06

Configurando docker-compose

Bom pessoal, antes de iniciarmos a tarefa de CI/CD de nosso projeto, vamos colocá-lo de pé em um container docker e garantir que tudo está funcionando. Para que o projeto funcione corretamente, precisamos de um ambiente que suporte projetos escritos em python, com um banco de dados em MYSQL. Vamos então subir um container com os dois serviços ativos, ok? Para isso, vamos criar na raiz do projeto um arquivo chamado `docker-compose.yml`, que será responsável pelas configurações do nosso container. Em seguida vamos colocar o seguinte conteúdo:

```
version: '3'


services:
  db:
    image: mysql:5.7
    ports:
      - '3306:3306'
    environment:
       MYSQL_DATABASE: 'todo_dev'
       MYSQL_USER: 'devops_dev'
       MYSQL_PASSWORD: 'mestre'
       MYSQL_ROOT_PASSWORD: 'senha'
  web:
    image: aluracursos/django_todolist_image_build
    volumes:
      - ./env:/usr/src/app/to_do/.env
    ports:
      - "8000:8000"
    depends_on:
      - db
```

Notem que setamos a versão 3 no início do arquivo, pois estamos utilizando o `docker-ce`. Em seguida no arquivo, percebam que abaixo da palavra reservada services temos dois serviços declarados: o serviço `"db"`, que é responsável por subir a parte do banco de dados mysql onde setamos todas as configurações de ambiente e o serviço "web", que se encarrega do serviço web na porta 8000. Ainda dentro do serviço web, notem que estou utilizando uma imagem chamada `"aluracursos/django_todolist_image_build"`. Como o foco do curso é CI/CD, já deixamos preparada uma imagem docker no docker-hub, pois caso não tenham essa imagem na maquina de vocês ao subir o container, o próprio docker se encarrega de baixá-la do repositório. Com o `docker-compose.yml` criado, vamos agora em um terminal. Digite:

```
docker-compose up --build
```

Com esse comando vamos subir o container, e caso não tenhamos ainda a imagem `aluracursos/django_todolist_image_build` em nossa máquina, ela será baixada pelo docker.

Ao concluir o build, percebam que o terminal ficou preso a essa tarefa. Vamos então deixar que o mesmo rode em background, para isso digite no terminal:

```
ctrl c
```

Em seguida vamos subir novamente o container:

```
docker-compose up -d
```

Passando o parâmetro -d para o comando, o container sobe mas não prende o terminal a esta tarefa.

Agora que o container está de pé, vamos rodar os migrations iniciais que o projeto precisa. Então, digite no terminal:

```
docker-compose run web python manage.py makemigrations
docker-compose run web python manage.py migrate
```

E para criarmos um usuário para essa aplicação, ainda no terminal, digite:

```
docker-compose run web python manage.py createsuperuser
```

É isso aí pessoal, agora temos o projeto funcionando em nosso container docker.