Docker Swarm

Se introduce después de hacer las prácticas de docker y docker-compose. Se hace explicación con los conceptos de stack, service, nodo, etc.

https://docs.docker.com/engine/swarm/key-concepts/

Usaremos un cluster de 3 nodos: 1 manager y dos workers.

- Opción 1: usar multipass para crear los tres nodos e instalar docker en ellos
- Opción 2: crear tres máquinas virtuales con vmware/dropbox e instalar docker en ellos (se puede hacer clone de la primera que creemos).

En el caso de hacerlo con multipass

```
$ multipass launch -v -n nodo1 -m 512MGB  # -m 1GB
$ multipass exec nodo1 -- curl -fsSL https://get.docker.com -o get-docker.sh
$ multipass exec nodo1 -- sudo sh get-docker.sh
```

Nota: no usar la instalación de docker con snap, da problema al desplegar un "stack"

Para ver los nodos creados

```
$ multipass list
Name                    State             IPv4             Image
manager                 Running           10.174.210.86    Ubuntu 20.04 LTS
                                          172.17.0.1
nodo1                   Running           10.174.210.224   Ubuntu 20.04 LTS
                                          172.17.0.1
nodo2                   Running           10.174.210.4     Ubuntu 20.04 LTS
                                          172.17.0.1
$
```

Para iniciar una sesión en un nodo puede usar multipass shell

```
$ multipass shell manager
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-65-generic x86_64)
...
ubuntu@manager:~$
```

Iniciar el "enjambre"

Se iniciará en la máquina manager o leader (puede haber más de un manager). Al iniciarlo nos mostrará el token de invitación para los nodos tipo worker. Una vez iniciado podrá ver que el único nodo existente es el manager y que aparece como activo y con status Leader

```
ubuntu@manager:~$ sudo docker swarm init
...
ubuntu@manager:~$ sudo docker node ls
...
ubuntu@manager:~$
```

Unir nodos de tipo worker al "enjambre"

Es necesario usar el token de invitación. Se muestra al iniciar el swarm, o ejecutando el siguiente comando

```
ubuntu@manager:~$ sudo docker swarm  join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
1w1qn4c8jxjk05q63x1lxp056kxywmfmhz2cfa3cgxjlyk9plt-eyz6k8igxbfeqkdtyqurjnvh6
10.174.210.104:2377

ubuntu@manager:~$
```

En los nodos ejecutamos el comando para hacer join

```
ubuntu@nodo1:~$ sudo docker swarm join --token SWMTKN-1-
1w1qn4c8jxjk05q63x1lxp056kxywmfmhz2cfa3cgxjlyk9plt-eyz6k8igxbfeqkdtyqurjnvh6
10.174.210.104:2377
```

Y en el manager podemos comprobar si el nodo está en el enjambre:

```
ubuntu@manager:~$ sudo docker node ls
```

Creamos un servicio a partir de una imagen que muestra el nombre del equipo

```
ubuntu@manager:~$ sudo docker service create --replicas 1 --name helloworld -p
8080:8080 drhelius/helloworld-node-microservice
```

Sólo se ha creado una réplica, luego el contenedor sólo se despliega en uno de los nodos. Para acceder y probar el servicio usamos `curl` en vez de un navegador

```
ubuntu@manager:~$ curl localhost:8080
Hello World from host "3d9db8be81a8".
ubuntu@manager:~$
```

Observe que accediendo desde cualquiera de los nodos con localhost o con la IP de los nodos se accede al servicio que está en uno de los nodos

```
ubuntu@nodo1:~$ curl 10.174.210.72:8080
Hello World from host "3d9db8be81a8".
ubuntu@nodo1:~$ curl 10.174.210.104:8080
Hello World from host "3d9db8be81a8".
ubuntu@nodo1:~$ curl localhost:8080
Hello World from host "3d9db8be81a8".
ubuntu@nodo1:~$
```

Si añadimos otros nodos

```
ubuntu@manager:~$ sudo docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS
  ENGINE VERSION
qv9nzd55y4noblv1p5sk90xhd *   manager    Ready     Active         Leader
  20.10.3
luvcep1wohoaodisy2in98nqv     nodo1      Ready     Active
 20.10.3
stdh16j1exxe14b385eg43gkx     nodo2      Ready     Active
 20.10.3
pmrljt3d1rux082dg3cvenw3j     nodo3      Ready     Active
19.03.13
ubuntu@manager:~$
```

y **escalamos** el servicio

```
ubuntu@manager:~$ sudo docker service scale helloworld=4
helloworld scaled to 4
overall progress: 4 out of 4 tasks
1/4: running   [==================================================>]
2/4: running   [==================================================>]
3/4: running   [==================================================>]
4/4: running   [==================================================>]
verify: Service converged
ubuntu@manager:~$
```

Ahora las replica se distribuyen en los nodos, y responden *round-robin*

```
ubuntu@nodo1:~$ curl 10.174.210.104:8080
Hello World from host "36a9643a68c4".
ubuntu@nodo1:~$ curl 10.174.210.104:8080
Hello World from host "c2119dd30f01".
ubuntu@nodo1:~$ curl 10.174.210.104:8080
Hello World from host "3d9db8be81a8".
ubuntu@nodo1:~$ curl 10.174.210.104:8080
Hello World from host "ce0c3a7c004b".
```

