

# Digital Skills für Ingenieur\*innen

12. Vorlesungstag

*„Versionskontrollsystem Git“*

# Versionskontrollsystem – Git

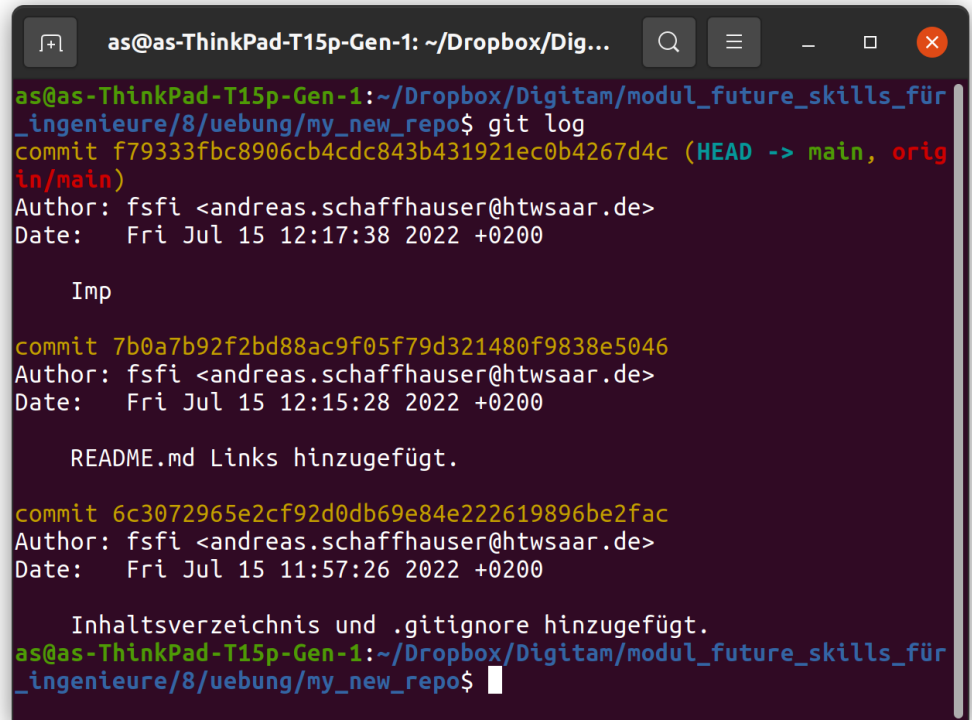
## Git Log

- Nachdem Sie einige Commits angelegt oder ein bestehendes Repository geklont haben, möchten Sie vielleicht wissen, **welche Änderungen zuletzt vorgenommen wurden**. Der grundlegende und mächtige Befehl, mit dem Sie das tun können, ist `git log`.
- Der Befehl `git log` **listet die Historie der Commits eines Projekts in umgekehrter chronologischer Reihenfolge auf**, wenn man ihn ohne weitere Argumente ausführt, d.h. die letzten Commits stehen oben.

# Versionskontrollsystem – Git

## Git Log

- Jeder Commit wird mit seiner **SHA-256 Checksumme**, **Namen** und **E-Mail Adresse des/der Autor\*in**, dem **Datum** und der **Commit Meldung** aufgelistet.



```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Dig...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$ git log
commit f79333fbc8906cb4cdc843b431921ec0b4267d4c (HEAD -> main, origin/main)
Author: fsfi <andreas.schaffhauser@htwsaar.de>
Date:   Fri Jul 15 12:17:38 2022 +0200

    Imp

commit 7b0a7b92f2bd88ac9f05f79d321480f9838e5046
Author: fsfi <andreas.schaffhauser@htwsaar.de>
Date:   Fri Jul 15 12:15:28 2022 +0200

    README.md Links hinzugefügt.

commit 6c3072965e2cf92d0db69e84e222619896be2fac
Author: fsfi <andreas.schaffhauser@htwsaar.de>
Date:   Fri Jul 15 11:57:26 2022 +0200

    Inhaltsverzeichnis und .gitignore hinzugefügt.
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$
```

Abbildung 1: git log

# Versionskontrollsystem – Git

## Git Status

- Der Befehl `git status` gibt den **Status des Arbeitsverzeichnisses** und der **Staging-Umgebung** zurück. So können Sie sehen, **welche Änderungen** sich in der **Staging-Umgebung** befinden, **welche nicht** und **welche Dateien nicht** von Git verfolgt werden.



