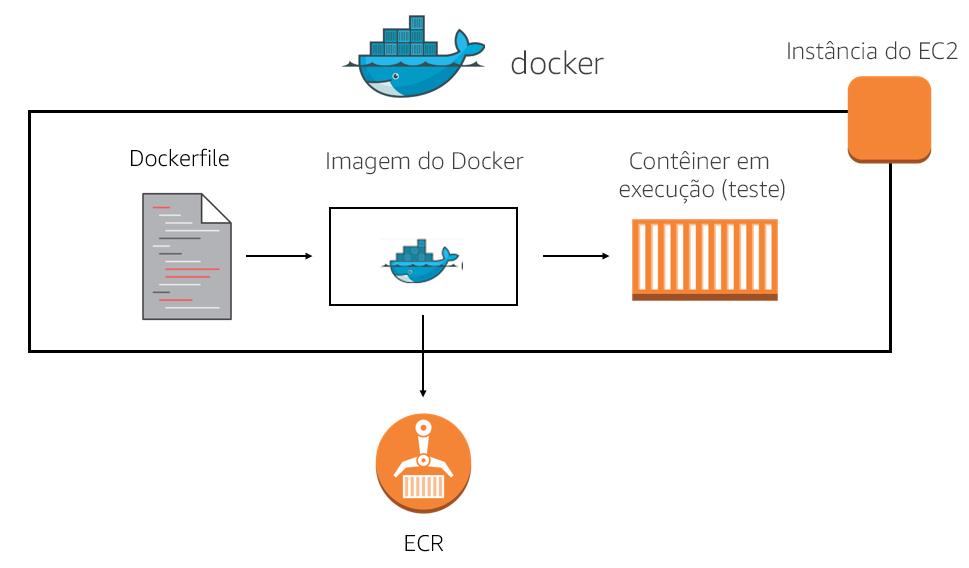
# **Contêineres do Docker**



**Visão geral**

Saber criar imagens do Docker e iniciar contêineres é uma habilidade muito útil para um desenvolvedor. Neste laboratório, você aprenderá a hospedar um site básico usando contêineres do Dockers no **AWS Cloud9**.

Você usará o AWS Cloud9 para criar uma instância de desenvolvimento do Amazon Elastic Compute Cloud (Amazon EC2) que tenha o Docker instalado. Usando essa instância, você criará uma imagem do Docker e, em seguida, iniciará um contêiner. Em seguida, você criará um repositório do Amazon Elastic Container Registry (Amazon ECR) para enviar a imagem do Docker que criará.



**Objetivos**

Depois de concluir este laboratório, você será capaz de:

* Criar um Dockerfile.
* Criar uma imagem do Docker usando um Dockerfile.
* Executar um contêiner a partir de uma imagem do Docker.
* Interagir com seus contêineres e administrá-los.
* Criar um repositório do Amazon ECR.
* Autenticar o cliente do Docker no Amazon ECR.
* Enviar uma imagem do Docker para o Amazon ECR.

**Duração**

O laboratório requer aproximadamente **60 minutos** para ser concluído.

## **Tarefa 1: Abrir o IDE do AWS Cloud9**

Um ambiente do AWS Cloud9 foi preparado para você.

1. No Console de Gerenciamento da AWS, acesse o menu **Services** (Serviços) e escolha **Cloud9**.
2. Para abrir o ambiente do AWS Cloud9 fornecido, escolha **Open IDE** (Abrir IDE).

## **Tarefa 2: Modificar um grupo de segurança**

A implantação do AWS Cloud9 criou uma instância EC2 em sua conta. Será necessário modificar o grupo de segurança (firewall) dessa instância EC2 para que você possa testar o aplicativo que implantará. Certifique-se de anotar o endereço IP público da instância EC2 para fazer esse teste.

Para modificar o grupo de segurança, você precisa voltar para o Console de Gerenciamento da AWS.

1. Para retornar ao Console de Gerenciamento da AWS a partir do console do AWS Cloud9, escolha **AWS Cloud 9** e, em seguida, **Go To Your Dashboard** (Ir para o painel).
2. No menu **Services** (Serviços), escolha **EC2**.
3. No painel de navegação à esquerda, escolha **Instances** (Instâncias). Você verá uma instância com um nome que começa com **aws-cloud9-Docker-...**.

**OBSERVAÇÃO**: se a instância não estiver visível, você está na região errada. Selecione a mesma região selecionada durante a criação do ambiente do AWS Cloud9.

