Servicio web de alta disponibilidad con GlusterFS, Galera, OpenStack y Owncloud.

Manuel Jesús Begines González

Proyecto Fin de Ciclo

Índice

Introducción

En este proyecto vamos a construir un sistema de alta disponibilidad para un sistema de almacenamiento remoto de datos como lo es Owncloud (aunque se podría hacer con cualquier CMS, página web o plataforma), en muchos casos la necesidad nos lleva a tener un sistema lo mas resistente posible contra caídas o perdida de datos. Para ellos haremos que la información no este en un solo nodo de la red si no, que esté replicada en tantos servidores como resistente queremos que sea nuestra infraestructura, a más servidores mayor disponibilidad y mejor se repartirán las peticiones que vayan llegando al sistema.

Esta infraestructura debe actuar de manera que para el usuario sea totalmente transparente, para eso usaremos herramientas como sistemas de ficheros distribuidos, cluster de la base de datos y owncloud como frontal web para que el acceso de los usuarios, también se podría instalar el cliente de escritorio que tiene owncloud y la app gratuita para tal fin.

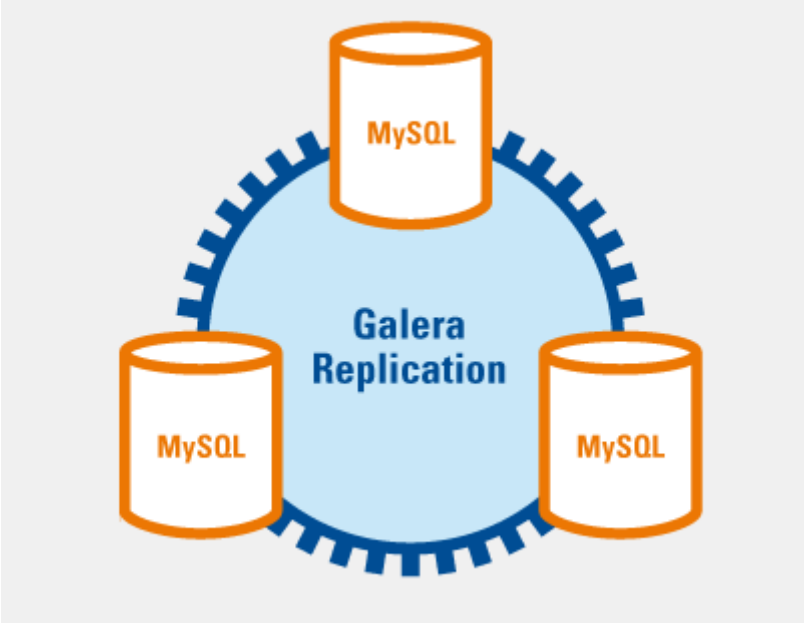
Arquitectura.

A continuación vamos a explicar como será la arquitectura que va a tener nuestro sistema para montar la alta disponibilidad para Owncloud.

En primer lugar decir que todo se hará en un cloud privado con ***OpenStack*** en el cual levantaremos las instancias que queramos usar para montar la infraestructura, en este caso usaremos 3 servidores en los que estará la base de datos replicada que lo haremos usando las posibilidades que nos ofrece ***Galera Clustes***. Los servidores también contarán con un disco añadido de 10GB, el cual usaremos para configurar un sistema de ficheros distribuidos que será ***GlusterFS,*** que que los datos que quieran almacenar los usuarios se distribuyan por en los 3 servidores, luego comfiguramos un balanceador de cargar que será el quien dirija la carga a un servidor o a otro y será el a quién el usuario atacará usando la ***IP Virtual***, el balanceador configuraremos el que openstack nos proporciona, sin necesidad de tener que montar una máquina para tal fin.

A continuación vamos a comentar por separado las distintas tecnologías que hemos elegido:

* Galera Cluster:



Galera cluster es un multicluster que realiza replica sincrona entre los nodos que esta configurado, es una solución de alta disponibilidad que evita la perdida de datos y proporciona escalabilidad para el futuro.

Es un claro compertidor también de software libre de MySQL Cluster, Galera también esta disponible a través de percona. Galera Cluster usa MariaDB que es un fork de MySQL con el motor **innoDB.**

Características:

- Replica síncrona.
- Topología Multimaestro activo-activo.
- Leer y escribir en cualquier nodo del clúster.
- Control automático de la membresía, los nodos fallidos caen del cluster.
- Unión automática de los nodos
- Verdadero nivel de replicación paralela a nivel de fila.

Beneficios:

- Sin lag en esclavo
- No hay transacciones perdidas
- Escalabilidad tanto al leer como al escribir
- Latencias de clientes mas pequeñas

En definitiva Galera es una solución bastante aceptable para el cluster de base de datos que queremos crear en nuestro sistema, y tendríamos cubierto ese punto.

* GlusterFS:

Otra tecnología que vamos a usar son los sistemas de ficheros distribuidos y redundante como es el caso de GlusterFS.

GlusterFS es un multiescalable sistema de archivos NAS que permite agregar varios servidores de archivos sobre ethernet. Se basa en la utilización del espacio de usuario y de esta manera no compromete su rendimineto, se usa en computación en la nube y en almacenamiento de archivos como es el caso para que lo queremos.

Se basa en la interacción de cliente y servidor. Los servidor se implementa como almacenamiento de bloques y exporta un sistema de archivo local como un volumen y el cliente se conecta a los

servidores. Por defecto, los archivos son almacenados enteros, pero también puede configurarse que que se fragmente en múltimples porciones en cada servidor.

Los volúmenes pueden ser montados tanto en el modulo FUSE(que permite a usuarios no privilegiados crear sus propios sistemas de archivos sin necesidad de editar el código del núcleo) o con las librería cliente *libglusterfs.*

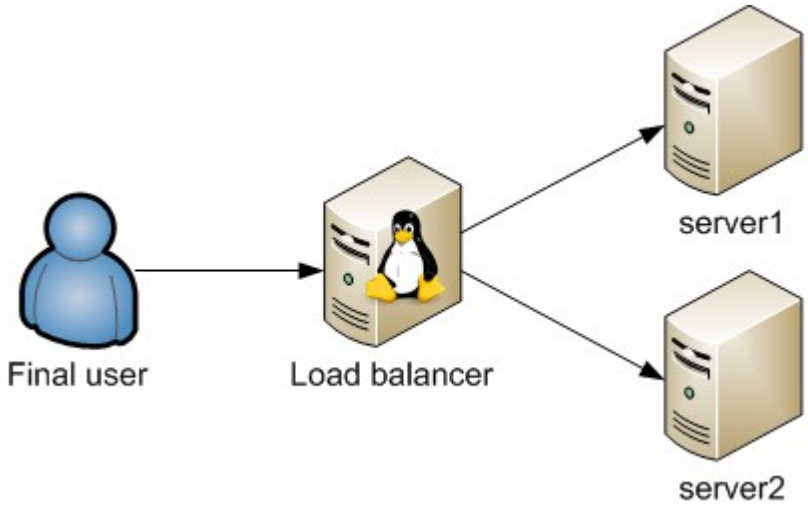
Características:

- Espejado y replicación de archivos.
- Fragmentación de archivos o Data striping.
- Planificación de E/S y almacenamienton en caché de disco.
- Las cuotas de almacenamiento.

Beneficios:

- Volúmenes con tolerancia a fallos.
- Balanceo de carga para la lectura y escritura de archivos.
- Datos replicados.

* Balanceador de carga en OpenStack.



Un balanceador de carga es un dispositivo que actúa como proxy inverso distribuyendo el tráfico de red o de una aplicación a varios servidores. Los balanceadores se utilizan para incrementar la capacidad de procesamiento y confiabilidad, asegurando la disponibilidad monitorizando el estado de las aplicaciones y enviando las peticiones a los servidores que puedan responder.

Hay dos categorías en las que se agrupan los balanceadores de carga que son:

- **Layer4:** actúan sobre los datos de la red y protocolos IP, TCP, FTP y UDP.

-**Layer7:** distribuyen peticiones en la capa de aplicación con protocolos como HTTP o TCP.

Ambos distribuyen las peticiones en base a algoritmos como ***Round robin***, ***Weighted round robin*** (se basa en la ponderación de las peticiones, de manera que el servidor mas potente recibirá mas peticiones que el servidor menos potente), ***Least connections*** (asigna las peticiones que menos conexiones tenga en ese memento), ***Least response time*** (pasa la petición al servidor que menos tiempo tarde en responder).

