

# Docker SWARM

Alta disponibilidad con Docker-Swarm  
Manuel Alcocer Jiménez  
manuel@alcoer.net

## Índice

---

1.Introducción.....	4
1.1.Nomenclatura empleada.....	4
1.2.Objetivos.....	4
1.3.Descripción del montaje.....	4
1.3.1.Proxmox.....	4
Almacenamiento en Proxmox.....	4
1.3.2.GlusterFS.....	4
1.3.3.Galera.....	5
Haproxy.....	5
2.Eschema de montaje.....	6
3.Configuración de NFS en Proxmox.....	7
3.1.Descripción.....	7
3.2.Instalación de paquetes.....	7
3.3.Configuración de las unidades.....	7
3.3.1.Nodo2.....	7
3.3.2.Nodo3.....	7
3.3.3.Nodo4.....	7
3.4.Configuración de FSTAB.....	8
3.4.1.Nodo2.....	8
3.4.2.Nodo3.....	8
3.4.3.Nodo4.....	8
3.5.Configuración de los exports.....	8
3.5.1.Nodo2.....	8
3.5.2.Nodo3.....	8
3.5.3.Nodo4.....	8
3.6.Configuración de PROXMOX.....	8
3.6.1.Añadir unidades NFS.....	9
Carpeta de red del Nodo2.....	9
Carpeta de red del Nodo3.....	10
Carpeta de red del Nodo4.....	10
Carpetas en el clúster.....	11
3.7.Sistema Operativo plantilla.....	11
4.Proveer docker a todas las máquinas.....	13
5.Almacenamiento compartido, GlusterFS.....	14
5.1.Paquetes necesarios.....	14
5.2.Configuración de los servidores Gluster.....	14
5.3.Crear bricks.....	14
5.4.Montaje de las unidades con Systemd.....	14
6.Creación de Managers y workers.....	16
6.1.Descripción.....	16
6.2.Creación de managers.....	16
6.2.1.Manager principal.....	16
6.2.2.Manager secundario.....	16
En el manager principal.....	16
En el manager secundario.....	16
6.3.Creación de workers.....	16
6.3.1.Workers del manager principal.....	16
Swarm-mgr21.....	16
swarm-wrkr21.....	17
swarm-wrkr22.....	17
6.3.2.Workers del Manager secundario.....	17

Swarm-mgr31.....	17
swarm-wrkr31.....	17
swarm-wrkr32.....	17
7.Estado de los nodos.....	18
7.1.Lista.....	18
8.Docker Registry.....	19
8.1.Crear registry.....	19
8.1.1.Certificados necesarios.....	19
8.1.2.Servicio de actualización de certificados.....	19
Timer.....	19
Servicio.....	19
Copia de certificados.....	19
8.1.3.Despliegue del Registry.....	20
9.Creación de servicios en HA.....	21
9.1.Agregar imágenes al registry.....	21
9.1.1.Descarga de una imagen.....	21
9.1.2.Etiquetado de una imagen en el registry.....	21
9.1.3.Subida de la imagen.....	21
10.MariaDB + Galera.....	22
10.1.Paquetes necesarios.....	22
10.1.1.En los 2 nodos.....	22
10.2.Configuración de Galera.....	22
10.2.1./etc/mysql/conf.d/galera.cnf.....	22
10.2.2.Arranque de Galera.....	22
Parar MYSQL.....	22
Iniciar Galera en mysql-i.....	22
Iniciar mysql-ii.....	22
11.Galera HA-PROXY.....	24
11.1.Paquetes necesarios.....	24
11.2.Configuración de HAPROXY.....	24
11.2.1.Configuración de MySQL.....	24
12.Despliegue de Wordpres.....	25
12.1.Dockerfile.....	25
12.1.1.Script.sh.....	25
12.2.Subida de la imagen.....	25
12.2.1.Compose.....	25
12.3.Despliegue final.....	26
12.3.1.Estado.....	26
12.3.2.MariaDB-Haproxy.....	26
13.Conclusiones, inconvenientes y problemas.....	27
13.1.Inconvenientes y problemas.....	27
13.1.1.Swarm.....	27
Volúmenes.....	27
Balanceo de carga.....	27
13.1.2.Proxmox.....	27
13.1.3.GlusterFS.....	27
13.2.Conclusiones.....	27

# 1. Introducción

---

Este proyecto sienta las bases y soluciona algunos inconvenientes que tiene desplegar microservicios con Docker Swarm.

Docker Swarm es la solución a la Alta Disponibilidad en contenedores que ofrece Docker. Entre las principales características que ofrece Docker Swarm se encuentran:

- Balanceo de Carga
- Auto-levantamiento de servicios
- DNS interno
- Replicación de servicios

## 1.1. Nomenclatura empleada

- Swarm manager: Rol de máquina para gestión de los contenedores del despliegue SWARM. No tiene porque albergar las máquinas virtuales, se encarga de coordinar, levantar y parar los contenedores del despliegue.
- Swarm worker: Este rol implica que la máquina solo servirá para levantar contenedores.
- Stack: Conjunto de servicios o contenedores.

## 1.2. Objetivos

El objetivo es desplegar un CMS, Wordpress, replicado en contenedores con Docker Swarm.

## 1.3. Descripción del montaje

Todo el montaje se hará sobre máquinas virtuales usando KVM.

### 1.3.1. Proxmox

Se usará Proxmox para la gestión de las máquinas virtuales.

#### **Almacenamiento en Proxmox**

La solución adoptada para el almacenamiento ha sido NFS, ya que en Proxmox, LVM-THIN y LVM no se pueden usar para compartir entre los Nodos del Clúster los discos de las máquinas virtuales.

### 1.3.2. GlusterFS

Para que los contenedores de las máquinas virtuales tengan acceso a los mismos archivos y su contenido siempre esté actualizado entre ellos, he usado GlusterFS.

Esto garantiza que en caso de caída y levantamiento de cualquiera de los workers el contenido siempre será el mismo para todas las máquinas.

