Musterlösung

IP API

Aufgabe: Entwicklung und Bereitstellung einer React-App mit Docker (und Nginx)

Ziel: In dieser Aufgabe sollst du eine React-App erstellen, die eine IP-Adresse über ein Formular entgegennimmt und dann mithilfe der ip-api.com-API zusätzliche Informationen zur IP-Adresse abruft. Schließlich sollst du die React-App in einem Docker-Container verpacken und auf einer AWS EC2-Instanz bereitstellen. Als Herausforderung soll Nginx als Reverse-Proxy verwendet werden.

React-App entwickeln
Erstelle eine React-App mit einem Formular, das eine IP-Adresse akzeptiert. Implementiere eine Funktion, die die eingegebene IP-Adresse verwendet, um einen API-Aufruf an ip-api.com durchzuführen und die erhaltenen Daten anzuzeigen.

Unset

```
npx create-react-app ip-info-app
cd ip-info-app
```

- dann in src/App.js diesen Inhalt hinzufügen

Unset

```
import React, { useState } from 'react';

function App() {
 const [ip, setIp] = useState('');
 const [data, setData] = useState(null);

 const fetchIpInfo = async () => {
  const response = await fetch(`http://ip-api.com/json/${ip}`);
  const data = await response.json();
  setData(data);
 };

 return (
  <div className="App">
```

```
        <input
         value={ip}
         onChange={e => setIp(e.target.value)}
         placeholder="IP-Adresse eingeben"
        />
        <button onClick={fetchIpInfo}>Suchen</button>
        {data && (
         <div>
          <p>Stadt: {data.city}</p>
          <p>Region: {data.regionName}</p>
          // Zeige hier weitere gewünschte Daten an.
         </div>
        )}
       </div>
      );
    }

    export default App;
```

Docker-Container für die React-App erstellen
Erstelle ein Dockerfile für deine React-App.
Baue den Docker-Container, der deine React-App enthält.

- dann im Hauptverzeichnis (ip-info-app) Dockerfile erstellen

Unset

```
FROM node:16 AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . ./
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- den Container mit diesem Befehl bauen

Unset

```
docker build -t ip-info-app:latest .
```

- nun kannst du deine Image ins Docker Hub pushen um es später auf der EC2 Instanz zu pullen
- im Terminal das eingeben

```
Unset
docker login
docker tag <name der Image> <usernameHUB>/<name der Image>
docker images
docker push <usernameHUB>/<name der Image>
```

- gehe auf deinem Docker Hub und aktualisiere um zu sehen
- kopier dir den pull befehl

AWS-Ressourcen ersetllen und deployen
Starte eine EC2-Instanz in AWS. Deploye den Docker Container auf der EC2 Instanz und stelle sicher, dass die App über das Internet erreichbar ist.

- Nachdem du die EC2-Instanz erstellt und erfolgreich den Docker darauf installiert hast

```
Unset
sudo apt-get update
sudo apt-get install docker.io
sudo systemctl start docker
docker pull <usernameHUB>/<imagename>
```

- falls permission denied, dann <sudo usermod -aG docker ubuntu>, anschließend die Maschine neu starten (exec su -l $USER)
- nochmal docker pull befehl ausführen
- Container starten

```
Unset
docker run -p 80:8 -d <usernameHUB>/<imagename>
```

- public-IP-der EC2 im Browser eingeben

Herausforderungen

Nginx als Reverse-Proxy konfigurieren
Installiere Nginx auf deiner EC2-Instanz.
Konfiguriere Nginx als Reverse-Proxy, um Anfragen an die React-App in deinem Docker-Container weiterzuleiten.
sudo rm /etc/nginx/sites-enabled/default

Schütze deine App!
Stelle sicher, dass die App nicht einfach aufgerufen werden kann. Konfiguriere NGINX so, sodass der Zugriff erst nach Eingabe von einem bestimmten Username und Passwort erfolgen kann (BasicAuth).