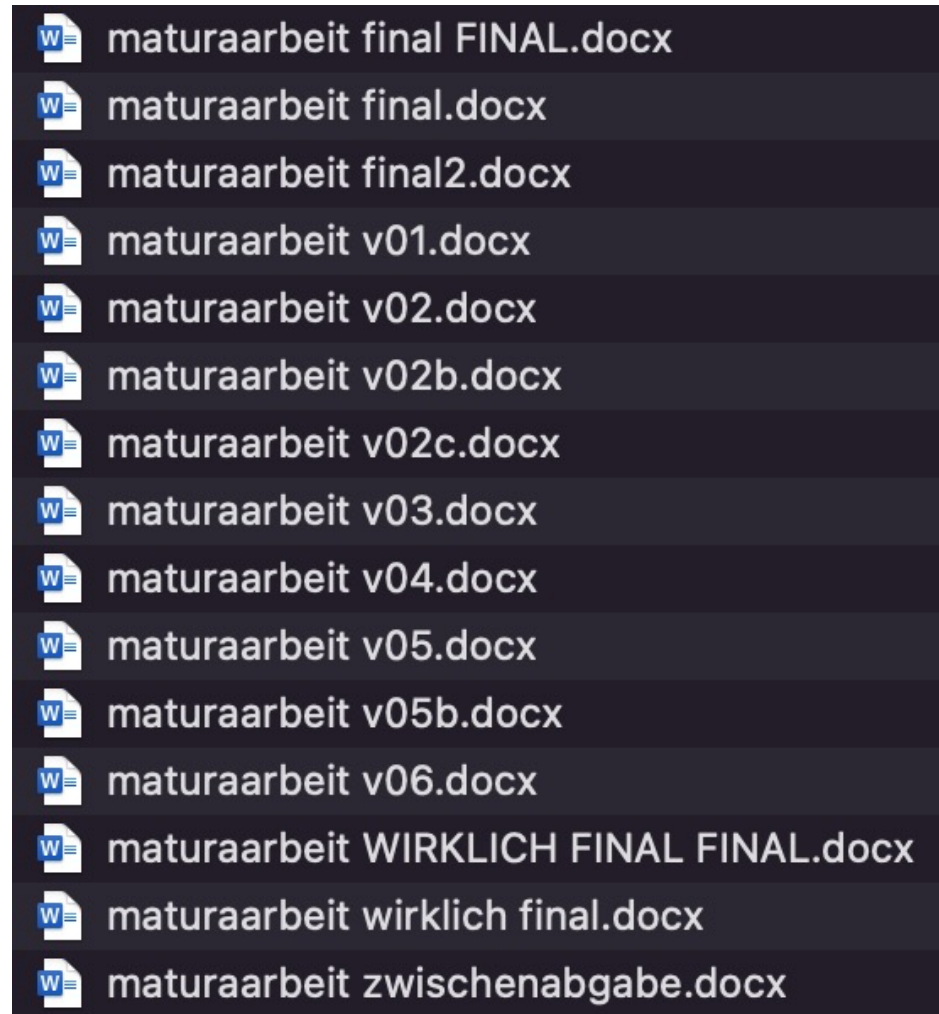


Git & GitHub

Problem 1: Versionen einer Arbeit



Problem 2: Laptop futsch, Arbeit futsch

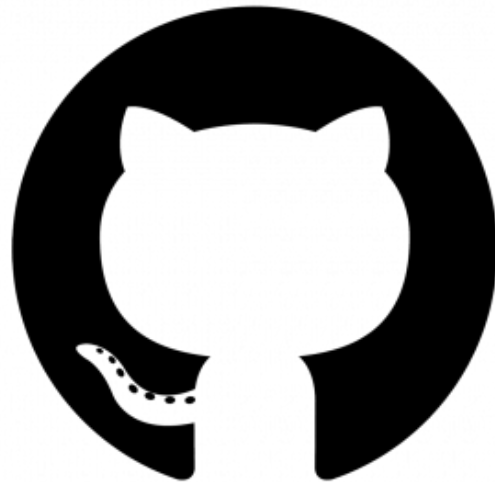


Lösung

- Git (-> Problem 1)
- GitHub (-> Problem 2)