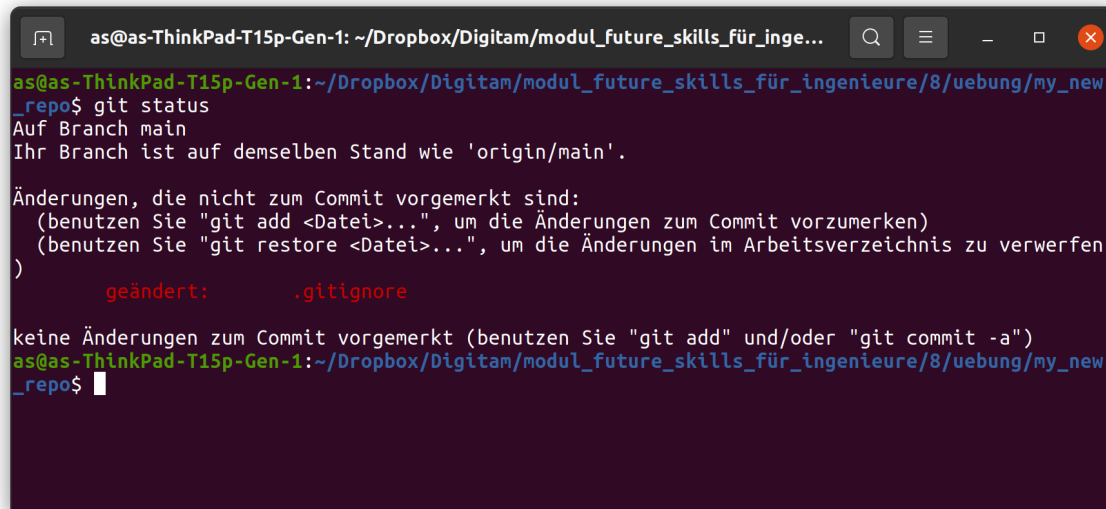
```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Dig...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$ git status
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

nichts zu committen, Arbeitsverzeichnis unverändert
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$
```

Abbildung 2: git status

## Git Status

- Wird eine Datei im Arbeitsverzeichnis verändert, ist dies auch im `git status` sichtbar. Die geänderte Datei wird **namentlich aufgeführt**.



```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul_future_skills_für_inge...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$ git status
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
)
   geändert:      .gitignore

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$
```

Abbildung 3: git status modified  
Andreas Schaffhauser, MSc.

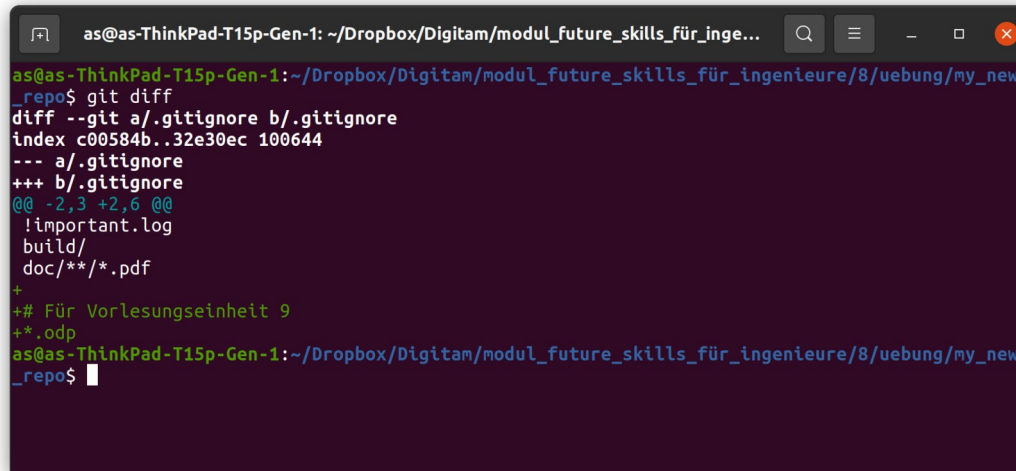
## Git Diff

- `git diff` ist ein vielseitiger Git-Befehl, durch dessen Ausführung ein **Vergleich von Git-Datenquellen durchgeführt wird**. Diese **Datenquellen können Commits, Branches, Dateien und vieles mehr sein**.
- Bei einem Aufruf von `git diff` **ohne Dateipfad** werden **Änderungen im gesamten Repository verglichen**.

# Versionskontrollsystem – Git

## Git Diff

- In unserem Beispiel ist nun sichtbar, dass sich die Datei `.gitignore` verändert hat. Es wurde eine Leerzeile, ein Kommentar und ein neues Pattern eingefügt, mit dem `*.odp` Dokumente zukünftig im Root des Repositorys ignoriert werden.



```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul_future_skills_für_inge...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$ git diff
diff --git a/.gitignore b/.gitignore
index c00584b..32e30ec 100644
--- a/.gitignore
+++ b/.gitignore
@@ -2,3 +2,6 @@
!important.log
build/
doc/**/*.pdf
+
+# Für Vorlesungseinheit 9
+*.odp
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$
```

Abbildung 4: git\_diff.png  
Andreas Schaffhauser, MSc.

