

Cluster: Docker Swarm

- [1. Introducción](#)
- [2. Docker swarm en máquinas virtuales](#)
- [3. Iniciar el "enjambre"](#)
- [4. Unir nodos de worker al "enjambre"](#)
- [5. Borrar el servicio](#)
- [6. Desplegar usando stack y fichero compose](#)
- [7. Herramienta gráfica: portainer](#)
- [8. Anexo Docker swarm con multipass](#)

1. Introducción

Se introduce este tema después de hacer las prácticas de docker y docker compose. Puede consultar esta referencia con una explicación de los conceptos de stack, service, nodo, etc:

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

Para esta práctica usaremos un cluster de 3 nodos: 1 manager y dos workers. Tenemos dos posibilidades:

- Opción 1: crear tres máquinas virtuales (vmware o dropbox) e instalar docker en ellas (se puede hacer un "linked clone" de la que estamos usando en clase xdebian que ya tiene instalado docker).
- Opción 2: usar la utilidad multipass para crear los tres nodos e instalar docker en ellos (ver anexo).

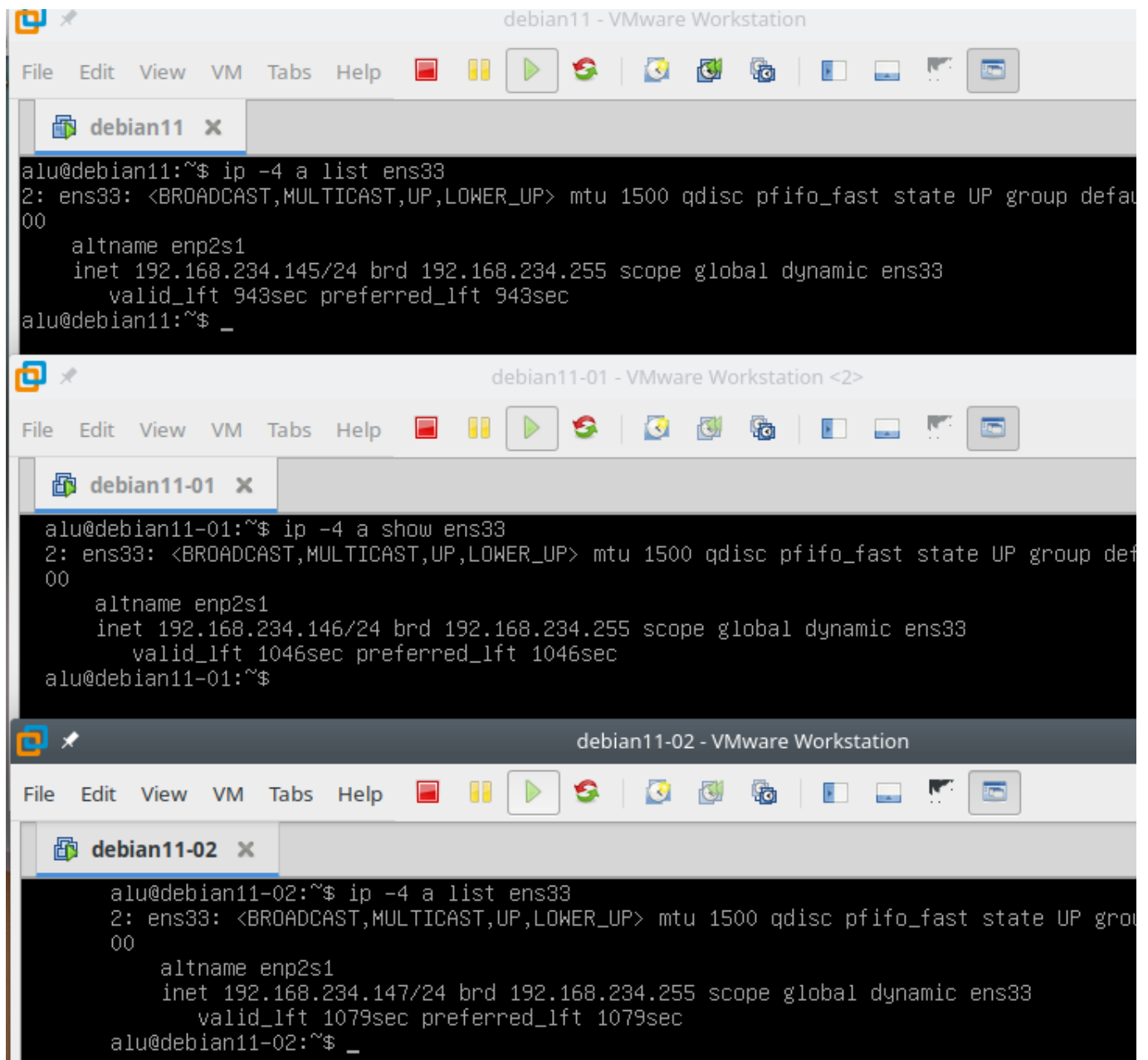
Optamos por la primera de ellas.

