Dominik Scherrer

Docker-Projekt Dokumentation

24.10.2025

Inhalt

[Einleitung 2](#_Toc212214372)

[Klonen des Repositories 3](#_Toc212214373)

[Containerisieren der App 4](#_Toc212214374)

[Einrichtung der Entwicklungsumgebung 5](#_Toc212214375)

[Tests 8](#_Toc212214376)

[Erstellung der README.md & Verwendung von Git 9](#_Toc212214377)

[Schluss 10](#_Toc212214378)

# Einleitung

In dieser Dokumentation dokumentiere ich wichtige Schritte in der Abschlussaufgabe der Kalenderwoche 43 «Erstellen einer ToDo-Applikation mit Markdown, Git, GitHub und Docker»

# Klonen des Repositories

Im ersten Schritt musste ich das Repositorie von GitHub auf meinen Lokalen Rechner Klonen:

1. Zuerst habe ich den SSH-Schlüssel des Repositories kopiert.

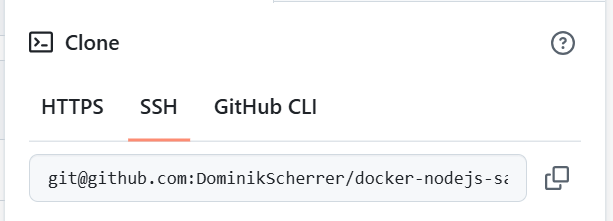


Abbildung - Screenshot von meinem GitHub

1. Danach bin ich in GitBash mit «cd» zum Ordner gewechselt, in dem ich das Repositorie haben möchte. Anschliessend habe ich mit «git clone <kopierter SSH-Schlüssel>» das Repositorie Lokal auf meinen Rechner geklont.

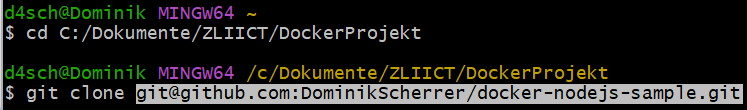


Abbildung - Screenshot von GitBash

# Containerisieren der App

In folgenden Schritten habe ich die App containerisiert und im Browser gestartet:

1. Im ersten Schritt habe ich den Docker Initialisiert, indem ich im Terminal in den Pfad zum Ordner «docker-nodejs.sample» gegangen bin und dann mit «docker init» den Docker Initialisierte:

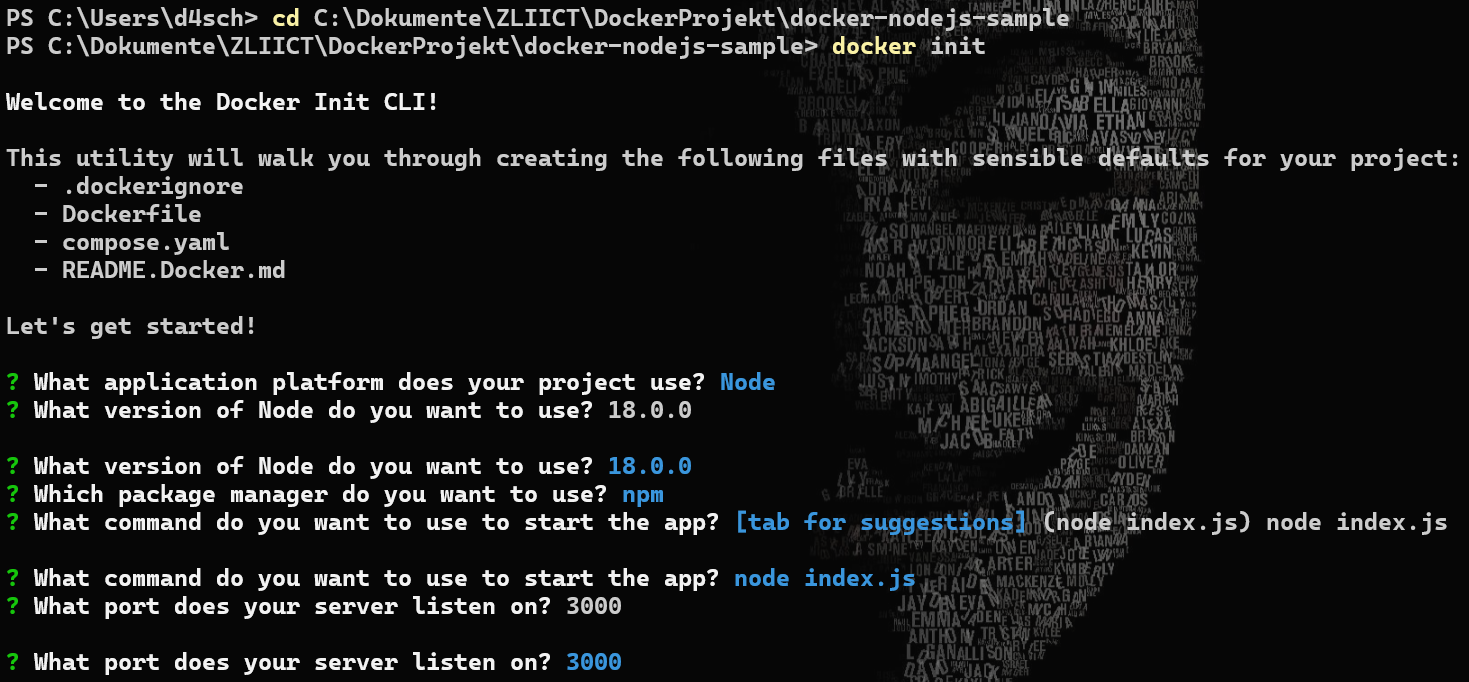


Abbildung - Screenshot aus meinem Terminal

1. Danach habe ich folgende Fragen nach der Anleitung von Dockerdocs so beantwortet:

* ? What application platform does your project use? **Node**
* ? What version of Node do you want to use? **18.0.0**
* ? Which package manager do you want to use? **Npm**
* ? What command do you want to use to start the app: **node src/index.js**
* ? What port does your server listen on? **3000**

