Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Proseminar SoSe 2021
Softwareentwicklung in der Wissenschaft

Versionsverwaltung mit

git

Rana Mostafa

Inhaltsverzeichnis

Einleitung
- Was ist eine Versionsverwaltung? Welche Arten?
- Warum Versionsverwaltung?

Versionsverwaltung mit Git
- Git
- Wichtige Befehle zur Arbeit mit Git

Git: Repository
- Was ist eine Repository in Git?
- Lokales und remotes Arbeiten mit Git

Git: Zustände **und** Breiche
- Dateistatus(Lebenszyklus)
- Hauptbereichen eines Git-Projekts

Git: Branching and Merging
- Git: Branch
- Git: Merge
- Git-flow-Workflow

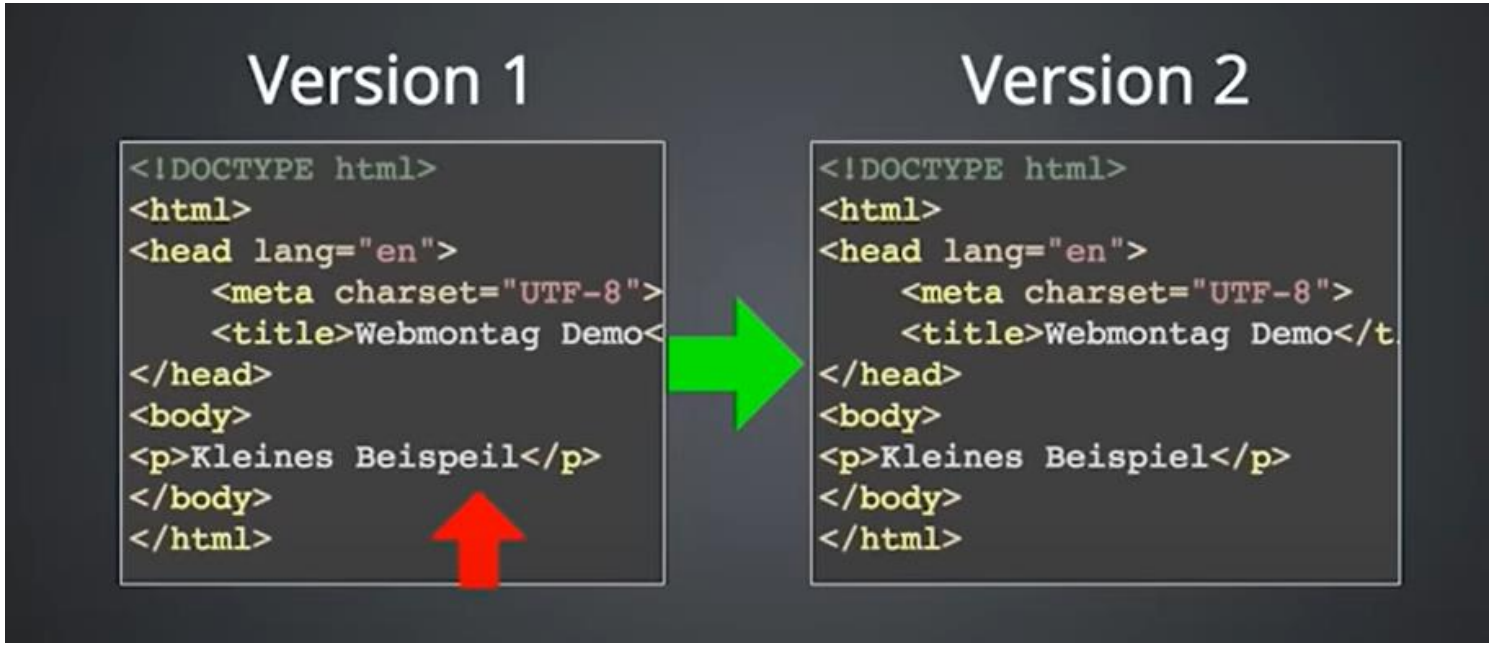
Git mit Kommandozeile(Getting Started)

Fazit

Einleitung

Was ist eine Versionsverwaltung? Welche Arten?

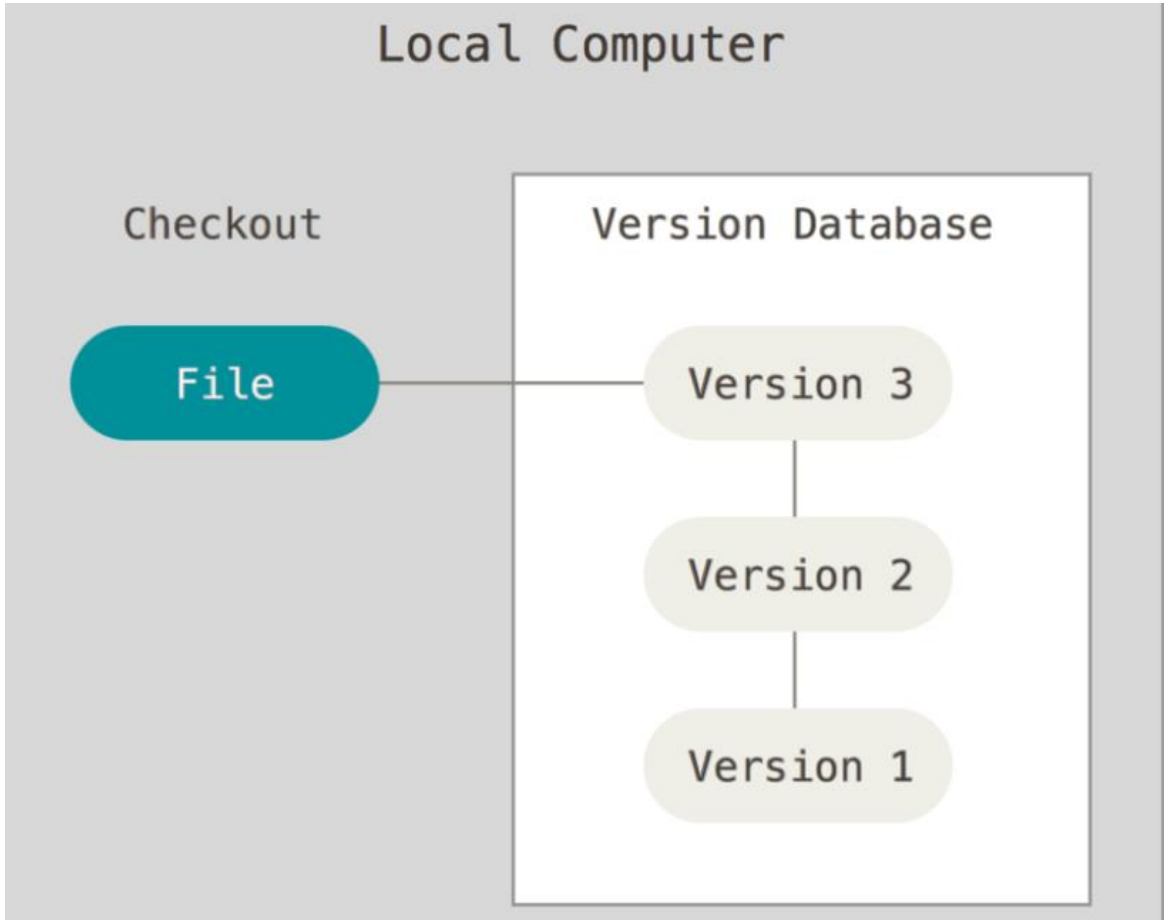
- Versionsverwaltung ist ein System, welches die Änderungen an einer oder einer Reihe von Dateien (z. B. Quellcode eines Programmes oder um Bilddateien in der Bildbearbeitung) über die Zeit hinweg protokolliert, sodass man später auf eine bestimmte Version zurückgreifen kann.



