

Willst du wissen, wie GitHub und Docker funktionieren? Hier wirst du das lernen.

github und Docker

Eine Dokumentation über github und Docker

Eril Elibol

Inhalt

Klonen des Repositories	2
Einrichtung der Entwicklungsumgebung	2
Erstellung der README.md	2
Verwendung von Git (Commit, Push)	3
Erstellung und Nutzung von Docker-Containern	4
Quellenangabe	5

Klonen des Repositories

Zuerst wird das Repository geklont. Dazu einfach den Befehl `git clone` «Repository-URL» in das Terminal `git Bash` eingeben. So wird eine Kopie des Projekts auf den eigenen Computer geladen, und alle Dateien sind sofort verfügbar.

(z.B. `git clone https://github.com/benutzername/repositoryname.git`)

```
Eril@DESKTOP-2KKEMKL MINGW64 ~
$ git clone https://github.com/Eril42/docker-nodejs-sample
fatal: destination path 'docker-nodejs-sample' already exists and is not an empty directory.
```

(Bei mir existiert schon ein Clone)

Einrichtung der Entwicklungsumgebung

Jetzt geht es an die Einrichtung der Entwicklungsumgebung. Die notwendigen Tools sollten bereitgestellt werden, wie zum Beispiel ein Code-Editor, der oft Visual Studio Code ist. Je nach Projekt müssen möglicherweise auch einige Bibliotheken (Extensions können auch nützlich sein) installiert werden. Diese Schritte am besten im `README.md` festhalten, um später alles nachschlagen zu können.

Erstellung der README.md

Die `README.md`-Datei ist eine wichtige Informationsquelle für das Projekt. Sie sollte grundlegende Informationen enthalten, darunter:

Projektname: Der Titel des Projekts.

Installation: Eine Schritt-für-Schritt-Anleitung, wie man das Projekt installiert.

Nutzung: Eine Erklärung, wie das Projekt verwendet wird.

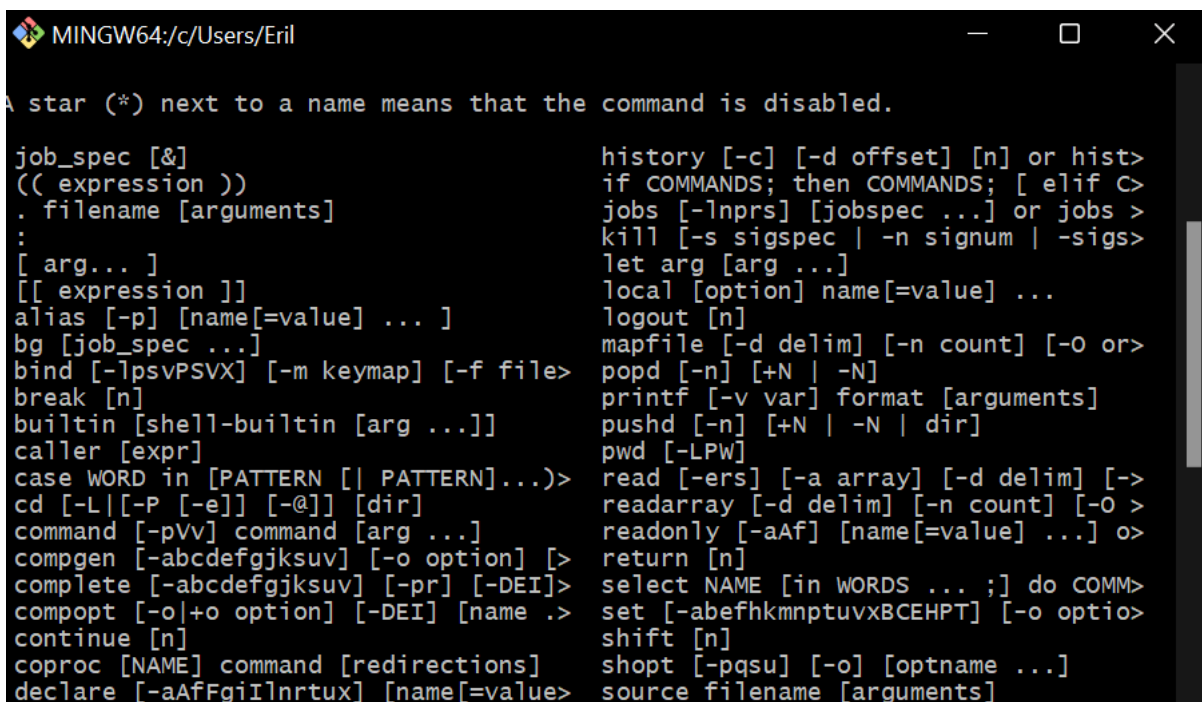
So haben alle eine gute Übersicht. Ziel ist es eine leicht verständliche Anleitung zu erstellen

```
1 # Thema: Installation des Projekts
2
3 ## Hier findest du eine Anleitung fuer die Schritte. Ich hoffe es kann dir helfen.
4
5 **Repository klonen**
6 ```bash
7 git clone https://github.com/benutzername/repositoryname.git
8 ```
9
10 **In das Verzeichnis wechseln**
11 ```bash
12 cd repository
13 ```
14
15 **Installation der notwendigen Pakete und Docker-Konfiguration und -Installation**
16 ## Du musst Docker herunterladen. Dafuer gehe auf: https://docs.docker.com/engine/install/
17 ## Schau was fuer ein Prozessor du hast und waehle das richtige Docker Version aus und klicke auf die Hyperlink, um die Datei herunterzuladen.
18 ## Offne die Installer und klick immer auf weiter
19 ## Nun sollte es funktionieren, falls es nicht funktioniert, sollst du cmd.exe als Administrator oeffnen und folgende Code ausfuehren
20 ```
21 wsl --update
22 ```
23 ## Jetzt sollte es gehen, da die wsl nicht mehr "outdated" ist.
24
25 **Starten der Applikation in einem Docker-Container**
26 ## Stelle sicher, dass Docker installiert ist.
27 ```bash
28 docker --version
29 ```
30
31 ## Oeffne das Terminal und erstelle einen neuen Ordner fuer dein Projekt:
32 ```bash
33 cd
34 mkdir mein-node-app
35 cd mein-node-app
36 ```
```

