

01 Git und GitHub

Multimedia Engineering (PS siehe 36609b)

M.Sc. Nils Hellwig

Lehrstuhl für Medieninformatik

FAKULTÄT FÜR INFORMATIK UND DATA SCIENCE



Universität Regensburg

Was ist Git?

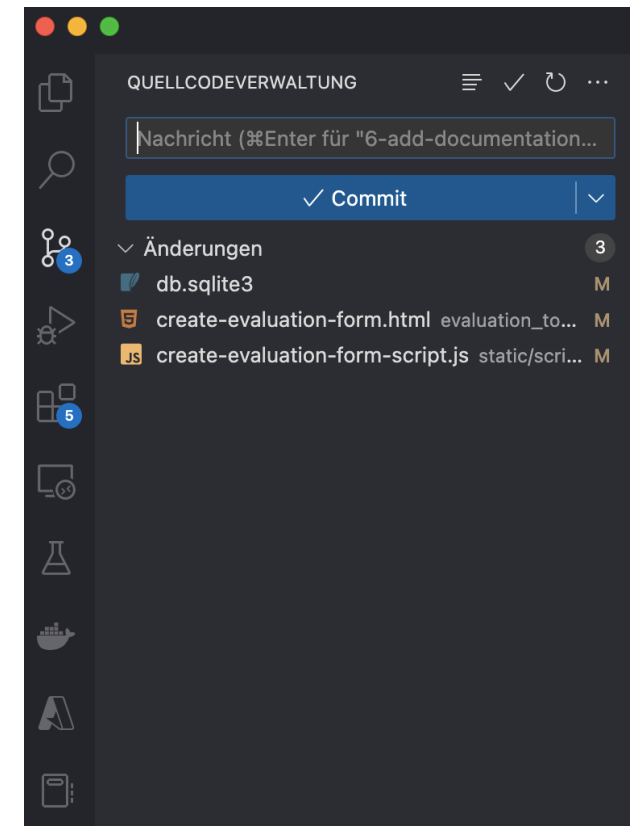
- Git ist ein Versionskontrollsystem, das es ermöglicht, Änderungen an Dateien und Codeprojekten zu verfolgen, zu verwalten und zu teilen.
- Git bietet eine Möglichkeit für mehrere Personen, an einem Projekt zusammenzuarbeiten und Änderungen auf einfache und effiziente Weise zu verwalten.
- Git ist plattformunabhängig und kann auf verschiedenen Betriebssystemen wie Windows, macOS und Linux verwendet werden.
- Git wurde im Jahr 2005 von Linus Torvalds, dem Schöpfer des Linux-Kernels entwickelt.

Was ist Git?

- Git kann sowohl über die Kommandozeile als auch über eine grafische Benutzeroberfläche (GUI) verwendet werden
- Einige der populären Git-GUI-Tools sind **GitHub Desktop**, Sourcetree, GitKraken, etc.
- Auch in Visual Studio Code gibt es eine integrierte Git-Verwaltung mit GUI

```
nils_hellwig@MacBook-Pro-von-Nils edu-quality-eval % git status
On branch 6-add-documentation-to-start-page
Your branch is up to date with 'origin/6-add-documentation-to-start-page'.

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:   db.sqlite3
        modified:   evaluation_tool/templates/evaluation_tool/components/create-evaluation-form.html
        modified:   static/scripts/create-evaluation-form-script.js
```



Git mit der Kommandozeile verwenden

- Um Git zu verwenden, öffnen wir zunächst die Kommandozeile
- Als Erstes können wir prüfen, ob Git richtig installiert wurde:

```
git --version  
>> git version 2.30.2.windows.1
```

Erstellen eines Ordners mit Git-Versionskontrolle

- Ist man mit der Kommandozeile in dem Ordner, in dem man Git initialisieren möchte, so kann man dies mit `git init`
- Damit wird ein Verzeichnis mit Git-Versionskontrolle initialisiert, sodass man die Versionsgeschichte des Codes verfolgen, verschiedene Versionen wiederherstellen oder Änderungen vergleichen kann
- Das Verzeichnis ist dadurch aber noch nicht für andere Programmierer zugänglich, sondern nur auf dem eigenen Gerät
- Um an dem Verzeichnis kollaborativ zu arbeiten, benötigt man ein **Remote-Repository**

