



SÃO  
PAULO  
TECH  
SCHOOL

**Sistemas Operacionais**

**LAB [Desafio]**

**Docker**

**Container + Web-Data-Viz**

**Monitor Matheus Matos**

`matheus.matos@sptech.school`

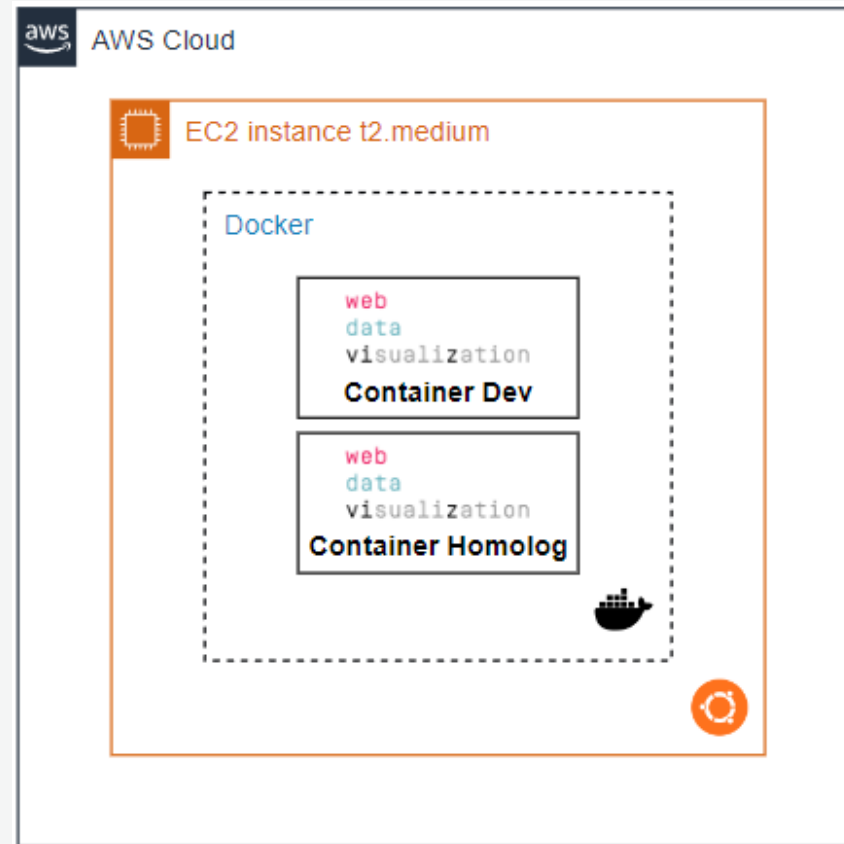
# Tópicos da Aula

1. Criar script para deploy da aplicação web-data-viz em containers
2. Executar script e fazer deploy das aplicações

# Objetivo

Utilizando uma EC2 pela conexão SSH, vamos instalar e configurar o software Docker, com o objetivo de criar dois containers para hospedagem da aplicação web-data-viz, em dois ambientes:

**desenvolvimento** e **homologação**.





AWS Cloud



EC2 instance t2.medium

Docker

web  
data  
visualization  
**Container Dev**

web  
data  
visualization  
**Container Homolog**



**Ambiente de Dev**  
**Porta TCP/IP:8080**

**Ambiente de Homolog**  
**Porta TCP/IP:8081**



**1. Criar um arquivo com a extensão  
shell script e acessar com o editor  
de texto.**

# Preparando a EC2



1. Acesse o terminal da EC2, pode ser via protocolo SSH.
2. Crie um arquivo com o comando touch:

`sudo touch deploy-container.sh`

```
ubuntu@ip-172-31-90-234:~$ sudo touch deploy-container.sh
ubuntu@ip-172-31-90-234:~$ ls
deploy-container.sh
```

3. Acesse o arquivo criado via editor de texto, podemos usar o nano:

`sudo nano deploy-container.sh`

```
GNU nano 6.2                                     deploy-container.sh
|
```

- **touch** é um comando que é usado para criar um arquivo vazio.
- **nano** é um editor de texto de linha de comando