git



GitHub

Git in a Nutshell

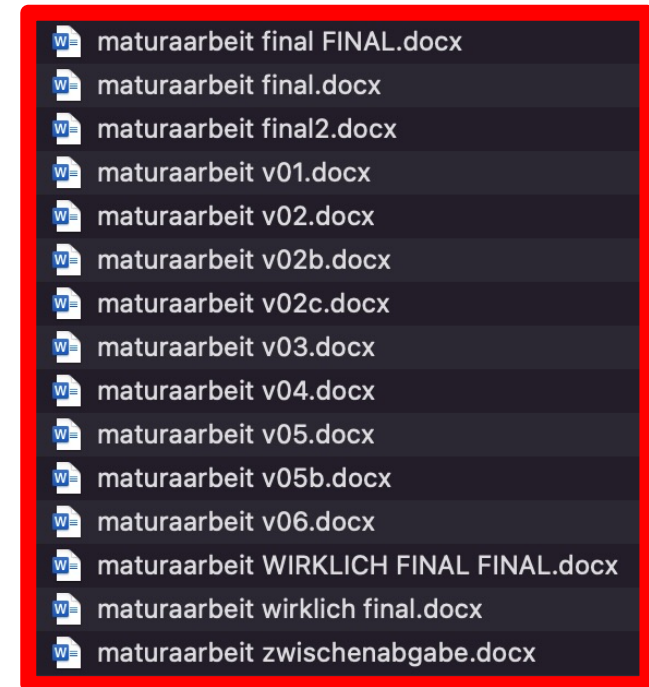
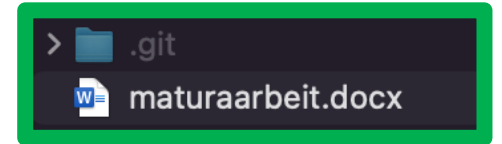


- **Software zur Versionsverwaltung**
- Absoluter **Standard** in Softwareentwicklung -> Must Know!
- Geeignet für Verwaltung von Projekten bestehend aus **Textdateien** und *nicht* Binärdateien
- **Geeignet für:**
 - **Programmierprojekte** (Python, C#, Websites, ...)
 - **LaTeX**
- **NICHT** geeignet für:
 - Office Dokumente (Word, PowerPoint, ...)
 - PhotoShop, ...
 - Hier besser OneDrive, Dropbox, iCloud, Google Drive, ...

Git in a Nutshell



- **Repo:** Git-Repository: Projekt/Ordner, welches mit Git verwaltet wird
- **Versionierung:**
 - Kann beliebig viele Versionen von Repo erstellen
 - Wird aber nur *eine angezeigt* (-> kein Chaos!)
 - Alle Versionen können aber einfach wiederhergestellt werden
 - Wird alles in verstecktem Ordner '.git' gespeichert.
 - Git ist **clever**: Macht bei neuer Version **nicht komplette Kopie** ...
 - ... sondern merkt sich nur die **Differenz** zur letzten Version.
- **Branches:**
 - Kann gleichzeitig an verschiedenen Versionen parallel arbeiten
 - Kann diese dann zusammenfügen (mergen)



Git in a Nutshell



- Entwickelt von Linus Torvalds (Erfinder von Linux OS)
- Name 'Git': britischen Umgangssprache für 'Blödmann'.
- Linus Torvalds Quotes:
 - "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'."
 - "The joke 'I name all my projects for myself, first Linux, then git' was just too good to pass up. But it is also short, easy-to-say, and type on a standard keyboard. And reasonably unique and not any standard command, which is unusual."