https://de.wikipedia.org/wiki/Versionsverwaltung

Lokale Versionsverwaltung

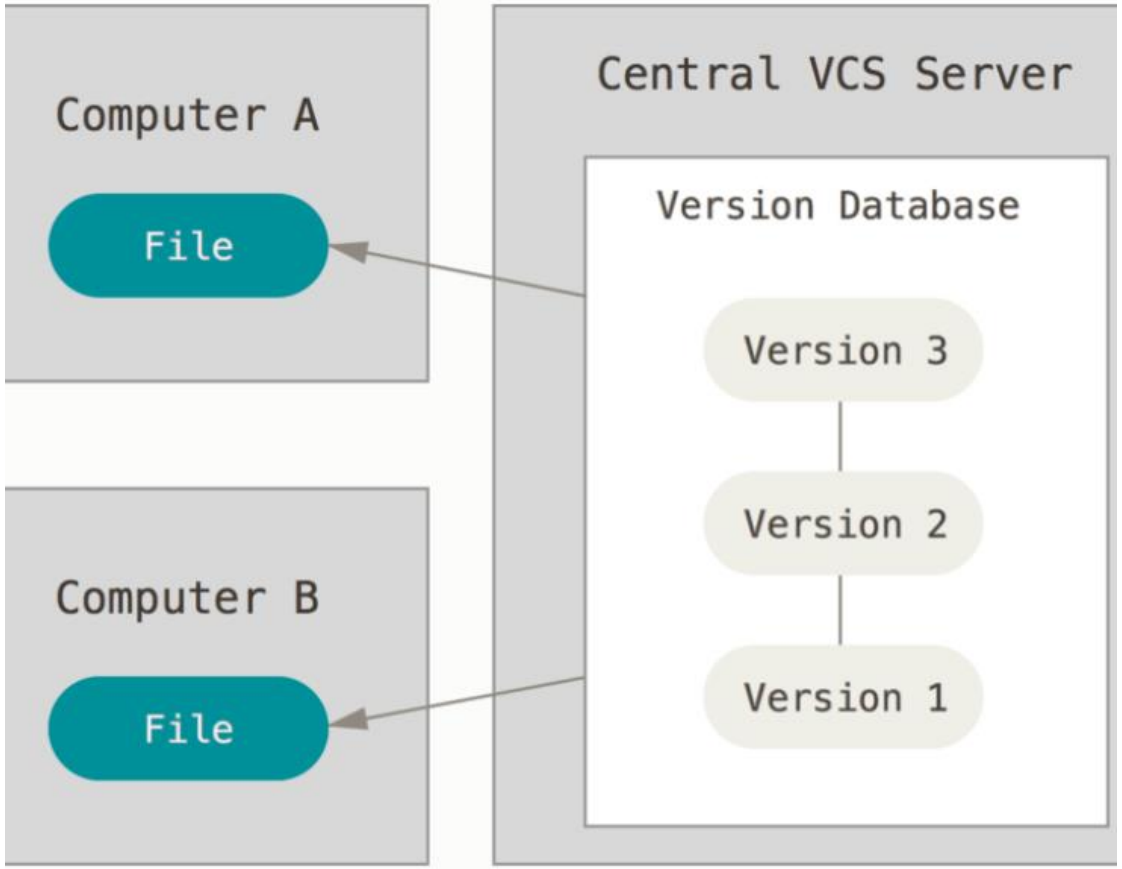
- Erste Generation von Versionsverwaltung.
- funktionieren nur auf einem Computer.
- oft nur eine einzige Datei versioniert.
- z. B. SCCS



https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F

Zentrale Versionsverwaltung

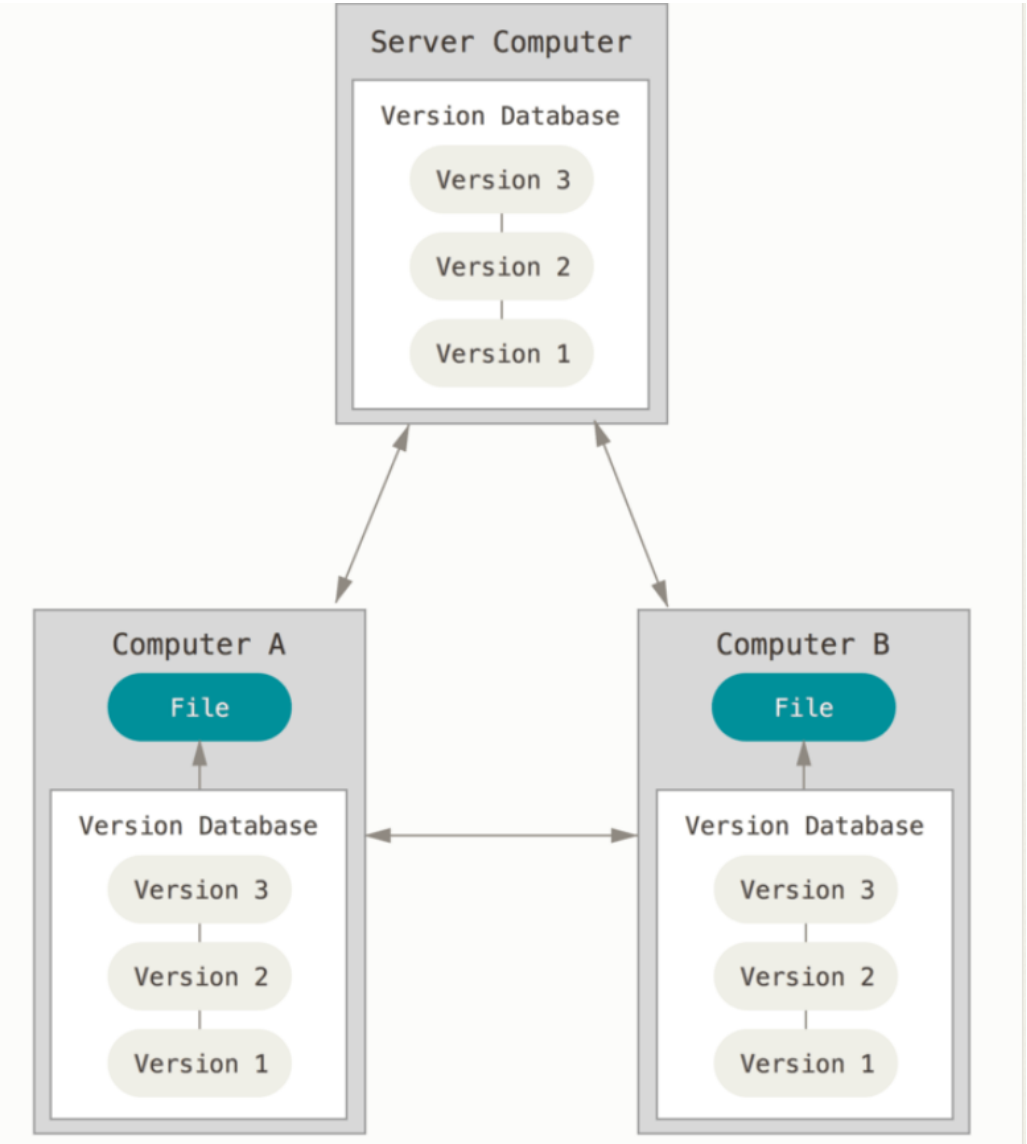
- Als Client-Server-System aufgebaut.
- Also, basieren auf einem zentralen Server, der alle versionierten Dateien verwaltet.
- Nur berechtigte Personen neue Versionen in das Archiv legen können.
- Nachteil: bei einem Ausfall das gesamte System lahmlegt.
- z. B. Subversion (SVN).



https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F

Verteilte Versionsverwaltung

- Änderungen können lokal verfolgt werden, ohne eine Verbindung zu einem Server aufbauen.
- Erhalten Anwender eine vollständige Kopie des Repositorys.
- z. B. Git.

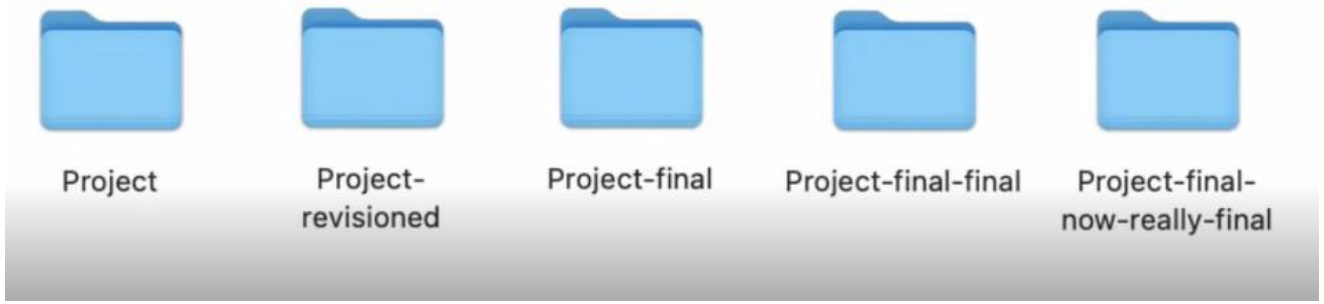


https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F

Einleitung

Warum Versionsverwaltung?

