



Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn
Relatório - Ferramenta de monitoramento de log's

Múcio Jaziel Alves de Andrade
Flávio da Silva Neves
Vitor Cardin Meneses
Lucas Pires Silveira
Gabriel Santana Fontanari

Email's:
{mjaa, fns2, vcm3, lps6, gsf4}@cin.ufpe.br

03 de Julho de 2019
Recife - PE

1. Introdução:

Um software pode ser desenvolvido seguindo uma abordagem monolítica, neste caso, o software atua como um “bloco” funcionando e respondendo como um todo, trazendo alguns problemas, em caso de mal funcionamento de partes chaves da aplicação. Ainda é preciso salientar que mesmo que uma aplicação use conceitos tais como MVC, ou outras abordagens que tentam modularizar, em muitos casos, cada módulo, funciona ainda dentro da aplicação “mãe” por assim dizer.

A abordagem de microsserviços busca minimizar possíveis problemas, como escalabilidade, tempo de resposta, entre outros, “quebrando” a aplicação em partes menores e mais facilmente gerenciáveis.

Um dos desafios dessa abordagem é o monitoramento de logs. Alguns sistemas, fazem o monitoramento de máquinas, linguagens etc, alguns exemplos são: Nagios, Zabbix, Zipkin, esses sistemas precisam em muitas vezes, serem adaptados para atender a demanda acima citada.

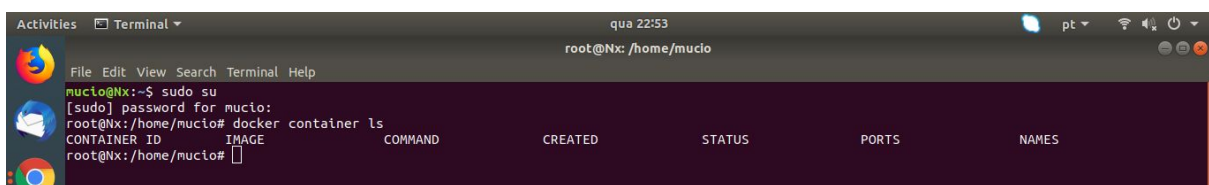
Assim, o projeto `monitoring_ms` nasceu dessa observação, monitorar log's em aplicações em produção, uma vez que várias soluções eram adaptadas para este fim, tornou-se uma necessidade aplicações que fossem focadas em análises desse tipo, possibilitando respostas mais rápidas aos SysAdmin's, e que pudesse ser implantada em sistemas em produção.

2. Pré-requisitos:

- `Monitoring_ms`
 - Python 3
 - Docker-ce
 - Configure docker swarm nodes
 - Configure a docker network of type overlay
 -
- `Music_ms`
 - Docker ce
 - Docker compose

3. Passos:

Usamos o comando abaixo para listar os container's do docker, nesse passo já tínhamos desinstalado e instalado novamente o docker, para confirmar que estávamos corretos quanto a instalação do mesmo. E portanto, optamos por remover todos container's e possíveis máquinas virtuais dockerizadas, nenhum container nesse caso existe até o momento.



```
Activities  Terminal  qua 22:53
root@Nx: /home/mucio

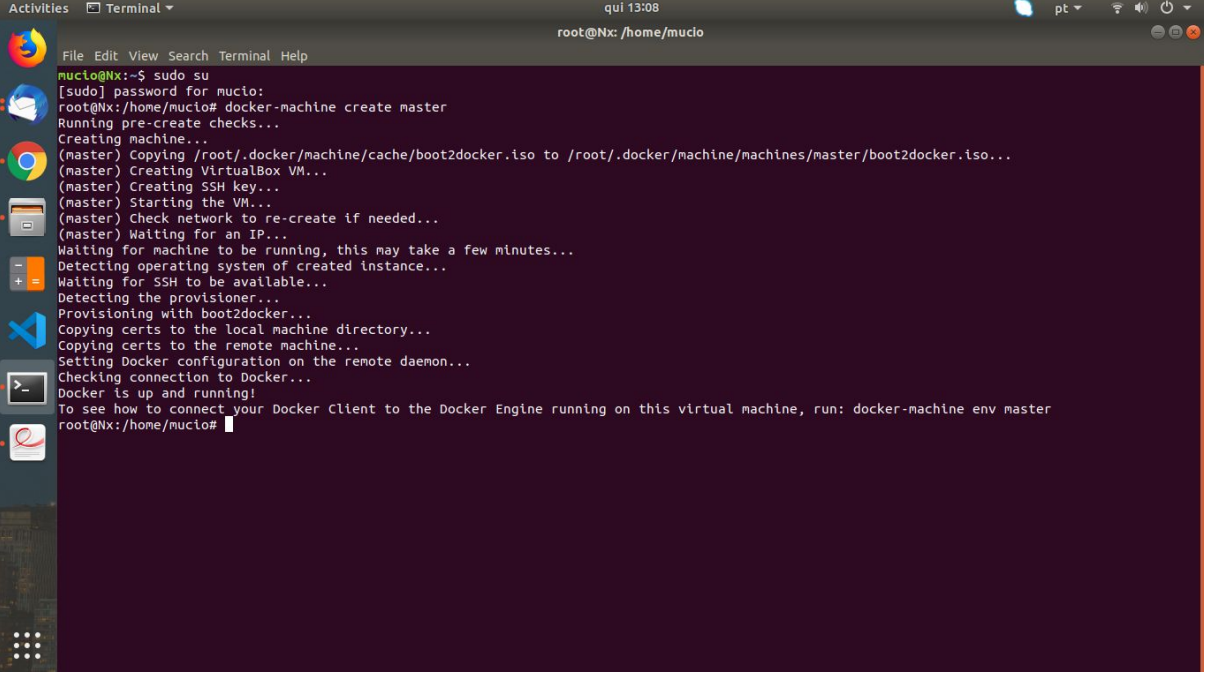
File Edit View Search Terminal Help
mucio@Nx:~$ sudo su
[sudo] password for mucio:
root@Nx:/home/mucio# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
root@Nx:/home/mucio#
```

