**DOCKER LOGS, STATS, DOCKERFILE**

**Agenda**

1. **O que é um container**
2. **O que é o Docker**
3. **Laboratório**
   1. **Instalação do docker**
   2. **Validar a instalação**
   3. **Execução de containers (inspect, stats, logs, build)**
   4. **Dockerfile**

**O que é um Container?**

* Uma forma de **empacotar** uma aplicação com todas as **dependências** e **configurações** necessárias;
* **Portabilidade**, facilmente transportado para outro ambiente;
* Contribui para a **eficiência** do desenvolvimento
* É **efêmero**

A ideia é que tenha em um ambiente, tudo que aquela aplicação necessita para executar.

Tem a característica de portabilidade muito presente.

Os containers são feitos para morrer, executar a sua função e serem apagados logo em seguida.

**O que é Docker?**

* Iniciado em 2013;
* PAAS;
* Alternativa aos **virtualizadores**;
* Trabalha no nível de **aplicação**, o que garante uma maior **velocidade** quando comparado com virtualizadores;
* Kernel compartilhado.

**Vamos Praticar**

**Checklist**

1. **Instalação do docker**
2. **Comandos docker (inspect, stats, logs, build)**
3. **DockerFile**
4. **Build multi stage**

**Instalação Docker**

**Remover as versões antigas**

sudo apt-get remove docker docker-engine docker.io containerd runc

**Instalar pacotes complementares**

sudo apt-get install ca-certificates curl gnupg lsb-release

**Adicione a chave GPG oficial do Docker**

curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo gpg –dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

**Adicione o repositório do docker na source list da distro Ubuntu**

echo \

“deb [arch=$ (dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] <https://download.docker.com/linux/ubuntu> \

$(lsb\_release -cs) stable” | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

**Instalação**

apt-get update && apt-get install docker-ce docker-ce-cli containerd.io

**Digite:** docker -v

**Resultado:** Docker version xx.xx.xx, build xxxxx

OU

wget -qO- <https://get.docker.com/> | sh

**Validar instalação**

**Adicione seu user no grupo do docker:** sudo usermod -aG docker <nome usuario>

**Digite:** docker run hello-world

**LOGS**

Sintaxe para verificar os logs do container:

* docker logs [OPTIONS] [CONTAINER ID]
* docker logs -f –until=5s CONTAINER ID: visualiza os ultimos logs registrados, atualizando a cada 5s, equivalente ao comando tail -f

Este comando serve para verificar a saída de erros e acertos para fazer o troubleshooting, caso aconteça algum problema no container, assim, o usuário consegue ter acesso ao que está acontecendo com o container, sem precisar acessar ele.

Fazer troubleshooting de dentro do container não é uma boa prática

**STATS**

Sintaxe para verificar as métricas de consumo do container dentro do host

* docker stats CONTAINER ID
* docker stats --format “{{.Container}}”: lista os containers ID ativos
* docker stats --format “{{.Container}} → {{.CPUPerc}}”: lista a percentagem de de uso do container dentro do host

Com os comando LOG e STATS, é possível verificar como está a saúde do seu container em cima do seu host

**DOCKERFILE**

**O que é DockerFile**

O Dockerfile é um meio que utilizamos para criar nossas próprias imagens.

**Exemplos:**

From -> Caminho da imagem

RUN -> Execução de comandos

Expose -> Exposição de portas

CMD -> Comando para execução do container

COPY -> Copia arquivos do seu ambiente para o container. Ex: COPY [origem] [destino]

ADD -> Adiciona seus arquivos e pasta do seu host para o container

Exemplo feito na aula de um Dockerfile:

FROM debian:latest -> Imagem e versão que será baixada para criação do container

LABEL mantainer=”Equipe formacao SRE Elvenworks” -> Instrução para melhor identificação da imagem, com a informação da equipe que desenvolveu o dockerfile

RUN apt-get update && apt-get upgrade -> comandos para atualizar e instalar as atualizações das listas de pacotes

RUN apt-get install nginx -y -> comando para instalar o nginx no container e autorizando automaticamente a instalação (se não tiver a flag -y o docker build não funciona)

EXPOSE 80 -> expor a porta 80 do host para o container utilizar

CMD [“nginx”, “-g”, “daemon off;”] -> commando para iniciar o serviço do nginx e desligar o serviço do daemon

Para fazer a criação de uma imagem com dockerfile, o comando é:

* docker build [endereço do arquivo Dockerfile no host]

E o docker irá criar a imagem com as especificações de acordo com os comandos de dentro do dockefile, tanto imagem, quanto configurações de porta, atualização de pacotes, serviços, etc…

ps: É importante sempre informar no dockerfile o usuário/imagem:versão, para identificação da imagem, uma imagem sem identificação, se for criada outra, será sobrescrita

**MULTI-STAGE BUILD**

No dia a dia na hora de subir os containers com as imagens criadas, é importante que elas sejam o menor possível, pois em ambientes corporativos é comum existirem muitos containers sendo executados ao mesmo tempo, assim consumindo muita memória e quanto maior, menos containers poderão ser executados.

O multi-stage build faz todo o processamento da imagem solicitada e disponibiliza só o necessário para a criação da imagem que irá ser utilizada no container, diminuindo assim o seu tamanho.

Ex:

FROM golang:1.16 -> Criar uma imagem do go

WORKDIR /go/src/github.com/exemplo/ -> Criar o diretório do projeto

RUN go get -d -v golang.org/x/net/html -> Processamento os arquivos do go

COPY app.go ./ -> Copiar os arquivos

RUN CGO\_ENABLED=0 GOOS=linux go build -a installsuffix cgo -o app . -> Binário disponível

FROM alpine:latest -> Criar imagem linux alpine

RUN apk –no-cache add ca-certificates -> Carregar as dependências

WORKDIR /root/ -> Criar o diretório do projeto que estará o binário

COPY --from=0 /go/src/github.com/exemplo/app ./ -> Copiar os dados do diretório go para o do container

CMD [“./app”] -> executar o binário

A primeira imagem, realizou o processamento pesado das dependências necessárias

A segunda imagem utilizou somente o que é preciso para o funcionamento da mesma

**Remover o LAB**

**Exemplo:**

* docker stop $(docker ps -a -q) -> **Parar todos os containers**
* docker rm $(docker ps -a -q) -> **Apagar todos os containers**
* docker rmi -f $(docker images -q) -> **Apagar todas as imagens**