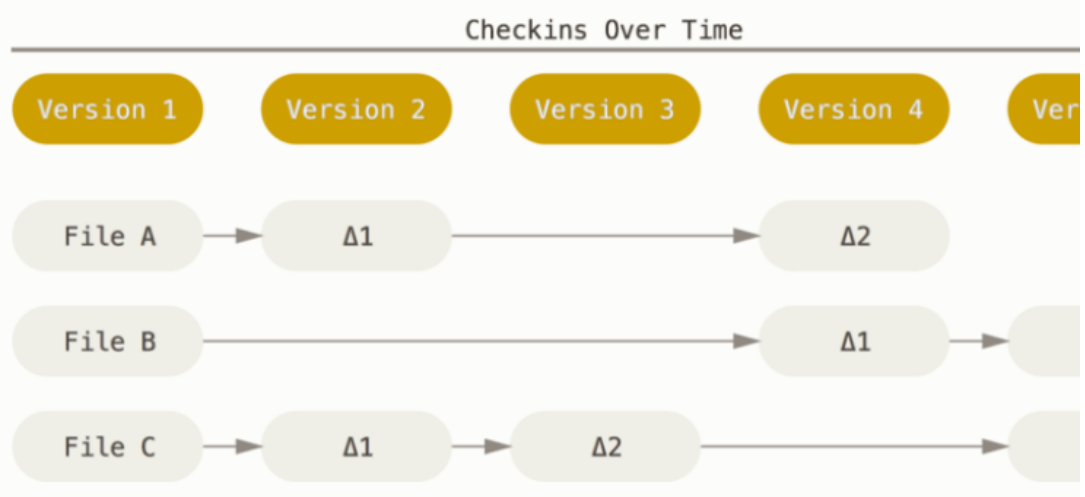
- Änderungen nachvollziehen, rückgängig machen und auf mehreren Rechnern synchronisieren.
- Strukturierte Zusammenarbeit der Teams am Projekt, auch asynchron & offline.
- Insbesondere in der Softwareentwicklung zur Verwaltung der Quelltexte.
- Erleichterte Fehlersuche/Debugging.
- ….

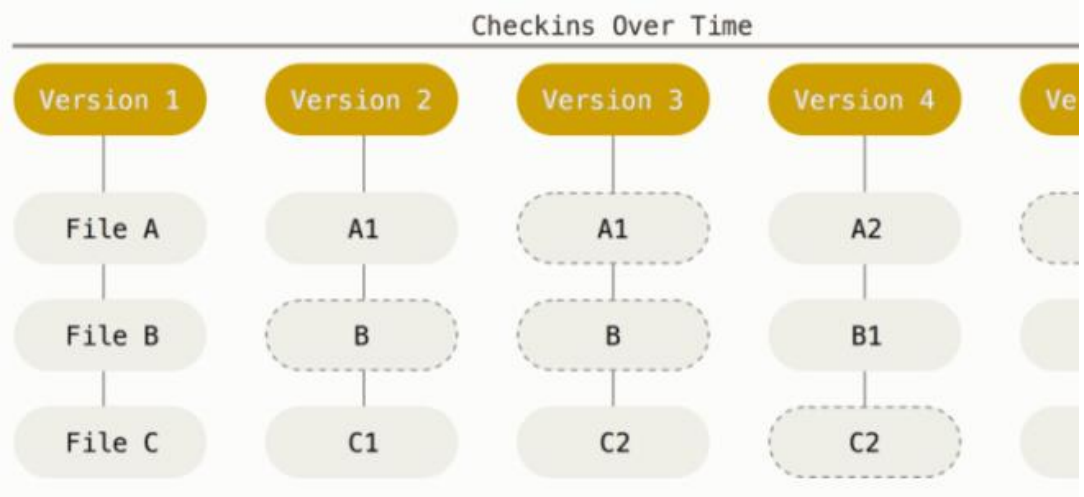


Versionsverwaltung mit Git

Git:

- Eins der wohl bekanntesten Versionskontroll-Systeme.
- Würde 2005 von Linux-Schöpfer Linus Thorvalds entwickelt und unter der freien GNU-GPLv2-Lizenz veröffentlicht.
- Versionsverwaltung stellt somit ein wichtiges Werkzeug im Softwareentwicklungsprozess dar.
- Snapshots statt Unterschiede (Ein Versionsstand in Git ist ein vollständiger Schnappschuss eines Projekts).





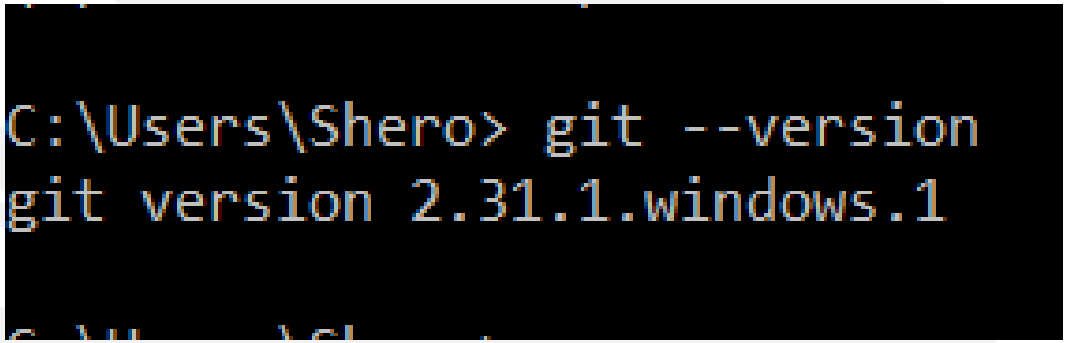
https://www.git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Git%3F

Versionsverwaltung mit Git

Git:

Installation: Git ist bei den meisten Distributionen bereits vorinstalliert.

Sonst: https://git-scm.com/downloads



Möglichkeiten zur Benutzung von Git:

- -Kommandozeile
- -integriert in Eclipse, IntelliJ oder Netbeans
- -GUIs

Wichtige Befehle zur Arbeit mit Git

- Git Commit:
  - Ein Schnappschuss eines bestimmten Versionsstandes zu einem bestimmten Zeitpunkt.
  - Commit enthält sowohl die Veränderungen als auch Metadaten, wie eine ID, den Autor der Veränderungen, Datum und Uhrzeit, und eine Nachricht (Commit Message), die die Veränderungen beschreibt.
- Git Clone: um ein bestehendes Remote-Repository in ein lokales Verzeichnis zu kopieren.
- Git Pull: Um Änderungen aus einem anderen Repository in das eigene zu übernehmen.
- Git Push: Um die im lokalen Repository vorliegenden Änderungen in ein Remote-Repository zu übertragen.

Git: Repository

Was ist eine Repository in Git?