En OpenStack:

OpenStack nos proporciona una solución mas que factible para el balanceo de carga, nos da la posibilidad de no tener que usar máquinas para tal fin de manera que seria el propio OpenStack el que haría el trabajo.

El balanceador del que dispone OpenStack es ***HAproxy*** implementado de serie, el cual solo hay que configurarlo ya sea desde el panel web como por comando, **Horizon** (interfaz de usuario web para OpenStack) nos dá una posibilidad muy fácil de configurarlo vía web.

*HAproxy: Es un software libre que actúa como balanceador de carga ofreciendo alta disponibilidad para comunicaciones TCP y HTTP.

* Owncloud.

Es una aplicación de software libre de servicio de alojamiento de archivos, que permite el almacenamiento en línea y aplicaciones en línea. Los archivos que guardemos podemos acceder a ellos mediante aplicaciones móvil (gratuitas o de pago), clientes de escritorio desde el panel web que proporciona el propio servidor. Es una herramienta muy parecida a dropbox pero tiene ventajas respecto a él, los datos estan guardados en nuestros propios servidores, además se le puede instalar multitud de aplicaciones como por ejemplo de noticias, emails, reproductor, editor de texto, lector de pdf, juegos, etc.

Se suele confundir owncloud con cloud computing, aunque si es un servicio en la nube no llega a la categoría de este.

Una característica a destacar es el cifrado de datos, por defecto esta desactivado, pero se puede activar fácilmente ya que es una aplicación integrada en el sistema. Este método, a diferencia del cifrado en el lado del cliente, reduce las garantías de seguridad, pero posibilita el uso de Owncloud a través del navegador.

Características:

- Almacenamiento de archivos en una estructura de direcctorios convencionales.
- Reproductor de música.
- Administración de usuarios y grupos.
- El intercambio de contenidos a través de grupos o dirrecciones URL públicas.
- En línea editor de texto con resaltado de sintaxis y plegado de códigos.
- Marcadores.
- Galeria de fotos.
- Visor de PDF (usando pdf.js).

Beneficios:

- Replica de datos en servidores remotos.
- Acceso a la información en cualquier momento siempre que haya acceso a internet.
- Acceso a multiples aplicaciones centralizadas en Owncloud.
- Recuperación de archivos borrados y control de versiones.
- Privacidad.

Despliegue de la infraestructura.

A continuación vamos a comenzar con la instalación de en los servidores de los distintos servicios. Lo primero que debemos de hacer es instalar los paquetes necesarios que necesitaremos en toda la configuración.

```
#apt-get install python-software-properties
#apt-key adv --recv-keys --keyserver keyserver.ubuntu.com
0xcbcb082a1bb943db
```

En el siguiente paquete que debemos de instalar debemos de asegurarnos de que pone “trusty” en vez de “precise”:

```
#add-apt-repository 'deb
http://mirror3.layerjet.com/mariadb/repo/5.5/ubuntu trusty main'
```

Una vez instalado esto hacemos un “update” e instalamos los paquetes necesarios para apache y php:

```
#apt-get update && apt-get install -y apache2 php5 php5-common
libapache2-mod-php5 php5-xmlrpc php5-gd php5-mysql
```

*** <u>Instalación y configuración de Galera Cluster:</u>**

Ahora con los paquetes necesarios instalados y actulizados debemos de instalar Galera que se hará con el siguiente comando:

```
#//DEBIAN_FRONTEND=noninteractive apt-get install -y rsync galera
mariadb-galera-server
```

Si nos da errores al instalar es porque no tenemos una firma “pgp”y debemos de forzar para que se instale y le añadimos las versiones de galera para que no nos vuelvan a salir, como es en el siguiente comando:

```
#DEBIAN_FRONTEND=noninteractive apt-get install -y rsync galera-3
mariadb-galera-server-5.5 --force-yes
```

Una vez instalado Galera debemos de editar el archivo en todos los servidores que vaya a tener galera y debemos de poner exactamente lo mismo en este caso en los 3 servidores:

```
#nano /etc/mysql/conf.d/galera.cnf
```

Luego dentro del archivo debemos de poner las direcciones IP de los servidores en el argumente “wsrep_cluster_address”

```
[mysqld]
#mysql settings
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
query_cache_size=0
query_cache_type=0
bind-address=0.0.0.0
#galera settings
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="avengers"
wsrep_cluster_address="gcomm://10.0.0.96,10.0.0.97,10.0.0.98"
wsrep_sst_method=rsync
```

Luego paramos el servicio mysql en todas las maquinas:

```
#service mysql stop
```

NODO1

Luego nos vamos al nodo 1 al que hemos llamado “Mark-XVI” en el cual vamos a inicializar el cluster con la opción “--wsrep-new-cluster” de la siguiente manera:

```
#service mysql start --wsrep-new-cluster
```

Ahora comprobaremos que tenemos un nodo en el cluster:

```
#mysql -u root -e 'SELECT VARIABLE_VALUE as "cluster size" FROM
INFORMATION_SCHEMA.GLOBAL_STATUS WHERE
VARIABLE_NAME="wsrep_cluster_size"'
```

Y nos aparecerá lo siguiente por pantalla:

```
+--------------+
| cluster size |
+--------------+
| 1            |
+--------------+
```

NODO 2

Ahora nos vamos al segudo nodo al cual hemos llamado “Mark-XVII” e iniciamos el servicio MySQL:

```
#service mysql start
```

Nos aparecerá un error, pero debemos de ignorarlo al menos por el momento, no afecta al funcionamiento.

Luego volvemos a ejecutart el comando:

```
#mysql -u root -e 'SELECT VARIABLE_VALUE as "cluster size" FROM
INFORMATION_SCHEMA.GLOBAL_STATUS WHERE
VARIABLE_NAME="wsrep_cluster_size"'
```

y nos deberia de salir 2 nodos en vez de uno.

```
+--------------+
| cluster size |
+--------------+
| 2            |
+--------------+
```

NODO 3

Ahora volvemos a hacer el mismo procedimiento en el nodo 3 en el cual nos volverá aparecer el mismo error que antes, pero volveremos a ignorarlo.

```
# service mysql start
 * Starting MariaDB database server mysqld
[ OK ]
 * Checking for corrupt, not cleanly closed and upgrade needing
tables.
# ERROR 1045 (28000): Access denied for user 'debian-sys-
maint'@'localhost' (using password: YES)
```

Luego volvemos a realizar el mismo comando que hemos ejecutado anteriormente para ver el numero de nodos del cluster.

```
# mysql -u root -e 'SELECT VARIABLE_VALUE as "cluster size" FROM
INFORMATION_SCHEMA.GLOBAL_STATUS WHERE
VARIABLE_NAME="wsrep_cluster_size"'
+--------------+
| cluster size |
+--------------+
| 3            |
+--------------+
```

Una vez hemos hecho esto para probar que el cluster esta funcionado perfectamente lo que debemos de hacer es crear una bbdd en un nodo y ver que se replica en los demás, para eso ejecuatamos el comando:

```
#mysql -u root
```

Y una vez ahi ejecutamos el comando:

```
> create database proyecto;
```

Luego nos vamos a los demás nodos y entramos en mysql como hemos visto antes y luego ejecutamos el siguiente comando con el que debería salir la bbdd que hemos creado:

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| proyecto           |
+--------------------+
4 rows in set (0.01 sec)

MariaDB [(none)]>
```

Para solucionar los errores que nos dieron en el 2º y 3º nodo lo que debemos de hacer es copiar el contenido del fichero "/etc/mysql/debian.cnf" del nodo 1 en los demas nodos:

```
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = 2LQDVllI2gNaVID4
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = 2LQDVllI2gNaVID4
socket   = /var/run/mysqld/mysqld.sock
basedir  = /usr
```

En el caso de que todos los servidores se hayan caido hay que arrancar el cluster con el siguiente comando en cualquiera de los nodos

```
#service mysql start --wsrep-new-cluster
```

*** <u>Implementación y configuración de GlusterFS</u>**

En primer lugar lo que vamos a hacer es instalar el paquete necesario con el comando:

```
#apt-get install -y glusterfs-server
```

una vez hecho esto creamos la carpeta "gluster" o el nombre que prefiramos en la raíz "/":

```
#mkdir /gluster
```

A continuación nos vamos al fichero "/etc/glusterfs/glusterfsd.vol" e introducimos las siguientes líneas en los 3 nodos exactamente iguales:

