04

Criando o docker-compose.yml

Transcrição

Para utilizar o **Docker Compose**, devemos criar o seu arquivo de configuração, o **docker-compose.yml**, na raiz do projeto. Em todo arquivo de **Docker Compose**, que é uma espécie de receita de bolo para construirmos as diferentes partes da nossa aplicação, a primeira coisa que colocamos nele é a versão do Docker Compose que estamos utilizando:

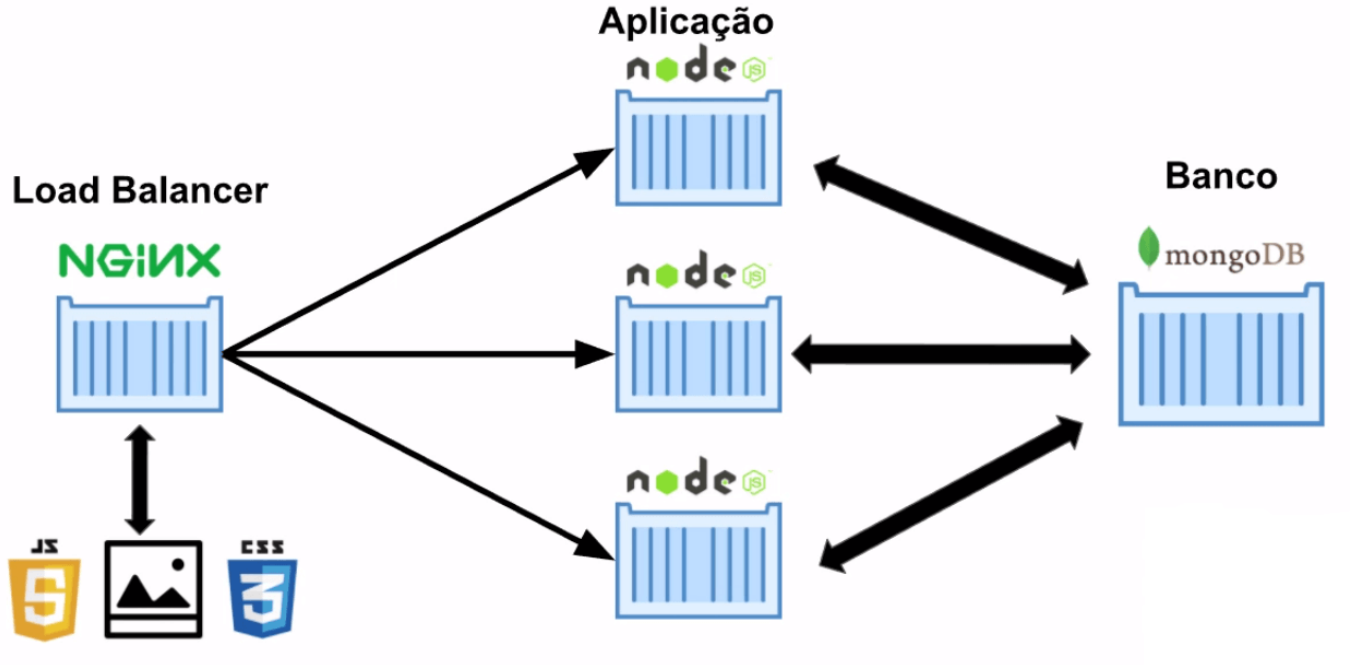
```
version: '3'
```

Estamos utilizando a versão 3 pois é a versão mais recente no momento da criação do treinamento. O YAML lembra um pouco o JSON, mas ao invés de utilizar as chaves para indentar o código, ele utiliza **espaços**.

Agora, começamos a descrever os nossos serviços, os nossos `services`:

```
version: '3'
services:
```

Um serviço é uma parte da nossa aplicação. Lembrando do nosso diagrama:



Temos NGINX, três Node, e o MongoDB como serviços. Logo, se queremos construir cinco *containers*, vamos construir cinco serviços, cada um deles com um nome específico.

