

Professor: Danilo Sibov

Microserviços | Aula 2

Laboratórios

Este módulo aborda os seguintes tópicos:

- **Laboratório 4 - Criando instância Venom**

Será necessário:

- E-mail para acesso a conta DockerHub
- Ambiente Hospedeiro para o Docker já preparado

Referência Bibliográfica

- Site oficial Documentação Docker - <https://www.docker.com/>
- Site oficial DockerHub - <https://hub.docker.com/>
- Instalação Docker no Ubuntu: <https://docs.docker.com/engine/install/ubuntu/>

Laboratório 4 - Criando instância Venom

Neste laboratório você aprenderá a usar o Docker, configurá-lo para utilizar a CLI, além de procurar imagens oficiais no Docker Hub, baixar imagens e aprender sobre as boas práticas.

Usaremos essa conta para a construção do nosso repositório de imagens Doc

Etapa 1 - Baixar o Docker

Etapa 2 - Configurar usuário comum para acessar os comandos do Docker

Etapa 3 - Baixar imagem do DockerHub

Etapa 4 - Ensinar a mostrar imagens baixadas (docker images)

Etapa 5 - Boas práticas para criação de imagens Docker

Etapa 1 - Baixar o Docker

01. Para baixar o Docker, acesse sua máquina virtual Ubuntu, através do terminal e digite o seguinte **comando**: “**sudo su**” para acesso como super usuário,

```
root@ip-172-31-24-3: /home/ubuntu
ubuntu@ip-172-31-24-3:~$ sudo su
root@ip-172-31-24-3: /home/ubuntu#
```

02. Com acesso “root”, iniciaremos atualizando o sistema

```
root@ip-172-31-24-3: /home/ubuntu#
root@ip-172-31-24-3: /home/ubuntu# apt-get update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [698 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [159 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [10.8 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [417 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [63.9 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [580 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [747 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [123 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [4404 B]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [7220 B]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [2360 B]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [420 B]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [3008 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [1432 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [272 B]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [6744 B]
```

comando: **apt-get update -y**

03. Após atualizar o sistema, instale os seguintes pacotes:

- **ca-certificates**
- **curl**
- **gnupg**
- **lsb-release**

```
root@ip-172-31-24-3:/home/ubuntu# apt-get install ca-certificates curl gnupg lsb-release -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20211016).
ca-certificates set to manually installed.
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following additional packages will be installed:
  libcurl4
The following packages will be upgraded:
  curl libcurl4
2 upgraded, 0 newly installed, 0 to remove and 77 not upgraded.
Need to get 484 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcurl4 amd64 7.81
Fetched 484 kB in 0s (17.3 MB/s)
(Reading database ... 63663 files and directories currently installed.)
Preparing to unpack .../curl_7.81.0-1ubuntu1.6_amd64.deb ...
```

comando: `apt-get install ca-certificates curl gnupg lsb-release -y`

04. Adicione a chave **GPG** oficial

Criar o diretório “**keyrings**” dentro de /etc/apt

```
root@ip-172-31-24-3:/home/ubuntu# sudo mkdir -p /etc/apt/keyrings
root@ip-172-31-24-3:/home/ubuntu#
```

comando: `sudo mkdir -p /etc/apt/keyring`

Após criar o diretório acima, vamos adicionar chave GPG

05. adicionar a chave GPG: curl -fsSL

```
root@ip-172-31-24-3:/home/ubuntu#  
root@ip-172-31-24-3:/home/ubuntu# sudo mkdir -p /etc/apt/keyrings  
root@ip-172-31-24-3:/home/ubuntu# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc  
root@ip-172-31-24-3:/home/ubuntu#
```

comando:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/etc/apt/keyrings/docker.gpg
```

06. Configurar repositório apt, comando:

```
echo \ "deb [arch=$(dpkg --print-architecture)  
signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee  
/etc/apt/sources.list.d/docker.list > /dev/null
```

07. Atualize o sistema novamente:

```
root@ip-172-31-24-3:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:5 https://download.docker.com/linux/ubuntu jammy InRelease [48.9 kB]
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [9481 B]
Fetched 58.3 kB in 0s (147 kB/s)
Reading package lists... Done
root@ip-172-31-24-3:/home/ubuntu#
```

comando: `apt-get update -y`

08. Instalar todos os pacotes do Docker, comando:

`apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin -y`

09. Inicializar o Docker:

`systemctl enable docker && systemctl restart docker`

10. Verificar estado do serviço:

