



**Certified Tech  
Developer**

The Ultimate Degree

## Objetivos

Construir um arquivo do Docker Compose para fazermos um ambiente semelhante ao da Aula 6 que tem um Wordpress e um MySQL.

## O que vamos construir?

Vamos fazer um arquivo do Docker Compose para criar um ambiente com o Wordpress e MySQL semelhante ao que fizemos na Aula 6.

## Instruções

Abrir sua IDE preferida, bloco de notas ou algum editor de texto no Linux como VI, VIM ou nano.

Então devemos criar um arquivo chamado docker-compose.yml

O primeiro item a ser definido é a versão do arquivo. Está linda é opcional, mas ela descreve o qual formato de arquivo nós iremos usar no Docker Compose, se deixamos em branco iremos utilizar a última versão:

```
version: "3.8"
```

Já o item Services, eles são necessários em nosso arquivo, pois ele determina os serviços que serão implementados em nosso projeto:

```
services:
```

Agora nós começamos a listar os serviços que vamos subir, ou podemos dizer, quais os containers nós iremos criar. O nomes dos serviços podem conter qualquer nome, mas é recomendável usar um nome que tenha a ver com o tipo de servidor que será criado, tipo um serviço que criará um banco de dados pode ser chamado de "db":

```
db:
```



Aqui definiremos todas as informações referente ao nosso container de banco de dados:

```
image: mysql:latest
volumes:
  - ./db/data:/var/lib/mysql
environment:
  MYSQL_ROOT_PASSWORD: my-secret-pw
  MYSQL_DATABASE: wordpressdb
  MYSQL_USER: wpuser
  MYSQL_PASSWORD: my-secret-pw
networks:
  - internal
```

#### Detalhando as informações:

- **image:** define a imagem que iremos usar, é semelhante ao FROM do Dockerfile ou ao PULL do Docker CLI, ele define qual imagem será baixada.
- **volumes:** define um volume ou seja uma pasta gerenciada da qual o container faz utilização
- **environment:** aqui definimos as variáveis de ambiente, no caso as informações necessárias para nosso banco de dados
- **networks:** aqui adicionamos as redes que iremos usar no nosso container

Aqui vamos definir as informações referentes ao container do Wordpress:

```
wordpress:
  image: wordpress:latest
  depends_on:
    - db
  volumes:
    - ./wp/wp-content:/var/www/html/wp-content
  ports:
    - "8081:80"
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_NAME: wordpressdb
    WORDPRESS_DB_USER: wpuser
    WORDPRESS_DB_PASSWORD: my-secret-pw
```

```
networks:
  - internal
  - public

networks:
  internal:
    internal: true
  public:
```

**Detalhando as informações:**

- **depends\_on:** aqui estamos dizendo que o container dependerá do container: “db” ou seja se o container em questão não subir ou der erro o container “wordpress” também não será iniciado.
- **ports:** faz a liberação e define o redirecionamento das portas que iremos utilizar no container.

Agora para iniciarmos nossos containers devemos digitar o comando abaixo via terminal no mesmo local que o arquivo do docker-compose.yml.

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

Este comando irá baixar as imagens necessárias para criar e iniciar nossos containers, agora se listarmos os containers veremos:

```
➤ docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
0a86f54ab63e   wordpress:latest "docker-entrypoint.s..." 27 minutes ago Up 27 minutes   0.0.0.0:8081→80/tcp      dockercompose_wordpress_1
9d880fcca346   mysql:latest   "docker-entrypoint.s..." 27 minutes ago Up 27 minutes                               dockercompose_db_1
```

Também podemos acessar o endereço <http://localhost:8081/>

