Introduction to Git

Naoki Pross — `np@0hm.ch`

XX. March 2025

Table of Contents

What do we want?

The Problem

Synchronize data across multiple computers, with multiple people working on (possibly the same) files.

Linus' Wishes (The guy who invented Git)

- Synchronization *always* works
- Teamwork is possible and efficient
- Works offline
- Fast

intuitive or easy to use was not on his list!

Other Solutions?

Other Tools at Linus' Time

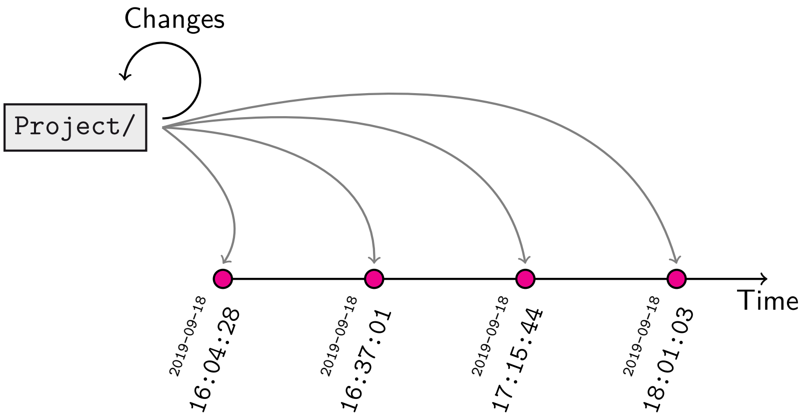
CVS	Slow to synchronize. CVS requires a centralized server which can get overloaded, was usually set up by the company IT.
E-Mail	People sent patch files to each other via email.

Other Tools Today

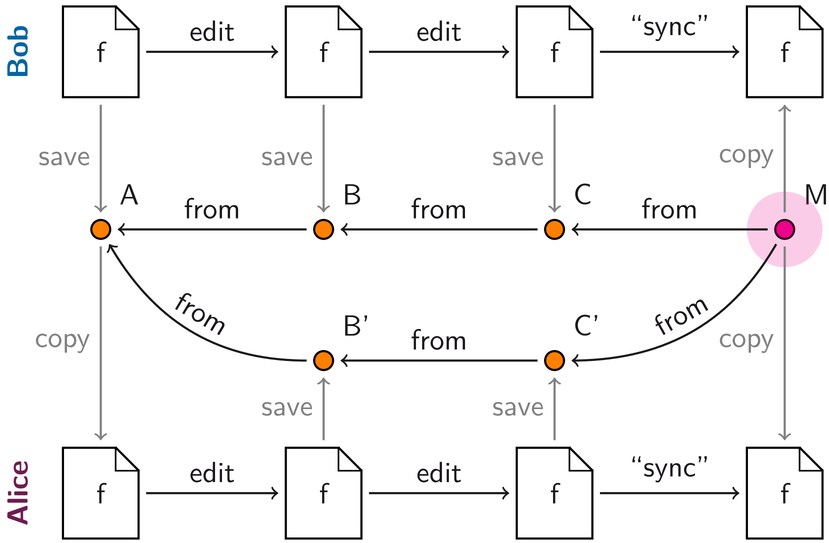
Cloud Storage	Does not work offline. Their whole business model is against you. You have no (real) control over when to sync.
Mercurial (hg)	Learn to walk (Git) before you run.

Table of Contents

Solving the Problem: Snapshots



Solving the Problem: Concurrent Changes I



Solving the Problem: Concurrent Changes II

High Level Overview

Store changes using a *directed acyclic graph* (DAG) called the *commit graph*.

- Nodes are saved points in time called *commits*
- Arcs point to state from which change was made
- Commits with multiple children (A) are *branching commits*
- Commits with multiple parents (M) are *merge commits*

Problems

1. We care about file content not the files itself
2. Alice and Bob are not working on the same computer
3. How do we merge changes?

