Docker

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

Considerações Iniciais

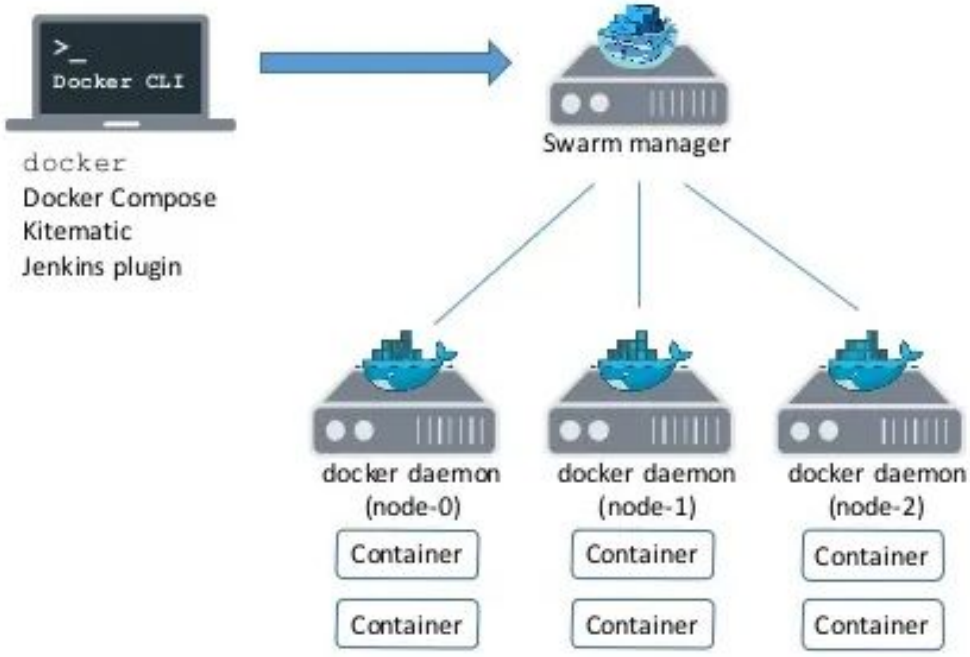
- Nosso horário é das **19:00 às 22:00 (10 min tolerância de chegada)**
- Intervalo às **20:30 de 20 minutos**
- Interaja na aula!
- Se tiver dúvidas, levanta a mão no chat e eu explico novamente
- Me corrija!!!!
- Façam as atividades práticas, só se aprende praticando!
- Qualquer coisa me procurem no **slack**

Docker swarm

Docker swarm

Com Docker Swarm é possível dividirmos a carga de trabalho com outros nodes Docker e criar um cluster. Com o cluster criado, podemos fazer a orquestração (ou deixar que o docker faça) dos containers.

Docker swarm



Docker swarm

Vamos lá!! Primeiramente vamos criar 3 VMs, uma sendo o master e outros dois sendo workers. Para isso, usaremos o Multipass (disponível no site da Ubuntu) por ser uma solução leve e prática para criação das virtual machines.

Outra opção é usar o ambiente de teste do Docker Swarm (disponível por 4 horas):

https://labs.play-with-docker.com/

Docker swarm

Com multipass instalado vamos criar as instâncias:

- worker1-swarm
  - multipass launch --name worker1-swarm
- worker2-swarm
  - multipass launch --name worker2-swarm
- Master-Swarm
  - multipass launch --name master-swarm

Docker swarm

O multipass sobe as instâncias em seu virtualizador local. No meu caso, usei o Hyper-v. A opção local.driver pode ser alterado para outro, como VirtualBox.

Podemos acessar o bash das instâncias com o comando:

multipass exec nomeInstancia /bin/bash

Docker swarm

Como boa prática, alterei a senha do usuário ubuntu com comando:

sudo passwd ubuntu

No Hyper-v, limitei o uso de CPU e RAM e coloquei as 3 VMs na mesma rede do meu Host.

Docker swarm

Para copiar e colar os comandos com mais facilidade decidi acessar as VMs via SSH. Para habilitar esse serviço siga os comandos:

```
cd /etc/ssh
vim sshd_config
```

alterar PasswordAutentication para yes

reiniciar a VM

Acessando a VM:

```
ssh ubuntu@ip-da-VM
```

Docker swarm

Agora, vamos executar tudo com usuário root:

sudo su

faremos agora a instalação do Docker nas 3 VMs:

curl -fsSl https://get.docker.com | bash

agora a instação do docker-compose:

apt-get install docker-compose

Docker swarm

Para conferir se tudo correu bem nas instalações:

docker version

docker-compose version

;)

