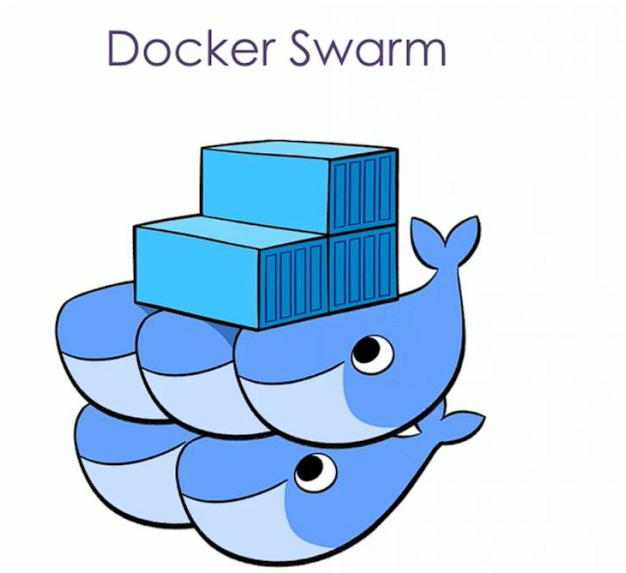
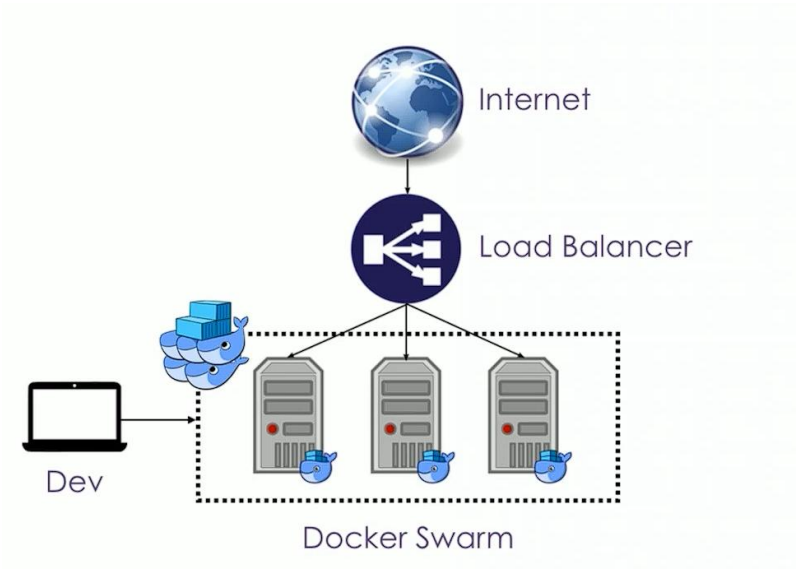
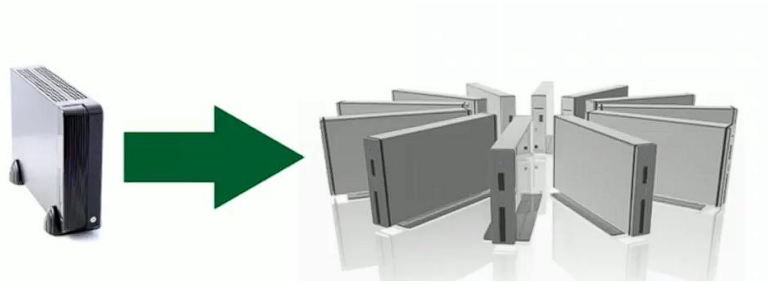
Docker Swarm



