Development Fundamentals

[Klonen des Repositories 2](#_Toc177563106)

[Fork erstellen 2](#_Toc177563107)

[Repository Klonen 2](#_Toc177563108)

[Einrichtung der Entwicklungsumgebung 2](#_Toc177563109)

[Erstellung der README.md 2](#_Toc177563110)

[Verwendung von Git (Commit, Push) 3](#_Toc177563111)

[Erstellung und Nutzung von Docker-Containern 3](#_Toc177563112)

# Klonen des Repositories

## Fork erstellen

Um den Fork zu erstellen, musste ich auf GitHub [docker-nodejs-sample](https://github.com/ICT-BLJ/docker-nodejs-sample) öffnen und oben rechts auf Fork klicken und danach auf Fork erstellen. Dadurch hatte ich das Projekt jetzt als eigenes Repository in meinem GitHub-Account.

## Repository Klonen

Damit ich mein GitHub-Repository auch lokal auf meinem PC habe musste ich es auf meinen Computer Klonen. Dafür öffnete ich das CMD in dem Ordner, in den ich das Repository Klonen wollte und gab den Befehl «git clone https://github.com/maelseewald/docker-nodejs-sample.git» ein. Im Anschluss konnte ich den Ordner direkt in Visual Studio Code öffnen.

# Einrichtung der Entwicklungsumgebung

Zu meiner Entwicklungsumgebung gehören ein Funktionsfähiger Laptop auf dem Visual Studio Code, Docker und Word laufen. Auch wichtig ist ein GitHub Konto auf welsches man den Zugriff hat. Wichtig ist es auch das ich ein gutes WLAN habe und somit schnell Uploads und Downloads machen kann.

# Erstellung der README.md

Da es bereits ein README in dem Ordner hatte, indem die Aufgabenstellung stand, musste ich kein neues erstellen. Also löschte ich den Text im README, sodass ich Platz für meine eigenen Informationen hatte. Im README machte ich eine Schritt-für-Schritt-Anleitung, von wie man ein Repository klont, bis wie man es mit Docker laufen lassen kann.

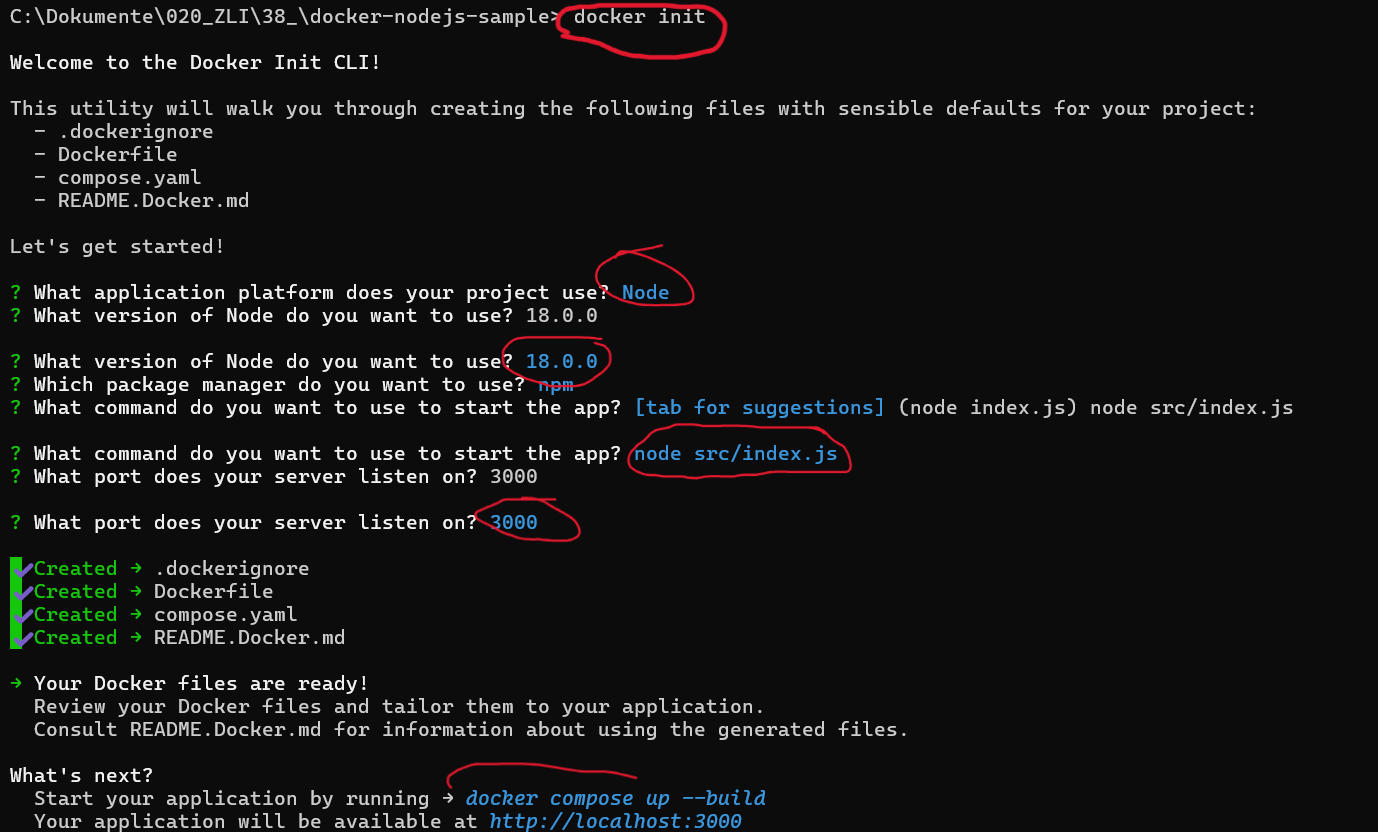
# Verwendung von Git (Commit, Push)

