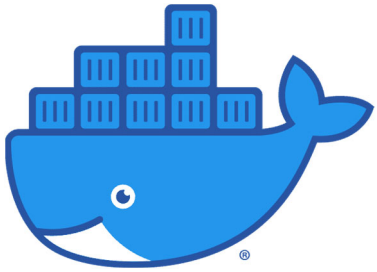
Praktische Aufgabenstellung Block 3 (1. Tag): Dockercontainer für Redirect Manager



In dieser Aufgabe werden Sie ein Backend zum Umleiten von URLs erstellen und daraus einen Dockercontainer bauen. Die Aufgabe stellen Sie dann am 2. Tag dem Dozenten zur Abnahme vor.

Für die Installation von Docker gehen Sie wie folgt vor:

- Über folgende URL können Sie sich die offizielle Docker-Desktopanwendung für Ihr Betriebssystem installieren: https://docs.docker.com/get-docker/
- Folgen Sie den Schritten für die Installation für Ihr jeweiliges Betriebssystem. Nach Abschließen und eventuellen Neustarts, sollte Ihr Terminal den docker-Befehl erkennen und Sie können mit der Entwicklung beginnen!

Entwicklung des Redirect Managers:

- Öffnen Sie Visual Studio Code und erstellen Sie ein neues Projekt
  - Initialisieren Sie Ihr Node-Projekt und installieren Sie `express`
  - Es ist einfach, einen Dockercontainer mit Umgebungsvariablen zu starten. Es ist üblich Variablen wie Bearer Tokens etc. über diese zu beziehen.
    Hierfür können Sie die eingebaute Funktion von Node mittels `process.env.VARIABLENNAME` nutzen, um dies im Code zu nutzen. Ebenfalls können Sie über den logischen ODER-Operator eine Standardzuweisung übernehmen, falls keine Umgebungsvariable gefunden wurde.
  - Importieren Sie express zum Bearbeiten der Anfragen und fs zum Speichern der Werte in einer JSON-Datei mit folgender Struktur:

    ```
    {
            "slug": "zielURL",
            "rwu": "https://rwu.de/"
    }
    ```

    Als "URL Slug" bezeichnet man den Teil einer URL, der eine Seite identifiziert und der oft direkt nach der Domain-Endung kommt
  - Zur Authentifizierung des Nutzers, der die Umleitungen verwalten kann, soll ein Bearer-Token verwendet werden. Erstellen Sie hierfür eine Middleware, die Sie für solche Endpunkte verwenden können.
    Middlewares können bei express entweder einfach global für alle Anfragen genutzt werden, bspw. als log, oder nur für bestimmte Endpunkte, indem man diese direkt nach dem Pfad aufruft.

    ```
    app.post('/entry/', authenticate, (req, res) => {});
    ```

Für eine Middleware ist es wichtig am Ende dieser den next() Parameter aufzurufen, sodass die Anfrage weiter behandelt werden kann und der Server nicht stehenbleibt.

```
const authenticate = (req, res, next) => {}
```

- Erstellen Sie folgende Endpunkte:
  - GET /:slug
    Dieser Endpunkt soll dazu dienen, dass die slug zur entsprechenden Domain aufgelöst und mittels express.redirect() umgeleitet wird
  - GET /entries (Mit Authentication Middleware)
    Soll alle Einträge aus der JSON-Datei ausgeben
  - DELETE /entry/:slug
    Eintrag mit der gegebenen Slug aus der Datei entfernen
  - POST /entry
    Soll eine URL und eine Slug entgegennehmen zum Abspeichern für späteres weiterleiten. Wird keine Slug mitgegeben, soll eine zufällige generiert werden.

Bauen des Containers:

- Sobald Sie Ihren Redirect Manager programmiert und getestet haben, kann dieser nun in einen Container gebaut werden.

- Zum Bauen eines Containers benötigt Docker die nötigen Bauschritte. Diese werden in der sogenannten Dockerfile, mit Hilfe von Dockerfile befehlen (https://docs.docker.com/engine/reference/builder/) angegeben.