```
volume posix
     type storage/posix
     option directory /gluster
end-volume

volume locks
     type features/posix-locks
     subvolumes posix
end-volume

volume brick
     type performance/io-threads
     option thread-count 8
     subvolumes locks
end-volume

volume server
     type protocol/server
     option transport-type tcp
     subvolumes brick
     option auth.addr.brick.allow *
end-volumevolume posix
     type storage/posix
     option directory /gluster
end-volume

volume locks
     type features/posix-locks
     subvolumes posix
end-volume

volume brick
     type performance/io-threads
     option thread-count 8
     subvolumes locks
end-volume

volume server
     type protocol/server
     option transport-type tcp
     subvolumes brick
     option auth.addr.brick.allow *
end-volume
```

Luego reiniciamos el servicio de gluster:

```
#service glusterfs-server restart
```

Un vez hecho esto lo que debemos de hacer es editar el fichero "/etc/glusterfs/glusterfs.vol" y añadimos las siguientes líneas donde especificamos las IP's de las máquinas:

```
volume node1
        type protocol/client
        option transport-type tcp
        option remote-host 10.0.0.96
        option remote-subvolume brick
end-volume

volume node2
        type protocol/client
        option transport-type tcp
        option remote-host 10.0.0.97
        option remote-subvolume brick
end-volume

volume node3
        type protocol/client
        option transport-type tcp
        option remote-host 10.0.0.98
        option remote-subvolume brick
end-volume

volume replicated_storage
        type cluster/replicate
        subvolumes node1 node2 node3
end-volume
```

Una vez hecho todo esto lo que debemos de hacer es probar que funciona bien, para eso lo que debemos de es desde el nodo 1 ejecutar los siguientes comandos con las direcciones IP's de los otros dos servidores:

```
#gluster peer probe 10.0.0.97
#gluster peer probe 10.0.0.98
```

Comprobado que funciona lo siguiente que hacemos es crear un volumen llamado por ejemplo “owncloud” replicado en las 3 máquinas:

```
#gluster volume create owncloud replica 3 10.0.0.96:/mnt/gluster
10.0.0.97:/mnt/
#gluster 10.0.0.98:/mnt/gluster
```

Luego arrancamos el volumen y lo montamos en las 3 maquinas de la misma manera:

```
#gluster volume create owncloud replica 3 10.0.0.96:/mnt/gluster
10.0.0.97:/mnt/gluster
#10.0.0.98:/mnt/gluster force
#gluster volume start owncloud
```

Una vez hecho esto para asegurarnos de que cada vez que arranquemos las maquinas este montado el volumen editamos el fichero “/etc/rc.local” y de lojamos de la siguiente manera el las 3 máquinas.

```
mount localhost:owncloud /var/www/html/ -t glusterfs &&
exit 0
```

Nos aseguramso de que el directorio esté vacio antes de montarlo, porque de otro modo no nos dejará

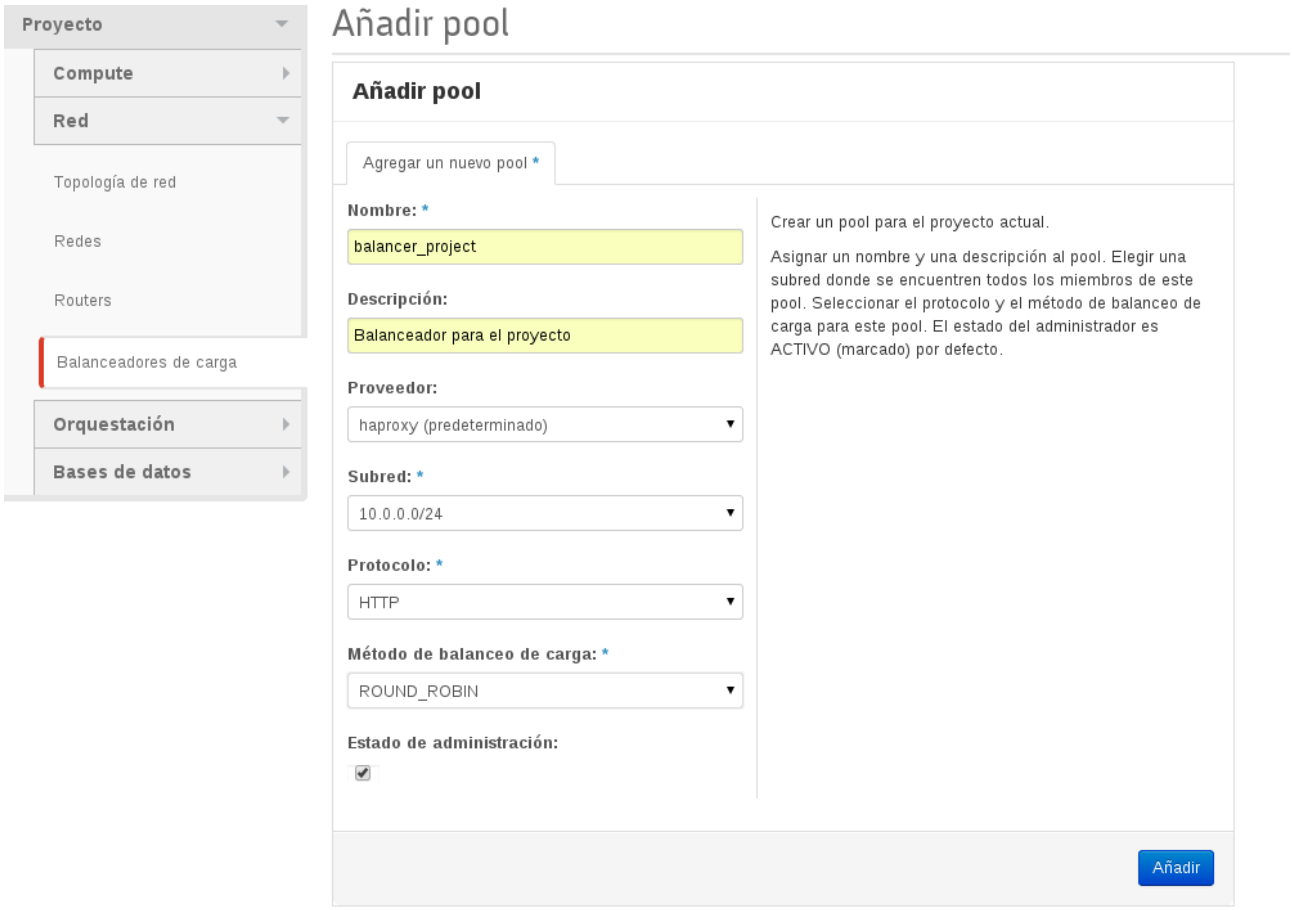
También comentar que el disco secundario esta automaticamente montado, ya que esta configurado en el “/etc/fstab” de la siguiente forma:

```
/dev/vdb        /mnt    auto
defaults,nobootwait,comment=cloudconfig 0       2
```

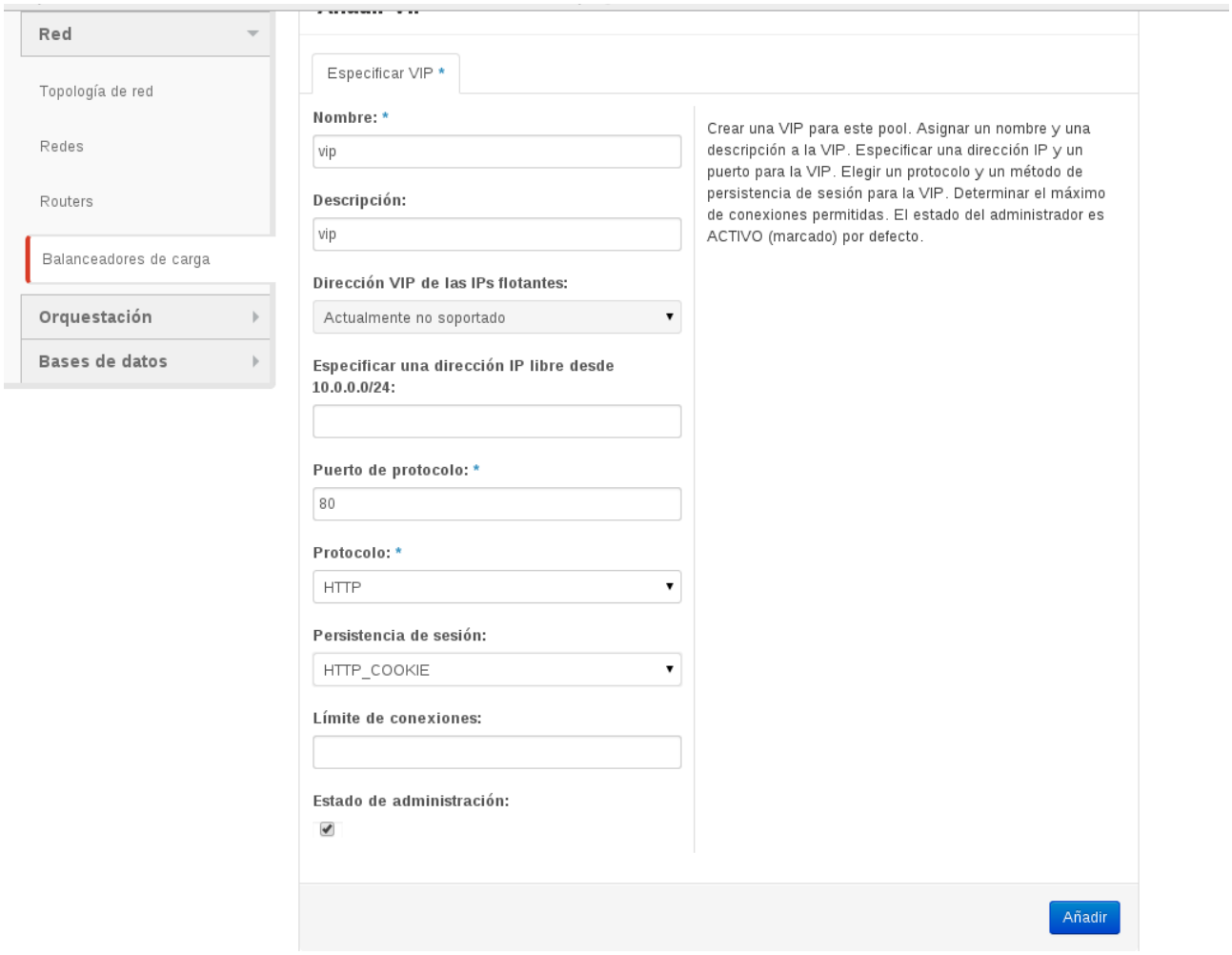
*** <u>Configuración del balanceador de carga en OpenStack</u>**

Ahora lo que debemos de hacer es configurar el balanceador de carga para que los clientes sean repartidos entre los 3 nodos.

En primer lugar lo que debemos de hacer es irnos en Horizon a “Red” y una vez ahí a “Balanceadores de carga”. Una vez ahí pinchamos en añadir “Pool”, luego nos aparacerá un formulario que deberemos de rellenar con el nombre de pool, descripción, proveedor (que será el tipo de software para el balanceo de carga que por defecto esta HAproxy), subred, protocolo y metodo de balanceo de carga que podemos coger entre “round robin”, “least conections” (el que tenga menos conexiones o “source IP” que sea dependiendo de la IP.

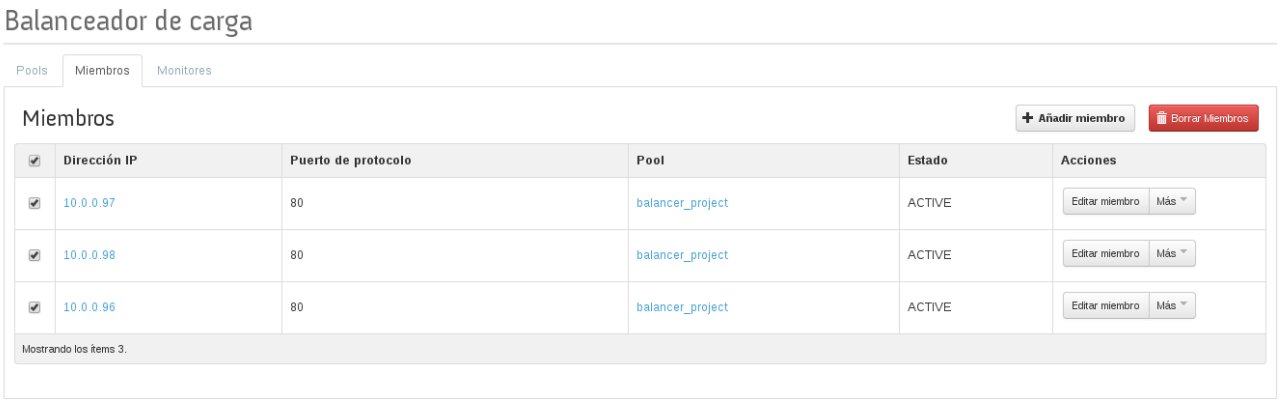


Una vez creado le damos a añadir “VIP” en los ajustes del nuevo pool y volvemos a rellenar el formulario en el cual elegimos como será el tipo de persistencia de las sesiones.



Luego en la pestaña monitores podemos añadir un monitor el cual será el encargado de comprobar que los servidores están levantados mediante ping, http o tcp.