As aplicações, back-end que servirá como validação (Music_ms) e front-end (Monitoring_ms) devem usar a tecnologia do swarm, você pode aprender mais aqui *, vamos criar 1 nó, que os chamaremos de master, e 3 slaves0, 1 e 2, a ordem de criação não importa, e sim o que cada nó, ou host irá desempenhar;

Usaremos o comando, conforme mostra a imagens a seguir:

\$ docker-machine create master

Repetimos o passo para criar as demais máquinas;



```
Activities Terminal
qui 13:08
root@Nx: /home/mucio

mucio@Nx:~$ sudo su
[sudo] password for mucio:
root@Nx:/home/mucio# docker-machine create master
Running pre-create checks...
Creating machine...
(master) Copying /root/.docker/machine/cache/boot2docker.iso to /root/.docker/machine/machines/master/boot2docker.iso...
(master) Creating VirtualBox VM...
(master) Creating SSH key...
(master) Starting the VM...
(master) Check network to re-create if needed...
(master) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env master
root@Nx:/home/mucio#
```

Feito isso, usamos o comando:

\$ docker-machine ssh master

Desta forma, vamos acessar o terminal da mesma, conforme mostra a imagem a seguir,

```
Activities Terminal qui 13:12
root@Nx: /home/mucio

File Edit View Search Terminal Help

mucio@Nx:~$ sudo su
[sudo] password for mucio:
root@Nx: /home/mucio# docker-machine create master
Running pre-create checks...
Creating machine...
(master) Copying /root/.docker/machine/cache/boot2docker.iso to /root/.docker/machine/machines/master/boot2docker.iso...
(master) Creating VirtualBox VM...
(master) Creating SSH key...
(master) Starting the VM...
(master) Check network to re-create if needed...
(master) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env master
root@Nx: /home/mucio# docker-machine ssh master
( ' > ' )
/) TC ( \ Core is distributed with ABSOLUTELY NO WARRANTY.
(/-__-_- \) www.tinycorelinux.net

docker@master:~$
```

No terminal, clonamos o projeto com o seguinte comando:

```
File Edit View Search Terminal Help

Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env slave1
root@Nx: /home/mucio# docker-machine create slave2
Running pre-create checks...
Creating machine...
(slave2) Copying /root/.docker/machine/cache/boot2docker.iso to /root/.docker/machine/machines/slave2/boot2docker.iso...
(slave2) Creating VirtualBox VM...
(slave2) Creating SSH key...
(slave2) Starting the VM...
(slave2) Check network to re-create if needed...
(slave2) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env slave2
root@Nx: /home/mucio# docker-machine ssh master
( ' > ' )
/) TC ( \ Core is distributed with ABSOLUTELY NO WARRANTY.
(/-__-_- \) www.tinycorelinux.net

docker@master:~$ git clone https://github.com/fabiopina/monitoring_ms/
Cloning into 'monitoring_ms'...
remote: Enumerating objects: 1244, done.
remote: Total 1244 (delta 0), reused 0 (delta 0), pack-reused 1244
Receiving objects: 100% (1244/1244), 827.36 KiB | 703.00 KiB/s, done.
Resolving deltas: 100% (654/654), done.
Checking connectivity... done.
docker@master:~$
```

Execute os seguintes comandos, para acessar a pasta e executar o arquivo de script run.sh [ele será responsável por fazer o build do grafana, zuul, registry, mariaDB, influxDB, eureka:

\$ cd monitoring_ms

\$./run.sh

```
File Edit View Search Terminal Help
root@NX: /home/mucio

Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env slave1
root@NX:/home/mucio# docker-machine create slave2
Running pre-create checks...
Creating machine...
(slave2) Copying /root/.docker/machine/cache/boot2docker.iso to /root/.docker/machine/machines/slave2/boot2docker.iso...
(slave2) Creating VirtualBox VM...
(slave2) Creating SSH key...
(slave2) Starting the VM...
(slave2) Check network to re-create if needed...
(slave2) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env slave2
root@NX:/home/mucio# docker-machine ssh master
( ' > ' )
/) TC ( \   Core is distributed with ABSOLUTELY NO WARRANTY.
(/ - _ - _ \)   www.tinycorelinux.net

docker@master:~$ git clone https://github.com/fablopina/monitoring_ms/
Cloning into 'monitoring_ms'...
remote: Enumerating objects: 1244, done.
remote: Total 1244 (delta 0), reused 0 (delta 0), pack-reused 1244
Receiving objects: 100% (1244/1244), 827.36 KiB | 703.00 KiB/s, done.
Resolving deltas: 100% (654/654), done.
Checking connectivity... done.
docker@master:~$ cd monitoring_ms
docker@master:~/monitoring_ms$
```