1. Im dritten Schritt musste ich noch die Applikation mit «docker compose up –build» starten:

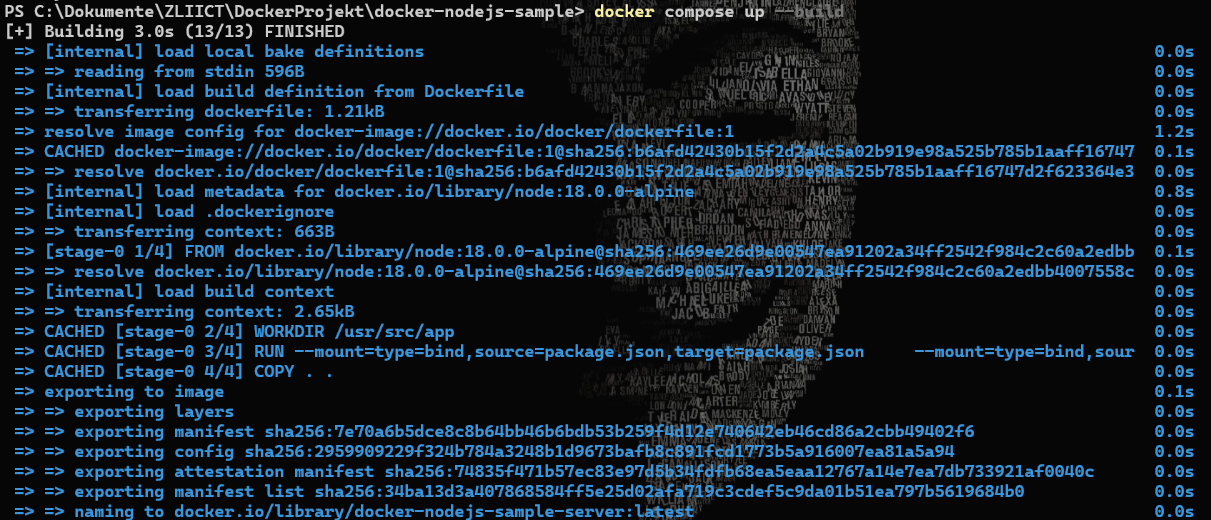


Abbildung - Screenshot aus meinem Terminal

# Einrichtung der Entwicklungsumgebung

In diesem Abschnitt habe ich die Entwicklungsumgebung für die Containeranwendung eingerichtet.

1. Im ersten Schritt habe ich im compose.yaml File die Kommentarzeichen aus den Datenbankanweisungen entfernt.
2. Danach habe ich folgende Umgebungsvariablen hinzugefügt, die die Datenbankkonfiguration angeben:

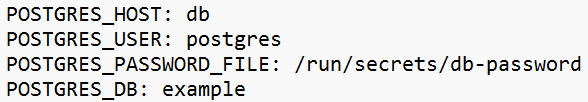


Abbildung - Screenshot von meinem Editor

1. Im dritten Schritt habe ich noch den «secrets» Abschnitt unter dem «server» Dienst hinzugefügt, damit meine Anwendung das Datenbankkennwort sicher verarbeitet.



Abbildung - Screenshot von meinem Editor

1. Nun habe ich im «docker-nodejs-sample» einen Ordner mit dem Namen «db» erstellt. Darin habe ich eine Datei namens «password.txt» erstellt, die mein Datenbankkennwort enthält.



Abbildung - Screenshot aus meinem Explorer

1. Danach habe ich im «password.txt» file ein Passwort erstellt. Das Kennwort musste in einer einzelnen Zeile stehen und durfte keine versteckte Zeichen enthalten.
2. Jetzt habe ich die Applikation gestartet, eine ToDo hineingeschriebe, die Applikation wieder schliessen und erneut starten, um zu testen, ob die ToDo’s drinnen bleiben.

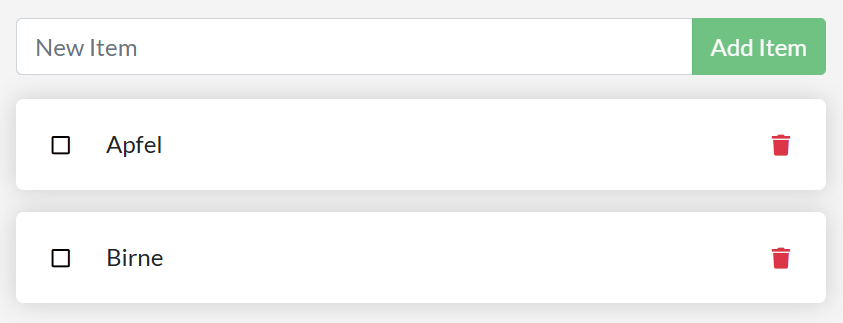


Abbildung - Screenshot von der Applikation

1. Zum Schluss habe ich das Dockerfile und compose.yaml -file nochmals bearbeitet. Hiermit habe ich ein mehrstufiges Dockfile erstellt. Ich habe folgende Sachen hinzugefügt bzw. ersetzt:

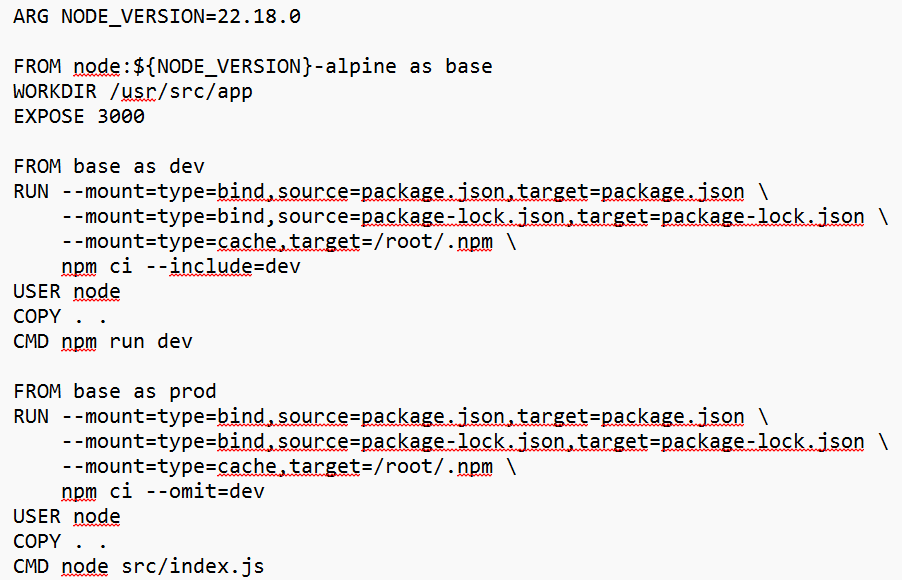


Abbildung - Screenshot aus meinem Editer («Dockerfilel»-file)



Abbildung - Screenshots aus meinem Editor («compose.yaml» -file)

# Tests

In diesen Schritten habe ich mehrere Tests durchgeführt.

1. Zuerst habe ich mit dem Befehl «docker compose run server npm run test» im Terminal die «package.json» -Datei getestet.

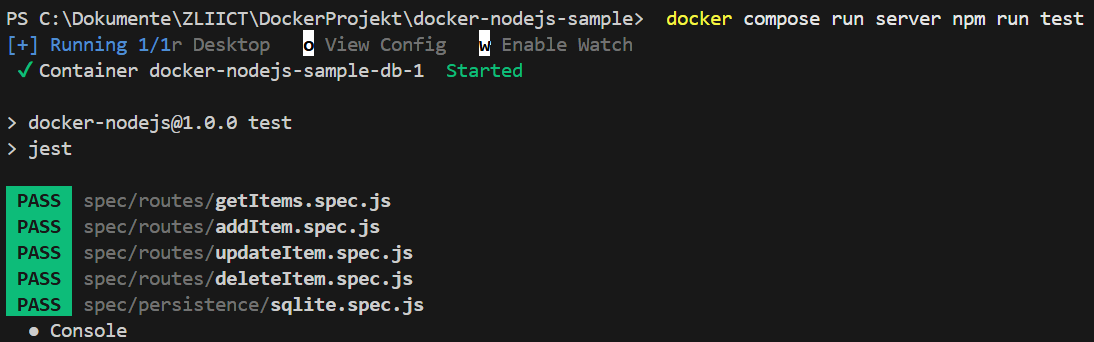


Abbildung - Screenshot aus meinem Terminal (Visual Studio Code)

1. Im zweiten Schritt habe ich die Docker-Datei aktuallisiert, um eine neue Testphase hinzuzufügen.

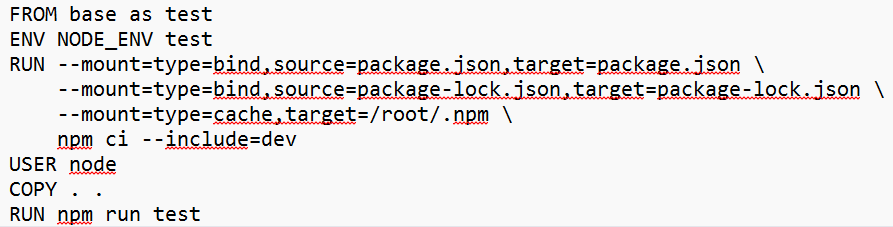


Abbildung - Screenshot von meinem Editor

1. Nun konnte ich mit dem Befehl «docker build -t node-docker-image-test --progress=plain --no-cache --target test .» ein neues Image mit der Testphase als Ziel erstellen und die Testergebnisse anzeigen.