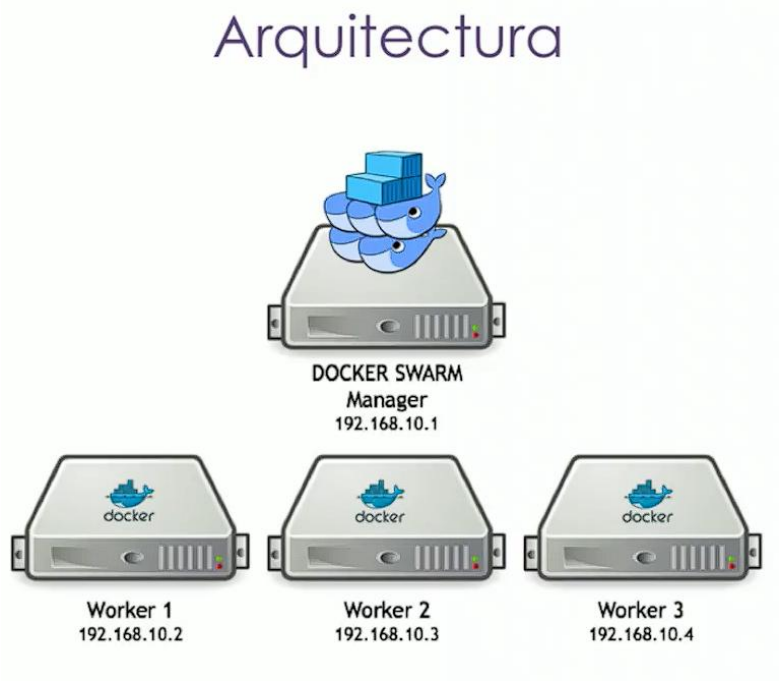


Escalabilidad

La escalabilidad horizontal es el enfoque mas utilizado hoy en día en lugar del escalado vertical, esto va muy de la mano con la disponibilidad.



Arquitectura de Docker Swarm



Dos tipos de máquinas:

- Manager nodes: deciden donde se corren los contenedores, como se comunican, donde hay recursos, vigila el estado de los contenedores para garantizar disponibilidad.
- Worker nodes: son más que los manager, aquí se van a ejecutar contenedores, son el núcleo de la aplicación, la parte productiva. Todos deben tener docker daemon, idealmente la misma versión. Deben estar todos en la misma red.
- Debemos saber si nuestras aplicaciones están listas para correr con docker swarm

Preparación de aplicaciones para Docker Swarm

12 factores de la aplicación:

- **Codebase**, tu código debe estar en un repositorio y este debería estar en relación de 1 a 1 entre código y repositorio.
- **Depencias**, deberían venir empaquetadas con la aplicación.
- **La configuración**, debe ser parte de tu aplicación.
- **Backing Service**, como bases de datos, deben ser tratados como servicios externos a la aplicación.
- **Build**, **Release**, **Run**. estas tres fases deben estar separadas en tu aplicación.
- **Processes**, la ejecución de tu aplicación no puede depender de que exista cierto estado, todo proceso lo debe realizar de forma atómica, stateless.
- **Port binding**, la aplicación debe poder exponerse a si misma, sin intermediarios.

- **Concurrencia**, que la aplicación pueda correr con múltiples instancia en paralelo.
- **Disposability**, la aplicación debe estar diseñada para ser fácilmente destruible e iniciar rápidamente.
- **Dev/prod parity**, lograr que entorno de desarrollo, sea los más parecido a producción.
- **Logs**, Todos los logs de la aplicación deben tratarse como un flujo de device.
- **Admin Process**, la aplicación debe poder ejecutar como procesos independientes.

Uso de Docker Swarm

```
docker swarm init
```

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
g10n92@Giancarlo:~$ docker swarm init
Swarm initialized: current node (db71uwl889g38kjzkd1e1g63p) is now a manager.

