



Ü101 - Container 2

Martin | 22.09.2023

Aufgabe 1

Im Repository unter <https://github.com/martindubb/docker-task.git> findet ihr eine einfache NodeJS App. Beschreibe kurz, was die App macht.

Die App, basierend auf dem bereitgestellten Code, ist eine einfache Webanwendung, die mit NodeJS und dem Express-Framework erstellt wurde. Hier ist eine kurze Beschreibung dessen, was die App macht:

Initialisierung und Konfiguration:

Die App verwendet die Module fs (zum Arbeiten mit Dateisystemen), express (ein Web-Framework für NodeJS) und body-parser (zum Parsen von eingehenden Anforderungsdaten).

Ein Logfile wird erstellt oder verwendet, wenn bereits vorhanden. Der Standardname des Logfiles ist "mylogfile.log", aber es kann durch eine Umgebungsvariable (LOGFILE) geändert werden. Beim Starten des Logfiles wird "Logfile started" in die Datei geschrieben.

Die App verwendet den body-parser-Middleware, um URL-codierte Daten zu parsen.

GET-Route /:

Wenn ein Benutzer die Hauptseite der App besucht, wird eine HTML-Seite angezeigt, die einen Abschnitt mit dem Titel "Testeingabe" und einen darunter liegenden Text (standardmäßig "Beispieltext") enthält.

Darunter befindet sich ein Formular, mit dem der Benutzer einen Text eingeben kann. Nach dem Absenden des Formulars wird die Eingabe an die POST-Route /Eingabe gesendet.

POST-Route /Eingabe:

Wenn das Formular abgesendet wird, nimmt diese Route die Eingabe des Benutzers entgegen.

Die Eingabe wird sowohl in der Konsole als auch im Logfile protokolliert.

Der zuvor angezeigte Text (unter "Testeingabe") wird durch die neue Eingabe des Benutzers ersetzt.

Nach dem Protokollieren der Eingabe wird der Benutzer zur Hauptseite umgeleitet.

Server:

Die App hört auf Port 8080.

package.json:

Dies ist die Konfigurationsdatei für das Projekt. Es enthält Informationen über das Projekt wie den Namen, die Version und die Abhängigkeiten.

Die App hat zwei Hauptabhängigkeiten: body-parser und express.

Zusammenfassend ist die App eine einfache Webanwendung, die es dem Benutzer ermöglicht, einen Text einzugeben. Dieser Text wird dann sowohl in der Konsole als auch in einem Logfile protokolliert und auf der Hauptseite der App angezeigt.

Aufgabe 2

- a) Klone das Repo auf deinen Rechner und starte die App.
- b) Stelle sicher, dass die App den Log Output in eine Datei namens "output.log" schreibt, ohne den Code zu ändern! Wie kann das bewerkstelligt werden?
(Tipp: Zeile 6-8 in der app.js)

Erstelle einen Screenshot für die Abgabe.

```
PS C:\Users\ericy\OneDrive\Desktop\Hallo Docker> cd ..
PS C:\Users\ericy\OneDrive\Desktop> mkdir docker-task-folder

Verzeichnis: C:\Users\ericy\OneDrive\Desktop

Mode                LastWriteTime         Length Name
----                -
d-----          22.09.2023   16:41             docker-task-
                        folder

PS C:\Users\ericy\OneDrive\Desktop> cd docker-task-folder
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder> git clone https://github.com/martindubb/doc
ker-task.git
Cloning into 'docker-task'...
Receiving objects: 100% (5/5), done.
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder> cd docker-task
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> export LOGFILE=output.log
export : Die Benennung "export" wurde nicht als Name eines
Cmdlet, einer Funktion, einer Skriptdatei oder eines
ausführbaren Programms erkannt. Überprüfen Sie die
Schreibweise des Namens, oder ob der Pfad korrekt ist (sofern
enthalten), und wiederholen Sie den Vorgang.
In Zeile:1 Zeichen:1
+ export LOGFILE=output.log
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (export:String)
  [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> $env:LOGFILE="output.log"
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> node app.js
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> cat output.log
Logfile started
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task>
```

Aufgabe 3

Containerisiere die App!

Schreibe ein Dockerfile, welches ein alpine-Image nutzt und in dem NodeJS zur Verfügung steht. Baue das Image und vergib den Namen "simple-node-app". Liste alle Images von deinem Hostsystem auf.