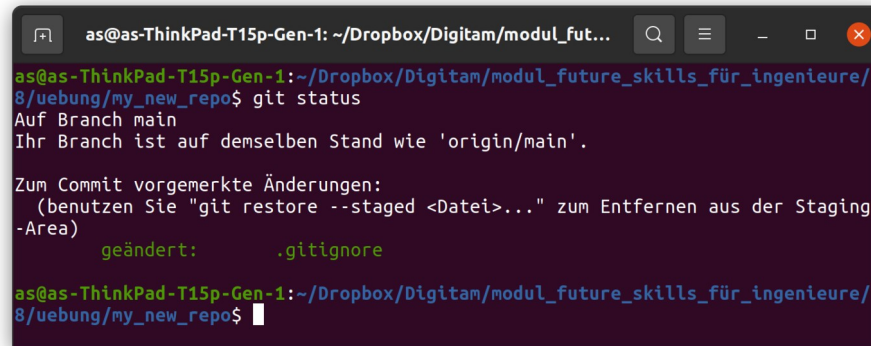
## Git Restore

- Stellt **spezifizierte Pfade im Arbeitsbereich mit Inhalt aus einer Wiederherstellungsquelle wieder her**. Wenn ein Pfad überwacht wird, aber in der Wiederherstellungsquelle nicht vorhanden ist, wird er entfernt, damit er mit der Quelle übereinstimmt.
- `git restore` kann somit genutzt werden, um **Dateien aus der Staging Area wieder so herzustellen, wie sie in HEAD zu finden sind**.
- Somit kann die mittels `git add` für den nächsten Commit vorgemerkte Datei ersetzt werden. Resultat ist, dass **die Datei aus der Staging Area verschwunden ist und wieder als unstaged markiert ist**.



## Git Restore

- Folgendes Szenario: die Datei `.gitignore` wurde verändert und sie wurde der Staging Area für den nächsten Commit hinzugefügt. Sie möchten die Datei allerdings nochmal aus der Staging Area herausnehmen, weitere Änderungen machen und sie dann der Staging Area nochmals hinzufügen.

A terminal window with a dark background and light text. The window title is "as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul\_fut...". The terminal shows the command "git status" being executed. The output indicates the user is on the "main" branch and that the branch is up-to-date with "origin/main". It lists a change to ".gitignore" as staged for the next commit. A note suggests using "git restore --staged <Datei>..." to unstage the file. The prompt is ready for the next command.

```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul_fut...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/
8/uebung/my_new_repo$ git status
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.


Zum Commit vorgemerkte Änderungen:
  (benutzen Sie "git restore --staged <Datei>..." zum Entfernen aus der Staging
  -Area)
   geändert:      .gitignore

as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/
8/uebung/my_new_repo$
```

Abbildung 5: git restore

## Git Restore

- Folgendes Szenario: dafür benutzen Sie den Befehl `git restore --staged .gitignore`. Nach dem erneuten Absetzen des Befehls `git status` wird die Datei wieder als geändert angezeigt.



```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul_fut...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/
8/uebung/my_new_repo$ git restore --staged .gitignore
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/
8/uebung/my_new_repo$ git status
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

Änderungen, die nicht zum Commit vorgemerkt sind:
(benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
(benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
    geändert:    .gitignore

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/
8/uebung/my_new_repo$
```

Abbildung 6: git restore  
Andreas Schaffhauser, MSc.

## Git Reset

- Der Befehl `git reset [<mode>] [<commit>]` ist ein **komplexes und vielseitiges Werkzeug zum Rückgängigmachen von Änderungen**. Diesen Befehl können Sie auf **dreierlei Arten** aufrufen:
  - × `--soft`: **verändert weder die Indexdatei, noch das Arbeitsverzeichnis**. Setzt allerdings den HEAD auf `<commit>` zurück (wie alle Modi). Somit bleiben all ihre modifizierten Dateien unberührt.

Szenario: Sie möchten den letzten Commit rückgängig machen (weil er z.B. nicht alles notwendige enthielt oder die Commit Message enthielt einen Fehler)

```
git reset --soft HEAD~1
```

## Git Reset

- Der Befehl `git reset [<mode>] [<commit>]` ist ein **komplexes und vielseitiges Werkzeug zum Rückgängigmachen von Änderungen**. Diesen Befehl können Sie auf **dreierlei Arten** aufrufen:
  - × `--mixed`: **Zurücksetzen vom Index, allerdings nicht vom Arbeitsverzeichnis**, d.h. die geänderten Dateien bleiben erhalten, werden aber nicht für den nächsten Commit markiert.

Szenario: Sie haben eine Datei gestaged, die Sie allerdings wieder aus der Staging Area entfernen möchten.

```
git reset -- <file_path>
```

## Git Reset

- Der Befehl `git reset [<mode>] [<commit>]` ist ein **komplexes und vielseitiges Werkzeug zum Rückgängigmachen von Änderungen**. Diesen Befehl können Sie auf **dreierlei Arten** aufrufen:
  - × `--hard`: **Setzt den Index und das Arbeitsverzeichnis zurück**. Alle Änderungen an nachverfolgten Dateien in der Arbeitsstruktur seit `<commit>` werden verworfen.