Remote Repository am Beispiel GitHub

- Ein Remote-Repository kann bei Diensten wie GitHub, GitLab oder BitBucket gehostet werden
- Um ein Remote-Repository in GitHub zu erstellen, benötigt man einen GitHub Account.
- Nachdem das Remote-Repository erstellt wurde, kann man es mit `git clone <repo url>` auf den eigenen Computer laden bzw. "clonen"
- Wenn mehrere Entwickler an einem Projekt arbeiten, können alle eine Kopie des Remote-Repositories auf ihren eigenen Computer herunterladen und lokal Änderungen vornehmen

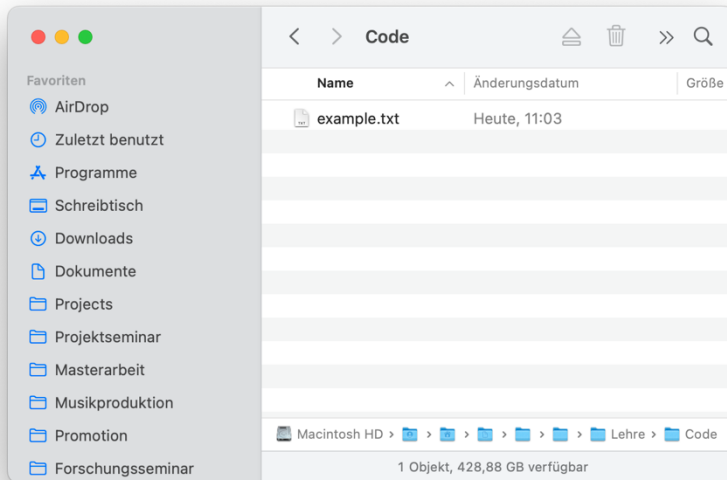
```
git add <filename>
```

```
git commit -m <commit message> + git push
```

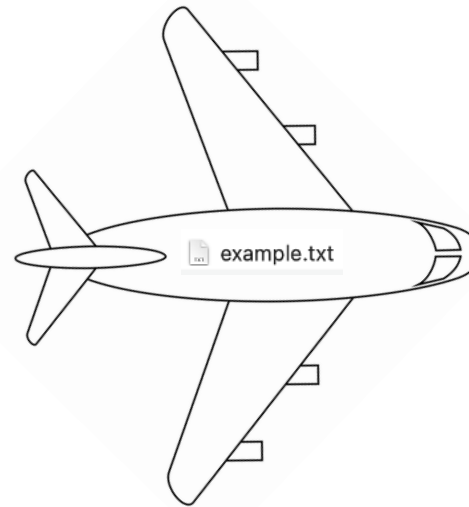
(1)

(2)

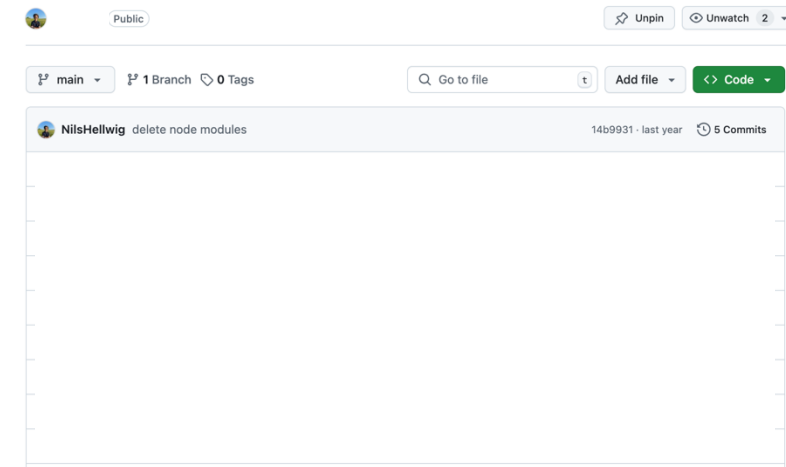
(3)



