

Centro de Datos y Provisión de Servicios

**CDPSfy**

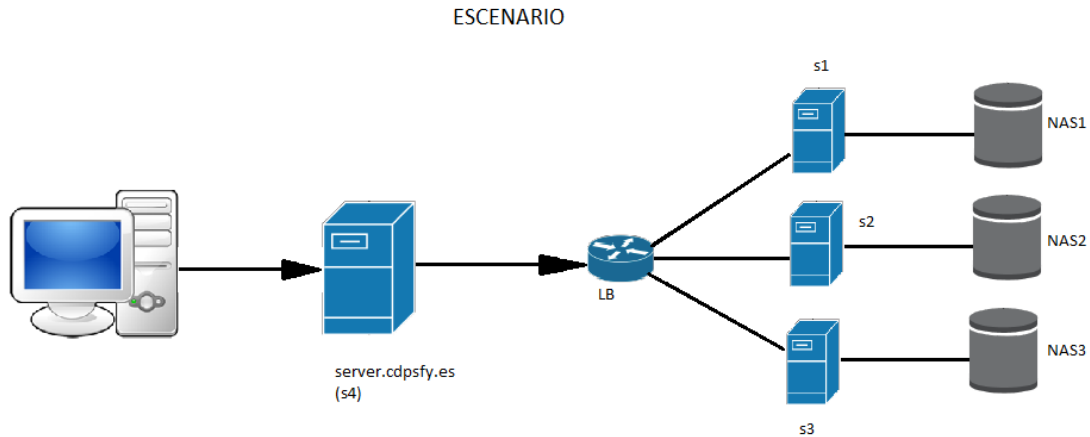
*Equipo 38*

Alberto Sánchez López

Ignacio Pascual Landa

El siguiente documento trata sobre la creación de una red social para el intercambio de música en Internet.

El escenario sobre el que se apoyará será el siguiente:



El servidor **s4**, será el encargado de renderizar la web donde tendremos la vista de todas las canciones subidas por los usuarios. El resto de los servidores **s1, s2 y s3**, tendrán como labor interactuar directamente con los servidores de disco. Estos últimos, **NAS1, NAS2 y NAS3**, serán los que contendrán las canciones subidas a **CDPSfy** y serán accedidos por los servidores web según las diferentes acciones de los usuarios.

Además, el escenario consta de un **balanceador de carga**, utilizado para dividir el tráfico y la carga de trabajo entre los tres servidores web conectados a los servidores de disco.

En el cliente1, **c1** se ha configurado el sistema de monitorización de nagios para obtener información acerca de los estados, usos de memoria y disco de los servidores s1, s2, s3 y s4.

Por último, hay que resaltar que desde el servidor **s4** realiza peticiones POST y GET hacia los servidores de tracks (s1, s2, s3) con el fin de subir y obtener los ficheros. Por tanto tendrá que establecer una conexión con uno de los servidores web, según dictamine el balanceador de carga.

## 1. Crear o Iniciar el escenario

Para crear el escenario nos servimos de la herramienta **vnx**, a partir del archivo proporcionado .xml

```
sudo vnx -f p7.xml -v -create #Crear escenario  
sudo vnx -f p7.xml -v -shutdown #Pararlo conservando los cambios  
sudo vnx -f p7.xml -v -destroy #Pararlo borrando los cambios  
sudo vnx -f p7.xml -v -show-map #Muestra mapa del escenario
```