Nesse momento, a aplicação reclama, devido [propositadamente] não termos criado o cluster usando o swarm; Deixamos este comando para depois, a ideia agora foi somente mostrar que é preciso da infraestrutura do swarm, para implantar as aplicações;

```
docker@master:~$ cd monitoring_ms
docker@master:~/monitoring_ms$ ./run.sh
./run.sh: line 3: python: command not found
Error: No such image: grafana
Sending build context to Docker daemon 31.23kB
Step 1/4 : FROM grafana/grafana:5.1.5
5.1.5: Pulling from grafana/grafana
f2aa67a397c4: Pull complete
e10ea42b45e3: Pull complete
fa279080e10e: Pull complete
Digest: sha256:50ad73ed0631d4f35d178fbaa471a8b353e2e806a08623909817a5ca63302e98
Status: Downloaded newer image for grafana/grafana:5.1.5
--> 6216a7d47b26
Step 2/4 : ADD ./provisioning /etc/grafana/provisioning
--> bb22c81329ae
Step 3/4 : ADD ./config.ini /etc/grafana/config.ini
--> bbb22de67b89
Step 4/4 : ADD ./dashboards /var/lib/grafana/dashboards
--> f6e4f8000099
Successfully built f6e4f8000099
Successfully tagged grafana:latest
this node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again
docker@master:~/monitoring_ms$ git clone https://github.com/fablopina/mucio_ms/
```


Isso não nos impede de clonarmos o projeto de back-end Music_ms;

```
Activities Terminal
root@NX: /home/mucio

File Edit View Search Terminal Help
Resolving deltas: 100% (654/654), done.
Checking connectivity... done.
docker@master:~$ cd monitoring_ms
docker@master:~/monitoring_ms$ ./run.sh
./run.sh: line 3: python: command not found
Error: No such image: grafana
Sending build context to Docker daemon 31.23kB
Step 1/4 : FROM grafana/grafana:5.1.5
5.1.5: Pulling from grafana/grafana
f2aa67a397c4: Pull complete
e10ea42b45e3: Pull complete
fa279080e10e: Pull complete
Digest: sha256:50ad73ed0631d4f35d178fbaa471a8b353e2e806a08623909817a5ca63302e98
Status: Downloaded newer image for grafana/grafana:5.1.5
--> 6216a7d47b26
Step 2/4 : ADD ./provisioning /etc/grafana/provisioning
--> bb22c81329ae
Step 3/4 : ADD ./config.ini /etc/grafana/config.ini
--> bbbc2de67b89
Step 4/4 : ADD ./dashboards /var/lib/grafana/dashboards
--> f6e4f8000099
Successfully built f6e4f8000099
Successfully tagged grafana:latest
this node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again
docker@master:~/monitoring_ms$ git clone https://github.com/fabiopina/music_ms
Cloning into 'music_ms'...
remote: Enumerating objects: 117, done.
remote: Total 117 (delta 0), reused 0 (delta 0), pack-reused 117
Receiving objects: 100% (117/117), 27.13 KiB | 0 bytes/s, done.
Resolving deltas: 100% (61/61), done.
Checking connectivity... done.
docker@master:~/monitoring_ms$ cd music_ms
docker@master:~/monitoring_ms/music_ms$ docker stack deploy -c music_ms/music-example.yml music
open music_ms/music-example.yml: no such file or directory
docker@master:~/monitoring_ms/music_ms$ cd ..
docker@master:~/monitoring_ms$ docker stack deploy -c music_ms/music-example.yml music
this node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again
docker@master:~/monitoring_ms$
```

E executamos o comando abaixo, para criar o cluster, escolhemos o IP 10.0.2.15, inicialmente,

\$ docker swarm init --advertise-addr 10.0.2.15

Na imagem, não tínhamos passado --advertise-addr <IP-MANAGER>, isso ocorre quando o swarm detecta mais de uma interface de rede;

```
Activities Terminal
root@NX: /home/mucio

File Edit View Search Terminal Help
docker@master:~$ docker swarm init
Error response from daemon: could not choose an IP address to advertise since this system has multiple addresses on different interfaces (10.0.2.15 on eth0 and 192.168.99.100 on eth1) - specify one with --advertise-addr
docker@master:~$
```

Consertamos, e após o comando estar correto, nos é retornado um token, para que possamos juntar as máquinas que aqui, chamamos de SLAVE, copiamos o token e saímos do terminal do master;

```
Activities Terminal
root@NX: /home/mucio

File Edit View Search Terminal Help
docker@master:~$ docker swarm init
Error response from daemon: could not choose an IP address to advertise since this system has multiple addresses on different interfaces (10.0.2.15 on eth0 and 192.168.99.100 on eth1) - specify one with --advertise-addr
docker@master:~$ docker swarm init --advertise-addr 10.0.2.15
Swarm initialized: current node (qjqf81t75yr7nro6fe17yay) is now a manager.

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

    docker swarm join --token SWMTKN-1-4usaavsx98mqav71aenx5utp6o3dtdajow5n4dkfqp3cpqib6-ezw9wLkoq3ncb645ak4796vfn 10.0.2.15:2377

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

docker@master:~$ exit
logout
root@NX: /home/mucio#
```