**Working
Directory**



**Staging
Environment**



**Remote
Repository**



Dateien zu einem Git-Repository hinzufügen

- Erstellen wir nun eine Datei (z.B. test.html) in unserem Arbeitsverzeichnis
- Mithilfe von `git status` können wir prüfen, ob die Datei im Staging Environment ist
- Wenn eine Datei im Arbeitsverzeichnis ist, aber nicht zum Staging-Environment hinzugefügt wurde, wird sie als *untracked* bezeichnet
- Wenn eine Datei zum Staging Environment hinzugefügt wurde, gilt diese als *tracked*
- Dateien die *untracked* sind und in das Repository aufgenommen werden sollen, müssen *gestaged* werden (in das Staging Environment aufgenommen werden)

Git Staging Environment

- Das Staging-Environment, auch Index oder Zwischenablage genannt, ist ein Bereich in Git, in dem Dateien vor dem Committing vorbereitet werden
- Immer, wenn man einen Meilenstein erreichen oder einen Teil der Arbeit abschließen, sollten die Dateien zum Staging Environment hinzugefügt werden
- Dateien können zum Staging Environment hinzugefügt werden mithilfe von
`git add <Dateiname/Verzeichnisname>`
- Es können auch mehrere Dateien auf einmal committet werden, beispielsweise mit `git add .` können alle Dateien hinzugefügt werden im Verzeichnis, in dem man sich gerade in der Kommandozeile befindet
- Wenn man mit den Änderungen zufrieden ist, kann man sie schließlich committen

Git Staging Environment

- Um eine Datei aus dem Staging Environment zu entfernen, kann "`git reset HEAD -- <Dateiname/Verzeichnisname>`" verwendet werden
- Führt man "`git reset HEAD -- .`", dann werden alle Dateien aus dem Staging Environment entfernt, die sich in dem Verzeichnis befinden, in dem man gerade in der Kommandozeile ist

Git Commit

- Dateien im Staging Environment können dann in unser Repository committet werden
- Zu jedem Commit sollte immer eine Nachricht hinzugefügt werden, der die darin enthaltene Änderung beschreibt
- Nachricht: Kurze und prägnante Beschreibung, wozu der Commit dient
- Durch das Hinzufügen einer Beschreibung zu jedem Commit ist es für Sie (und andere Entwickler) einfach zu sehen, was wann geändert wurde
- Aktiver statt passiver Sprachstil, z.B. „add device list“ oder „fix authentication bug“

Git Commit

- Ein Commit kann mit `git commit -m <Nachricht>` erstellt werden
- Um den Verlauf der Commits für ein Repository anzuzeigen, kann `git log` ausgeführt werden