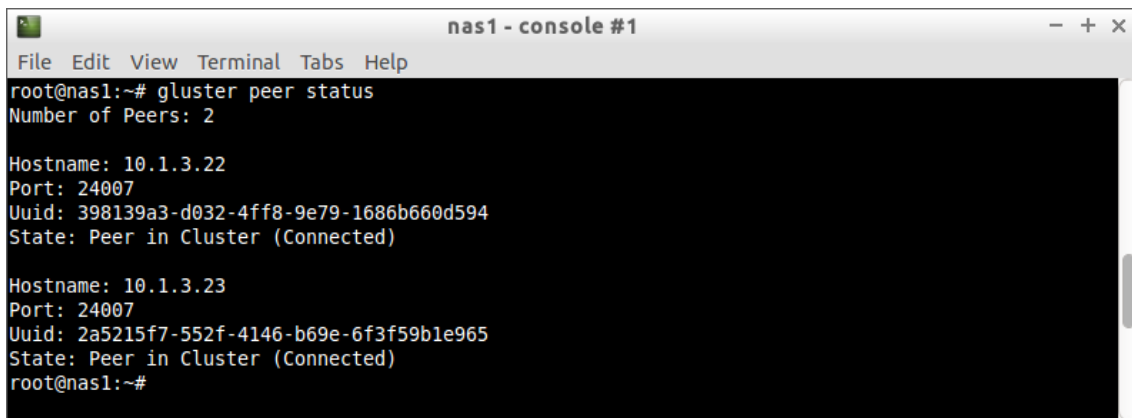
## 2. Configuración Gluster

Comenzamos configurando el gluster desde el NAS1:

```
gluster peer probe 10.1.3.21  
gluster peer probe 10.1.3.22  
gluster peer probe 10.1.3.23
```

*# Consultar el estado de los servidores*

```
gluster peer status
```



```
nas1 - console #1  
File Edit View Terminal Tabs Help  
root@nas1:~# gluster peer status  
Number of Peers: 2  
  
Hostname: 10.1.3.22  
Port: 24007  
Uuid: 398139a3-d032-4ff8-9e79-1686b660d594  
State: Peer in Cluster (Connected)  
  
Hostname: 10.1.3.23  
Port: 24007  
Uuid: 2a5215f7-552f-4146-b69e-6f3f59b1e965  
State: Peer in Cluster (Connected)  
root@nas1:~#
```

### 3. Realizar la réplica en NAS2 y NAS3.

Mediante esta operación conseguimos que al copiar un fichero se replique en los otros

*#Creación del volumen con 3 servers que replican información*

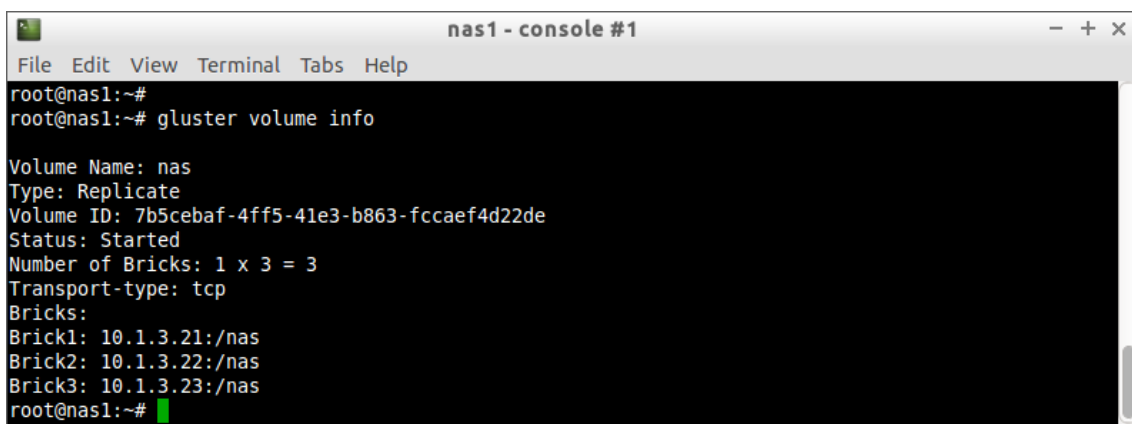
```
gluster volume create nas replica 3 10.1.3.21:/nas
10.1.3.21:/nas 10.1.3.21:/nas force
```

*#Arrancamos el volumen*

```
gluster volume start nas
```

*#Vemos el estado de los volúmenes creados*

```
gluster volume info
```



```
nas1 - console #1
File Edit View Terminal Tabs Help
root@nas1:~#
root@nas1:~# gluster volume info

Volume Name: nas
Type: Replicate
Volume ID: 7b5cebaf-4ff5-41e3-b863-fccae4d22de
Status: Started
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: 10.1.3.21:/nas
Brick2: 10.1.3.22:/nas
Brick3: 10.1.3.23:/nas
root@nas1:~#
```