### 1.3.3. Galera

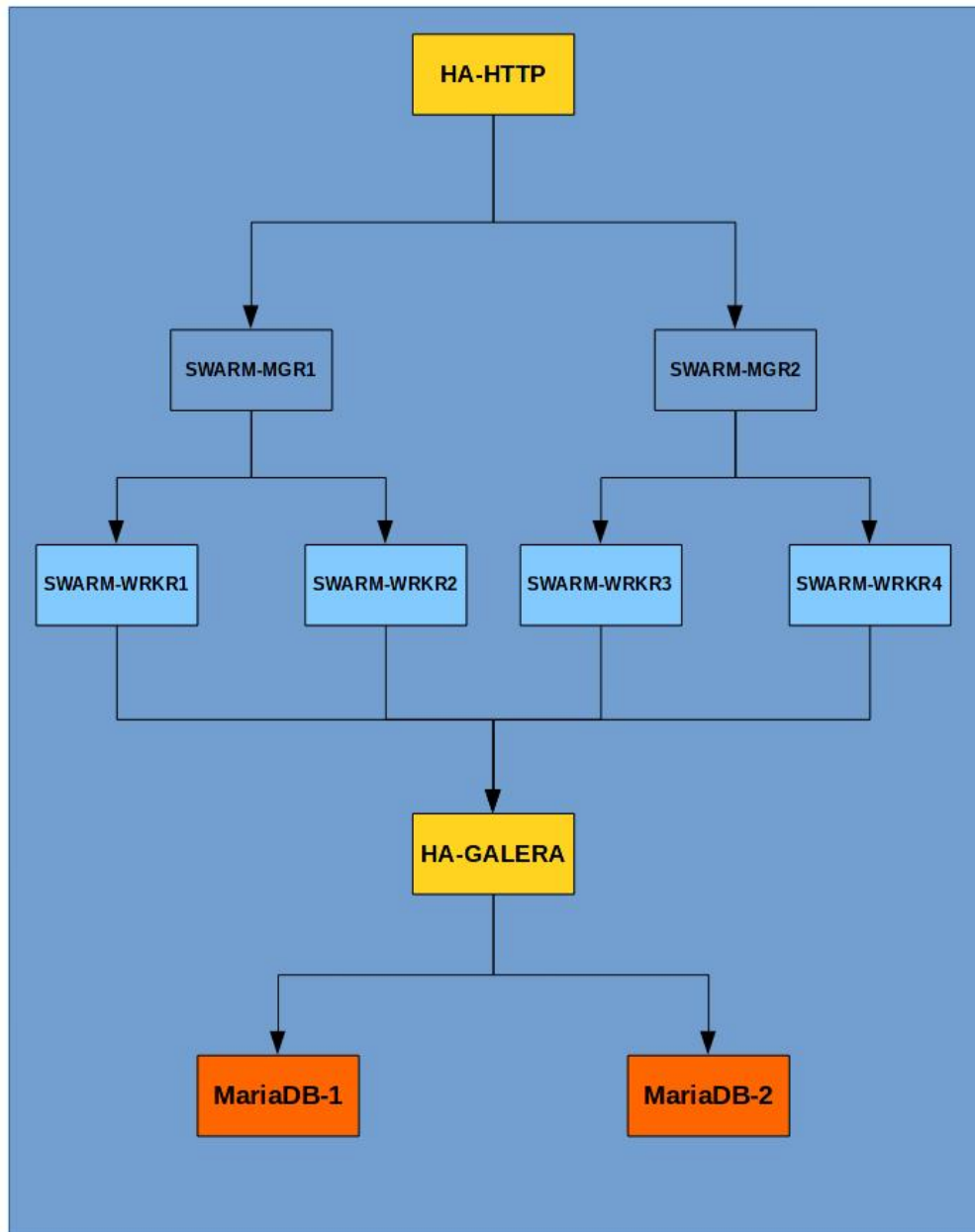
Galera con MariaDB es un sistema maestro-maestro que permite tener replicado el servicio de base de datos.

#### **Haproxy**

Para repartir la carga entre los nodos del clúster de bases de datos se usará Haproxy.

## 2. Esquema de montaje

---



## 3. Configuración de NFS en Proxmox

---

### 3.1. Descripción

Como cada carpeta compartida NFS estará en los propios nodos del cluster de Proxmox, hay que instalar el servicio de NFS-Server en todos ellos.

### 3.2. Instalación de paquetes

En todos los nodos del clúster:

```
root@pve2:~# apt-get install nfs-kernel-server
```

### 3.3. Configuración de las unidades

El sistema de archivos para los volúmenes será XFS.

#### 3.3.1. Nodo2

```
root@pve2:~# mkfs.xfs -L PVE2NFS-1 /dev/sdb1
meta-data=/dev/sdb1      isize=256    agcount=4, agsize=61047598 blks
                    =               sectsz=512   attr=2, projid32bit=1
                    =               crc=0        finobt=0
data        =             bsize=4096    blocks=244190390, imaxpct=25
                    =             sunit=0     swidth=0 blks
naming     =version 2     bsize=4096   ascii-ci=0 ftype=0
log        =internal log  bsize=4096   blocks=119233, version=2
                    =             sectsz=512   sunit=0 blks, lazy-count=1
realtime   =none        extsz=4096   blocks=0, rtextents=0
```

#### 3.3.2. Nodo3

```
root@pve3:~# mkfs.xfs -L PVE3NFS-1 /dev/sdb1
meta-data=/dev/sdb1      isize=256    agcount=4, agsize=61047598 blks
                    =               sectsz=512   attr=2, projid32bit=1
                    =               crc=0        finobt=0
data        =             bsize=4096    blocks=244190390, imaxpct=25
                    =             sunit=0     swidth=0 blks
naming     =version 2     bsize=4096   ascii-ci=0 ftype=0
log        =internal log  bsize=4096   blocks=119233, version=2
                    =             sectsz=512   sunit=0 blks, lazy-count=1
realtime   =none        extsz=4096   blocks=0, rtextents=0
```

#### 3.3.3. Nodo4

```
root@pve4:~# mkfs.xfs -L PVE3NFS-1 /dev/sdb1
meta-data=/dev/sdb1      isize=256    agcount=4, agsize=61047598 blks
                    =               sectsz=512   attr=2, projid32bit=1
                    =               crc=0        finobt=0
data        =             bsize=4096    blocks=244190390, imaxpct=25
                    =             sunit=0     swidth=0 blks
naming     =version 2     bsize=4096   ascii-ci=0 ftype=0
log        =internal log  bsize=4096   blocks=119233, version=2
                    =             sectsz=512   sunit=0 blks, lazy-count=1
```

```
realtime =none                extsz=4096  blocks=0, rtextents=0
```

## 3.4. Configuración de FSTAB

Monto cada partición en una carpeta en `/srv`.