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

    docker swarm join --token SWMTKN-1-163cbqijuw62msg49ms0zcpwjt08s7ccsbn5ejt5wo9mppll5t-blvnnlqfgwp9303s2z3g2pl6b 192.
168.65.3:2377

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

Agregar un manager a docker

```
docker swarm join-token manager
```

Ver los nodes de docker

```
docker node ls
```

```
PS C:\Users\g10n_> docker node ls
ID                            HOSTNAME         STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
db71uwl889g38kjzkd1e1g63p *   docker-desktop   Ready     Active         Leader           28.4.0
PS C:\Users\g10n_>
```

Inspeccionar node:

```
docker node inspect self
```

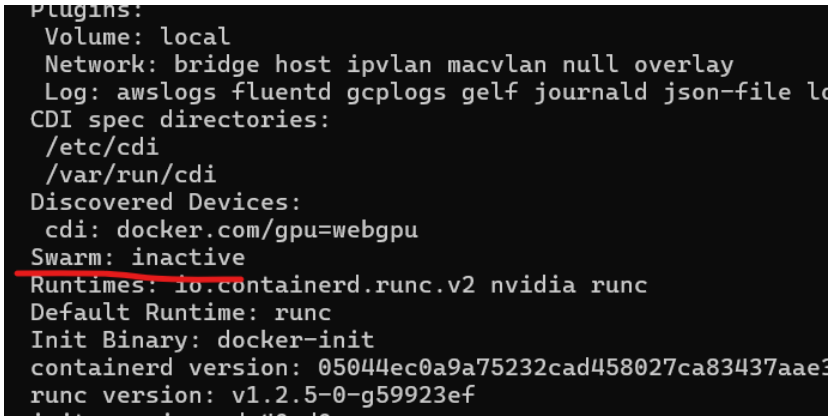
```
PS C:\Users\g10n_> docker node inspect self
[
    {
        "ID": "db71uwl889g38kjzkd1e1g63p",
        "Version": {
            "Index": 9
        },
        "CreatedAt": "2025-09-19T13:59:30.440642902Z",
        "UpdatedAt": "2025-09-19T13:59:30.953077877Z",
        "Spec": {
            "Labels": {},
            "Role": "manager",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "docker-desktop",
```

Apagar swarm

```
docker swarm leave
docker swarm leave --force # forzar el cierre
```

Verificar si swarm esta activo

```
docker info
```



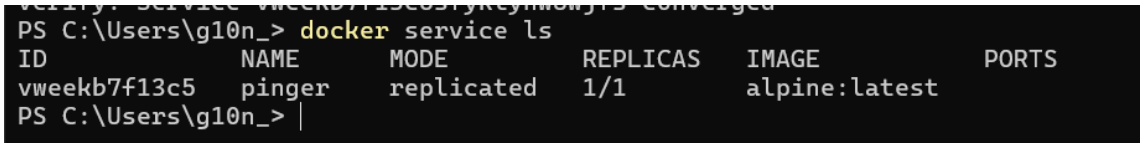
Creación de servicios de Docker Swarm

Imagen de prueba

```
docker service create --name pinger alpine ping www.google.com
```

Ver los servicios que estan corriendo

```
docker service ls
```



Ver el docker ps

Limpieza automatica de imagenes sin uso

Consulta usuario:

Tengo corriendo docker swarm para unos microservices en mi trabajo, tenemos un CI para hacer constantemente deploys al docker swarm. La cuestión es que el server cada día va consumiendo mas disco duro y es porque las imágenes de docker se van acumulando. La solución que hice fue crear un cron job que lo que hace es que todos los días hace un docker system prune -f para eliminar todas las imágenes que ya no están en uso.

Cual seria una buena practica para esto ?

Mejor respuesta:

En swarm siempre tienes que tener algún mecanismo de mantenimiento y limpieza. Yo uso meltwater/docker-cleanup corriendo como un servicio global de Swarm, lo que me garantiza que corre en todos los nodos, y delego en él la tarea de limpiar todo. Lo hago así:

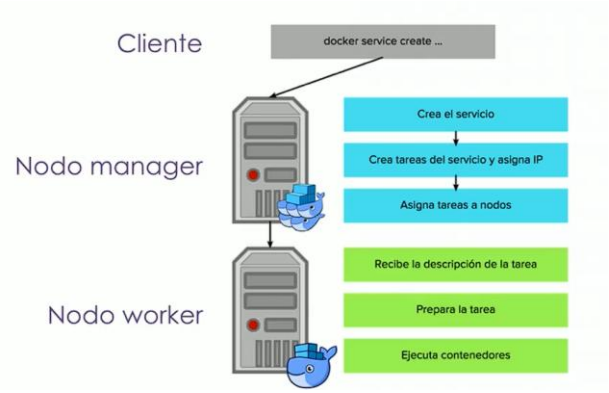
```
docker service create \
  --detach \
  -e CLEAN_PERIOD=900 \
  -e DELAY_TIME=600 \
  --log-driver json-file \
  --log-opt max-size=1m \
  --log-opt max-file=2 \
  --name cleanup \
  --mode global \
  --mount type=bind,source=/var/run/docker.sock,target=/var/run/docker.sock \
  meltwater/docker-cleanup
```

Estados, Tareas y Logs

Ver el estado

```
docker service ps pinger
```

```
PS C:\Users\g10n_> docker service ps pinger
ID             NAME       IMAGE          NODE             DESIRED STATE   CURRENT STATE            ERROR     PORTS
rl1jpg8orybk   pinger.1   alpine:latest   docker-desktop   Running         Running 35 minutes ago
PS C:\Users\g10n_> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
5ef13a80492d   alpine:latest   "ping www.google.com"   36 minutes ago   Up 36 minutes            pinger.1.rl1jpg8orybkr6w0bhzvpkkvb
PS C:\Users\g10n_>
```

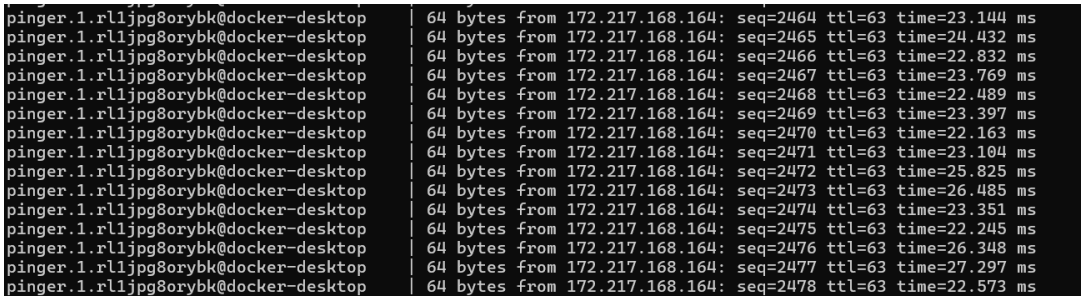


Inspeccionar servicio

```
docker inspect service pinger
```

Ver los logs

```
docker service logs -f pinger # Con el -f va a estar mostrando siempre lo ultimo
```



Eliminar servicio

El contenedor no se elimina inmediatamente, hay que esperar unos segundos

```
docker service rm pinger
```

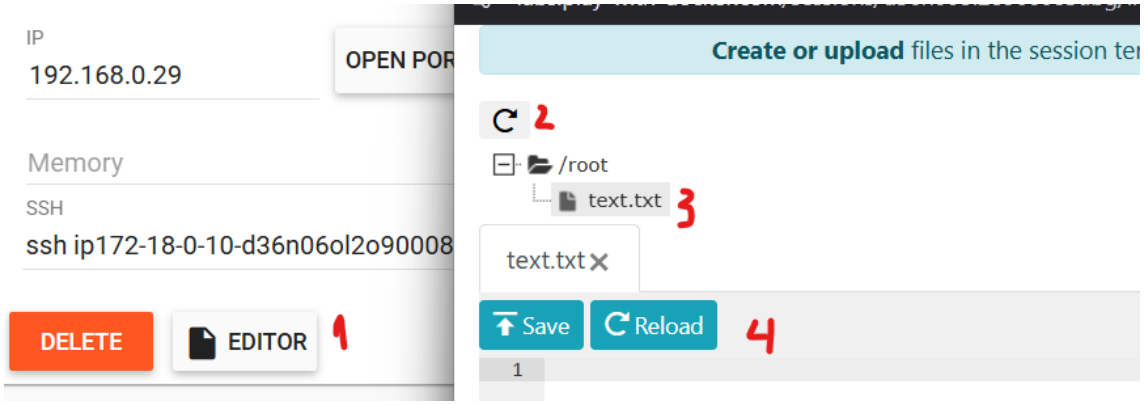
Conectar con Play with docker antes del multinodo