### 4. Hacemos accesible el sistema de ficheros del Gluster desde S1, S2, S3

*#En s1*

```
mkdir /mnt/nas
mount -t glusterfs 10.1.3.21:/nas /mnt/nas
```

*#En s2*

```
mkdir /mnt/nas
mount -t glusterfs 10.1.3.22:/nas /mnt/nas
```

*#En s3*

```
mkdir /mnt/nas
mount -t glusterfs 10.1.3.23:/nas /mnt/nas
```

*\*hay que hacer service apache2 restart en cada uno para que al configurar el lb funcione (en s4 también).*

## 5. Configuración del Balanceador de Carga.

La configuración del balanceador de carga puede realizarse ejecutando las siguientes sentencias en el terminal del balanceador:

```
service apache2 stop
```

*#Utilizamos round robin (atributo -dr)*

```
xr -dr --verbose --server tcp:0:80 --backend 10.1.2.11:3030
--backend 10.1.2.12:3030 --backend 10.1.2.13:3030 --web-
interface 0:8001
```

*#Los backends corresponden con las ips de los s1, s2 y s3 y el puerto donde ejecutamos el server*

## 6. Definición de direcciones

En el host (nuestra consola) añadimos las siguientes líneas en `/etc/hosts`

```
10.1.2.14 server.cdpsfy.es    #[s4]
```

10.1.1.1 tracks.cdpsfy.es *#[lb]*

```
cdps@cdps-vm: /etc
File Edit Tabs Help
127.0.0.1      localhost
127.0.1.1      cdps-vm
10.1.2.14      server.cdpsfy.es
10.1.1.1       tracks.cdpsfy.es
-- INSERT --           2,10           Top
```

Con esto conseguimos que al acceder a esas URLs, nos redirija a las direcciones IP especificadas.


En `s4,/etc/apache2/sites-available/000-default.conf` añadimos debajo de Document Root:

ServerName server.cdpsfy.es

ServerAlias \*.server.cdpsfy.es

Y en `/etc/hosts` añadimos (para acceder usando esa URL)

#### 10.1.2.1 tracks.cdpsfy.es



```
s4 - console #1
File Edit View Terminal Tabs Help
127.0.0.1 localhost.localdomain localhost
127.0.1.1 s4
10.1.2.1 tracks.cdpsfy.es
1,1 Top
```

## 7. Preparar s1, s2, s3, y s4 para copiar los archivos de node

En cada uno de los servidores:

```
service apache2 stop  
sudo apt-get install nodejs
```

Para preparar los servidores para después ejecutar los servidores de node.

## 8. Copiamos los ficheros de tracks y de server a su respectivo

Desde nuestro terminal ejecutamos los siguientes comandos para copiar los archivos de los servidores de node a los servers s1, s2, s3, y s4:

```
cp /carpeta/FUNCIONAV4.tar.gz /var/lib/lxc/sX/rootfs/tmp
```

*#Desde cada servidor (s1, s2, s3 y s4):*

```
cp /tmp/FUNCTIONALV4.tar.gz /  
tar -zxvf FUNCTIONALV4.tr.gz
```

*#Con esto ya tendríamos copiados en la raíz y descomprimidos los servidores de node*

A continuación nos disponemos a ejecutarlos:

*#En s4*

```
cd sanxlop-server  
npm install request  
npm start
```

*#En s1, s2, s3*

```
cd sanxlop-tracks  
npm start
```

Con esto ya tendríamos los 4 servidores ejecutándose y balanceados por el lb

## 9. Monitorización con Nagios

Para esta última parte, ejecutaremos los siguientes comandos en c1 para la monitorización con Nagios:

```
sudo apt-get update
```

#Por si hubiera problemas con las versiones anteriores.

```
sudo apt-get install apache2
```

```
sudo apt-get install nagios3
```

```
service apache2 restart
```

Configuramos algunos valores de nagios tras la instalación y accedemos al siguiente directorio: */etc/init.d/*

En el fichero *hostgroups\_nagios2.cfg*, tendremos que realizar una serie de cambios en el campo *members*, sustituyendo el valor "localhost" por "\*".