Mathematical Digression: DAG

Directed Acyclic Graph

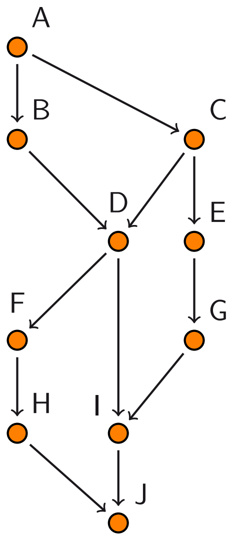
A DAG $G = (V, A)$ is defined by a finite set of vertices V and a finite set of *arcs* A and may not contain loops.

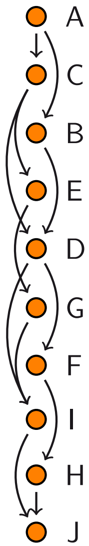
Partial Order

DAG have a partial order relation $u \succeq v$ for all $u, v \in V$.

Topological Order

A DAG $G = (V, A)$ has a total order $\succeq^*$ by having that for all $(u, v) \in A$ $u \succeq^* v$. If G has a Hamiltonian path $\succeq^*$ is unique.





Solving The Problem: Multiple Files

Solving The Problem: Other Computers I

remote branches

Solving the Problem: Other Computers II

push and fetch and pull

Table of Contents

Mathematical Digression: Hashes and Merkle DAG

“One-way fast” functions

Hash Function

A (cryptographic) *hash* function is an $h : \Omega \to \{0,1\}^d$ for a fixed hash length d such that:

1. Given $y = h(x)$ it is hard to find x
2. It is hard to find $x, y \in \Omega$ s.t. $h(x) = h(y)$
3. Given x it is hard to find y s.t. $h(x) = h(y)$
4. Given $h(x)$ and a function f it is hard to find $h(f(x))$

Merkle DAG

A Merkle DAG is a DAG $G = (V, A)$ with a hash

$$h : U \subseteq V \to \{0,1\}^d$$

that defines a label function

$$\ell(v) = h(\{v\} \cup \text{neighbors}^+(v))$$

Properties

- Immutable data structure
- Has cryptographic verification

Git Commits

Commit Contents

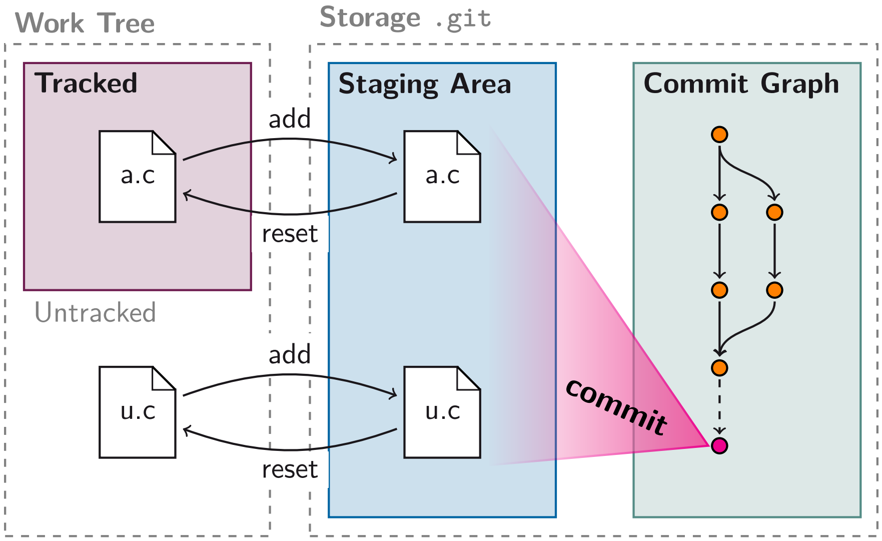
- Content (Blobs and Trees) hash
- Parent(s) commit(s) hash(es)
- Metadata: Author, Date, Message

Example

```
commit 1cfdf5c198f1c74c2f894067baf4670f5bca8e70
Author: Nao Pross <np@0hm.ch>
Date:   Wed Feb 9 19:53:06 2022 +0100

    Fix arrayobject.h path on Debian based distros

    On Debian Linux and its derivatives such as Ubuntu and LinuxMint, Python
    packages installed through the package manager are kept in a different
    non-standard directory called 'dist-packages' instead of the normal
    'site-packages' [1].

    To detect the Linux distribution the 'platform' library (part of the
    Python stdlib) provides a function 'platform.freedesktop_os_release()'
```

Git Repositories

The 3 (or 4) Conceptual Areas of Git



Branches, Remotes and your HEAD

Git Services (GitHub, GitLab, ...)