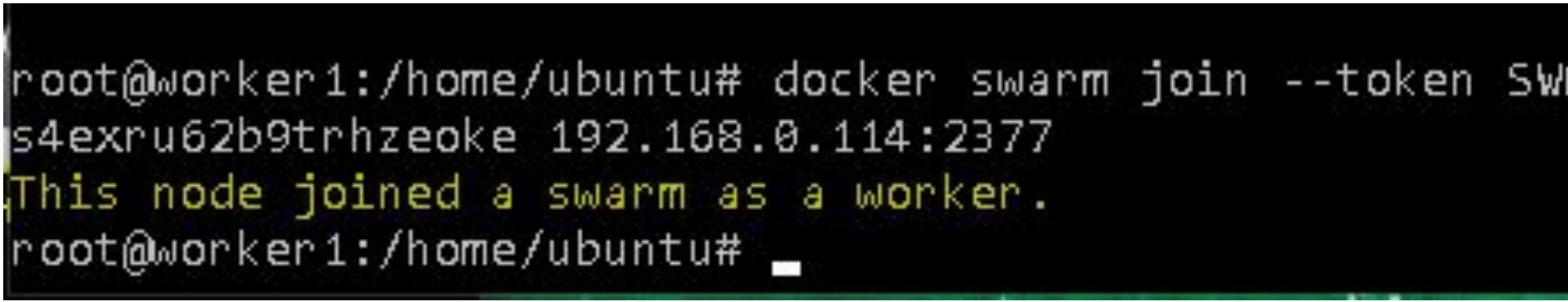
Docker swarm

com comando docker swarm init podemos iniciar o cluster. O parâmetro –-listen-addr foi usado para especificar o ip do manager

```
root@docker18:/home/ubuntu# docker swarm init --listen-addr 192.168.0.114
Swarm initialized: current node (fcmjjs085jizzsuoxbmxrsueb) is now a mana

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-249dzesgyybe5m66skb3v1wtbelb590n9j
2.168.0.114:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and

root@docker18:/home/ubuntu#
```

Docker swarm

Pronto, agora temos nosso cluster iniciado!! O docker informa o comando com token de ingressão ao cluster. Esse comando deve ser copiado e colado nos demais nodes que serão os workers. Abaixo, um exemplo.



Docker swarm

após incluirmos todos os nodes em nosso cluster podemos dar o comando

docker nodes ls para saber status de cada um

```
root@docker18:/home/ubuntu# docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
fcmjjs085jizzsuoxbmxrsueb *   docker18   Ready     Active         Leader           20.10.11
j0v11z08htqso7ytsj7qcnn8d     worker1    Ready     Active                          20.10.13
xq9yjswcltqyzbj726lwy51bv     worker2    Ready     Active                          20.10.13
root@docker18:/home/ubuntu#
```

Docker swarm

Agora podemos executar arquivos docker compose e dividir as cargas de trabalho com workers. Para isso usaremos o arquivo docker-compose.yml em cluster com o comando:

docker stack deploy --compose-file=docker-compose.yml testeswarm

```
root@swarm-master:/home/ubuntu# docker stack deploy --compose-file=docker-compose.yml testeswarm
Ignoring unsupported options: links

Ignoring deprecated options:

container_name: Setting the container name is not supported.

Creating network testeswarm_default
Creating service testeswarm_postgres-compose
Creating service testeswarm_adminer-compose
root@swarm-master:/home/ubuntu#
```

Docker swarm

com o comando docker service ls vemos que foram criados os dois serviços do compose file. Com comando docker service ps nomeServico podemos saber em qual node está sendo executado.

```
root@swarm-master:/home/ubuntu# docker service ls
ID             NAME                              MODE         REPLICAS   IMAGE              PORTS
zil0m7ase6dl   testeswarm_adminer-compose        replicated   1/1        adminer:latest     *:8080->8080/tcp
kjqrejz22qsn   testeswarm_postgres-compose       replicated   1/1        postgres:latest    *:5432->5432/tcp
root@swarm-master:/home/ubuntu# docker service ps testeswarm_adminer-compose
ID             NAME                               IMAGE            NODE            DESIRED STATE   CURRENT STATE
RROR     PORTS
6pznvltcdrvf   testeswarm_adminer-compose.1   adminer:latest   worker1-swarm   Running         Running 3 minutes ago

root@swarm-master:/home/ubuntu#
```

Docker swarm

Colocando o ip do manager no navegador na porta 8080 podemos acessar o adminer, por mais que esteja executando no worker1.

Docker swarm

Podemos escalar nossos serviços de acordo com a demanda de requisições por exemplo. No nosso exemplo, escalamos o adminer para duas réplicas com o comando docker service scale testeswarm_adminer-compose=2

```
root@swarm-master:/home/ubuntu# docker service scale testeswarm_adminer-compose=2
testeswarm_adminer-compose scaled to 2
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
root@swarm-master:/home/ubuntu#
```

Docker swarm

```
root@swarm-master:/home/ubuntu# docker service ls
ID             NAME                                MODE         REPLICAS   IMAGE             PORTS
zil0m7ase6dl   testeswarm_adminer-compose          replicated   2/2        adminer:latest    *:8080->8080/tcp
kjqrejz22qsn   testeswarm_postgres-compose         replicated   1/1        postgres:latest   *:5432->5432/tcp
root@swarm-master:/home/ubuntu#
```

Podemos ver que o serviço possui dois container em execução. O novo, foi criado no worker2-swarm

```
root@swarm-master:/home/ubuntu# docker service ps testeswarm_adminer-compose
ID             NAME                                IMAGE             NODE              DESIRED STATE   CURRENT STA
    ERROR     PORTS
6pznvltcdrvf   testeswarm_adminer-compose.1         adminer:latest    worker1-swarm     Running         Running 16

0vsulu47wq6v   testeswarm_adminer-compose.2         adminer:latest    worker2-swarm     Running         Running abo
o
root@swarm-master:/home/ubuntu#
```

Docker swarm

No worker2-swarm, podemos ver a réplica extra criada no serviço testeswarm_adminer-compose no manager

```
root@worker2-swarm:/home/ubuntu# docker container ls
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS      NAMES
26dcb567436c   adminer:latest  "entrypoint.sh docke…"   4 minutes ago   Up 4 minutes   8080/tcp   testeswarm_adminer-co
mpose.2.0vsulu47wq6vcx1odcgxrzs6u
root@worker2-swarm:/home/ubuntu#
```

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>