I DID NOT

CREATE GIT TO MAKE YOU SMILE

I HATED GIT



**NOW I HATE
IT LESS**

GitHub in a Nutshell

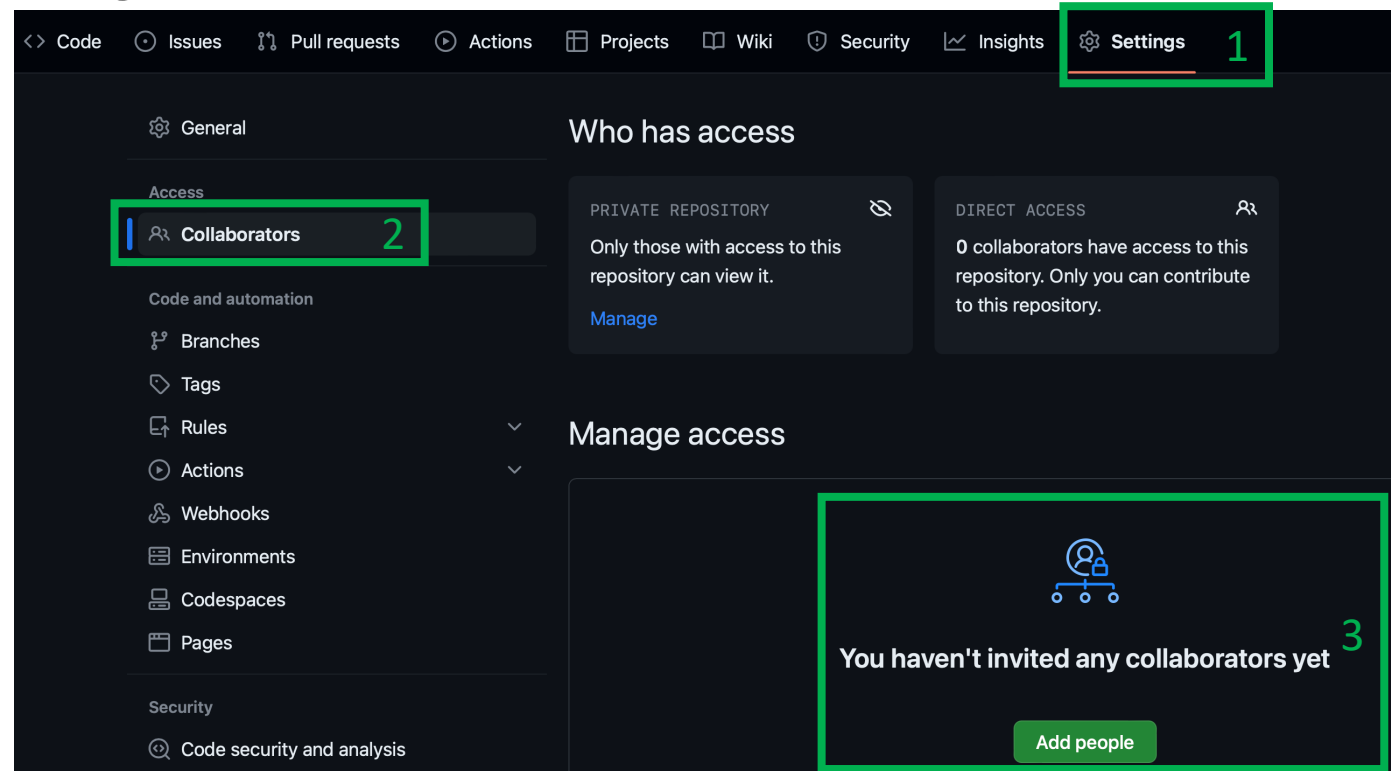


- **Onlinedienst um Git-Repositories zu Verwalten**
- **Warum GitHub?**
 - Backup
 - Später (z.B. im Studium) immer noch verfügbar
 - Collaboration: mehrere Personen zusammen am gleichen Repo arbeiten
 - Open Source: Code allen zur Verfügung stellen
- Good practice: Erstelle für **jedes Programmierprojekt** ein **eigenes Git(Hub)-Repo**

Collaborations



- **Zusammen am gleichen Repo arbeiten:**
 - Gemeinsames Projekt
 - Abgabe von Hausaufgaben, Aufträgen in Schule
- Repo **freigeben** →
- Gibt noch viele weitere Möglichkeiten wie Branches, ...