Szenario: Sie möchten alle Dateien auf einen bestimmten Commit zurücksetzen

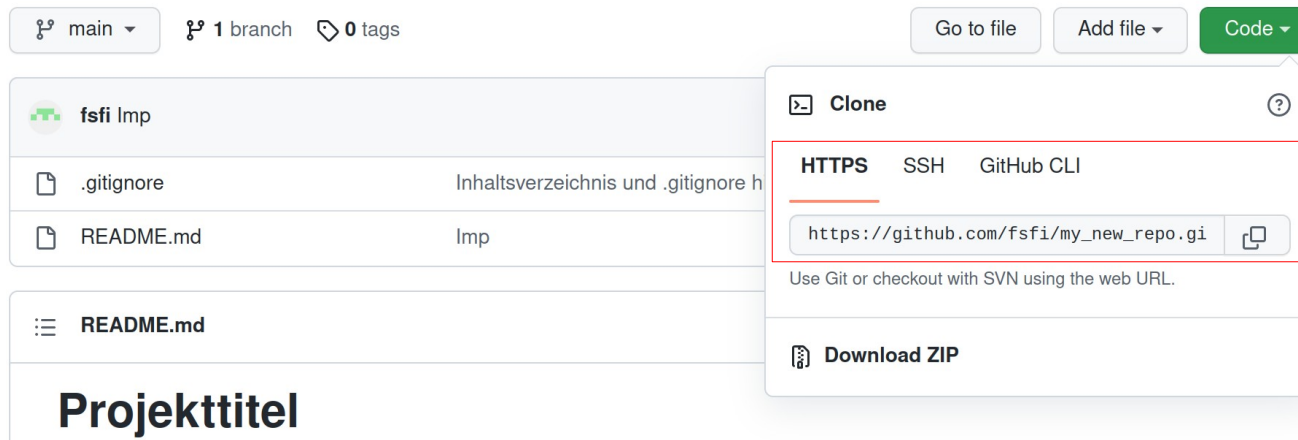
```
git reset --hard <commit>
```

## Git Clone

- Der Befehl `git clone` dient dazu, ein vorhandenes Repository als Ziel festzulegen und einen Klon oder eine Kopie des Ziel-Repositorys zu erstellen.
- Szenario: Sie finden unter Github ein öffentlich zugängliches Repository, welches für Sie interessanten Quellcode beinhaltet. Sie möchten eine lokale Kopie dieses Repositorys auf ihren Rechner ziehen.

## Git Clone

- Der Befehl `git clone` dient dazu, ein vorhandenes Repository als Ziel festzulegen und einen Klon oder eine Kopie des Ziel-Repositorys zu erstellen.



git clone via  
https → Token  
notwendig!

Abbildung 7: git clone via https

## Git Clone

- Der Befehl `git clone` dient dazu, ein vorhandenes Repository als Ziel festzulegen und einen Klon oder eine Kopie des Ziel-Repositorys zu erstellen.

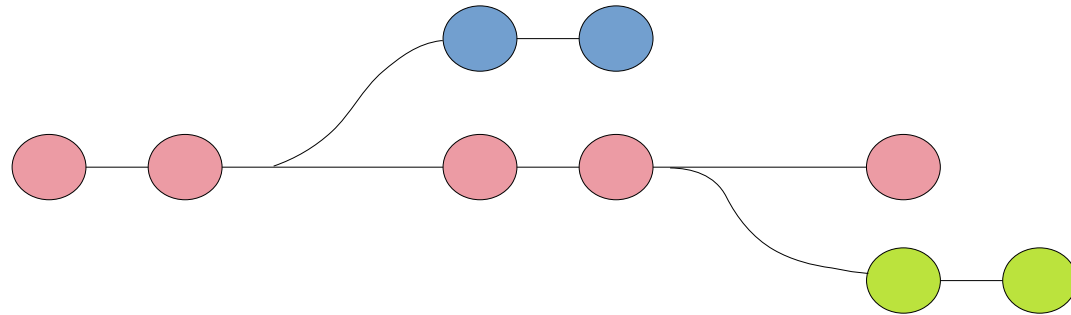


git clone via  
ssh → SSH  
Schlüssel  
notwendig!

Abbildung 8: git clone via ssh



*„Git Branches – Entwicklungszweige“*



## Branches (Entwicklungsstrang)

- Ein Branch ist ein **eigenständiger Entwicklungsstrang**, der von der Hauptcodebasis **verzweigt**. Er repräsentiert eine unabhängige Entwicklungslinie.
- Er dient als Abstrahierung des Prozesses "Bearbeitung/Staging/Commit".
- Man kann sich Branches als eine Möglichkeit vorstellen, **ein vollständig neues Arbeitsverzeichnis inklusive neuer Staging-Umgebung und neuem Projektverlauf einzurichten**. Neue Commits werden im Verlauf des aktuellen Branch aufgezeichnet.