Erstelle einen Screenshot für die Abgabe.

```
docker-task  
COMMIT_EDITMSG Dockerfile X output.log U  
Dockerfile > ...  
[...]  
[internal] load metadata for docker.io/library/node:alpine 1.8s  
[auth] library/node:pull token for registry-1.docker.io 0.0s  
[1/5] FROM docker.io/library/node:alpine@sha256:a329b146bcc99a36caa73056e60714d0911ca5c 8.7s  
=> resolve docker.io/library/node:alpine@sha256:a329b146bcc99a36caa73056e60714d0911ca5c 0.0s  
=> sha256:43a70fe68b37cf278ad0f05119f3c3ef7f7e598de13aadd0e3298474793da 6.77kB / 6.77kB 0.0s  
=> sha256:7264a8db6415046d36d16ba98b79778e18accee6ffa71850405994cffa9be 3.40MB / 3.40MB 1.4s  
=> sha256:4f5fe51f59037e9e27e758c8577b0c971906800b5fdcc60fe9b0ba893d8 49.75MB / 49.75MB 4.7s  
=> sha256:215c46f31dc4f99e5e6220dc1075367abe11aeca7b9c0be65cb98d468282c 2.34MB / 2.34MB 1.7s  
=> sha256:a329b146bcc99a36caa73056e60714d0911ca5c229ade3eb27e9283dc78c9 1.43kB / 1.43kB 0.0s  
=> sha256:ad38de8de343e0fed46efcb5b8df47f4996da2a3fb359eef3f53243b1670b 1.16kB / 1.16kB 0.0s  
=> extracting sha256:7264a8db6415046d36d16ba98b79778e18accee6ffa71850405994cffa9be7de 0.2s  
=> sha256:656ad3bd84b07ed333733109380bc2018a23fc761ca856d5b41a8c4af0be16f5 454B / 454B 1.6s  
=> extracting sha256:4f5fe51f59037e9e27e758c8577b0c971906800b5fdcc60fe9b0ba893d8e6e8c 3.6s  
=> sha256:215c46f31dc4f99e5e6220dc1075367abe11aeca7b9c0be65cb98d468282c669 0.1s  
=> extracting sha256:656ad3bd84b07ed333733109380bc2018a23fc761ca856d5b41a8c4af0be16f5 0.0s  
[internal] load build context 0.1s  
=> transferring context: 33.20kB 0.1s  
[2/5] WORKDIR /usr/src/app 0.9s  
[3/5] COPY package*.json ./ 0.0s  
[4/5] RUN npm install 3.5s  
[5/5] COPY . . 0.0s  
=> exporting to image 0.1s  
=> exporting layers 0.1s  
=> writing image sha256:744867542cdebd197c186291c82d47be27583a2956571e1b630a46fcbbd3843 0.0s  
=> naming to docker.io/library/simple-node-app 0.0s  
  
What's Next?  
View a summary of image vulnerabilities and recommendations → docker scout quickview  
● PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
simple-node-app latest 744867542cde 21 seconds ago 188MB  
hallo-docker latest d5002b52f71f 20 hours ago 1.09GB  
<none> <none> 9afcc478c8ef 21 hours ago 1.09GB  
mysql latest 483a8bc460a9 6 months ago 530MB  
mongo latest 9a5e0d0cf6de 6 months ago 646MB  
mongo-express latest 2d2fb2cab8f 23 months ago 136MB  
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> 
```

Aufgabe 4

- a) Starte einen Container vom zuvor erstellten Image "simple-node-app". Stelle sicher, dass die App auf deinem Hostsystem erreichbar ist (Stichwort Port-Binding).
Erstelle einen Screenshot für die Abgabe und stoppe den Container wieder.

```
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker run -d -p 8080:8080 --name simple-node-app-container simple-node-app  
57ee52ad56930f81453050a7a31cb8ce4de3e3c6f747dca1aeca7b427739cc09  
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> 
```

- b) Starte einen Container und stelle wie in Aufgabe 2 sicher, dass die App den Log Output in eine Datei namens "output.log" schreibt, ohne den Code in der Datei app.js zu ändern. Welche Option muss dafür beim Start des Containers mitgegeben werden? (Tipp: Umgebungsvariablen können auch im "docker run"-Befehl angegeben werden)

Erstelle einen Screenshot für die Abgabe und stoppe den Container wieder.

```
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker run -d -p 8080:8080 -e LOGFILE=output.log --name simple-node-app-log-container simple-node-app
9bb7adf9734aa54ce4c4ce3f9fde18f9ce44dfa957b913f5332c831b6533cf9e
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task>

PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker exec -it simple-node-app-log-container cat output.log
Logfile started
Logfile started
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task>
```

- c) Erstelle einen separaten Ordner und eine Textdatei mit dem Namen "output.log" auf deinem Hostsystem. Nutze einen Bind-Mount, um den Ordner und die Datei im Container zur Verfügung zu stellen und starte den Container wie im Aufgabenteil b). Der Log Output sollte nun in der Datei auf deinem Hostsystem erscheinen! Erstelle einen Screenshot für die Abgabe und stoppe den Container wieder.

```
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker exec simple-node-app-bindmount-container cat /logs/output.log
Logfile started
Hausaufgabe 22.09.2023
es funktioniert
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task>
```

```
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker exec simple-node-app-bindmount-container cat /logs/output.log
Logfile started
Hausaufgabe 22.09.2023
es funktioniert
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task> docker stop simple-node-app-bindmount-container
simple-node-app-bindmount-container
PS C:\Users\ericy\OneDrive\Desktop\docker-task-folder\docker-task>
```

Hilfen:

- Ports: <https://docs.docker.com/engine/reference/commandline/run/#publish>
- env: <https://docs.docker.com/engine/reference/commandline/run/#env>
- mount: <https://docs.docker.com/engine/reference/commandline/run/#mount>