Forking and Pull / Merge Requests

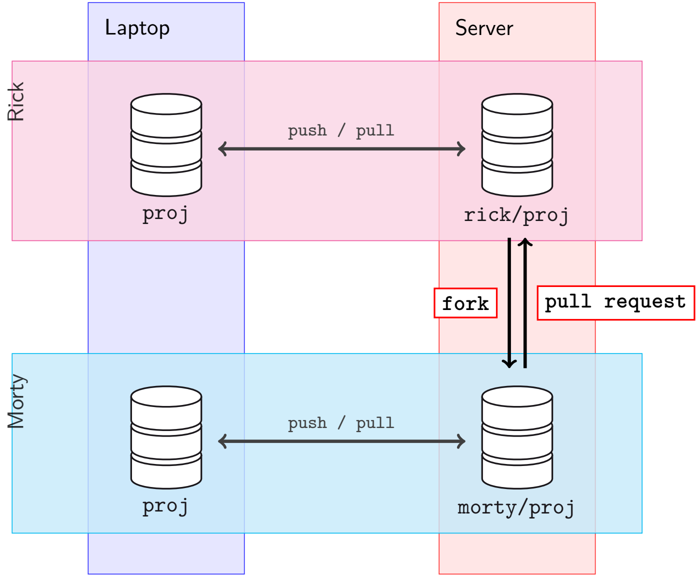
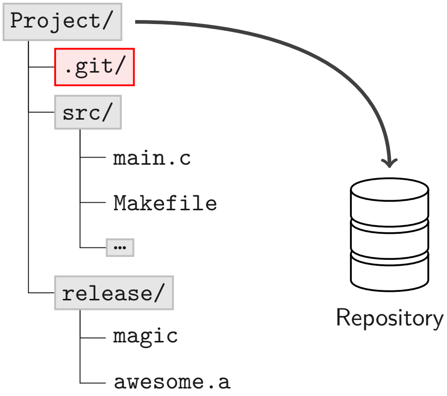


Table of Contents

Begriff: Repository



Für Git ein *Repo* ist eine Verzeichnis mit einer spezieller (unsichtbar) Unterverzeichnis `.git`

Man soll **nie** `.git` löschen.

Begriff: Commit

Ein Commit enthalt

- Die Änderungen der Dateien **“snapshot”**
- Autor Name + Email
- Zeitstempel
- **Eine Beschreibung der Änderungen**
- kryptographisches Hash der Dateien

**Commits können nicht ändert werden,
weil sie “Geschichte” des Projekts sind.**

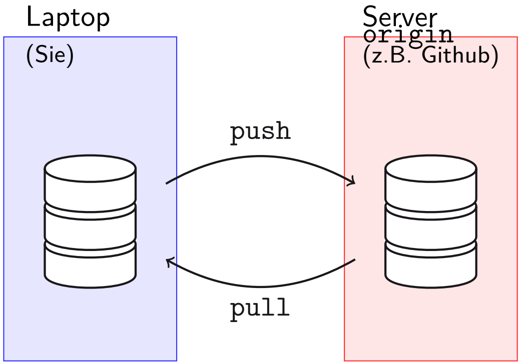
Aber was ist ein Commit?

