

Padronização de ambientes com a ferramenta *Docker*

Docker é uma ferramenta de *open source* (código aberto) sem nenhum custo, que permite um conjunto completo do ambiente, desde configurações de máquinas, aplicações e programas necessários, permitindo ser distribuído e virtualizado de maneira fácil e rápida para outras máquinas. Para utilização da ferramenta é necessário seguir as seguintes etapas, este tutorial foi aplicado em um ambiente *Linux* (Ubuntu) para a padronização dos ambientes do CRS e DTI Unochapecó:

1 – Atualização pacotes;

Quadro 1: Comando para atualizar pacotes

```
sudo apt-get update
```

Fonte: Autor.

2 – Instalação do *Docker*;

Quadro 2: Comando para instalação *Docker*

```
sudo apt-get install docker.io
```

Fonte: Autor.

3 – Configuração *docker* para inicialização;

Quadro 3: Comando configurar *Docker* para inicialização

```
sudo systemctl enable docker
```

Fonte: Autor.

4 – Reiniciar os serviços *NetworkManager* e *Docker*;

Quadro 4: Comando para reiniciar serviços

```
sudo restart network-manager  
sudo restart docker
```

Fonte: Autor.

5 – Verificar se o *Docker* está rodando;

Quadro 5: Comando para verificar se *Docker* está rodando

```
sudo docker run hello-world
```

Fonte: Autor.

6 – Depois de tudo instalado, o que falta é o *container* onde contem a máquina com as configurações e programas necessários para o ambientes de desenvolvimento. O *Docker* mantém um local pra armazenar repositórios¹. E para este caso foi criado um com o nome de crs-uno-server² e também no github³.

7 – Para baixar o repositório crs-uno-server;

Quadro 6: Comando para baixar repositório crs-uno-server

```
docker pull alefevariani/crs-uno-server
```

Fonte: Autor.

8 – Listando os *container*;

Quadro 7: Comando lista *container Docker*

```
docker ps
```

Fonte: Autor.

9 – Listar *images Docker* existentes na máquina;

Quadro 8: Comando lista *images Docker*

```
docker images
```

Fonte: Autor.

10 – Rodando e acessando o *shell* do *container*, informando o ID *image*;

```
docker run -it ID IMAGE bash
```

Quadro 9: Comando para rodar e acessar *shell container* crs-uno-server

```
docker run -it CONTEINER ID IMAGE bash
```

Fonte: Autor.

1 <https://hub.docker.com>

2 <https://hub.docker.com/r/alefevariani/crs-uno-server/>

3 <https://github.com/AlefeVariani/crs-uno-server>

Criando um repositório no *Docker*.

11 – Para a criação de um repositório é necessário criar uma conta no *site*⁴

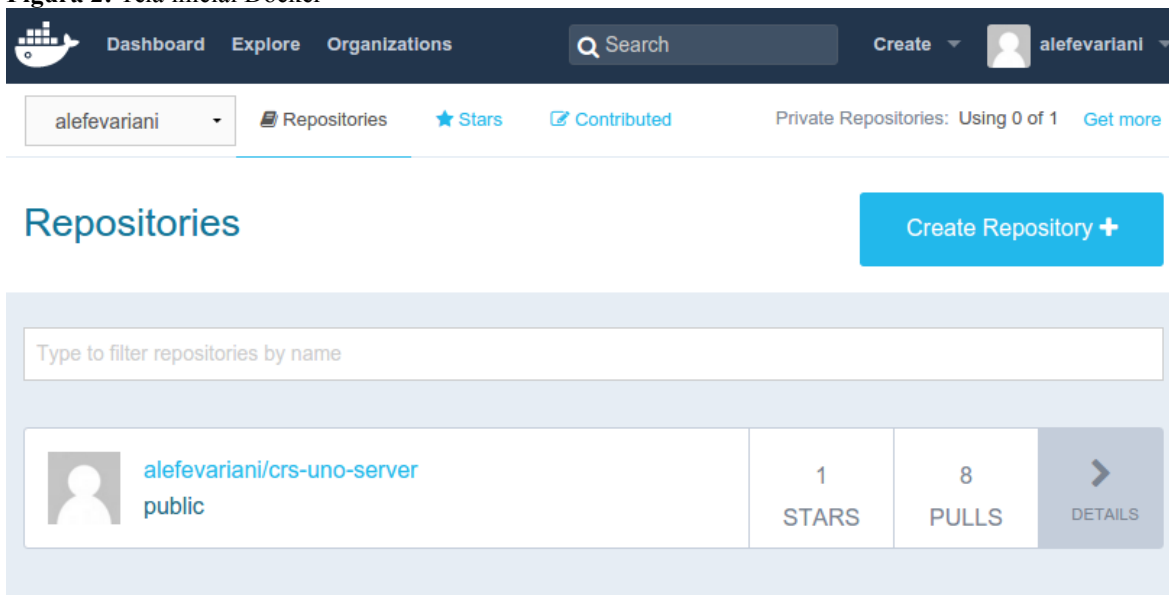
Figura 1: Tela login cadastro Docker



Fonte: Autor.

