der History runter

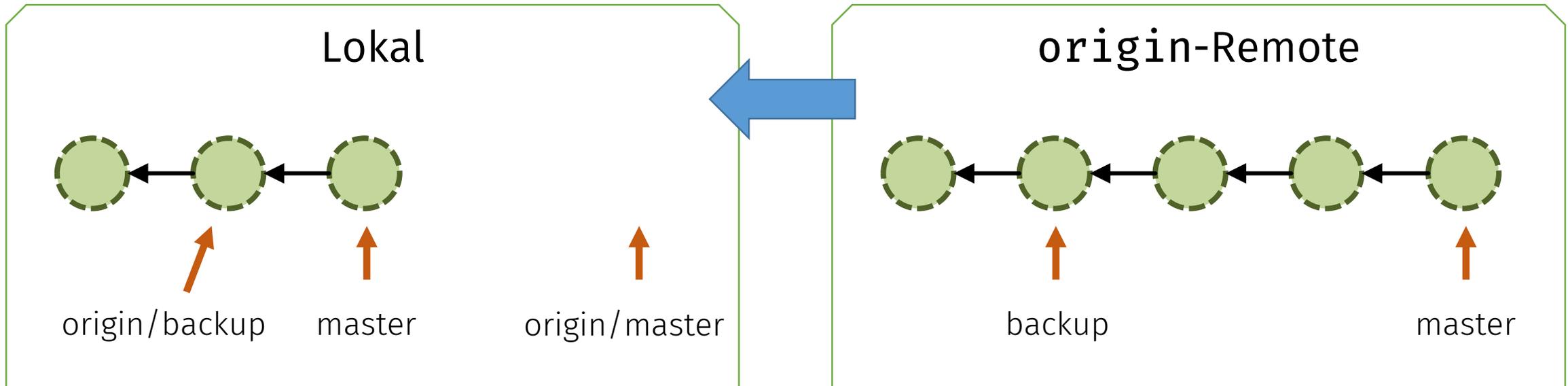


Remote Branches aktualisieren

„Git Fetch“

```
$ git fetch <remote>
```

- Kontaktiert Remote, aktualisiert Remote Branches
- Läd alle Commits zur Vervollständigung der History runter