# Versionskontrollsystem – Git

## Branches (Entwicklungsstrang)

- Folgendes Schaubild zeigt ein Repository mit drei isolierten Entwicklungslinien (bug fix, main & feature).

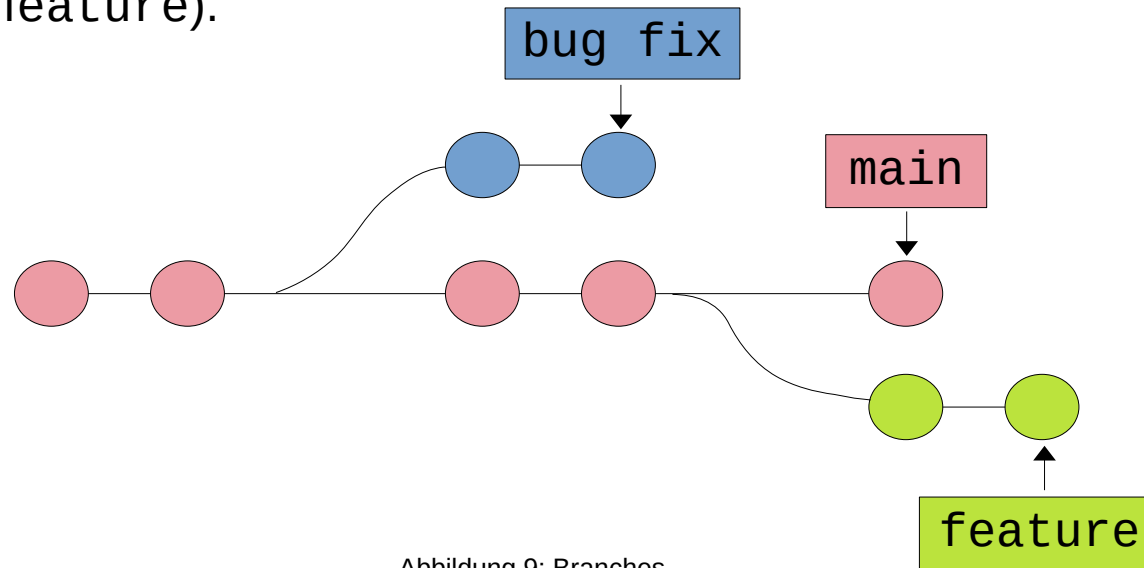


Abbildung 9: Branches

## Branches (Entwicklungsstrang)

- Welche Vorteile bringt die Möglichkeit von Branches mit sich?
  - × Die Auslagerung der Entwicklung in Branches macht es möglich, **an dem Branch bug fix und feature unabhängig vom main zu arbeiten.**
  - × Beide Entwicklungsstränge können voneinander **isoliert getestet** werden.
  - × Außerdem wird verhindert, dass **fragwürdiger Code in den main Branch überführt wird.**

## Branches (Entwicklungsstrang)

- Der Branch, **der standardmäßig bei der Initialisierung eines Repositorys erstellt wird**, ist der `master` Branch.
- Im Juni 2020 veröffentlichte die Software Freedom Conservancy eine Erklärung, in der sie zusammenfasst, warum der Standardbegriff `master` aufgrund seiner Geschichte für viele Benutzer anstößig ist.
- Git 2.28.0, veröffentlicht im Juli 2020, **führte die Konfigurationsoption `init.defaultBranch` ein**, die es `git` Benutzern ermöglicht, einen anderen Standard-Branch-Namen als `master` zu definieren und zu konfigurieren.

```
git config --global init.defaultBranch main
```

## Branch umbenennen

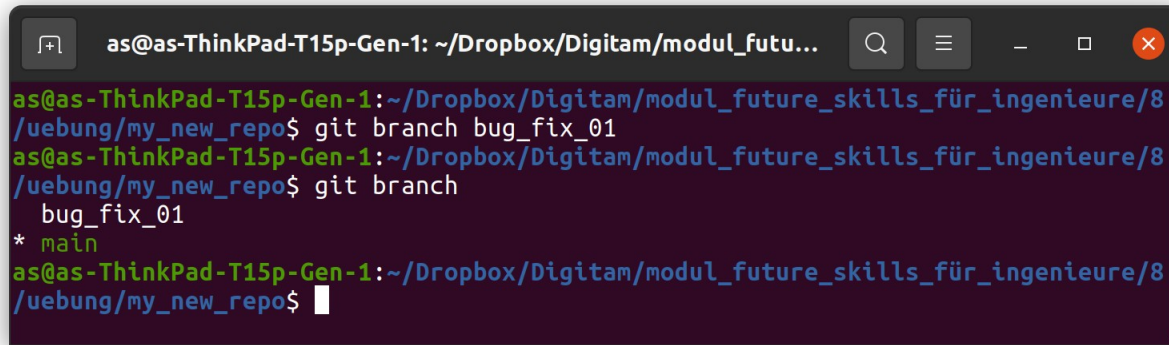
- Falls Sie den **Branch umbenennen wollen**, da Sie die globale Einstellung noch nicht gesetzt haben, **geht das mit den zwei Flags -m und -M**.
  - × `git branch -m <oldname> <newname>`
  - × `git branch -m <newname>` (umbenennen des aktuellen Branches)
  - × `git branch -M <newname>` (auf case-sensitiven Dateisystemen, z.B. Windows)