### 3.4.1. Nodo2

```
root@pve2:~# grep -E '^U' /etc/fstab
UUID=1744d410-c47b-4932-8146-0480a35fc223    /srv/nfs-2    xfs    defaults    0 2
```

### 3.4.2. Nodo3

```
root@pve3:~# grep -E '^U' /etc/fstab
UUID=0df3db0d-99b4-4ca7-a375-eb492b9fdb21    /srv/nfs-3    xfs    defaults    0 2
```

### 3.4.3. Nodo4

```
UUID=a143b575-8345-4ea1-81ec-1bc1c11cb860    /srv/nfs-4    xfs    defaults    0 2
```

## 3.5. Configuración de los exports

Los directorios de red solo serán visibles para el rango de IPS 10.0.0.[0-7]

### 3.5.1. Nodo2

```
root@pve2:~# grep -Eiv '^#' /etc/exports
/srv/nfs-2    10.0.0.0/29(rw, sync, no_subtree_check, no_root_squash)
```

### 3.5.2. Nodo3

```
root@pve3:~# grep -Eiv '^#' /etc/exports
/srv/nfs-3    10.0.0.0/29(rw, sync, no_subtree_check, no_root_squash)
```

### 3.5.3. Nodo4

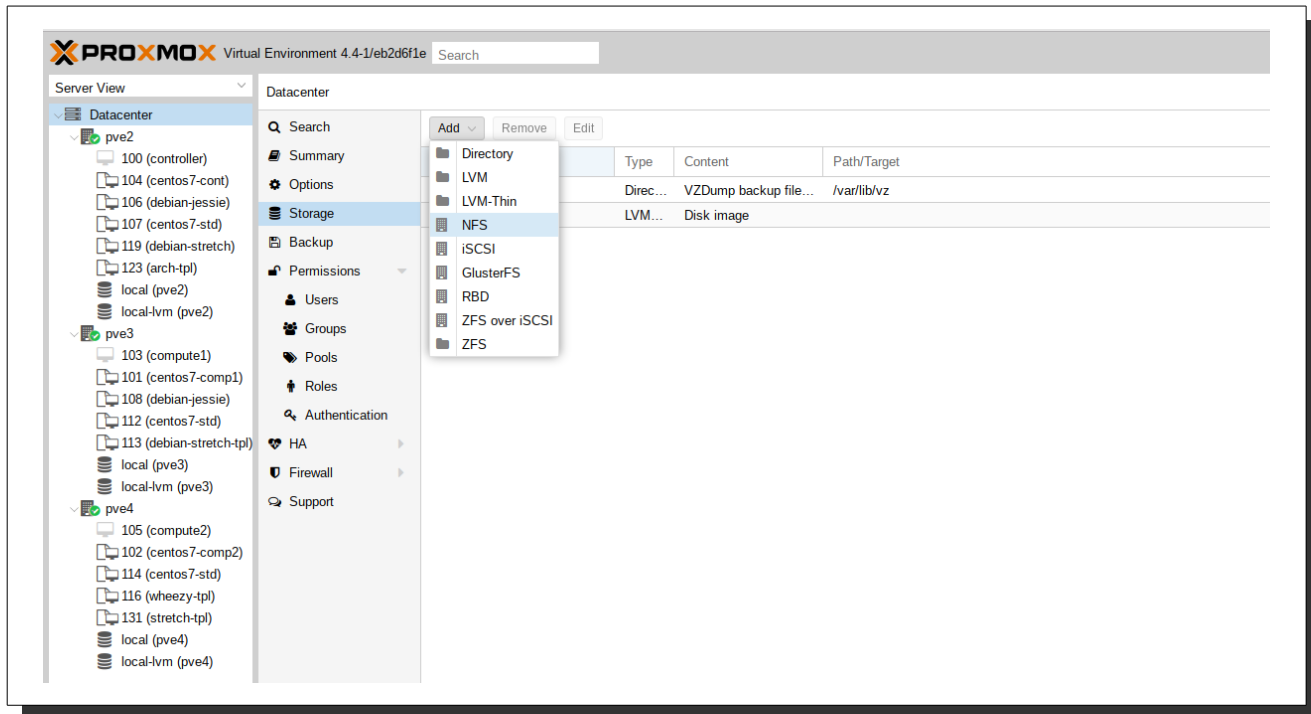
```
root@pve4:~# grep -Eiv '^#' /etc/exports
/srv/nfs-4    10.0.0.0/29(rw, sync, no_subtree_check, no_root_squash)
```

