Parcial 2 – Sistemas Distribuidos

Jefry Cardona Chilito

Cód. A00320232v

El primer paso era colocar al pc host en modo manager para la cual se utilizó el siguiente comando:

```
PS D:\2018-1\Sistemas Distribuidos\sd-exam2\A00320232> docker swarm init
Swarm initialized: current node (3zg0gyl05itxtbq1y6dmkgchi) is now a manager.

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

    docker swarm join --token SWMTKN-1-4s3u7pxoby02f0adwxoda68nsyw2ajmnss8uxksixsry0b0r26-6264cslx78gxw4q21xxwejja3 192.168.65.3:2377

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

Este nos ofrece el token para conectar otras máquinas en caso de ser necesarias para realizar las pruebas.

Ahora debemos creara una imagen de fluentd para la cual debemos construir un archivo Dockerfile con el siguiente contenido:

```
FROM fluent/fluentd:v0.12-debian
RUN echo "fluentd"
RUN ["gem", "install", "fluent-plugin-elasticsearch", "--no-rdoc", "--no-ri", "--version", "1.9.2"]
COPY ./conf/fluent.conf /fluentd/etc
```

Ahora debemos iniciar sesión dentro de docker para poder subir esta imagen al repositorio. Luego debemos escribir los siguientes comandos:

```
PS D:\2018-1\Sistemas Distribuidos\sd-exam2\A00320232\fluentd> docker build -t jeffrycardona/fluentdimage:latest .
Sending build context to Docker daemon  4.608kB
Step 1/4 : FROM fluent/fluentd:v0.12-debian
 ---> 3bac1dacdc31
Step 2/4 : RUN echo "fluentd"
 ---> Using cache
 ---> 6b03d8d5d359
Step 3/4 : RUN ["gem", "install", "fluent-plugin-elasticsearch", "--no-rdoc", "--no-ri", "--version", "1.9.2"]
 ---> Using cache
 ---> d12c8530e735
Step 4/4 : COPY ./conf/fluent.conf /fluentd/etc
 ---> Using cache
 ---> c4360d674d3e
Successfully built c4360d674d3e
Successfully tagged jeffrycardona/fluentdimage:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build con
text will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```

```
PS D:\2018-1\Sistemas Distribuidos\sd-exam2\A00320232\fluentd> docker push jeffrycardona/fluentdimage
The push refers to repository [docker.io/jeffrycardona/fluentdimage]
457a75307bcd: Pushed
205c06cc536a: Pushed
984e6626a1a1: Pushed
dcb1878777b2: Pushed
b4e8d93a725a: Pushed
f758be5cb7ea: Pushed
d0cd4349ee21: Pushed
9f1cc85b84e8: Pushed
d626a8ad97a1: Pushed
latest: digest: sha256:fdc88d0149ac2d81882e9c691bbf4cf7a73e3894fc1d678aba642762895093a0 size: 2193
```

Ya con la imagen de fluentd cargada en nuestro repositorio, el siguiente paso es construir el docker-compose.yml que nos permitirá montar los servicios replicados de kabana, whoami, elastic search y fluentd. A continuación sólo se mostrará el principio del archivo, pero si desea consultarlo, lo podrá encontrar adjuntado en la carpeta del repositorio.

```yaml
version: "3"

services:

  whoami:
    image: tutum/hello-world
    networks:
      - net
    ports:
      - "80:80" # puerto de funcionamient
    logging:
      driver: "fluentd" # Logging Driver
      options:
        tag: tutum     # TAG
        fluentd-address: 127.0.0.1:24224
    deploy:
      resources:  #Restricciones de cpu y
        limits:
          cpus: '0.10'
          memory: 20M
        reservations:
          cpus: '0.05'
          memory: 10M

      restart_policy:
          condition: on-failure # Si ocu
          delay: 20s
          max_attempts: 3
          window: 120s
      mode: replicated
      replicas: 4
```

Ahora, para montar los servicios debemos escribir el siguiente comando:

```
PS D:\2018-1\Sistemas Distribuidos\sd-exam2\A00320232> docker stack deploy -c docker-compose.yml exam2
Updating service exam2_vizualizer (id: 3uo7ggkvhfnp3fu4nlue29vjc)
Creating service exam2_fluentd
Creating service exam2_elasticsearch
Creating service exam2_kibana
Creating service exam2_whoami
```

Verificamos que los servicios estén arriba con el comando:

```
PS D:\2018-1\Sistemas Distribuidos\sd-exam2\A00320232> docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
q9950sekuu2m        exam2_elasticsearch replicated          1/1                 elasticsearch:latest                 *:9200->9200/tcp
p5j0bztp6rli        exam2_fluentd       replicated          1/1                 jeffrycardona/myfluentdimage:latest  *:24224->24224/tcp, *:24224->24224/udp
9s69qfx90gc4        exam2_kibana        replicated          1/1                 kibana:latest                        *:5601->5601/tcp
3uo7ggkvhfnp        exam2_vizualizer    replicated          1/1                 dockersamples/visualizer:latest      *:8080->8080/tcp
xp3z26vquj0n        exam2_whoami        replicated          4/4                 tutum/hello-world:latest             *:80->80/tcp
```

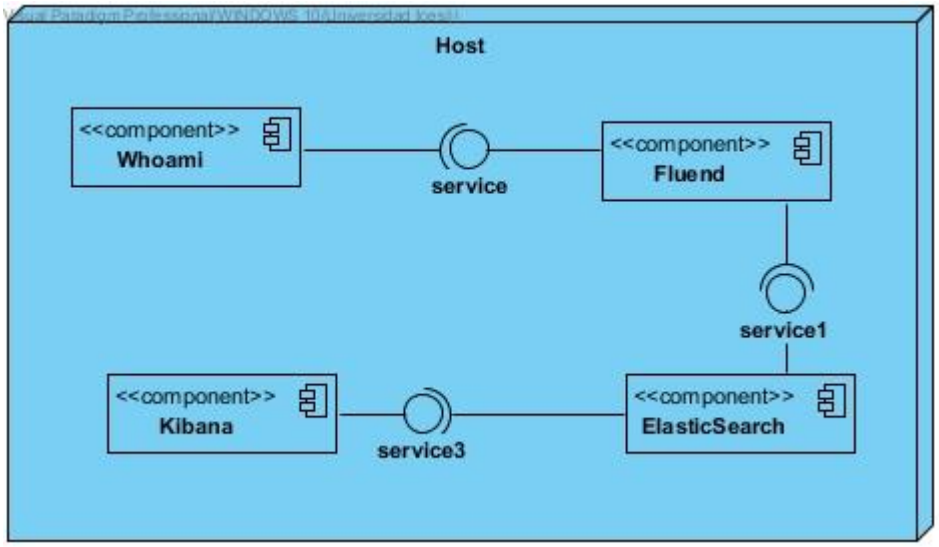
Ahora podemos ver los servicios funcionando

Tutum/hello-world

Hello world!

My hostname is 917e07985e34

Diagrama deployment UML



Problemas:

El parcial no lo logré terminar con éxito ya que en principio no subían los servicios, siempre quedaban faltando servicios, por lo cual no podía verificar los logs. Y cuando subieron todos, aparecía un error con la configuración del patrón de los logs, y así fue un ciclo tratando de solucionarlo, así que pensé que era por Windows y lo ensayé en Ubuntu en los pc´s del laboratorio y tampoco funcionó.