2. Docker swarm en máquinas virtuales

Usaremos una máquina virtual ya preparadas con debian, Versión server sin interfaz gráfico. Llevan ya instaladas docker y un servidor ssh.

Como necesitaremos tres máquinas virtuales lo más sencillo es instalar una y clonar las otras dos. Al clonar especificar en el tipo de clone *linked clone*.

Al iniciar las máquinas modificamos el hostname (`sudo hostname <nuevo_nombre>`) en las clonadas para que sea más sencillo identificarlas:



The image displays three stacked screenshots of a VMware Workstation interface, each showing a terminal window for a different Debian 11 virtual machine. The top window, titled 'debian11 - VMware Workstation', shows the output of the command 'ip -4 a list ens33', displaying details for the network interface ens33 with IP 192.168.234.145. The middle window, titled 'debian11-01 - VMware Workstation <2>', shows the output of 'ip -4 a show ens33', displaying details for the same interface with IP 192.168.234.146. The bottom window, titled 'debian11-02 - VMware Workstation', shows the output of 'ip -4 a list ens33', displaying details for the same interface with IP 192.168.234.147. Each terminal window has a menu bar (File, Edit, View, VM, Tabs, Help) and a toolbar with various icons.

```
alu@debian11:~$ ip -4 a list ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
    altname enp2s1
    inet 192.168.234.145/24 brd 192.168.234.255 scope global dynamic ens33
        valid_lft 943sec preferred_lft 943sec
alu@debian11:~$ _
```

```
alu@debian11-01:~$ ip -4 a show ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
    altname enp2s1
    inet 192.168.234.146/24 brd 192.168.234.255 scope global dynamic ens33
        valid_lft 1046sec preferred_lft 1046sec
alu@debian11-01:~$
```

```
alu@debian11-02:~$ ip -4 a list ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
    altname enp2s1
    inet 192.168.234.147/24 brd 192.168.234.255 scope global dynamic ens33
        valid_lft 1079sec preferred_lft 1079sec
alu@debian11-02:~$ _
```

Accederemos con ssh a las máquinas (nos va a permitir copia y pegar, y además refleja mejor una situación real de trabajo). Tras conectar iniciamos el manager y unimos los workers:

```
t8s4i3dhd0a7supzx7xm727ia    debian11-02    Ready    Active
alu@debian11:~$ sudo docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1ynqpz3vz53jsy45awc1ywqhdX0urwk996lv1k22i3yh8

alu@debian11:~$ docker node ls
ID                                HOSTNAME        STATUS        AVAILABILITY    MANAGER STATUS
yf0o08ybnjhcyo3or4id1rjd *      debian11        Ready         Active           Leader
ilwa7bwxh3019sb4dece2m9rm       debian11-01     Ready         Active
t8s4i3dhd0a7supzx7xm727ia       debian11-02     Ready         Active
alu@debian11:~$
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 30 09:29:55 2022
alu@debian11-01:~$ docker swarm join --token SWMTKN-1-1ynqpz3vz53jsy45awc1ywqhdX0urw
This node joined a swarm as a worker.
alu@debian11-01:~$
```

```
Warning: Permanently added '192.168.234.147' (ECDSA) to the list of known hosts.
alu@192.168.234.147's password:
Linux debian11-02 5.10.0-13-amd64 #1 SMP Debian 5.10.106-1 (2022-03-17) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 30 09:29:05 2022
alu@debian11-02:~$
alu@debian11-02:~$ docker swarm join --token SWMTKN-1-1ynqpz3vz53jsy45awc1ywqhdX0urw
This node joined a swarm as a worker.
alu@debian11-02:~$
```