Remote repository on GitHub



Chan's machine

Maria's machine

D'Angelo's machine



Local clone of repo



Local clone of repo



Local clone of repo

How to Git(Hub)?

Setup

1. Installation Git:

- macOS / Linux: bereits installiert
- Windows: <https://gitforwindows.org>

2. GitHub-Account erstellen:

- <https://github.com>
- Verwende private Mail-Adresse, die du in 20 Jahren noch haben wirst!

3. Versteckte Dateien & Ordner anzeigen können

- Z.B. '.git'-Ordner
- macOS: Cmd + Shift + .
- Windows: Datei-Explorer: Ansicht > Zeigen > Ausgeblendete Elemente

Bei Installation von git unter Windows:

- „Use Git from Git Bash only“
- „Use the OpenSSL library“
- „Checkout Windows-style, commit Unix-style line endings“
- „Use MinTTY“
- Wähle bei den „Configuring extra options“ alle Optionen an



Repo erstellen

- Empfohlenes Vorgehen:
 - a) Neues **Repo auf GitHub** erstellen (†)
 - b) Auf Computer **klonen** (lokale Kopie des Repos)
 - c) Auf Computer lokale Kopie des Repos **bearbeiten**
- Empfehlung: Führe git-Befehle in **Konsole** aus. Good Practice! (‡)
- Alternativen:
 - †Kann neues Repo lokal auf Computer erstellen mit `git init`
 - ‡GUI-Programme wie GitKraken

a) Neues Repo auf GitHub erstellen

1. Repo-Name
2. Private or Public?
 - -> meist private!
 - Kann später gezielt teilen
3. Add gitignore
 - File '.gitignore'
 - Textdatei welche angibt, welche Files oder Filetypen von git ignoriert werden sollen
 - Wähle passendes Template für Projekt aus (Python, VisualStudio für C#, ...)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * / **Repository name *** 1

Great repository names are short and memorable. Need inspiration? How about **shiny-lamp** ?

Description (optional)

☐ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☒ **Private** 2
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