Wenn man ein Lokales-Dokument von seinem Computer auf GitHub hochlädt und es dann immer Aktualisieren möchte, kann man das mit verschiedenen Comands machen. Das ist immer die gleiche Abfolge von verschiedenen Comands.  
1. git add .  
2. git commit -m "der Name des Updates"  
3. git push  
Git add . verschiebt Änderungen aus dem Arbeitsverzeichnis in die Staging-Umgebung danach werden die Änderungen von git commit -m "der Name des Updates" in git gespeichert damit man später noch zu früheren Zeitpunkten zurückkann. Am Ende kann man die Änderungen noch mit git push zu GitHub schicken.

# Erstellung und Nutzung von Docker-Containern

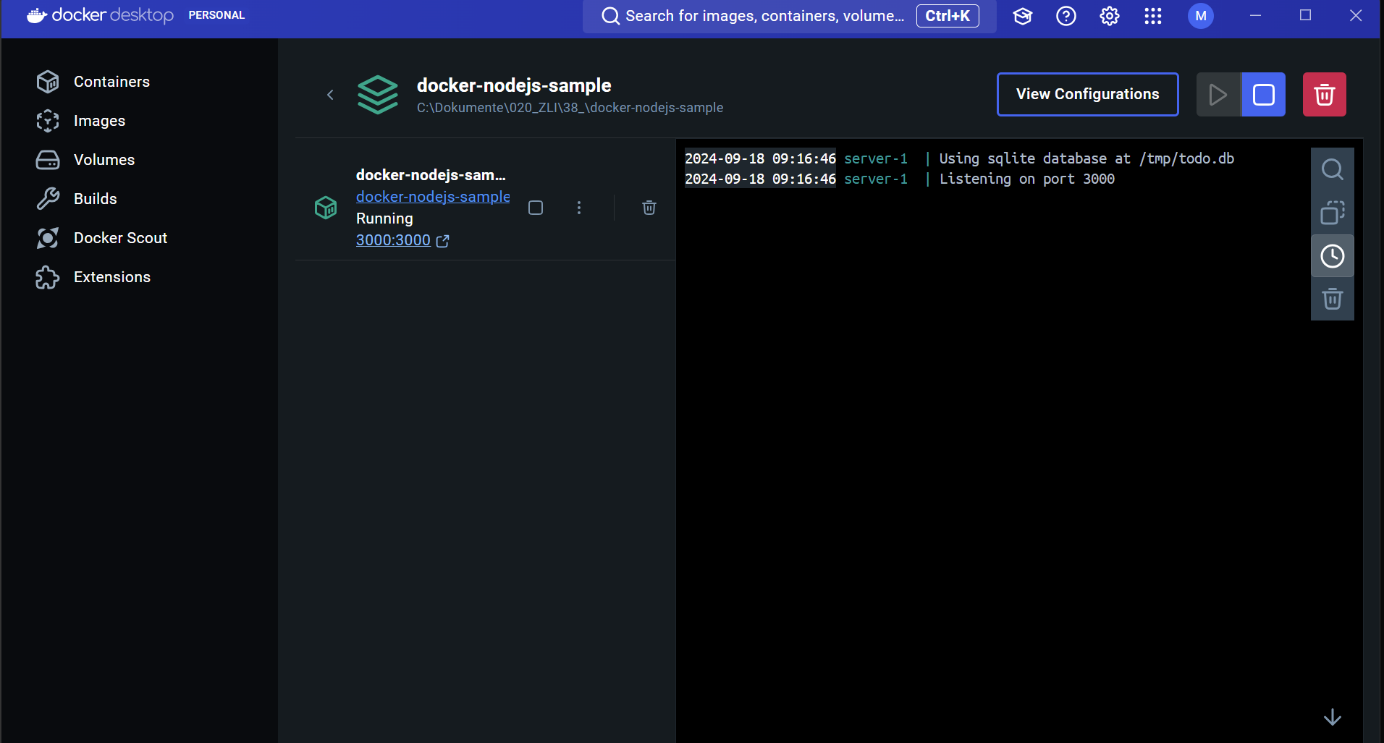
Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte BeschreibungUm ein Programm zu testen, kann man Docker verwenden. Docker ist eine sehr Benutzerfreundliche und einfach einzurichtende Testumgebung. Ich habe zuerst das CMD geöffnet und bin dann in den Ordner navigiert, den ich laufen lassen möchte. Anschliessend habe ich docker init eingegeben.

Anschliessend hat es mich einige Fragen gestellt wie zum Beispiel welchen Port mein Server laufen soll oder welche Version verwendet werden soll.

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungAls ich diese Fragen beantwortet hatte, musste ich docker compose up --build eingeben, um den Server zu starten. Danach hat es einige Minuten gedauert und der Server lief.

Danach musste ich nur noch auf den Port in Docker klicken und schon konnte ich die To-do-Liste vollkommen gebrauchen.



# Zusammenfassung

Ich hatte am Anfang des Projekts bisschen Probleme, weil ich nicht ganz verstanden hatte, wie Docker funktioniert jedoch hab ich es dann durch die Hilfe von Matteo verstanden und konnte mit dem README weiter machen. Am Ende machte Ich noch meine Dokumentation und gab den Link zu GitHub auf Moodle ab.