

Criando um Cluster com Docker Swarm.

Nomes: João Pedro Toledo - Lucas Monteiro Ribeiro

5°ADS

Redes de Computadores.

Para realizar este trabalho, foi utilizado o “**play with docker**” ou “**PWD**”. O PWD é um playground do Docker que permite aos usuários executarem os comandos do Docker em questão de segundos. Ele oferece a experiência de ter uma máquina virtual Alpine Linux gratuita no navegador, na qual você pode criar e executar contêineres do Docker e até mesmo criar clusters no modo Docker Swarm. O PWD disponibiliza um acesso de 4h até que seja encerrado a sessão do usuário.

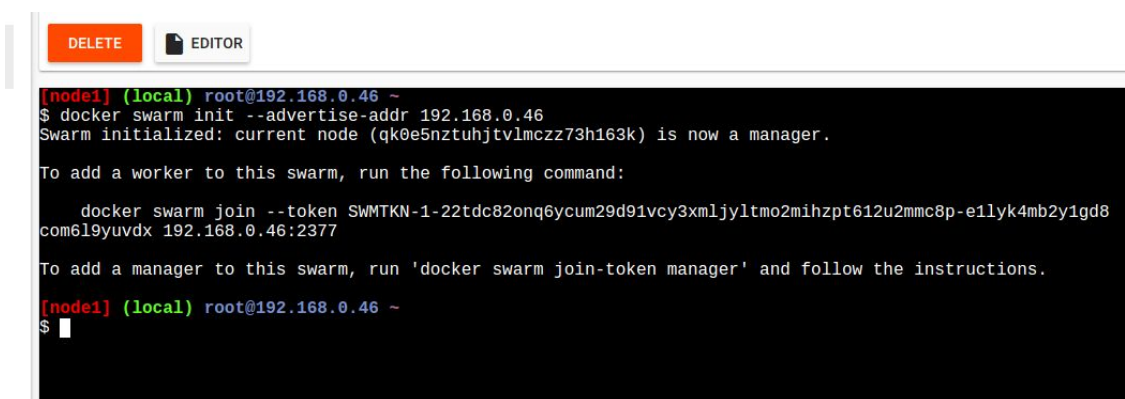
Para ter acesso ao Play with docker, basta acessar o site <https://labs.play-with-docker.com/> então é só criar uma conta, e começar a utilizar o docker.

Cluster com Docker Swarm.

1. Para inicializar o Docker Swarm é necessário começar com este comando:

```
docker swarm init --advertise-addr $(hostname -i)
```

Logo após de digitar o comando acima para inicializar o docker swarm, ele retornará uma mensagem como esta logo abaixo:



```
[node1] (local) root@192.168.0.46 ~
$ docker swarm init --advertise-addr 192.168.0.46
Swarm initialized: current node (qk0e5nztuhjtvlmcz73h163k) is now a manager.

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

    docker swarm join --token SWMTKN-1-22tdc82onq6ycum29d91vcy3xmljy1tmo2mihzpt612u2mmc8p-e1lyk4mb2y1gd8
com6l9yuvdx 192.168.0.46:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[node1] (local) root@192.168.0.46 ~
$
```

Na saída do seu swarm init, você recebe um comando no meio que se parece com o **docker swarm join -token <numeroToken>** que você usa para unir nós de **worker** ao swarm. Você também recebe um segundo comando **docker swarm join-token manager** para adicionar outros **Manager**.

2. Para adicionar outros managers no docker swarm basta digitar o comando "**docker swarm join-token manager**" como mostra na imagem abaixo:

ip172-18-0-10-bg5fon49cs9g00c79h50@direct.labs.play-with-c

DELETE

EDITOR

```
root@192.168.0.46 ~
docker swarm join-token manager
To add a manager to this swarm, run the following command:

docker swarm join --token SWMTKN-1-22tdc82onq6ycum29d91vcy3xmljyltmo2mihzpt612u2mmc8p-azl7j1xvrgpf951f97pm4qmf 192.168.0.46:2377

root@192.168.0.46 ~
```

Após digitar o comando, o docker lhe dará um outro comando como mostra abaixo com um token e um endereço de host, basta copiar este novo comando e executar em outra máquina, e assim será criado um novo manager.