## Branches anzeigen

- Um auf die Existenz der verschiedenen Branches in einem Repository aufmerksam zu werden, können Sie sich auf **verschiedene Arten Branches anzeigen lassen**.
  - × Anzeigen **aller lokalen Branches**: `git branch`
  - × Anzeigen **aller remote Branches**: `git branch -r`
  - × Anzeigen **aller Branches (lokal und remote)**: `git branch -a`

## Neuen Branch anlegen

- Für das **Anlegen eines neuen Entwicklungszweiges**, nutzen Sie das Kommando `git branch <branch>`.
- Dadurch **erstellen Sie einen neuen Branch mit dem Namen <branch>**. Der Branch wird allerdings **nicht ausgecheckt**, d.h. **man arbeitet nicht auf dem neuangelegten Branch**.

A terminal window screenshot showing the execution of the 'git branch' command. The prompt is 'as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul\_futu...'. The user enters 'git branch bug\_fix\_01' and then 'git branch'. The output shows 'bug\_fix\_01' and '\* main'. The prompt then changes to 'as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul\_future\_skills\_für\_ingenieure/8/uebung/my\_new\_repo\$' and the user enters 'git branch' again, resulting in a blank line.

```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul_futu...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8
/uebung/my_new_repo$ git branch bug_fix_01
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8
/uebung/my_new_repo$ git branch
bug_fix_01
* main
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8
/uebung/my_new_repo$
```

Abbildung 10: git branch  
Andreas Schaffhauser, MSc.



## Branch löschen

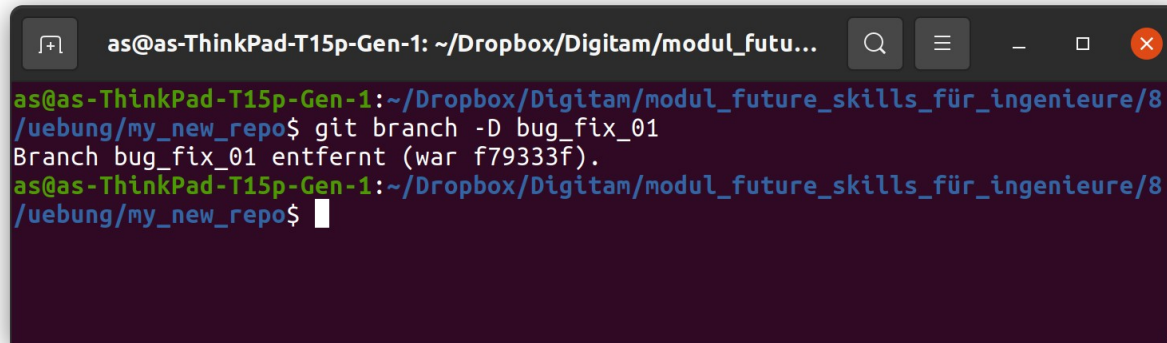
- Damit Sie den Entwicklungszweig <branch> auch **wieder löschen können** – sei es nach getaner Arbeit oder, wenn Sie diesen nur zu Testzwecken genutzt haben – nutzen Sie den -d Schalter.

```
git branch -d <branch>
```

- Hierbei handelt es sich um ein **„sicheres“ Löschen**. Falls nicht gemergte Änderungen existieren, wird folgendes angezeigt: „Der Branch <branch> ist nicht vollständig zusammengeführt. Wenn Sie sicher sind diesen Branch zu entfernen, führen Sie git branch -D aus.“

## Branch löschen

- Das -D Flag **erzwingt das Löschen** des angegebenen Branches, **selbst wenn es nicht gemergte Änderungen gibt**.
- Mit diesem Befehl werden alle Commits, die mit einem Entwicklungszweig in Verbindung stehen dauerhaft verworfen.