12 – Após o login efetuado, será apresentado uma tela semelhante a imagem a seguir, onde terá a opção *Create Repository* (criar repositório);

Figura 2: Tela inicial Docker

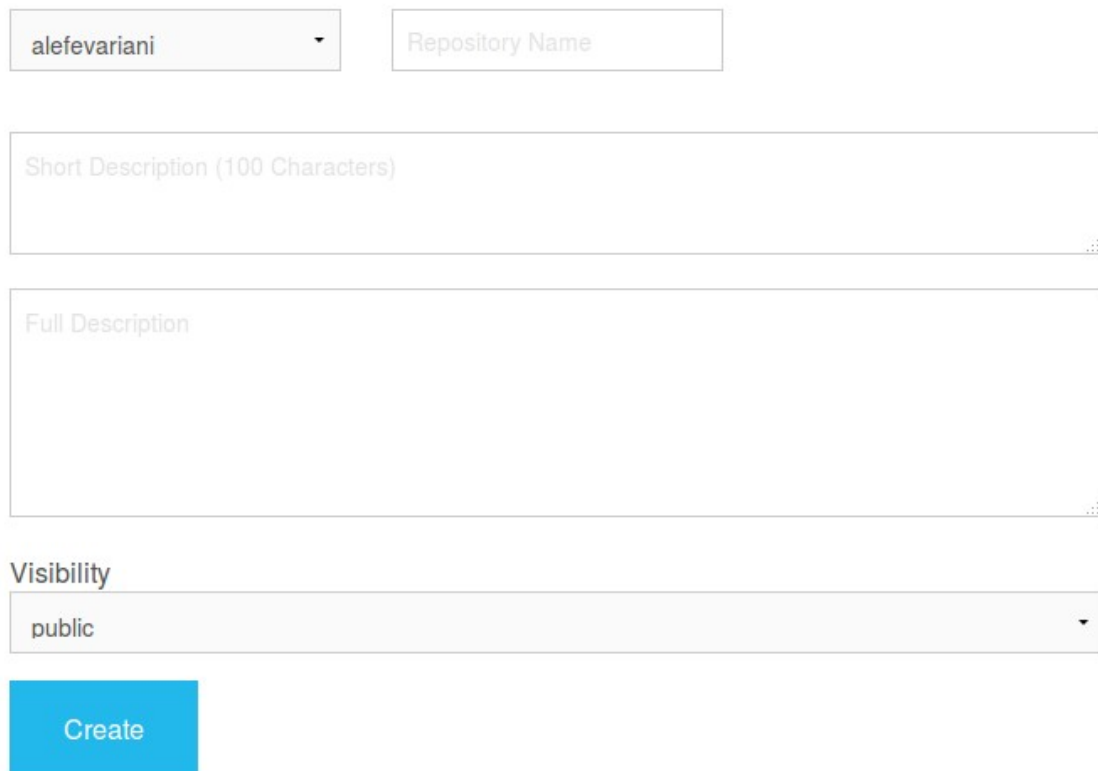


Fonte: Autor.

⁴ <https://hub.docker.com/>

13 – Seleccionada a opção *Create Repository* (criar repositório), a tela será esta a seguir, que possui o nome, descrições e a visibilidade que pode ser privada (*private*) ou pública (*public*);