DELETE

EDITOR

```
node2] (local) root@192.168.0.47 ~
docker swarm join --token SWMTKN-1-22tdc82onq6ycum29d91vcy3xmljyltmo2mihzpt612u2mmc8p-azl7j1xvrgpf951f97pm4qmf 192.168.0.46:2377
This node joined a swarm as a manager.

node2] (local) root@192.168.0.47 ~
```

3. Para criar um worker, e obter um token, basta seguir o mesmo comando do manager, porém substituindo a palavra manager por worker ao final do comando, veja o exemplo abaixo:

```
[node2] (local) root@192.168.0.47 ~
$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-22tdc82onq6ycum29d91vcy3xmljyltmo2mihzpt612u2mmc8p-e1lyk4mb2y1gd8com6l9yuvdx 192.168.0.47:2377

[node2] (local) root@192.168.0.47 ~
$
```

Assim que obter o token para se criar o worker, basta copiá-lo para outra máquina e executar o comando. (Deve ser uma máquina diferente da máquina que foi criado o manager)

```
[node3] (local) root@192.168.0.48 ~
$ docker swarm join --token SWMTKN-1-22tdc82onq6ycum29d91vcy3xmljyltmo2mihzpt612u2mmc8p-e1lyk4mb2y1gd8com6l9yuvdx 192.168.0.47:2377
This node joined a swarm as a worker.
[node3] (local) root@192.168.0.48 ~
$
```

4. Com o comando “**docker node ls**” voce consegue verificar todos os nós do seu cluster, e também os workers que foram criados.

```
[node1] (local) root@192.168.0.46 ~
$ docker node ls
```

ID	ENGINE VERSION	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS
qk0e5nztuhjtv1mczz73h163k *	18.06.1-ce	node1	Ready	Active	Leader
l0griepgc0qr4vi6l3vr5yffzq	18.06.1-ce	node2	Ready	Active	Reachable
uk4tb8b3874k0ykebjfor0rqu	18.06.1-ce	node3	Ready	Active	

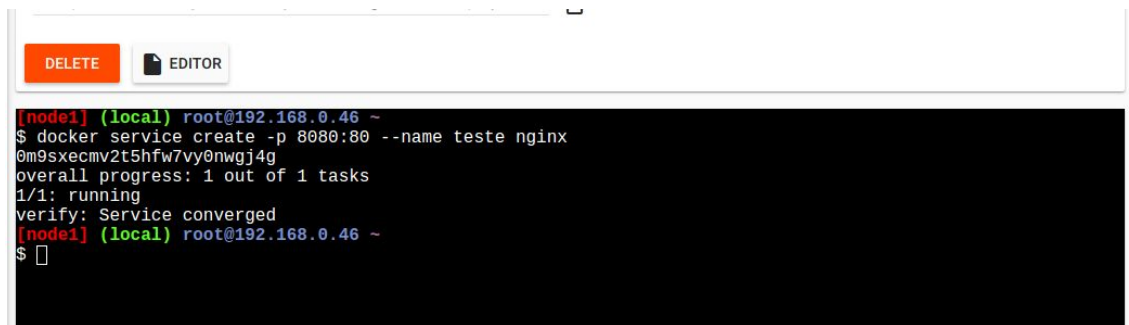
```
[node1] (local) root@192.168.0.46 ~
```

O manager que tem o status como “**leader**” é quem está gerenciando todo o processo do cluster, e o segundo manager com status “**reachable**” é o manager que pode assumir a liderança a qualquer momento caso o primeiro saia. Os workers não possuem status, e são responsáveis apenas por executar os containers do docker.

A partir de agora já temos um cluster criado com o docker swarm, basta apenas levantarmos um serviço.

Criando serviço.

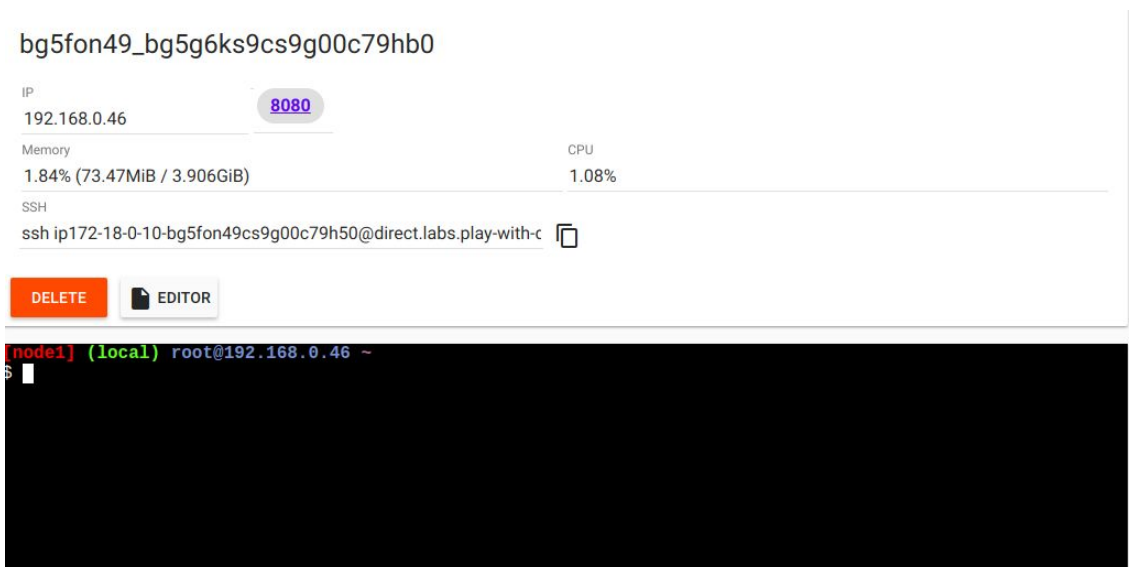
5. O comando para se levantar o serviço no docker swarm é “**docker service create -p 8080:80 --name <nome do serviço><imagem do serviço>**”



```
[node1] (local) root@192.168.0.46 ~
$ docker service create -p 8080:80 --name teste nginx
0m9sxecmv2t5hfw7vy0nwgj4g
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
[node1] (local) root@192.168.0.46 ~
$
```