```
root@ip-172-31-24-3:/home/ubuntu# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-11-10 17:43:24 UTC; 17min ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Main PID: 2497 (dockerd)
      Tasks: 7
     Memory: 29.2M
        CPU: 343ms
     CGroup: /system.slice/docker.service
            └─2497 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.473511360Z" level=info msg="scheme \"unix\" not registered, fallback to default scheme"
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.473700261Z" level=info msg="ccResolverWrapper: sending update to ccResolverImpl"
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.473863590Z" level=info msg="ClientConn switching balancer to \"pickfirst\""
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.531290395Z" level=info msg="Loading containers: start."
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.743600664Z" level=info msg="Default bridge (docker0) is assigned to docker network"
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.844085451Z" level=info msg="Loading containers: done."
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.919991368Z" level=info msg="Docker daemon" commit=3056208 graphdriver=overlay2
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.920327342Z" level=info msg="Daemon has completed initialization"
Nov 10 17:43:24 ip-172-31-24-3 systemd[1]: Started Docker Application Container Engine.
Nov 10 17:43:24 ip-172-31-24-3 dockerd[2497]: time="2022-11-10T17:43:24.969497825Z" level=info msg="API listen on /run/docker.sock"

lines 1-22/22 (END)
```

comando: **systemctl status docke**

Saída esperada: **Active (running)**, instalação foi bem sucedida.

OBS: Ao executar a etapa 07, caso apareça algum erro, execute os dois comandos abaixo e execute a etapa 07 novamente:

sudo chmod a+r /etc/apt/keyrings/docker.gpg

sudo apt-get update

Etapa 2 - Configurar usuário comum para acessar os comandos do Docker

01. Como usuário **root**, verifique a instalação do Docker

```
root@ip-172-31-24-3:/home/ubuntu# docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers
```

comando: **docker**

Retorno esperado na saída do comando, a instalação está correta.

Retorno parte 1:

```
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

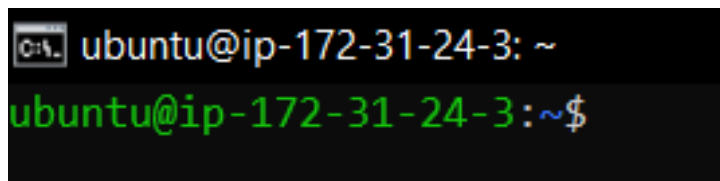
Options:
  --config string      Location of client config files (default "/root/.docker")
  -c, --context string  Name of the context to use to connect to the daemon (overrides
                        DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default "/root/.docker/ca.pem")
  --tlscert string       Path to TLS certificate file (default "/root/.docker/cert.pem")
  --tlskey string        Path to TLS key file (default "/root/.docker/key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit
```


Retorno parte 2:

```
Management Commands:
app*      Docker App (Docker Inc., v0.9.1-beta3)
builder   Manage builds
buildx*    Docker Buildx (Docker Inc., v0.9.1-docker)
compose*   Docker Compose (Docker Inc., v2.12.2)
config     Manage Docker configs
container  Manage containers
context    Manage contexts
image      Manage images
manifest   Manage Docker image manifests and manifest lists
network    Manage networks
node       Manage Swarm nodes
plugin     Manage plugins
scan*      Docker Scan (Docker Inc., v0.21.0)
secret     Manage Docker secrets
service    Manage services
stack      Manage Docker stacks
swarm      Manage Swarm
system     Manage Docker
trust      Manage trust on Docker images
volume     Manage volumes
```

02. Configurar o usuário:

- Digite no terminal: **exit**



```
C:\> ubuntu@ip-172-31-24-3: ~
ubuntu@ip-172-31-24-3: ~$
```

- Copiar o nome do usuário “lado esquerdo, antes do @” neste caso é **ubuntu**

- c. Após copiar o nome, digite novamente: **sudo su**

```
root@ip-172-31-24-3: /home/ubuntu  
ubuntu@ip-172-31-24-3:~$ sudo su  
root@ip-172-31-24-3:/home/ubuntu#
```

- d. Com usuário **root**, execute o **comando**: **usermod -a -G docker ubuntu**
“Substitua <usuário-copiado> para o nome do seu **usuário**”.

```
root@ip-172-31-24-3:/home/ubuntu# usermod -a -G docker ubuntu  
root@ip-172-31-24-3:/home/ubuntu#
```

Feito isso, feche o terminal e abra novamente para as modificações terem efeito.

Comando: **ctrl + D**

Após reiniciar o terminal, para verificar se o usuário comum tem acesso aos comandos do docker, digite o **comando**: **docker images ls**

```
ubuntu@ip-172-31-24-3:~$ docker images ls  
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE  
ubuntu@ip-172-31-24-3:~$
```

O retorno esperado do comando está na foto acima, a configuração de usuário está correta.

Etapa 3 - Baixar imagem do DockerHub

Neste tutorial, você vai precisar do conhecimento do LAB 3, caso não tenha feito esse laboratório, volte antes de seguir.

Vamos usar a imagem com Apache2: **httpd**

01. Para baixar essa imagem **httpd** em nosso servidor hospedeiro execute o **comando: docker pull httpd**