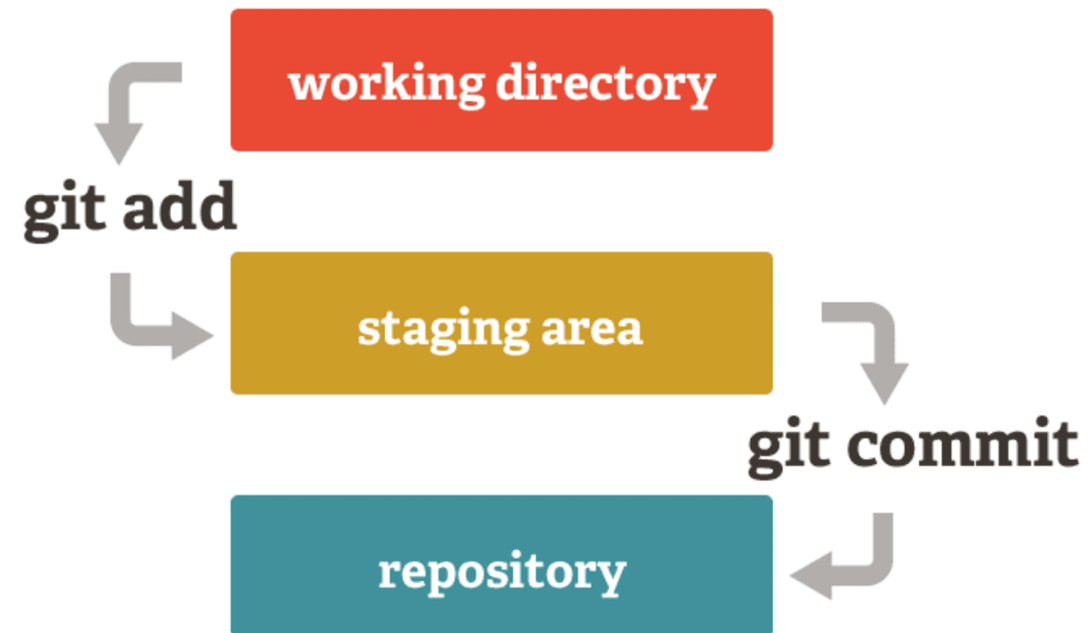
```
• nils_hellwig@MacBook-Pro-von-Nils git-test % git log
commit 2e6802bea6070e9fb8f065be78e9906486c94b6c (HEAD -> main)
Author: NilsHellwig
Date:   Sat Mar 4 11:28:29 2023 +0100

    add directory with test files

commit 5b61e4fa346eb4704fb89d9736cc6f8c27b960ea
Author: NilsHellwig
Date:   Sat Mar 4 11:28:08 2023 +0100

    add test.html
```