A imagem acima mostra o serviço criado rodando na porta 8080 com o nome de “**teste**” e com a imagem do “**nginx**”, que é um servidor HTTP. Logo após executarmos o comando, o docker mostra o ID do serviço e o status “**running**” que significa que ele está rodando.

Por fim, com o serviço rodando basta clicar na porta logo acima ao lado do endereço IP que mostrada no play with docker:



bg5fon49_bg5g6ks9cs9g00c79hb0

IP
192.168.0.46 **8080**

Memory
1.84% (73.47MiB / 3.906GiB)

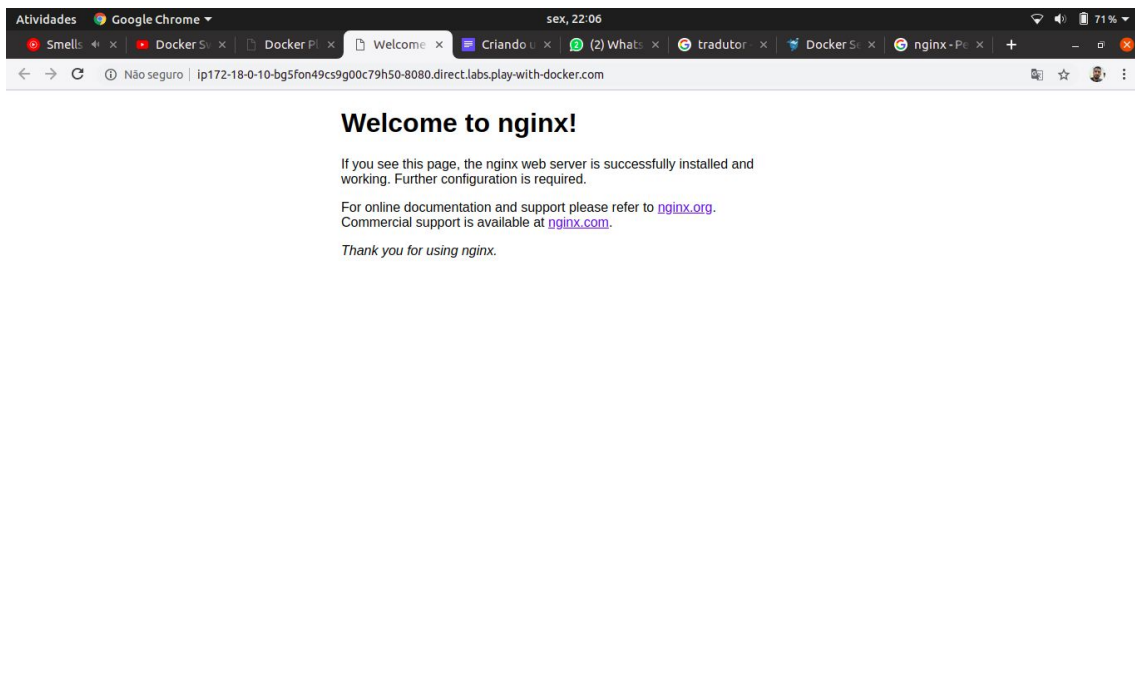
CPU
1.08%

SSH
ssh ip172-18-0-10-bg5fon49cs9g00c79h50@direct.labs.play-with-c

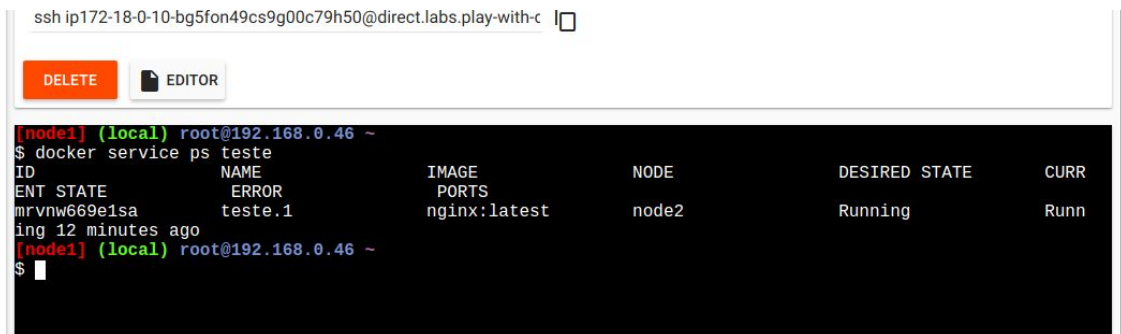
DELETE EDITOR

```
[node1] (local) root@192.168.0.46 ~
$
```