Y ya podemos hacer un despliegue del servicio hello, del portainer, etc.

```

alu@debian11-01: ~
alu@debian11: ~ 211x12
ID                HOSTNAME        STATUS        AVAILABILITY    MANAGER STATUS    ENGINE VERSION
yf0o08ybnjhcyo3or4idirjd *  debian11        Ready         Active           Leader             20.10.14
ilwa7bwxh3019sb4dece2m9rm    debian11-01     Ready         Active            20.10.14
t8s4i3dhd0a7supzx7xm727ia    debian11-02     Ready         Active            20.10.14
alu@debian11:~$ docker service create --replicas 3 --name helloworld -p 8080:8080 drhelius/helloworld-node-microservice
kzw926b0fbwoq62o0ntmuh5zs
overall progress: 3 out of 3 tasks
1/3: running  [=====>]
2/3: running  [=====>]
3/3: running  [=====>]
verify: Service converged
alu@debian11:~$ 
alu@debian11-01: ~ 211x8
This node joined a swarm as a worker.
alu@debian11-01:~$ curl 127.0.0.1:8080
Hello World from host "c51c96c50132".
alu@debian11-01:~$ curl 127.0.0.1:8080
Hello World from host "dfe4d86d3c47".
alu@debian11-01:~$ curl 127.0.0.1:8080
Hello World from host "1fe1058c9f49".
alu@debian11-01:~$ 
alu@debian11-02: ~ 211x24

```

Y ahora lo vemos paso a paso.

3. Iniciar el "enjambre"

Primero nos conectamos por ssh al manager:

```

clases@vm:~$ ssh alu@192.168.125.136
alu@192.168.125.136's password:
Linux manager 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-
09-29) x86_64

```

```

...
clases@vm:~$
...

```

El cluster se iniciará en la máquina manager con ``docker swarm init``. Al iniciarlo nos mostrará el token de invitación para los nodos tipo worker. Una vez iniciado podrá comprobar (``docker node ls``) que el único nodo existente es el manager y que aparece como activo y con status Leader

```

...

```

```

alu@manager:~$ docker swarm init
Swarm initialized: current node (szbjfji2mbaiqif01al7u0qjg) is now a
manager.

```

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

```

docker swarm join --token SWMTKN-1-
0fuvq18xzr6we68t2n584wernrf1nm0n0e3gz90cvaj4qm1a4z-
8lefbc8r8qxwa7zhm2ulytpv0 192.168.125.136:2377

```

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

```
alu@manager:~$ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER
szbzjfi2mbaiqif01al7u0qjg *	manager	Ready	Active	Leader

```
24.0.6
alu@manager:~$
```

4. Unir nodos de worker al "enjambre"

Es necesario usar el token de invitación para unirse al cluster. Se muestra al iniciar el swarm en el manager (o ejecutando el comando `docker swarm join-token worker` en el manager).

Vamos a añadir un worker al cluster:

```
alu@worker-01:~$ docker swarm join --token SWMTKN-1-
0fuvq18xZR6we68t2n584wernrf1nm0n0e3gz90cvaj4qm1a4z-
8lefbc8r8qxa7zhm2ulytpv0 192.168.125.136:2377
This node joined a swarm as a worker.
alu@worker-01:~$
```

En el manager podemos comprobar el nodo añadido:

```
alu@manager:~$ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER
9nhxurspobgirwh5m1d5h0fvf *	manager	Ready	Active	Leader
zm7acf20ysqblhalyeicrse6w	worker-01	Ready	Active	

```
24.0.6
alu@manager:~$
```

Para probar el cluster **crearemos en el manager** un servicio usando la imagen `drhelius/helloworld-node-microservice`. Esta imagen muestra el nombre (id del contenedor) del equipo.

