Centro de Procesamiento de Datos

Práctica 3. Docker Swarm: Combinando múltiples máquinas para la ejecución de contenedores Docker.

Objetivo:
Crear un entorno basado en tres máquinas virtuales y evaluar diversas configuraciones de ejecución de contenedores Docker utilizando Docker Swarm para el despliegue de servicios.

Presentar un documento pdf en SWAD→Actividades→ Práctica 3 con la siguiente información:

-(obligatorio): Realizar diversas capturas donde se muestren:
-La creación de las máquinas virtuales con el acceso al laboratorio remoto.
-El inicio del manager de docker swarm. Ej:

```
To add a worker to this swarm, run the following command:
docker swarm join --token SWMTKN-1-
263n2zpxp5f5odhnqdcx67ejh199ig1lzlq62w88nsaue9smwk-eerzcjju0ukrrp22q3vnb7vfw
192.168.99.100:2377
```

- Ejecución del servicio web
- Cuando los 3 nodos están activos
- Cuando se cambia de escala a 2
- Cuando apagamos un nodo activo y sólo ejecuta un nodo,
- y la activación automática del segundo nodo.

(opcional):
-Si se crean las 3 máquinas virtuales utilizando Vagrant (ver práctica 4) en lugar de los laboratorios remotos.

Rúbrica:

Capturas de pantalla	30%
Descripción del trabajo desarrollado	20%
Entrega de apartados opcionales	20%
Entrega en plazo	30%

Desarrollo:
En esta práctica estudiamos cómo distribuir la ejecución de contenedores Docker, creando un escenario con 3 máquinas virtuales.

Utilizaremos el laboratorio remoto:
https://labs.play-with-docker.com/

o bien podemos utilizar el laboratorio de Docker:
https://labs.play-with-k8s.com/
Si estos laboratorios no están disponibles, localmente podemos desplegar 3 máquinas virtuales utilizando Vagrant (ver práctica 4).
(Nota: Docker-machine ya no está actualmente disponible)

II) Evaluando Docker Swarm

Docker Swarm permite distribuir contenedores entre distintas máquinas de forma que pueda distribuirse la ejecución.

Para los siguientes apartados necesitamos 3 máquinas virtuales.

Dentro de la máquina *node1* que hemos creado ejecutamos:

```
docker swarm init --advertise-addr 192.168.99.100
```

Esa IP es la dirección de la subred interna visible entre los nodos.
Obtenemos:

```
docker@m1:~$ docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (nu1lrod1xq93p4ag05i7cdzf3) is now a manager.

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

docker swarm join --token SWMTKN-1-263n2zpxp5f5odhnqdcx67ejh199ig1lzlq62w88nsaue9smwk-eerzcjju0ukrrp22q3vnb7vfw 192.168.99.100:2377

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

Comprobamos los nodos con:

```
docker node ls
```

Creamos un segundo nodo *node2*

Y ejecutamos la orden que nos apareció para añadirlo al swarm:

```
docker swarm join --token SWMTKN-1-263n2zpxp5f5odhnqdcx67ejh199ig1lzlq62w88nsaue9smwk-eerzcjju0ukrrp22q3vnb7vfw 192.168.99.100:2377
```

En el nodo *node1* podemos comprobar que ya aparece el segundo nodo

```
docker node ls
```

Añadimos el tercer nodo *node3* .

III) Creamos un servicio

Vamos a lanzar un servicio web con 3 replicas que se distribuyen entre los nodos.

```
docker service create --name web  --replicas 3 --mount type=bind,src=/etc/hostname,dst=/usr/share/nginx/html/index.html,readonly --publish published=8080,target=80 nginx
```

De esta forma cuando accedemos a cualquiera de los tres nodos.

Con la red *ingress* que utiliza por defecto se define una red que balancea el tráfico y reencamina automáticamente entre los nodos.
Para más información sobre los diversos modelos de red en Docker:
https://docs.docker.com/v17.09/engine/swarm/networking/

Si estamos en nuestro host principal o bien desde cualquier nodo podemos ejecutar:

```
curl http://192.168.99.102:8080
```

Cada vez que lo ejecutamos podemos ver que responde un nodo distinto (m1, m2 o m3).

Podemos comprobar que el servicio ha lanzado 3 contenedores, uno en cada nodo.

```
docker service ps web
```

Reducimos el número de nodos:

```
docker service scale web=2
```

El sistema sigue funcionando y comprobamos el numero de contenedores

Forzamos una parada de uno de los nodos que sigue activo, ej.*node3*

```
docker-machine stop node3
```

o bien en el laboratorio remoto:

```
systemctl stop docker
```

Como sólo está funcionando un contenedor, a los pocos segundos se activará automáticamente el segundo contenedor. (Lo comprobamos de forma periódica).
Si reactivamos el nodo *node3*, podremos ver también que el nodo también aparece

En el laboratorio remoto:

```
systemctl start docker
```

y si reescalamos a 3 contenedores activos aparece automáticamente en el nuevo nodo.

IV) Monitorizar Docker Swarm

En el nodo *node1* desplegamos Swarmprom

```
$ git clone https://github.com/stefanprodan/swarmprom.git
$ cd swarmprom

ADMIN_USER=admin \
ADMIN_PASSWORD=admin \
SLACK_URL=https://hooks.slack.com/services/TOKEN \
SLACK_CHANNEL=devops-alerts \
SLACK_USER=alertmanager \
docker stack deploy -c docker-compose.yml mon
```

Tras esperar a que se inicien los servicios

Accedemos al puerto 3000 podemos acceder a los dashboards:
-Docker Swarm Nodes
-Docker Swarm Services

V) Acceso al registro (log) de un contenerdor

Para acceder:

```
docker logs -f <contenedor>
```

O bien para mostrar los últimos 2 segundos:

```
docker logs -f –until=2s <contenedor>
```

Si queremos acceder de forma interactiva dentro del contenedor:

```
docker exec -it <contenedor> /bin/sh
```