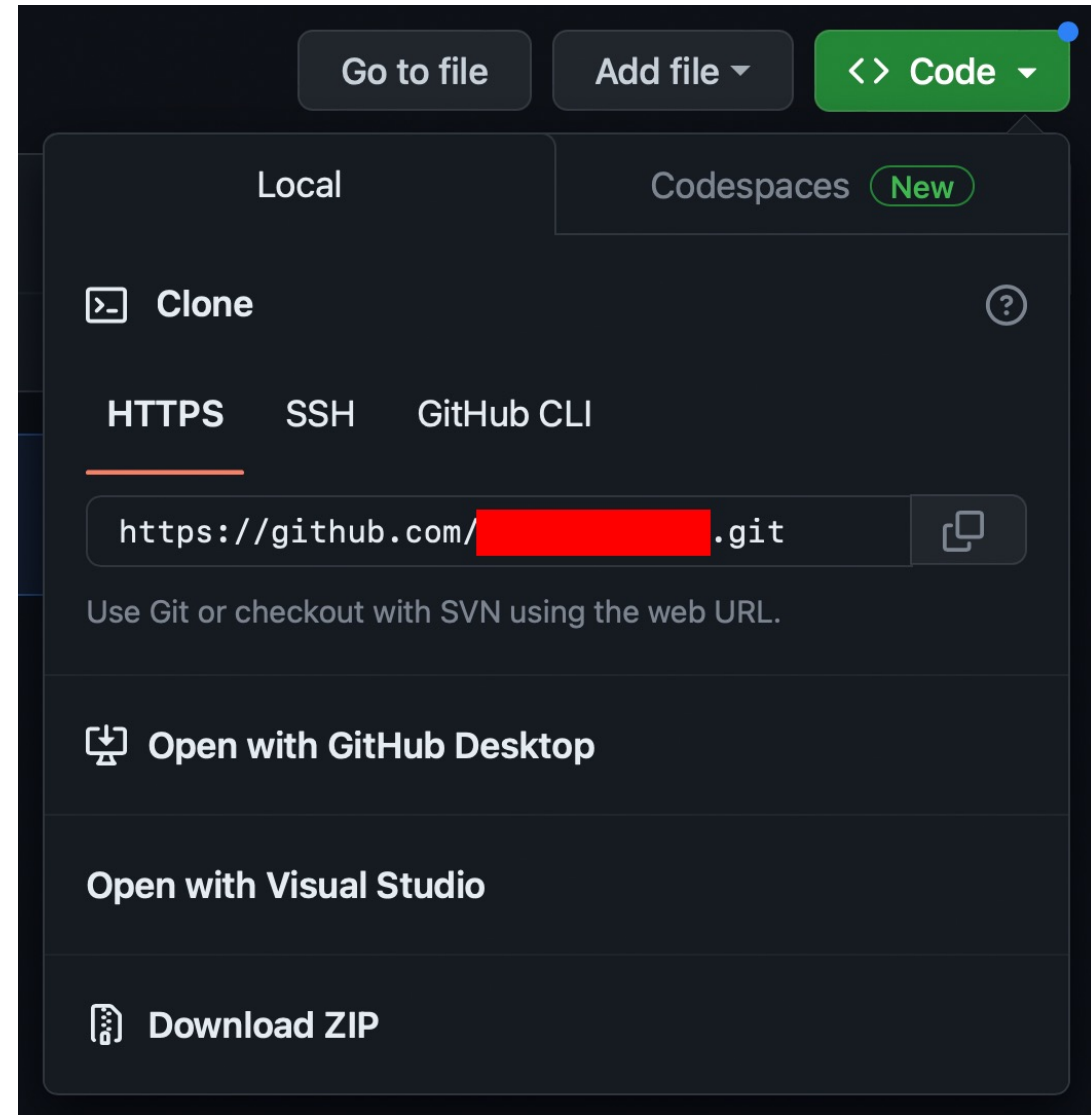
Add .gitignore 3

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

b) Auf Computer klonen

- Link zu GitHub-Repo kopieren
- Auf Computer in Konsole:
 - In gewünschten Überordner **navigieren** mit `cd` (change directory).
 - **Repo klonen**
`git clone <repo name>`
- Lokale Kopie ist nun auf Computer
- -> kann darin arbeiten
- Achtung: Bei erster Durchführung muss mit **GitHub-Account verbinden**
-> befolge Schritte



c) In Repo arbeiten

- Normales Arbeiten: Füge Files hinzu, verändere diese usw. ...
- Wichtigster Befehl: `git status`
- **Neue Version** von Repo erstellen:
 - Neue **Files hinzufügen**:
 - Einzeln: `git add <file name>`
 - Alle auf 1x: `git add .` (mit Punkt am Schluss)
 - Version committen: `git commit -am "describe commit"`
 - Füge immer kurze aber passende Beschreibung der Version hinzu, z.B. "fixed division by zero bug".
 - Achtung: neue Version existiert erst lokal auf Computer, muss noch ...
- **auf GitHub pushen**: `git push`
- Zu diesem Zeitpunkt sind **beiden Versionen** (lokal & auf GitHub) von Repo **identisch**.