## **2. Adicionando os comandos e executando o script.**



# Criando o Script

4. Cole os comandos a seguir no editor de texto, onde iremos executar o script (**parte 01**):

```
#!/bin/bash
```

```
# Exibe uma mensagem indicando que o assistente está verificando a instalação do Docker
```

```
echo -e "$(tput setaf 10)[ Assistente ]:$(tput setaf 7) Verificando se o docker está instalado...\n"
```

```
# Verifica se o Docker está instalado
```

```
if ! command -v docker &> /dev/null; then
```

```
    # Se o Docker não estiver instalado, inicia o processo de instalação
```

```
    echo -e "$(tput setaf 10)[ Assistente ]:$(tput setaf 7) Docker não está instalado, iniciando instalação...\n"
```

```
    sudo apt update
```

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

```
    sudo systemctl start docker
```

```
    sudo systemctl enable docker
```

```
    echo -e "$(tput setaf 10)[ Assistente ]:$(tput setaf 7) Docker Instalado!\n"
```

```
fi
```

# Criando o Script



4. Cole os comandos a seguir no editor de texto, onde iremos executar o script (**parte 02**):

# Inicia o processo de deploy das aplicações em containers Docker

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Iniciando o processo de deploy das aplicações...\n"
```

# Inicia um container Docker de desenvolvimento

```
sudo docker run -d --name app-dev -p 8080:3333 matheusferreiramattos/web-data-viz:dev
```

# Inicia um container Docker de homologação

```
sudo docker run -d --name app-homolog -p 8081:3333 matheusferreiramattos/web-data-viz:homolog
```

# Exibe mensagens de sucesso após a inicialização dos containers

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Containers instanciados com sucesso!!!\n"
```

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Acesse o ambiente de Desenvolvimento http://$(wget -qO- http://ipecho.net/plain):8080\n"
```

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Acesse o ambiente de Homologação http://$(wget -qO- http://ipecho.net/plain):8081\n"
```

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Finalizando assistente... :) MM\n"
```

# Criando o Script

## 5. Salve o arquivo (Ctrl+O):

```
#!/bin/bash

# Exibe uma mensagem indicando que o assistente está verificando a instalação do Docker
echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Verificando se o docker está instalado...\n"

# Verifica se o Docker está instalado
if ! command -v docker &> /dev/null; then
    # Se o Docker não estiver instalado, inicia o processo de instalação
    echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Docker não está instalado, iniciando instalação...\n"
    sudo apt update
    sudo apt install -y docker.io
    sudo systemctl start docker
    sudo systemctl enable docker
    echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Docker Instalado!\n"
fi

# Inicia o processo de deploy das aplicações em containers Docker
echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Iniciando o processo de deploy das aplicações...\n"

# Inicia um container Docker de desenvolvimento
sudo docker run -d --name app-dev -p 8080:3333 matheusferreiramattos/web-data-viz:dev

# Inicia um container Docker de homologação
sudo docker run -d --name app-homolog -p 8081:3333 matheusferreiramattos/web-data-viz:homolog

# Exibe mensagens de sucesso após a inicialização dos containers
echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Containers instanciados com sucesso!!!\n"
echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Acesse via navegador o ambiente de Desenvolvimento http://$(wget -qO- http://ipecho.net/plain):8080\n"
echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Acesse via navegador o ambiente de Homologação http://$(wget -qO- http://ipecho.net/plain):8081\n"
echo -e "${tput setaf 10}[Assistente]:$(tput setaf 7) Finalizando assistente... :) MM\n"
```

## 6. Feche o arquivo (Ctrl+X)

# IMPORTANTE

1. A indentação no script é essencial para manter a clareza, organização e legibilidade do código.