Figura 3: Tela de criação repositório Docker

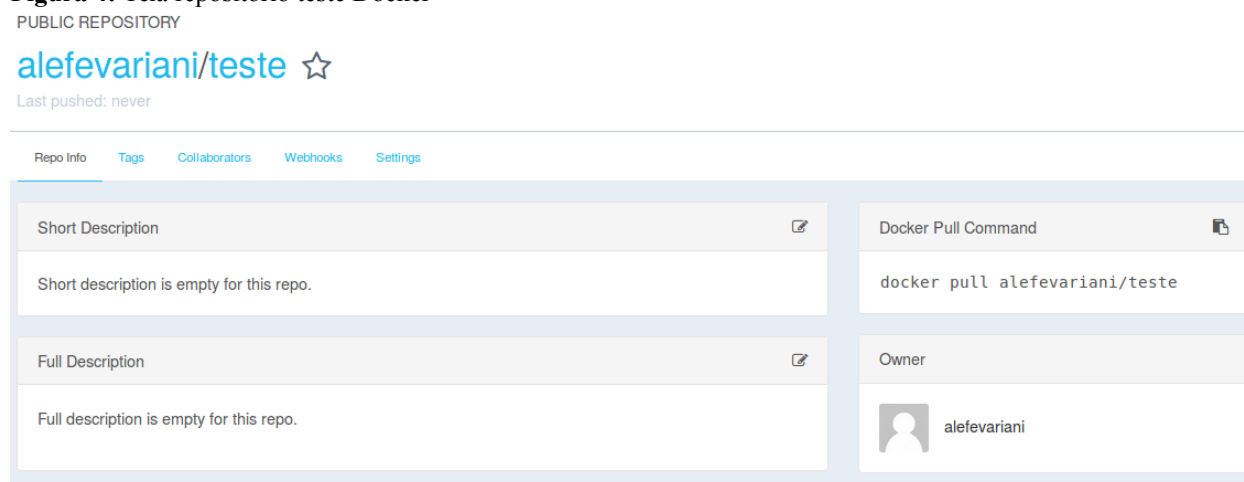


The screenshot shows the 'Create Repository' form on Docker Hub. At the top, there is a dropdown menu with 'alefevariani' selected and a text input field for 'Repository Name'. Below these are two large text areas: 'Short Description (100 Characters)' and 'Full Description'. Underneath the descriptions is a 'Visibility' dropdown menu set to 'public'. At the bottom left is a blue 'Create' button.

Fonte: Autor.

14 – Criado o repositório, neste caso a tela será semelhante a está;

Figura 4: Tela repositório teste Docker



The screenshot shows the Docker Hub page for a newly created public repository named 'alefevariani/teste'. The page header includes 'PUBLIC REPOSITORY' and a star icon. Below the repository name, it says 'Last pushed: never'. A navigation bar contains links for 'Repo Info', 'Tags', 'Collaborators', 'Webhooks', and 'Settings'. The main content area is divided into two columns. The left column has sections for 'Short Description' and 'Full Description', both with placeholder text indicating they are empty. The right column has a 'Docker Pull Command' section showing the command 'docker pull alefevariani/teste' and an 'Owner' section showing the user 'alefevariani' with a profile picture icon.

Fonte: Autor.

Nesta imagem 53 já é apresentado o comando “docker pull alefevariani/teste” para baixar o *container image* criado, mas como esta vazio é preciso criar a máquina com os itens a seguir;

15 – Via terminal *linux*, criar uma pasta, com o arquivo *Dockerfile*;

Quadro 10: Comando criar pasta e arquivo Dockerfile

```
mkdir teste  
cd teste/  
nano Dockerfile
```

Fonte: Autor.

Adicionar conteúdo no *Dockerfile*, neste arquivo ficam as configurações da máquina, mais informações para criação do arquivo *Dockerfile* no link⁵ e também para os repositórios oficiais oferecidos, segue link⁶;

Quadro 11: Exemplo arquivo Dockerfile

```
# INSTALAR SO UBUNTU, ÚLTIMA VERSÃO  
FROM ubuntu:latest  
  
# INSTALAR APACHE 2 E PHP  
RUN apt-get -y install apache2 php5  
  
#LIBERAÇÃO DE PORTAS 80 E 443  
EXPOSE 80 443
```

Fonte: Autor.

16 – Para executar o arquivo Dockerfile e verificara se os comando estão todos corretos:

Quadro 12: Executar arquivo Dockerfile

```
docker build -t alefevariani/teste .
```

Fonte: Autor.

No local de alefevariani/teste, deve-se passar nomeSeuUsuario/nomeRepositorio.

17 – Para manter o *container* para seu repositório no Hub Docker, são os seguintes passos, semelhante a um controle de versão;

Listar último *container*.

Quadro 13: Comando listar último container

```
docker ps -l
```

Fonte: Autor.

Fazer *commit* da imagem criada, informando o CONTAINER ID.

⁵ <https://docs.docker.com/reference/builder/>

⁶ <https://hub.docker.com/explore/>

Quadro 14: Comando fazer commit container Docker

```
docker commit CONTAINER ID alefevariani/teste
```

Fonte: Autor.

Enviar a imagem da máquina para o repositório, permitindo depois fazer *pull* por outras máquinas.

Quadro 15: Comando enviar imagem Docker

```
docker push alefevariani/teste
```

Fonte: Autor.

Feito isso, pode-se utilizar este *container* para várias máquinas, no caso, para o servidor de desenvolvimento e produção, evitando diferenças de versões em sistemas operacional, linguagem de programação, banco de dados e pacotes extras.

Autor: Alefe Variani

E-mail: alefevariani18@gmail.com

Informações: about.me/alefe_variani