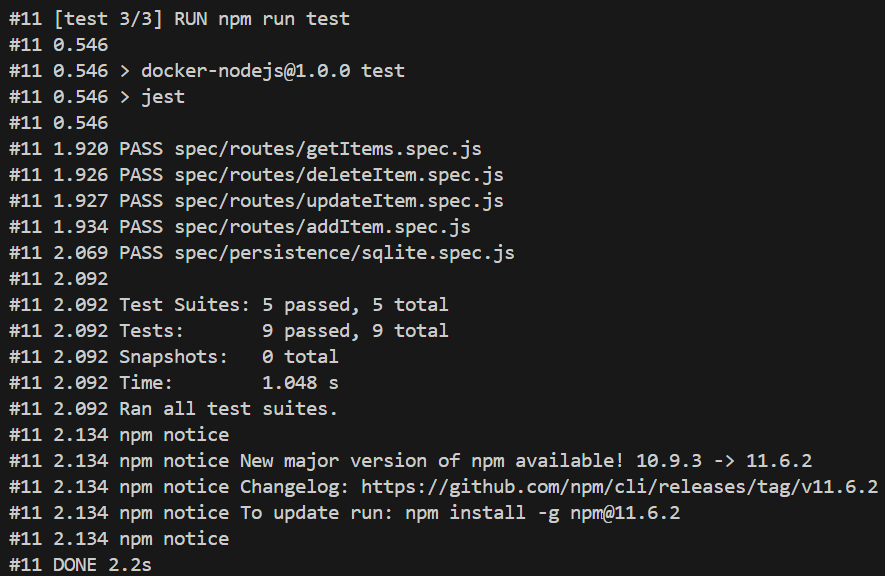


Abbildung - Screenshot Terminal - Erwartetes Ergebnis von Test

# Erstellung der README.md & Verwendung von Git

Um das readme zu machen musste ich kein neues erstellen. Ich konnte das bestehende markdown des Auftrages nehmen, den Inhalt löschen und meine Anleitung zur Installation der To-Do Applikation hineinschreiben.

Um die README.md -Dateil wieder auf GitHub Hochzuladen, habe ich folgende Schritte gemacht:

1. Während der gesammten Erstellung der Anleitung habe ich vortwährend Commits gemacht, um meine Änderungen festzuhalten. Dies macht man mit dem Git-Befehl «git commit -m «eine Nachricht für den Commit»»

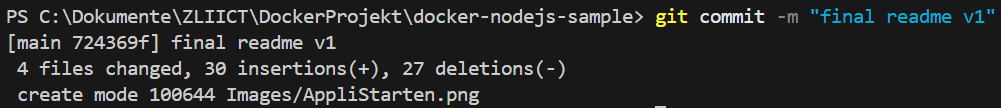


Abbildung - Screenshot Terminal Visual Studio Code

1. Im Zweiten Schritt musste ich die Änderungen wieder auf GitHub Hochladen. Das macht man mit dem Git-Befehl «git push».

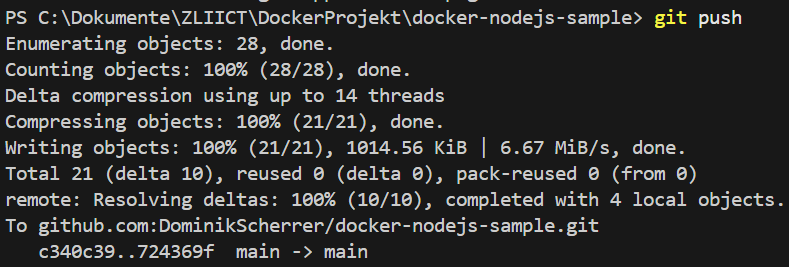


Abbildung - Screenshot Terminal

# Schluss

Leider bin ich mit der Anleitung von dockerdocs noch nicht ganz fertig geworden. Mit der Dokumentation und Anleitung bin ich fertig.