1. Escolha **aws-cloud9-Docker-...**.
2. Na seção inferior, anote o **IP público IPv4** e guarde-o para uso futuro.
3. Selecione o nome dos **grupos de segurança**, que devem começar com **aws-cloud9-Docker-...**. Isso o levará à seção Security Group do Console de Gerenciamento do EC2.
4. Escolha a guia **Entrada**.
5. Clique em **Edita**.
6. Escolha **Adicionar Regra** e insira o seguinte:

* Em **Tipo**, selecione **HTTP**.
* Em **Origem** , selecione **Qualquer lugar**.

As demais configurações serão preenchidas automaticamente para você.

1. Clique em **Save** (Salvar).

## **Tarefa 3: Trabalhar com o Docker no AWS Cloud9**

Nesta tarefa, você criará uma imagem do Docker, criará um Dockerfile, montará uma imagem do Docker e executará um contêiner do Docker no console do AWS Cloud9.

### **Criar uma imagem do Docker**

1. O Docker já está instalado na instância EC2 do AWS Cloud9. Para verificar se o Docker está instalado e ver as informações da versão, execute o seguinte comando no terminal do AWS Cloud9:

**docker -v**

Você verá uma resposta semelhante à seguinte:

*Docker version 18.03.1-ce, build 3dfb8343b139d6342acfd9975d7f1068b5b1c3d3*

1. Para propagar o sistema de arquivos do AWS Cloud9 com os recursos que incluem um site estático do Amazon Simple Storage Service (Amazon S3), execute o seguinte comando wget no terminal do AWS Cloud9:

**wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/DEV-ILT-TF-200-ACCDEV-1/lab-4-docker.zip**

Você verá que um arquivo *lab-4-docker.zip* foi adicionado ao sistema de arquivos do AWS Cloud9.

1. Para descompactar o arquivo *lab-4-docker.zip*, execute o seguinte comando:

**unzip lab-4-docker.zip**

Esse processo pode levar alguns minutos. No sistema de arquivos do AWS Cloud9, você verá uma pasta *resources* (recursos).

1. Para limpar seu ambiente, execute os seguintes comandos para remover os arquivos.zip e README:

rm \*.zip

rm README.md

### **Criar um Dockerfile**

Nesta etapa, você criará um novo Dockerfile na pasta **root** (raiz).

1. Na barra de menus do AWS Cloud9, escolha **File** (Arquivo) e **New file** (Novo arquivo).
2. Insira o texto a seguir na nova guia **Untitled1** recém-aberta:

FROM ubuntu:noble

# Install apache and remove the list of packages downloaded from apt-get update

RUN apt-get update -y && \

apt-get install -y apache2 && \

rm -r /var/lib/apt/lists/\*

# Copy the website into the apache web root directory

COPY resources /var/www/html

EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]

Este é um exemplo de um Dockerfile. Seu Dockerfile age da seguinte maneira:

* Faz download da imagem do ubuntu de um repositório de imagens.
* Instala o apache e remove os pacotes obtidos por download da atualização apt-get.
* Copia seu aplicativo web para a imagem.
* Expõe a porta tcp/80 para permitir a entrada de conexões HTTP.
* Inicia o Apache.

1. Para salvar o arquivo, escolha **File** (Arquivo) e **Save as** (Salvar como).
2. Em **Filename** (Nome do arquivo), insira: dockerfile.
3. Salve o arquivo na pasta raiz do **Docker**.

### **Criar uma imagem do Docker**

A próxima etapa será criar uma imagem para seu aplicativo.

1. No terminal bash, insira o seguinte comando:

**OBSERVAÇÃO**: copie todo o comando, incluindo o ..

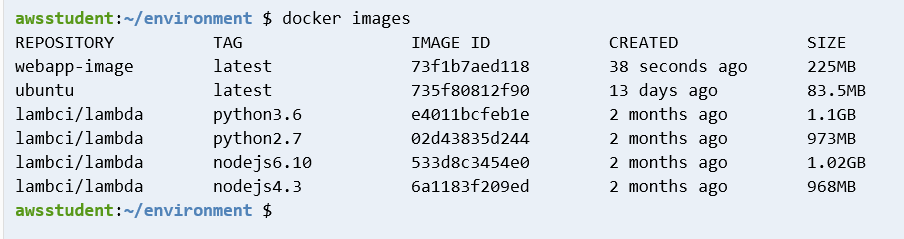
**docker build -t webapp-image .**

Esse comando cria uma imagem de um Dockerfile localizado em . (diretório atual). Em seguida, ele marca a imagem com um name\_webapp\_.