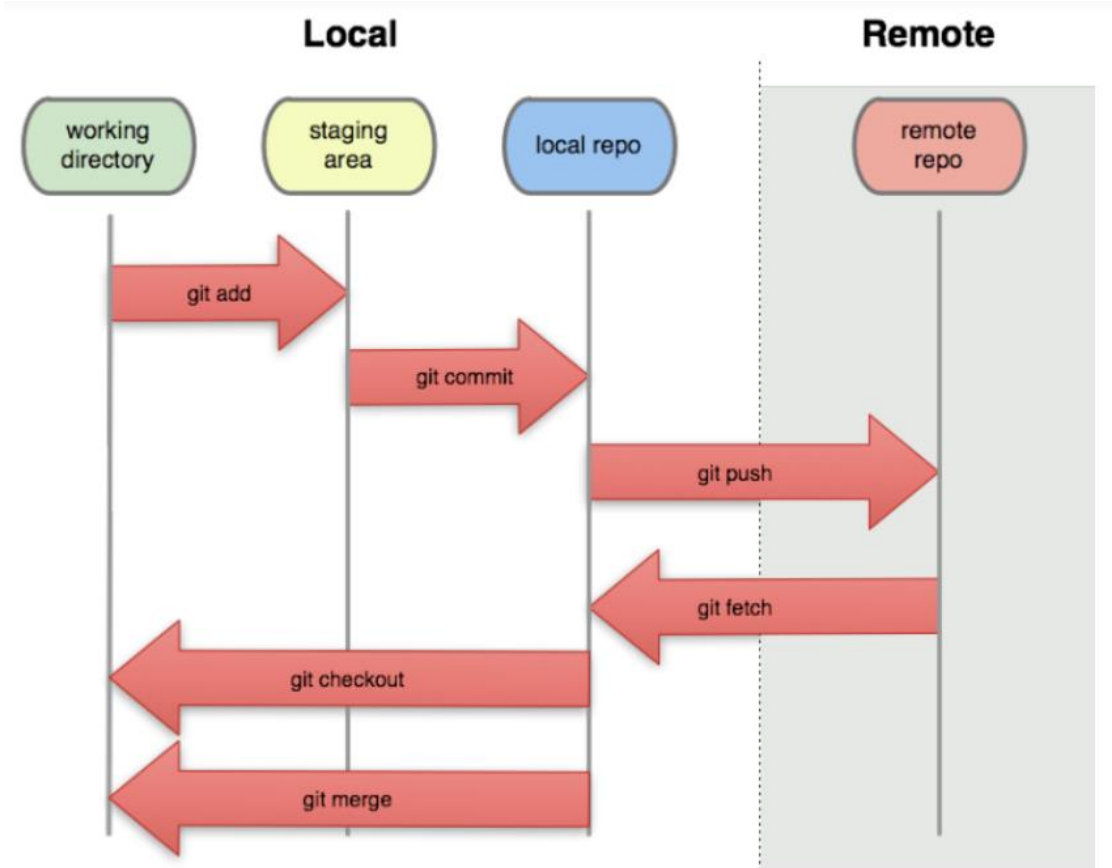
- Ein Repository oder Git-Projekt umfasst die gesamte Sammlung von Dateien und Ordnern, die einem Projekt zugeordnet sind.
- Verteilte Repository statt zentrale Repository. D.h. dass jeder Entwickler ein lokales Repository hat und damit arbeiten kann.

- Um Daten mit anderen Entwicklern auszutauschen, muss ein entferntes Repository erstellt werden.
- Entfernte Repository wird üblicherweise von einem Git-Server verwaltet wie z.B. github.

Git: Repository

Lokales und remotes Arbeiten mit Git

- Der grundlegende Git-Arbeitsablauf:
  - Datei ändern.
  - Zum Index hinzufügen.
  - Commiten.
- Git-Arbeitsablauf eines remoten Arbeiten
  - Klonen (ein lokales Repository durch git clone erzeugen der aktuellen Version eines entfernten Repostories entspricht).
  - Mit dem Repository lokal arbeiten.
  - Änderungen puschen.

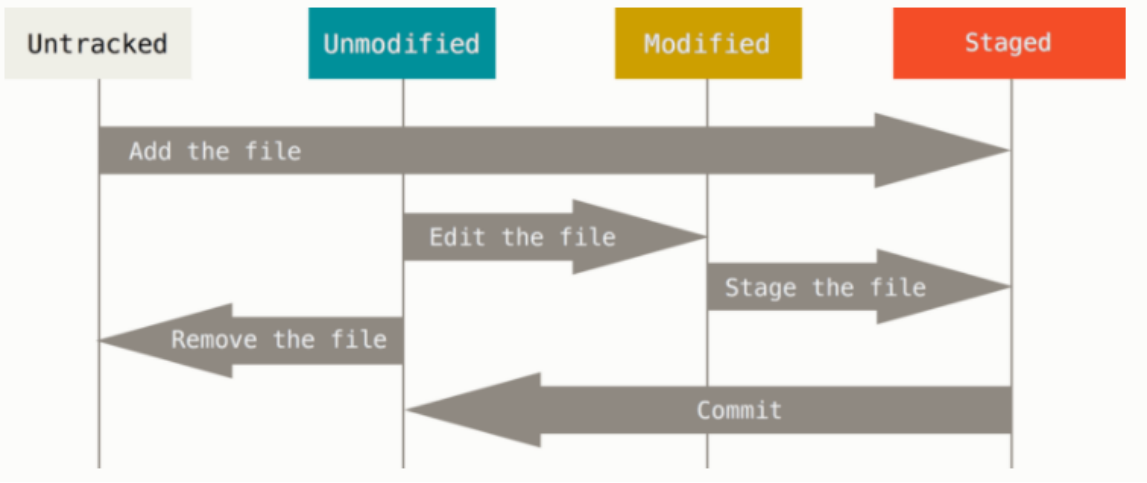


http://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/git.html

Git: Zustände und Breiche

Dateistatus(Lebenszyklus)

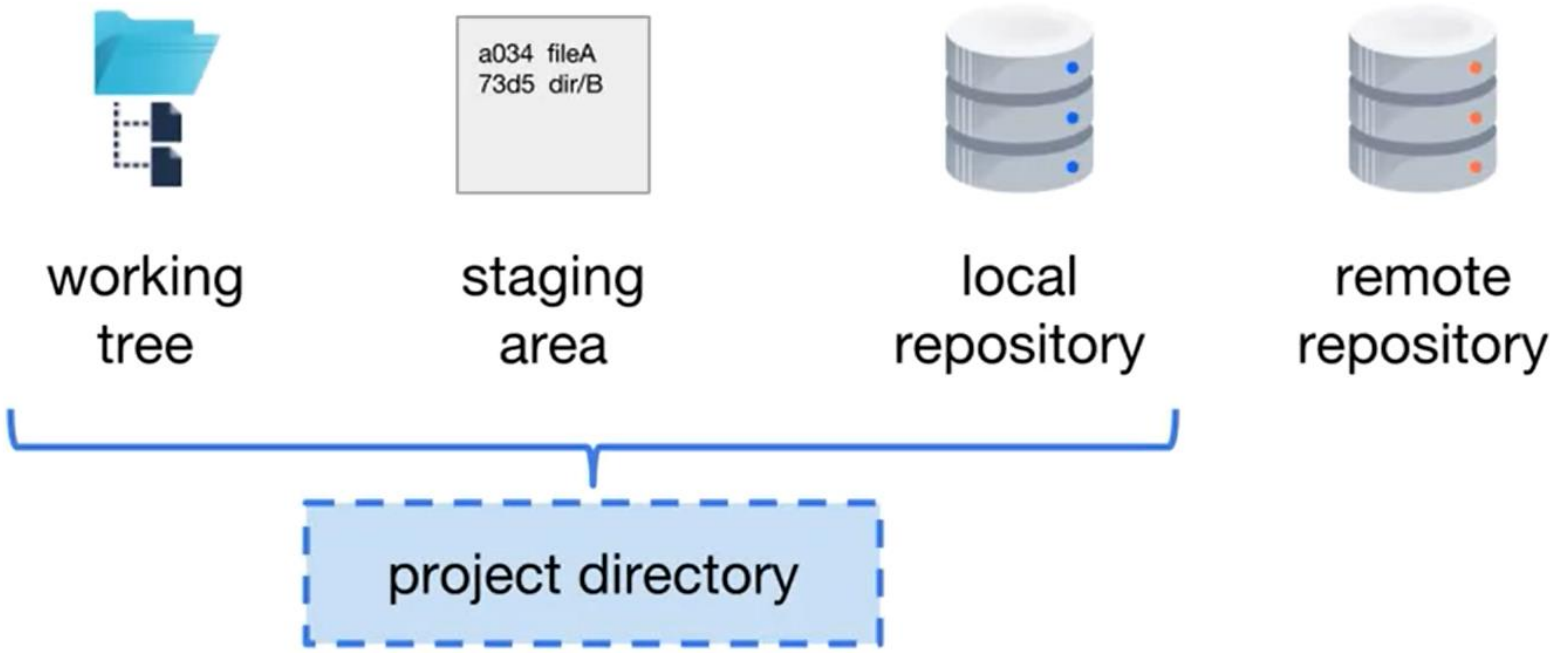
- Git definiert drei Hauptzustände, in denen sich eine Datei befinden kann:
- Unverändert (unmodified)
- Verändert (modified)
- Für den nächsten Commit vorgemerkt (staged)



