

**Professor: Danilo Sibov**

## **Microserviços | Aula 2**

### **Laboratórios**

**Este módulo aborda os seguintes tópicos:**

- **Laboratório 5 - Primeiro Container**
- 

Neste laboratório você aprenderá a criar um Dockerfile simples, vai aprender a **construir** sua imagem a partir deste Dockerfile, executar um container com alguns parâmetros específicos e vai aprender a ver os Logs de um container.

**Etapa 1 - Ensinar a criar um Dockerfile simples**

**Etapa 2 - Ensinar a buildar uma imagem**

**Etapa 3 - Ensinar a mostrar imagens do Docker**

**Etapa 4 - Ensinar a executar um container explicando**

**Etapa 5 - Ensinar a visualizar logs de um container**

**Etapa 6 - Acessar um Container**

## Etapa 1 - Como criar um Dockerfile simples

---

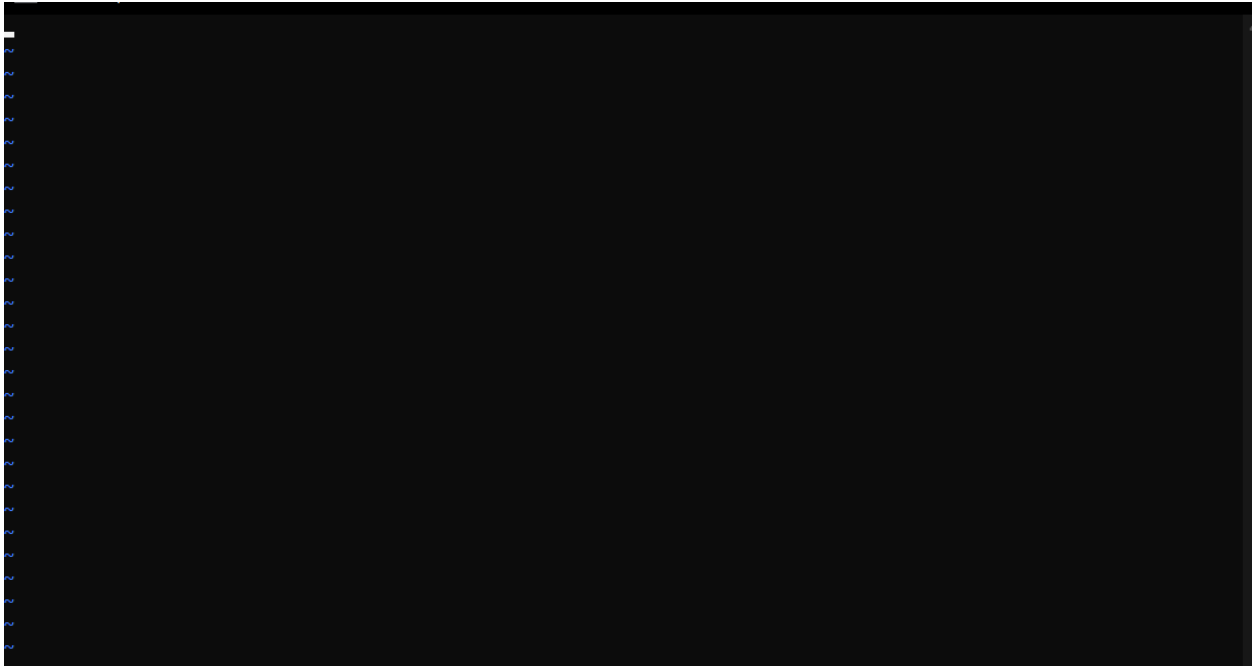
Para criarmos um Dockerfile simples, primeiro, vamos precisar de um recurso muito importante, um editor de texto, neste caso, estaremos utilizando o **vim**