## 3.6. Configuración de PROXMOX

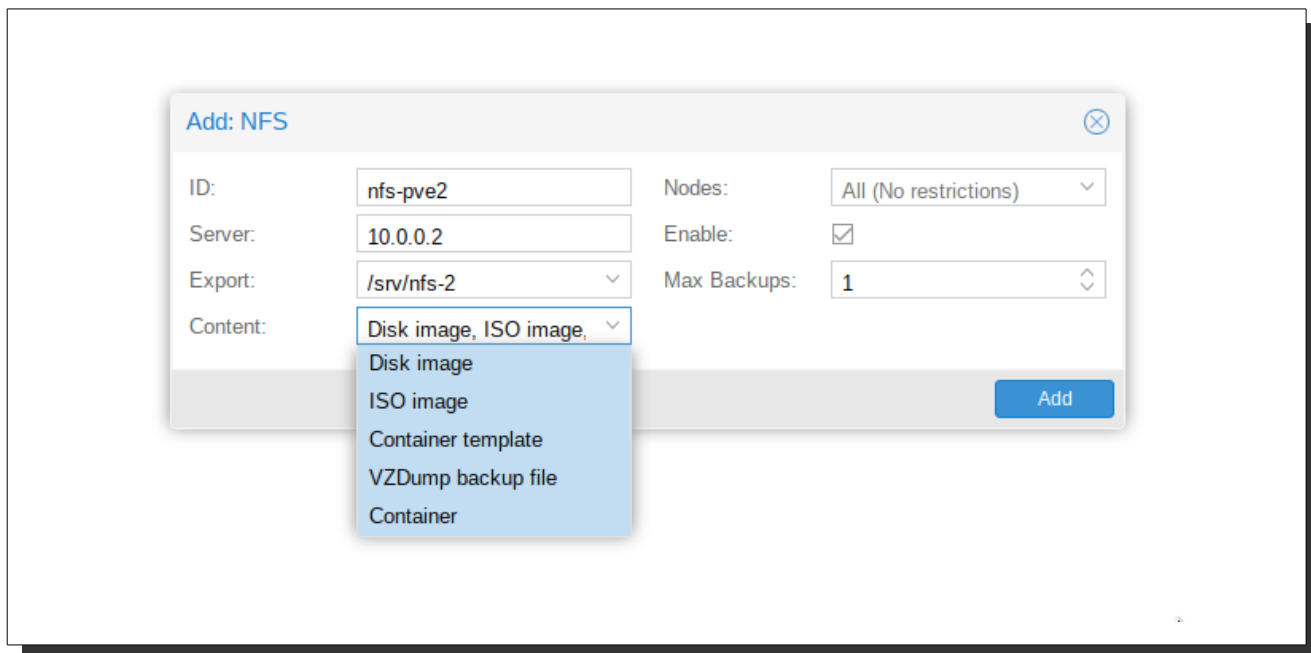
Para añadir las unidades se usará la GUI de PROXMOX.