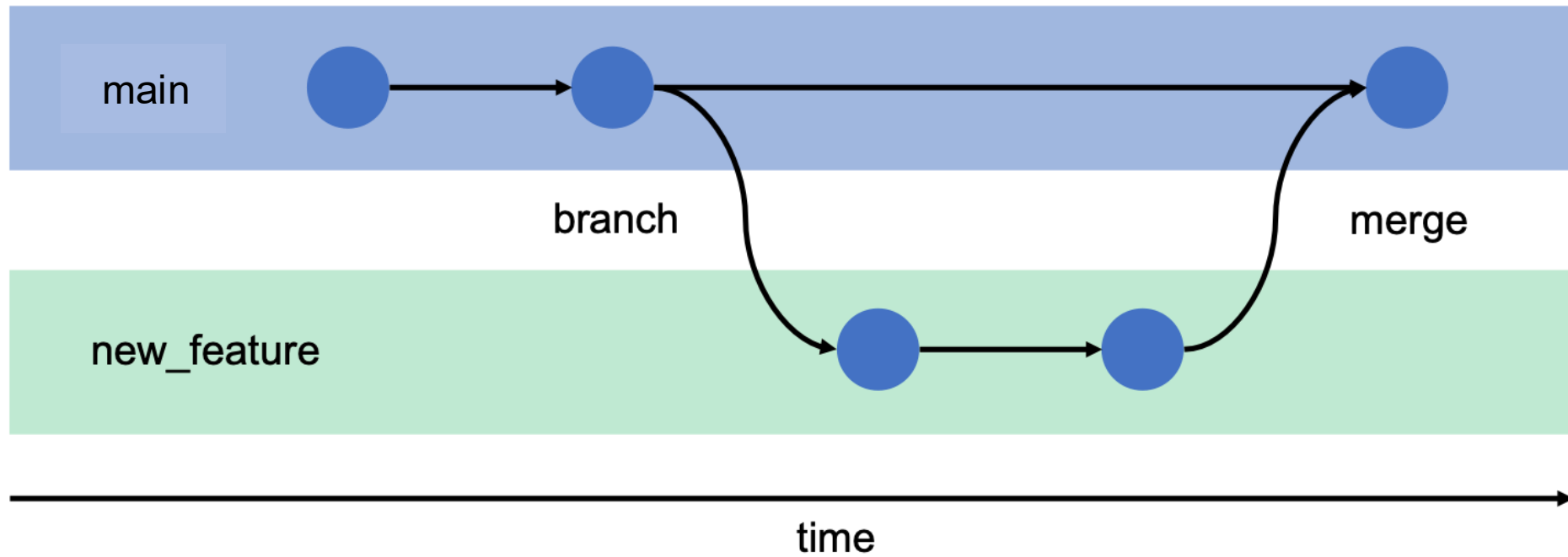
Git Summary



Git Branches

- In Git ist ein Branch eine neue/getrennte Version des Haupt-Repositorys (main-Branch)
- Jeder Branch kann unabhängig voneinander entwickelt und verändert werden, ohne dass dies den ursprünglichen Code beeinflusst
- Durch das Erstellen von Branches können verschiedene Versionen des Codes gleichzeitig existieren, was es Entwicklern ermöglicht, an unterschiedlichen Features oder Bugfixes zu arbeiten, ohne dass dies die Arbeit anderer Entwickler behindert
- Wenn ein Branch fertiggestellt ist, kann er in den main-Branch integriert werden
- Dadurch werden die Änderungen in den Hauptcode übernommen

Git Branches



Git Branches: Erstellung

- Einen neuen Branch erstellt man mit `git branch <Name des Branches>`
- Alle verfügbaren Branches sowie der Branch in dem man aktuell arbeitet können mit `git branch` angezeigt werden
- Zu einem anderen Branch kann gewechselt werden mithilfe von `git checkout <Name des Branches>`

Git Branch Merge

- Nachdem man mit der Implementierung in einem Branch fertig ist, kann dieser in den main-Branch überführt werden (oder ggf. in einen anderen Branch)
- Dafür muss man zunächst in den Branch wechseln (`git checkout`), in den der Branch an dem man gearbeitet hat überführt werden soll, also z.B. der main-Branch
- Das Merging kann dann mittels `git merge <Name des Branches>` ausgeführt werden
- Optional kann dann der überführte Branch gelöscht werden mithilfe von `git delete <Name des Branches>`

Git: Neue Änderungen seit dem letzten Commit verwerfen

- Wurde eine Änderung noch nicht committet und ist noch nicht im Staging Environment:
`git clean -f <Dateiname/Verzeichnisname>`
Achtung!: Änderungen, die man mit diesem Kommando verwirft, sind nicht gespeichert und können nachträglich nicht wiederhergestellt werden!
- Möchte man alle Änderungen verwerfen, kann auch `git clean -f` ausgeführt werden

Git Push

- `git push` ist ein Befehl, der verwendet wird, um Änderungen, die im lokalen Repository vorgenommen wurden, auf ein Remote-Repository hochzuladen

Git Pull

- Um Änderungen aus einem Remote-Repository herunterzuladen und in das lokale Repository zu integrieren, kann der Befehl `git pull` in Git verwendet werden
- Wenn eine Person Änderungen im Remote-Repository vornimmt, kann eine andere diese Änderungen herunterladen und mit `git pull` in ihr/sein lokales Repository integrieren, um sicherzustellen, dass sie/er immer mit den neuesten Änderungen arbeitet

.gitignore

- Die .gitignore ist eine Datei, die in einem Git-Repository angelegt werden kann
- Sie enthält eine Liste von Dateien oder Verzeichnissen, die von Git beim Tracking und Committing ausgeschlossen werden sollen
- Dadurch können beispielsweise sensible oder temporäre Dateien ausgeschlossen werden, um eine saubere Versionskontrolle zu gewährleisten und Konflikte zu vermeiden
- Häufig werden damit auch Paketinstallationen ignoriert, beispielsweise ein Virtual Environment (Python) oder node_modules (Node.js), also Dateien, die jeder Entwickler eigenständig installieren muss

Aufgabe bis zum nächsten Termin

- Erstellt ein Repository auf GitHub namens “todo-app-static”
- Committet und pusht in den main-Branch eine leere Datei namens *index.html*

Weiterführende Literatur / Quellen

- Offizielles Git Tutorial:

<https://git-scm.com/docs/gittutorial>

- Mehr zu Git Branching:

<https://git-scm.com/book/en/v2/Git-Branching-Branched-in-a-Nutshell>

<https://gitbookdown.dallasdatascience.com/branching-git-branch.html>