Una vez hecho esto nos vamos a la pestaña miembros donde elegiremos las máquinas que estarán integradas en el balanceador y a las cuales repartirá las peticiones.



Por último nos vamos a “Acceso y seguridad” y luego a la pestaña de “IP flotantes” donde asignaremos una IP flotante libre a balanceador que hemos creado reconociendolo por la IP interna que se le asigno cuando se creo el VIP.

Y ya tendríamos nuestro balanceador funcionando en OpenStack sin necesidad de tener que montar máquinas para tal fín.

*** Instalación de Owncloud**

Ahora procederemos con la instalación de owncloud. La instalación es bastante sencilla y muy parecida a la de cualquier CMS.

En primer lugar lo que debemos de hacer es descargarnos owncloud que lo podemos hacer desde la página oficial:

https://download.owncloud.org/community/owncloud-8.0.4.zip

Una vez descargado lo descomprimimos con el comando:

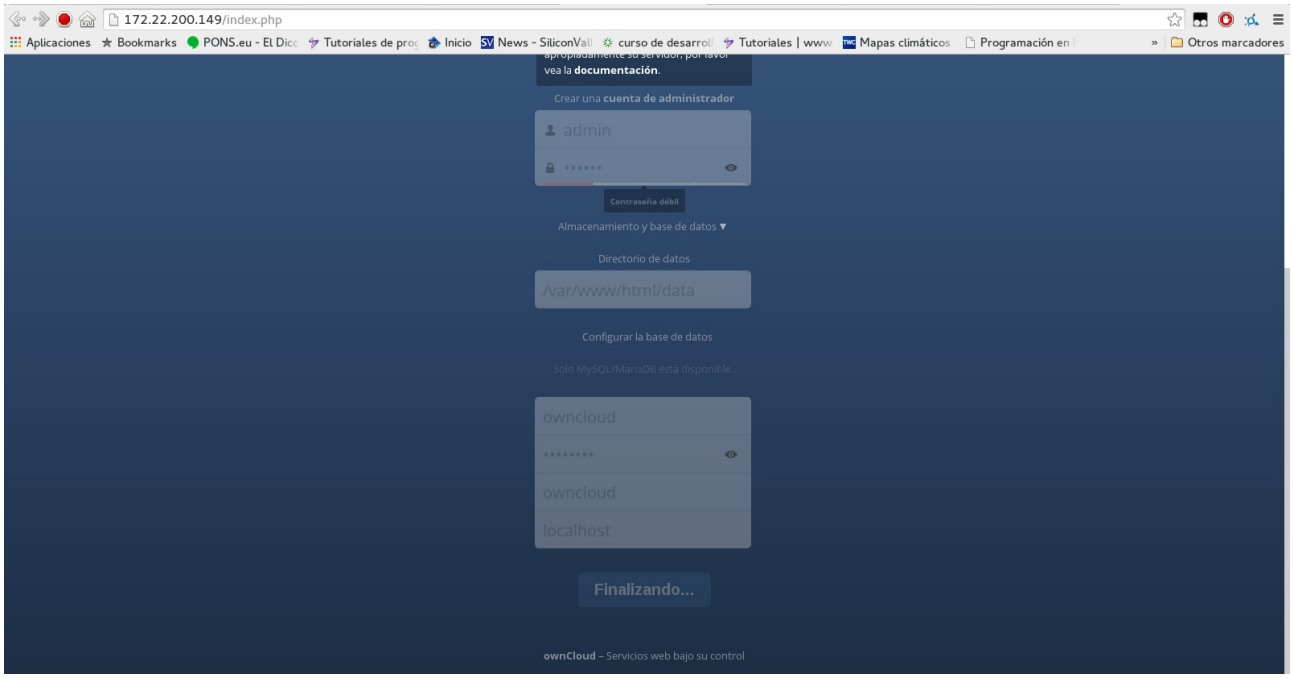
```
#unzip -r owncloud-8.0.4.zip
```

Cuando se haya descomprimido lo siguiente que debemos de hacer es crearle una base de datos en galera de la forma habitual y aplicarle permisos de "www-data":

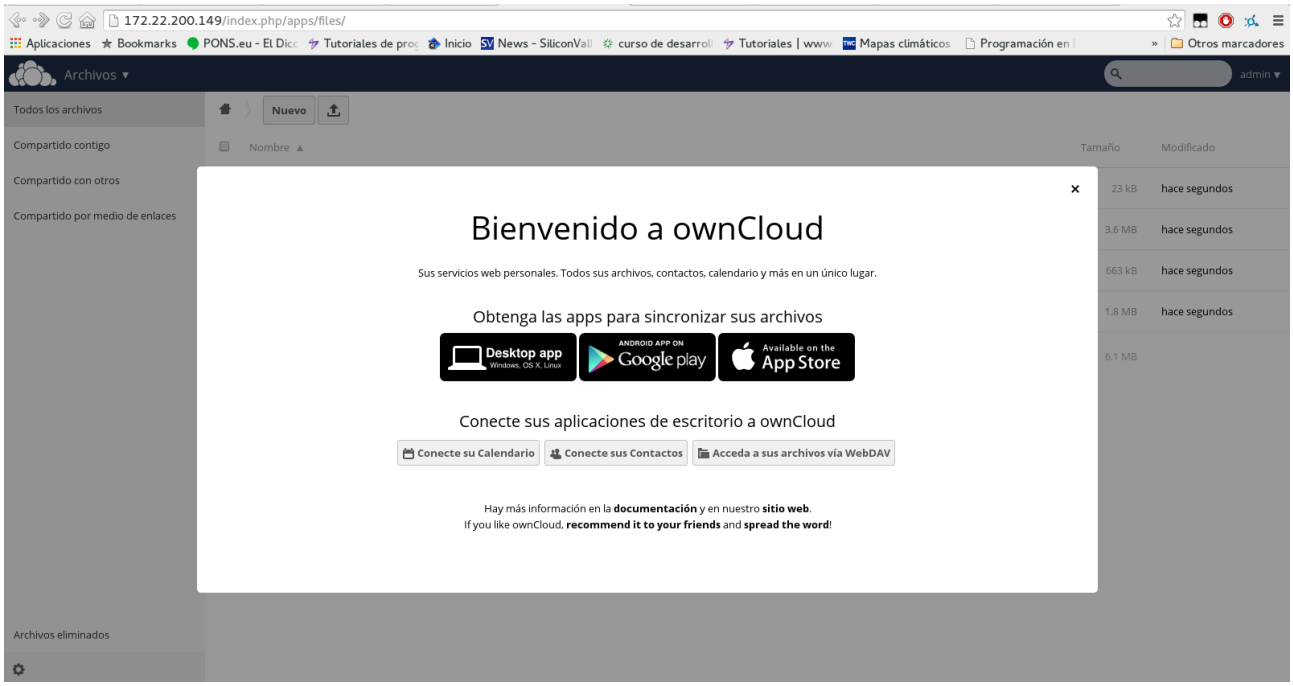
`>create database owncloud` (también podemos crearla usando phpmyadmin)