Usamos o comando abaixo, para acessar as outras máquinas

\$ docker-machine ssh slave0

E colamos, o token da máquina master, assim, o nó Slave 0 fará parte do cluster, e tivemos um erro ao tentar juntar o nó, com o IP do master 10.0.2.15;

```
root@nx:/home/mucio# docker-machine ssh slave0
( '~>' )
/) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
(/-__-_\) www.tinycorelinux.net

docker@slave0:~$ docker swarm join --token SWMTKN-1-4usauxs98mqav71aenx5utp6o3dtdajowsn4dkfqp3cpqib6-ezw9wLkoq3ncb645ak4796vfn 10.0.2.15:2377
Error response from daemon: rpc error: code = Unavailable desc = all SubConns are in TransientFailure, latest connection error: connection error: desc = "transport: Error while dialing dial tcp 10.0.2.15:2377: connect: connection refused"
docker@slave0:~$
docker@slave0:~$
docker@slave0:~$
```

Assim, precisamos mudar o IP do cluster para 192.168.99.100, e fizemos com que o nó master deixe/saia o cluster. Antes disso saímos do slave 0, com o comando:

\$ exit

```
docker@slave0:~$
docker@slave0:~$ exit
logout
exit status 1
```

E executamos, para acessar o terminal:

\$ docker-machine ssh master

Para sair do cluster, como nó líder/manager;

\$ docker swarm leave --force

E este comando para criar um cluster com o master como nó líder/manager. Repare, na imagem a mensagem: Node left the swarm, confirmando que o nó saiu do swarm.

\$ docker swarm init --advertise-addr 192.168.99.100

```
exit status 1
root@nx:/home/mucio# docker-machine ssh master
( '~>' )
/) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
(/-__-_\) www.tinycorelinux.net

docker@master:~$ docker swarm leave
Error response from daemon: You are attempting to leave the swarm on a node that is participating as a manager. Removing the last manager erases all current state of the swarm. Use '--force' to ignore this message.
docker@master:~$ docker swarm leave --force
Node left the swarm.
docker@master:~$
```

Copiamos o token do master, e acessamos o terminal do nó slave 0, colando o token nele, fazemos o mesmo juntar-se ao cluster;

Repetimos os procedimentos para juntar as máquinas slave1 e slave2, ao cluster;

Repare na imagem que a mensagem: **This node joined a swarm worker**, informando que o nó juntou-se ao cluster como trabalhador ou escravo, como queira chamar.

```
Activities Terminal
qui 10:07
root@Nx: /home/mucio

docker@master:~$ docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (zhtuict058la0toemdKrsy7k) is now a manager.

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

docker swarm join --token SWMTKN-1-2e2xoiym0ek9m86qmo8d3fvg5spjyz3k3mler61qykxc297h-33zmo1839gzocwse6jqahzvw 192.168.99.100:2377

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

docker@master:~$ exit
logout
root@Nx: /home/mucio# docker-machine ssh slave0
( ) TC ( ) Core is distributed with ABSOLUTELY NO WARRANTY.
( /-_-_- \) www.tinycorelinux.net

docker@slave0:~$ docker swarm join --token SWMTKN-1-2e2xoiym0ek9m86qmo8d3fvg5spjyz3k3mler61qykxc297h-33zmo1839gzocwse6jqahzvw 192.168.99.100:2377
This node joined a swarm as a worker.
docker@slave0:~$

docker@slave1:~$ docker swarm join --token SWMTKN-1-2e2xoiym0ek9m86qmo8d3fvg5spjyz3k3mler61qykxc297h-33zmo1839gzocwse6jqahzvw 192.168.99.100:2377
This node joined a swarm as a worker.
docker@slave1:~$

docker@slave2:~$ docker swarm join --token SWMTKN-1-2e2xoiym0ek9m86qmo8d3fvg5spjyz3k3mler61qykxc297h-33zmo1839gzocwse6jqahzvw 192.168.99.100:2377
This node joined a swarm as a worker.
docker@slave2:~$
```

Tentamos [propositadamente], deployar a aplicação de back-end Music_ms, sem definir a rede de overlay, o que nos retorna o erro:

\$ network “my-network” is declared as external, but could not be found. You need to create a swarm-scoped network before the stack is deployed.

--

Esse nome “my-network” é definido no arquivo music-example.yml.

Então, para que os container’s possam comunicarem-se, uma vez que os processos/serviços de cada container não consegue comunicação externa, é preciso de um canal para isso, no caso, criamos uma overlay e a nomeamos de my-network.

```
root@Nx: /home/mucio# docker-machine ssh master
( ) TC ( ) Core is distributed with ABSOLUTELY NO WARRANTY.
( /-_-_- \) www.tinycorelinux.net

docker@master:~$ cd monitoring_ms
docker@master:~/monitoring_ms$ docker stack deploy -c music_ms/music-example.yml music
network "my-network" is declared as external, but could not be found. You need to create a swarm-scoped network before the stack is deployed
docker@master:~/monitoring_ms$ cd
docker@master:~$ docker network create -d overlay my-network
q4almeutp19cem81mv3qxq5x
docker@master:~$ cd monitoring_ms
```

Uma vez que a infraestrutura, está montada, voltamos ao nó master para fazer o deploy do monitoring_ms e music_ms; Execute estes comandos:

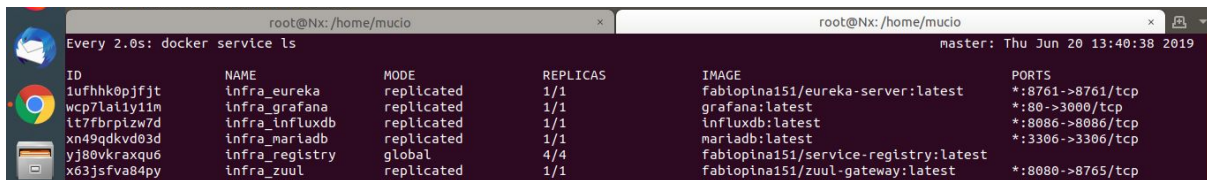
No diretório do monitoring_ms:

\$./run.sh

É possível verificar o deploy com o comando

\$ watch docker service ls

Para sair, é só teclar: **Ctrl + C**



```
root@Nx: /home/mucio
Every 2.0s: docker service ls
```

| ID | NAME | MODE | REPLICAS | IMAGE | PORTS |
|--------------|----------------|------------|----------|--------------------------------------|------------------|
| 1ufhkh0pjft | infra_eureka | replicated | 1/1 | fablopina151/eureka-server:latest | *:8761->8761/tcp |
| wcp7latiy11m | infra_grafana | replicated | 1/1 | grafana:latest | *:80->3000/tcp |
| lt7fbrplzw7d | infra_influxdb | replicated | 1/1 | influxdb:latest | *:8086->8086/tcp |
| xn49qdkvd83d | infra_mariadb | replicated | 1/1 | mariadb:latest | *:3306->3306/tcp |
| yj80vkraxqu6 | infra_registry | global | 4/4 | fablopina151/service-registry:latest | |
| x63jsfva84py | infra_zuul | replicated | 1/1 | fablopina151/zuul-gateway:latest | *:8080->8765/tcp |

Precisamos instalar o docker-compose, a princípio, tentamos rodar o comando abaixo, para subir os serviços, o que nos retornou um erro, não tínhamos instalado o docker-compose:

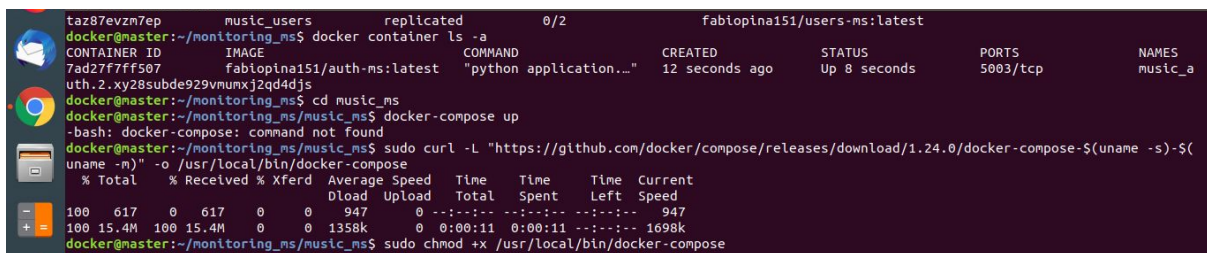
\$ docker-compose up

Assim, prosseguimos com a instalação do docker-compose:

\$ sudo curl -L

"https://github.com/docker/compose/releases/download/1.24.0/docker-compose-\$(uname -s)-\$(uname -m)" -o /usr/local/bin/docker-compose

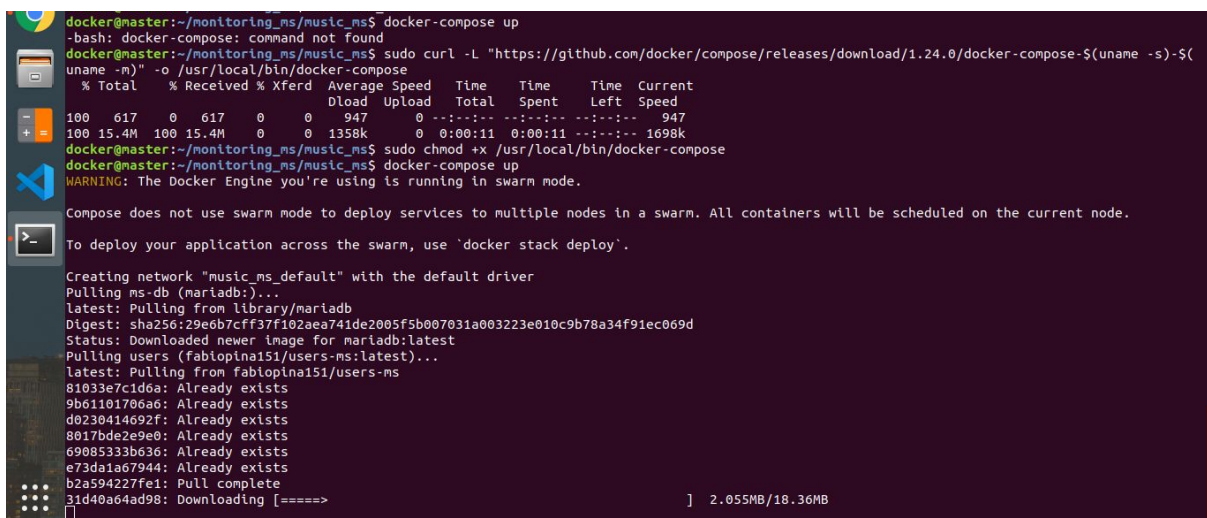
\$ sudo chmod +x /usr/local/bin/docker-compose



```
taz87evzm7ep music_users replicated 0/2 fablopina151/users-ms:latest
docker@master:~/monitoring_ms$ docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7ad27f7ff507 fablopina151/auth-ms:latest "python application..." 12 seconds ago Up 8 seconds 5003/tcp music_a
uth.2.xy28subde929vnmuxj2qd4djs
docker@master:~/monitoring_ms$ cd music_ms
docker@master:~/monitoring_ms/music_ms$ docker-compose up
-bash: docker-compose: command not found
docker@master:~/monitoring_ms/music_ms$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 617 0 617 0 0 947 0 --:--:-- --:--:-- --:--:-- 947
100 15.4M 100 15.4M 0 0 1358k 0 0:00:11 0:00:11 --:--:-- 1698k
docker@master:~/monitoring_ms/music_ms$ sudo chmod +x /usr/local/bin/docker-compose
```