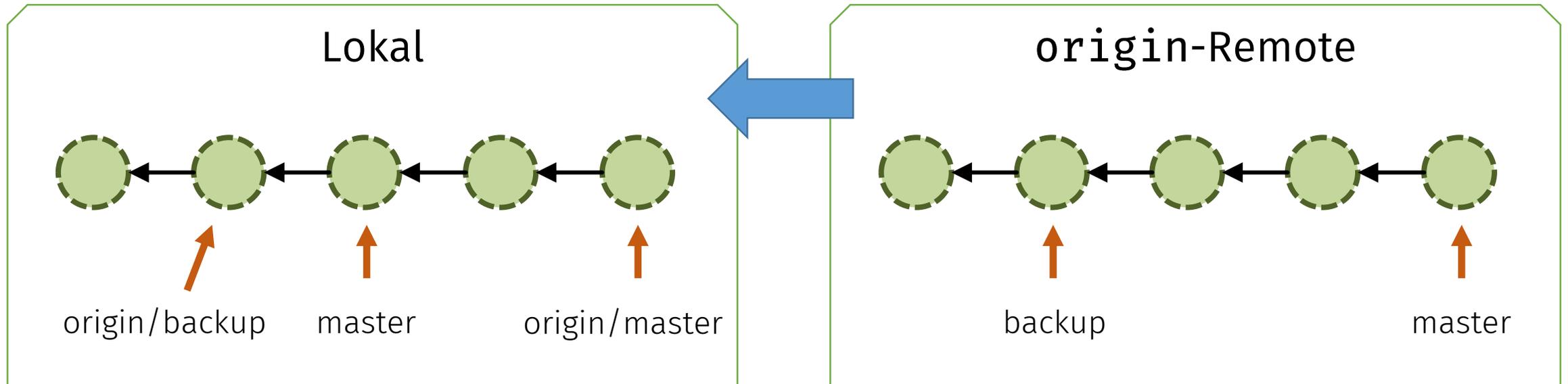


Remote Branches aktualisieren

„Git Fetch“

```
$ git fetch <remote>
```

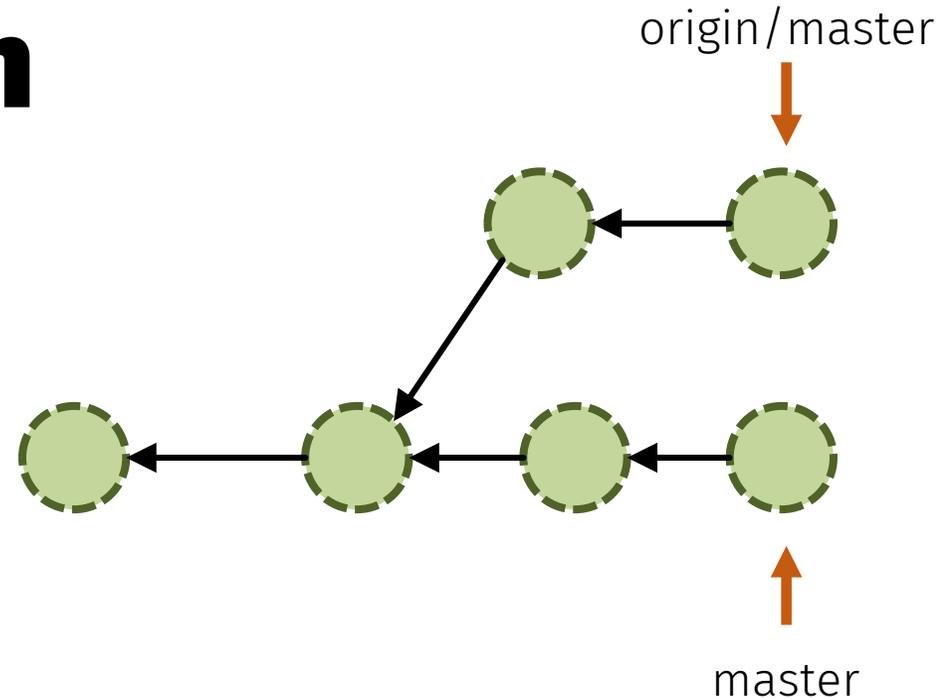
- Kontaktiert Remote, aktualisiert Remote Branches
- Läd alle Commits zur Vervollständigung der History runter



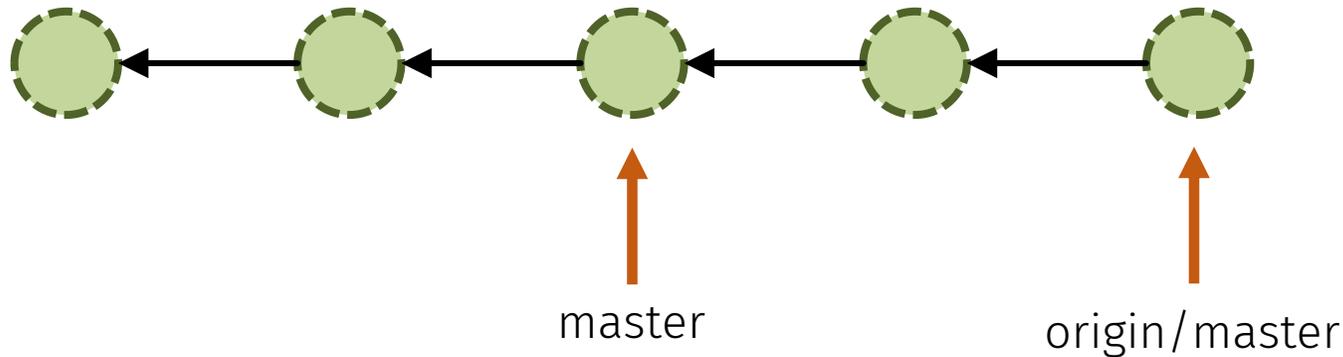
Branches mergen

```
$ git merge <branch>
```

