Lerndokumentation Docker, Markdown, Git

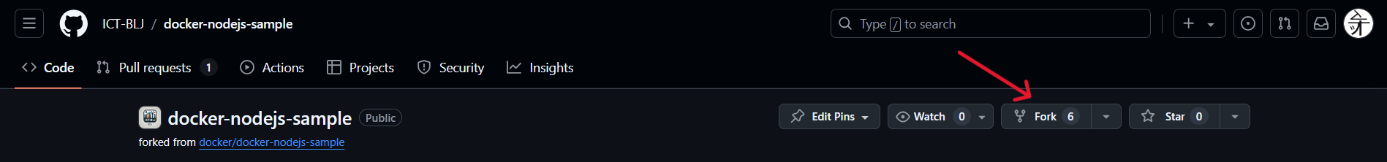
Inhaltsverzeichnis:

# Dokumentation

## Klonen des Repository

### Schritt 1: Start

Um eine [Fork](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo) auf [GitHub](https://www.github.com/) erstellen zu können, muss man zuerst auf das Repository gehen, was man "forken" will. Dort kann man auf den Fork-Knopf drücken.



### Schritt 2: Konfigurieren der Fork

Es sollte sich eine Seite öffnen, bei der man die Fork konfigurieren kann: Besitzer, Name und Beschreibung können alles angepasst und verändert werden.

A screenshot of a computer

Description automatically generated

### Schritt 3: Link zum eigenen Repository kopieren

Nun muss man den Link zu seinem eigenen Repository kopieren, um danach eine lokale Kopie anfertigen kann. Die Links sollten etwa so aussehen: https://github.com/deinBenutzername/nameDeinesRepository.

Man kann dort jeweils auch sehen, von wo man das Repository geforkt hat.

A screenshot of a computer

Description automatically generated

### Schritt 4: Repository lokal klonen

Danach geht man in den Ordner, in den man das Repository kopieren will (erfolgt mit **cd path/to/desired/location**). Danach kann man mit **git clone <Link to your repository>** das Repository von GitHub lokal duplizieren. A screen shot of a computer

Description automatically generated

## Einrichtung der Entwicklungsumgebung

### Grundsätzliches

Grundsätzlich muss man an seiner IDE nicht viel verändern. Eventuell müssen Sprachpakete als Extensions installiert werden, da die Programmiersprachen nicht von der Systemversion aus unterstützt sind. Für dieses Projekt benötigt man keine weiteren Extensions, aber es ist gut, wenn man einen Code Formatter wie [Prettier](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode) hat.

## Erstellung des README.md

### Basics von README.md

Das README.md ist eine spezielle Datei bei GitHub, sowie GitLab. Es handelt sich um ein Markdown File, wobei Markdown eine Programmiersprache ist, die so einfach wie möglich sein sollte. Jedoch kann man nur einfache Textstrukturen programmieren. Sie wird immer direkt unter den anderen Dokumenten angezeigt, was sie eine ausgezeichnete Option für Informationen sowie Guides macht. Sie ist meist im Wurzelverzeichnis eines Repositories zu finden und dient als erste Anlaufstelle für Benutzer, um sich über das Projekt zu informieren.

### Funktionen

* **Projektbeschreibung**: Sie bietet eine kurze und prägnante Erklärung, worum es bei dem Projekt geht.
* Zum Beispiel: Was ist das Ziel des Projekts? Welche Probleme löst es?
* **Installationsanleitung**: Eine Schritt-für-Schritt-Anleitung, wie man das Projekt auf einem lokalen Rechner installiert und ausführt.
* Zum Beispiel: Welche Abhängigkeiten müssen installiert werden?
* **Nutzung**: Erläuterungen, wie das Projekt verwendet wird, einschließlich Beispielen und Anwendungsfällen.
* **Beitragsrichtlinien**: Informationen für Entwickler, die zum Projekt beitragen möchten, wie sie vorgehen sollten, um neue Funktionen hinzuzufügen oder Fehler zu beheben.
* **Lizenz**: Eine Erklärung zur Lizenz des Projekts, die die Nutzungsrechte der Software beschreibt.

### Persönliches README zu diesem Projekt

Zu diesem Projekt habe ich auch ein README erstellt. Ich habe die genaue Installation von WSL (Windows Subsystem for Linux) und Docker dokumentiert und mit Bildern vereinfacht. Ich habe das Dokument im Style eine Installationsanleitung geschrieben, sodass (hoffentlich) jeder diese Programme ausführen und verstehen kann.

## Verwendung von Git

### Generell

Ich habe dieses Projekt mit Git auch auf GitHub geuploadet. Ich habe mit den Befehlen **git add .** alle bis zu diesem Punkt veränderte Dokumente in ein Paket "verpackt", sie mit **git commit -m "Commit Message"** "beim Postamt abgegeben" und sie zum Schluss mit **git push** "abgeschickt". Danach habe ich auf GitHub eine "Pull Request" erstellt, um die veränderten Dokumente automatisch überprüfen zu lassen, ob man die beiden Versionen der Dokumente ohne Probleme mergen kann.

## Erstellung und Nutzung von Docker-Containern

### Erstellung

Um ein Docker-Container zu erstellen, beginnt man mit **docker init**. Jedoch muss das in dem Ordner stattfinden, das zu einem Container werden sollte. Also müssen wir nochmals mit **cd** in den Ordner unserer Kopie des Repository gehen.



Danach können sie **docker init** ausführen. Docker sucht mit diesem Befehl nach einem Container mit dem Namen ihres Ordners. Falls keiner existiert, leitet Daemon einen Download-Prozess aus dem Docker Repository ein und erstellt einen Container mit ihrem Ordnernamen.

A screenshot of a computer

Description automatically generated

Beim Initialisieren des Docker Containers müssen Sie die Fragen so beantworten, dass sie auf ihre Programmierumgebung gut angepasst sind. Für unser Beispiel benutzen wir die Einstellungen, die im Bild zu sehen sind.

Als nächstes geben wir den Befehl **docker compose up --build** ein. Dies baut unseren Container und fügt automatisch die 3 Pakete - \_.dockerignore, compose.yaml, Dockerfile\_ - zu unserer IDE hinzu.

A computer screen shot of a program

Description automatically generated

Wenn der Befehl erfolgreich abläuft, können sie mit der Tasteneingabe **V** Docker Desktop öffnen, und von da aus die Webseite. Falls das nicht geht, können sie auch über **localhost:<Ihr Port, den sie eingestellt haben>** zur Seite gelangen. Bei mir ist das **localhost:3000**.

### Nutzung

In der Docker Desktop App hat man eine Übersicht über alle Container die bereits erstellt wurden. Ebenfalls haben wir eine Möglichkeit, die Container schnell zu starten oder stoppen.

A screenshot of a computer

Description automatically generated

Wenn wir einen der Container auswählen, bekommen wir eine Übersicht, in der man viele verschiedene Dinge sehen kann.A screenshot of a computer

Description automatically generated

Hier können wir den Container selbst anpassen, ihn starten und stoppen, sowie auf die dazugehörige Webseite gelangen.