```
#chown -R www-data:www-data /var/www/html
```

Luego desde el navegador accedemos a la IP virtual que hemos configurado con OpenStack y nos aparecerá el panel de instalación, realmente accedemos a uno de los nodos y es en uno de los nodos donde hacemos la instalación pero al tenerlo todo replicado se hará también en el resto de nodos.



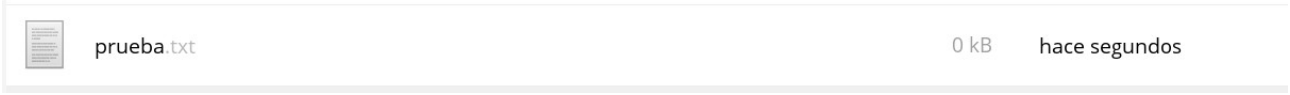
Una vez que se haya instalado nos aparecerá la primera pantalla de bienvenida donde nos sugiere descargarnos la APP para android ios y el cliente de escritorio. Cuando cerramos eso vemos la pagina de owncloud donde se encontraran nuestros archivos.



Pruebas del sistema.

* Comprobar funcionamiento cuando se caen los nodos.

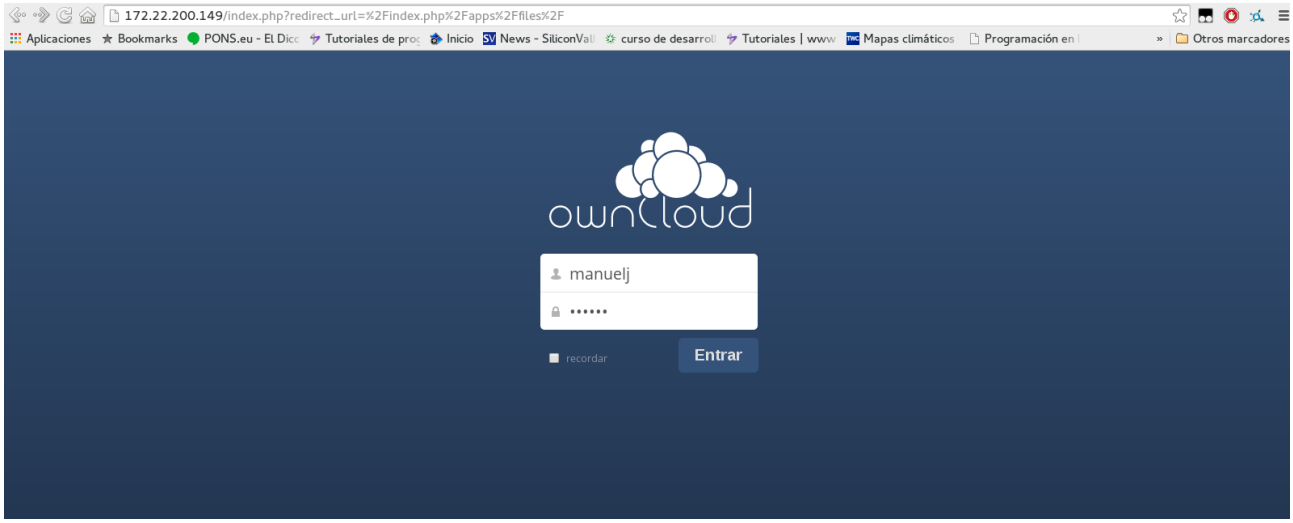
A continuación vamos a realizar una serie de pruebas sobre el funcionamiento del sistema, en primer lugar lo que vamos a hacer es apagar 2 de los 3 servidores y solo estaremos conectados a uno, en ese crearemos un archivo el cual cuando arranquemos los otros servidores se replicará.



	Nombre de la instancia	Nombre de la imagen	Dirección IP	Tamaño	Par de claves	Estado	Zona de disponibilidad	Tarea	Estado de energía	Tiempo de encendido	Acciones
☐	Mark-XVIII	Ubuntu 14.04 LTS	10.0.0.98 172.22.200.53	m1.small.windows | 1GB RAM | 1 VCPU | 20,0GB Disco	clave_manu	Shutoff	nova	None	Shutdown	1 semana, 5 días	Arrancar Instancia | Más
☐	Mark-XVII	Ubuntu 14.04 LTS	10.0.0.97 172.22.200.145	m1.small.windows | 1GB RAM | 1 VCPU | 20,0GB Disco	clave_manu	Shutoff	nova	None	Shutdown	1 semana, 5 días	Arrancar Instancia | Más
☐	Mark-XVI	Ubuntu 14.04 LTS	10.0.0.96 172.22.200.193	m1.small.windows | 1GB RAM | 1 VCPU | 20,0GB Disco	clave_manu	Active	nova	None	Running	1 semana, 5 días	Crear instantánea | Más

Ahora arrancamos las maquinas que hemos apagado anteriormente y apagamos la que hemos dejado encendida y actualizamos la página en el navegador.

	Nombre de la instancia	Nombre de la imagen	Dirección IP	Tamaño	Par de claves	Estado	Zona de disponibilidad	Tarea	Estado de energía	Tiempo de encendido	Acciones