- Integriert Änderungen aus anderem Branch in den aktuellen



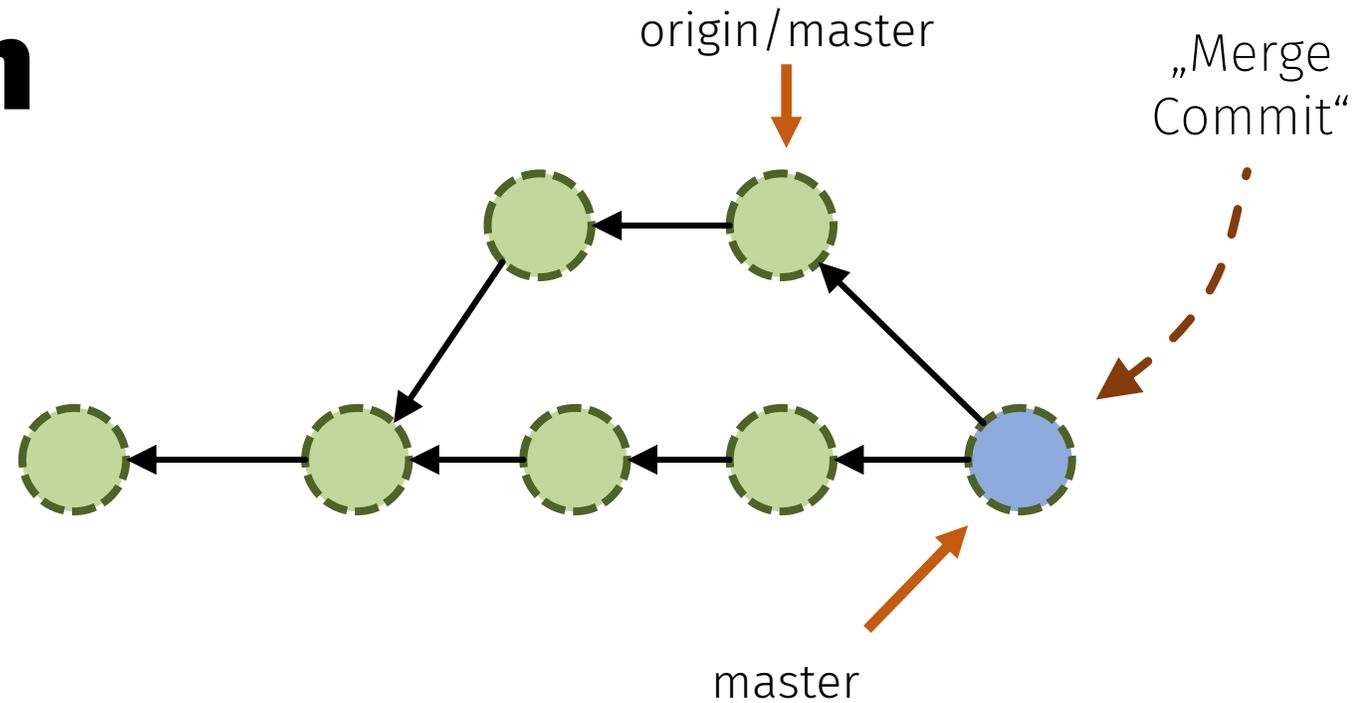
```
$ git checkout master  
$ git merge origin/master
```



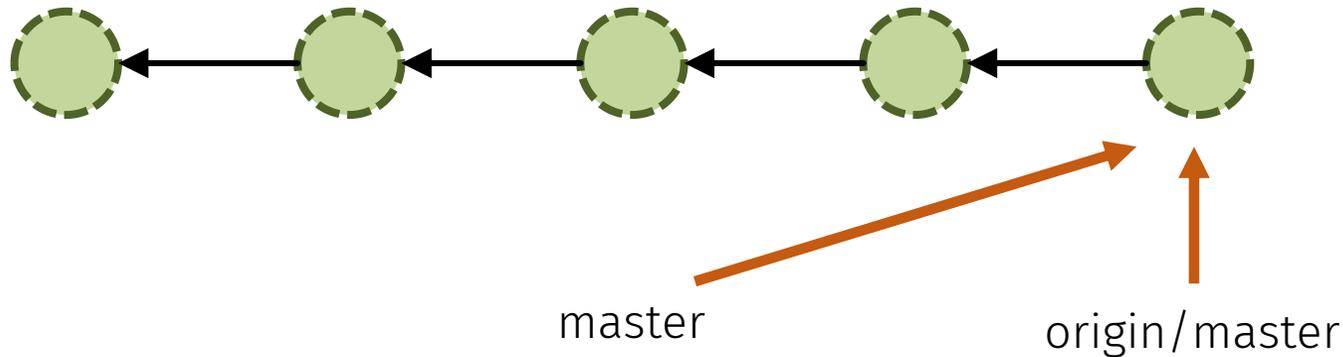
Branches mergen

```
$ git merge <branch>
```

- Integriert Änderungen aus anderem Branch in den aktuellen



```
$ git checkout master  
$ git merge origin/master
```



„Fast Forward Merge“

→ Nur Referenz-Update

Merge Conflicts

```
$ git merge --abort
```

```
$ git merge master
To Auto-merging src/hello.rs
CONFLICT (content): Merge conflict in src/hello.rs
Automatic merge failed; fix conflicts and then commit the result.
```

hello.rs

```
fn main() {
<<<<<<< HEAD
    println!("Hello peter");
=====
    println!("Hello anna");
>>>>>> master
}
```

„Meins“
(aktueller Br.)