https://www.git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Git%3F

Git: Zustände und Breiche

Hauptbereichen eines Git-Projekts

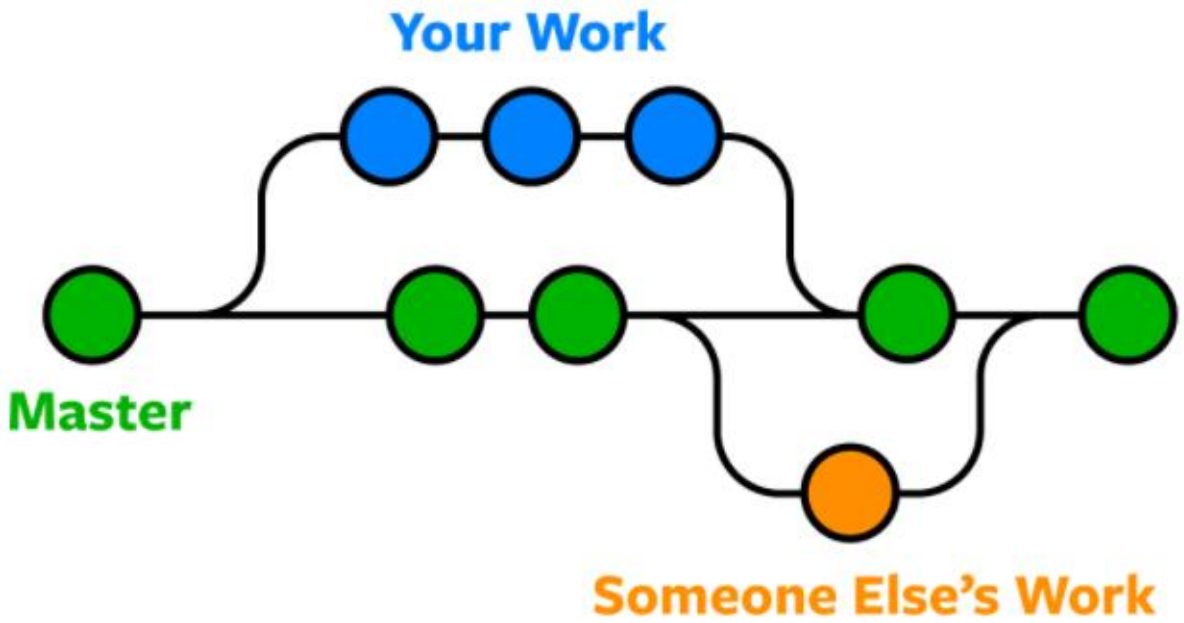


https://www.coursera.org/lecture/version-control-with-git/git-locations-SMJIy

Git: Branching and Merging

Git: Branches

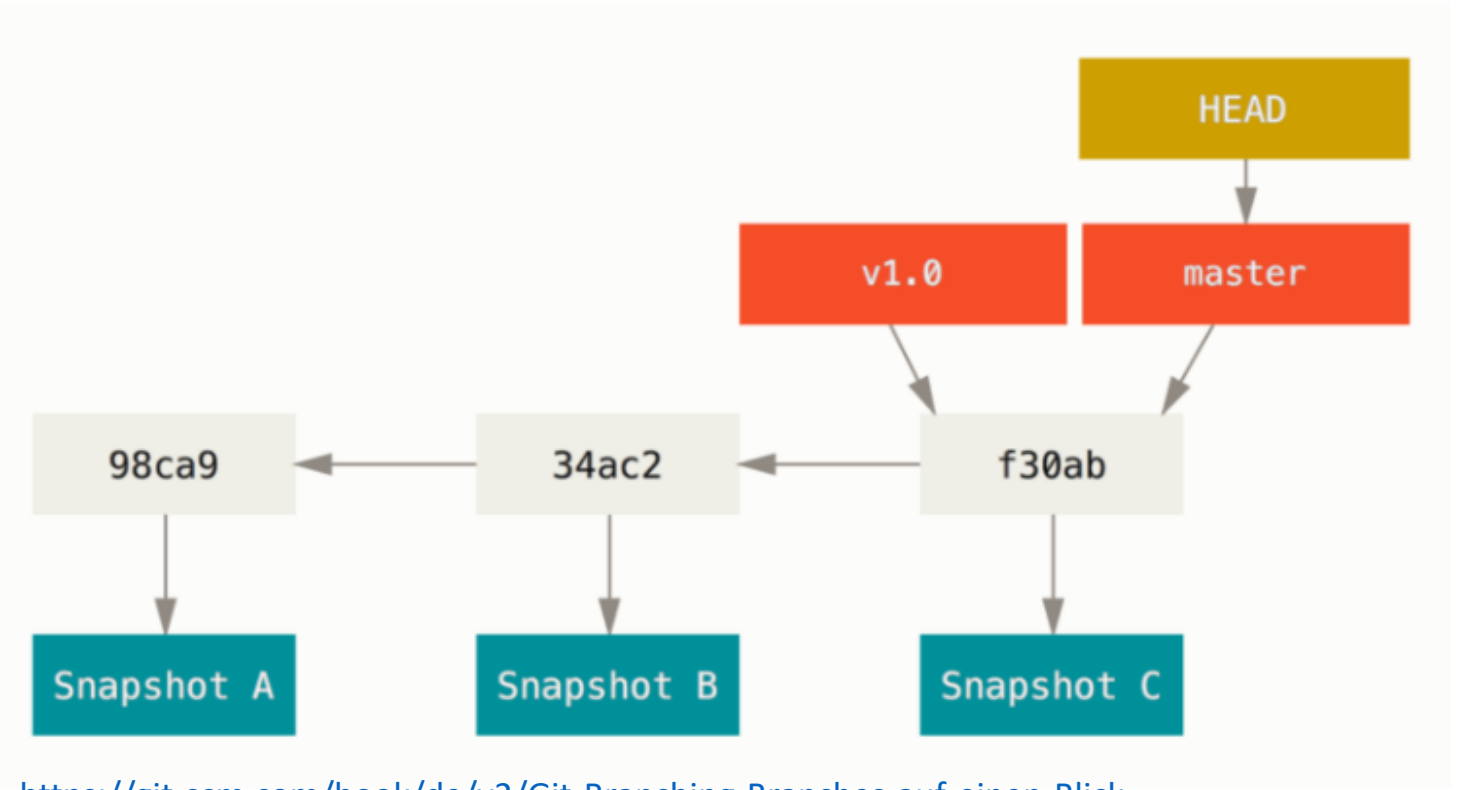
- Abzweigung in der Entwicklung.
- Ein Branch ist eine Reihe von Commits, die mit dem neuesten Commit im Branch beginnen und bis zum ersten Commit des Projekts zurückverfolgen.