### 1. Para instalar o vim digite: `sudo apt-get install vim -y`

```
ubuntu@ip-172-31-24-3:~$ sudo apt-get install vim -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  vim-common vim-runtime vim-tiny
Suggested packages:
  ctags vim-doc vim-scripts indent
The following packages will be upgraded:
  vim vim-common vim-runtime vim-tiny
4 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
Need to get 9345 kB of archives.
After this operation, 1024 B of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim amd64 2:8.2.3995-1ubuntu2.1 [1727 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-tiny amd64 2:8.2.3995-1ubuntu2.1 [704 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-runtime all 2:8.2.3995-1ubuntu2.1 [6832 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-common all 2:8.2.3995-1ubuntu2.1 [81.5 kB]
Fetched 9345 kB in 0s (36.1 MB/s)
(Reading database ... 63929 files and directories currently installed.)
Preparing to unpack .../vim_2%3a8.2.3995-1ubuntu2.1_amd64.deb ...
Unpacking vim (2:8.2.3995-1ubuntu2.1) over (2:8.2.3995-1ubuntu2) ...
Preparing to unpack .../vim-tiny_2%3a8.2.3995-1ubuntu2.1_amd64.deb ...
Unpacking vim-tiny (2:8.2.3995-1ubuntu2.1) over (2:8.2.3995-1ubuntu2) ...
Preparing to unpack .../vim-runtime_2%3a8.2.3995-1ubuntu2.1_all.deb ...
Unpacking vim-runtime (2:8.2.3995-1ubuntu2.1) over (2:8.2.3995-1ubuntu2) ...
Preparing to unpack .../vim-common_2%3a8.2.3995-1ubuntu2.1_all.deb ...
Unpacking vim-common (2:8.2.3995-1ubuntu2.1) over (2:8.2.3995-1ubuntu2) ...
```

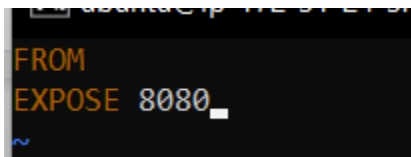
### 2. Após instalar o vim, vamos criar nosso primeiro Dockerfile,

digite: **vim Dockerfile**



**OBS:** O nome precisa ser exatamente: Dockerfile

3. Para começarmos a digitar no **vim** aperte a tecla “i” e digite a seguinte sintaxe:



4. Feito isso, neste exemplo, vamos estar utilizando a imagem oficial do Apache, conhecida como **httpd**, portanto, no campo FROM digite: **http**

```
ubuntu@ip-172-31-24-  
FROM httpd  
EXPOSE _
```

5. Depois, no campo EXPOSE, digite: **8080**

```
FROM httpd  
EXPOSE 8080 _
```

OBS: No campo EXPOSE, é onde definimos a porta que o Container irá rodar/escutar, ou seja, requisições do tipo <http://localhost:8080> iram ser redirecionadas para este container.

6. Após executar os dois últimos STEPS acima, aperte a tecla ESC e note que o insert saiu do campo inferior esquerdo

```
~  
~  
~
```

7. Para salvarmos o nosso arquivo após ter apertado ESC, aperte “:” e depois digite **wq!** E depois aperte a tecla ENTER

```
~  
~  
~:wq!
```

*W = Write*

*Q = quit*

*! = Force*

8. Feito isso, nós temos o nosso primeiro Dockerfile

## Etapas 2, 3 e 4 - Como “ buildar “ uma imagem Docker

1. Para buildar sua primeira imagem Docker, execute o seguinte comando:  
`docker build -t primeira-imagem-docker .`

```
ubuntu@ip-172-31-24-3:~$ docker build -t primeira-imagem-docker .
Sending build context to Docker daemon 14.85kB
Step 1/2 : FROM httpd
latest: Pulling from library/httpd
e9995326b091: Pull complete
ee55ccd48c8f: Pull complete
bc66ebea7efe: Pull complete
5d0f831d3c0b: Pull complete
e559e5380898: Pull complete
Digest: sha256:5fa96551b61359de5dfb7fd8c9e97e4153232eb520a8e883e2f4
Status: Downloaded newer image for httpd:latest
--> fe8735c23ec5
Step 2/2 : EXPOSE 8080
--> Running in 8c4ffaa77872
Removing intermediate container 8c4ffaa77872
--> 58630902867f
Successfully built 58630902867f
Successfully tagged primeira-imagem-docker:latest
```