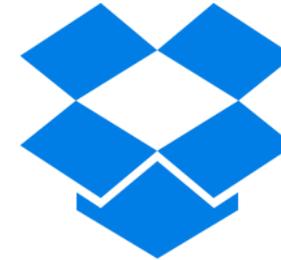
„Deren“
(zu mergender Br.)

```
$ git status
# On branch foo
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:   src/hello.rs
#
```

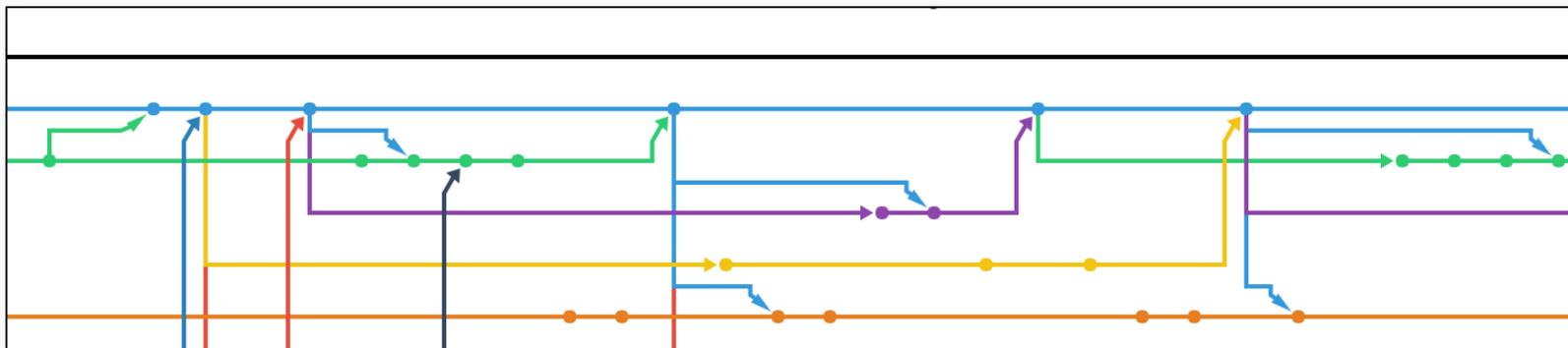
```
$ git add src/hello.rs
$ git commit
```

Aufpassen mit dem Mergen!

- Zu viele Merge-Commits vermüllen History
- Horror-Szenario:
 1. Bob und Anna starten beim selben Commit
 2. Jeder erstellt einen eigenen commit
 3. Bob möchte die Änderungen von Anna: **git merge anna**
 4. Wiederhole:
 1. Anna erstellt einen weiteren Commit
 2. Bob möchte nochmal die neuen Änderungen: **git merge anna**



≠



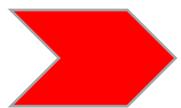
Git Pull

```
$ git pull <remote> <branch>
```

≈

```
$ git fetch <remote>  
$ git merge <remote>/<branch>
```

- Vereinfacht fetch und merge Operationen
- Aber: Birgt dadurch Gefahren (wie **git commit -a**)
 - Leute denken: **git pull** ist ein Dropbox-Update 😞



**Immer bewusst daran
denken, was es tut!**

Tracking Branches

```
$ git pull <remote> <branch>
```

```
$ git push <remote> <branch>
```

- Immer Argumente angeben?
 - Weglassen durch Einrichten von Tracking Branches

```
$ git branch -u <remote>/<branch> # current branch now tracks remote branch
```

- Lokalen und Remote Branch verbinden
- Push und Pull funktionieren ohne Argumente

Git Push

```
$ git push origin master
To https://github.com/USERNAME/REPOSITORY.git
 ! [rejected]          master -> master (non-fast-forward)
 ...
```

- Push aktualisiert Branches, merged nicht
- Pushen nur möglich, wenn Merge FF wäre