Docker Swarm

- Se introduce despues de ver docker-compose
- Buscar explicación con los conceptos de stack, service, nodo, etc.
- Usaremos un cluster de 3 nodos: 1 manager y dos workers.
    - Opción 1: usar multipass de ubuntu, e instalar docker en ellos

```
# crea_multipass.sh
multipass launch -v -n $1 -m 1GB
multipass exec -v $1  sudo snap install docker
# o usar procedimiento en
# https://docs.docker.com/engine/install/ubuntu/
```

    - 2ª opción: 3 máquinas virtuales con vmware/dropbox

Iniciar el "enjambre"

```
manager$ sudo docker swarm init
...
manager$ sudo docker node ls
```

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:

```
manager$ sudo docker node ls
```

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

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

Para acceder al servicio usamos curl

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

Observe que accediendo desde cualquier de nodos con localhost o con la IP de cualquiera de los nodos se acceder 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 y escalamos el servicio

```
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:~$
```

Si 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 respondenen 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".
```

```
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
4tqwnpu87f9j    \_ helloworld.1   drhelius/helloworld-node-microservice:latest
manager   Shutdown        Shutdown 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

```
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 formato compose

```
manager$ cat helloworld.yml
version: '3.7'

services:
  helloworld:
      image: drhelius/helloworld-node-microservice
      ports:
        - "8080:8080"
      deploy:
        replicas: 2
manager$ sudo docker stack deploy -c helloworld.yml demo
Creating network demo_default
Creating service demo_helloworld
manager$
```

Y para desplegarlo

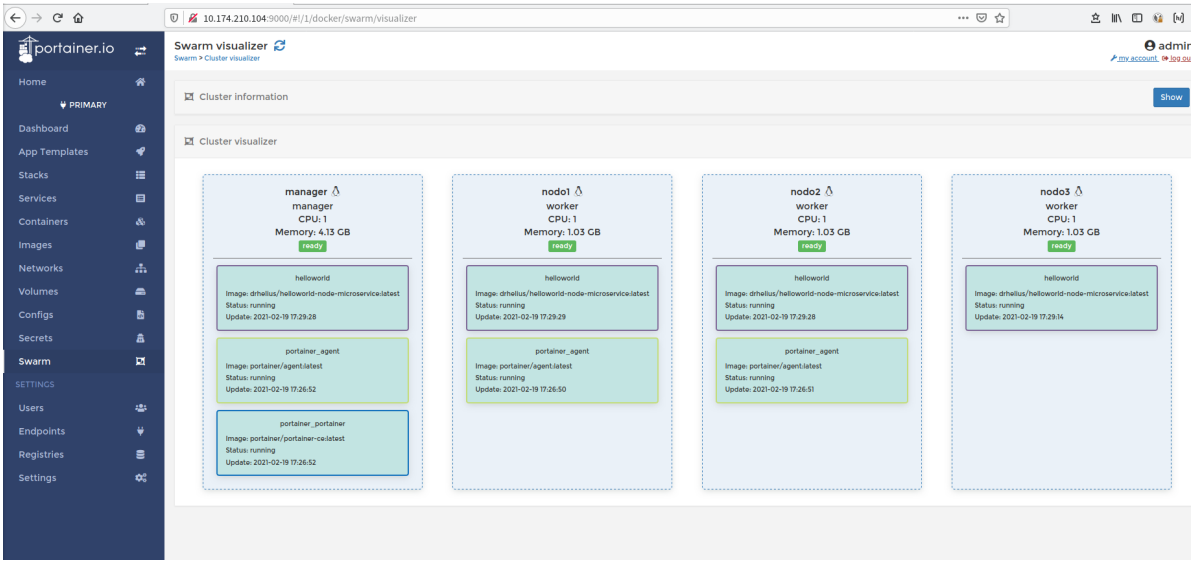
```
manager$ sudo docker stack deploy -c helloworld.yml demo
```

Herramienta gráfica: portainer

https://www.portainer.io/

```
manager$ sudo curl -L https://downloads.portainer.io/portainer-agent-stack.yml -o
portainer-agent-stack.yml
manager$ sudo docker stack deploy --compose-file=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



Pendiente para otro día

Set up a Docker registry

https://docs.docker.com/engine/swarm/stack-deploy/

Se puede usar docker hub o mantener uno propio:

```
$ docker service create --name registry --publish published=5000,target=5000
registry:2
$ docker service ls
$ curl http://localhost:5000/v2/
```

container con app que muestra IP y hostname

```
$ docker run -p 5000:5000 jcdemo/flaskapp
```

Flask aPP

```
[manager] $ docker service create --name registry --publish "5000:5000"
registry:2
[manager] $ docker network create --driver overlay --subnet 10.24.90.0/24 mynet
$ docker network inspect mynet
```

```
$ mkdir flask-demo
$ cd flask-demo
$ cat requirements.txt
flask
$ cat app.py
from flask import Flask
import os
import uuid

app = Flask(__name__)

@app.route('/')
def index():
    hostname = os.uname()[1]
    randomid = uuid.uuid4()
    return 'Container Hostname: ' + hostname + ' , ' + 'UUID: ' + str(randomid)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5098)
$ cat Dockerfile
FROM python:3.4-alpine
ADD . /app
```

```
WORKDIR /app
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
$ cat docker-compose.yml
version: '3'

services:
  web:
    image: master:5000/flask-app
    build: .
    ports:
      - "80:5098"

networks:
  default:
    external:
      name: mynet

### Testing locally on the host:

$ docker-compose up
 # do some testing
$ docker-compose down

$ docker-compose build
$ docker-compose push


[manager] $ docker service create --name flask-demo --network mynet --update-
delay 5s --publish 80:5098 --replicas 1 master:5000/flask-app
```

We can get the Endpoint Port info by using `inspect` and using the --format parameter to filter the output:

```
[manager] $ docker service inspect --format="{{json .Endpoint.Ports}}" my-web  |
python -m json.tool
[manager] $ docker service inspect --format="{{json .Endpoint.VirtualIPs}}" my-
web  | python -m json.tool
```

TODO

- Network
- secrets
- config
-

Enlaces

- https://docs.docker.com/engine/swarm/stack-deploy/
    - https://www.ionos.es/digitalguide/servidores/know-how/docker-compose-y-swarm-gestion-multicontenedor/

- Docker Swarm Visualizer
  https://github.com/dockersamples/docker-swarm-visualizer
- https://enmilocalfunciona.io/cluster-de-docker-con-swarm-mode/
- https://github.com/docker-archive/orchestration-workshop
  - https://container.training/swarm-selfpaced.yml.html#138
- Docker Swarm Tutorial
  https://rominirani.com/docker-swarm-tutorial-b67470cf8872
  - https://drive.google.com/file/d/0B49Q1BZG4BXcQ0RIZlg4RHFZenc/view
- Python app show id
  https://sysadmins.co.za/docker-swarm-getting-started-with-a-3-node-docker-swarm-cluster-with-a-scalable-app/
- Orchestration at Scale with Docker
  https://github.com/docker-archive/orchestration-workshop
- Deploy a stack to a swarm
  https://docs.docker.com/engine/swarm/stack-deploy/
- Docker Swarm: Getting Started with a 3 Node Docker Swarm Cluster with a Scalable App
  https://sysadmins.co.za/docker-swarm-getting-started-with-a-3-node-docker-swarm-cluster-with-a-scalable-app/