Commit = "Logical Unit of Work"

Ein Commit kann auch sehr klein sein.
Generell kann man sagen:

je mehr Commits, desto besser

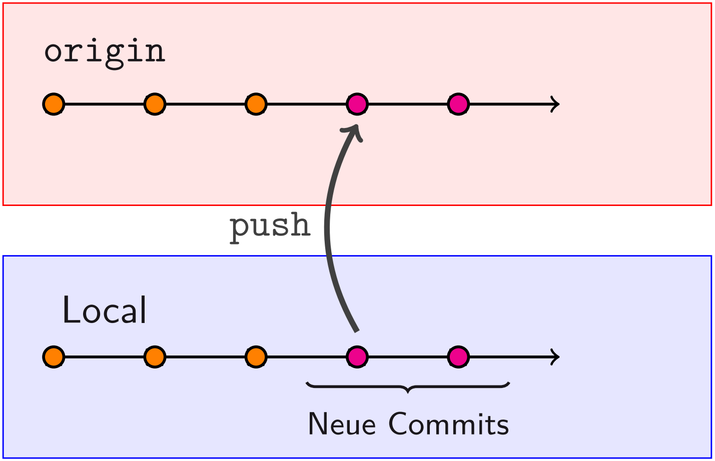
Begriff: Remote



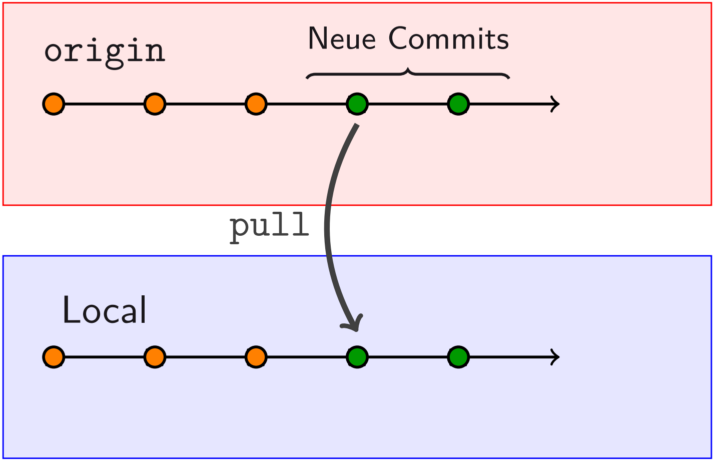
Ein *Remote* ist ein Clone (Kopie) des Repos auf eine andere Maschine