```
#!/bin/bash
```

```
# Comentário explicando a finalidade do script
```

```
if [condição]; then
```

```
    # Bloco de código indentado
```

```
    comando1
```

```
    comando2
```

```
else
```

```
    # Outro bloco de código indentado
```

```
    comando3
```

```
fi
```

```
#!/bin/bash
```

```
if [condição]; then
```

```
comando1
```

```
comando2
```

```
else
```

```
comando3
```

```
fi
```

\*Exemplo: Python

# Entendendo o Script



7. As primeiras instruções, apontamos o tipo de shell, no caso o bash e imprimimos no console uma mensagem:

```
#!/bin/bash

echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Verificando se o docker está instalado...\n"
```

8. No segundo bloco de instruções, fazemos uma simples validação:

```
if ! command -v docker &> /dev/null; then
    echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Docker não está instalado, iniciando instalação...\n"
    sudo apt update
    sudo apt install -y docker.io
    sudo systemctl start docker
    sudo systemctl enable docker
    echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Docker Instalado!\n"
fi
```

9. No terceiro bloco de instruções, subimos os contêineres das aplicações:

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Iniciando o processo de deploy das aplicações...\n"

sudo docker run -d --name app-dev -p 8080:3333 matheusferreiramattos/web-data-viz:dev

sudo docker run -d --name app-homolog -p 8081:3333 matheusferreiramattos/web-data-viz:homolog
```

# Entendendo o Script



10. No quarto e último bloco de instruções, imprimimos mensagens de instruções finais:

```
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Containers instanciados com sucesso!!!\n"  
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Acesse via navegador o ambiente de Desenvolvimento http://$(wget -qO- http://ipecho.net/plain):8080\n"  
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Acesse via navegador o ambiente de Homologação http://$(wget -qO- http://ipecho.net/plain):8081\n"  
echo -e "$(tput setaf 10)[Assistente]:$(tput setaf 7) Finalizando assistente... :) MM\n"
```

- **\$( )** é uma construção em shell que captura a saída de um comando e a utiliza em outro contexto.
- **tput setaf** é um comando que define a cor do primeiro plano do texto no terminal.

- **-e** é uma opção do comando echo que permite interpretar sequências de escape no texto, como `\n` para nova linha, `\t` para tabulação, etc.

# Executando o Script



11. Dê permissão de execução para o script:

`sudo chmod +x deploy-container.sh`

```
ubuntu@ip-172-31-90-234:~$ sudo chmod +x deploy-container.sh
ubuntu@ip-172-31-90-234:~$ ls -hl deploy-container.sh
-rwxr-xr-x 1 root root 1.3K Nov  7 20:44 deploy-container.sh
```

12. Execute o script:

`./deploy-container.sh`

```
ubuntu@ip-172-31-90-234:~$ ./deploy-container.sh
[Assistente]: Verificando se o docker está instalado...

[Assistente]: Docker não está instalado, iniciando instalação...

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 [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 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/main amd64 Packages [644 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [514 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [514 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [514 kB]
Fetched 1,840 kB in 10s (184 kB/s)
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  docker.io
The following NEW packages will be installed:
  docker.io
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 19.8 MB of archives.
After this operation, 79.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```



# Conferindo os Containers

13. Verifique se o script realizou o download das imagens:

`sudo docker images`

```
ubuntu@ip-172-31-90-234:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
matheusferreiramattos/web-data-viz	homolog	5eeec78465de	23 hours ago	1.06GB
matheusferreiramattos/web-data-viz	dev	ef219d225027	23 hours ago	1.06GB