☐	Mark-XVIII	Ubuntu 14.04 LTS	10.0.0.98 172.22.200.53	m1.small.windows | 1GB RAM | 1 VCPU | 20,0GB Disco	clave_manu	Active	nova	None	Running	1 semana, 5 días	Crear instantánea | Más
☐	Mark-XVII	Ubuntu 14.04 LTS	10.0.0.97 172.22.200.145	m1.small.windows | 1GB RAM | 1 VCPU | 20,0GB Disco	clave_manu	Active	nova	None	Running	1 semana, 5 días	Crear instantánea | Más
☐	Mark-XVI	Ubuntu 14.04 LTS	10.0.0.96 172.22.200.193	m1.small.windows | 1GB RAM | 1 VCPU | 20,0GB Disco	clave_manu	Shutoff	nova	None	Shutdown	1 semana, 5 días	Arrancar Instancia | Más

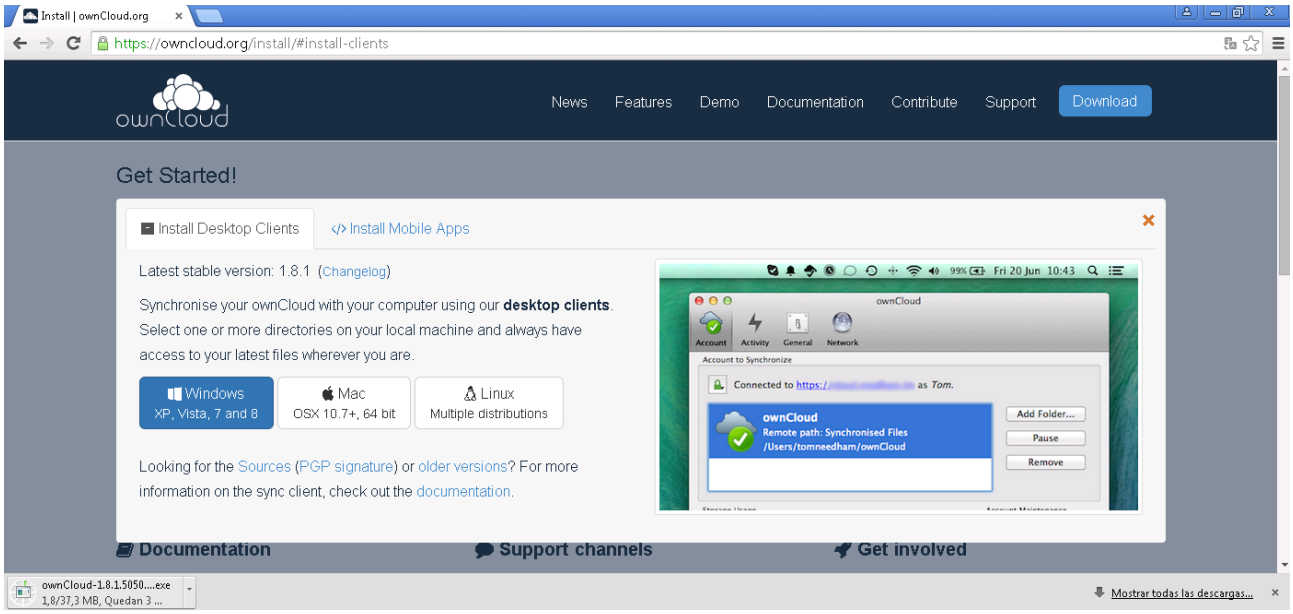


Como vemos debemos de autenticarnos de nuevo ya que ahora el balanceador nos redirige a otro nodo. Una vez logueado vemos que le archivo prueba esta creado y se ha sincronizado sabiendo que el nodo en el que lo creamos esta apagado.

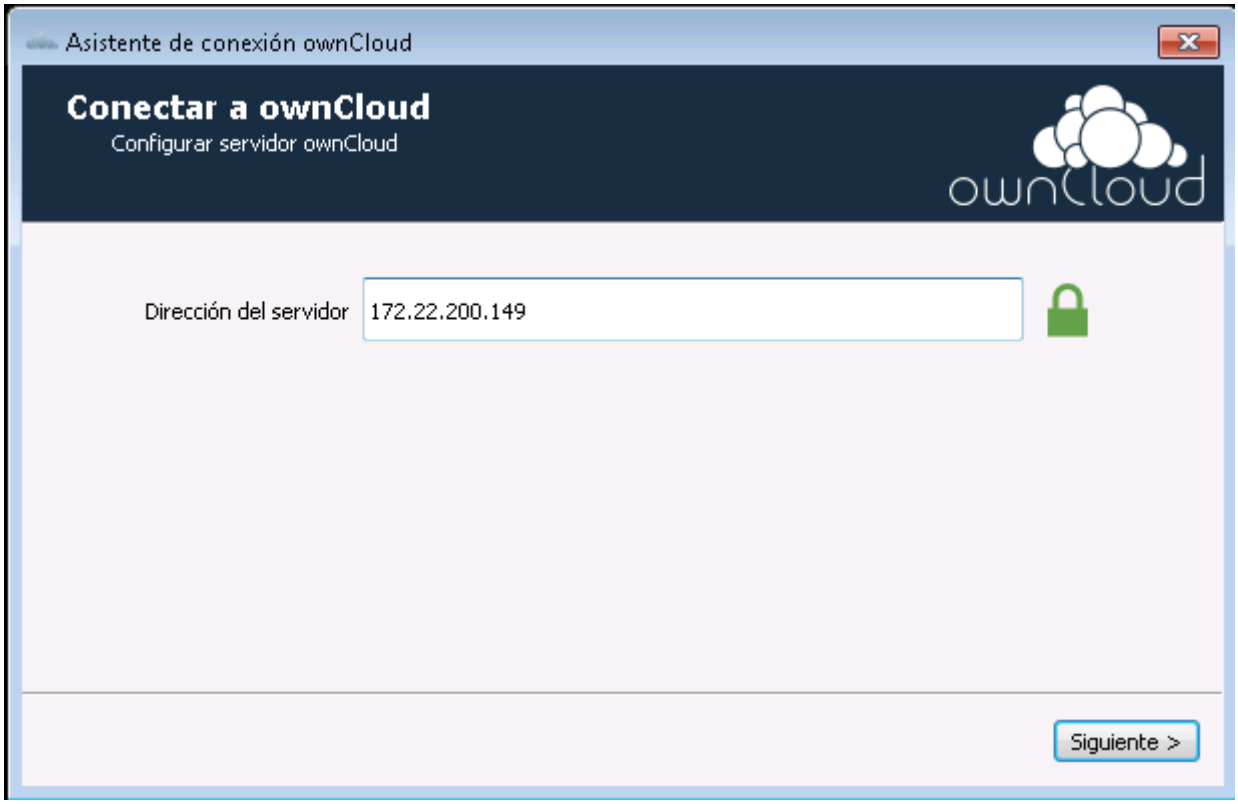
*** <u>Instalación de un cliente de escritorio</u>**

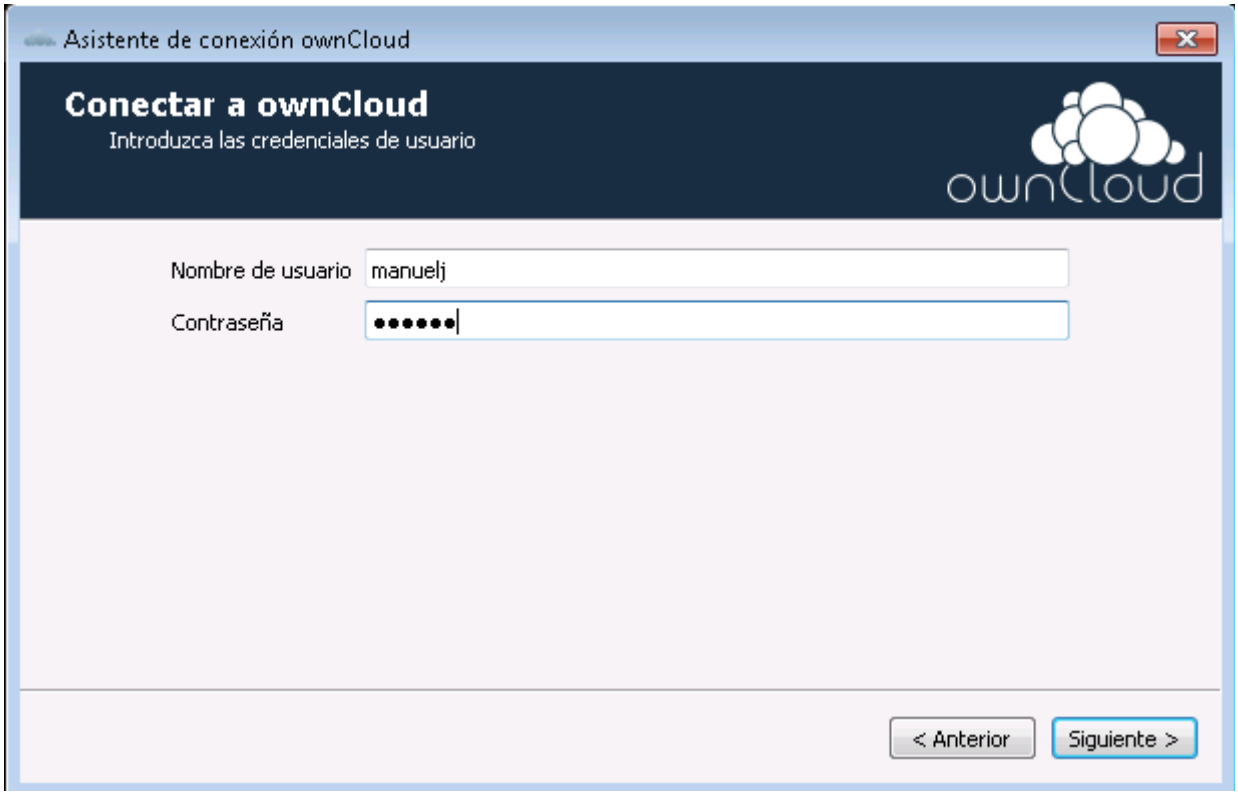
Lo que vamos a hacer ahora es lanzar una maquina en la cual instalaremos un cliente de escritorio de owncloud, nos loguearemos con una de las cuentas que tengamos y empezará a sincronizarnos.

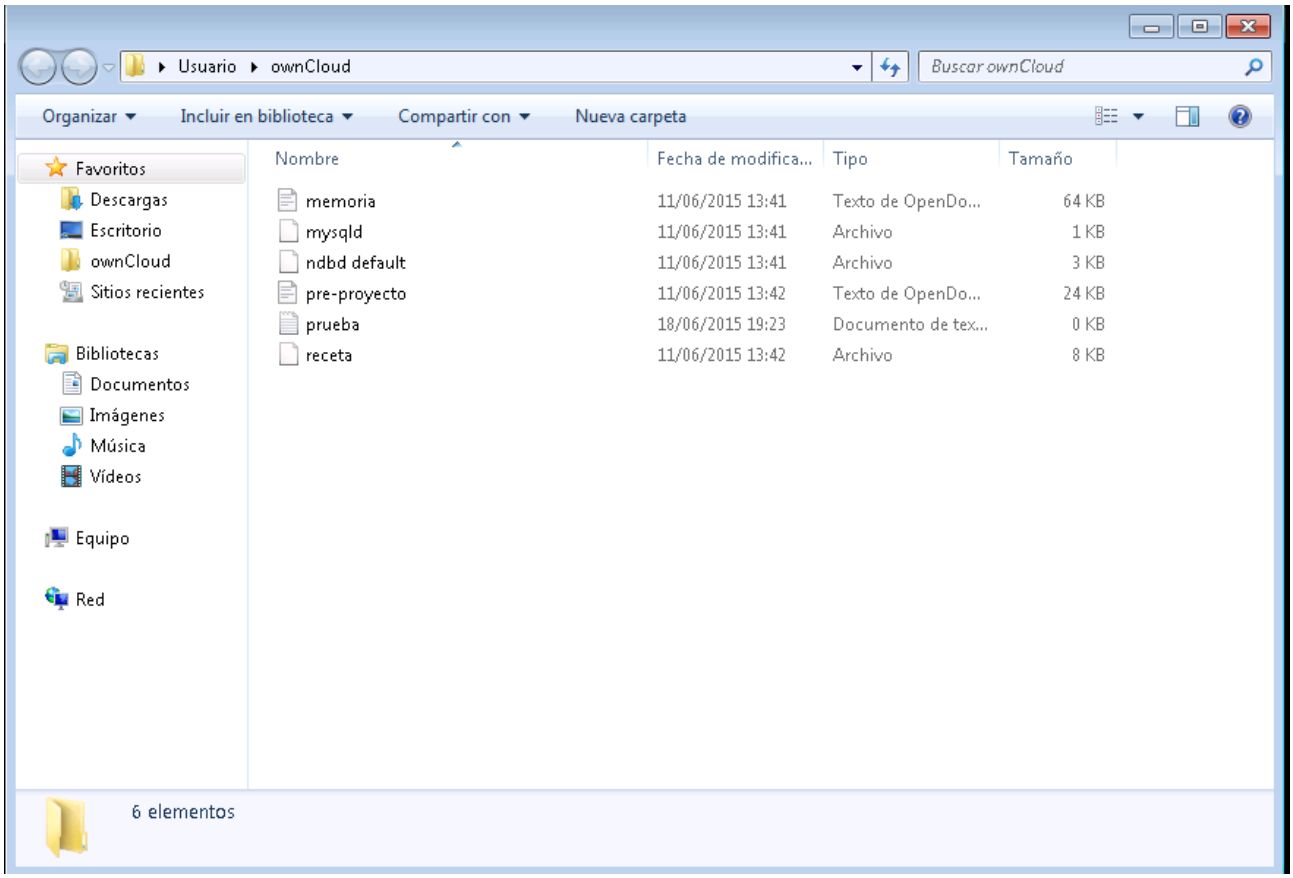
En primer lugar lo que vamos a hacer es lanzar una maquina con Windows 7 por ejemplo y nos descargaremos el cliente de owncloud desde su página oficial.



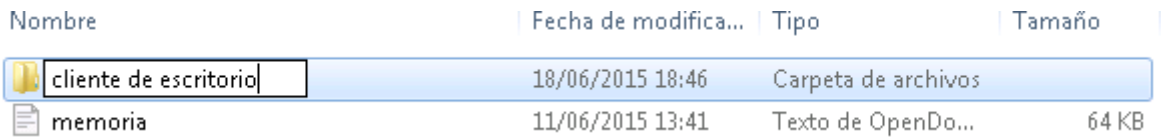
Luego la instalamos y una vez instalada arrancamos el programa y nos pedirá la dirección del servidor y el usuario y contraseña.

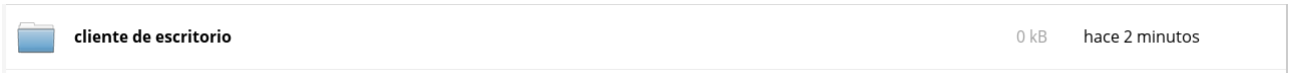

Asistente de conexión ownCloud
Conectar a ownCloud
Configurar servidor ownCloud
ownCloud
Dirección del servidor
172.22.200.149
Siguiente >


Asistente de conexión ownCloud
Conectar a ownCloud
Introduzca las credenciales de usuario
ownCloud
Nombre de usuario
manuelj
Contraseña
< Anterior
Siguiente >



Ahora si creamos un archivo en la carpeta del cliente, se sincronizará y nos aparecerá en cualquier lugar que este autenticado con nuestro usuario y contraseña.





Cuando miramos el log de las maquinas “tailf /var/log/apache2/access.log” vemos que el cliente de escritorio esta conectado a Mark-XVIII (nodo 3).