Probar la siguiente pagina

```
https://labs.play-with-docker.com/
```

Abrir el editor en play with docker, primero creamos un archivo

```
touch test.txt
```



Se puede conectar a play with docker desde mi terminal

Pasos a seguir (se probo en wsl, intentar con powershell):

Verifica si ya tienes claves SSH en tu máquina:

```
ls -al ~/.ssh
```

Si no tienes clave, genera una nueva:

```
ssh-keygen -t ed25519 -C "tu_correo@ejemplo.com"
```

Obtén tu clave pública y copiarla

```
cat ~/.ssh/id_ed25519.pub
```

Entra al contenedor desde el navegador en Play-with-Docker

```
mkdir -p ~/.ssh
echo "AQUÍ_PEGA_TU_CLAVE_PUBLICA" >> ~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Desde tu máquina, conecta con tu clave:

```
ssh -i ~/.ssh/id_ed25519 \

  ip172-18-0-10-d36n06ol2o900083tibg@direct.labs.play-with-docker.com
```

Comprobar la conexión

```
uname -a
```

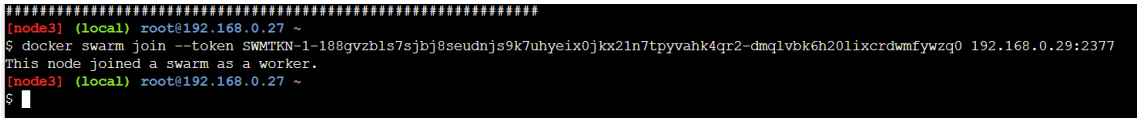
Docker Swarm Multinodo

Correr swarm en PWD

```
docker swarm init --advertise-addr 192.168.0.29
```

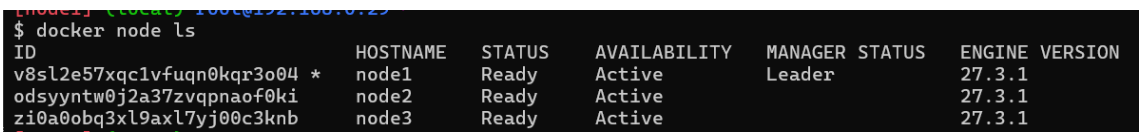
Unirse al swarm del nodo 1, este codigo lo da el comando anterior

```
docker swarm join --token SWMTKN-1-
188gvzbls7sjbj8seudnjs9k7uhyeix0jkx21n7tpyvahk4qr2-dmqlvbk6h20lixcrdwmfywzq0
192.168.0.29:2377
```



Verificar nodos asociados

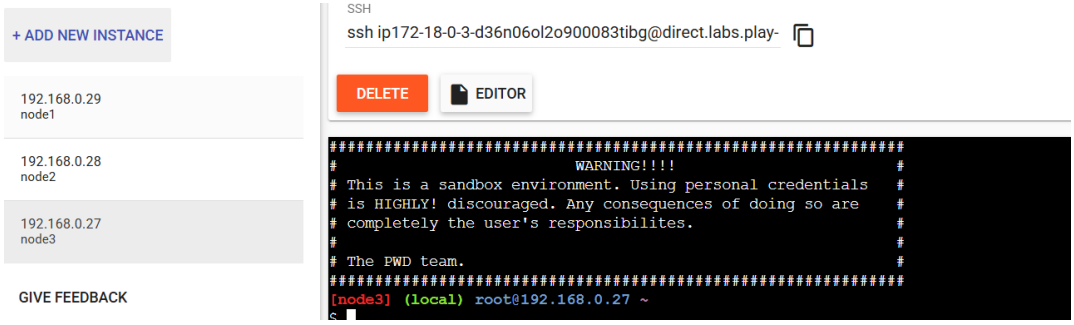
```
docker node ls
```



Correr contenedor