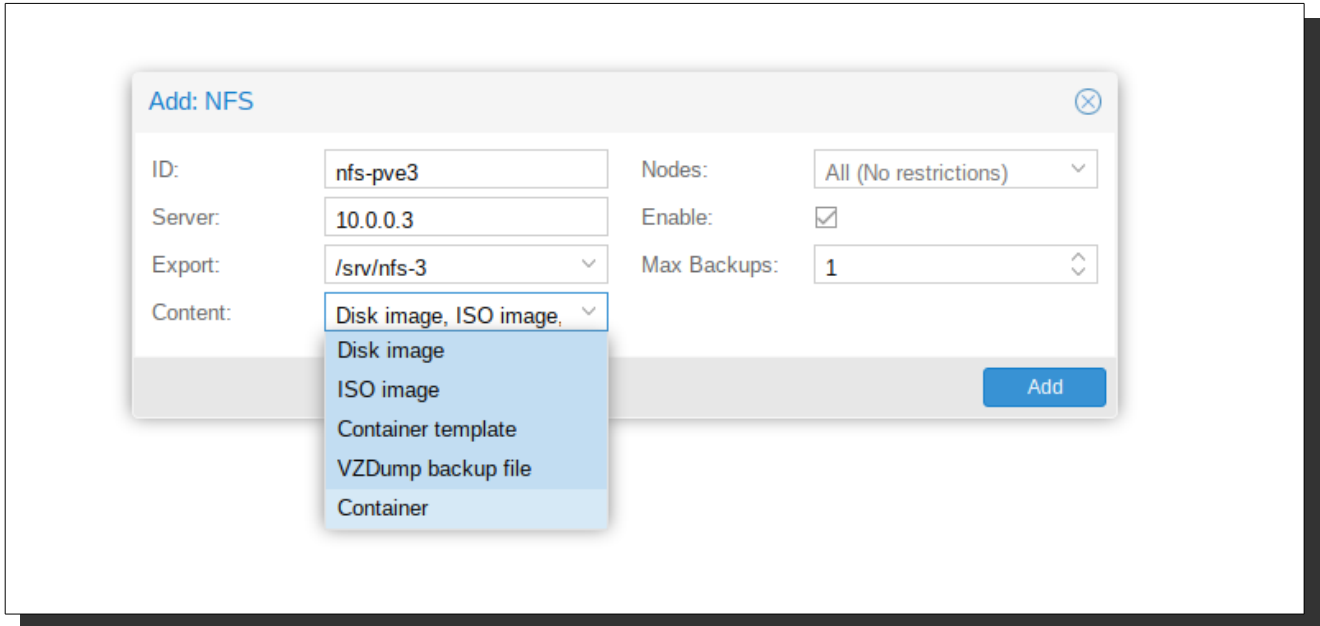
### 3.6.1. Añadir unidades NFS



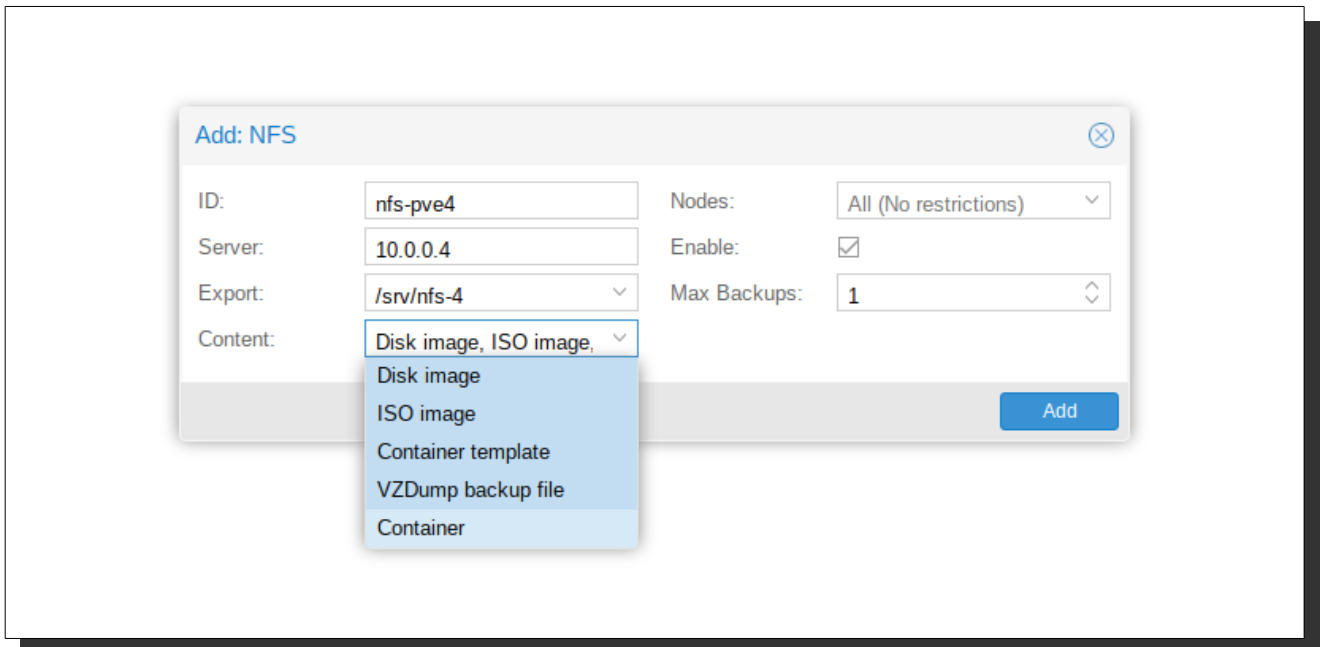
#### Carpeta de red del Nodo2



### Carpeta de red del Nodo3



### Carpeta de red del Nodo4



### Carpetas en el clúster

ID ↑	Type	Content	Path/Target
local	Direc...	VZDump backup file...	/var/lib/vz
local-lvm	LVM...	Disk image	
nfs-pve2	NFS	VZDump backup file...	/mnt/pve/nfs-pve2
nfs-pve3	NFS	VZDump backup file...	/mnt/pve/nfs-pve3
nfs-pve4	NFS	VZDump backup file...	/mnt/pve/nfs-pve4

### 3.7. Sistema Operativo plantilla

Todas las máquinas estarán montadas sobre un sistema operativo plantilla en modo de Aprovisionamiento Ligero.



## 4. Proveer docker a todas las máquinas

---

Se usará docker-machine para proveer las máquinas con la última versión de Docker.

```
→ ~ for x in swarm-{mgr21,wrkr21,wrkr22,mgr31,wrkr31,wrkr32}; do
docker-machine create -d generic \
--generic-ip-address=$x.alcocer.net \
--generic-ssh-user root \
$x
done
```

Estado de las máquinas

```
→ ~ docker-machine ls
NAME          ACTIVE   DRIVER   STATE   URL                                     SWARM
DOCKER ERRORS
swarm-mgr21   -        generic  Running tcp://swarm-mgr21.alcocer.net:2376
v17.05.0-ce
swarm-mgr31   -        generic  Running tcp://swarm-mgr31.alcocer.net:2376
v17.05.0-ce
swarm-wrkr21  -        generic  Running tcp://swarm-wrkr21.alcocer.net:2376
v17.05.0-ce
swarm-wrkr22  -        generic  Running tcp://swarm-wrkr22.alcocer.net:2376
v17.05.0-ce
swarm-wrkr31  -        generic  Running tcp://swarm-wrkr31.alcocer.net:2376
v17.05.0-ce
swarm-wrkr32  -        generic  Running tcp://swarm-wrkr32.alcocer.net:2376
v17.05.0-ce
```

## 5. Almacenamiento compartido, GlusterFS

---

Cada máquina tendrá un disco adicional para los datos compartidos, de 5GB.

GlusterFS permite tener directorios compartidos en la red tanto en modo 'replicado' como en 'distribuido'. Para el trabajo aquí descrito se usará 'replicado'.

'Distribuido' se asemeja bastante a un Raid-5, donde las escrituras se reparten a lo largo de todos los nodos del clúster.

'Replicado' sería el equivalente al Raid-1.

### 5.1. Paquetes necesarios

Instalación de paquetes necesarios en cada máquina.

```
→ ~ for x in swarm-{mgr21,wrkr21,wrkr22,mgr31,wrkr31,wrkr32}; do
ssh root@$x.alcocer.net apt -y install glusterfs-server
done
```

### 5.2. Configuración de los servidores Gluster

El comando 'peer probe [host]' solo se ejecuta en una de las máquinas, su efecto es recíproco.

Aunque Gluster lo permite, no usaré limitación a los recursos restringida por IP, ese no es el propósito de este proyecto.

En uno de los servidores:

```
root@swarm-mgr21:~# for x in swarm-{mgr31,wrkr21,wrkr22,wrkr31,wrkr32}; do
> gluster peer probe $x.alcocer.net
> done
peer probe: success.
peer probe: success.
peer probe: success.
peer probe: success.
peer probe: success.
peer probe: success.
root@swarm-mgr21:~#
```

### 5.3. Crear bricks

Brick es la nomenclatura que se da a la unidad mínima de almacenamiento de GlusterFS.