Ao clicar na porta, você será redirecionado para a página do serviço:



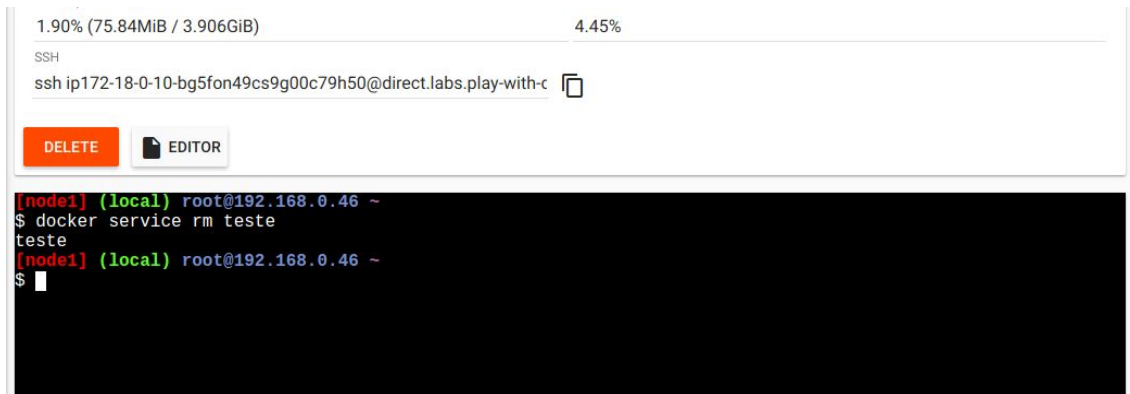
6. Com o comando “**docker service ps <nome ou ID do serviço>**” você consegue verificar todos os serviços criados no seu cluster e o status de cada um como mostra na imagem abaixo:



Caso preferir você também pode rodar o comando “**docker service ls**”, ele irá mostrar os detalhes do serviço como a porta em que está rodando, nome e id.



7. Caso deseje excluir um serviço do seu cluster, basta digitar o comando “**docker service rm <nome do serviço>**”, assim o serviço especificado será excluído do cluster:



```
1.90% (75.84MiB / 3.906GiB) 4.45%
SSH
ssh ip172-18-0-10-bg5fon49cs9g00c79h50@direct.labs.play-with-c
[DELETE] [EDITOR]
[node1] (local) root@192.168.0.46 ~
$ docker service rm teste
teste
[node1] (local) root@192.168.0.46 ~
$
```

Complemento para melhor entendimento.

Manager nodes

Os nós do Manager manipulamos as tarefas de gerenciamento de cluster:

- mantendo o estado do cluster
- agendamento de serviços
- servindo pontos de extremidade da API HTTP do modo swarm

Os Managers mantêm um estado interno consistente de todo o enxame e todos os serviços executados nele. Para fins de teste, não há problema em executar um enxame com um único Manager. Se o Manager em um Manager único falhar, seus serviços continuarão sendo executados, mas você precisará criar um novo cluster para recuperar.

Para aproveitar os recursos de tolerância a falhas do modo de enxame, o Docker recomenda que você implemente um número ímpar de nós de acordo com os requisitos de alta disponibilidade da organização. Quando você tem vários Managers, pode recuperar-se da falha de um nó do Manager sem tempo de inatividade.

- Um enxame de três Managers tolera uma perda máxima de um Manager
- Um enxame de cinco Managers tolera uma perda máxima simultânea de dois nós Managers.
- Um N cluster de Managers tolera a perda da maioria dos $(N-1)/2$ Managers

Work nodes

Os nós de worker também são instâncias do Docker Engine cujo único propósito é executar contêineres. Os nós do worker não participam do estado distribuído do Raft, não tomam decisões de planejamento nem exibem a API HTTP do modo swarm.

Você pode criar um enxame de **um nó manager, mas não pode ter um nó worker sem pelo menos um nó manager**. Por padrão, todos os manager também são worker. Em um único cluster de nó de manager, você pode executar comandos como **docker service create** e o planejador coloca todas as tarefas no Mecanismo local.