1. Agora você deve ter uma imagem do Docker. Você pode verificar a imagem inserindo o seguinte comando:

**docker images**

Esse comando deve retornar o seguinte:



Você deve ver algumas imagens listadas do docker.

### **Executar um contêiner do Docker**

Nesta seção, você iniciará um contêiner a partir da imagem do Docker que criou.

1. Para iniciar um contêiner, no terminal do AWS Cloud9, insira o seguinte comando:

**docker run --name webapp -d -p 80:80 webapp-image**

Esse comando solicita que o Docker execute um contêiner, com o nome *webapp*, no modo daemon (não interativo) e mapeie tcp/80 fora do contêiner para tcp/80 no interior do contêiner.

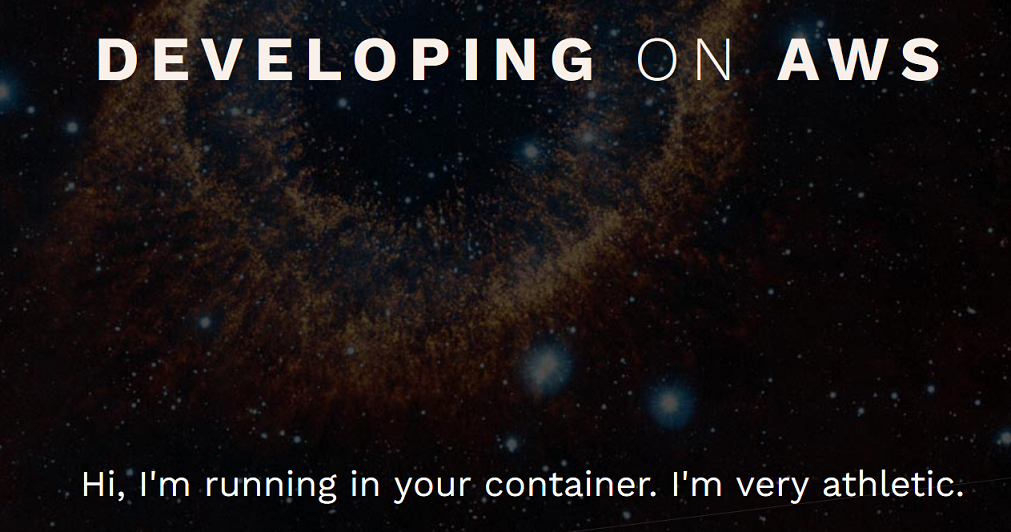
1. Para ver se o contêiner está em execução, insira o seguinte comando:

**docker ps -a**

1. Em seguida, você testará o site. Para fazer isso, use o **IP público IPv4** que você anotou anteriormente e cole-o em um navegador da web.

Por exemplo: **http://<PublicIpAddress>**

Você deve ver uma página da Web semelhante à seguinte:



**Parabéns!** Você iniciou seu primeiro contêiner do Docker.

## **Tarefa 4: Interagir com o contêiner do Docker**

Nesta seção, você usará vários comandos para interagir com o contêiner do Docker.

1. Volte para o seu ambiente do AWS Cloud9.
2. Para abrir um prompt bash no contêiner em execução, execute o seguinte comando:

**docker exec -i -t webapp /bin/bash**

O comando **docker exec** executa um novo comando em um contêiner em execução. O sinalizador *i* o torna interativo por meio do redirecionamento do STDIN.

Agora você deve estar conectado ao contêiner.

O prompt será semelhante ao seguinte:

**root@container-id:/#**

1. No terminal do AWS Cloud9, insira o seguinte comando:

**ls /var/www/html**

Examine o sistema de arquivos. Confirme se os arquivos da pasta resources agora estão na pasta */var/www/html*.

1. Para sair do contêiner, execute o seguinte comando:

**exit**

1. Para visualizar os contêineres em execução, execute o seguinte comando:

**docker ps -a**

**Observação** Anote o *ID DO CONTÊINER*. É possível usar o ID DO CONTÊINER para iniciar e interromper os contêineres.

1. Para interromper o contêiner, insira o seguinte comando:

**docker stop <CONTAINER ID>**

ou

**docker stop webapp**

Por exemplo, se o container-id for *4b6c759654cd* e o nome for *webapp*, será possível interromper o contêiner usando qualquer um dos seguintes comandos:

docker stop webapp

ou

docker stop 4b6c759654cd

ou

docker stop 4b

**Observação** O último comando funciona porque o Docker pode verificar que há apenas um contêiner em execução com um ID de contêiner que começa com '4b'. Esse procedimento pode ser útil e mais rápido do que usar o ID inteiro, especialmente para testes.