2. Após você construir/buildar sua primeira imagem, digite: `docker images`

```
ubuntu@ip-172-31-24-3:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
primeira-imagem-docker  latest     58630902867f  About a minute ago  145MB
httpd                latest     fe8735c23ec5  2 weeks ago    145MB
ubuntu@ip-172-31-24-3:~$
```

## Etapa 4 - Executando um Container

1. Para executarmos um container, é necessário de uma Imagem Docker, anteriormente construímos uma, vamos utilizar ela, digite: `docker run -d -p 8080:8080 --name primeiro-container primeira-imagem-docker:latest`
2. Note que ao rodar o comando acima, um ID foi gerado automaticamente:

```
ubuntu@ip-172-31-24-3:~$ docker run -d -p 8080:8080 --name primeiro-container primeira-imagem-docker:latest
5f77de296303288e36701aca0b8fc41bb32c69106968497aa9338a51ade9986c
ubuntu@ip-172-31-24-3:~$
```

**3. Vamos ver se o container está UP, digite:** docker container ls

```
ubuntu@ip-172-31-24-3:~$ docker container ls
CONTAINER ID   IMAGE                                COMMAND                  CREATED
5f77de296303   primeira-imagem-docker:latest       "httpd-foreground"      About a minute ago
ubuntu@ip-172-31-24-3:~$
```

Se a saída do comando acima mostrar o seu container, significa que seu container está UP/Healthy

**4. Para podermos ver os LOGS de um container e identificar possíveis problemas, digite:** docker logs <nome/id do container>

```
ubuntu@ip-172-31-24-3:~$ docker logs primeiro-container
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, usi
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, usi
[Thu Nov 10 18:44:41.637694 2022] [mpm_event:notice] [pid 1:tid 139661584325952] AH00489:
[Thu Nov 10 18:44:41.637899 2022] [core:notice] [pid 1:tid 139661584325952] AH00094: Comma
ubuntu@ip-172-31-24-3:~$
```

## Etapa 6 - Como acessar um Container

1. Para acessar um container Docker é simples, digite: **docker exec -it**  
**<nome/id do container> bash**

```
ubuntu@ip-172-31-24-3:~$ docker exec -it primeiro-container bash
root@5f77de296303:/usr/local/apache2#
```

NOTE:

**Exec:** é um argumento do comando docker que pode ser utilizado para executar comandos ou entrar no container

**-it:** significa interactive, ou seja, vamos acessar o modo iterativo do container

**Bash:** em conjunto com o **-it**, utilizamos o parâmetro bash, para dizer ao comando que queremos conectar com o container Docker

2. Para sair do container, digite: **exit**

```
ubuntu@ip-172-31-24-3:~$ docker exec -it primeiro-container bash
root@5f77de296303:/usr/local/apache2# exit
exit
ubuntu@ip-172-31-24-3:~$ _
```

3. Para executar comandos remotamente em um container digite: **docker exec**  
**<nome/id do container> env**

```
ubuntu@ip-172-31-24-3:~$ docker exec primeiro-container env
PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=5f77de296303
HTTPD_PREFIX=/usr/local/apache2
HTTPD_VERSION=2.4.54
HTTPD_SHA256=eb397feefccaf254f8d45de3768d9d68e8e73851c49afd5b7176d1ecf80c340
HTTPD_PATCHES=
HOME=/root
ubuntu@ip-172-31-24-3:~$
```

Neste caso acima, não utilizamos o **-it**, apenas passamos um **comando** para o **exec** Executar dentro do container, neste caso, o comando foi **env** que lista todas as variáveis de ambiente do Linux.