Pós-instalação, acessamos o diretório do music_ms, e executamos novamente:

\$ docker-compose up



```
docker@master:~/monitoring_ms/music_ms$ docker-compose up
-bash: docker-compose: command not found
docker@master:~/monitoring_ms/music_ms$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 617 0 617 0 0 947 0 --:--:-- --:--:-- --:--:-- 947
100 15.4M 100 15.4M 0 0 1358k 0 0:00:11 0:00:11 --:--:-- 1698k
docker@master:~/monitoring_ms/music_ms$ sudo chmod +x /usr/local/bin/docker-compose
docker@master:~/monitoring_ms/music_ms$ docker-compose up
WARNING: The Docker Engine you're using is running in swarm mode.
Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use 'docker stack deploy'.
Creating network "music_ms_default" with the default driver
Pulling ms-db (mariadb):...
latest: Pulling from library/mariadb
Digest: sha256:29e6b7cfff37f102aea741de2005f5b007031a003223e010c9b78a34f91ec069d
Status: Downloaded newer image for mariadb:latest
Pulling users (fablopina151/users-ms:latest)...
latest: Pulling from fablopina151/users-ms
81033e7c1d6a: Already exists
9b61101706a6: Already exists
d0230414692f: Already exists
8017bde2e9e0: Already exists
69085333b636: Already exists
e73da1a67944: Already exists
b2a594227fe1: Pull complete
31d40a64ad98: Downloading [====>] 2.055MB/18.36MB
```

```

root@Nx: /home/mucio
users_1 20/06/2019 01:27:15 [DEBUG] Adding /login...
users_1 20/06/2019 01:27:15 [DEBUG] ... Adding POST -> application.check_login
users_1 20/06/2019 01:27:15 [DEBUG] ... Produces: ['application/json']
users_1 20/06/2019 01:27:15 [DEBUG] ... Produces json
users_1 20/06/2019 01:27:15 [DEBUG] ... Adding produces decorator (<function Operation.__content_type_decorator.<locals>.<lambda> at 0x7f337c98e2f0>)
users_1 20/06/2019 01:27:15 [DEBUG] ... Security: None
users_1 20/06/2019 01:27:15 [DEBUG] ... Adding security decorator (<function security_passthrough at 0x7f337d58f9d8>)
users_1 20/06/2019 01:27:15 [DEBUG] Starting flask HTTP server..
users_1 20/06/2019 01:27:15 [WARNING] * Debugger is active!
users_1 20/06/2019 01:27:15 [INFO] * Debugger PIN: 358-851-110
ms-db_1 2019-06-20 13:37:09 8 [Warning] Aborted connection 8 to db: 'unconnected' user: 'root' host: '172.19.0.6' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:09 9 [Warning] Aborted connection 9 to db: 'Playlists_MS' user: 'root' host: '172.19.0.6' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:10 10 [Warning] Aborted connection 10 to db: 'unconnected' user: 'root' host: '172.19.0.6' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:10 11 [Warning] Aborted connection 11 to db: 'Playlists_MS' user: 'root' host: '172.19.0.6' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:11 12 [Warning] Aborted connection 12 to db: 'unconnected' user: 'root' host: '172.19.0.4' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:11 13 [Warning] Aborted connection 13 to db: 'Songs_MS' user: 'root' host: '172.19.0.4' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:12 14 [Warning] Aborted connection 14 to db: 'unconnected' user: 'root' host: '172.19.0.3' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:12 15 [Warning] Aborted connection 15 to db: 'Users_MS' user: 'root' host: '172.19.0.3' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:14 16 [Warning] Aborted connection 16 to db: 'unconnected' user: 'root' host: '172.19.0.4' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:14 17 [Warning] Aborted connection 17 to db: 'Songs_MS' user: 'root' host: '172.19.0.4' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:15 18 [Warning] Aborted connection 18 to db: 'unconnected' user: 'root' host: '172.19.0.3' (Got timeout reading communication packets)
ms-db_1 2019-06-20 13:37:15 19 [Warning] Aborted connection 19 to db: 'Users_MS' user: 'root' host: '172.19.0.3' (Got timeout reading communication packets)

```

Executando o comando abaixo, vemos os serviços, a quantidade de réplicas do mesmo, bem como portas, status e id.

\$ docker service ls

```

docker@master:~/monitoring_ms$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS              NAMES
docker@master:~/monitoring_ms$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                                  PORTS
oessipmt9srhg      music_aggr          replicated          0/2                  fabiopina151/aggr-ms:latest
o52rr9i5qrwr       music_auth          replicated          0/2                  fabiopina151/auth-ms:latest
yssdxub3196p       music_ms-db         replicated          0/1                  mariadb:latest
t4aasumdqfgg       music_playlists     replicated          0/2                  fabiopina151/playlists-ms:latest
tow84h29wy48       music_songs         replicated          0/2                  fabiopina151/songs-ms:latest
taz87evzm7ep       music_users         replicated          0/2                  fabiopina151/users-ms:latest
docker@master:~/monitoring_ms$

```

E o comando abaixo, verificamos os containers, os quais os serviços foram implantados:

\$ docker ps

```

docker@master:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS              NAMES
MES
d6038cb7fc69       fabiopina151/aggr-ms:latest    "python application..." 4 minutes ago      Up 4 minutes       5004/tcp           mu
s1c_ms_aggr_1      fabiopina151/playlists-ms:latest "python application..." 4 minutes ago      Up 4 minutes       5002/tcp           mu
5a425bc950c5       fabiopina151/auth-ms:latest    "python application..." 4 minutes ago      Up 4 minutes       5003/tcp           mu
6af2b35f0ece       fabiopina151/songs-ms:latest    "python application..." 4 minutes ago      Up 4 minutes       5001/tcp           mu
0bfaf776a31b       fabiopina151/users-ms:latest    "python application..." 4 minutes ago      Up 4 minutes       5000/tcp           mu
49691e4ed1d4       mariadb             "docker-entrypoint.s..." 4 minutes ago      Up 4 minutes       3306/tcp           mu
s1c_ms_db_1        mariadb:latest         "docker-entrypoint.s..." 15 minutes ago     Up 15 minutes      3306/tcp           mu
d5120a2b0f94       fabiopina151/playlists-ms:latest "python application..." 15 minutes ago     Up 15 minutes      5002/tcp           mu
b40f9833e2bc       fabiopina151/auth-ms:latest    "python application..." 15 minutes ago     Up 15 minutes      5003/tcp           mu
7ad27f7f507       fabiopina151/auth-ms:latest    "python application..." 15 minutes ago     Up 15 minutes      5003/tcp           mu
s1c_auth_2.xy28subde929vmumxj2qd4djs
docker@master:~$

```

Com o comando abaixo, listamos os serviços [parecido com o docker service ls]

\$ docker-compose ps

```
docker@master:~$ cd monitoring_ms/music_ms
docker@master:~/monitoring_ms/music_ms$ docker-compose ps

```

| Name | Command | State | Ports |
|----------------------|-----------------------------|-------|----------|
| music_ms_aggr_1 | python application.py | Up | 5004/tcp |
| music_ms_auth_1 | python application.py | Up | 5003/tcp |
| music_ms_ms-db_1 | docker-entrypoint.sh mysqld | Up | 3306/tcp |
| music_ms_playlists_1 | python application.py | Up | 5002/tcp |
| music_ms_songs_1 | python application.py | Up | 5001/tcp |
| music_ms_users_1 | python application.py | Up | 5000/tcp |

```
docker@master:~/monitoring_ms/music_ms$
```

Escolhemos o serviço de playlists, e verificamos em qual IP:Porta ele responde:

\$ docker inspect music_ms_playlists_1

```
docker@master:~/monitoring_ms/music_ms$ docker inspect music_ms_playlists_1
[
  {
    "Id": "5a425bc950c54a06ddb7c720c792532628f9d5774daab3e5ff8cd483d8014fa1",
    "Created": "2019-06-20T13:25:47.015247958Z",
    "Path": "python",
    "Args": [
      "application.py"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 5518,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2019-06-20T13:25:48.456278934Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    }
  }
]
```

Continuação do retornado do comando abaixo,

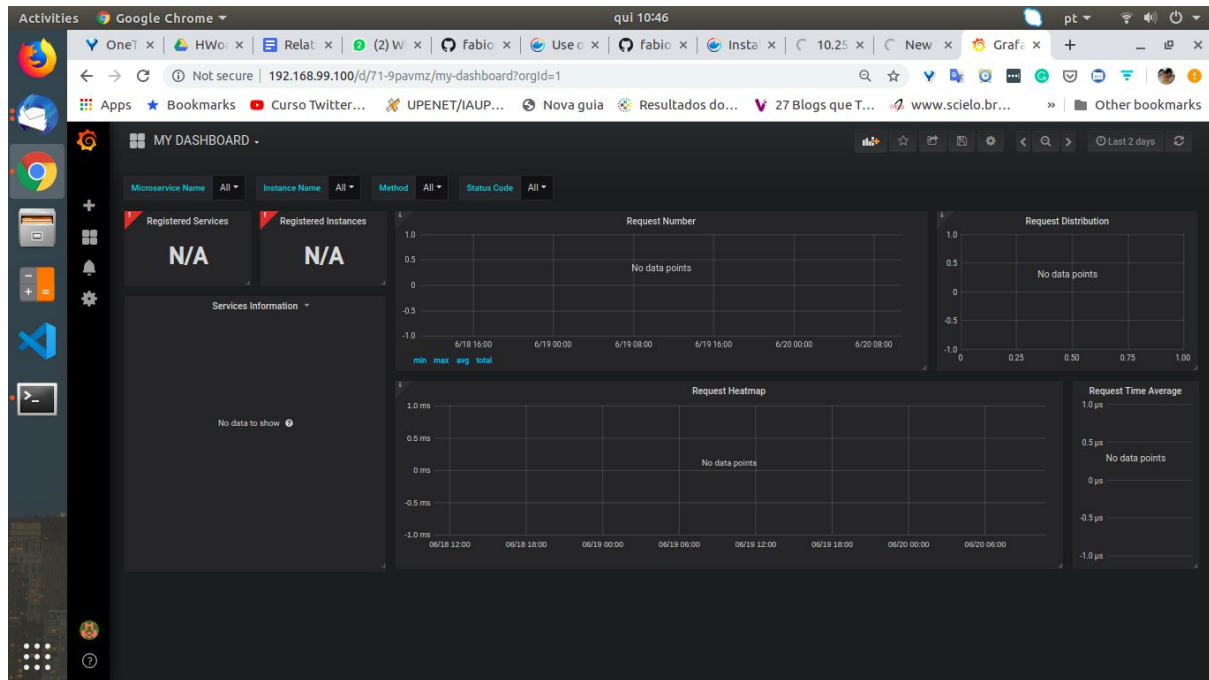
```
root@Nx: /home/mucio
root@Nx: /home/mucio
{
  "Ports": {
    "5002/tcp": null
  },
  "SandboxKey": "/var/run/docker/netns/d0d2dc8728e9",
  "SecondaryIPAddresses": null,
  "SecondaryIPv6Addresses": null,
  "EndpointID": "",
  "Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "IPAddress": "",
  "IPPrefixLen": 0,
  "IPv6Gateway": "",
  "MacAddress": "",
  "Networks": {
    "music_ms default": {
      "IPAMConfig": null,
      "Links": [
        "music_ms_ms-db_1:ms-db",
        "music_ms_ms-db_1:ms-db_1",
        "music_ms_ms-db_1:music_ms_ms-db_1",
        "music_ms_songs_1:music_ms_songs_1",
        "music_ms_songs_1:songs",
        "music_ms_songs_1:songs_1"
      ],
      "Aliases": [
        "5a425bc950c5",
        "playlists"
      ],
      "NetworkID": "45d3178c2948176b61abd6a704bd298d82501342d01e2279141afef636aa8631",
      "EndpointID": "026e4aa83b446b5d672d6b66cfb2f21ef41cd924ec0322bde017d9361247b841",
      "Gateway": "172.19.0.1",
      "IPAddress": "172.19.0.6",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "GlobalIPv6Address": ""
    }
  }
}
```

