

Prints e passo a passo:

(Caso os prints não fiquem claros no documento, os disponibilizarei também no repositório)

Começamos clonando o repositório git do professor que disponibilizou o curso:

```
lucas@docker: ~/Lucas/AV-07
lucas@docker:~/Lucas/AV-07$ git clone https://github.com/programadorabordo/docker-introducao
Cloning into 'docker-introducao'...
remote: Enumerating objects: 45, done.
remote: Total 45 (delta 0), reused 0 (delta 0), pack-reused 45
Unpacking objects: 100% (45/45), 456.50 KiB | 1.23 MiB/s, done.
lucas@docker:~/Lucas/AV-07$
```

Após isso, instalamos tanto o Docker como o Compose:

```
lucas@docker:~/Lucas/AV-07/docker-introducao$ sudo apt install docker.io docker-compose
[sudo] password for lucas:
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (1.25.0-1).
docker-compose is already the newest version (20.10.12-ubuntu20.04.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
lucas@docker:~/Lucas/AV-07/docker-introducao$
```

Após esse passo, deveremos criar um arquivo de extensão yml ou yaml chamado docker-compose (utilizando-se de qualquer editor de texto, eu utilizei o nano) com as configurações seguintes:

```
docker@docker: ~/Lucas/AV-07/docker-introducao
GNU nano 4.8
services:
  db:
    image: mysql
    container_name: mysql-container
    command: --default-authentication-plugin=mysql_native_password
    environment:
      MYSQL_ROOT_PASSWORD: programadorabordo
    volumes:
      - ./api/db/data:/var/lib/mysql
    restart: always
  api:
    build: './api'
    container_name: node-container
    restart: always
    volumes:
      - ./api:/home/node/app
    ports:
      - "9001:9001"
    depends_on:
      - db
  web:
    image: "php:7.3-apache"
    container_name: php-container
    volumes:
      - ./website:/var/www/html
    ports:
      - "8888:80"
    depends_on:
      - db
      - api
```

Agora, é necessário que rodemos os nossos containers com o comando mostrado abaixo:

```
docker@docker:~/Lucas/AV-07/docker-introducao$ sudo docker-compose up -d
Creating network "docker-introducao_default" with the default driver
Pulling db (mysql):...
latest: Pulling from library/mysql
824b15f81065: Pull complete
556a09130a: Pull complete
62801c1961446: Pull complete
74247a8f6125: Pull complete
0c9ef6d9c45d: Pull complete
4f3787edd16d: Pull complete
a0b4a8f875d1: Pull complete
75f6b647ddb1: Pull complete
a09ca8f6cb24: Pull complete
a223a3cd7fd: Pull complete
2b03a8d826c65: Pull complete
833ac0857fc9: Pull complete
Digest: sha256:a042470aa5f3c704b15a3700bfa39f4c009262d7753fa09de2d9faf5f83
Status: Downloaded newer image for mysql:latest
Building api
Step 1/3 : FROM node:10-slim
10-slim: Pulling from library/node
c26a0e7a20d: Pull complete
f698164f6049: Pull complete
8c2932c0b29: Pull complete
89e804c1c58f: Pull complete
6c72a4359589: Pull complete
Digest: sha256:8b923293e3022d79161b09628c4c350e836437455e2d1b1a080d90367b10d6
Status: Downloaded newer image for node:10-slim
--> 6fbcbb5c6a3
Step 2/3 : WORKDIR /home/node/app
--> Running in 069070dd0616
Removing intermediate container 069070dd0616
--> 02c37aa009f4
Step 3/3 : CMD npm start
--> Running in 08385906a5
Removing intermediate container d8838506a5
--> 03c4b0fe9d55
Successfully built 03c4b0fe9d55
Successfully tagged docker-introducao_api:latest
WARNING: Image for service api was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Pulling web (php:7.3-apache)...
7.3-apache: Pulling from library/php
ae13d6578326: Pull complete
f15d4750809f: Pull complete
38e6516103f3: Pull complete
aa7666573a25: Pull complete
39357a0f0582: Pull complete
6c34f60c774a: Pull complete
533e9383f6d4: Pull complete
69e0073fa0b0: Pull complete
265d46c25392: Pull complete
f2994b88806e: Pull complete
740eb0c106a7: Pull complete
44620150904f: Pull complete
405ec7393903: Pull complete
a1888e2a9ff0: Pull complete
Digest: sha256:b9872cd287ef72bc17d45d713aa2742f3d3bcf2503fea2506f093aa94995219f
Status: Downloaded newer image for php:7.3-apache
Creating mysql-container ... done
Creating node-container ... done
Creating php-container ... done
```

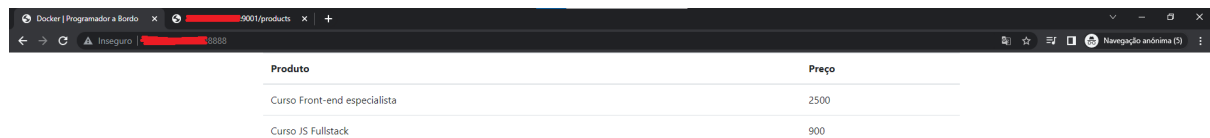
É possível que tenhamos dificuldades (como eu tive) em rodar a API configurada pelo professor. Para solucionar esse problema, modificaremos a Dockerfile da API e criamos um script.sh com os respectivos comandos:

```
docker@docker:~/Lucas/AV-07/docker-introducao/api$ ls
Dockerfile  node_modules  package.json  package-lock.json  script.sh  src
docker@docker:~/Lucas/AV-07/docker-introducao/api$ cat Dockerfile
FROM node:10-slim
WORKDIR /home/node/app
CMD ["/script.sh"]
docker@docker:~/Lucas/AV-07/docker-introducao/api$ cat script.sh
#!/bin/bash
npm install
npm start
docker@docker:~/Lucas/AV-07/docker-introducao/api$
```

Feito isso, deve-se rodar o docker-compose up -d novamente e obter acesso ao banco de dados com o seguinte comando:

```
Selecionar docker@docker: ~/Lucas/AV-07/docker-introducao
docker@docker:~/Lucas/AV-07/docker-introducao$ sudo docker exec -i mysql-container mysql -uroot -pprogramadorabordo < api/db/script.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
```

Configuramos tudo que é necessário para a execução das aplicações. Como estava acessando a máquina via SSH peguei o respectivo IP com o ifconfig e acessei as 2 aplicações com o respectivo IP, o resultado foi:



The screenshot shows a web browser window with a single tab titled 'Docker | Programador a Bordo'. The address bar shows a redacted IP address followed by '/products'. The page content displays a table with two columns: 'Produto' and 'Preço'. The table contains two rows of data.

Produto	Preço
Curso Front-end especialista	2500
Curso JS Fullstack	900



The screenshot shows a web browser window with a single tab titled 'Docker | Programador a Bordo'. The address bar shows a redacted IP address followed by '/products'. The page content displays a JSON array of objects.

```
[{"name":"Curso Front-end especialista","price":2500}, {"name":"Curso JS Fullstack","price":900}]
```

Como podemos ver, temos acesso tanto a parte cliente quanto a API que foram configuradas com o nosso Docker.

E é isso, pomos em prova a comunicação entre containers e o funcionamento dessa grande ferramenta!