Además, a partir del fichero *localhost\_nagios2.cfg* creamos los ficheros *s1.cfg*, *s2.cfg*, *s3.cfg* y *s4.cfg*, cambiando el valor de los campos *localhost* y *alias* por el nombre del servidor (*s1*, *s2*, *s3* o *s4*) y el campo *address* por el valor de la ip correspondiente de cada servidor.

Una vez modificados y añadidos estos ficheros:

```
service nagios3 restart
```

```
service apache2 restart
```

Incluimos estos ficheros ya configurados en la entrega como muestra de la configuración para Nagios.

Para utilizar Nagios, basta con introducir en el navegador la dirección del cliente C1, donde lo hemos configurado, y el directorio donde lo instalamos (*/nagios3*). Sería: *10.1.1.11/nagios3*.

A continuación, nos piden un login, usuario *nagiosadmin* y contraseña *xxxx*.

Accediendo al apartado *Hosts*, comprobaremos los servidores activos en el momento del acceso.

## 10. Explicación de los servidores de node server y tracks

En la entrega hemos incluido los servidores node (FUNCIONALV3.tar.gz y FUNCIONALV4.tar.gz). El primero de ellos contiene el server y el tracks configurados para que puedan ser desplegados de forma local en los puertos 8080 (server) y 3030 (tracks) y almacenando los ficheros de audio en una carpeta de tracks llamada media.

El segundo contiene también los servidores tracks y server de node con la configuración necesaria para desplegarlo en nuestro escenario.

El denominado **sanxlop-tracks** habrá que desplegarlo en los servidores s1, s2, s3 y será el encargado de responder a los POST y GET recibidos de sanxlop-server. Para ello se han implementado 3 funciones que responden a los GET y los POST:

`addTrack` → POST

Permite peticiones POST (<http://tracks.cdpsfy.es/api/tracks/>) para subir las canciones en el directorio /mnt/nas a través de tracks con un nombre específico para poder recuperarlo posteriormente mediante la siguiente función.

`findTrackByName` → GET

Permite peticiones GET (<http://tracks.cdpsfy.es/api/tracks/name>) para obtener los archivos de audio almacenados en los NAS y poder reproducirlos.

`deleteTrackByName` → POST

Permite peticiones POST (<http://tracks.cdpsfy.es/api/tracks/name>) para eliminar los correspondientes archivos de audio subidos.

El **sanxlop-server** es el proporcionado para la realización de la práctica. En el hemos implementado los métodos create y delete con un *form request* para enviar peticiones POST al sanxlop-tracks y así poder subir y borrar las canciones en los NAS. En el create también almacenamos las urls correspondientes al directorio donde se encuentran los ficheros de audio con su disckname en el models para realizar las peticiones GET y reproducir los archivos. De esta manera los datos no se almacenan de forma persistente y convendría implementar una base de datos MySQL y MongoDB para conseguir persistencia.



## 11. Implementación en OpenStack y AWS

En **OpenStack** utilizaríamos las diferentes APIs disponibles para la computación en la nube. Utilizaríamos *Nova* (OpenStack Compute) para iniciar las máquinas virtuales necesarias para el despliegue de los servidores y gestionar la red virtual para cada instancia. También destacar el uso de *Swift* (Object Storage) como sustituto de las NAS sirviendo para el almacenamiento masivo de objetos a través de un sistema escalable, redundante y tolerante a fallos, usándolo como almacenamiento de ficheros de audio. Usaríamos *Glance* (Image Service) para recuperar las imágenes de las máquinas virtuales, siendo estas utilizadas por Nova.

En **Amazon Web Services** disponemos de diferentes servicios, usaríamos *EC2* para desplegar las máquinas virtuales de los servidores. Además necesitaríamos usar alguno de los servicios disponibles para el almacenamiento de los ficheros de audio como pueden ser *S3* que es escalable. Por último deberíamos hacer uso de algún sistema de bases de datos para conseguir persistencia en los datos de las canciones subidas.