Nesse caso, o serviço está funcionando em 172.19.0.6:5002, aqui quando tentamos acessar a aplicação pelo browser, não obtivemos sucesso;

Tentamos debugar, pingando entre o master-slaves, e entre os slaves-slaves, obtivemos sucesso, o que nos levou a crer que devido a estarem em VM's houve problema de comunicação ao tentar acessar via navegador. Outro detalhe importante, verificamos que o serviço foi criado no IP acima, e a VM em 192.168.99.100, .101, .102, .103 para o master, slave 0, slave 1, slave 2 respectivamente. Essa é também a possível causa do impedimento do acesso via

navegador. Bem como, usamos o mesmo IP do master, com porta dos serviços, sem sucesso.

Já o grafana, foi possível acessar. Mesmo não conseguindo exibir os dados, conforme imagem abaixo;



4. Considerações Finais:

O artigo reporta problemas relevantes de monitoramento de logs, mas encontramos inconsistências ao tentar replicá-lo, tecnicamente acreditamos que não deveria ser difícil a implantação dos microserviços citados, somado que os autores colocam que é facilmente implantável, e trazem consigo no título do artigo uma abordagem não intrusiva, nos direcionando para implantação de sistemas de monitoramento de logs em sistemas de produção; O que discordamos neste ponto, apesar de saber que artigos geralmente são mais enxutos, as tecnologias usadas no mesmo, deveriam funcionar facilmente uma vez que as aplicações estão prontas, uma vez que também a documentação estava presente no gitHub e a premissa seria abordagem não intrusiva, chamada pelos autores de black-box. Assim com simples comandos, poderíamos fazer a comunicação acontecer, e implantar as aplicações, o que aconteceu parcialmente, ambas funcionaram, porém não houve comunicação, e acreditamos não ser na infraestrutura criada por nós.

Quanto a fazer uso da estrutura do swarm, sentimos dificuldade para entender inicialmente conceitos, e após como seria aplicada no problema, isso nos levou ao método da tentativa e erro. Depois que implantamos, seja usando o docker puro sem VM ou com swarm, chegamos a conclusão que alguma informação relevante ficou faltando, pois solidificamos o entendimento de todo stack pedido

para o funcionamento da mesmas, desde instanciação de máquinas virtuais, docker-swarm/-compose, a acesso via browser.

Ainda assim, consideramos que as questões levantadas pelos autores, apontam para um futuro muito promissor, é preciso de iniciativas como esta, em busca de refinamentos e melhorias para esse problema de monitoramento em microsserviços.

Referências:

Docker. **Create a swarm.** <https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/>. acesso em: 08/06/19

Docker. **Use overlay networks.** <https://docs.docker.com/network/overlay/> . Acesso em: 08/06/19

Docker. **Install Docker Compose.** <https://docs.docker.com/compose/install/>. Acesso em: 08/06/19.

Pina, Fabio (2018). **Music Application Example.** Repositório: https://github.com/fabiopina/music_ms. Acesso em: 08/06/19.

Pina, Fabio (2018). **Nonintrusive Monitoring of Microservice-based Architectures** Repositório: https://github.com/fabiopina/monitoring_ms/. Acesso em: 08/06/19.

Pina, F., Correia, J., Filipe, R., Araujo, F., & Cardroom, J. (2018). **Nonintrusive monitoring of microservice-based systems.** NCA 2018 - 2018 IEEE 17th International Symposium on Network Computing and Applications. <https://doi.org/10.1109/NCA.2018.8548311>

Trucco, Christian, Blog (2018). **Vamos Conhecer o Docker Swarm.** Site: <https://www.concrete.com.br/2018/02/06/vamos-conhecer-o-docker-swarm/>. Acessado em: 05/06/19.