```
docker service create --name pinger alpine ping www.google.com
```

Crear mas instancias en PWD



Escalar servicios

```
docker service scale pinger=5
```

```
$ docker service scale pinger=5
pinger scaled to 5
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service pinger converged
```

Ver estado

```
docker service ps pinger
```

```
$ docker service ps pinger
ID             NAME       IMAGE          NODE      DESIRED STATE   CURRENT STATE           ERROR     PORTS
gqsg5jxt1eny   pinger.1   alpine:latest   node1     Running         Running 3 minutes ago
tjh7nskm5cry   pinger.2   alpine:latest   node3     Running         Running 8 seconds ago
xoby5m28subd   pinger.3   alpine:latest   node1     Running         Running 9 seconds ago
zf50uyifu29l   pinger.4   alpine:latest   node2     Running         Running 8 seconds ago
ktm8wct3u2o0   pinger.5   alpine:latest   node3     Running         Running 8 seconds ago
```

Ver logs

```
docker service logs -f pinger
```

Actualizar alguna configuración del servicio

```
docker service update --args "ping www.amazon.com" pinger
```

Actualiza 1x1 para que este siempre disponible

```
$ docker service update --args "ping www.amazon.com" pinger
pinger
overall progress: 1 out of 5 tasks
1/5: running   [==================================================>]
2/5: ready     [=========================================>         ]
3/5:
4/5:
5/5:
```

realiza rollback o cambia al spec (cambio) anterior, solo guarda 1 anterior, de igual manera los hace 1x1.

```
docker service rollback pinger
```

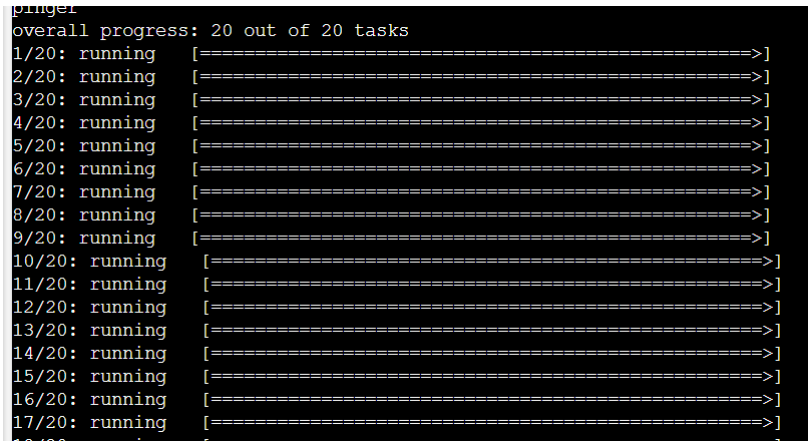
Gestion de actualizaciones o fallos en Swarm

Aumentar replicas

```
docker service update --replicas=20 pinger
```

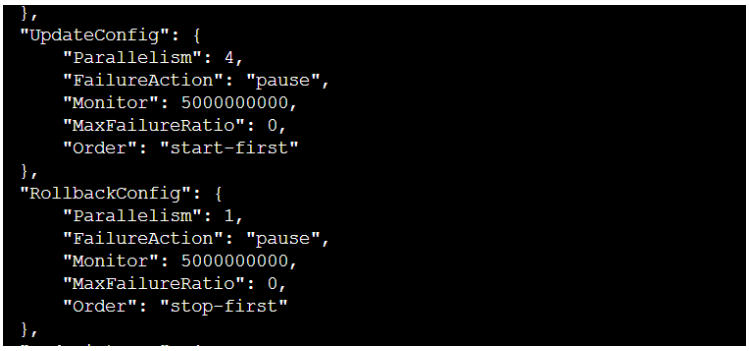
Cambiar paralelismo del servicio (actualiza la cantidad definida al mismo tiempo), con update order (start first, eliminamos primero el contenedor antes de crear el nuevo)