```
ubuntu@ip-172-31-24-3:~$ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
e9995326b091: Extracting [=====> ] 27.85MB/31.42MB
ee55ccd48c8f: Download complete
bc66e7bea7efe: Download complete
5d0f831d3c0b: Download complete
e559e5380898: Download complete
```

Note que o próprio docker irá começar o download da imagem.

02. Para localizar e listar todas as nossas imagens baixadas no Docker, execute o **comando: docker images**

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

No commando acima, nós podemos ver todas as imagens baixadas, agora, entenda um pouco mais sobre os campos da saída desse comando:

CAMPOS:

- **REPOSITORY:** repositório da imagem
- **TAG:** tag da imagem, pode ser latest (última versão) ou v1, slim, v2.
- **IMAGE ID:** ID gerado aleatoriamente pelo Docker para identificar a imagem
- **CREATED:** Quando a imagem foi criada (muitas vezes não por você)

- **SIZE:** Um dos campos mais importantes, mostra o tamanho da imagem, pode ser de MB até GB.

7. Agora que baixamos nossa imagem, vamos aprender como exclui-la, pois imagens armazenadas localmente utilizam espaço em Disco e em breve, nós aprenderemos como armazenar essas imagens na nuvem. Para deletar as imagens, execute: `docker rmi <nome-do-repository ou ID-da-imagem>`

```
ubuntu@ip-172-31-24-3:~$ docker rmi fe8735c23ec5
Untagged: httpd:latest
Untagged: httpd@sha256:5fa96551b61359de5dfb7fd8c9e97e4153232eb520a8e883e2f47fc80dbfc33e
Deleted: sha256:fe8735c23ec5da867dea9f7e69a1d120c12329f32b8a45710ff1a873a1e456ad
Deleted: sha256:7a14b4a55c1c0b4f536267e2c713b8ad391219f3069bbea1720fac9e5ae80677
Deleted: sha256:98f1e8c3ff8870a7cc398c170415c5a5ac4b52acb309b34885c538d3f5df1540
Deleted: sha256:695c5338fe1686b56319c4a4638ef885ee27e131e18e7e9a82dc18c3fccb2452
Deleted: sha256:3b5052f6e76817d8dd11ecd1a29cd0cc9f80a7598c3079643518bce1a5c62dd
Deleted: sha256:a12586ed027fafddcdcc63b31671f406c25e43342479fc92a330e7e30d65f2e
ubuntu@ip-172-31-24-3:~$
```

8. Verifique se existe mais alguma imagem baixada na sua VM, digite: `docker`

```
ubuntu@ip-172-31-24-3:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu@ip-172-31-24-3:~$
```

images

6 - Boas práticas para criação de imagens Docker

1 - Utilizar imagens oficiais e recursos específicos

Ao construir um arquivo de Dockerfile, sempre utilize imagens oficiais, ou seja, se sua aplicação é feita em NodeJS, utilize uma imagem oficial do NodeJS e assim por diante.

Isso reduz o nível de vulnerabilidade em imagens de terceiros, aumentando a segurança.

2 - Sempre otimize sua imagem ou utilize imagens bases otimizadas

Um conceito muito importante quando falamos de container é o seu “tamanho/peso”, se uma imagem Docker for muito pesada, fará com que o container demore para ficar **healthy (saúdavel)** e/ou, se estiver utilizando Kubernetes, fará com que demore para baixar a imagem.

Sempre construa suas imagens Docker o mais otimizadas possível. Você pode fazer isso de diversas maneiras, um exemplo é utilizar imagens oficiais com a tag **slim**, são imagens mais leves e com menos recursos como por exemplo **curl**, além disso, você pode reduzir o número de instruções e complexidade na hora de criar seu Dockerfile.

3 - Usufua do multi-stage building

Quando você quer otimizar ainda mais sua imagem, você pode utilizar um conceito em Docker chamado **multi-stage building**, que consiste em:

- Na primeira série de instruções da sua imagem, ele executará os comandos necessários para gerar um binário executável da sua aplicação, esses comandos podem ser instalar pacotes, requisitos mínimos ou comandos obrigatórios antes da aplicação ficar pronta.
- Na segunda série de instruções, basicamente, sua aplicação será apenas executada.

Isso reduz o nível de complexidade da aplicação, e também é a melhor maneira de otimizar sua aplicação, pois basicamente, na primeira etapa, baixará centenas de arquivos, por exemplo, executar diversos comandos que irão gerar **cache** e isso fará com que a imagem fique maior do que o necessário. Na segunda etapa, você elimina esse cache e pega somente a aplicação que foi construída, um exemplo, o binário executável.

Veja abaixo um exemplo prático desse conceito sendo aplicado:

```
FROM golang:1.16 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go ./
RUN CGO_ENABLED=0 go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app ./
CMD ["/app"]
```

4 - Menos é mais

Quando falamos de segurança, esse conceito que eu chamo de **menos é mais** tem que ser levado a sério, mas o que seria esse conceito?

Simplesmente menos é mais, ou seja, quando formos construir uma imagem Docker para nosso APP que precisa somente de um recurso para funcionar, por exemplo o **curl**, nós podemos procurar imagens oficiais e utilizá-las certo? Sim, porém, se utilizarmos imagens oficiais dessa forma, além do **curl** poderá vir diversos outros recursos e pacotes desnecessários o que pode comprometer a segurança do container.

Uma forma de resolver esse tipo de problema é criar por si só imagens que contenham os pacotes e recursos necessários, ou, procurar imagens específicas oficiais que contenha o mínimo de recursos possíveis.