Verwendung von Git (Commit, Push)

Für die Versionskontrolle wird Git verwendet. Wenn Änderungen an den Dateien vorgenommen wurden, sollten sie zuerst mit `git add «Dateiname»` zur Staging-Area (Das heisst Zwischenspeicher) hinzugefügt werden. Danach kann mit `git commit -m "kurze Beschreibung der Änderungen"` ein Commit erstellt werden. Um die Änderungen ins Remote-Repository (Cloud Zwischenspeicher) zu übertragen, einfach «`git push`» eingeben. So bleibt alles auf dem neuesten Stand.

Hier kann man nützliche und hilfreiche Commands sehen. (gib `help` in Git Bash ein, um weiter Commands zu sehen)



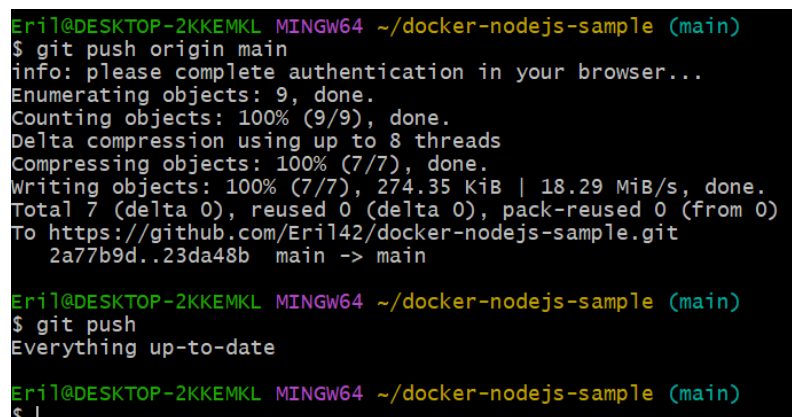
```

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f file]
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...)>
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abcdefgjkuv] [-o option] [>
complete [-abcdefgjkuv] [-pr] [-DEI]>
compropt [-o|+o option] [-DEI] [name .>
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFgiIlNrtux] [name[=value>
history [-c] [-d offset] [n] or hist>
if COMMANDS; then COMMANDS; [ elif C>
jobs [-lnprs] [jobspec ...] or jobs >
kill [-s sigspec | -n signum | -sigs>
let arg [arg ...]
local [option] name[=value] ...
logout [n]
mapfile [-d delim] [-n count] [-O or>
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LPW]
read [-ers] [-a array] [-d delim] [->
readarray [-d delim] [-n count] [-O >
readonly [-aAf] [name[=value] ...] o>
return [n]
select NAME [in WORDS ... ;] do COMM>
set [-abefhkmnptuvxBCEHPT] [-o optio>
shift [n]
shopt [-pqsu] [-o] [optname ...]
source filename [arguments]
```