En uno de los servidores:

```
root@swarm-mgr21:~# gluster volume create gv1 replica 6 swarm-
{mgr21,mgr31,wrkr21,wrkr22,wrkr31,wrkr32}.alcocer.net:/data/brick1/gv1
volume create: gv1: success: please start the volume to access data
root@swarm-mgr21:~# gluster volume start gv1
volume start: gv1: success
```

### 5.4. Montaje de las unidades con Systemd

En cada servidor hay que crear una unidad de systemd. Cada servicio debe llamarse obligatoriamente ruta-de-montaje.mount, sustituyendo las / por -.

```
root@swarm-mgr21:~# cat /etc/systemd/system/mnt-gv1.mount
```

```
[Unit]
Description=Mount glusterfs vol1
After=network.target glusterfs-server.service

[Mount]
What=swarm-mgr21.alcocer.net:/gv1
Where=/mnt/gv1
Type=glusterfs
Options=defaults

[Install]
WantedBy=multi-user.target
```

## 6. Creación de Managers y workers

---

### 6.1. Descripción

Los 'Managers' son los nodos encargados de la vigilancia y gestión de los contenedores. En este caso se puede atacar a cualquiera de los 2 managers que estarán en el clúster.

### 6.2. Creación de managers

#### 6.2.1. Manager principal

```
root@swarm-mgr21:~# docker swarm init --advertise-addr 10.0.10.146
Swarm initialized: current node (g9zcydz44pz1tzwcstgtqmsjo) is now a manager.
```

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

```
docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwppocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.146:2377
```

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

```
root@swarm-mgr21:~#
```

#### 6.2.2. Manager secundario

##### En el manager principal

```
root@swarm-mgr21:~# docker swarm join-token manager
To add a manager to this swarm, run the following command:
```

```
docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwppocg8ldmoecpygpao-
7ezb0zf0gm56ej9w2f5yqavvm \
  10.0.10.146:2377
```

```
root@swarm-mgr21:~#
```

##### En el manager secundario

```
root@swarm-mgr31:~# docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwppocg8ldmoecpygpao-
7ezb0zf0gm56ej9w2f5yqavvm \
  10.0.10.146:2377
This node joined a swarm as a manager.
```

### 6.3. Creación de workers

#### 6.3.1. Workers del manager principal

##### Swarm-mgr21

```
root@swarm-mgr21:~# docker swarm join-token worker
```



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

```
docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwpocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.146:2377
```

### swarm-wrkr21

```
root@swarm-wrkr21:~# docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwpocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.146:2377
This node joined a swarm as a worker.
root@swarm-wrkr21:~#
```

### swarm-wrkr22

```
root@swarm-wrkr22:~# docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwpocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.146:2377
This node joined a swarm as a worker.
```

## 6.3.2. Workers del Manager secundario

### Swarm-mgr31

```
root@swarm-mgr31:~# docker swarm join-token worker
To add a worker to this swarm, run the following command:
```

```
docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwpocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.149:2377
```

### swarm-wrkr31

```
root@swarm-wrkr31:~# docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwpocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.149:2377
This node joined a swarm as a worker.
root@swarm-wrkr31:~#
```

### swarm-wrkr32

```
root@swarm-wrkr32:~# docker swarm join \
  --token SWMTKN-1-3spyv1s10oeg2fyqmv9ne5atdqp3dqwpocg8ldmoecpygpao-
795rx84hq4uvzjt7okxq6pxwg \
  10.0.10.149:2377
This node joined a swarm as a worker.
```

## 7. Estado de los nodos

---

### 7.1. Lista

```
root@swarm-mgr21:~# docker node ls
ID                                HOSTNAME        STATUS        AVAILABILITY
MANAGER STATUS
3z2dsfh3owx5gfz2lgkui2f15       swarm-mgr31     Ready        Active
Reachable
a0dbr16vqdcmpn6wrfq13tt75       swarm-wrkr31    Ready        Active
g9zcydz44pz1tzwcstgtqmsjo *    swarm-mgr21     Ready        Active
Leader
php9ovzop0rvyqivv7smvy6zq       swarm-wrkr32    Ready        Active
uys4sx1m6tm2nlxv73ypa5ogf       swarm-wrkr22    Ready        Active
xen1nvtqcrzmr53sozhgofop        swarm-wrkr21    Ready        Active
root@swarm-mgr21:~#
```

Leader: Manager principal

Reachable: En caso de fallo del principal, este se activaría.

## 8. Docker Registry

---

### 8.1. Crear registry

#### 8.1.1. Certificados necesarios

Para que de forma remota se pueda acceder a un 'Registry' es necesario que la comunicación con él sea cifrada.

Para evitar tener que instalar el certificado raíz en cada máquina que haga uso del registry, instalaré certificados de Letsencrypt.

```
root@vps:~# certbot certonly --webroot -w /var/www/html/ --expand
-d{alcocer.net,correo.alcocer.net,imap.alcocer.net,mail.alcocer.net,pop3.alcocer.net,smtp.
alcocer.net,vps.alcocer.net,webmail.alcocer.net,git.alcocer.net,registry.alcocer.net}
```

#### 8.1.2. Servicio de actualización de certificados

##### Timer

```
root@vps:~# cat /etc/systemd/system/letsencrypt.timer
[Unit]
Description=Renew certs from Let's Encrypt

[Timer]
OnCalendar=*-*-* 03:00:00

[Install]
WantedBy=multi-user.target
```

##### Servicio

```
root@vps:~# cat /etc/systemd/system/letsencrypt.service
[Unit]
Description=Renew certs from Let's Encrypt

[Service]
User=root
Type=oneshot
ExecStart=/usr/bin/certbot renew
ExecStart=/usr/local/bin/copycerts.sh

[Install]
WantedBy=multi-user.target
```

##### Copia de certificados

Es necesario mapear la ruta donde se alojan los certificados, pero Letsencrypt tiene los certificados actualizados mediante enlaces simbólicos, eso es incompatible con la directiva de mapeo de volúmenes de Docker.

Este script mantiene actualizados los certificados que usará el registry.

```
root@vps:~# cat /usr/local/bin/copycerts.sh
```

```
#!/usr/bin/env bash

if [[ ! -d "/certs" ]]; then
    mkdir /certs
fi

cp /etc/letsencrypt/live/alcocer.net/* /certs
```

### 8.1.3. Despliegue del Registry

```
root@vps:~# docker pull registry
root@vps:~# docker run -d -p 5000:5000 --restart=always \
-v /certs:/certs \
-e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/fullchain.pem \
-e REGISTRY_HTTP_TLS_KEY=/certs/privkey.pem \
registry
```

## 9. Creación de servicios en HA

---

### 9.1. Agregar imágenes al registry

Para agregar una imagen al 'registry' solo hace falta etiquetarla y luego subirla.