```
docker service update --update-parallelism 4 --update-order start-first pinger
```



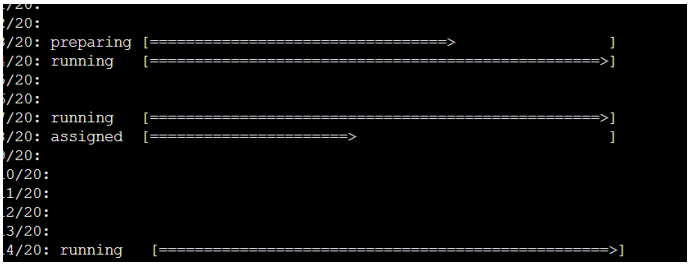
Ver la configuracion con

```
docker service inspect pinger
```



Actualizar servicio, segun la configuracion lo hara de a 4 que sera mas rapido

```
docker service update --args "ping www.facebook.com" pinger
```



Ver contenedores

```
docker service ps pinger
```

```
$ docker service ps pinger
ID             NAME            IMAGE          NODE      DESIRED STATE   CURRENT STATE                ERROR     PORTS
q7lkd117wmeo   pinger.1        alpine:latest  node1     Running         Running 6 minutes ago
a6dub4vuo4zc   pinger.2        alpine:latest  node3     Running         Running 5 minutes ago
abq4trc6kwbv   pinger.3        alpine:latest  node2     Running         Running about a minute ago
mpwo5qyjrago    \_ pinger.3    alpine:latest  node2     Shutdown        Shutdown about a minute ago
pw67jil68obx   pinger.4        alpine:latest  node2     Running         Running 5 minutes ago
7rsp3cc65jvl   pinger.5        alpine:latest  node1     Running         Running 5 minutes ago
pkuc28t18dci   pinger.6        alpine:latest  node1     Running         Running 5 minutes ago
enc1fwbggxzb   pinger.7        alpine:latest  node2     Running         Running about a minute ago
nbc8anp0j21b    \_ pinger.7    alpine:latest  node3     Shutdown        Shutdown about a minute ago
```

Si falla el 50% de las updates que se haga un rollback

```
docker service update --update-failure-action rollback --update-max-failure-ratio 0.5 pinger
```

```
},
"UpdateConfig": {
    "Parallelism": 4,
    "FailureAction": "rollback",
    "Monitor": 5000000000,
    "MaxFailureRatio": 0.5,
    "Order": "start-first"
},
"RollbackConfig": {
    "Parallelism": 1,
    "FailureAction": "pause",
    "Monitor": 5000000000,
    "MaxFailureRatio": 0,
    "Order": "stop-first"
},
```

Cambiar la configuracion del paralelismo para el rollback.

```
docker service update --rollback-parallelism 0 pinger
```

```
},
"UpdateConfig": {
    "Parallelism": 4,
    "FailureAction": "rollback",
    "Monitor": 5000000000,
    "MaxFailureRatio": 0.5,
    "Order": "start-first"
},
"RollbackConfig": {
    "Parallelism": 0,
    "FailureAction": "pause",
    "Monitor": 5000000000,
    "MaxFailureRatio": 0,
    "Order": "stop-first"
},
```

Producir un fallo a proposito

```
docker service update --args "asdasdsa" pinger
```

```
9/20: running   [==================================================>]
10/20: running   [==================================================>]
11/20: running   [==================================================>]
12/20: running   [==================================================>]
13/20: running   [==================================================>]
14/20: running   [==================================================>]
15/20: running   [==================================================>]
16/20: running   [==================================================>]
17/20: running   [==================================================>]
18/20: running   [==================================================>]
19/20: running   [==================================================>]
20/20: running   [==================================================>]
rollback: update rolled back due to failure or early termination of task mxou80vlr6py3js2ktkrxuwf2
verify: Service pinger converged
```