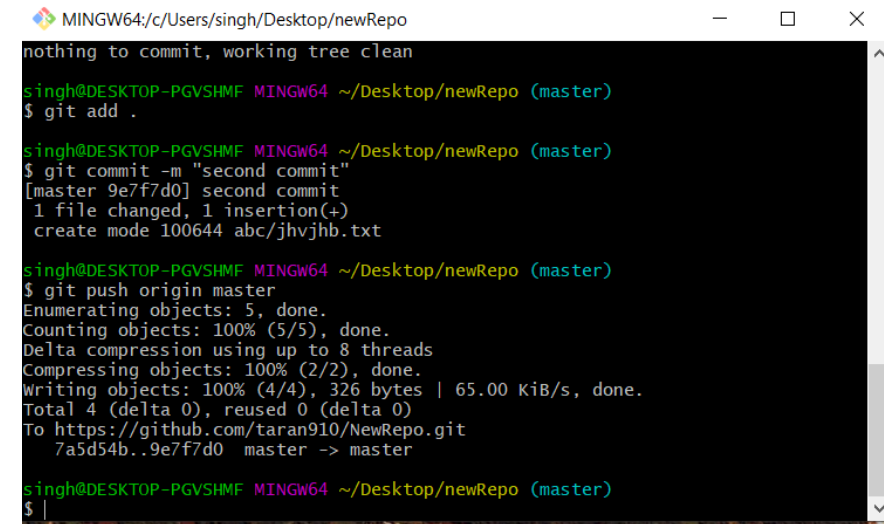
Wichtigste git-Commands

- **Übersicht** verschaffen über Status des Repositorys: `git status`
- Neue Files **hinzufügen**:
`git add <file name>` oder `git add .`
- **Neue Version** erstellen, Change committen:
`git commit -am "describe your commit here"`
- Auf GitHub **pushen**: `git push`

Wichtigste Console-Commands

- Verwende Konsole:
 - «**Git Bash**» auf Windows
 - «**Terminal**» auf macOS
- **cd** <subdirectory name> : change directory
- **cd** .. : wechsle zu Überordner
- **ls** : Zeige alle Unterordner & Files im aktuellen Ordner an
- **pwd** : zeige Pfad zu aktuellem Ordner an
- Weitere wichtige Befehle:

<https://www.digitalocean.com/community/tutorials/linux-commands>



```
MINGW64:/c/Users/singh/Desktop/newRepo
nothing to commit, working tree clean

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git add .

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git commit -m "second commit"
[master 9e7f7d0] second commit
1 file changed, 1 insertion(+)
create mode 100644 abc/jhvjhb.txt

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 326 bytes | 65.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/taran910/NewRepo.git
7a5d54b..9e7f7d0 master -> master

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$
```

Wichtigste Console-Commands



Top 50 Linux Commands you must know



1.is	1.clear	1.diff	1.kill and killall	1.apk, pacman, yum, rpm
2.pwd	2.echo	2.cmp	2.df	2.sudo
3.cd	3.less	3.comm	3.mount	3.cal
4.mkdir	4.man	4.sort	4.chmod	4.alias
5.mv	5.unman	5.export	5.chown	5.dd
6.cp	6.whoami	6.zip	6.ifconfig	6.whereis
7.rm	7.tar	7.unzip	7.traceroute	7.whatis
8.touch	8.grep	8.ssh	8.wget	8.top
9.in	9.head	9.service	9.ufw	9. useradd
10.cat	10.tail	10.ps	10.iptables	10.passwd

- <https://www.digitalocean.com/community/tutorials/linux-commands>

Weitere git-Infos & Commands

- Bisherige Befehle sind nur Spitze des Git-Eisbergs
- Wiki:
https://sca.ksr.ch/doku.php?id=informatik:git_github

Informatik allgemein

Python for C# developer and vice versa

Arduino Programmieren

Websites

Git & GitHub



Need help?