#### 9.1.1. Descarga de una imagen

```
root@swarm-mgr21:~# docker pull debian
```

#### 9.1.2. Etiquetado de una imagen en el registry

```
root@swarm-mgr21:~# docker tag debian:latest registry.alcocer.net:5000/debian:latest
```

#### 9.1.3. Subida de la imagen

```
root@swarm-mgr21:~# docker push registry.alcocer.net:5000/debian:latest
The push refers to a repository [registry.alcocer.net:5000/debian]
007ab444b234: Pushed
latest: digest: sha256:e283dc7bdfe4df3672ba561cf50022528c493cc5800e80670ca47315aad6a5de
size: 529
root@swarm-mgr21:~#
```

## 10. MariaDB + Galera

---

### 10.1. Paquetes necesarios

#### 10.1.1. En los 2 nodos

Esto instalará también Galera-3.

```
root@mysql-i:~# apt-get install software-properties-common && \
    apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 0xcbc082a1bb943db && \
    add-apt-repository 'deb [arch=amd64]
http://mariadb.mirror.nucleus.be/repo/10.1/debian jessie main' && \
    apt-get update && apt-get install mariadb-server
```

### 10.2. Configuración de Galera

#### 10.2.1. /etc/mysql/conf.d/galera.cnf

```
[mysqld]
#mysql settings
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
innodb_doublewrite=1
query_cache_size=0
query_cache_type=0
bind-address=0.0.0.0

#galera settings
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="mysql_cluster"
wsrep_cluster_address=gcomm://mysql-i.alcocer.net,mysql-ii.alcocer.net
wsrep_sst_method=rsync
```

#### 10.2.2. Arranque de Galera

##### Parar MYSQL

```
root@mysql-i:~# systemctl stop mysql
root@mysql-ii:~# systemctl stop mysql
```

##### Iniciar Galera en mysql-i

```
root@mysql-i:~# galera_new_cluster
root@mysql-i:~# mysql -root -e 'SELECT VARIABLE_VALUE as "cluster size" FROM
INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_NAME="wsrep_cluster_size"'
+-----+
| cluster size |
+-----+
| 1            |
+-----+
root@mysql-i:~#
```

##### Iniciar mysql-ii

```
root@mysql-ii:~# systemctl start mysql
```

```
-----+
root@mysql-ii:~# mysql -proot -e 'SELECT VARIABLE_VALUE as "cluster size" FROM
INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_NAME="wsrep_cluster_size"
+-----+
| cluster size |
+-----+
| 2             |
+-----+
root@mysql-ii:~#
root@mysql-i:~# mysql -proot -e 'SELECT VARIABLE_VALUE as "cluster size" FROM
INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_NAME="wsrep_cluster_size"
+-----+
| cluster size |
+-----+
| 2             |
+-----+
root@mysql-i:~#
-----+
```

# 11. Galera HA-PROXY

---

## 11.1. Paquetes necesarios

```
root@ha-mysql:~# apt install haproxy
```

## 11.2. Configuración de HAPROXY

Añadir las siguientes líneas a /etc/haproxy/haproxy.cfg

```
listen galera 10.0.10.155:3306
    balance source
    mode tcp
    option tcpka
    option mysql-check user haproxy
    server node1 mysql-i.alcocer.net:3306 check weight 1
    server node2 mysql-ii.alcocerr.net:3306 check weight 1
listen stats :9000
    mode http
    stats enable
    stats hide-version
    stats realm Haproxy\ Statistics
    stats uri /haproxy_stats
    stats auth admin:admin
```

### 11.2.1. Configuración de MySQL

```
MariaDB [(none)]> create user 'haproxy'@'10.0.10.155';
Query OK, 0 rows affected (0.06 sec)

MariaDB [(none)]> Bye
root@mysql-i:~#
```



## 12. Despliegue de Wordpres

---

### 12.1. Dockerfile

```
FROM registry.alcocer.net:5000/debian

LABEL maintainer Manuel Alcocer Jiménez <manuel@alcocer.net>

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get -y update && apt-get -y install apache2 \
    php5 libapache2-mod-php5 php5-mysqldb php5-curl php5-gd \
    php5-intl php-pear php5-imagick php5-imap php5-mcrypt \
    php5-memcache php5-pspell php5-recode php5-snmp \
    php5-sqlite php5-tidy php5-xmlrpc php5-xsl php5-apcu

COPY script.sh /root/script.sh
RUN chown www-data. -R /var/www
ENTRYPOINT ["/bin/bash", "/root/script.sh"]
```

#### 12.1.1. Script.sh

```
#!/bin/bash

apache2ctl start
while true; do
    :
done
```

### 12.2. Subida de la imagen

```
root@swarm-mgr21:/mnt/gv1/swarm-data/composes/apache# docker build -t apache-wp .
root@swarm-mgr21:/mnt/gv1/swarm-data/composes/apache# docker tag apache-wp
registry.alcocer.net:5000/apache-wp
root@swarm-mgr21:/mnt/gv1/swarm-data/composes/apache# docker push
registry.alcocer.net:5000/apache-w
```

#### 12.2.1. Compose

```
version: "3"

services:
  apache:
    image: registry.alcocer.net:5000/apache-wp
    deploy:
      mode: replicated
      replicas: 2
      restart_policy:
        condition: on-failure
        delay: 5s
        max_attempts: 3
        window: 120s
      placement:
        constraints:
          - node.role == worker
    ports:
      - "80:80"
```

```
volumes:
- ./apache/wordpress:/var/www/html/wordpress
```