Ich habe ein PDF-Datei in diese Ordner hochgeladen. Um den Dateien in github Repository hochzuladen, macht man:

```
cd docker-nodejs-sample
git add .
git commit -m «commit nachricht »
git push
```



```

Eril@DESKTOP-2KKEMKL MINGW64 ~/docker-nodejs-sample (main)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 274.35 KiB | 18.29 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Eril42/docker-nodejs-sample.git
  2a77b9d..23da48b  main -> main

Eril@DESKTOP-2KKEMKL MINGW64 ~/docker-nodejs-sample (main)
$ git push
Everything up-to-date

Eril@DESKTOP-2KKEMKL MINGW64 ~/docker-nodejs-sample (main)
$ !
```

Erstellung und Nutzung von Docker-Containern

Docker wird genutzt, um Anwendungen in Containern zu isolieren. Um einen neuen Container zu erstellen, wird ein Dockerfile benötigt, das die notwendigen Anweisungen enthält. Mit dem Befehl `docker build -t «Container-Name»` wird der Container gebaut. Um ihn dann zu starten, wird `docker run «Container-Name»` eingegeben. Docker macht die Verwaltung der Entwicklungsumgebung einfacher und sorgt für eine einheitliche Umgebung. Dafür habe ich die Anleitung von der offiziellen Docker Webseite benutzt. Schritt für Schritt erkläre ich hier:

- Verzeichnis von der gewünschte Datei in cmd eingeben

You should now have at least the following contents in your `docker-nodejs-sample` directory.

```
├─ docker-nodejs-sample/  
| └─ spec/  
| └─ src/  
| └─ .dockerignore  
| └─ .gitignore  
| └─ compose.yaml  
| └─ Dockerfile  
| └─ package-lock.json  
| └─ package.json  
└─ README.md
```

Abbildung 1

- Applikation im Hintergrund laufen lassen

Run the application

Inside the `docker-nodejs-sample` directory, run the following command in a terminal.

```
$ docker compose up --build
```

Abbildung 2

- Nun soll man die gewünschte Applikation öffnen können. Dafür kann man noch diesen letzten Schritt machen.

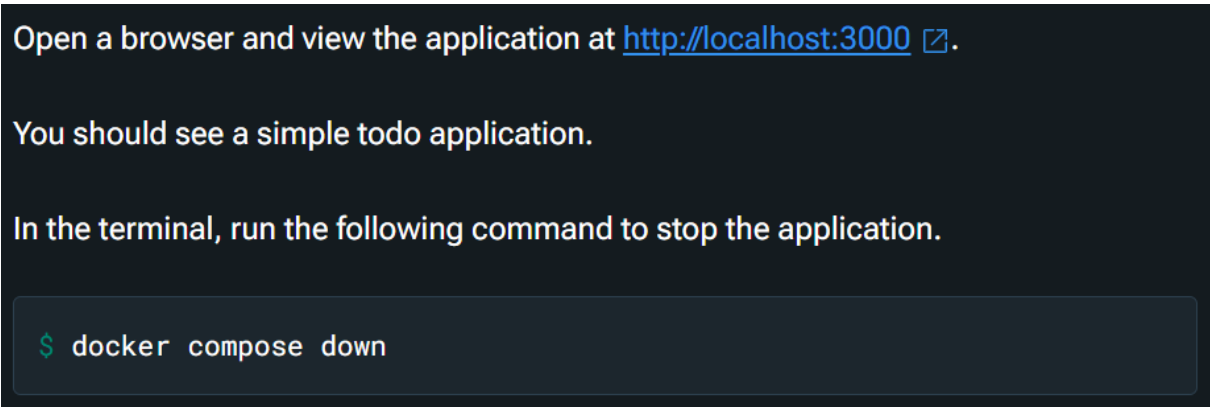


Abbildung 3

Quellenangabe

ChatGPT

> Es wurde verwendet, um den Text grammatisch zu verbessern (kein neuer Inhalt)

<https://docs.docker.com/guides/nodejs/containerize/>

> Es wurde für die Erstellung des Docker Container verwendet. Es ist eine hilfreiche Anleitung. Alle Abbildung sind Bilder, die als Screenshot von Docker offizielle Webseite gemacht wurden.

Abbildung 1: (01.11.2024) Docker

Abbildung 2: (01.11.2024) Docker

Abbildung 3: (01.11.2024) Docker