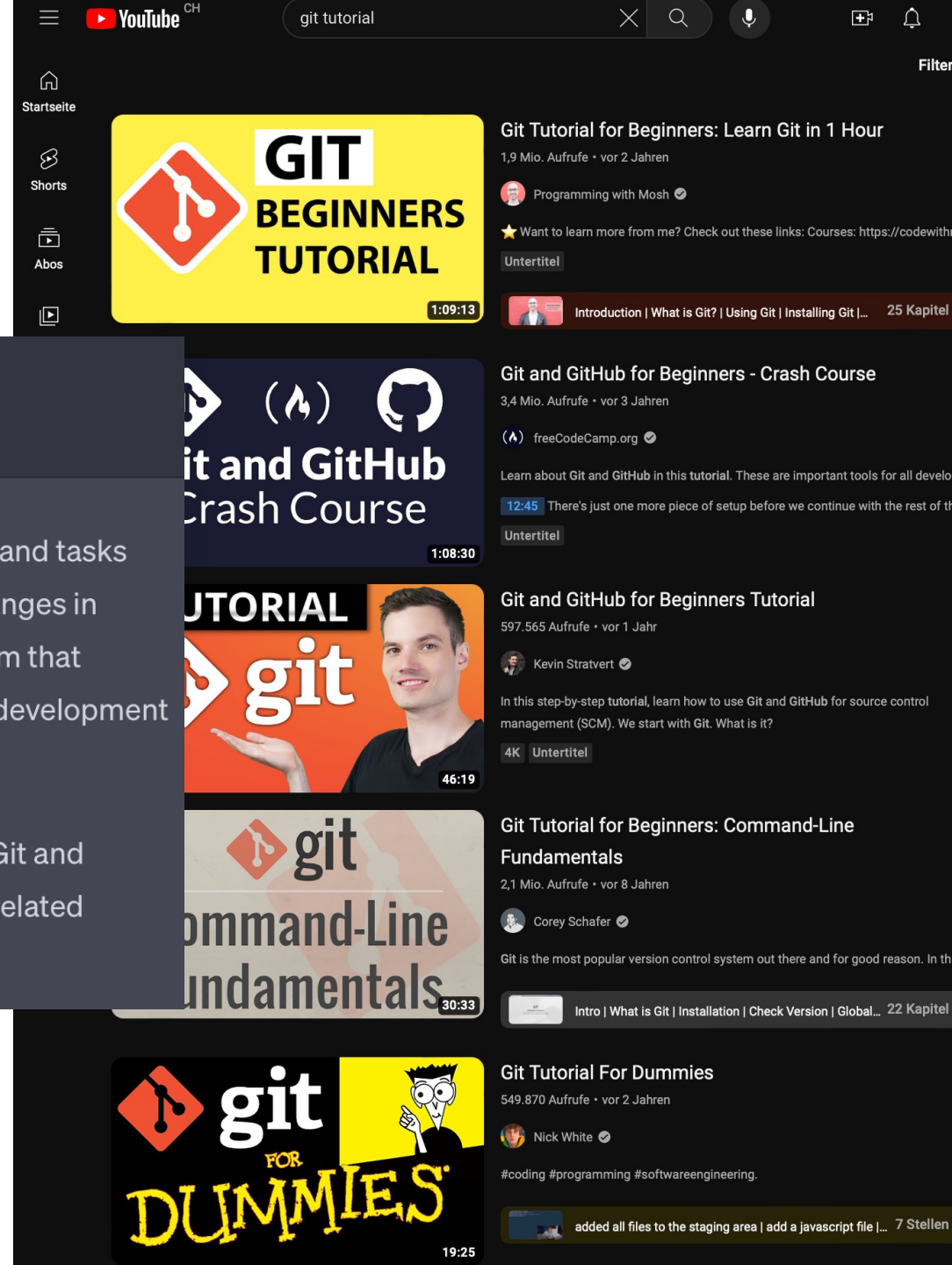
AN

Hi, do you know about git and GitHub? Can you help me?



Yes, I'm familiar with Git and GitHub, and I can certainly help you with questions and tasks related to them. Git is a distributed version control system used for tracking changes in source code during software development, while GitHub is a web-based platform that provides hosting for Git repositories and offers collaboration tools for software development teams.

Feel free to ask any specific questions or let me know how I can assist you with Git and GitHub, whether it's about basic concepts, commands, workflows, or any other related topics.





Git merge