IPCA
INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE

Sistemas de Computação na Cloud

1º Entrega - Relatório

João Cunha, a22189
Luís Ventuzelos, a22188
Ricardo Faria, a4479

Grupo 6

1. Dockerfile

1.1 `FROM node:16`

O Dockerfile usa como base a imagem do Docker-Hub **node:16**.

1.2 `WORKDIR /usr/src/app`

Através do comando ***WORKDIR*** consideramos a pasta /usr/src/app como a nossa pasta principal de trabalho.

1.3 `COPY package*.json ./`

Através do comando ***COPY***, copiamos todas as referências de package*.json de maneira a conseguirmos instalar todas as dependências necessárias para correr a nossa API.

1.4 `RUN npm install`

O comando ***RUN*** vai correr o comando "*npm install*" que vai permitir instalar as dependências do node que estão referenciadas nos ficheiros package*.json.

1.5 `COPY . /usr/src/app`

De seguida corremos o comando ***COPY*** para copiar os restantes ficheiros da aplicação. A decisão de fazer os comandos ***COPY*** faseados foi tomada por uma questão de eficiência. Se alguma dependência do *node* der erro ao correr o comando "*npm install*" a build da imagem vai falhar antes de copiarmos os restantes ficheiros.

1.6 `EXPOSE 3000`

A instrução ***EXPOSE*****,** expõe uma porta específica com um protocolo específico dentro do Docker Container. A instrução diz ao Docker para obter todas as informações necessárias, durante o tempo de execução, de uma porta específica.

No caso da nossa imagem usaremos a porta 3000.

1.7 `CMD [ "node", "app.js" ]`

Preparamos o comando ***CMD*** para que assim que o nosso container for lançado execute o comando "*node app.js*" .

2. Docker-Compose

2.1 `version: "3.9"`

Indica que estamos a utilizar a versão 3.9 do Docker Compose, desta forma o Docker fornece os recursos apropriados.

2.2 `services`

Esta seção define todos os diferentes containers que iremos criar.

2.2.1 `api`

Este será o nome do nosso 1º serviço. O Docker Compose criará um container com esse nome.

```
build: .
ports:
   - "8080:3000"
depends_on:
   - db
```

No parâmetro ***build*** definimos a localização do ficheiro docker-compose.yml em relação ao Dockerfile, de seguida em ***ports*** mapeamos as portas do container para a máquina host e por fim adicionamos o parâmetro ***depends_on*** para que este serviço inicie após o serviço definido for iniciado. Controlando assim a ordem pela qual os serviços serão inicializados.
No serviço api mapeamos a porta 3000 do container para a porta 8080 da máquina host.

2.2.2 db

Este 2º serviço será a nossa base de dados.

```
image: "mongo"
restart: always
ports:
   - "5432:27017"
env_file:
   - .env
volumes:
   - ./scripts/mongo/init/:/docker-entrypoint-initdb.d
   - ./scripts/mongo/init:/home/mongodb
   - ./scripts/mongo/seed/:/home/mongodb/seed
   - ./data/db:/data/db
```

No parâmetro ***image*** definimos uma imagem pré-construída do mongo que será utilizada como um serviço de base de dados, no nosso caso Mongo (esta imagem está armazenada no Docker-Hub). O Docker Compose irá criar um container com uma cópia (“*fork”)* dessa imagem.
Adicionamos a instrução de ***restart:*** **always** de maneira a que sempre que o nosso container parar seja, logo de seguida, inicializado de maneira a que o container que contém a base de dados se mantenha sempre ativo.
Em ***ports*** mapeamos as portas do container expondo a porta 5432 na máquina host.
Em ***env_file*** definimos a localização do ficheiro com as variáveis de ambiente que queremos passar para o serviço (*usernames*, *passwords*, etc…) que utilizamos ,por exemplo, para autenticação na base de dados.
Por fim, definimos ***volumes***, onde estão montados os diretórios que achamos relevantes para aceder através da máquina host.
Desta forma, não teremos que reconstruir a imagem sempre que for necessário realizar alterações.
Os primeiros 3 volumes foram criados de modo a facilitar a criação de ***mock data*** na base de dados, sendo o último volume usado para armazenar todos os dados do MongoDB.