Observe cómo se crea el servicio con una única replica:

```
alu@manager:~$ docker service create --replicas 1 --name helloworld -p
8080:8080 drhelius/helloworld-node-microservice
zzcmp70yivslrzdghqyxhjblyl
overall progress: 0 out of 1 tasks
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
alu@manager:~$
```

Para probar el servicio usamos `curl` (en vez de un navegador) desde el manager:

```
alu@manager:~$ curl 127.0.0.1:8080
Hello World from host "8e088d6016c9".
alu@manager:~$
```

O desde el worker-01

```
alu@worker-01:~$ curl 127.0.0.1:8080
Hello World from host "8e088d6016c9".
alu@worker-01:~$
```

Podemos acceder también desde cualquiera de los dos nodos con la IP de los nodos, y en los dos casos se accede al servicio (aunque el contenedor esté sólo en uno de los nodos):

```
alu@manager:~$ curl 192.168.125.136:8080
Hello World from host "692f6380f27b".
alu@manager:~$ curl 192.168.125.141:8080
Hello World from host "692f6380f27b".
alu@manager:~$
```

Añadimos otro nodo al enjambre:

```
alu@worker-02:~$ docker swarm join --token SWMTKN-1-
0fuvq18x zr6we68t2n584wernrf1nm0n0e3gz90cvaj4qm1a4z-
8lefbc8r8qxwa7zhm2ulytpv0 192.168.125.136:2377
This node joined a swarm as a worker.
alu@worker-02:~$
```

Comprobamos en el manager

```
alu@manager:~$ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER
9nhxurspobgirwh5m1d5h0fvf *	manager	Ready	Active	Leader
zm7acf20ysqblhalyeicrse6w	worker-01	Ready	Active	
s53vme3dp6tr2jvjmf88avpt	worker-02	Ready	Active	

```
24.0.6
24.0.6
24.0.6
alu@manager:~$
```

Ahora **escalamos** el servicio ya iniciado **helloworld** a 4 replicas o contenedores

```
alu@manager:~$ 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
alu@manager:~$
```

Puede observar que las replicas se distribuyen en los nodos, y responden a las peticiones según un algoritmo *round-robin*

```
alu@manager:~$ curl 127.0.0.1:8080
Hello World from host "eb5da9711349".
alu@manager:~$ curl 127.0.0.1:8080
Hello World from host "acb9bc2b68ba".
alu@manager:~$ curl 127.0.0.1:8080
Hello World from host "1f4f0b7c2822".
alu@manager:~$ curl 127.0.0.1:8080
Hello World from host "8e088d6016c9".
alu@manager:~$ curl 127.0.0.1:8080
Hello World from host "eb5da9711349".
alu@manager:~$
```

Hasta ahora hemos hecho las pruebas desde los nodos, pero podemos acceder al servicio desde fuera usando la IP de cualquiera de los nodos del cluster:

```
clases@vm:~$ curl 192.168.125.141:8080
Hello World from host "eb5da9711349".
clases@vm:~$ curl 192.168.125.141:8080
Hello World from host "1f4f0b7c2822".
clases@vm:~$ curl 192.168.125.141:8080
Hello World from host "acb9bc2b68ba".
clases@vm:~$ curl 192.168.125.141:8080
Hello World from host "8e088d6016c9".
clases@vm:~$ curl 192.168.125.141:8080
Hello World from host "eb5da9711349".
clases@vm:~$
```

Puede comprobar la distribución del servicio entre los nodos con **docker service ps** **<nombreServicio>**