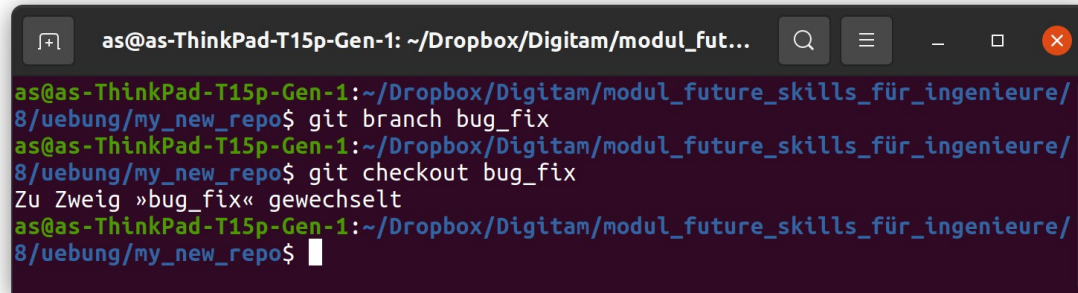


```
as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul_futu...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8
/uebung/my_new_repo$ git branch -D bug_fix_01
Branch bug_fix_01 entfernt (war f79333f).
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8
/uebung/my_new_repo$
```

Abbildung 11: branch delete  
Andreas Schaffhauser, MSc.

## Branch wechseln

- Damit ein Branch nach dem Anlegen auch **ausgecheckt und damit gearbeitet werden kann**, existiert das `git checkout <branch>` Kommando.
- Wenn Sie einen Branch auschecken, werden **die Dateien im Arbeitsverzeichnis mit den in dem betreffenden Branch gespeicherten Versionen aktualisiert und Git speichert alle neuen Commits in dem ausgecheckten Branch.**

A terminal window screenshot showing the execution of Git commands. The terminal title is "as@as-ThinkPad-T15p-Gen-1: ~/Dropbox/Digitam/modul\_fut...". The commands and their outputs are: 1. `git branch bug_fix` (no output shown). 2. `git checkout bug_fix` (output: "Zu Zweig »bug\_fix« gewechselt"). 3. The prompt returns to the shell.

```
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_fut...
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$ git branch bug_fix
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$ git checkout bug_fix
Zu Zweig »bug_fix« gewechselt
as@as-ThinkPad-T15p-Gen-1:~/Dropbox/Digitam/modul_future_skills_für_ingenieure/8/uebung/my_new_repo$
```

Abbildung 12: checkout  
Andreas Schaffhauser, MSc.

# Versionskontrollsystem – Git

## Entwicklungszeige kombinieren

- Ein **Merge** findet statt, wenn **zwei Entwicklungszeige kombiniert** werden.
- Git nimmt dazu **zwei (oder mehr) Commit-Pointer** und versucht, einen **gemeinsamen Basis-Commit** zu erstellen.

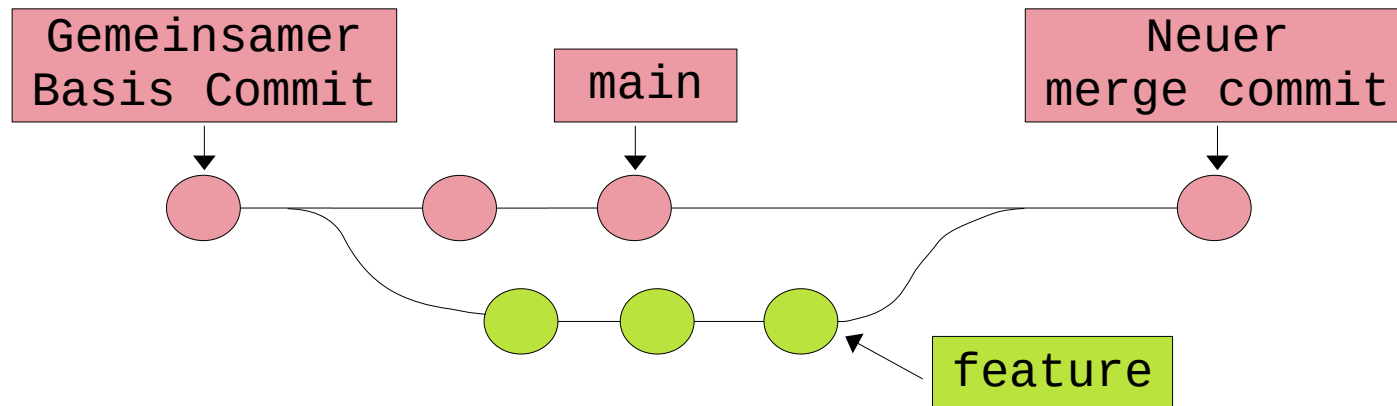


Abbildung 13: merge

## Entwicklungszeige kombinieren

- Damit Entwicklungszeige **kombiniert** werden können, verfügt git über das Kommando `git merge`.

