**Monitoria linux Docker + Imersão Linux**

**Instalando o Docker via script**

#!/bin/bash  
echo "Instalando o Docker......."  
apt-get update -y  
apt-get install ca-certificates curl gnupg lsb-release -y  
mkdir -p /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o /etc>

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gp>

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

apt-get update -y

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

systemctl start docker

systemctl enable Docker

**Criar volumes para espelhando conteiners**

docker volume create <volume>

**Rodar um container (mysql)**

docker run --name mysql-server -p 3306:3306 -v data:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=123 -d mysql:8.0.3

**Rodar um container (aplicação dentro de um volume Apache)**

docker run --name web-server -dt -p 80:80 --mount type=volume,src=app,dst=/app/ webdevops/php-apache:alpine-php7

**Parando aplicação para poder ativar o swarm**

docker rm --force <nome do container>

docker ps - verifica os containers ativos

**Inicia o gerenciamento de cluster Swarm**

docker swarm init – gera código para adicionar nas máquinas que serão parte do cluster

exemplo : docker swarm join –token   
SWMTKN-1-662rcs658pklegd9f1xtyb0g5us91jbvn8aeyg1lkdp74f358m-2h3cxcwfxledl330gtvwo2ct6 192.168.15.86:2377

**Verificando os nós(hosts) do cluster**

docker node ls

retorno

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION

sh7a8z6jqnqp5su08nyvt1w5d \* servidor1 Ready Active Leader 20.10.17

qdjeo602hb71pdg7n36qm67x0 servidor2 Ready Active 20.10.17

rqblowonv3q6qtk2c6eexhvhn servidor3 Ready Active 20.10.17

**Replicando 3 containers do web-server na maquina gerente**

docker service --name web-server --replicas 3 -dt -p 80:80 --mount type=volume,src=app,dst=/app/ wepdevops/php-apache:alpine-php7

**Verificando a carga das replicas criadas**

docker service ps web-server

ele não duplica o conteúdo do volume(conteúdo da aplicação) automaticamente..

**Replicando o conteúdo do volume**

1-instalar o serviço nfs-server para a replicação  
 apt-get install nfs-server

2-instalar o serviço nfs-common nas demais maquinas  
 apt-get install nfs-common

3-configurar o nfs

Nano /etc/exports - acrescentar no final para liberar para qualquer um ip(outras configurações pesquisar)   
 /var/lib/docker/volumes/app/\_data \*(rw,sync,subtree\_check)

4- subir o serviço

Systemctl start nfs-server

**Na maquina server iniciar o compartilhamento**

exportfs -ar

showmount -e ---- mostra o que está compartilhado na maquina

**Na maquina cliente buscar o compartilhamento**

mount -o v3 <ip maquina server>:<path da pasta origem> <path da pasta destino>

Agora criar um proxy onde para cada requisição na principal replique o conteúdo para todas

1. Criar arquivo de configuração /proxy/nginx.conf

http {

upstream all {

server 192.168.15.86:80;

server 192.168.15.45:80;

server 192.168.15.162:80;

}

server {

listen 4500;

location / {

proxy\_pass http://all/;

}

}

}

events {}

1. Criar o container com o nginx e colocar o arquivo de conf dentro criando um arquivo /proxy/dockerfile

FROM nginx

COPY nginx.conf /etc/nginx/nginx.conf

1. Criar a imagem com a conf feita dentro de /proxy  
   docker build -t proxy-app .
2. Subir o container com essa imagem  
   docker container run --name my-proxy-app -dti -p 4500:4500 proxy-app

Acessar a aplicação pela nova porta e todas as requisições serão balanceadas pelas maquinas e containers