### 12.3. Despliegue final

```
root@swarm-mgr21:/mnt/gv1/swarm-data/composes# docker stack deploy --compose-file
wordpress.yml wp-1
```

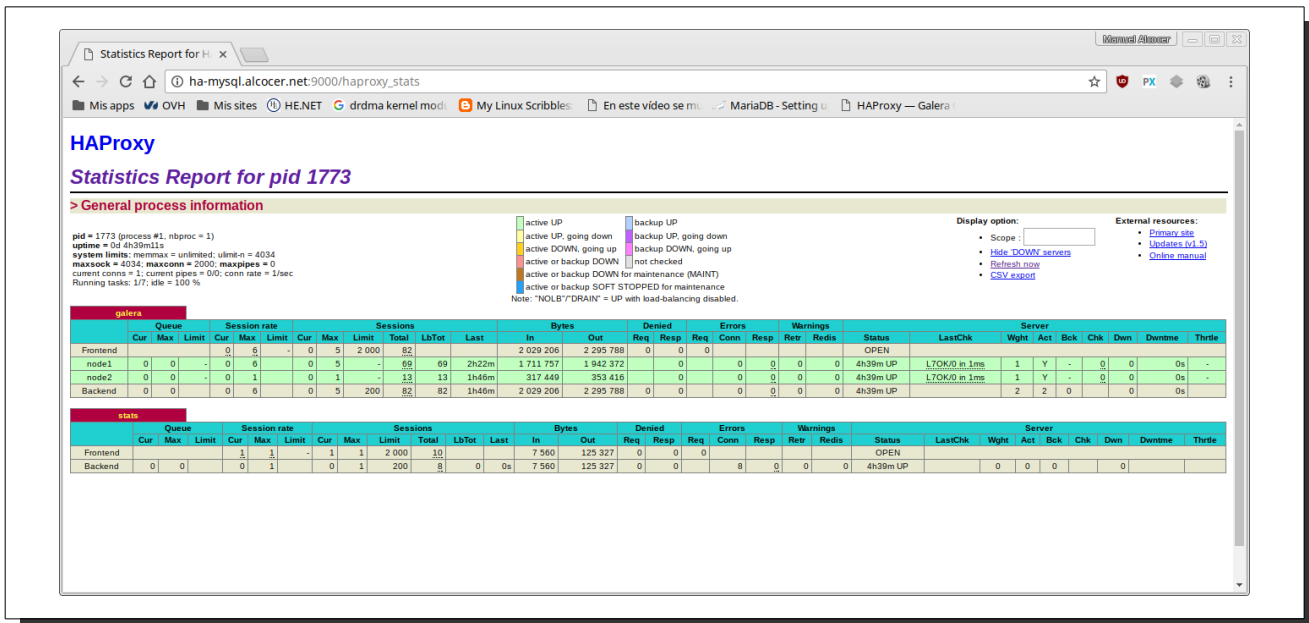
#### 12.3.1.Estado

```
root@swarm-mgr21:/mnt/gv1/swarm-data/composes# docker stack ps wp-1
```

ID	NAME	IMAGE	NODE
DESIRED STATE	CURRENT STATE	ERROR	PORTS
s4u9c1lk7fzu	wp-1_apache.1	registry.alcocer.net:5000/apache-wp:latest	
swarm-wrkr32	Running	Running 2 hours ago	
woktuo79cl24	wp-1_apache.2	registry.alcocer.net:5000/apache-wp:latest	
swarm-wrkr21	Running	Running 2 hours ago	

```
root@swarm-mgr21:/mnt/gv1/swarm-data/composes#
```

#### 12.3.2.MariaDB-Haproxy



## 13. Conclusiones, inconvenientes y problemas

---

En este proyecto me he ceñido al uso de software libre y manuales oficiales.

### 13.1. Inconvenientes y problemas

#### 13.1.1. Swarm

##### Volúmenes

Docker swarm es una muy buena solución a la alta disponibilidad en microservicios pero aún está carente de una buena gestión para los datos y volúmenes compartidos entre los nodos del cluster.

Para la gestión de volúmenes compartidos existen multitud de aplicaciones y plugins de terceros, pero su fiabilidad es cuanto menos cuestionable.

Yo probé un plugin de NFS para docker swarm pero al apagar y levantar la máquina con el servicio funcionando tuve pérdida de datos. Lo probé con un servicio BIND replicado.

##### Balanceo de carga

El balanceo de carga de docker swarm es mediante RoundRobin, eso ocasiona muchos problemas cuando se necesitan conexiones persistentes, obligando a desplegar de una forma muy concreta.

#### 13.1.2. Proxmox

Proxmox presume de una GUI de fácil manejo para la gestión de almacenamiento, y ofrece muchas posibilidades a la hora de levantar unidades. Sin embargo, la mitad de esas funciones en la GUI están muy limitadas, y cuando hay que configurarlas a base de comandos en la shell, los comandos son propios de Proxmox para los cuales hay poca documentación oficial.

Tuve montado LVM sobre unidades ISCSI compartidas que dejaron de funcionar para uno de los nodos.

LVM-Thin no tiene la capacidad de ser compartido si se levanta con la GUI.

#### 13.1.3. GlusterFS

GlusterFS, de forma básica, es muy sencillo de configurar, sin embargo, para que las unidades estén montadas al iniciar el sistema sin intervención humana, tuve que recurrir a `systemd.mount`. No encontré esa solución descrita, casi todo lo que encontré fue a base de AutoFS o scripts Bash.

### 13.2. Conclusiones

Docker swarm es una solución muy rápida para desplegar servicios en Alta disponibilidad pero he observado ciertos comportamientos y errores extraños que a veces aparecen y desaparecen sin haber tocado absolutamente nada ni en configuración de ficheros y en el propio servicio Docker.

También he perdido muchísimo tiempo con Proxmox intentando configurar de forma óptima ISCSI cosa que a última hora tuve que quitar por falta de fiabilidad.

Ceph en proxmox también tenía un comportamiento inesperado, y a eso hay que añadir la carencia de velocidad que ofrecía el servicio.