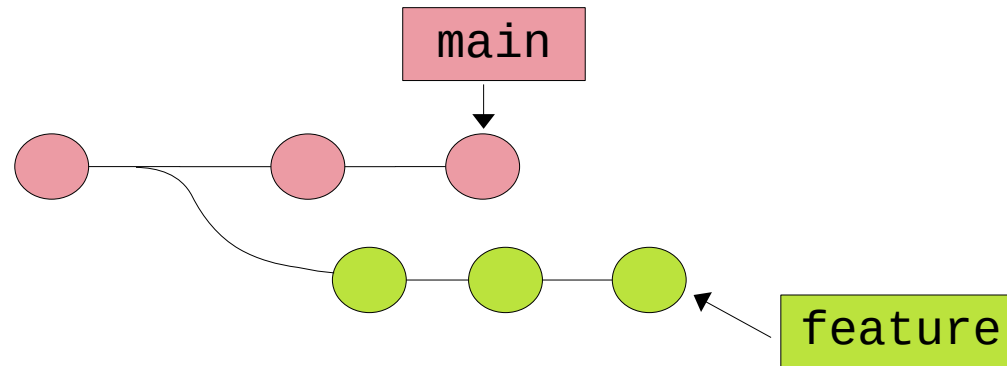


Abbildung 14: „geforkte“ Commit Historie  
Andreas Schaffhauser, MSc.

## Entwicklungszweige kombinieren (mergen)

- Mittels `git merge feature main` können Sie die Commits aus dem `main` Branch in den `feature` Branch übernehmen.

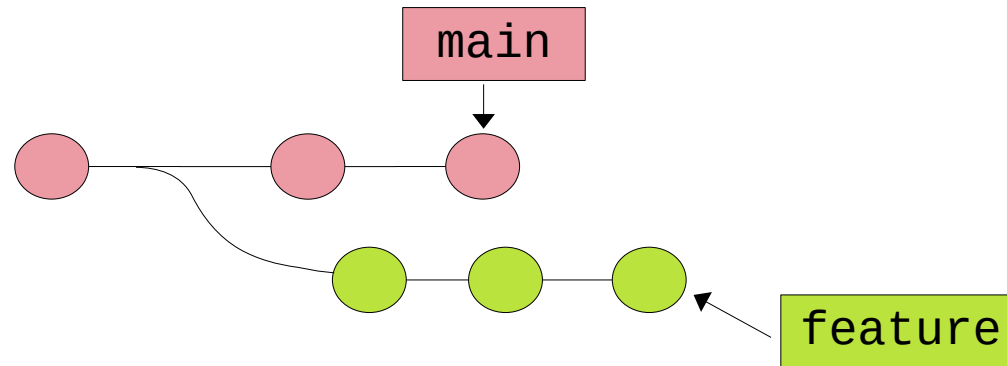


Abbildung 14: „geforkte“ Commit Historie  
Andreas Schaffhauser, MSc.

## Entwicklungszweige kombinieren (mergen)

- Dabei **entsteht** ein **neuer Merge-Commit im feature Branch**, der den Verlauf beider Branches vereint und eine Branch-Struktur erstellt, die wie folgt aussieht:

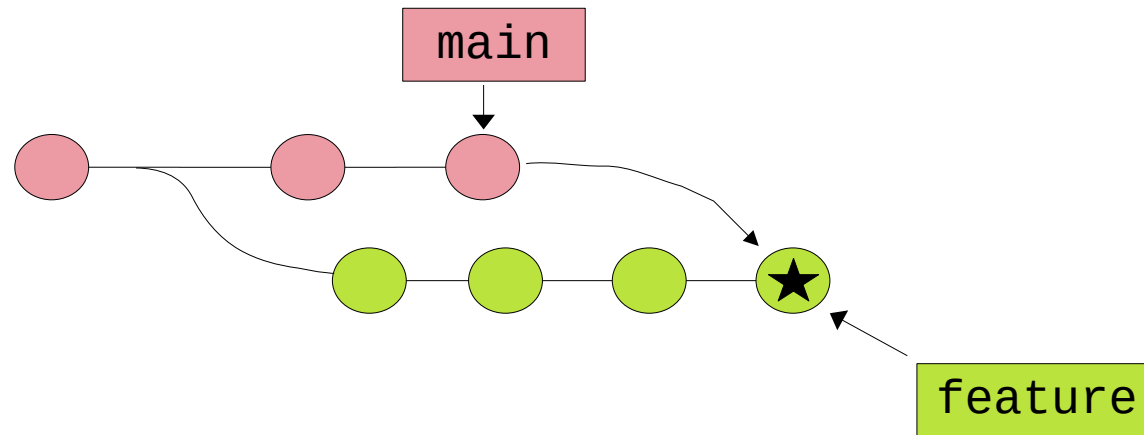


Abbildung 15: Änderungen von Main werden in Feature übernommen

## Entwicklungszweige kombinieren (mergen)

- Vorteil des Kommandos merge:
  - × Befehl ist **nicht destruktiv**. Die vorhanden Branches werden in keinster Weise geändert.
- Nachteil des Kommandos merge:
  - × **Jede notwendig integrierte Upstream Änderung erzeugt einen irrelevanten Merge Commit.**
    - **Verlauf** des feature Branches **wird** somit **schnell unübersichtlich**.