https://www.nobledesktop.com/learn/git/git-branches

Git: Branches

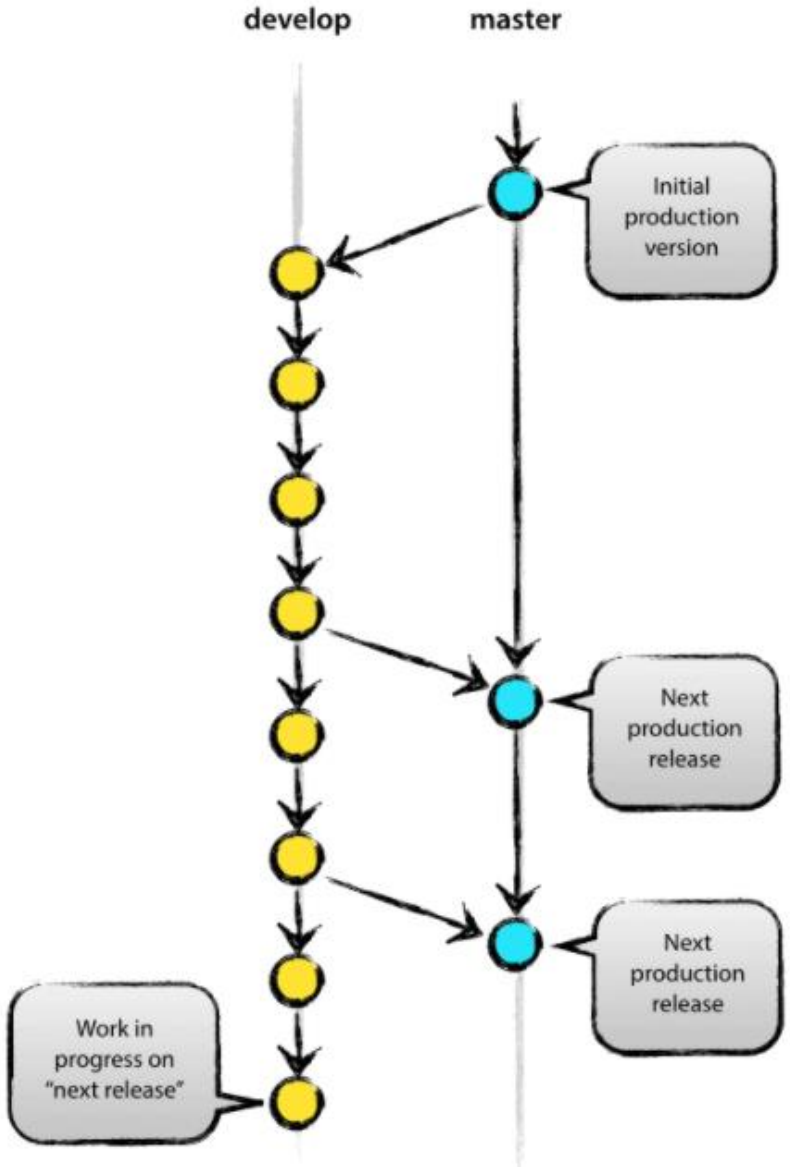
- Der „master“-Branch.
- HEAD: Eine symbolische Referenz auf der neuesten Commit im aktuellen Branch.



https://git-scm.com/book/de/v2/Git-Branching-Branches-auf-einen-Blick

Git: Branches

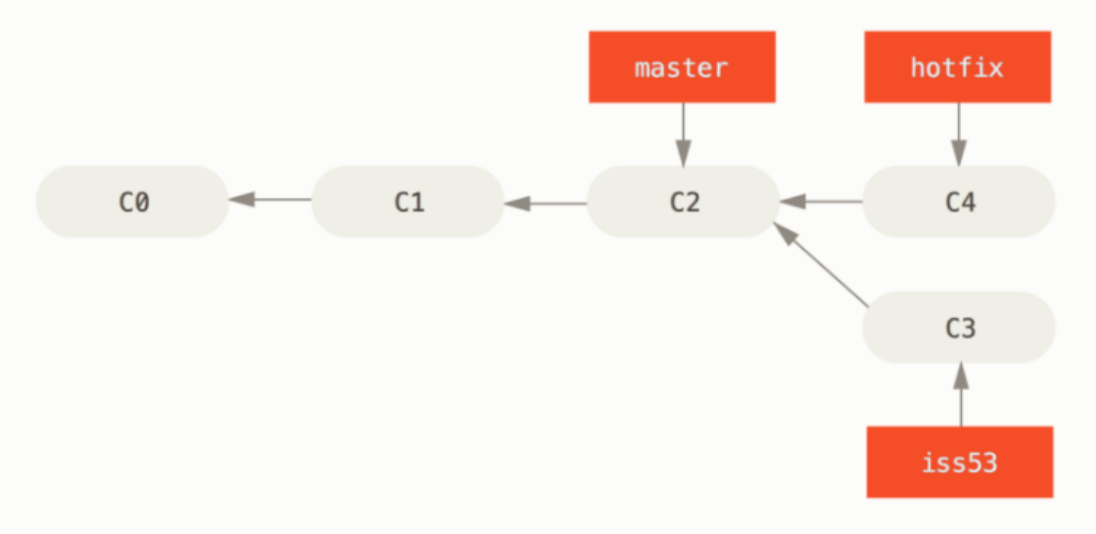
- Beliebig viele.
- Um beispielsweise neue Features zu entwickeln, Releases vorzubereiten oder um alte Versionen mit Bugfixes zu versorgen.

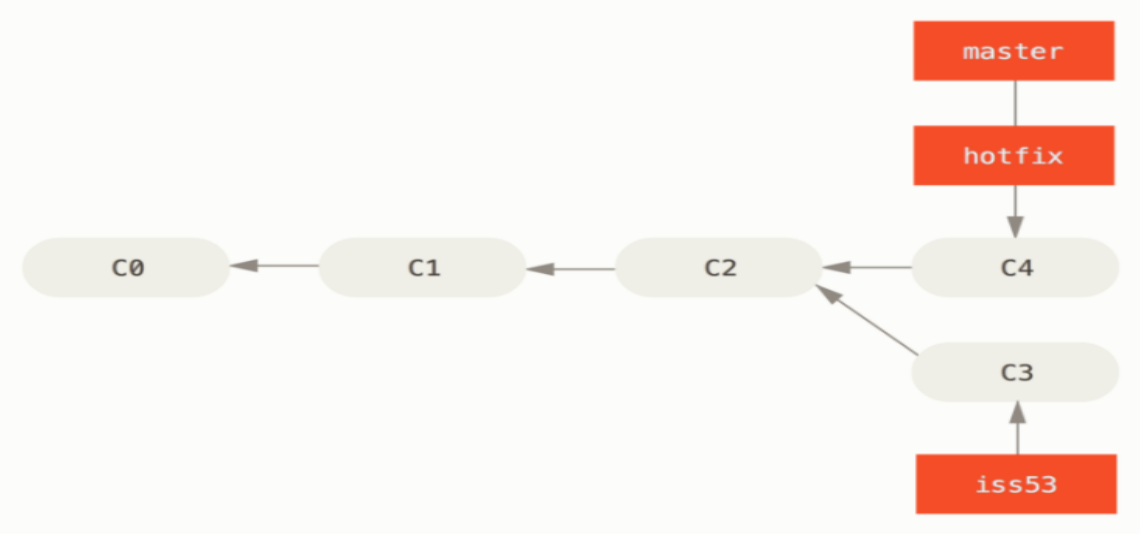


https://nvie.com/posts/a-successful-git-branching-model/

Git: Merge

- Fast-Forward-Merge: Wenn der einzufügende Branch ein direkter Nachfolger des anderen Branch ist.