```
alu@manager:~$ docker service ps helloworld
ID                NAME                IMAGE
NODE             DESIRED STATE      CURRENT STATE      ERROR      PORTS
z1phnld17fcz     helloworld.1       drhelius/helloworld-node-microservice:latest
manager          Running            Running 6 minutes ago
pa5hp3zsudf3     helloworld.2       drhelius/helloworld-node-microservice:latest
worker-02        Running            Running about a minute ago
pppd9mdjj06c     helloworld.3       drhelius/helloworld-node-microservice:latest
worker-01        Running            Running about a minute ago
or8ythvqgb8q     helloworld.4       drhelius/helloworld-node-microservice:latest
worker-01        Running            Running about a minute ago
alu@manager:~$
```

5. Borrar el servicio

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

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

6. 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 docker-compose visto previamente. Este fichero además tendrá etiquetas específicas de un despliegue swarm como por ejemplo el número de replicas.

Es este ejemplo desplegamos el servicio anterior usando el fichero `helloworld.yml` Observe los campos `deploy` y `replicas`:

```
version: '3.7'

services:
  helloworld:
    image: drhelius/helloworld-node-microservice
    ports:
      - "8080:8080"
    deploy:
      replicas: 2
```

Para copiarlo en el manager usamos el comando `scp`:


```

726
clases@vm:~$ ls
alu@192.168.125.136 helloworld.yml portainer-agent-stack.yml
clases@vm:~$ scp portainer-agent-stack.yml alu@192.168.125.136:
alu@192.168.125.136's password:
portainer-agent-stack.yml      100% 791      1.6MB/s   00:00
clases@vm:~$

```

Y usamos el fichero desde el manager para el deploy:

```

alu@manager:~$ ls
helloworld.yml portainer-agent-stack.yml
alu@manager:~$ docker stack deploy -c portainer-agent-stack.yml portainer
Creating network portainer_agent_network
Creating service portainer_agent
Creating service portainer_portainer
alu@manager:~$

```

Y si vemos el despliegue se observan tantos agentes como nodos, y un gestor:

```

alu@manager:~$ docker service ls
ID                NAME                MODE                REPLICAS    IMAGE
PORTS
d3ispnqyqm6s     portainer_agent     global              3/3          portainer/agent:2.11.1
dynd71w6qjxn     portainer_portainer replicated          1/1          portainer/portainer-ce:2.11.1
*:8000->8000/tcp, *:9000->9000/tcp, *:9443->9443/tcp
alu@manager:~$

```

Puede comprobar la distribución:

```

alu@manager:~$ docker service ps portainer_portainer
ID                NAME                IMAGE                NODE
DESIRED STATE    CURRENT STATE      ERROR               PORTS
ad94lkp4oqjg     portainer_portainer.1 portainer/portainer-ce:2.11.1
manager Running          Running 3 minutes ago
alu@manager:~$ docker service ps portainer_agent
ID                NAME                IMAGE                NODE
DESIRED STATE    CURRENT STATE      ERROR               PORTS
d2wrk62bcgwj     portainer_agent.9nhxurspobgirwh5m1d5h0fvf
portainer/agent:2.11.1 manager Running          Running 3 minutes ago
ur56o8xcj9ja     portainer_agent.s53vme3dp6tr2jvjmfr88avpt
portainer/agent:2.11.1 worker-02 Running          Running 3 minutes ago
ke16b2mwa9oy     portainer_agent.zm7acf20ysqblhalyeicrse6w

```

```
portainer/agent:2.11.1    worker-01    Running    Running 3 minutes ago
alu@manager:~$
```

Acceda ahora usando un navegador a la IP del manager (o cualquier otra del enjambre) usando el puerto 9000. En el primer acceso deberá crear un usuario y su clave. A continuación puede hacer un deploy de un servicio. En este caso se ha realizado el despliegue de `helloworld` desde portainer usando el fichero `helloworld.yml`.

La captura se obtiene desde "Dashboard", enlace "Cluster visualizer":

8. Anexo Docker swarm con multipass

Aviso: Este anexo no ha sido probado este curso.

En el caso de hacerlo con la herramienta `multipass` en vez de con máquinas virtuales

```
$ 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: evitar la instalación de docker con snap, parece que 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 el comando `multipass shell`

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