Para ver la distribución del servicio entre los nodos usamos `docker service ps <nombreServicio>`

```
ubuntu@manager:~$ sudo docker service ps helloworld
ID             NAME               IMAGE
NODE      DESIRED STATE   CURRENT STATE             ERROR     PORTS
ya1xwsc4tzcw   helloworld.1       drhelius/helloworld-node-microservice:latest
nodo1     Running         Running 51 minutes ago
ttt43688evr9   helloworld.2       drhelius/helloworld-node-microservice:latest
nodo3     Running         Running 5 minutes ago
kphdxl3gitt7   helloworld.3       drhelius/helloworld-node-microservice:latest
manager   Running         Running 5 minutes ago
vbkiui0hrl9y   helloworld.4       drhelius/helloworld-node-microservice:latest
nodo2     Running         Running 5 minutes ago
ubuntu@manager:~$
```

Borrar el servicio

Para eliminar el servicio desplegado `docker service rm <nombreServicio>`

```
ubuntu@manager:~$ sudo docker service rm helloworld
helloworld
ubuntu@manager:~$ sudo docker service ls
ID        NAME      MODE      REPLICAS   IMAGE     PORTS
ubuntu@manager:~$
```

Desplegar usando stack y fichero compose

Cuando la aplicación a desplegar incluye varios servicios, volúmenes, networks, etc es recomendable hacer el despliegue desde un fichero que tendrá un formato similar al de docker-compose visto previamente. Este fichero además tendrá etiquetas específica de un despliegue swarm, como el número de replicas.

Es este ejemplo desplegamos el servicio anterior usando un fichero.

```
ubuntu@manager:~$ cat helloworld.yml
version: '3.7'

services:
  helloworld:
      image: drhelius/helloworld-node-microservice
      ports:
        - "8080:8080"
      deploy:
        replicas: 2
ubuntu@manager:~$
```

Y para desplegar esta pila o stack `docker stack deploy -c <fichero.yml>  <nombreStack>`

```
ubuntu@manager:~$ sudo docker stack deploy -c helloworld.yml demo
Creating network demo_default
Creating service demo_helloworld
ubuntu@manager:~$
```

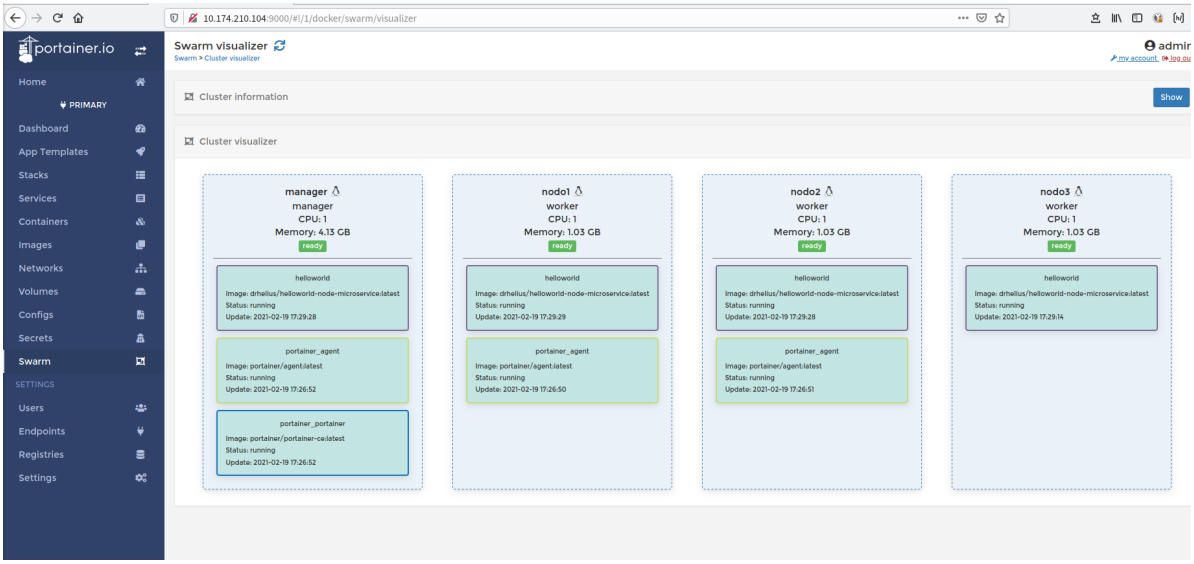
Herramienta gráfica: portainer

Si deseamos poder visualizar y trabajar con el cluster de forma gráfica una opción es usar portainer:
https://www.portainer.io/

Esta aplicación se despliega sobre el cluster, de forma que para utilizarla descargamos el fichero para desplegar el stack con curl .

```
ubuntu@manager:~$ curl -L https://downloads.portainer.io/portainer-agent-stack.yml -o portainer-agent-stack.yml
ubuntu@manager:~$ sudo docker stack deploy -c portainer-agent-stack.yml portainer
ubuntu@manager:~$ sudo docker service ls
ID             NAME                   MODE         REPLICAS   IMAGE                                          PORTS
l6149hp801zm   helloworld             replicated   4/4        drhelius/helloworld-node-microservice:latest   *:8080->8080/tcp
4hfl1cjrbzir   portainer_agent        global       3/3        portainer/agent:latest
shdl0bubedhc   portainer_portainer    replicated   1/1        portainer/portainer-ce:latest                  *:8000->8000/tcp, *:9000->9000/tcp
ubuntu@manager:~$
```

Acceda a la ip del manager usando el puerto 9000



En el primer acceso deberá crear un usuario y especificar su clave. La captura previa se obtiene desde "*Dashboard*", enlace *"Cluster visualizer"*.