1. Inicie o contêiner novamente. Para iniciar o contêiner, insira o seguinte comando:

docker start webapp

1. Em seguida, visualize os logs do contêiner. Para visualizar os logs de contêiner, insira o seguinte comando:

**docker logs webapp**

Você deve ver os logs do apache. Não se preocupe com os logs exibidos, pois você testou o seu contêiner usando o navegador na tarefa anterior, então sabe que ele funciona.

1. Liste os mapeamentos de porta para o contêiner. Para listar os mapeamentos de porta, insira o seguinte comando:

**docker port webapp**

Isso deve retornar o seguinte: *80/tcp -> 0.0.0.0:80*

## 

## 

## 

## 

## 

## 

## 

## 

## **Tarefa 5: Criar um repositório do ECR e enviar sua imagem**

Agora com a imagem, a próxima etapa de um desenvolvedor normalmente seria enviar a imagem para um repositório.

Nesta seção, você usará o Amazon ECR como seu repositório privado. Você fará o seguinte:

* Criar um repositório.
* Marcar sua imagem com este novo repositório.
* Autenticar o cliente docker no repositório do ECR.
* Enviar sua imagem por este repositório.

1. A primeira etapa é criar o repositório do ECR chamado webapp. Para criar o repositório, insira o seguinte comando:

**aws ecr create-repository --repository-name webapp**

O valor de retorno deve ser semelhante a este:

{

"repository": {

"registryId": "012345678901",

"repositoryName": "webapp",

"repositoryArn": "arn:aws:ecr:us-east-1:012345678901:repository/webapp",

"createdAt": 1533951996.0,

"repositoryUri": "012345678901.dkr.ecr.us-east-1.amazonaws.com/webapp"

}

}

Tome nota de **repositoryUri** sem aspas duplas. Você o usará nas etapas a seguir. Neste exemplo, o repositoryUri é: *975049916224.dkr.ecr.us-east-1.amazonaws.com/webapp*

1. A seguir, marque o repositório do ECR webapp-image que você acabou de criar.
2. Para marcar a imagem, no comando a seguir, substitua <FMI> pelo **repositoryUri** anotado na etapa anterior e execute o comando:

**docker tag webapp-image <FMI>**

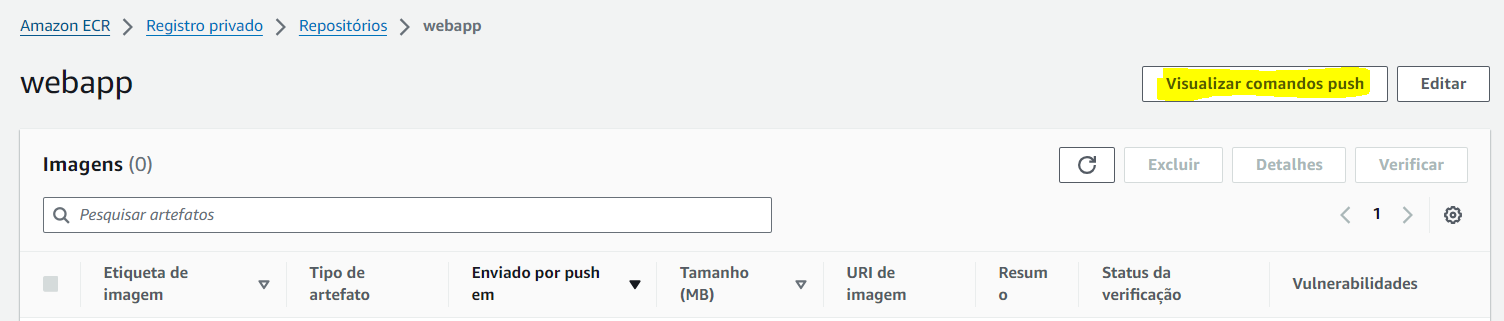
**docker tag webapp-image** *975049916224.dkr.ecr.us-east-1.amazonaws.com/webapp*