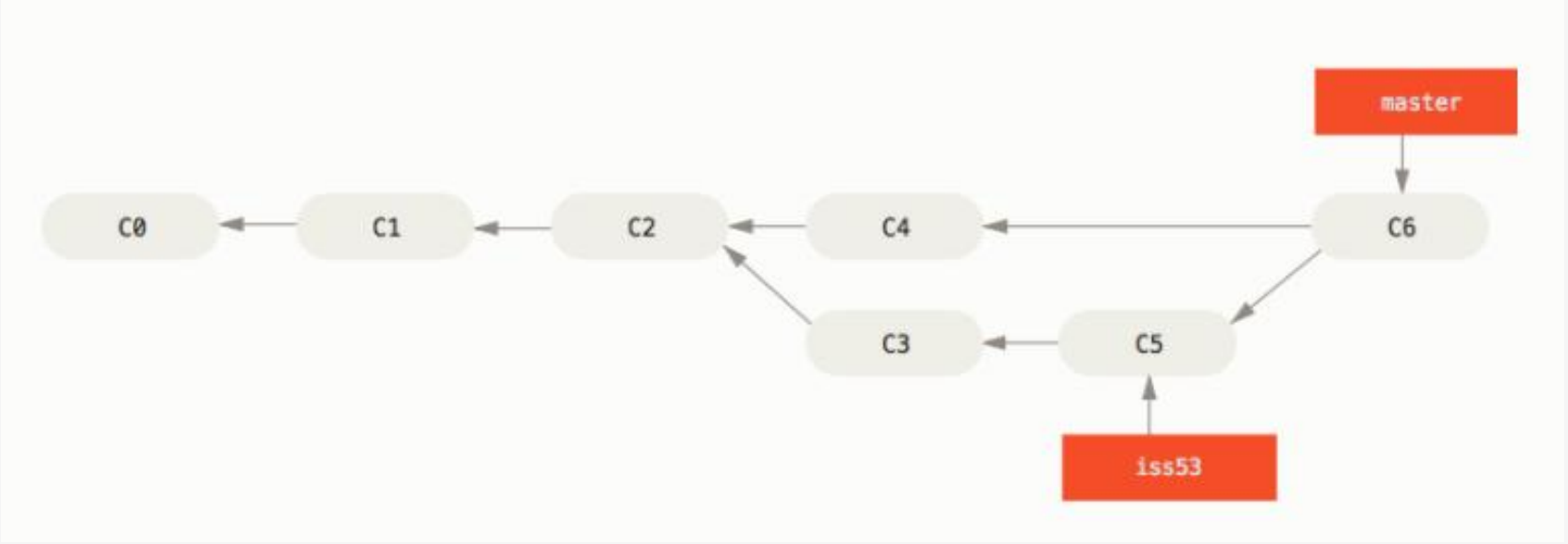




https://git-scm.com/book/de/v2/Git-Branching-Einfaches-Branching-und-Merging

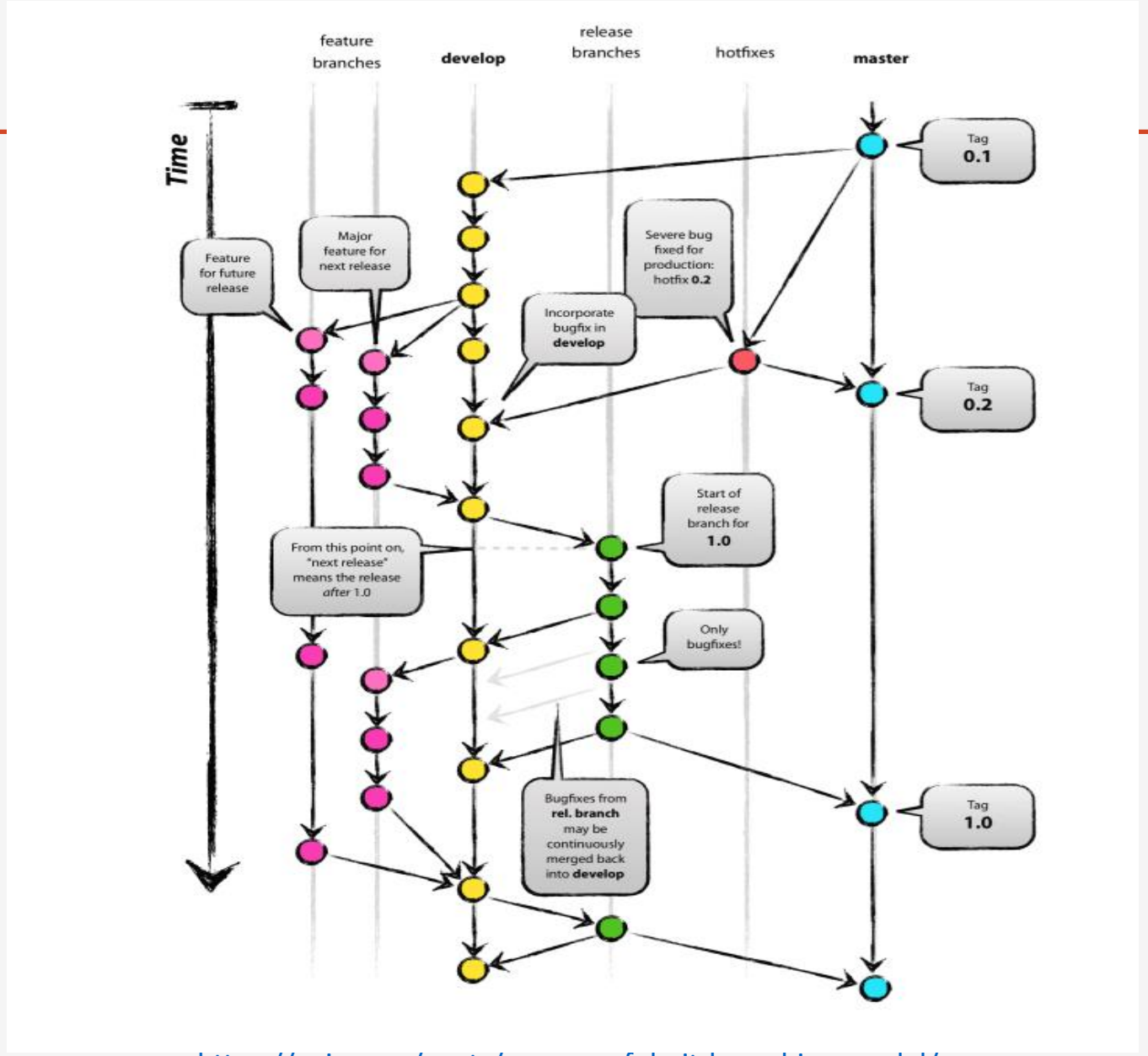
Git: Merge

- Merge-Commit: Wenn eine Branch kein direkter Vorgänger des anderen ist.



https://git-scm.com/book/de/v2/Git-Branching-Einfaches-Branching-und-Merging

Git-flow-Workflow

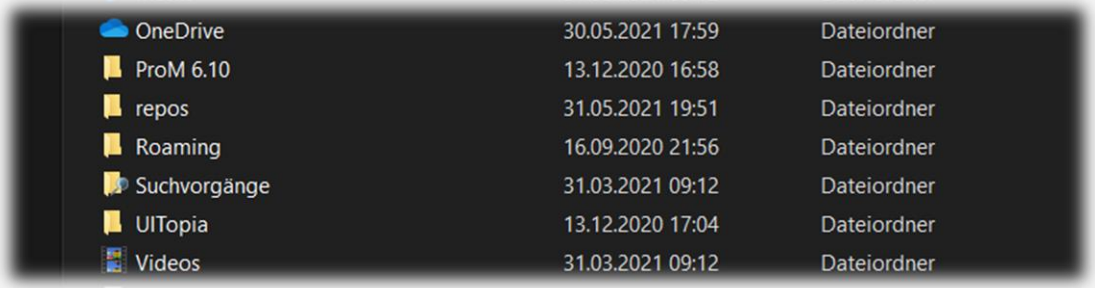


https://nvie.com/posts/a-successful-git-branching-model/

Git mit Kommandozeile(Getting Started)

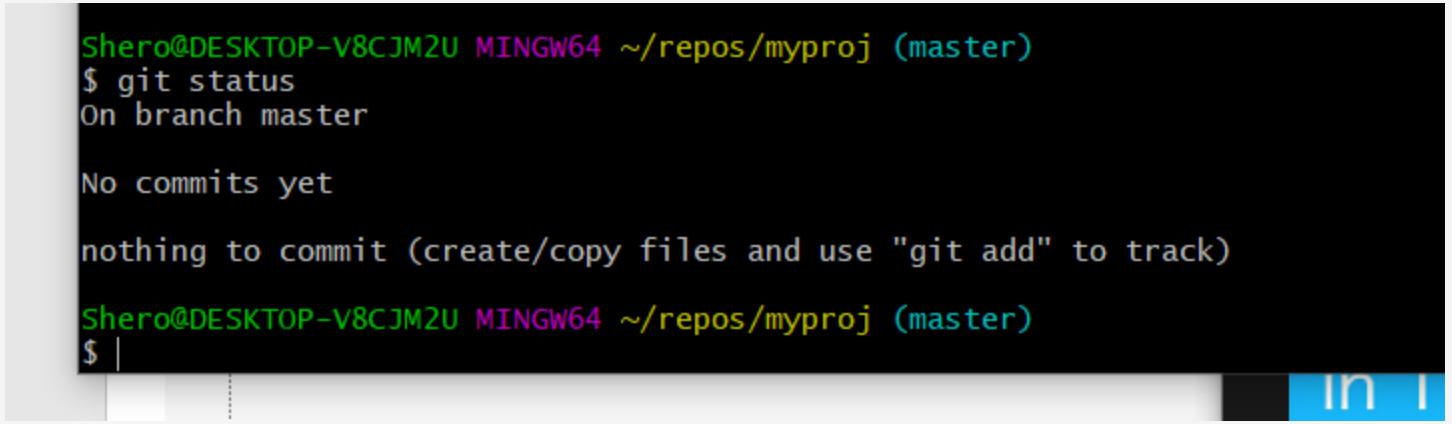
- Lokal Repository erstellen.

```
MINGW64:/c/Users/Shero/repos/myproj

Shero@DESKTOP-V8CJM2U MINGW64 ~
$ mkdir repos

Shero@DESKTOP-V8CJM2U MINGW64 ~
$ cd repos

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos
$ mkdir myproj

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos
$ cd myproj

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj
$ git init
Initialized empty Git repository in C:/Users/Shero/repos/myproj/.git/

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$
```