- Um das Bauen zu beschleunigen, erstellen Sie eine `.dockerignore`-Datei und tragen Sie den `node_modules` Ordner ein. Ebenso kann hier, falls gewünscht, beispielsweise auch die Dockerfile selbst ignoriert werden, sodass diese nicht mitkopiert wird.

- Erstellen Sie ebenfalls die `Dockerfile`
  - Um einen Container zu bauen, benötigen Sie in den meisten fällen ein sogenanntes baseimage, dies können Sie mit der Instruktion ‚FROM' angeben.
    In diesem Beispiel nutzen Sie am besten das `node:alpine` Image, das fertige Image für den Container kleiner zu halten.
  - Geben Sie ein Arbeitsverzeichnis für den fertigen Container mittels ‚WORKDIR' an, dies ist der Pfad, wo später im fertigen Container die Dateien aus Ihrem Verzeichnis kopiert werden.
  - Mittels ‚COPY . .' kopieren Sie den gesamten Ordnerinhalt in das eben angegebene working directory
  - Da das baseimage bereits Node beinhaltet, können Sie als nächstes direkt die ‚RUN' Instruktion verwenden und mittels `npm install` die nötigen packages aus der package.json installieren.
  - Als Letztes müssen Sie Ihrem Image den nötigen Befehl geben, um den express-Server zu starten. Dies kann etwa ein Skript aus der package.json sein

```
CMD [ "npm", "start" ]
```

- Nachdem Sie die Dockerfile gespeichert haben, können Sie Ihr Image einfach per Befehl bauen, sodass dies später als Container ausgeführt werden kann.
  Zum Bauen eines Containers wird `docker build` verwendet. Dieser Befehl benötigt das Verzeichnis der eben erstellten Dockerfile und einen tag, um dieses Image zuordnen zu können. Häufig wird dieser Befehl im selben Verzeichnis ausgeführt, wo die Dockerfile liegt, weshalb hier meistens einfach nur ein ‚.' verwendet wird.

  ```
  docker build . -t redirectmanager
  ```

  Dieser Befehl wird im ebenfalls im Verzeichnis mit der Dockerfile ausgeführt und speichert das Image mit dem tag „redirectmanager" ab.

- Ist das Image erfolgreich gebaut, kann ein Dockercontainer mit diesem Image mittels `docker run` erstellt werden.

  ```
  docker run -d \
        --name redirect \
        -p 80:3000 \
        -e PORT=3000 \
        -e BEARER_TOKEN=TOKEN \
        redirectmanager
  ```

  - -d lässt den Container im Hintergrund laufen und gibt die erstellte Docker ID aus
  - --name gibt dem Container einen Namen. Wird dieser Parameter weggelassen, wird ein Zufalls-generierter Name vergeben.
  - -p gibt Ports aus dem Container nach draußen zum Hostsystem frei. Links des ‚:' ist der Port des Hosts und rechts der Port des Containers. Dies ist auch allgemeingültig für andere Befehle, die den ‚:' verwenden
  - -e key=value gibt Umgebungsvariablen an den Container weiter, die, wie oben bereits beschrieben wurde, im Code genutzt werden können.
- Die Containerlogs können mittels `docker logs redirect` eingesehen und mit der -f flag verfolgt werden.

Beachten Sie, dass bei dieser Methode des Startens die Daten verloren gehen, sollte der Container gelöscht und neu erstellt werden. Um dies zu verhindern, bietet Docker bind mounts an. Entweder können hier persistente Volumes erstellt werden, die im Dockerbereich des Betriebssystems liegen, oder direkt auf Ordner oder gar einzelne Dateien gebunden werden.

Möchten Sie nun etwa Ihre Daten persistent in Ihrem System unter /docker/redirectData.json speichern, können Sie dies mit folgendem Befehl tun

```
docker run -d \
      --name redirect \
      -p 80:3000 \
      -v /docker/redirectData.json:/usr/src/app/data.json \
      -e PORT=3000 \
      -e BEARER_TOKEN=TOKEN \
      redirectmanager
```

Wobei der rechte Pfad das working directory des eben gebauten Images ist. So bleiben die Daten über verschiedene Container gleich.