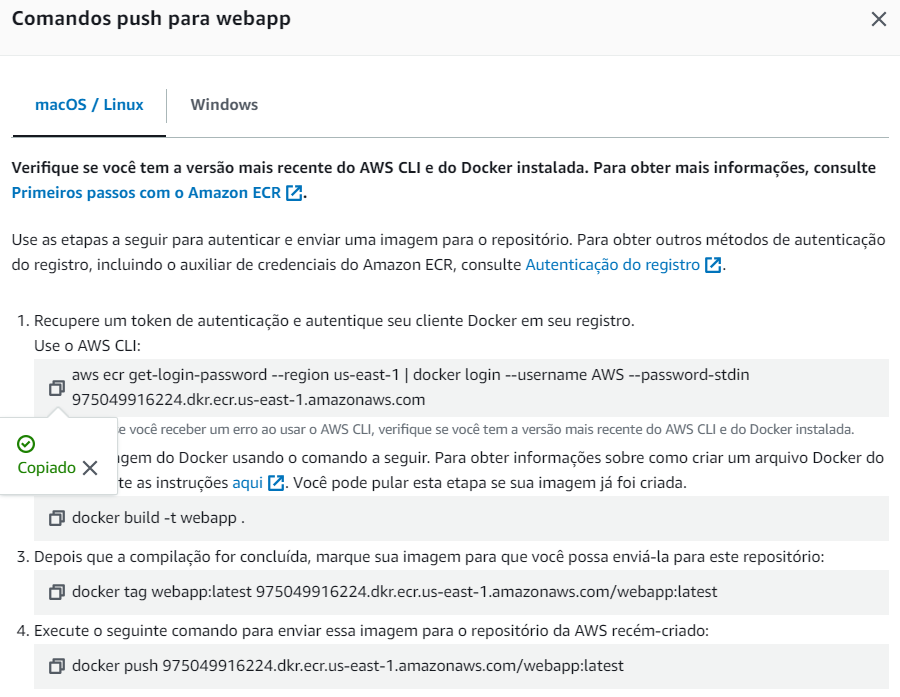
Para fazer um push para o repositório do ECR, a primeira etapa é autenticar o cliente do Docker.

A AWS criou um comando da CLI da AWS, *aws ecr get-login*, para simplificar esse processo.

O comando retorna o comando *docker login -u ...*.

Em vez de copiar a instrução de retorno e colá-la de volta, você executará o comando usando *eval*, que executará o comando retornado por *aws ecr get-login*.





1. Para autenticar o cliente *docker*, insira o seguinte comando:

**eval $(aws ecr get-login --no-include-email)**

**aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 975049916224.dkr.ecr.us-east-1.amazonaws.com**

A saída deve indicar *Login Succeeded* (Login bem-sucedido). É possível ignorar o Aviso porque usar o comando *eval* permitiu que você não colasse a senha na entrada padrão e ficasse visível no histórico.

1. Agora que você está conectado, pode enviar a imagem para o repositório do ECR. Para enviar a imagem, no comando a seguir, substitua <FMI> pelo **repositoryUri** e execute o comando:

**docker push <FMI>**

**docker push** *975049916224.dkr.ecr.us-east-1.amazonaws.com/webapp*

A saída desse comando deve ser semelhante a esta:

The push refers to repository [012345678901.dkr.ecr.us-east-1.amazonaws.com/webapp]

2fc92baef76f: Pushed

9d66cbb6a6ab: Pushed

268a067217b5: Pushed

c01d74f99de4: Pushed

ccd4d61916aa: Pushed

8f2b771487e9: Pushed

f49017d4d5ce: Pushed

latest: digest: sha256:d5f7beae13aed16d307629a943f3d687ae30e437db3ee3cf4a565672d2b39bd2 size: 1779

1. Quando a imagem foi extraída para o repositório

**aws ecr list-images --repository-name webapp**

A saída desse comando deve ser semelhante a esta:

{

"imageIds": [

{

"imageDigest": "sha256:d5f7beae13aed16d307629a943f3d687ae30e437db3ee3cf4a565672d2b39bd2",

"imageTag": "latest"

}

]

}

**Parabéns!** Você concluiu a tarefa e também o laboratório. Você conseguiu realizar com êxito as seguintes tarefas:

* Criar e interagir com contêineres do Docker usando o AWS Cloud9.
* Criar um repositório do ECR e enviar sua imagem para ele.