Remote können ein name haben, z.B. `origin`

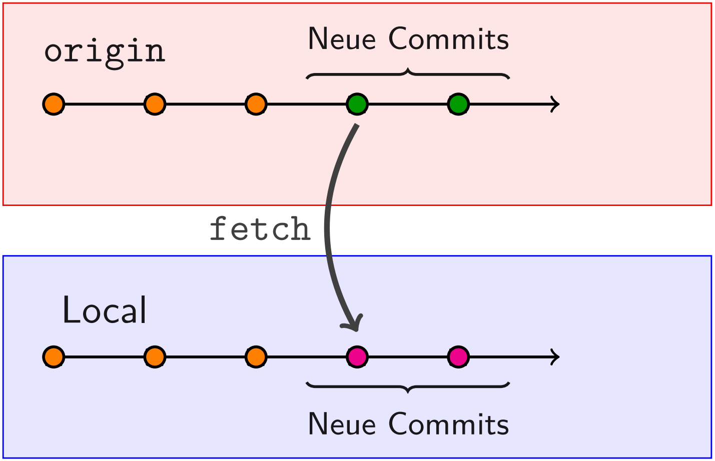
Zu einem Remote synchronisieren



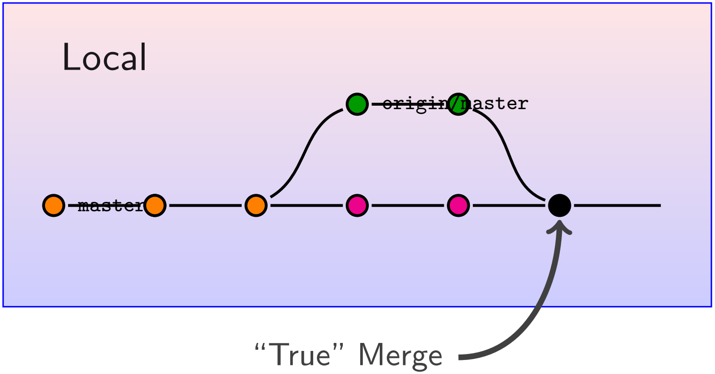
Von einem Remote synchronisieren



Merge



Begriff: 3-Way Merge



```
Merge branch 'origin/master' into master
```

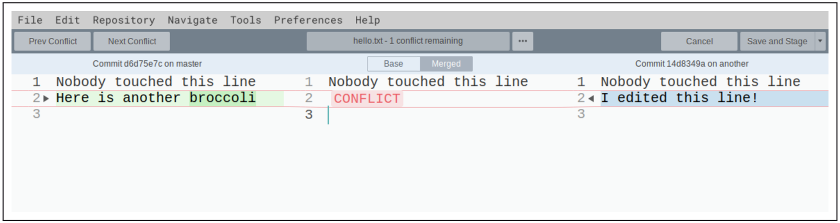
Konflikte

D.h. Was passiert wenn 2 Leute die gleiche Linie in einem Dokument bearbeiten?

Git kann nicht wissen welche version besser ist, und so fragt dir.

```
...
Here is a line that nobody touched
<<<<<<< HEAD
I have edited this line
=======
Someone else has also edited this line
>>>>>>> origin/master
...
```

Warum “3-Way”?


File Edit Repository Navigate Tools Preferences Help
Prev Conflict
Next Conflict
hello.txt - 1 conflict remaining
Cancel
Save and Stage
Base
Merged

Commit d6d75e7c on master

```
1 Nobody touched this line
2 Here is another broccoli
3
```

```
1 Nobody touched this line
2 CONFLICT
3
```

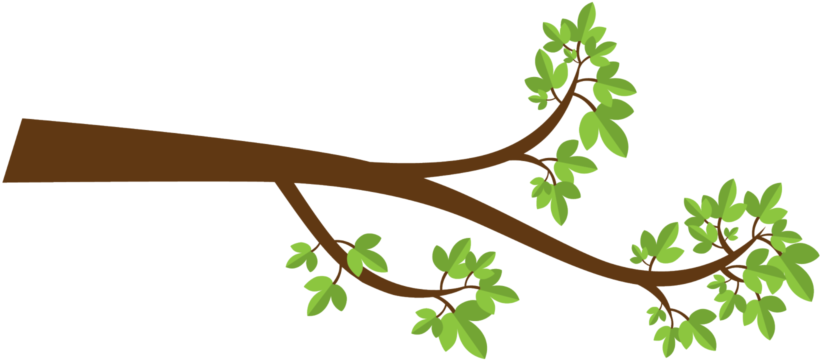
Commit 14d8349a on another

```
1 Nobody touched this line
2 I edited this line!
3
```

Figure 1: Sublime Merge

`pull` ist ein Alias für `fetch` + `merge`

Branch



Begriff: Branch

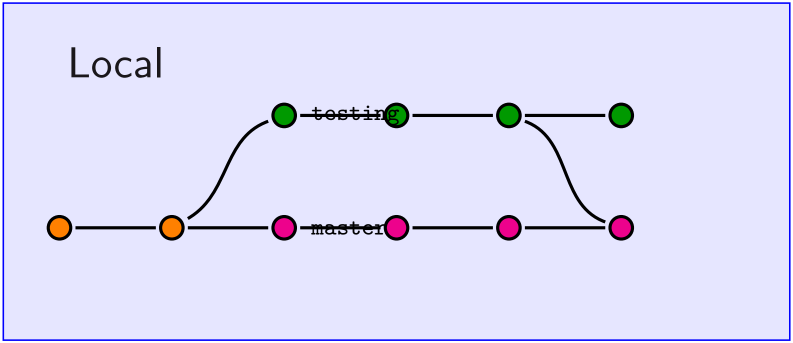


Table of Contents