```
10.0.0.5 - manuelj [18/Jun/2015:20:27:03 +0000] "MKCOL /remote.php/webdav/Nueva
%20carpeta HTTP/1.1" 201 629 "-" "Mozilla/5.0 (Windows) mirall/1.8.1"
```

Probaremos a apagar la maquina y ver como se conecta a otro nodo creando un archivo nuevo:

```
10.0.0.5 - manuelj [18/Jun/2015:20:28:58 +0000] "PROPFIND /remote.php/webdav/
HTTP/1.1" 207 1179 "-" "Mozilla/5.0 (Windows) mirall/1.8.1"
10.0.0.5 - manuelj [18/Jun/2015:20:29:07 +0000] "PROPFIND /remote.php/webdav/
HTTP/1.1" 207 5363 "-" "Mozilla/5.0 (Windows) mirall/1.8.1"
10.0.0.5 - manuelj [18/Jun/2015:20:29:15 +0000] "PROPFIND /remote.php/webdav/
HTTP/1.1" 207 5363 "-" "Mozilla/5.0 (Windows) mirall/1.8.1"
10.0.0.5 - manuelj [18/Jun/2015:20:29:20 +0000] "PUT
/remote.php/webdav/ttht/edede.txt HTTP/1.1" 201 674 "-" "Mozilla/5.0 (Windows)
mirall/1.8.1"
10.0.0.5 - manuelj [18/Jun/2015:20:29:30 +0000] "PROPFIND /remote.php/webdav/
HTTP/1.1" 207 1108 "-" "Mozilla/5.0 (Windows) mirall/1.8.1"
10.0.0.5 - manuelj [18/Jun/2015:20:29:33 +0000] "PROPFIND /remote.php/webdav/
HTTP/1.1" 207 1147 "-" "Mozilla/5.0 (Windows) mirall/1.8.1"
```

*** <u>Monitorizar las conexiones</u>**

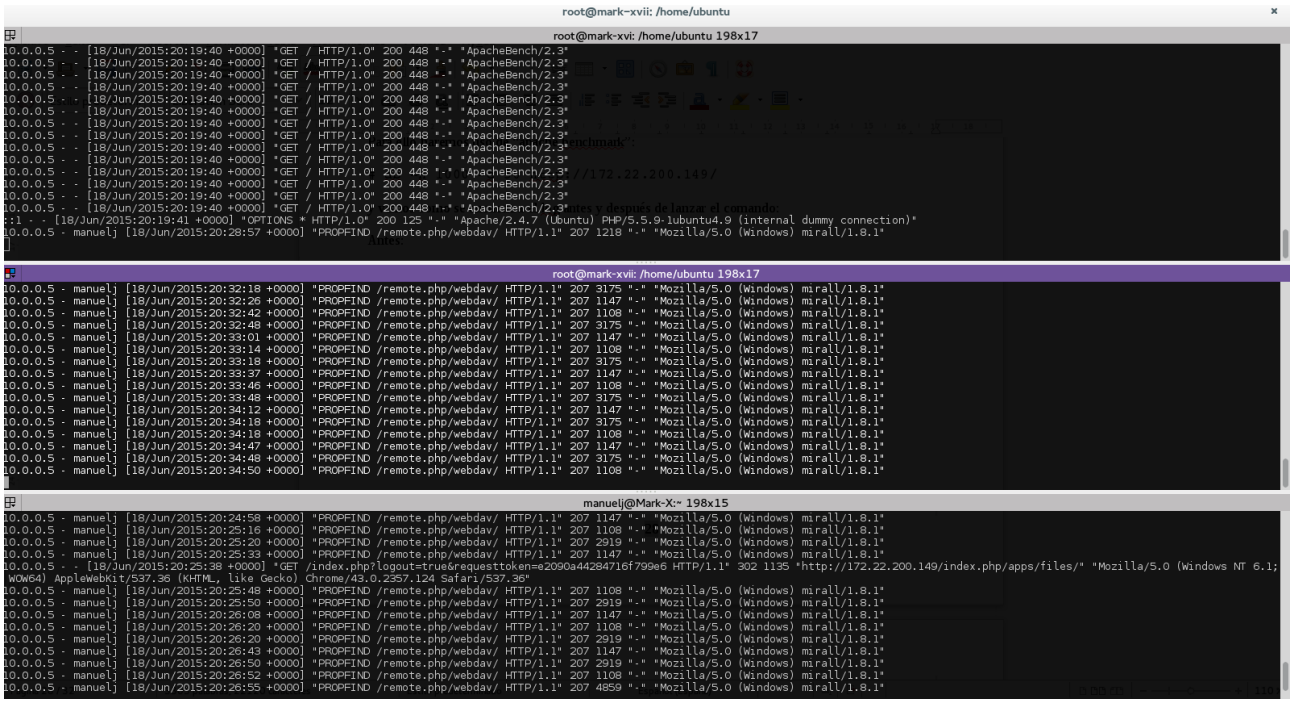
Ahora lo que vamos a intentar comprobar como el balanceador reparte las conexiones a un nodo u a otro.

Para ello haremos uso de “apache benchmark”:

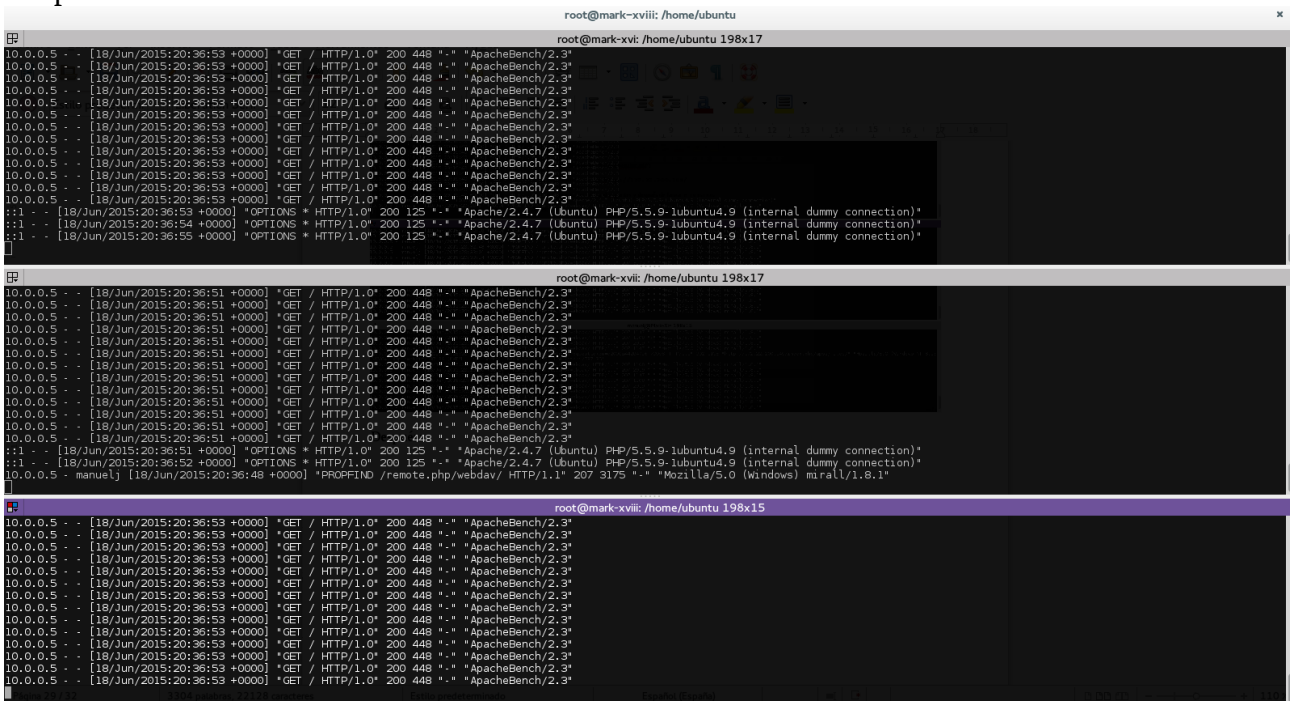
```
# ab -n 1000 -c 50 http://172.22.200.149/
```

Y vemos como se encuentra el log antes y después de lanzar el comando:

Antes:



Después:



Vemos que se reparte las peticiones en los 3 nodos de la red.

Conclusión.

Esta infraestructura nos soluciona el problema de que el sistema dependa de un solo nodo, dando una alta disponibilidad y garantizando la integridad de los datos además de estar totalmente distribuidos. Con ello los usuarios de forma transparente tendrán garantizando su almacenamiento en la aplicación Owncloud, además de tener aplicaciones para usuarios como ya hemos mencionado anteriormente.

También podemos decir que podemos usar esta infraestructura para cualquier servicio como por ejemplo paginas webs y siempre estar protegidos contra perdidas y sobrecargas del sistemas.

Una mejora interesante sería la creación de un script que detecte cuando el número de peticiones es muy alto y agregue automáticamente un nodo o los que sea necesario al sistema y lo quitara cuando ya no lo fuera.

Webgrafía

http://www.severalnines.com/blog/high-availability-file-sync-and-share-deploying-owncloud-galera-cluster-mysql-and-glusterfs/

http://docs.openstack.org/high-availability-guide/content/ha-aa-db-mysql-galera.html/

http://www.fromdual.com/ugly-way-to-install-mysql-galera-cluster-5.6-on-ubuntu-14.04

http://www.catharinegeek.com/set-up-mysql-cluster-on-ubuntu-14041/

http://www.titaniumsystem.es/?p=329

http://www.severalnines.com/blog/scaling-wordpress-and-mysql-multiple-servers-performance