Então, vamos começar construindo o **NGINX**, que terá o nome `nginx`:

```
version: '3'
services:
    nginx:
```

Em cada serviço, devemos dizer como devemos construí-lo, como devemos fazer o seu `build`:

```
version: '3'
services:
    nginx:
        build:
```

O serviço será construído através de um **Dockerfile**, então devemos passá-lo onde ele está. E também devemos passar um **contexto**, para dizermos a partir de onde o **Dockerfile** deve ser buscado. Como ele será buscado a partir da pasta atual, vamos utilizar o ponto:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
```

Construída a imagem, devemos dar um nome para ela, por exemplo **douglasq/nginx**:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
```

E quando o Docker Compose criar um *container* a partir dessa imagem, vamos dizer que o seu nome será **nginx**:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
```

Sabemos também que o NGINX trabalha com duas portas, a **80** e a **443**. Como não estamos trabalhando com HTTPS, vamos utilizar somente a porta **80**, e no próprio arquivo, podemos dizer para qual porta da nossa máquina queremos mapear a porta **80** do *container*. Vamos mapear para a porta de mesmo número da nossa máquina:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
```

```
        container_name: nginx
        ports:
            - "80:80"
```

No YAML, toda vez que colocamos um traço, significa que a propriedade pode receber mais de um item. Agora, para os *containers* conseguirem se comunicar, eles devem estar na mesma rede, então vamos configurar isso também. Primeiramente, devemos criar a rede, que não é um serviço, então vamos escrever do começo do arquivo, sem as tabulações:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"

networks:
```

O nome da rede será **production-network** e utilizará o *driver* **bridge**:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"

networks:
    production-network:
        driver: bridge
```

Com a rede criada, vamos utilizá-la no serviço:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"
        networks:
            - production-network
```

```
networks:
    production-network:
        driver: bridge
```

Isso é para construir o serviço do NGINX, agora vamos construir o serviço do MongoDB, com o nome **mongodb**. Como ele será construído a partir da imagem **mongo**, não vamos utilizar nenhum Dockerfile, logo não utilizamos a propriedade `build` . Além disso, não podemos nos esquecer de colocá-lo na rede que criamos:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"
        networks:
            - production-network

    mongodb:
        image: mongo
        networks:
            - production-network

networks:
    production-network:
        driver: bridge
```

Falta agora criarmos os três serviços em que ficará a nossa aplicação, **node1**, **node2** e **node3**. Para eles, será semelhante ao NGINX, com Dockerfile **alura-books.dockerfile**, contexto, rede **production-network** e porta **3000**:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"
        networks:
            - production-network

    mongodb:
        image: mongo
        networks:
            - production-network

    node1:
        build:
            dockerfile: ./docker/alura-books.dockerfile
```

```
            context: .
        image: douglasq/alura-books
        container_name: alura-books-1
        ports:
            - "3000"
        networks:
            - production-network

    node2:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-2
        ports:
            - "3000"
        networks:
            - production-network

    node3:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-3
        ports:
            - "3000"
        networks:
            - production-network

networks:
    production-network:
        driver: bridge
```

Com isso, a construção dos nossos serviços está finalizada.

Ordem dos serviços

Por último, quando subimos os *containers* na mão, temos uma ordem, primeiro devemos subir o **mongodb**, depois a nossa aplicação, ou seja, **node1**, **node2** e **node3** e após tudo isso subimos o **nginx**. Mas como que fazemos isso no **docker-compose.yml**?

Nós podemos dizer que os serviços da nossa aplicação **dependem** que um serviço suba antes deles, o serviço do **mongodb**:

```
version: '3'
services:
    nginx:
        build:
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"
        networks:
```

```
            - production-network

    mongodb:
        image: mongo
        networks:
            - production-network

    node1:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-1
        ports:
            - "3000"
        networks:
            - production-network
        depends_on:
            - "mongodb"

    node2:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-2
        ports:
            - "3000"
        networks:
            - production-network
        depends_on:
            - "mongodb"

    node3:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-3
        ports:
            - "3000"
        networks:
            - production-network
        depends_on:
            - "mongodb"

networks:
    production-network:
        driver: bridge
```

Da mesma forma, dizemos que o serviço do **nginx** depende dos serviços **node1**, **node2** e **node3**:

```
version: '3'
services:
    nginx:
        build:
```

```
            dockerfile: ./docker/nginx.dockerfile
            context: .
        image: douglasq/nginx
        container_name: nginx
        ports:
            - "80:80"
        networks:
            - production-network
        depends_on:
            - "node1"
            - "node2"
            - "node3"

    mongodb:
        image: mongo
        networks:
            - production-network

    node1:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-1
        ports:
            - "3000"
        networks:
            - production-network
        depends_on:
            - "mongodb"

    node2:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-2
        ports:
            - "3000"
        networks:
            - production-network
        depends_on:
            - "mongodb"

    node3:
        build:
            dockerfile: ./docker/alura-books.dockerfile
            context: .
        image: douglasq/alura-books
        container_name: alura-books-3
        ports:
            - "3000"
        networks:
            - production-network
        depends_on:
            - "mongodb"

networks:
```

```
    production-network:
        driver: bridge
```

Assim, encerramos a configuração do **docker-compose.yml**. Vamos ver como subir a aplicação a partir desse arquivo no próximo vídeo.