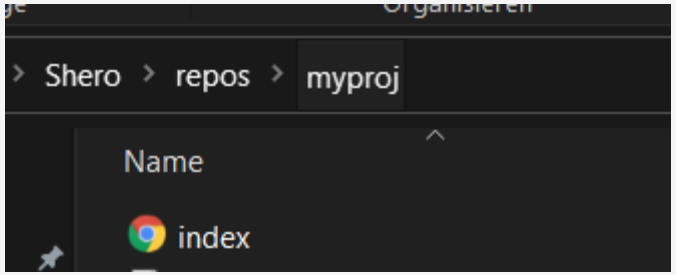
Git mit Kommandozeile(Getting Started)

- Status überprüfen.



Git mit Kommandozeile(Getting Started)

- Html Datei erstellen, in staging area hinzufügen und Status nochmal überprüfen.




```
Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ touch index.html

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git add index.html

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html


Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$
```

Git mit Kommandozeile(Getting Started)

- Html Datei index commiten und commits history zeigen.

```
Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git commit -m 'index file added'
[master (root-commit) 07c7e43] index file added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git log
commit 07c7e431bd0250aa9dfad774fd8bed9992ef5468 (HEAD -> master)
Author: 8rmostaf <rana.mostafa@studium.uni-hamburg.de>
Date:   Mon May 31 21:57:51 2021 +0200

    index file added

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$
```

Git mit Kommandozeile(Getting Started)

- Neue Branch anlegen und Branch wechseln.

```
Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git branch featureX

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git branch
  featureX
* master

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git checkout featureX
Switched to branch 'featureX'

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git branch
* featureX
  master

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$
```

Git mit Kommandozeile(Getting Started)

- Eine Textdatei erstellen, Änderung durchführen und commiten.

```
Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git status
On branch featureX
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git diff
diff --git a/test.txt b/test.txt
index e69de29..53f5007 100644
--- a/test.txt
+++ b/test.txt
@@ -0,0 +1 @@
+Hallo
\ No newline at end of file
```


```
Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git add test.txt

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git commit -m'Hallo added'
[featureX 79782c3] Hallo added
 1 file changed, 1 insertion(+)

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git log
commit 79782c37f0fe9c836574a1dca0f24fbaf4ca973d (HEAD -> featureX)
Author: 8rmostaf <rana.mostafa@studium.uni-hamburg.de>
Date:   Mon May 31 22:40:55 2021 +0200

    Hallo added
```

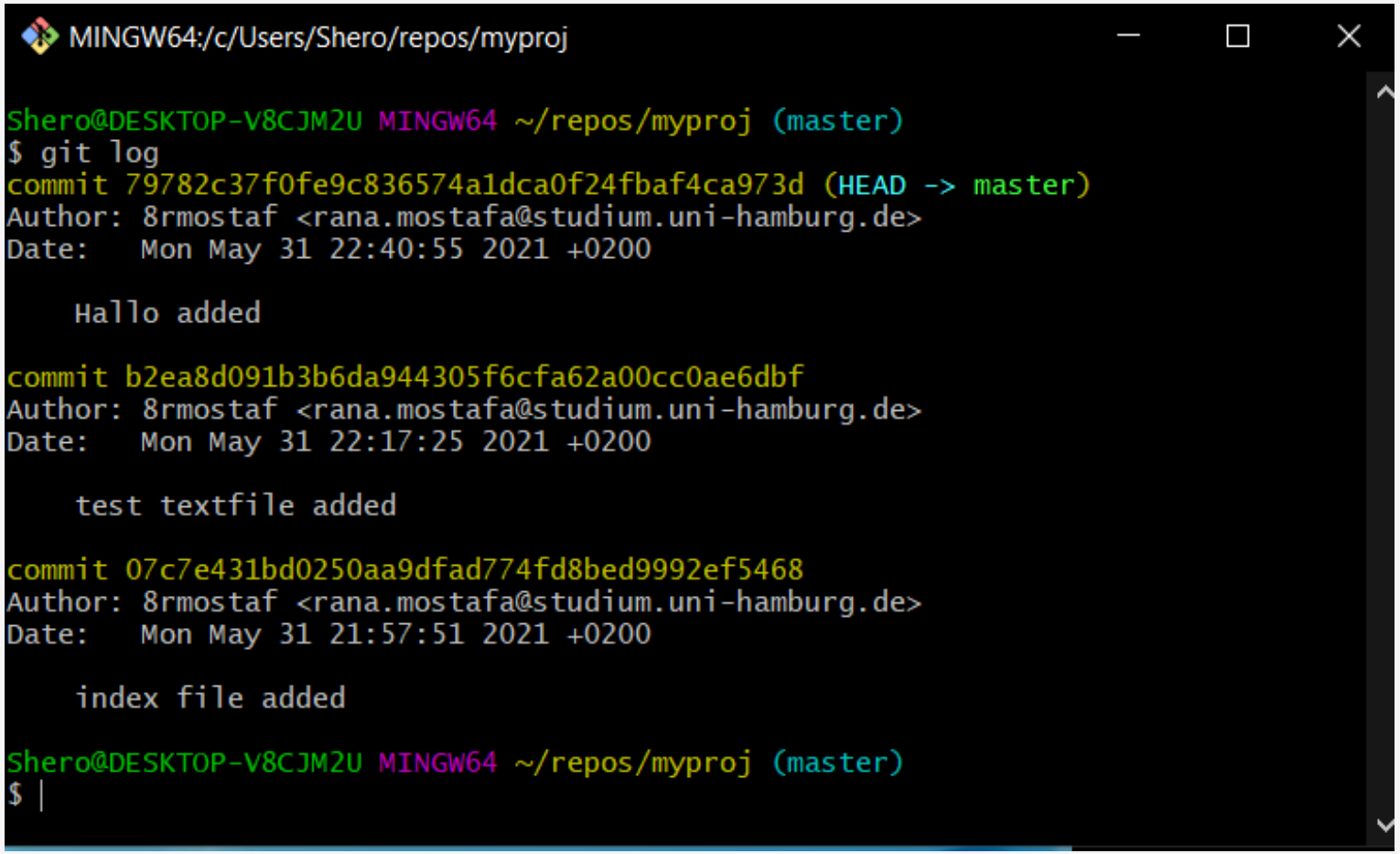
Git mit Kommandozeile

- Branch mergen und löschen.

```
Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (featureX)
$ git checkout master
Switched to branch 'master'

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git merge featureX
Updating 07c7e43..79782c3
Fast-forward
 test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git status
On branch master
nothing to commit, working tree clean

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$ git branch -d featureX
Deleted branch featureX (was 79782c3).

Shero@DESKTOP-V8CJM2U MINGW64 ~/repos/myproj (master)
$
```

Git mit Kommandozeile(Getting Started)

- Liste alle Commits (Commit-Historie).



Fazit

- Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien.
- Benutzer können mit lokalen Kopien des Repositorys arbeiten.
- Ein Repository enthält den Projektverlauf als Commits.
- Ein Commit ist eine snapshot des gesamten Projekts zu einem bestimmten Zeitpunkt.
- Ein Branch ist eine Reihe von Commits, die bis zum ersten Commit des Projekts zurückverfolgen.
- Das Merging kombiniert die Arbeit mehrerer Branche.

Quellen

- https://www.git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Git%3F
- https://nvie.com/posts/a-successful-git-branching-model/
- https://guides.github.com/introduction/git-handbook/
- http://gitbu.ch/
- Ich empfehle:
  - https://ohmygit.org/
  - https://rogerdudler.github.io/git-guide/

Danke für Ihre Aufmerksamkeit