14. Verifique se o script realizou a criação dos containers:

`sudo docker ps`

```
ubuntu@ip-172-31-90-234:~$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
cfb801a070d5	matheusferreiramattos/web-data-viz:homolog	"docker-entrypoint.s..."	27 seconds ago	Up 26 seconds	0.0.0.0:8081->3333/tcp, :::8081->3333/tcp	app-homolog
2e3b822dfeda	matheusferreiramattos/web-data-viz:dev	"docker-entrypoint.s..."	36 seconds ago	Up 32 seconds	0.0.0.0:8080->3333/tcp, :::8080->3333/tcp	app-dev


**\*IMPORTANTE:** Para acessar a aplicação via navegador, é importante adicionar as regras de entradas das portas 8080 e 8081 no SG.




# Atenção!

- Importante lembrar de adicionar as regras de entradas no grupo de segurança na EC2, das portas que estamos utilizando.

**Regras de entrada (2)**

 [Gerenciar tags](#) [Editar regras de entrada](#)

< 1 > 

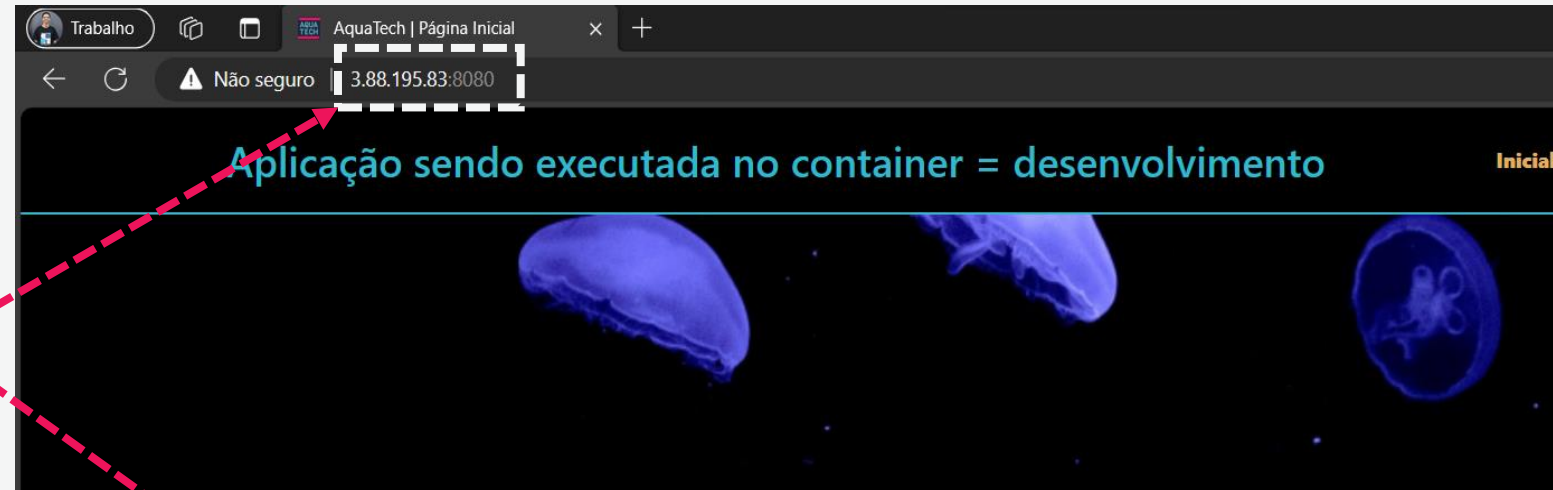
Name ▾	ID da regra do grup... ▾	Versão do IP ▾	Tipo ▾	Protocolo ▾	Intervalo de portas ▾
acesso-ssh	sgr-03e157efa13ccadd2	IPv4	SSH	TCP	22
acesso-web	sgr-0baa1e373d144d2...	IPv4	TCP personalizado	TCP	8080 - 8081

# Resultados

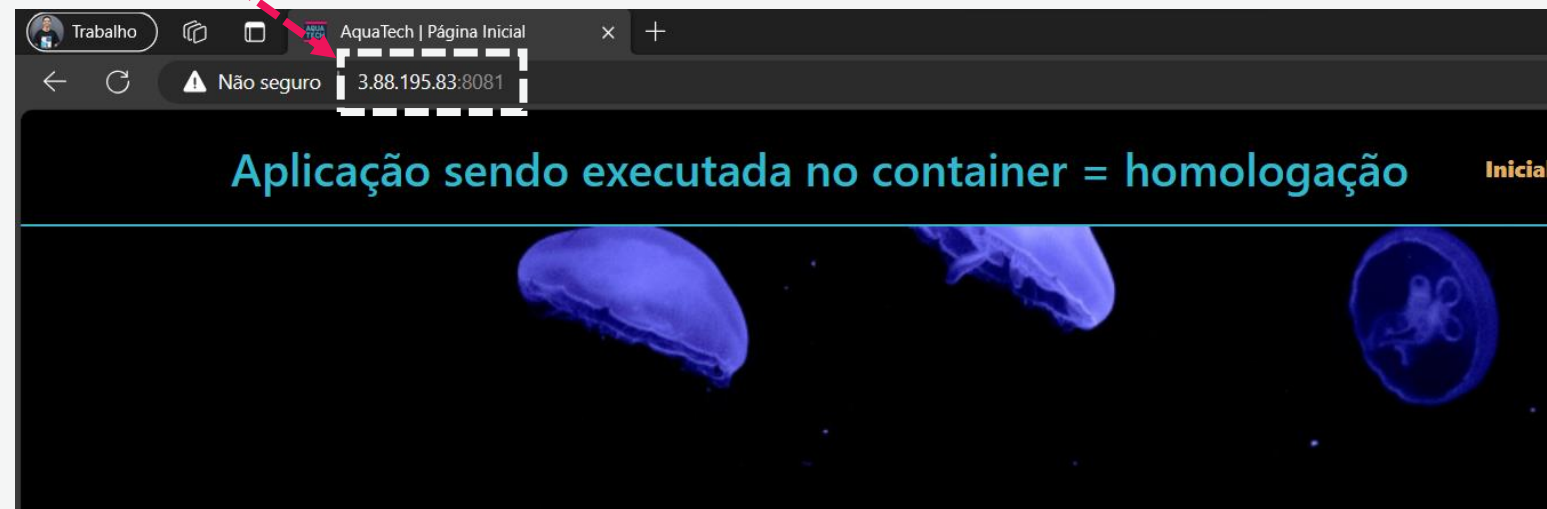
- IP
- PORTA



\*Observe, o IP público é o mesmo, porém a porta referenciada é diferente!!!



Dev



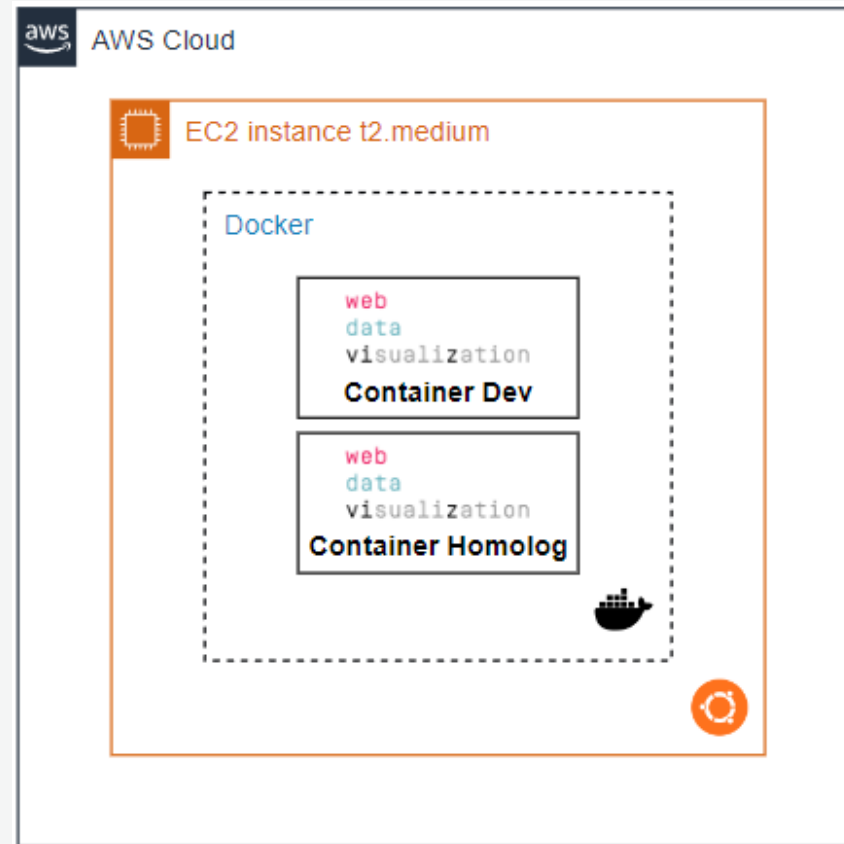
Homolog

**Acesse:**  
**IP:Porta**

# Objetivo

Utilizando uma EC2 pela conexão SSH, vamos instalar e configurar o software Docker, com o objetivo de criar dois containers para hospedagem da aplicação web-data-viz, em dois ambientes:

**desenvolvimento** e **homologação**.



**Agradeço**  
a sua atenção!

**Monitor Matheus Matos**

[matheus.matos@sptech.school](mailto:matheus.matos@sptech.school)

SÃO  
PAULO  
TECH  
SCHOOL